DIMACS Technical Report 2005-41
December 2005

Inferring (Biological) Signal Transduction Networks via Transitive Reductions of Directed Graphs

by

Reka Albert[1]
Department of Physics
Pennsylvania State University
University Park, PA 16802
`ralbert@phys.psu.edu`

Bhaskar DasGupta[2]
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
`dasgupta@cs.uic.edu`

Riccardo Dondi
Dipartimento di Informatica
Sistemistica e Comunicazione
Università di Milano-Bicocca
Milano, Italy 20126
`riccardo.dondi@unimib.it`

Eduardo Sontag[3]
Department of Mathematics
Rutgers University
New Brunswick, NJ 08903
`sontag@math.rutgers.edu`

[1]Partly supported by a Sloan Research Fellowship in Science and Technology.
[2]Partly supported by NSF grants CCR-0296041, CCR-0206795, CCR-0208749 and IIS-0346973.
[3]Partly supported by NSF grants EIA 0205116 and DMS-0504557.

ABSTRACT

In this paper we consider the p-ary transitive reduction (TR_p) problem where $p > 0$ is an integer; for $p = 2$ this problem arises in inferring a sparsest possible (biological) signal transduction network consistent with a set of experimental observations with a goal to minimize false positive inferences even if risking false negatives. Special cases of TR_p has been investigated before in different contexts; the best previous results are as follows:

(1) The *minimum equivalent digraph* problem, that correspond to a special case of TR_1 with *no critical edges*, is known to be MAX-SNP-hard, admits a polynomial time algorithm with an approximation ratio of $1.617 + \varepsilon$ for any constant $\varepsilon > 0$ [13] and can be solved in linear time for directed acyclic graphs [1].

(2) A 2-approximation algorithm exists for TR_1 [9, 15].

In this paper, our contributions are as follows:

- We observe that TR_p, for any integer $p > 0$, can be solved in linear time for directed acyclic graphs using the ideas in [1].
- We provide a 1.78-approximation for TR_1 that improves the 2-approximation mentioned in **(2)** above.
- We provide a $2 + o(1)$-approximation for TR_p on general graphs for any prime $p > 0$.

1 Introduction

In this paper, we study the p-ary transitive reduction problem which can be defined as follows. We are given a directed graph $G = (V, E)$ with an edge labeling function $w : E \mapsto \{0, 1, 2, \ldots, p-1\}$ for some *given* integer $p > 0$. The following definitions and notations are used:

- 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 be 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 $\sum_{e \in P} w(e) \pmod{p}$. For the special case of $p = 2$, a path of parity 0 (resp., 1) is called a path of *even* (resp, *odd*) parity.
- The notation $u \stackrel{x}{\Rightarrow} v$ denotes a path from u to v of parity $x \in \{0, 1, 2, \ldots, p-1\}$. If we do not care about the parity, we simply denote the path as $u \Rightarrow v$. An edge will simply be denoted by $u \stackrel{x}{\to} v$ or $u \to v$.
- 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')[1].

The p-ary transitive reduction problem is defined as follows:

> **Problem name:** p-ary Transitive Reduction (TR_p)
>
> **Instance:** A directed graph $G = (V, E)$ with an edge labeling function $w : E \mapsto \{0, 1, 2, \ldots, p-1\}$ 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* $|E'|$.

We are specially interested in the special case of $p = 2$; the corresponding TR_2 problem will be simply referred to as the *Binary* Transitive Reduction (BTR) problem. In the next subsection, we explain the application of BTR to the problem of inferring signal transduction networks in biology.

[1] 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').

2 Motivations

Most biological characteristics of a cell arise from the complex interactions between its numerous constituents such as DNA, RNA, proteins and small molecules [2]. 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 cellular components [10, 11, 17, 18]; however these experiments are rarely conducted in the specific cell type of interest, are not able to probe all possible interactions and regulatory mechanisms, and the resulting maps do not reflect the strength and timing of the interactions. The existing theoretical literature on signaling is focused on networks where the elementary reactions and direct interactions are known [8, 12]; however quantitative characterization 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. 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. For example, in plant signal transduction, the putative G protein coupled receptor GCR1 can physically interact with GPA1 as supported by split-ubiquitin and co-immunoprecipitation experiments [21]. Second, 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. For example, the EMS-generated ost1 mutant is less sensitive to abscisic acid (ABA); thus one can infer that the OST1 protein is a part of the ABA signaling cascade [20]. Third, pharmacological evidence, in which a chemical is used either to mimic the elimination of a particular component, or to exogenously provide a certain component, can lead to similar inferences. For example, a nitric oxide (NO) scavenger inhibits ABA-induced stomatal closure while a NO donor promotes stomatal closure, thus NO is a part of the ABA network [6]. The last two types of inference do not give direct interactions but correspond to pathways and pathway regulation. To synthesize all this information into a consistent network, we need to determine how the different pathways suggested by experiments fit together.

The BTR problem considered in this paper is useful for determining the sparsest graph consistent with a set of experimental observations. Note that we are not assuming that real signal transduction networks are the sparsest possible, since that is clearly not the case. Our goal is to minimize false positive (spurious) inferences, *even if risking false negatives.*

The first requirement of our method is to distill experimental conclusions into qualitative regulatory relations between cellular components. Following [5, 19, 22], we distinguish between positive and negative regulation, usually denoted by the verbs "promote" and "inhibit" and represented graphically as $\rightarrow$ and $\dashv$. Biochemical evidence is represented as component-to-component relationships, such as "A promotes B". However, both genetic and pharmacological evidence leads to inferences of the type "C promotes process(A promotes B)". In this case we use one of the following three representations (i) if the process describes an enzymatic reaction, we represent it as both A and C activating B; (ii) if the interaction between A and B is direct and C does not have a catalytic role, we assume that C activates A; (iii) in all other cases we assume that C activates an unknown intermediary vertex of the AB pathway.

Most edges of the obtained directed graph will not correspond to direct interactions, but starting and end points of directed paths in the (unknown) interaction graph, in other words they represent reachability relationships. Thus our goal is to find the sparsest graph that maintains all reachability relationships, or the minimal transitive reduction of the starting graph. Arcs of the starting graph corresponding to direct interactions will be marked as such and will need to be maintained during the transitive reduction algorithm. A path between node i and j is inhibitory if it contains an odd number of inhibitory interactions. Thus indirect inhibitory edges will need to be reduced to paths that that contain an odd number of inhibitory edges. *It is this transitive reduction problem that is formalized as the BTR problem where edge labels* 0 *and* 1 *correspond to* activations *and* inhibitions*, respectively, and a set of* critical *edges* $E_{\text{critical}} \subseteq E$ *corresponding to direct interactions.*

For the sake of completeness, we briefly describe the entire approach of minimizing the network (see [19] for a specific example done by manual curation). We start by synthesizing the vertex-to-vertex reachability relationships, then we turn to the vertex-to-path influences. If existing paths already explains a vertex-to-path relationship, no new intermediaries need to be added, otherwise we incorporate it in one of the three representations described above. Next we determine the obtained graph's transitive reduction subject to the constraints that no edges flagged as direct are eliminated. Finally we identify and collapse pairs of equivalent intermediary vertices (*e.g.*, adjacent vertices in a linear chain) if that procedure does not change the reachability relationships of the real vertices.

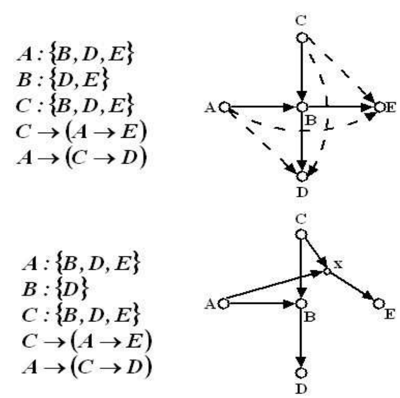

Figure 1: Illustration of the graph synthesis process in two cases.

Figure 1 shown here illustrates the graph synthesis process in two cases (specified by the reachability sets and pathway influence information displayed on the left side) that differ in a single reachability relationship only. In both cases the edges marked as dashed on the top graph will be eliminated. In the first case the pathway regulatory information is already incorporated thus no new intermediary vertices need to be added. In the second case the relationship between C, A and E necessitates the addition of a new vertex x. The addition of the BE edge would make B and x equivalent in terms of reachability; thus x could be identified with B.

3 Previous Results

Obviously, TR_p is NP-complete since TR_1 includes the problem of finding directed Hamiltonian cycle in a graph. If $E_{\mathrm{critical}} = \emptyset$, TR_1 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 $\varepsilon > 0$ [13] and can be solved in polynomial time for directed acyclic graphs [1]. 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 [9, 15]; this implies a 2-approximation for TR_1 without the restriction $E_{\mathrm{critical}} = \emptyset$.

4 Summary of Our Results

In this paper we investigate the TR_p problem with a special emphasis on $p > 1$ since TR_2 captures the most important part of the network minimization algorithm mentioned in the introduction, namely that of finding efficient approximate solution of the binary transitive reduction (BTR) problem. The rest of the paper is organized as follows:

- In Section 6, we observe that TR_p for any $p > 0$ can be solved in polynomial time for directed acyclic graphs using the ideas in [1]. The solution is valid even if there are parellel edges or edges with multiple labels.

- In Section 7.3 we provide a 1.78-approximation for TR_1 and a $2 + o(1)$-approximation for TR_p for any prime $p > 1$ when the given graph is strongly connected.

- In Section 7.4 we generalize the approximation results of the previous section to provide a 1.78-approximation for TR_1 and a a $2+o(1)$-approximation for TR_p for any prime $p > 1$ for any given graph; the 1.78-approximation for TR_1 for arbitrary graphs improves the previous 2-approximation for this case [9, 15].

Informally, our approach for solving TR_p for general graphs involve developing approximate solutions for strongly connected components and then combining these solutions. Because of the presence of edge labels which participate via a modulo p addition in the parity of a path and the existence of critical edges, our solution differs from that in [9, 13, 15]. We combine these solutions via appropriate modifications to our solutions for TR_p on DAG and designs of "gadgets" that would ensure that parities of various paths are preserved.

5 Basic Notations and Terminologies

An approximation algorithm for a minimization problem, that seeks to minimize an objective function, of performance or approximation ratio α (or simply an α-approximation) is a polynomial-time algorithm that provides a solution with a value of the objective function that is at most α times the optimal value of the objective function. We also use the following notations for convenience:

- $OPT(G) = |E_{\text{opt}}(G)|$ denotes the number of edges in an optimal solution $E_{\text{opt}}(G)$ of TR_p for the graph G.
- $\oplus_p$, $\ominus_p$ and $=_p$ denote addition, subtraction and equality modulo p.

The following fact from elementary number theory will prove very useful to us.

Fact 1 *For any prime $p > 1$ and any integer $0 < x < p$, $\{i \cdot x \pmod p \mid 0 < i < p+1\} = \{0, 1, \ldots, p-1\}$.*

6 Polynomial-time Solution for Directed Acyclic Graphs (DAG)

Lemma 1 *TR_p can be solved in polynomial time when the given graph G is a DAG even if G has*

- *parallel edges,*
- *edges that are labeled by $\{0, 1, 2, \ldots, p-1\}$,* i.e., *edges of the form $u \stackrel{0,1,2,\ldots,p-1}{\longrightarrow} v$ such that traversing the edge provide a path of parity x for every $x \in \{0, 1, 2, \ldots, p-1\}$.*

Proof. We first remove edges, if necessary, to ensure that no two parallel edges have the same label. Also, obviously, if there is an edge from a vertex u to another vertex v labeled by $\{0, 1, 2, \ldots, p-1\}$, we can remove any other edge that exists from u to v. After this, the algorithm shown below can be used; it essentially uses ideas similar to that in [1].

compute in $O(|V| + |E|)$ time a topological sorting of G with
    $v_1, v_2, \ldots, v_n$ as the topological order of nodes
for $j = n, n-1, n-2, \ldots, 1$
    for $i = j-1, j-2, \ldots, 1$
        if $v_i \stackrel{S}{\to} v_j$ exists for some $S \subseteq \{0, 1, 2, \ldots, p-1\}$:
        then
            if there exists another path $v_i \stackrel{x}{\Rightarrow} v_j$ for every $x \in S$ and $v_i \stackrel{S}{\to} v_j \notin E_{\text{critical}}$
            then
                delete edge $v_i \stackrel{S}{\to} v_j$

The algorithm can be obviously implemented in polynomial time. Moreover, no edge belonging to E_{critical} is deleted and since an edge is deleted only if there is an alternate path of same parity between its two endpoints, reachable(E') =reachable(E); thus the algorithm returns a valid solution.

Now we show that solution is optimal. We will prove the claim by induction on $|V|$. The claim is trivial for $|V| = 1$. Suppose that the claim is true for all $|V| < k$. Consider the case of $|V| = k$. Consider the subgraph $G' = (V', E')$ of $G = (V, E)$ induced by the vertices $V' = \{v_2, \ldots, v_k\}$ for which our algorithm an optimal solution $E^{G'}_{\text{opt}}$. Remember that $v_i \to v_j \in E$ implies $i < j$, thus $E^{G}_{\text{opt}} \subseteq E^{G'}_{\text{opt}} \bigcup \left(\cup_{v_1 \to v_j \in E} \{v_1 \to v_j\} \right)$. Now, consider running our algorithm on G and let $v_1 \xrightarrow{S} v_j$ be the first edge in which our solution differs from every optimal solution for some $S \subseteq \{0, 1, 2, \ldots, p-1\}$. There are now two cases to consider:

- We select $v_1 \xrightarrow{S} v_j$ but no optimal solution selects it. But, any optimal must contain the path $v_1 \overset{x}{\Rightarrow} v_j$ for each $x \in S$. Since $v_i \to v_j \in E$ implies $i < j$ and no two parallel edges have the same label, the optimal solution contains a path of the form $v_1 \xrightarrow{x'} v_i \overset{x \ominus_p x'}{\Longrightarrow} v_j$ for some $1 < i < j$ and each $x \in S$. But then these paths ensure that we will not select $v_1 \xrightarrow{S} v_j$, a contradiction.
- We did not select $v_1 \xrightarrow{S} v_j$ but every optimal solution selected it. Then, in fact, we are selecting fewer edges than any optimal solution upto this point.

❑

7 Efficient Approximation Algorithms for TR_p

The goal of this section is to design an efficient approximation algorithm for TR_p by proving the following result.

Theorem 2 *There is a $2 + o(1)$-approximation algorithm for TR_p for any prime $p > 1$ and a 1.78-approximation algorithm for TR_1.*

The next few subsections prove the above theorem step-by-step.

7.1 Characterization of Strongly Connected Components

Consider a strongly connected component $C = (V_C, E_C)$ of the given graph G. We consider two types of such components:

Multiple Parity Components: for any two vertices $u, v \in V_C$, $u \overset{x}{\Rightarrow} v$ exists in C for every $x \in \{0, 1, 2, p-1\}$.

Single Parity Components: for any two vertices $u, v \in V_C$, $u \overset{x}{\Rightarrow} v$ exists in C for exactly one x from $\in \{0, 1, 2, p-1\}$.

Lemma 3
(a) *Every strongly connected component of G is either of single parity or of multiple parity.*
(b) *A strongly connected component C is of multiple parity if and only if C contains a simple cycle of non-zero parity.*

Proof. Suppose that C is not a single parity component. Then there exists a pair of vertices u and v such that $u \stackrel{\alpha}{\Rightarrow} v$ and $u \stackrel{\beta}{\Rightarrow} v$ exists in C with $\alpha \neq \beta$. Since C is strongly connected, there exists a path, say of parity γ, from v to u. Since $\alpha \neq \beta$, either $\alpha + \gamma \neq 0 \pmod{p}$ or $\beta + \gamma \neq 0 \pmod{p}$. Thus, we have a cycle $u \stackrel{x}{\Rightarrow} u$ for some $x \in \{1, 2, \ldots, p-1\}$. By Fact 1 and traversing this cycle an appropriate number of time, we therefore have a cycle $u \stackrel{x}{\Rightarrow} u$ for every $x \in \{0, 1, 2, \ldots, p-1\}$. If the cycle is not simple, then either it contains a simple cycle of non-zero parity or it contains only simple cycles of zero parity whose removals will make it a simple cycle of non-zero parity. This proves the "only if" part of (b) of the lemma.

Thus, suppose that $u \stackrel{x}{\Rightarrow} u$ for every $x \in \{0, 1, 2, \ldots, p-1\}$. Now, consider any two vertices u' and v' in C and consider any $x \in \{0, 1, 2, \ldots, p-1\}$. To prove (a), we wish to show that $u' \stackrel{x}{\Rightarrow} v'$ exists in C. Since C is strongly connected, there exists a path of some parity z from u' to u and there exists a path of some parity y from u to v'. Finally, there exists a path $u \stackrel{w}{\Rightarrow} u$ where $w =_p x - (y + z)$. To see the "if" part of (b), note that again a non-zero parity simple cycle $u \stackrel{x}{\Rightarrow} u$ for some $x \in \{1, 2, \ldots, p-1\}$ implies $u \stackrel{x}{\Rightarrow} u$ for every $x \in \{0, 1, 2, \ldots, p-1\}$ and thus provides $u' \stackrel{x}{\Rightarrow} v'$ for any $u', v' \in V_C$ and $x \in \{0, 1, 2, \ldots, p-1\}$. ❑

It is easy to state a straightforward dynamic programming approach to determine, given a strongly connected component C, if C contains a simple cycle of non-zero parity using ideas similar to that in the Floyd-Warshall transitive closure algorithm [4]. Let $V_C = \{1, 2, \ldots, n\}$. Let $N(i, j, k, x)$ be 1 if there is a simple path of parity x from vertex i to vertex j using intermediate vertices numbered no higher than k and $P(i, j, k, x)$ denote the corresponding path if it exists. Then,

- **for** each $x \in \{0, 1, 2, \ldots, p-1\}$ and **for** each $i, j \in \{1, 2, \ldots, n\}$:
  - **if** $i \stackrel{x}{\rightarrow} j \in E_C$ **then** $N(i, j, 0, x) = 1$ and $P(i, j, k, x) = i \stackrel{x}{\rightarrow} j$
    else $N(i, j, 0, x) = 0$ and $P(i, j, k, x) = \emptyset$.
- **for** $k > 0$, each $i, j \in \{1, 2, \ldots, n\}$ and each $x \in \{0, 1, 2, \ldots, p-1\}$:
  - **if** $N(i, j, k-1, x) = 1$ **then** $N(i, j, k, x) = 1$ and $P(i, j, k, x) = P(i, j, k-1, x)$;
  - **else if** $N(i, k, k-1, y) = N(k, j, k-1, z) = 1$ and $y + z =_p x$, **then** $N(i, j, k, x) = 1$ and $P(i, j, k, x)$ is the concatenation of the paths $P(i, k, k-1, y)$ and $P(k, j, k-1, z)$.
  - **else** $N(i, j, k, x) = 0$ and $P(i, j, k, x) = \emptyset$.

The running time is $O(p \cdot |V_C|^3)$. The final answer is obtained by checking $N(i, i, n, 1)$ for each $i, j \in \{1, 2, \ldots, n\}$. Moreover, such a simple non-zero parity cycle, if it exists, can be found from the corresponding $P(i, j, n, 1)$'s and by observing that if the cycle is not simple, then either it contains a simple cycle of non-zero parity or it contains only simple cycles of zero parities whose removals will make it a simple cycle of non-zero parity. As a by-product of the above discussions and due to the results in [9], we also obtain the following corollary.

Corollary 4 *Consider the TR_p problem, when p is 1 or a prime number, on a strongly connected graph $G = (V, E)$. Then, $|V| \leq OPT(G) \leq 3|V|$.*

7.2 Solving TR_p for a Strongly Connected Component

The main result of this subsection is as follows.

Theorem 5 *Let the given graph G be strongly connected. Then, we can design a 2-approximation algorithm for TR_p when $p > 1$ is a prime and a 1.78-approximation algorithm for the TR_1.*

In the rest of this subsection, we provide a proof of the above theorem. First, we will need to review some existing algorithms for special cases of BTR. For the remainder of this subsection, we assume that our input graph $G = (V, E)$ is strongly connected; in this case obviously $OPT(G) \geq |V|$.

The following notations and terminologies will be used:

- By "the TR_1 problem on a graph G" we mean the TR_1 problem on G with all edge labels of G being set to zero.

- By "the TR_1 problem on a graph G with no critical edges" we mean the TR_1 problem on G with every edge marked as *not critical*, *i.e.*, E_{critical} being temporarily set to $\emptyset$.

- $E_{\mathrm{opt}}(G)$ is an optimal solution of TR_p on G.

- $E^1_{\mathrm{opt}}(G)$ is an optimal solution of TR_1 on G. For notational convenience, let:

  - $\alpha = |E_{\mathrm{critical}}|$;
  - $\beta = |E^1_{\mathrm{opt}} \setminus E_{\mathrm{critical}}|$.

- $E_{\mathrm{max_{opt}}}(G)$ is an optimal solution of TR_1 on G with no critical edge.

Note that:

- $|E^1_{\mathrm{opt}}(G)| = \alpha + \beta$;

- $|E_{\mathrm{opt}}(G)| \geq |E^1_{\mathrm{opt}}(G)| = \alpha + \beta$;

- $|E_{\mathrm{max_{opt}}}(G)| \leq |E^1_{\mathrm{opt}}(G)| = \alpha + \beta$.

7.2.1 The Cycle Contraction Algorithm of Khuller, Raghavachari and Young [13, 14]

This algorithm for TR_1 with no critical edges, which we refer to as Algorithm A1, works as follows. Contraction of an edge $u \to v$ is to merge u and v into a single vertex and delete any resulting self-loops or multi-edges. The algorithm selects a constant $k \geq 3$, and then, for $i = k, k-1, k-2, \ldots, 3$, as long as the graph contains a cycle C of at least i edges contracts the edges of C and selects these edges in the solution. Finally, when the graph contains cycles of length at most 3, an exact algorithm for MED [14] is used. Let E_1 be the set of edges selected by this algorithm. The results of Khuller et al. translate to the following facts:

- E_1 is a correct solution if G is of single parity and $E_{\text{critical}} = \emptyset$.
- $|E_1| \leq \left(\frac{\pi^2}{6} - \frac{1}{36} + \frac{1}{k(k-1)}\right) E_{\max_{\text{opt}}}(G) < \left(1.617 + \frac{1}{k(k-1)}\right) E_{\max_{\text{opt}}}(G)$.

7.2.2 The Spanning Arborescence Algorithm [9]

This algorithm for TR_1, which we refer to as Algorithm A2, works as follows when adapted to BTR. Given the graph $G = (V, E)$, define the weight[2] $\texttt{wt}(e) = 0$ of every edge in $e \in E_{\text{critical}}$ and the weights of all other edges to be 1. In the *first stage*, find a minimum weight spanning arborescence[3] T of minimum weight of G, say rooted at vertex v. Then, in the second stage, set $\texttt{wt}(e) = 0$ for $e \in E_{\text{critical}} \cup T$, set the weights of the rest of the edges to be 1, and find a minimum weight *reverse* spanning arborescence T' rooted to vertex v, *i.e.*, a minimum weight spanning directed acyclic subgraph such that v has no outgoing edges and every other vertex has exactly one outgoing edge. Then, the union E_2 of the edges in T and T' together with all critical edges not selected in T or T' is returned as the solution. The proofs in [9] imply that:

- E_2 is a correct solution if G is of single parity.
- Algorithm A2 is a 2-approximation, *i.e.*, the sum of weights of edges in E_2 is at most twice the sum of weights of edges in an optimal solution of TR_1 on G. In other words, $|E_2| \leq \alpha + 2\beta$.

We now review the algorithm for finding the minimum weight spanning arborescence [3, 7, 16] since it will be necessary to modify it slightly later. First, we select for each node an incoming edge of minimum weight. If these edges do not give a spanning arborescence, then there must be a directed cycle C formed by some of these edges. Let $\delta_C \in \{0, 1\}$ be the minimum of the weights of the edges in C. We contract C to a "supernode" and decrease the weight of every edge $i \to k$ from vertex $i \notin C$ to vertex $k \in C$ by $p - \delta$ where p the weight of

[2] These weights should not be confused with the edge labeling function w used in the definition of BTR.

[3] A spanning arborescence is a directed acyclic spanning subgraph such that every node except one node (the root) has exactly one incoming edge; its weight is just the sum of the weight of its edges.

the unique edge in C that is incoming to k. The process is then repeated on the reduced graph, and continued until a pseudonode is created with zero in-degree. The supernodes are then expanded in reverse order. Each time a supernode is expanded, *exactly one of its edge is discarded which would produce two incoming edges to a node.* The same algorithm with minor modification can be used to find the reverse spanning arborescence rooted to v also. Algorithm A2 can be implemented to run in $O(\min\{|E|\log|V|,\ |V|^2\})$ time; see [23] for details.

7.3 Augmenting and Combining the Two Algorithms

7.3.1 Augmenting Algorithms A1

Lemma 6 *Algorithm A1 can be modified such that it solves the* TR_p *problem on* G *and uses at most* $2.236\alpha + 1.618\beta$ *edges.*

Proof. We will show how to augment Algorithm A1 to handle

- a non-empty E_{critical} and
- a multiple parity component.

By Lemma 3(b), if $G = (V, E)$ is of multiple parity, we can find a simple cycle C of non-zero parity whose addition to the solution of Algorithm A1 would provide paths of all parities between every pair of vertices. Let k be the constant used in the description of Algorithm A1. If $|V| < k^2$, we can solve the problem in $O(1)$ time. Otherwise, obviously $|E_{\text{opt}}(G)| \geq |E^1_{\text{opt}}(G)| = \alpha + \beta \geq |V| = k^2$. If C contains k or more edges, we can let C be the first cycle contracted by Algorithm A1. Otherwise, we just add the edges in C, if necessary, to the solution returned by Algorithm A1. With this modification, $|E_1| \leq \left(1.617 + \frac{1}{k(k-1)} + \frac{1}{k}\right)(\alpha + \beta)$, or $|E_1| \leq 1.618(\alpha + \beta)$ by appropriate choice of k.

Now, we show how to handle the edges in E_{critical}. We replace an edge $u \xrightarrow{x} v \in E_{\text{critical}}$ by introducing a new vertex $X_{u,v}$ and two new edges $u \xrightarrow{0} X_{u,v}$ and $X_{u,v} \xrightarrow{x} v$. Let $G' = (V', E')$ be the new graph. It is clear that all the new edges must be selected by Algorithm A1 since otherwise either reachability from some $X_{u,v}$ or reachability to some $X_{u,v}$ will be violated. Moreover, there are exactly 2α new edges. Consider a solution returned by Algorithm A1 on G'. Then, we can contract the new edges to the original edges in G to get a solution E_1 for G. Note that an optimal solution for TR_1 on G' with no critical edges uses at most $2\alpha + \beta$ edges since, in particular, one solution of TR_1 on G' with no critical edges involve taking 2α new edges and β edges from $E^1_{\text{opt}} \setminus E_{\text{critical}}$. The contraction removes α edges from this solution. Thus, the total number of edges selected with this modification is atmost $|E_1| \leq 1.618(2\alpha + \beta) - \alpha = 2.236\alpha + 1.618\beta$. ❑

7.3.2 Augmenting Algorithms A2

Lemma 7 *Algorithm A2 can be modified such that it solves the* TR_p *problem on* G *and uses at most* $2\alpha + 2\beta + 1$ *edges.*

Proof. We show how to augment Algorithm A2 to handle a multiple-parity graph. Again, by Lemma 3(b), we can find a simple cycle C of non-zero parity whose addition to the solution of Algorithm A2 would provide paths of all parities between every pair of vertices. Assume that at least one edge of C does not belong to E_{critical} since otherwise we do not incur any additional cost in adding C to our solution. Remember that the first stage of Algorithm A2 starts by selecting one incoming edge of minimum weight for every vertex. We will modify $\mathtt{wt}(e)$ for some edges $e \in C$ such that Algorithm A2 starts by selecting all the edges in C and show that the additional cost incurred is not too much. We classify each edge $u \to v \in C$ in the following way:

(i) $\mathtt{wt}(u \to v) = 0$. We can then surely select this edge.

(ii) $\mathtt{wt}(u \to v) = 1$ and every edge $u' \to v$ has $\mathtt{wt}(u' \to v) = 1$. Then also we can select this edge.

(iii) $\mathtt{wt}(u \to v) = 1$ and some edge $u' \to v$ has $\mathtt{wt}(u' \to v) = 0$. Then, we change $\mathtt{wt}(u \to v)$ to zero which would allow us to select this edge. Let S be the set of all these edges. Since $\mathtt{wt}(u' \to v) = 0$ implies $u' \to v \in E_{\text{critical}}$, $|S| \leq \alpha$ and thus the total number of additional edges that appear in the solution of Algorithm A2 due to these changes is at most α

(iv) When the supernode for C is expanded, all except one edge of C, say the edge $u \to v$, are selected. Thus, this edge which was not selected adds 1 to the count of additional edges.

Since we changed some edge weights from one to zero, the sum of edge weights in the solution of Algorithm A2 does not increase. Thus, with this modification, we get $|E_2| \leq 2(\alpha+\beta)+1$. ❑

7.3.3 Combining Algorithms A1 and A2

1.78-approximation for TR_1 for a Strongly Connected Component

For TR_1 a strongly connected graph cannot be of multiple parity. In this case, our solution is the better of the solutions provided by modified Algorithm A1 and Algorithm A2 without the modification. The approximation ratio is $\rho = \min\left\{\frac{2.236\alpha+1.618\beta}{\alpha+\beta}, \frac{\alpha+2\beta}{\alpha+\beta}\right\} = \min_{0\leq\beta<\infty}\{f(\beta), g(\beta)\}$ where $f(\beta) = \frac{2.236\alpha+1.618\beta}{\alpha+\beta}$ and $g(\beta) = \frac{\alpha+2\beta}{\alpha+\beta}$. Note that $f(\beta)$ is a decreasing function of β, $g(\beta)$ is an increasing function of β, $f(0) > g(0)$ and $\lim_{\beta\to\infty} f(\beta) < \lim_{\beta\to\infty} g(\beta)$. Thus, $\rho \leq f(\beta_0)$ where β_0 is that value of β that satisfies $f(\beta_0) = g(\beta_0)$. Calculations show that $\beta_0 = \frac{1.136}{0.38}\alpha$ and thus $f(\beta_0) = \frac{2.236+\frac{1.618\times1.136}{0.382}}{1+\frac{1.136}{0.382}} < 1.78$.

$2 + o(1)$-approximation for TR_p for a Strongly Connected Component

In this case, our solution is simply the output of modified Algorithm A2. Since $|E_2| < 2(\alpha+\beta)+1$ and $|V| \leq |E_{\max_{\rm opt}}| = \alpha+\beta$, we have a $2+o(1)$-approximation.

7.4 Approximating TR_p for General Graphs: Combining DAG and Strongly Connected Component Solutions

Now that we have an approximation algorithm for each strongly connected component and an exact algorithm for DAGs, can we use these to design an approximation algorithm for a general graph? First, we outline a general strategy for this purpose. Then we show how to apply the strategy for single and multiple parity components. The details of the strategy are somewhat different from that in [1, 13] because the strongly connected components can be of two types of parity.

Let $G = (V, E)$ be the given graph with $C_1, C_2, \ldots, C_m$ being the m strongly connected components where the $i^{\rm th}$ component C_i contains n_i vertices $v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}$. By a *gadget* for component C_i we mean a DAG of $O(p^2)$ size on a new set of vertices. No two gadgets share a vertex.

Suppose that we have gadgets $\Gamma_1, \Gamma_2, \ldots, \Gamma_m$ for the components $C_1, C_2, \ldots, C_m$, respectively. Let $E_{\rm gadget}$ be the set of all edges in all the gadgets. Let G' be a new graph obtained from G by a polynomial-time procedure $T_{\rm cycle-to-gadget}$ of the following nature:

- Every C_i was replaced by the corresponding gadget Γ_i.
- Every incoming edge $u \stackrel{x}{\to} v$ to some vertex v in C_i is replaced by either one edge (the "first type") $u \stackrel{x}{\to} v'$ or by at most a group of $1 < j \leq p$ edges (one group of "second type") $u \stackrel{\sigma_1}{\to} v', \ldots, u \stackrel{\sigma_j}{\to} v'$ to vertices in Γ_i, and similarly every outgoing edge $u \stackrel{x}{\to} v$ from some vertex u in C_i is replaced by either one edge (the "first type") $u' \stackrel{x}{\to} v$ or by at most a group of $1 < r \leq p$ edges (one group of "second type") $u' \stackrel{\sigma_1}{\to} v, \ldots, u' \stackrel{\sigma_r}{\to} v$ from vertices u' in Γ_i. Let $E_{in/out,1}$ $E_{in/out,2}$ be the set of all such edges of the first and second types, respectively.

Furthermore, suppose the above procedure $T_{\rm cycle-to-gadget}$ guarantees the following invariants:

(P1) any valid solution of TR_p for G' must contain all the edges in $E_{\rm gadget}$, and

(P2) if an edge $u \stackrel{x}{\to} v$ in C_i was replaced by a set of second type edges in G', then there is an optimal solution of TR_p on G' that selects *all* of them.

Due to Invariant **(P2)**, for the purpose of calculating the correspondence of an optimal solution of G' with that of G, we can consider one group of second type edges as just one edge. Abusing notations slightly, let $E_{in/out,2}$ be the set of edges obtained by such replacements. Let $E_{in/out} = E_{in/out,1} \cup E_{in/out,2}$.

Observe that G' is a DAG. To see this, consider the graph G'' obtained by removing the edges in $E_{\rm gadget}$ from G'. This graph specified the interconnections between strongly connected components and is therefore a DAG. Now, since each Γ_i is a DAG, it is easy to put back the missing edges in a topological sorting of G'' such that the property of topological ordering is not violated. Thus, we can solve the TR_p problem for G' exactly via the algorithm described in Lemma 1. Suppose that the solution uses a subset $E'_{in/out} \subseteq E_{in/out}$ of the edges.

Now, suppose that we have an optimal solution $E_{\mathrm{opt}}(C_i)$ for TR_p for the connected component C_i. Given an optimal solution $E_{\mathrm{opt}}(G')$ of G', we associate it with a subgraph G_{approx} of G via a procedure $T_{\mathrm{gadget-to-cycle}}$ in the following manner.

- Replace each Γ_i by the vertices and edges in the optimal solution $E_{\mathrm{opt}}(C_i)$ of C_i.
- Replace an incoming/outgoing edge to/from Γ_i by an incoming/outgoing edge to/from the corresponding vertex in C_i.

Suppose that the set of edges $E'_{in/out}$ are mapped to subsets E' of edges of G by $T_{\mathrm{gadget-to-cycle}}$. Note that, due to Invariant **(P2)**, $|E|' = |E'_{in/out}|$. Suppose that our transformation satisfies the following invariant:

> ($\star$) G_{approx} is an optimal solution of TR_p for G.

Then, $OPT(G) = OPT(G') - |E_{\mathrm{gadget}}| + \sum_{i=1}^{m} |E_{\mathrm{opt}}(C_i)|$. Suppose now that we have a ρ-approximation of TR_p on each C_i and in G_{approx} we replace each C_i by this approximate solution. Then, we have a ρ-approximation of TR_p on G since we have a valid solution of TR_p on G and the number of edges we use is at most $OPT(G') - |E_{\mathrm{gadget}}| + \rho \cdot \sum_{i=1}^{m} |E_{\mathrm{opt}}(C_i)| \leq \rho \cdot OPT(G)$.

But, how do we check Invariant ($\star$) described above? Suppose that we could maintain the following two invariants:

(P3) $G_{\mathrm{approx}} = (V_{\mathrm{approx}}, E_{\mathrm{approx}})$ is a valid solution of TR_p on G.

(P4) A subgraph that is an optimal solution $E_{\mathrm{opt}}(G)$ of TR_p on G, after application of the procedure $T_{\mathrm{cycle-to-gadget}}$, is transformed to a subgraph $G_{\min} = (V_{\min}, E_{\min})$ that is a valid solution of TR_p on G'.

To show that the above invariants are sufficient, the following proposition becomes useful.

Proposition 1 *$E_{\mathrm{opt}}(G) \cap E_{C_i} = E_{\mathrm{opt}}(C_i)$ for every strongly connected component $C_i = (V_{C_i}, E_{C_i})$ and $E_{\mathrm{opt}}(G') \cap E_{\Gamma_i} = E_{\mathrm{opt}}(\Gamma_i)$ for every gadget $\Gamma_i = (V_{\Gamma_i}, E_{\Gamma_i})$.*

Proof. This was essentially observed in [1]. We only prove the claim $E_{\mathrm{opt}}(G) \cap E_{C_i} = E_{\mathrm{opt}}(C_i)$; the other one is similar. The claim follows since otherwise there will be a path from some vertex in V_{C_i} to another vertex in C_i in which at least one intermediate vertex in the path does not belong to V_{C_i}; thus, this intermediate vertex also belongs to V_{C_i} contradicting the maximality of C_i. ❑

Now suppose that $|E_{\text{opt}}(G)| < |E_{\text{approx}}|$. Obviously,

$$OPT(G') = |E_{\text{approx}}| - \sum_{i=1}^{m} |E_{\text{opt}}(C_i)| + |E_{\text{gadget}}|$$

But, using Proposition 1 and Invariant **(P4)**

$$|E_{\min}| = |E_{\text{opt}}(G)| - \sum_{i=1}^{m} |E_{\text{opt}}(C_i) + |E_{\text{gadget}}|$$

Thus $|E_{\min}| < OPT(G')$, contradicting the optimality of $OPT(G')$. Thus, $|E_{\text{opt}}(G)| \geq |E_{\text{approx}}|$ and since, by Invariant **(P3)**, G_{approx} is a valid solution of TR_p on G we must have $|E_{\text{opt}}(G)| = |E_{\text{approx}}|$. Summarizing, we have proved the following lemma.

Lemma 8 *Suppose that we can design the procedures $T_{\text{cycle-to-gadget}}$ and $T_{\text{gadget-to-cycle}}$ such that invariants* **(P1)**–**(P4)** *are satisfied. Then a ρ-approximation of TR_p for a strongly connected graph implies a ρ-approximation of TR_p for general graphs.*

Finally, note that due to the Proposition 1, when checking for Invariants **(P3)** or **(P4)**, we need to check only for those paths $u \overset{x}{\Rightarrow} v$ in G such that u and v do not belong to the same component and those paths $u \overset{x}{\Rightarrow} v$ in G' such that u and v do not belong to the same gadgets in G'.

7.4.1 Handling Strongly Connected Components of Multiple Parity

First assume that the graph has no components of single parity. Note that by definition a multiple parity component must have at least 2 vertices. Each C_i corresponds to a gadget Γ_i in the following manner. We introduce one new vertex X_{C_i}. An incoming edge in G to some vertex in C_i now becomes p incoming edges of parities $0, 1, \ldots, p-1$ to X_{C_i} and an outgoing edge from some vertex in C_i now becomes p outgoing edges from X_{C_i} of parities $0, 1, \ldots, p-1$. Figure 2 illustrates the gadget for $p = 2$.

We need to verify the invariants **(P1)**–**(P4)**. **(P1)** vacuously holds. By induction on the number of components, it suffices to verify **(P2)**–**(P4)** when there is one component of multiple parity, say $C = (V_C, E_C)$.

First we verify **(P2)**. For a vertex X_C, let the set of *violating* neighbors N be the set of vertices v such that for some $\emptyset \subset R_v \subset \{0, 1, 2, \ldots, p-1\}$ there is an optimal solution that selects $X_C \overset{x}{\to} v$ for every $x \in R_v$ but not $X_C \overset{z}{\to} v$ for every $z \in \{0, 1, 2, \ldots, p-1\} \setminus R_v$. We will show how to decrease the number of violating neighbors by one without increasing the number of edges in the solution; iterating the process will ensure that **(P2)** is satisfied. Consider such a violating neighbor v. But, $E_{\text{opt}}(G')$ must contain the paths $X_C \overset{z}{\Rightarrow} v$ for every $z \in \{0, 1, 2, \ldots, p-1\} \setminus R_v$. Fix any specific z and let $X_C \overset{y}{\to} v'$ be the first edge on this path. Then, for every edge $X_C \overset{x}{\to} v$ remove this edge and add the edge $X_C \overset{y'}{\to} v'$ if it was not already selected where $y' =_p x - z + y$. The case of edges incoming to X_C is similar.

To verify **(P3)**, one must consider the following cases.

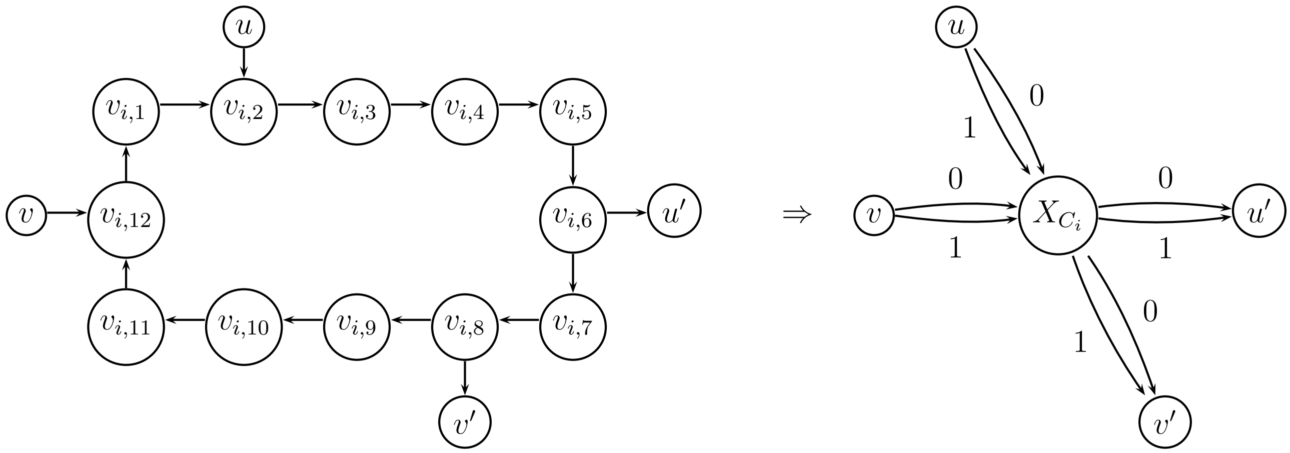

Figure 2: Gadget Γ_i for a multiple parity component C_i for the TR_2 problem. Due to Invariant **(P2)**, we can replace the pairs of edges $\{u \xrightarrow{0} X_{C_i}, u \xrightarrow{1} X_{C_i}\}$, $\{v \xrightarrow{0} X_{C_i}, v \xrightarrow{1} X_{C_i}\}$, $\{X_{C_i} \xrightarrow{0} u', X_{C_i} \xrightarrow{1} u'\}$ and $\{X_{C_i} \xrightarrow{0} v', X_{C_i} \xrightarrow{1} v'\}$ by $u \xrightarrow{0,1} X_{C_i}$, $v \xrightarrow{0,1} X_{C_i}$, $X_{C_i} \xrightarrow{0,1} u'$ and $X_{C_i} \xrightarrow{0,1} v'$, respectively.

(I) $u \overset{x}{\Rightarrow} v$ is in G when $u \in V_C$ and $v \notin V_C$. Suppose that u' is the last vertex on this path that belongs to C; thus the path is of the form $u \overset{x_1}{\Rightarrow} u' \xrightarrow{x_2} v' \overset{x_3}{\Rightarrow} v$. Thus $X_C \overset{x_2 \oplus_p x_3}{\Longrightarrow} v$ exists in $E_{\text{opt}}(G')$. Suppose that this path translates to the path $u'' \overset{x_2 \oplus_p x_3}{\Longrightarrow} v$ in G_{approx} for some $u'' \in C$. Since $u \overset{z}{\Rightarrow} u''$ exists in C for every $z \in \{0, 1, \ldots, p-1\}$, we have the path $u \overset{x}{\Rightarrow} v$ in G_{approx}.

(II) $u \overset{x}{\Rightarrow} v$ is in G when $u \notin V_C$ and $v \in V_C$. Similar to **(I)**.

(III) $u \overset{x}{\Rightarrow} v$ is in G when $u, v \notin V_C$ but the path contains at least one vertex from V_C. Let $u \overset{x_1}{\Rightarrow} u' \overset{x_2}{\Rightarrow} v' \overset{x_3}{\Rightarrow} v$ where u' and v' are the first and the last vertices that belong to C. But, then $u \overset{x_1}{\Rightarrow} u'$ and $v' \overset{x_3}{\Rightarrow} v$ exist in G_{approx} by **(I)** and **(II)**, respectively, and $u' \overset{x_2}{\Rightarrow} v'$ exist in G_{approx} since C is a multiple parity component and $T_{\text{gadget-to-cycle}}$ replaced every X_C by $E_{\text{opt}}(C)$.

To verify **(P4)**, one must consider the following cases.

(I) $X_C \Rightarrow v$ is in G'. Since every outgoing edge from X_C has parity z for every $z \in \{0, 1, 2, \ldots, p-1\}$, $X_C \overset{z}{\Rightarrow} v$ exist in G' each $z \in \{0, 1, 2, \ldots, p-1\}$. Pick any vertex $u \in C$. Obviously, $u \overset{0}{\Rightarrow} v = u \overset{x_1}{\Rightarrow} u' \xrightarrow{x_2} v' \overset{x_3}{\Rightarrow} v$ exists in $E_{\text{opt}}(G)$ where u' is the last vertex on the path that is in C. Then, both $X_C \xrightarrow{\{0,1,2,\ldots,p-1\}} v'$ and $v' \overset{x_3}{\Rightarrow} v$ are in $G_{\min}$.

(II) $v \Rightarrow X_C$ is in G'. Similar to **(I)**.

(III) $u \Rightarrow X_C \Rightarrow v$ is in G'. Follows from **(I)** and **(II)** since both $u \Rightarrow X_C$ and $X_C \Rightarrow v$ are in $G_{\min}$.

7.4.2 Handling Strongly Connected Components of Single Parity

Now all the multiple parity components have been replaced by $T_{\text{cycle-to-gadget}}$ and G has components of single parity. Let v be any vertex in a single parity component $C_q = (V_{C_q}, E_{C_q})$. Define the following notation:

$$[j, C_q] = \{x \in V_{C_q} \mid v \overset{j}{\Rightarrow} x \text{ exists in } C_q\}$$

Note that for any $u, v \in V_{C_q}$, $u \overset{x}{\Rightarrow} v$ is in C_q if and only if $v \overset{p-x}{\Longrightarrow} u$ is in C_q since otherwise $u \overset{y}{\Rightarrow} u$ is in C_q for some $y \in \{1, 2, \ldots, p-1\}$ which is not allowed due to Lemma 3(b).

Lemma 9 *For any $u \in [i, C_q]$ and $u' \in [j, C_q]$, $u \overset{x}{\Rightarrow} u'$ if and only if $x =_p j - i$.*

Proof. Consider the path $u \overset{p-i}{\Longrightarrow} v \overset{j}{\Rightarrow} u'$ and note that there are paths of only one parity between any two vertices in a single parity component. ❑

Via the above lemma, we can design the following gadget for the single parity component $C_q = (V_{C_q}, E_{C_q})$:

- The new vertices and edges in Γ_q are as follows:
  - the set of new vertices are $\bigcup_{i=0}^{p-1} \{ [i, C_q]', [i, C_q]'' \}$;
  - for each $[i, C_q]' \in \{ [0, C_q]', [1, C_q]', \ldots, [p-1, C_q]' \}$ and $[j, C_q]'' \in \{ [0, C_q]'', [1, C_q]'', \ldots, [p-1, C_q]'' \}$, there is an edge $[i, C_q]' \xrightarrow{x} [j, C_q]''$. where $x =_p j - i$.
- An incoming edge $u' \xrightarrow{x} u$ to C_q with $u' \notin V_{C_q}$ and $u \in V_{C_q}$ is mapped by $T_{\text{cycle-to-gadget}}$ to the following set of edges:

Case	Corresponding set of second-type edges	Reference
$u \in [j, C_q]$	$u' \xrightarrow{x \oplus_p \ell \ominus_p j} [\ell, C_q]'$ for each $\ell \in \{0, 1, \ldots, p-1\}$	$(\mathbf{C_{q,j,x,u'}})'$

- An outgoing edge $u \xrightarrow{x} u'$ from C_q with $u' \notin C_q$ and $u \in C_q$ is mapped by $T_{\text{cycle-to-gadget}}$ as shown below:

Case	Corresponding set of second-type edges	Reference
$u \in [j, C_q]$	$[\ell, C_q]'' \xrightarrow{x \oplus_p j \ominus_p \ell} u'$ for each $\ell \in \{0, 1, \ldots, p-1\}$	$(\mathbf{C_{q,j,x,u'}})''$

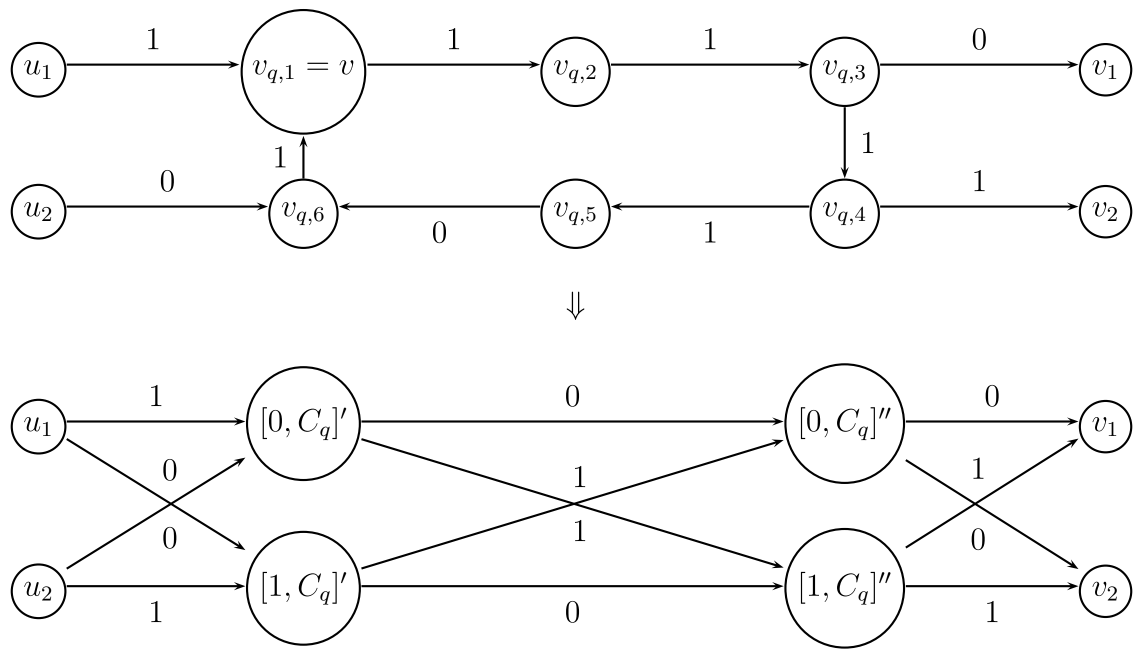

Figure 3: Gadget Γ_q for a single parity component C_q for an instance of the TR_2 problem. $[0, C_q] = \{v_{q,1}, v_{q,3}, v_{q,5}, v_{q,6}\}$, $[1, C_q] = \{v_{q,2}, v_{q,4}\}$.

- Finally, if any pair of constraints $(\mathbf{C_{q_1,j_1,x_1,u_1}})'$, $(\mathbf{C_{q_2,j_2,x_2,u_2}})'$ or $(\mathbf{C_{q_1,j_1,x_1,u_1}})''$, $(\mathbf{C_{q_2,j_2,x_2,u_2}})''$ generate the same set of edges, we remove one of the sets.

As a specific illustration, consider the graph on the top of Figure 3 for the TR_2 problem; the corresponding gadget is shown below.

The following properties of gadget edges will be useful.

Lemma 10 *The following statements are true:*

(a) $[j, C_q]'' \xrightarrow{y} u'$ *with* $u' \notin V_{C_q}$ *is in* G' *implies that, for every* $0 \le \ell < p$, $[\ell, C_q]'' \xrightarrow{y \oplus_p j \ominus_p \ell} u'$ *is also in* G'*. Moreover, the edge* $[\ell, C_q]'' \xrightarrow{y'} u'$ *is an edge among the set of edges in* $(\mathbf{C_{q,j,y' \oplus_p \ell \ominus_p j,u'}})$ *for exactly one* $0 \le j < p$*.*

(b) $u' \xrightarrow{y} [j, C_q]'$ *with* $u' \notin V_{C_q}$ *is in* G' *implies that, for every* $0 \le \ell < p$, $u' \xrightarrow{y \oplus_p \ell \ominus_p j} [\ell, C_q]'$ *is also in* G'*. Moreover, the edge* $u' \xrightarrow{y'} [\ell, C_q]'$ *is an edge among the set of edges in* $(\mathbf{C_{q,j,y' \ominus_p \ell \oplus_p j,u'}})$ *for exactly one* $0 \le j < p$*.*

Proof.
(a) Suppose that the edge $[j, C_q]'' \xrightarrow{y} u'$ was introduced by the the edge $w \xrightarrow{x} u'$ of G with

$w \in [t, C_q]$ under the constraint $(\mathbf{C_{q,t,x,u'}})''$. Thus, $y =_p x + t - j$. The same constraint also generated the edge $[\ell, C_q]'' \xrightarrow{y'} u'$ where $y' =_p x + t - \ell =_p y + j - \ell$.

For the second part, note that two constraints $(\mathbf{C_{q,j_1,y' \oplus_p \ell \ominus_p j_1,u'}})$ and $(\mathbf{C_{q,j_2,y' \oplus_p \ell \ominus_p j_2,u'}})$ generate the same set of edges and hence one of them will be removed.

(b) Suppose that the edge $u' \xrightarrow{y} [j, C_q]'$ was introduced by the the edge $u' \xrightarrow{x} w$ of G with $w \in [t, C_q]$ under the constraint $(\mathbf{C_{q,t,x,u'}})'$. Thus, $y =_p x + j - t$. The same constraint also generated the edge $u' \xrightarrow{y'} [\ell, C_q]'$ where $y' =_p x + \ell - t =_p y + \ell - j$.

For the second part, note that two constraints $(\mathbf{C_{q,j_1,y' \ominus_p \ell \oplus_p j_1,u'}})$ and $(\mathbf{C_{q,j_2,y' \ominus_p \ell \oplus_p j_2,u'}})$ generate the same set of edges and hence one of them will be removed. ❑

We need to verify Invariants **(P1)**–**(P4)**. **(P1)** obviously holds. To verify **(P2)**, we need to prove the following lemma.

Lemma 11 *Consider any valid solution for TR_p on G' that violates Invariant* **(P2)** *for a set of $(\mathbf{C_{q,j,x,u}})'$ or $(\mathbf{C_{q,j,x,u}})''$. Then we can compute in polynomial time another solution for BTR on G' that does not violate* **(P2)** *for any $(\mathbf{C_{q,j,x,u}})'$ or $(\mathbf{C_{q,j,x,u}})''$ and uses no more edges.*

Proof. Let us first verify the result when the constraint is violated for a set of $(\mathbf{C_{r,a,z,v}})''$ for various a, v, z and r. Select a specific $(\mathbf{C_{q,j,x,u}})''$ from this set. We will show how to add and delete edges such that $(\mathbf{C_{q,j,x,u}})''$ will cease to violate the constraint but the number of remaining $(\mathbf{C_{r,a,z,v}})''$ that violate the constraint will not increase; iterating the process will ensure that all violations of this type have been removed. Let $P = \{[\ell, C_q]'' \xrightarrow{x \oplus_p j \ominus_p \ell} u\} \mid \ell \in \{0, 1, \ldots, p-1\}$ be the set of edges specified by this constraint $(\mathbf{C_{q,j,x,u}})''$. Since this constraint is violated, there exists a pair of indices i and r such that $[i, C_q]'' \xrightarrow{x \oplus_p j \ominus_p i} u$ is in the solution but $[r, C_q]'' \xrightarrow{x \oplus_p j \ominus_p r} u$ is not. But the solution must contain the path $[r, C_q]'' \overset{x \oplus_p j \ominus_p r}{\Longrightarrow} u$. Let the path be $[r, C_q]'' \xrightarrow{y} w \overset{x \oplus_p j \ominus_p r \ominus_p y}{\Longrightarrow} u$. By Lemma 10(a) the existence of the edge $[r, C_q]'' \xrightarrow{y} w$ implies the existence of the edge $[i, C_q]'' \xrightarrow{r \ominus_p i \oplus_p y} w$. Consider adding the edge $[i, C_q]'' \xrightarrow{r \ominus_p i \oplus_p y} w$ to the solution if it is not already there. Then, the path $[i, C_q]'' \xrightarrow{r \ominus_p i \oplus_p y} w \overset{x \oplus_p j \ominus_p r \ominus_p y}{\Longrightarrow} u$ is of parity $r - i + y + x + j - r - y =_p x + j - i$ and thus the edge $[i, C_q]'' \xrightarrow{x \oplus_p j \ominus_p i} u$ can be deleted. To see why the number of remaining $(\mathbf{C_{r,a,z,v}})''$ that violate the constraint did not increase, suppose that we indeed added the edge $[i, C_q]'' \xrightarrow{r \ominus_p i \oplus_p y} w$ to the solution. By Lemma 10(a), the edge $[i, C_q]'' \xrightarrow{r \ominus_p i \oplus_p y} w$ is an edge among the set of edges in $(\mathbf{C_{q,j,y''}})$ for exactly one $0 \le j < p$ where $y'' =_p y + r - i + i - j =_p y + r - j$. But the constraint $(\mathbf{C_{q,j,y''}})$ also contains the edge $[r, C_q]'' \xrightarrow{y} w$ which was selected but contains the edge $[i, C_q]'' \xrightarrow{r \ominus_p i \oplus_p y} w$ which was not selected and thus the constraint $(\mathbf{C_{q,j,y''}})$ was already violated.

The case when the constraint is violated for a set of $(\mathbf{C_{r,a,z,v}})'$ for various r, a, z and v is similar. ❑

It now remains to verify the invariants **(P3)** and **(P4)** By induction on the number of components, it suffices to verify **(P3)** and **(P4)** when there is one component of single parity, say C.

To verify **(P3)**, one must consider the following cases.

(I) $u \overset{x}{\Rightarrow} w$ is in G when $u \in C$ and $w \notin C$. Suppose that u' is the last vertex on this path that belongs to C. Thus the path is of the form $u \overset{x_1}{\Rightarrow} u' \overset{x_2}{\to} w' \overset{x_3}{\Rightarrow} w$ with $x =_p x_1 + x_2 + x_3$. Suppose that $u \in [r, C]$ and thus $u' \in [s, C]$ where $s =_p r + x_1$. $E_{\text{opt}}(G')$ contains the path $[s, C]'' \overset{x_2}{\Rightarrow} w' \overset{x_3}{\Rightarrow} w$. Suppose that $T_{\text{gadget-to-cycle}}$ translated this path to a path $u'' \overset{x_2 \oplus_p s \ominus_p t}{\Longrightarrow} w' \overset{x_3}{\Rightarrow} w$ for some $u'' \in [t, C]$. Then the path $u \overset{x_1}{\Rightarrow} u' \overset{t \ominus_p s}{\Longrightarrow} u'' \overset{x_2 \oplus_p s \ominus_p t}{\Longrightarrow} w' \overset{x_3}{\Rightarrow} w$ is of parity x.

(II) $w \overset{x}{\Rightarrow} u$ is in G when $u \in C$ and $w \notin C$. Similar to **(I)**.

(III) $u \overset{x}{\Rightarrow} w$ is in G when $u, w \notin C$ but the path contains at least one vertex in C. Let $u \overset{x_1}{\Rightarrow} u' \overset{x_2}{\Rightarrow} v' \overset{x_3}{\Rightarrow} v$ where u' and v' are the first and the last vertices that belong to C. But, then $u \overset{x_1}{\Rightarrow} u'$ and $v' \overset{x_3}{\Rightarrow} v$ exist in G_{approx} by **(I)** and **(II)**, respectively, and $u' \overset{x_2}{\Rightarrow} v'$ exist in G_{approx} because of the gadget edges.

We first do the following polynomial-time local transformations before verifying **(P4)**. Using Lemma 11 we first change $G_{\min}$ to ensure that Invariant **(P2)** is not violated for any $(\mathbf{C_{q,j,x,u}})'$ or $(\mathbf{C_{q,j,x,u}})''$ without increasing the number of edges. Then, we map this new $G_{\min}$ using $T_{\text{gadget-to-cycle}}$ to a solution of G that uses no more edges than before. Since all the gadget edges are selected in any valid solution for G', to verify **(P4)** one needs to consider the following cases.

(I) $[i, C]'' \overset{x}{\to} w$ is in G' for $w \notin \bigcup_{i=0}^{p-1} \{\, [i, C_q]', [i, C_q]'' \,\}$. By Lemma 10(a) this implies that $E_{\text{opt}}(G)$ contains $u \overset{x \oplus_p i \ominus_p j}{\Longrightarrow} w$ for some $u \in [j, C]$. Suppose that this path is of the form $u \overset{y_1}{\Rightarrow} w' \overset{y_2}{\Rightarrow} w'' \overset{x \oplus_p i \ominus_p j \ominus_p y_1 \ominus_p y_2}{\Longrightarrow} w$ where w' is the last vertex on the path that belongs to C. By Lemma 9 $w' \in [j \oplus_p y_1, C]$. Thus, the path $w' \overset{y_2}{\Longrightarrow} w'' \overset{x \oplus_p i \ominus_p j \ominus_p y_1 \ominus_p y_2}{\Longrightarrow} w$ in $E_{\text{opt}}(G)$ translates to the path $[j \oplus_p y_1, C]'' \overset{y_2}{\Longrightarrow} w'' \overset{x \oplus_p i \ominus_p j \ominus_p y_1 \ominus_p y_2}{\Longrightarrow} w$. By Lemma 10(a) the edge $[j \oplus_p y_1, C]'' \overset{y_2}{\Longrightarrow} w''$ implies that the edge $[i, C]'' \overset{j \oplus_p y_1 \ominus_p i \oplus_p y_2}{\Longrightarrow} w''$ also exists and, by Invariant **(P2)**, selected in $G_{\min}$. Then the path $[i, C]'' \overset{j \oplus_p y_1 \ominus_p i \oplus_p y_2}{\Longrightarrow} w'' \overset{x \oplus_p i \ominus_p j \ominus_p y_1 \ominus_p y_2}{\Longrightarrow} w$ is of parity x.

(II) $w \overset{x}{\to} [i, C]$ is in G' for $w \notin \bigcup_{i=0}^{p-1} \{\, [i, C_q]', [i, C_q]'' \,\}$. Similar to **(I)**.

(III) $w_1 \overset{x_1}{\to} [i, C] \overset{j \ominus_p i}{\to} [j, C] \overset{x_2}{\to} w_2$ is in G' for $w_1, w_2 \notin \bigcup_{i=0}^{p-1} \{\, [i, C_q]', [i, C_q]'' \,\}$. $w_1 \overset{x_1}{\Rightarrow} [i, C]$

and $[j, C] \overset{x_2}{\Rightarrow} w_4$ exist in $G_{\min}$ by **(II)** and **(I)**, respectively, and $[i, C] \overset{j \ominus_p i}{\rightarrow} [j, C]$ is provided by one of the gadget edges for C.

Acknowledgments

The second author would like to thank Samir Khuller for pointing out to him that the results in reference [9] provided a 2-approximation for TR_1.

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

[3] Y. Chu and T. Liu. *On the shortest arborescence of a directed graph*, Scientia Sinica, 4, pp. 1396-1400, 1965.

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

[5] B. DasGupta, G. A. Enciso, E. D. Sontag and Y. Zhang. *Algorithmic and Complexity Results for Decompositions of Biological Networks into Monotone Subsystems*, arXiv q-bio.MN/0509040, available from `http://arxiv.org/abs/q-bio/0509040`

[6] R. Desikan, R. Griffiths, J. Hancock and S. Neill. *A new role for an old enzyme: nitrate reductase-mediated nitric oxide generation is required for abscisic acid-induced stomatal closure in Arabidopsis thaliana*, Proc. Natl. Acad. Sci. USA 99, 16314-16318, 2002.

[7] J. Edmonds. *Optimum Branchings*, Mathematics and the Decision Sciences, Part 1, G. B. Dantzig and A. F. Veinott Jr. (eds.), Amer. Math. Soc. Lectures Appl. Math., 11, pp. 335-345, 1968.

[8] C. P. Fall, E. S. Marland, J. M. Wagner and J. J. Tyson. Computational Cell Biology, New York: Springer, 2002.

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

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

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

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

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

[14] S. Khuller, B. Raghavachari and N. Young. *On strongly connected digraphs with bounded cycle length*, UMIACS-TR-94-10/CS-TR-3212, January 1994.

[15] 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.

[16] E. Lawler. *Combinatorial Optimization: networks and matroids*, Dover Publications, Inc., 2000.

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

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

[19] S. Li, S. M. Assmann and R. Albert. *Predicting essential components of signal transduction networks: a Boolean model of guard cell signaling*, preprint, 2005.

[20] A. C. Mustilli, S. Merlot, A. Vavasseur, F. Fenzi and J. Giraudat. *Arabidopsis OST1 protein kinase mediates the regulation of stomatal aperture by abscisic acid and acts upstream of reactive oxygen species production*, Plant Cell 14, 3089-3099, 2002.

[21] S. Pandey and S. M. Assmann. *The Arabidopsis putative G protein-coupled receptor GCR1 interacts with the G protein alpha subunit GPA1 and regulates abscisic acid signaling*, Plant Cell 16, 1616-1632, 2004.

[22] E.D. Sontag. *Some new directions in control theory inspired by systems biology*, Systems Biology 1, 9-18, 2004.

[23] R. Tarjan. *Finding optimum branchings*, Networks, 7, pp. 25-35, 1977.