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

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

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

November 23, 2006

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 have 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 *fixed* prime $p > 1$.

*Partly supported by a Sloan Research Fellowship in Science and Technology.
†Partly supported by NSF grants CCR-0296041, CCR-0206795, CCR-0208749 and IIS-0346973.
‡Partly supported by NSF grants EIA 0205116 and DMS-0504557.

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 *fixed* 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 $u \stackrel{x}{\rightarrow} v$ 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')[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.

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

[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').

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 the G protein 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 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.

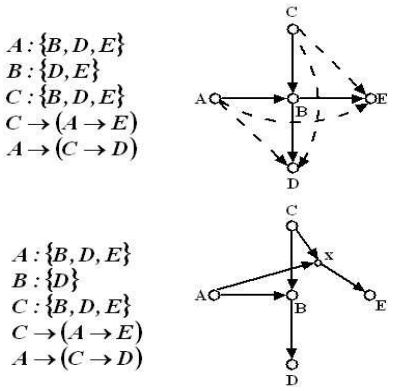

Figure 1: Illustration of the graph synthesis process in two cases.

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

A given strongly connected directed graph $G = (V, E)$ on n vertices has a Hamiltonian cycle if and only if an instance of the TR_1 problem with G as input and $E_{\text{critical}} = \emptyset$ has an optimal solution with exactly n edges. Thus TR_p is NP-complete since TR_1 includes the problem of finding directed Hamiltonian cycle in a graph. The TR_1 problem with $E_{\text{critical}} = \emptyset$ was called the *minimum equivalent digraph* (MED) problem by previous researchers. 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 $E_{\text{critical}} = \emptyset$ in the following manner: given an instance of TR_1, set the weight of every edge in E_{critical} to zero, set the weights of all other edges to 1 and run the 2-approximation algorithm for the weighted version of the MED problem.

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 a more general version of the problem is considered.
- In Section 7.3 we provide a 1.78-approximation for TR_1 and a $2 + o(1)$-approximation for TR_p for any *fixed* 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 *fixed* 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 via 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)

Here we show that the TR_p problem and in fact a more general version of it, which we refer to as the "grouped TR_p problem with c-limited overlap", can be solved in polynomial time if the input graph is a DAG and c is a constant. We formally define this general version of the problem below.

Definition 1 *A grouped TR_p problem with c-limited overlap on a graph G is defined as follows.*

- *The input graph $G = (V, E)$ is an instance of the TR_p problem.*
- *Non-empty distinct groups $E_1, E_2, \ldots, E_t$ of E are given such that:*

- $\cup_{i=1}^{t} E_i = E$;
- $E_i \neq E_j$ *if* $i \neq j$;
- $E_i \neq \emptyset$ *for* $i \in \{1, 2, \ldots, t\}$;
- *the groups have overlaps "limited" by* c, i.e., *there is a partition* $I_1, I_2, \ldots, I_q$ *of the set of indices* $\{1, 2, \ldots, t\}$ *such that*
  - *any two groups with indices from different sets of the partition do not intersect,* i.e., *for all* i, j, x, y *with* $i \neq j$ *and* $x \in I_i, y \in I_j$, $E_x \cap E_y = \emptyset$.
  - *the total number of edges in the groups with indices in any one set of the partition is at most* c, i.e., *for all* i, $|\cup_{x \in I_i} E_x| \leq c$.

- *For each* $i \in \{1, 2, \ldots, t\}$, *a valid solution* must *contain* all *the edges in* E_i *or* none *of the edges in* E_i.
- *We are also given a set of critical edges* $E_{\text{critical}} \subseteq E$. *Any solution must contain every edge in* E_{critical}.

Example 1 (Illustration of the grouped TR$_p$ problem with 6-limited overlap) *Suppose that* $G = (V, E)$ *is an instance of the grouped TR$_p$ problem with* $E = \{e_1, e_2, e_3, \ldots, e_9\}$, $t = 4$, $E_1 = \{e_1, e_2, e_4, e_5, e_9\}$, $E_2 = \{e_1, e_2, e_3\}$, $E_3 = \{e_6, e_7\}$, $E_4 = \{e_6, e_7, e_8\}$, $q = 2$, $I_1 = \{1, 2\}$, $I_2 = \{3, 4\}$. *Then, this is an instance of a TR$_p$ problem with 6-limited overlap on G since both $|E_1 \cup E_2|$ and $|E_3 \cup E_4|$ are no more than 6 and no edge from $E_1 \cup E_2$ belongs to $E_3 \cup E_4$ (and vice versa). Moreover, by definition of grouped TR$_p$ problem with c-limited overlap, a valid solution for this grouped TR$_p$ problem on G is required to use:*

- *all or none of the edges in* E_1;
- *all or none of the edges in* E_2;
- *all or none of the edges in* E_3;
- *all or none of the edges in* E_4.

Note that the TR$_p$ problem is in fact an instance of the grouped TR$_p$ problem with 1-limited overlap when each group of E contains *exactly one edge* and the partition of the set of indices consists of singleton sets.

Lemma 2 *The grouped TR$_p$ problem with c-limited overlap can be solved in polynomial time when the given graph G is a DAG and c is a constant.*

Proof. The algorithm shown below can be used; it essentially generalizes the ideas in [1]. The comments for various steps in the algorithm begin with //.

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
start with the graph $G' = (V, E')$ with $E' = E$ and every edge in E' marked as "not selected"
for $i = n - 1, n - 2, \ldots, 1$
  for $j = n, n - 1, n - 2, \ldots, i + 1$

for every edge $v_i \xrightarrow{y} v_j \in E$ (in arbitrary order of y) // the edge is now "examined"
if $v_i \xrightarrow{y} v_j \in E_{\text{critical}}$ **then**
mark $v_i \xrightarrow{y} v_j$ as "selected"
else if there exists another path $v_i \overset{y}{\Rightarrow} v_j$ in G'
then $E' = E' \setminus \{v_i \xrightarrow{y} v_j\}$ // the edge is redundant and thus deleted from E'
else mark $v_i \xrightarrow{y} v_j$ as "selected"
// Note that now E' contains only edges marked as "selected"
// first partition the edges in E' based on their memberships in the partitions indexed by $I_1, I_2, \ldots, I_q$
let $e_1, e_2, \ldots, e_{|E'|}$ be the edges in E'
let $J_1, J_2, \ldots, J_r$ be a partition of the indices $\{1, 2, \ldots, |E'|\}$ such that
for all $\ell \neq \ell'$, both ℓ and ℓ' belongs to the same set J_s of the partition
if and only if both e_ℓ and $e_{\ell'}$ are in $\cup_{x \in I_t} E_x$ for some t
// Notice that due to c-limitedness $|\cup_{x:\, \ell \in J_s \wedge e_\ell \in E_x} E_x| \leq c$ for any set J_s of the partition;
// thus, for any set J_s of the partition, the numbers of edge groups E_x such that $\ell \in J_s$ and $e_\ell \in E_x$ is at
// thus, the following step takes constant time per J_s by a trivial exhaustive method;
// the most obvious exhaustive way takes $O(2^{2^c}) = O(1)$ time per J_s
$E'' = \emptyset$
for each $s \in \{1, 2, \ldots, r\}$
find a minimum collection of groups from $\{E_x : \ell \in J_s \wedge e_\ell \in E_x\}$ that includes the edges in J_s
add these edges to E''
output the final $G' = (V, E'')$ as the solution

For ease of understanding of the reader, we first work through a small example illustrating the above algorithm before continuing with the formal proof.

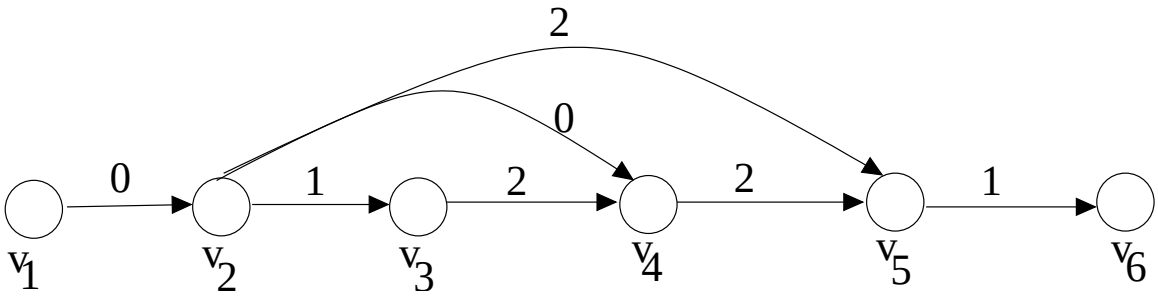

Figure 2: Illustration of the algorithm for the grouped TR_3 problem with 6-limited overlap on a DAG.

Example 2 (Illustration of the algorithm for the grouped TR_3 problem with 6-limited overlap) *Consider the DAG shown in Figure 2 with* $p = 3$, $t = 3$, $E_1 = \{v_5 \to v_6, v_4 \to v_5, v_3 \to v_4, v_2 \to v_3, v_2 \to v_5\}, E_2 = \{v_5 \to v_6, v_2 \to v_4\}, E_3 = \{v_1 \to v_2\}, E_{\text{critical}} = \{v_1 \to v_2\}$, $q = 2$, $I_1 = \{1, 2\}, I_2 = \{3\}$. *The instance is* 6*-limited since both* $|E_1 \cup E_2|$ *and* $|E_3|$ *are no more than* 6. *The successive steps in the algorithm can be summarized as follows:*

- *mark all the edges as "not selected";*
- *examine* $v_5 \to v_6$*, mark the edge* $v_5 \to v_6$ *as selected;*

- *examine* $v_4 \rightarrow v_5$*, mark the edge* $v_4 \rightarrow v_5$*, as selected;*
- *examine* $v_3 \rightarrow v_4$*, mark the edge* $v_3 \rightarrow v_4$*, as selected;*
- *examine* $v_2 \rightarrow v_5$*, remove this edge from* E' *because of the path* $v_2 \xrightarrow{0} v_4 \xrightarrow{2} v_5$*;*
- *examine* $v_2 \rightarrow v_4$*, remove this edge from* E' *because of the path* $v_2 \xrightarrow{1} v_3 \xrightarrow{2} v_4$*;*
- *examine* $v_2 \rightarrow v_3$*, mark the edge* $v_2 \rightarrow v_3$*, as selected;*
- *examine* $v_1 \rightarrow v_2$*, mark the edge* $v_1 \rightarrow v_2$*, as selected;*
- *now* $E' = \{e_1 = v_5 \rightarrow v_6, e_2 = v_4 \rightarrow v_5, e_3 = v_3 \rightarrow v_4, e_4 = v_2 \rightarrow v_3, e_5 = v_1 \rightarrow v_2, \}$*, and thus* $J_1 = \{1, 2, 3, 4\}$*,* $J_2 = \{5\}$*;*
- *a minimum collection of groups from* $E_1 \cup E_2$ *that cover* $\{e_1, e_2, e_3, e_4\}$ *is* E_1*;*
- *a minimum collection of groups from* E_3 *that cover* $\{e_5\}$ *is* E_3*;*
- *the final solution* $E'' = E_1 \cup E_3$*.*

The algorithm can be obviously implemented in polynomial time provided we can check, for any given y, i and j, if there exists a path $v_i \stackrel{y}{\Rightarrow} v_j$. This can be done via a straightforward dynamic programming by using the following two well-known observations on topological sorting of any DAG:

($\star$) G does not contain a path $v_i \Rightarrow v_j$ if $i > j$.

($\star\star$) If G has a path $v_i \Rightarrow v_j$ for some $i \leq j$ then any intermediate vertex v_p of the path satisfies $i < p < j$.

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). Furthermore, either all the edges or none of the edges in any group of edges are selected. Thus the algorithm returns a valid solution.

Now we continue with a formal proof of optimality of our algorithm. Notice that an edge belongs to E' if and only if it was marked as selected. To prove our algorithm is optimal, it suffices to show:

any valid solution must contain the edges in E'

since we have an exact algorithm to select a minimum number of groups to include these edges. Note that the order in which pairs of vertices are examined by the nested loops in the algorithm are:

$$\begin{array}{ll} & (v_{n-1}, v_n) \\ \text{then} & (v_{n-2}, v_n), (v_{n-2}, v_{n-1}) \\ \text{then} & (v_{n-3}, v_n), (v_{n-3}, v_{n-1}), (v_{n-3}, v_{n-2}) \\ \text{then} & (v_{n-4}, v_n), (v_{n-4}, v_{n-1}), (v_{n-4}, v_{n-2}), (v_{n-4}, v_{n-3}) \\ \vdots & \end{array}$$

and an edge $v_i \stackrel{y}{\rightarrow} v_j$ is removed from E' only when it is examined and an alternate path $v_i \stackrel{y}{\Rightarrow} v_j$ exists in G. Thus, using Observations ($\star$) and ($\star\star$) the following holds:

($\star\star\star$) let $v_i \xrightarrow{y} v_j$ be an edge in G such that an alternate path $v_i \stackrel{y}{\Rightarrow} v_j$ exists in G; then when the edge $v_i \xrightarrow{y} v_j$ is examined by the algorithm then some (not necessarily the same as before) alternate path $v_i \stackrel{y}{\Rightarrow} v_j$ exists in G'.

Now we show that any valid solution must contain all the edges in E'. Suppose, for the sake of contradiction, that this is not the case and there is a valid solution that does not contain at least one edge in E'. In the rest of the proof, we concentrate on *this valid solution*. As we just said, there exists at least one edge that belongs to E' but does not belong to the valid solution. Among all such edges, let e be the *first such edge* examined during the execution of the algorithm. Now we have the following subcases:

Case 1.1. The reason why $e = v_i \xrightarrow{y} v_j$ belongs to E' is because of the following line of code in our algorithm:

if $v_i \xrightarrow{y} v_j \in E_{\text{critical}}$ **then**
mark $v_i \xrightarrow{y} v_j$ *as "selected"*

This implies that $e \in E_{\text{critical}}$. But then e must belong to any valid solution, a contradiction.

Case 1.2. $e \notin E_{\text{critical}}$. The reason why $e = v_i \xrightarrow{y} v_j$ belongs to E' is because of the last line of the following lines of code in our algorithm:

else if *there exists another path* $v_i \stackrel{y}{\Rightarrow} v_j$ *in* G'
then $E' = E' \setminus \{v_i \xrightarrow{y} v_j\}$ *// the edge is redundant and thus deleted from* E'
else *mark* $v_i \xrightarrow{y} v_j$ *as "selected"*

The above last line of code was executed because there existed no other path $v_i \stackrel{y}{\Rightarrow} v_j$ in G' when the edge $v_i \stackrel{y}{\Rightarrow} v_j$ was examined. Via Observation ($\star\star\star$) it follows that there exists no other path $v_i \stackrel{y}{\Rightarrow} v_j$ in G either. Thus, any valid solution must contain the edge $e = v_i \xrightarrow{y} v_j$, a contradiction.

❑

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 3 *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 $u, v \in V_C$, $u \stackrel{x}{\Rightarrow} v$ exists in C for every $x \in \{0, 1, 2, \ldots, p-1\}$.

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

Lemma 4
(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 times, we therefore have a cycle $u \stackrel{y}{\Rightarrow} u$ for every $y \in \{0, 1, 2, \ldots, p-1\}$. If the cycle $u \stackrel{x}{\Rightarrow} u$ is not simple, then we have the following two cases:

Case 1: $u \stackrel{x}{\Rightarrow} u$ contains a simple cycle of non-zero parity.

Case 2: $u \stackrel{x}{\Rightarrow} u$ contains no simple cycle of non-zero parity. Start following the cycle edges from u. Since $u \stackrel{x}{\Rightarrow} u$ is not simple, we will detect a self-intersection at some point. That is, our traversal will be of the type

$$u \rightarrow \ldots \rightarrow v' \rightarrow \mathbf{v} \rightarrow \ldots \rightarrow \mathbf{v} \rightarrow v'' \rightarrow \ldots$$

Since the cycle $\mathbf{v} \rightarrow \ldots \rightarrow \mathbf{v}$ is of zero parity, its removal will result in a cycle of the form

$$u \rightarrow \ldots \rightarrow v' \rightarrow v'' \rightarrow \ldots$$

which has the same parity as the original cycle. We can continue with such "removals" until we have a simple cycle.

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** $i \xrightarrow{x} j \in E_C$ **then** $N(i,j,0,x) = 1$ and $P(i,j,k,x) = i \xrightarrow{x} j$
    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 $N(i,i,n,x)$ for each $i \in \{1,2,\ldots,n\}$ and each $x \in \{1,2,\ldots,p-1\}$. Moreover, such a simple non-zero parity cycle, if it exists, can be found from the corresponding $P(i,i,n,x)$'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 which essentially states that the number of edges in any optimal solution for TR_p grows only *linearly* with the number of vertices in the given graph *irrespective of* p.

Corollary 5 *Consider the TR_p problem, when p is 1 or a prime number, on a strongly connected graph $G = (V,E)$ with $|E_{\text{critical}}| = \emptyset$. Then, $|V| \leq OPT(G) \leq 3|V| - 2$.*

Proof. For $p = 1$, it is well-known that an optimal solution of TR_1 satisfies $|V| \leq OPT(G) \leq 2|V| - 2$ (see the first paragraph of Section 1.2 in [13]). Now consider the case of $p > 1$. By Lemma 4, there are only two cases to consider:

Case 1. G is a single parity component. Then, an optimal solution of the TR_1 problem on a graph G, *i.e.*, an optimal solution of the TR_1 problem on G with all edge labels of G being set to zero, provides a valid solution of the TR_p problem on G. Thus, in this case, $|V| \leq OPT(G) \leq 2|V| - 2$.

Case 2. G is a multiple parity component. By Lemma 4 we can find a simple cycle X of non-zero parity in G. Obviously, X contains at most $|V|$ edges. Consider an optimal solution of the TR_1 problem on G and add those edges of X to this solution that are not already there. By the proof of Lemma 4, this solution contains a path of every parity from every vertex to every other vertex. In this case, $|V| \leq OPT(G) \leq 3|V| - 2$. ❑

7.2 Solving TR_p for a Strongly Connected Component

The main result of this subsection is as follows.

Theorem 6 *Let the given graph G be strongly connected. Then, we can design a $2 + o(1)$-approximation algorithm for TR_p when $p > 1$ is a prime and a 1.78-approximation algorithm for 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 TR_1 (and, thus, obviously a special case 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:

- p is either 1 or a prime number throughout this section unless otherwise stated explicitly.
- 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:

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

Proposition 1 *Let G be a single parity strongly connected graph that is an input instance of TR_p. Then a valid solution of TR_p on G is a valid solution of TR_1 on G and vice versa,* i.e.*, the two problems TR_p on G and TR_1 on G are equivalent.*

Proof. Obviously, a valid solution of TR_p on G is also a valid solution of TR_1 on G. Since G is of single parity, all the paths from a vertex u to another vertex v have the same parity and a valid solution of TR_1 of G includes at least one such path. Thus, a valid solution of TR_1 on G is a valid solution of TR_p on G as well. ❑

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

This algorithm for TR_1 with no critical edges (*i.e.*, for the MED problem), which we refer to as Algorithm A1, works as follows.[2] 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; see Figure 3 for an illustration. The algorithm proceeds as follows:

[2] We will use the cycle contraction algorithm in a "black box" fashion, so readers not interested in the details of the algorithm can skip the finer details of the algorithm and proceed directly to the performance bounds.

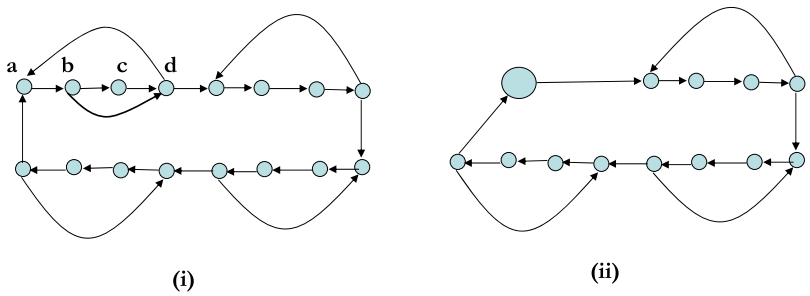

Figure 3: Illustration of a cycle contraction. (i) shows the original graph and (ii) shows the graph after the cycle $a \to b \to c \to d \to a$ is contracted.

Select a constant $k > 3$
for $i = k, k-1, \ldots, 4$
  while the graph contains a cycle $\mathcal{C}$ of at least i edges
    contract the cycle $\mathcal{C}$ and selects the edges in $\mathcal{C}$
// Now the graph contains no cycle of more than 3 edges //
Use the exact algorithm for MED from [14] on the remaining graph and
  select the edges in this exact solution

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 an instance of TR_1 with $E_{\text{critical}} = \emptyset$ (and thus, by Proposition 1, if G is an instance of TR_p, 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. Given the graph $G = (V, E)$, define the weight[3] $\mathtt{wt}(e) = 0$ of every edge in $e \in E_{\text{critical}}$ and the weights of all other edges to be 1. Then, the algorithm is as follows:

- Select an arbitrary vertex v.
- In the *first stage*, find a minimum weight spanning arborescence[4] T of minimum weight of G, say rooted at vertex v.
- In the second stage, set $\mathtt{wt}(e) = 0$ 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.*,

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

[4] 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.

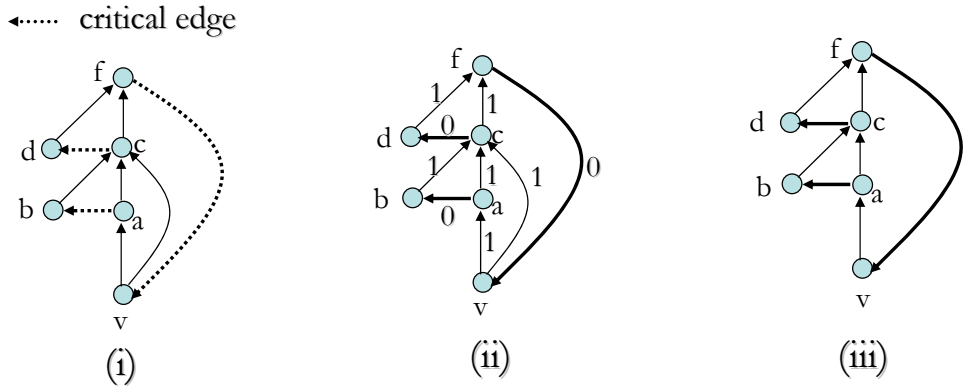

Figure 4: Illustration of the spanning arborescence algorithm for TR_1. (i) shows the original graph. (ii) shows the graph in which $\mathtt{wt}(e) = 0$ for every critical edge in e and $\mathtt{wt}(e) = 1$ otherwise. (iii) shows the final solution; the edges $a \to b, b \to c, c \to d, d \to f, f \to v$ belong to the spanning arborescence whereas the edges $v \to a, a \to c, c \to f$ are the additional edges introduced by the reverse spanning arborescence.

a minimum weight spanning directed acyclic subgraph such that v has no outgoing edges and every other vertex has exactly one outgoing edge.

- 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 final solution.

See Figure 4 for an illustration of how the algorithm works. The proofs in [9] imply that:

- E_2 is a correct solution if G is an instance of TR_1 (and thus, by Proposition 1, if G is an instance of TR_p and 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. Discard all incoming edges to the root. Then the algorithm proceeds as follows. First, we select for each node, except the root, an incoming edge of minimum weight (breaking ties arbitrarily). If these edges do not give a spanning arborescence, then there must be a directed cycle $\mathcal{C}$ formed by some of these edges. Let $\delta_{\mathcal{C}}$ be the minimum of the weights of the edges in $\mathcal{C}$. We contract $\mathcal{C}$ to a "supernode" and decrease the weight of every edge $i \to k$ from vertex $i \notin \mathcal{C}$ to vertex $k \in \mathcal{C}$ by $p - \delta_{\mathcal{C}}$ where p is the weight of the unique edge in $\mathcal{C}$ that is incoming to k. The process is then repeated on the reduced graph, and continued until either we have a spanning arborescence on the remaining graph. The supernodes are then expanded in reverse order. Each time a supernode is expanded, *exactly one of its edges that would produce two incoming edges to a node is discarded.* The same algorithm with minor modification can be used to find the reverse spanning arborescence rooted to v also. Figure 5 illustrates the algorithm. Algorithm A2 can be implemented to run in $O(\min\{|E| \log |V|, |V|^2\})$ time; see [23] for details.

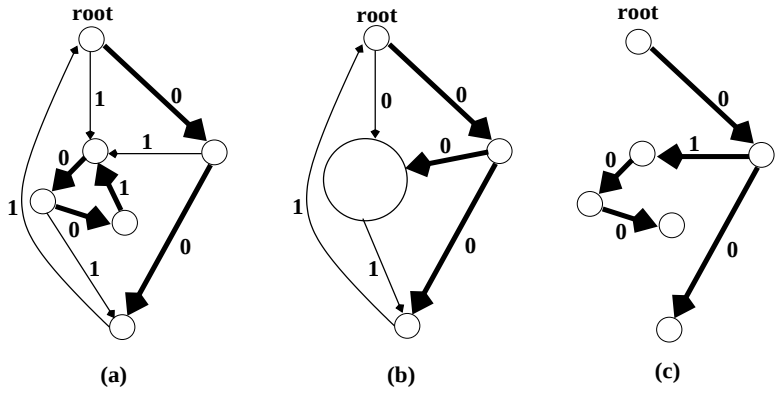

Figure 5: Illustration of the minimum weight spanning arborescence algorithm. The thick edges selected in (a) create a cycle. The cycle is replaced by a supernode and the edge weights are adjusted in (b). The thick edges selected in (b) give a spanning arborescence on the remaining graph. When the supernode is expanded for the final solution in (c), one of the edges in the cycle is discarded.

7.3 Augmenting and Combining the Two Algorithms

7.3.1 Augmenting Algorithm A1

Lemma 7 *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 4(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 involves 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 Algorithm A2

Lemma 8 *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 4(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 $\texttt{wt}(e)$ 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) $\texttt{wt}(u \to v) = 0$. We can then surely select this edge.

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

(iii) $\texttt{wt}(u \to v) = 1$ and some edge $u' \to v$ has $\texttt{wt}(u' \to v) = 0$. Then, we change $\texttt{wt}(u \to v)$ to zero which would allow us to select this edge. Let S be the set of all these edges. Since $\texttt{wt}(u' \to v) = 0$ 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

$$\rho = \min\left\{\frac{2.236\alpha + 1.618\beta}{\alpha + \beta},\ \frac{\alpha + 2\beta}{\alpha + \beta}\right\} = \min\left\{\frac{2.236 + 1.618\frac{\beta}{\alpha}}{1 + \frac{\beta}{\alpha}},\ \frac{1 + 2\frac{\beta}{\alpha}}{1 + \frac{\beta}{\alpha}}\right\}$$

There are now two cases to consider.

Case 1. $\frac{\beta}{\alpha} \leq \frac{1.136}{0.382}$. Since $\frac{1+2\frac{\beta}{\alpha}}{1+\frac{\beta}{\alpha}}$ increases with increasing $\frac{\beta}{\alpha}$,

$$\rho \leq \frac{1 + 2 \times \frac{1.136}{0.382}}{1 + \frac{1.136}{0.382}} < 1.78$$

Case 2. $\frac{\beta}{\alpha} > \frac{1.136}{0.382}$. Since $\frac{2.236+1.618\frac{\beta}{\alpha}}{1+\frac{\beta}{\alpha}}$ decreases with increasing $\frac{\beta}{\alpha}$,

$$\rho \leq \frac{2.236 + 1.618 \times \frac{1.136}{0.382}}{1 + \frac{1.136}{0.382}} < 1.78$$

$2 + o(1)$-approximation for $\mathbf{TR}_p$ 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 $|E_{\max_{\text{opt}}}(G)| \leq |E^1_{\text{opt}}(G)| = \alpha + \beta \leq |E_{\text{opt}}(G)|$, $\frac{|E_2|}{|E_{\text{opt}}(G)|} \leq 2 + o(1)$.

7.4 Approximating $\mathbf{TR}_p$ 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, how 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 $C_1 = (V_{C_1}, E_{C_1}), C_2 = (V_{C_2}, E_{C_2}), \ldots, C_m = (V_{C_m}, E_{C_m})$ being the m strongly connected components where the i^{th} component C_i contains n_i vertices $v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}$; 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 $i \neq j$;
- $\cup_{i=1}^{m} E(C_i) \subseteq E$.

By a *gadget* Γ_i for the component C_i we mean a DAG of $O(p^2)$ edges on a *new* set of $O(p)$ vertices. No two gadgets share a vertex.

Let $C_1, C_2, \ldots, C_m$ be the connected components of G in an arbitrary order and suppose that we have gadgets $\Gamma_1 = (V_{\Gamma_1}, E_{\Gamma_1}), \Gamma_2 = (V_{\Gamma_2}, E_{\Gamma_2}), \ldots, \Gamma_m = (V_{\Gamma_m}, E_{\Gamma_m})$ for the components $C_1, C_2, \ldots, C_m$, respectively. Let $E_{\text{gadget}} = \cup_{i=1}^{m} E_{\Gamma_i}$ be the set of all edges in all the gadgets. For notational convenience let $G^0 = (V^0, E^0)$ be a graph identical to the given graph G; thus $V^0 = V$ and $E^0 = E$. Also, for notational convenience define $e^0 = \{e\}$ for every edge $e \in E$. 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^{th} iteration is G^{i-1}.
  - The component C_i in G^{i-1} is replaced by its corresponding gadget Γ_i.
  - The edge replacement mapping is as follows:
    - An edge e in G^{i-1} with *both* end-points not in C_i stays the same, *i.e.*, the replacement of the edge is the edge itself.

* If C_i is a multiple parity component then we do the following.
  - For an incoming edge $u \xrightarrow{x} v$ in G^{i-1} from a vertex u not in C_i to a vertex v in C_i the replacement is a set of p edges (**a group of "first type" replaced edges**) $\{u \xrightarrow{0} v', u \xrightarrow{1} v', \ldots, u \xrightarrow{p-1} v'\}$ to some vertex v' in Γ_i. We will say that the edge $u \xrightarrow{x} v$ is the "corresponding edge" for these set of replaced edges.
  - Similarly, 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 a set of p edges (**a group of "first type" replaced edges**) $\{u' \xrightarrow{0} v, u' \xrightarrow{1} v, \ldots, u' \xrightarrow{p-1} v\}$ from some vertex u' in Γ_i. We will say that the edge $u \xrightarrow{x} v$ is the "corresponding edge" for these set of replaced edges.
* If C_i is a single parity component then we do the following.
  - For an incoming edge $u \xrightarrow{x} v$ in G^{i-1} from a vertex u not in C_i to a vertex v in C_i the replacement is a set of p edges (**a group of "second type" replaced edges**) $\{u \xrightarrow{0} v_0, u \xrightarrow{1} v_1, \ldots, u \xrightarrow{p-1} v_{p-1}\}$ to some vertices $v_0, v_1, \ldots, v_{p-1}$ in Γ_i. We will say that the edge $u \xrightarrow{x} v$ is the "corresponding edge" for these set of replaced edges.
  - Similarly, 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 a set of p edges (**a group of "second type" replaced edges**) $\{u_0 \xrightarrow{0} v, u_1 \xrightarrow{1} v, \ldots, u_{p-1} \xrightarrow{p-1} v\}$ from some vertices $u_0, u_1, \ldots, u_{p-1}$ in Γ_i. We will say that the edge $u \xrightarrow{x} v$ is the "corresponding edge" for these set of replaced edges.
* Let e^i denote the union of replacements of all the edges in e^{i-1}.

– The resultant graph at the end of the i^{th} iteration is denoted by is $G^i = (V^i, E^i)$.

- Remove identical groups of edges[5] from G^m, *i.e.*, if there are two groups of edges e^m and f^m with $e^m = f^m$, remove one of the groups. Let G' be the resulting graph.

Example 3 (Illustration of the notations in edge replacement) *Suppose that the given graph G has seven strongly connected components $C_1, C_2, \ldots, C_7$, $p = 2$, the edges e_1, e_2 connect from a vertex in C_1 to a vertex in C_2, the edge e_3 connect from a vertex in C_2 to a vertex in C_3, and let the edges $\{e_4, e_5, \ldots, e_9\}$ have both endpoints from $C_3, C_4, \ldots, C_7$,* i.e.*, for each edge $e \in \{e_4, e_5, e_6, e_7, e_8, e_9\}$, e connects two nodes u and v such that $u \in C$ and $v \in C'$ with $C, C' \in \{C_3, C_4, C_5, C_6, C_7\}$. Suppose that the replacement step for C_1 replaced the edge in e_1^0 and e_2^0 by a group of 2 second type edges α, β and α', β',respectively; the replacement step for C_2 replaced the edge α by a group of 2 first type edges γ, δ, replaced the edge β by a group of 2 second type edges κ, ν, replaced the edge α' by a group of 2 first type edges γ', δ', replaced the edge β' by a group of 2 second type edges κ', ν', and replaced the edge in e_3^1 by a group of 2 first type edges μ, θ. Then,*

$$e_1^0 = \{e_1\}, e_2^0 = \{e_2\}, e_3^0 = \{e_3\}, e_4^0 = \{e_4\}, e_5^0 = \{e_5\}, e_6^0 = \{e_6\}, e_7^0 = \{e_7\}, e_8^0 = \{e_8\}, e_9^0 = \{e_9\}$$

$$e_1^1 = \{\alpha, \beta\}, e_2^1 = \{\alpha', \beta'\}, e_3^1 = \{e_3\}, e_4^1 = \{e_4\}, e_5^1 = \{e_5\}, e_6^1 = \{e_6\}, e_7^1 = \{e_7\}, e_8^1 = \{e_8\}, e_9^1 = \{e_9\}$$

$$e_1^2 = \{\gamma, \delta, \kappa, \nu\}, e_2^2 = \{\gamma', \delta', \kappa', \nu'\}, e_3^2 = \{\mu, \theta\}, e_4^2 = \{e_4\}, e_5^2 = \{e_5\}, e_6^2 = \{e_6\}, e_7^2 = \{e_7\}, e_8^2 = \{e_8\}, e_9^2 = \{e_9\}$$

[5]Two groups e^m, f^m of edges are identical iff there is a bijection H from e^m to f^m such that, for $e \in e^m$ and $f \in f^m$, $H(e) = f$ iff e is an edge from u to v of parity x and f is an edge from u to v of parity x.

Proposition 2 *For any edge $e \in E$ and any i, $|e^i| \leq p^2$.*

Proof. Suppose that e connects the two components C_x and C_y with $x < y$. Then,

$$1 = |e^0| = |e^1| = \cdots = |e^{x-1}|$$

$$|e^x| = p \cdot |e^{x-1}| = p$$

$$p = |e^x| = |e^{x+1}| = \cdots = |e^{y-1}|$$

$$|e^y| = p \cdot |e^{y-1}| = p^2$$

$$p^2 = |e^y| = |e^{y+1}| = \cdots = |e^m|$$

❑

Proposition 3 *$G' = G^m$ is a DAG.*

Proof. Consider the graph G'' obtained by removing the edges in E_{gadget} from G'. Since an edge in G between two connected components was replaced by a set of edges between the same two connected components, G'' specifies the interconnections between strongly connected components and is therefore well-known to be a DAG (e.g., see [4]). Consider a topological ordering of the vertices of G'' in which all the vertices in V_{Γ_i} are placed consecutively in some arbitrary order for each i; this is possible since all the edges connecting two vertices in the same connected component were removed. Furthermore, since all the edges connecting two vertices in the same connected component were removed, the topological ordering of the vertices of G'' is not destroyed if the vertices in V_{Γ_i} in the above topological order are permuted in any arbitrary order as long as they are placed consecutively; thus we can rearrange the vertices in V_{Γ_i}, for each i, such that they are placed consecutively in an order corresponding to the topological sorting of the vertices of the DAG Γ_i. Now, if we put back the missing edges of E_{Γ_i}, for each i, in this topological sorting of G'' we arrive at a topological sorting of G'. ❑

Now, we show that the graph $G' = G^m$ in fact satisfies the c-limited overlap property of Definition 1 for some constant c once we remove identical groups of edges.

Proposition 4 *Let $e_1^m, e_2^m, \ldots, e_t^m$ be the non-identical groups of edges which make up $E' = \cup_{i=1}^t e_i^m$. Then, $e_1^m, e_2^m, \ldots, e_t^m$ define a c-limited partition of E' for some constant c.*

Proof. Note that two groups of edges e^m and f^m have a non-empty intersection only if both $e \in e^0$ and $f \in f^0$ are edges between the same pair of connected components in the given graph G. Since each connected component is replaced by $O(p)$ vertices in the gadget, there are only $O(p^2) = O(1)$ edges between these two components. Each edge in e or f is one of these $O(p^2) = O(1)$ edges. ❑

Example 4 (Illustration of the proof of Proposition 4) *Consider Example 3 again. Since e_1 and e_2 are the only two edges that connect the same pair of connected components C_1 and C_2, $e_1^2 = \{\gamma, \delta, \kappa, \nu\}$ can have a non-empty intersection with $e_2^2 = \{\gamma', \delta', \kappa', \nu'\}$ only and vice versa.*

Abusing terminologies slightly, we now state what we mean by a "valid solution" for G^i.

Definition 9 *By a "valid solution" for G^i, for $i \in \{0, 1, 2, \ldots, m\}$, we mean a valid solution of the grouped TR_p problem on G^i when the edge groups are e^i for each edge e in the given graph $G = G^0$ that connects two connected components. An "optimal solution" for G^i is a "valid solution" with a minimum number of edges.*

Example 5 (Illustration of a valid solution for G^i) *Consider Example 3 again. By a "valid solution" for $G^2 = (V^2, E^2)$ we mean a valid solution of the grouped TR_p problem for G^2 where the edge partitions of $E^2 = \{\gamma, \delta, \kappa, \nu, \gamma', \delta', \kappa', \nu', \mu, \theta, e_4, e_5, e_6, e_7, e_8, e_9\}$ are given by $\{\gamma, \delta, \kappa, \nu\}$, $\{\gamma', \delta', \kappa', \nu'\}$, $\{\mu, \theta\}$, $\{e_4\}$, $\{e_5\}$, $\{e_6\}$, $\{e_7\}$, $\{e_8\}$, $\{e_9\}$.*

Note that a "valid solution" for G^0 is obviously a valid solution of the TR_p problem on G since each group of edges in $G = G^0$ consists of one single edge. Since G' is a DAG with c-limited overlap for some constant c, we can find an "optimal solution" for G' exactly via the algorithm described in Lemma 2. Also, suppose the above procedure $T_{\mathrm{cycle-to-gadget}}$ guarantees the following invariants *for each* $i \in \{1, 2, \ldots, m\}$:

$(P1^i)$ any valid solution of TR_p for G^i *must* contain all the edges in the gadget Γ_i, and

$(P2^i)$ if an edge $u \xrightarrow{x} v$ to or from C_i was replaced by a set of first or second type edges in G^i, then any valid solution for G^i either selects *all* of these edges or selects *none* of these edges.

Notice that our definition of a valid solution for G^i in Definition 9 trivially satisfies the Invariant$(P2^i)$. As an illustration, consider the illustration of a valid solution in Example 5. A "valid solution" for G^2 must contain all or none of the edges in $\{\gamma, \delta, \kappa, \nu\}$; this obviously implies that the solution contains all or none of the set of edges in $\{\gamma, \delta\}$ and contains all or none of the set of edges in $\{\kappa, \nu\}$. Thus, we do not need to verify Invariant $(P2^i)$ at all.

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

- For $i = m, m-1, \ldots, 1$ do the following:
  - Notice that $E(i)$ is a subset of edges of G^i. We will maintain Invariant $(\star)$ described subsequently that will ensure that $E(i)$ is a "valid solution" of G^i.
  - Since $E(i)$ is a "valid solution" of G^i, it satisfies Invariants $(P1^i)$ and $(P2^i)$.
  - Replace the vertices and edges of Γ_i by the vertices and edges in an optimal solution $E_{\mathrm{opt}}(C_i)$ of C_i.
  - Replace a group of first type edges $\{u \xrightarrow{0} v', u \xrightarrow{1} v', \ldots, u \xrightarrow{p-1} v'\}$ incoming to some vertex v' in Γ_i by their "corresponding edge" $u \xrightarrow{x} v$ in G^{i-1}.
  - Replace a group of first type edges $\{u' \xrightarrow{0} v, u' \xrightarrow{1} v, \ldots, u' \xrightarrow{p-1} v\}$ outgoing from some vertex u' in Γ_i by their "corresponding edge" $u \xrightarrow{x} v$ in G^{i-1}.
  - Replace a group of second type edges $\{u \xrightarrow{0} v_0, u \xrightarrow{1} v_1, \ldots, u \xrightarrow{p-1} v_{p-1}\}$ incoming to some vertices $v_0, v_1, \ldots, v_{p-1}$ in Γ_i by their "corresponding edge" $u \xrightarrow{x} v$ in G^{i-1}.
  - Replace a group of second type edges $\{u_0 \xrightarrow{0} v, u_1 \xrightarrow{1} v, \ldots, u_{p-1} \xrightarrow{p-1} v\}$ outgoing from some vertices $v_0, v_1, \ldots, v_{p-1}$ in Γ_i. by their "corresponding edge" $u \xrightarrow{x} v$ in G^{i-1}.

– The replacement of any other edge is the edge itself.
– The resultant set of edges is denoted by $E(i-1) \subseteq E^{i-1}$.

Example 6 (Illustration of the reverse edge replacement) *Consider the same example described in Example 3. Suppose that a "valid solution" $E(2)$ for G_2 contains all the edges $e_1^2 = \{\gamma, \delta, \kappa, \nu\}$, none of the edges in $e_2^2 = \{\gamma', \delta', \kappa', \nu'\}$, none of the edges in $e_3^2 = \{\mu, \theta\}$, and other edges not in $e_1^2 \cup e_2^2 \cup e_3^2$. Then the "valid solution" $E(1)$ of G^1 now contains the edges $\{\alpha, \beta\}$ plus the other edges in $E(2)$ that are not in $e_1^2 \cup e_2^2 \cup e_3^2$.*

By $G_{\text{approx}} = (V_{\text{approx}}, E_{\text{approx}})$ we denote the solution of TR_p for $G = G^0$ produced by $T_{\text{gadget}-\text{to}-\text{cycle}}$ when $E(m)$ is an "optimal solution" $E_{\text{opt}}(G')$ for the DAG G'. Suppose that the procedure $T_{\text{gadget}-\text{to}-\text{cycle}}$ satisfies the following invariant *for each* $i \in \{m-1, m-2, \ldots, 0\}$:

> $(\star^i)$ if $E(i+1)$ is an "optimal solution" for G^{i+1}, then $E(i)$ is an "optimal solution" for G^i.

One problem in executing the procedure $T_{\text{gadget}-\text{to}-\text{cycle}}$ is that we cannot even compute an optimal solution $E_{\text{opt}}(C_i)$ for a strongly connected component C_i since the problem is NP-hard. Nonetheless, we have seen in the previous section how to compute an approximate solution for a strongly connected component C_i. The following proposition essentially states that, provided we satisfy the Invariants $(P1^i)$, $(P2^i)$ and $(\star^i)$, the approximation ratio for each strongly connected component carries over to the final solution $E(0)$.

Proposition 5 *Suppose that we have a ρ-approximation of TR_p on each C_i for some $\rho > 1$. Then, we can use the procedures $T_{\text{cycle}-\text{to}-\text{gadget}}$ and $T_{\text{gadget}-\text{to}-\text{cycle}}$ to design a ρ-approximation of TR_p on G.*

Proof. Suppose that we have a ρ-approximation of TR_p for each C_i. Our simple modification to the procedure $T_{\text{gadget}-\text{to}-\text{cycle}}$ involves replacing each Γ_i by this approximate solution; in other words, we replace the step

> "replace the vertices and edges of Γ_i by the vertices and edges in an optimal solution $E_{\text{opt}}(C_i)$ of C_i"

by the step

> "replace the vertices and edges of Γ_i by the vertices and edges in the ρ-approximate solution of C_i".

We call this modified procedure by $T_{\text{approximate}-\text{gadget}-\text{to}-\text{cycle}}$. To differentiate the outputs of $T_{\text{approximate}-\text{gadget}-\text{to}-\text{cycle}}$ from $T_{\text{gadget}-\text{to}-\text{cycle}}$ we denote the solution $E(i)$ for G^i by $T_{\text{gadget}-\text{to}-\text{cycle}}$ by $E(i)$ itself and the solution $E(i)$ for G^i by $T_{\text{approximate}-\text{gadget}-\text{to}-\text{cycle}}$ by $E(i)'$.

Obviously, $E(i)'$ still remains a "valid solution" for G^i with this change. Note that $|E(m)'| = |E(m)|$. To prove our claim it suffices to show that $|E(j)'| \leq \rho \cdot |E(j)|$ for any $j \in \{m-1, m-2, \ldots, 0\}$. Notice that the only difference between $T_{\text{gadget}-\text{to}-\text{cycle}}$ and $T_{\text{approximate}-\text{gadget}-\text{to}-\text{cycle}}$ in producing $E(j)$ and $E(j)'$ is that the step

"replace the vertices and edges of Γ_i by the vertices and edges in an optimal solution $E_{\text{opt}}(C_i)$ of C_i"

for $i \in \{m, m-1, m-2, \ldots, j+1\}$ was replaced by the step

"replace the vertices and edges of Γ_i by the vertices and edges in the ρ-approximate solution of C_i".

Thus,

$$|E(j)' \setminus \cup_{i=m}^{j+1} E_{C_i}| = |E(j) \setminus \cup_{i=m}^{j+1} E_{C_i}|$$
$$|E(j)' \cap \left(\cup_{i=m}^{j+1} E_{C_i}\right)| \leq \rho \cdot |E(j) \cap \left(\cup_{i=m}^{j+1} E_{C_i}\right)|$$

and finally

$$\begin{aligned} |E(j)'| &= |E(j)' \setminus \cup_{i=m}^{j+1} E_{C_i}| + |E(j)' \cap \left(\cup_{i=m}^{j+1} E_{C_i}\right)| \\ &\leq |E(j) \setminus \cup_{i=m}^{j+1} E_{C_i}| + \rho \cdot |E(j) \cap \left(\cup_{i=m}^{j+1} E_{C_i}\right)| \\ &\leq \rho \cdot \left(|E(j) \setminus \cup_{i=m}^{j+1} E_{C_i}| + |E(j) \cap \left(\cup_{i=m}^{j+1} E_{C_i}\right)|\right) \\ &= \rho \cdot |E(j)| \end{aligned}$$

❑

To summarize, what remains to be shown is the following:

for any arbitrary $i \in \{1, 2, \ldots, m\}$, show that

- procedure $T_{\text{cycle-to-gadget}}$ satisfies Invariant $(P1^i)$ and
- $T_{\text{gadget-to-cycle}}$ satisfies Invariant $(\star^{i-1})$.

But, how do we check the Invariant $(\star^{i-1})$ described above? Let E_1 be a "valid solution" for G^i and suppose that the procedure $T_{\text{gadget-to-cycle}}$ transformed E_1 to a subset E_2 of edges of G^{i-1}. Suppose that instead we could maintain the following two invariants for each $i \in \{1, 2, \ldots, m\}$:

$(P3^i)$ If E_1 is an "optimal solution" for G^i then E_2 is a "valid solution" for G^{i-1}.

$(P4^i)$ A subgraph that is an optimal solution $E_{\text{opt}}(G)$ for G^{i-1}, after application of the procedure $T_{\text{cycle-to-gadget}}$ on the connected component C_i, is transformed to a subgraph $G_{\min} = (V_{\min}, E_{\min})$ that is a valid solution for G^i.

To show that the above two invariants are sufficient to maintain instead of $(\star^{i-1})$, the following proposition becomes useful.

Proposition 6 *$|E_{\text{opt}}(G) \cap E_{C_i}| = |E_{\text{opt}}(C_i)|$ for a strongly connected component $C_i = (V_{C_i}, E_{C_i})$ and $|E_{\text{opt}}(G') \cap E_{\Gamma_i}| = |E_{\text{opt}}(\Gamma_i)|$ for a gadget $\Gamma_i = (V_{\Gamma_i}, E_{\Gamma_i})$.*

Proof. We first prove the claim $|E_{\text{opt}}(G) \cap E_{C_i}| = |E_{\text{opt}}(C_i)|$. This was essentially observed in [1]. One can prove this in two parts as follows.

- We need to show that $|E_{\text{opt}}(G) \cap E_{C_i}| \leq |E_{\text{opt}}(C_i)|$. Suppose that this is not true and hence $|E_{\text{opt}}(G) \cap E_{C_i}| > |E_{\text{opt}}(C_i)|$. But then $(E_{\text{opt}}(G) \setminus (E_{\text{opt}}(G) \cap E_{C_i})) \cup E_{\text{opt}}(C_i)$ gives another solution with fewer edges than in $E_{\text{opt}}(G)$.
- We need to show that $|E_{\text{opt}}(G) \cap E_{C_i}| \geq |E_{\text{opt}}(C_i)|$. Suppose that this is not true and hence $|E_{\text{opt}}(G) \cap E_{C_i}| < |E_{\text{opt}}(C_i)|$. This implies that there is a pair of vertices u and v in C_i such that there is a path $u \Rightarrow v$ from u to v that includes a vertex w not in C_i as an intermediate vertex. But this implies that w should be in the strongly connected component C_i, a contradiction.

$|E_{\text{opt}}(G') \cap E_{\Gamma_i}| = |E_{\text{opt}}(\Gamma_i)|$ is true since we maintain Invariant $(P1^i)$. ❑

Now, we claim the following.

Proposition 7 *If we satisfy Invariant $(P1^i)$ then Invariants $(P3^i)$ and $(P4^i)$ imply Invariant $(\star^{i-1})$.*

Proof. It suffices to show that if we satisfy Invariants $(P3^i)$ and $(P4^i)$ then E_2 is an optimal solution for G^{i-1}.

We first show the proof for $i = m$. The same proof with minor modifications can be applied for $i = m-1, m-2, \ldots, 1$, in that order, to complete the proof of this proposition.

For every $1 \leq q \neq r \leq m$, let $E^i_{q,r}$ be those edges in $\cup_{e \in E^i}\{e\}$ that connect a vertex in C_q (or Γ_q if C_q was already replaced by Γ_q in G^i) to a vertex in C_r (or Γ_r if C_r was already replaced by Γ_r in G^i).

For a pair of q and r, consider the set of edges in $E_1 \cap E^m_{q,r}$. Note that there is no edge common between the set of edges $E_1 \cap E^m_{q,r}$ and $E_1 \cap E^m_{q',r'}$ if either $q \neq q'$ or $r \neq r'$. Remember that in the proof of Lemma 2, we actually showed that any valid solution must contain a subset $E^{m'}_{q,r}$ of the edges in $E_1 \cap E^m_{q,r}$ and the rest of the edges in $E_1 \cap E^m_{q,r}$ come from a minimum collection of groups of edges from $E^m_{q,r}$ that included the edges in this subset. Suppose that the procedure $T_{\text{gadget-to-cycle}}$ mapped the edges in $E^{m'}_{q,r}$ to a subset $E^{(m-1)'}_{q,r}$ of edges in $E^{m-1}_{q,r}$. We claim that any valid (and thus optimal) solution of G^{m-1} must contain the edges in $E^{(m-1)'}_{q,r}$. Indeed, for the sake of contradiction, suppose that it is not so. Then, such an optimal solution does not contain an edge $e \in E^{(m-1)'}_{q,r}$ and applying the procedure $T_{\text{cycle-to-gadget}}$ on this valid solution for G^{m-1} creates a subset of edges in E^m that does not include at least one edge from $E^{m'}_{q,r}$. By $(P4^i)$ this subset of edges must be a valid solution of G^m but this is not possible since every valid solution of G^m must include all the edges in $E^{m'}_{q,r}$.

Now, to show that E_2 is indeed an optimal solution for G^{m-1}, all we need to show is that the rest of the edges in $E_2 \cap E^{m-1}_{q,r}$ that do not belong to $E^{(m-1)'}_{q,r}$ come from a minimum collection of groups of edges from $E^{m-1}_{q,r}$ that include the edges in $E^{(m-1)'}_{q,r}$. Indeed, suppose that it is not so and let a valid solution for G^{m-1} contain another subset E'_2 of $E^{m-1}_{q,r}$ that include the edges in $E^{(m-1)'}_{q,r}$, $|E'_2| < |E_2|$, and applying the procedure $T_{\text{cycle-to-gadget}}$ on this valid solution creates a subset E'_1 of $E^{m-1}_{q,r}$ that include the edges in $E^{m'}_{q,r}$. Note that the procedure $T_{\text{cycle-to-gadget}}$ replaces every edge of either first or second type in G^{m-1} by exactly p edges in G^m. Thus, $|E_1| = p \cdot |E_2|$, $|E'_1| = p \cdot |E'_2|$ and $|E'_2| < |E_2|$ implies $|E'_1| < |E_1|$ contradicting the optimality of E_1.

The same proof can be carried out for $i = m-1, m-2, \ldots, 1$, in that order, except that the line in the proof

> "Remember that in the proof of Lemma 2, we actually showed that any valid solution must contain a subset $E^{m'}_{q,r}$ of the edges in $E_1 \cap E^m_{q,r}$ and the rest of the edges in $E_1 \cap E^m_{q,r}$ come from a minimum collection of groups of edges from $E^m_{q,r}$ that included the edges in this subset."

should be slightly changed to

> "Remember that in the proof of this proposition for $i+1$ that precedes the proof for i, we actually showed that any valid solution must contain a subset $E^{i'}_{q,r}$ of the edges in $E_1 \cap E^i_{q,r}$ and the rest of the edges in $E_1 \cap E^i_{q,r}$ come from a minimum collection of groups of edges from $E^i_{q,r}$ that included the edges in this subset"

and the rest remains essentially the same. ❑

Example 7 (Example illustrating the proof of Proposition 7) *Consider the example in Example 3 again. Suppose that the edge γ is identical to the edge γ' and the edge κ is identical to the edge κ'; thus, we have*

- $E^2_{1,2} = \{\gamma, \delta, \kappa, \nu, \delta', \nu'\}$;
- $E^1_{1,2} = \{\alpha, \beta, \alpha', \beta'\}$.

Suppose that $E^{2'}_{1,2} = \{\gamma\}$. Then, one possible minimal way to cover $E^{2'}_{1,2}$ is by the group $\{\gamma, \delta, \kappa, \nu\}$; thus suppose that $E_1 \cap E^2_{1,2} = \{\gamma, \delta, \kappa, \nu\}$. Then, $E^{1'}_{1,2} = \{\alpha\}$ and $E_2 \cap E^1_{1,2} = \{\alpha, \beta\}$. Moreover, if there was another way to cover the edge α in G^1 with fewer than 2 edges that would have led to covering the edge γ with fewer than 4 edges.

Note that the Invariants $(P1^i)$, $(P3^i)$ and $(P4^i)$ involve only the graphs G^{i-1} and G^i, the procedure $T_{\text{cycle-to-gadget}}$ transforms G^{i-1} to G^i by replacing only one strongly connected component C_i and the procedure $T_{\text{gadget-to-cycle}}$ transforms a solution of G^i to a solution of G^{i-1}. Hence, we can simplify our notations by dropping the subscripts/superscripts and paraphrasing our requirement as follows (abusing the notation for G slightly):

- We are given a graph $G = (V, E)$ that contains at least one strongly connected component of either single or multiple parity.
- The procedure $T_{\text{cycle-to-gadget}}$, as described in detail before, replaces *just one arbitrarily strongly connected component*, say $C = (V_C, E_C)$, of G to transforms G to $G' = (V', E')$. The definition of a valid solution for G' includes the additional constraint that if an edge $u \xrightarrow{x} v$ to or from C was replaced by a set of first or second type edges in G', then a valid solution for G' must either select *all* of these edges or select *none* of these edges. Let $\Gamma = (V_\Gamma, E_\Gamma)$ be the gadget for C. This procedure must satisfy the following invariant:

  $(P1)$ any "valid solution" for G' *must* contain all the edges in the gadget Γ, and

- The procedure $T_{\text{gadget-to-cycle}}$, as described in details before, transforms an optimal $E_1 \subseteq E'$ solution of G' to a solution $E_2 \subseteq E$ of G. This procedure must satisfy the following invariant:

(*P3*) If E_1 is an "optimal solution" for G' then E_2 is a "valid solution" for G.

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

The above rephrasing makes it clear that it suffices to show that invariants are maintained when *just one strongly connected component is replaced*. Our entire discussion then summarizes to state that if we can design the procedures $T_{\text{cycle-to-gadget}}$ and $T_{\text{gadget-to-cycle}}$ as stated above 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 Proposition 6, the fact that Invariant (P1) ensures that all the edges in the gadget Γ for the component C are selected in any valid solution for G' and the fact that the procedure $T_{\text{gadget-to-cycle}}$ replaces the gadget by a valid solution for C, 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'.

In the next two sections, we consider the two cases corresponding to whether C is of multiple parity or single parity and show that the invariants are satisfied.

7.4.1 Handling Strongly Connected Components of Multiple Parity

Note that by definition a multiple parity component must have at least 2 vertices. The component C corresponds to a gadget Γ in the following manner. We introduce one new vertex X_C. An incoming edge in G to some vertex in C now becomes p incoming edges of parities $0, 1, \ldots, p-1$ to X_C and an outgoing edge from some vertex in C now becomes p outgoing edges from X_C of parities $0, 1, \ldots, p-1$. Figure 6 illustrates the gadget for $p = 2$.

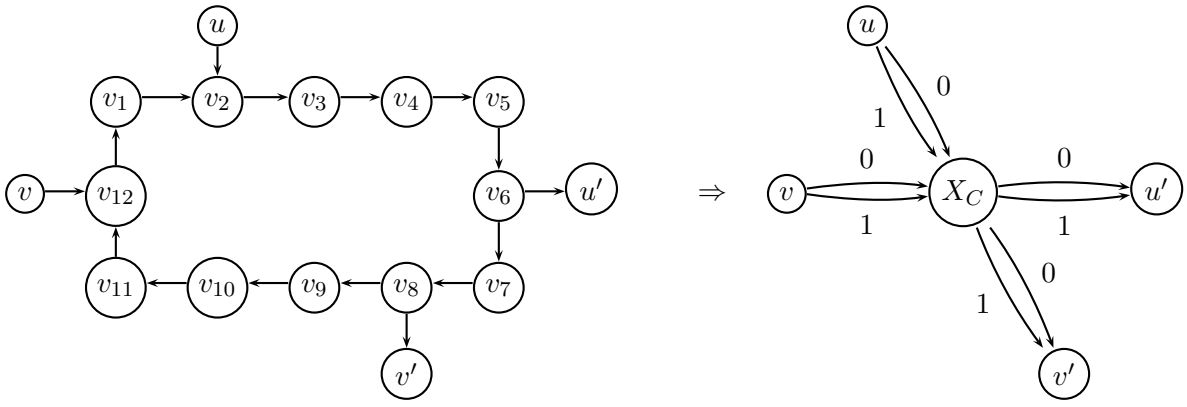

Figure 6: Gadget Γ for a multiple parity component C for the TR_2 problem. Due to Invariant (P2), both or none of the edges in each of the following pairs of edges will be selected in a "valid solution" for G': $\{u \xrightarrow{0} X_C, u \xrightarrow{1} X_C\}$, $\{v \xrightarrow{0} X_C, v \xrightarrow{1} X_C\}$, $\{X_C \xrightarrow{0} u', X_C \xrightarrow{1} u'\}$ and $\{X_C \xrightarrow{0} v', X_C \xrightarrow{1} v'\}$ by $u \xrightarrow{0,1} X_C$, $v \xrightarrow{0,1} X_C$, $X_C \xrightarrow{0,1} u'$, $X_C \xrightarrow{0,1} v'$.

We need to verify the invariants (P1), (P3) and (P4). (P1) vacuously holds.

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

(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 $u \overset{x_1}{\Rightarrow} u' \overset{x_2}{\rightarrow} v' \overset{x_3}{\Rightarrow} v$. 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 $z \in \{0, 1, 2, \ldots, p-1\}$, $X_C \overset{z}{\Rightarrow} v$ exists in G' each $z \in \{0, 1, 2, \ldots, p-1\}$. Pick any vertex $u \in C$. Obviously, $u \overset{0}{\Rightarrow} v = u \overset{x_1}{\Rightarrow} u' \overset{x_2}{\rightarrow} v' \overset{x_3}{\Rightarrow} v$ exists in $E_{\text{opt}}(G)$ where u' is the last vertex on the path that is in C. Then, both $X_C \overset{\{0,1,2,\ldots,p-1\}}{\longrightarrow} v'$ 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

Let v be any vertex in the single parity component $C = (V_C, E_C)$. Define the following notation:

$$[j] = \{x \in V_C \mid v \overset{j}{\Rightarrow} x \text{ exists in } C\}$$

Note that for any $u, v \in V_C$, $u \overset{x}{\Rightarrow} v$ is in C if and only if $v \overset{p-x}{\Longrightarrow} u$ is in C since otherwise $u \overset{y}{\Rightarrow} u$ is in C for some $y \in \{1, 2, \ldots, p-1\}$ which is not allowed due to Lemma 4(b).

Lemma 10 *For any $u \in [i]$ and $u' \in [j]$, $u \overset{x}{\Rightarrow} u'$ 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 Γ for the single parity component $C = (V_C, E_C)$:

- The new vertices and edges in Γ are as follows:

– the set of new vertices are $\bigcup_{i=0}^{p-1} \{\, [i]', [i]'' \,\}$;

– for each $[i]' \in \{\, [0]', [1]', \ldots, [p-1]' \,\}$ and $[j]'' \in \{\, [0]'', [1]'', \ldots, [p-1]'' \,\}$, there is an edge $[i]' \xrightarrow{x} [j]''$ where $x =_p j - i$.

- An incoming edge $u' \xrightarrow{x} u$ to C with $u' \notin V_C$ and $u \in V_C$ is mapped by $T_{\text{cycle-to-gadget}}$ to the following set of edges:

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

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

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

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

As a specific illustration, consider the graph on the top of Figure 7 for the TR_2 problem; the corresponding gadget is shown below.

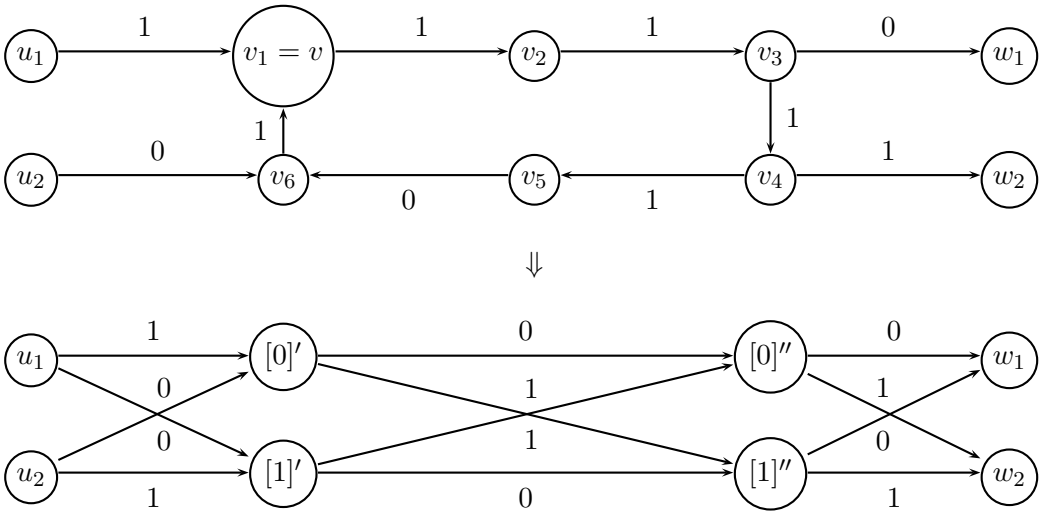

Figure 7: Gadget Γ for a single parity component C for an instance of the TR_2 problem. $[0] = \{v_1, v_3, v_5, v_6\}$, $[1] = \{v_2, v_4\}$.

The following properties of gadget edges will be useful.

Lemma 11 *The following statements are true:*

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

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

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

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

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

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

We need to verify Invariants (P1), (P3) and (P4). (P1) obviously holds. It now remains to verify the invariants (P3) and (P4)

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 $u \overset{x_1}{\Rightarrow} u' \xrightarrow{x_2} w' \overset{x_3}{\Rightarrow} w$ with $x =_p x_1 + x_2 + x_3$. Suppose that $u \in [r]$ and thus $u' \in [s]$ where $s =_p r + x_1$. $E_{\text{opt}}(G')$ contains the path $[s]'' \overset{x_2}{\Rightarrow} w' \overset{x_3}{\Rightarrow} w$ since the edge $[s]'' \xrightarrow{x_2} w'$ exists in G'. Suppose that $T_{\text{gadget-to-cycle}}$ translated this path to a path $u'' \xrightarrow{x_2 \oplus_p s \ominus_p t} w' \overset{x_3}{\Rightarrow} w$ for some $u'' \in [t]$. Then the path $u \overset{x_1}{\Rightarrow} u' \xrightarrow{t \ominus_p s} u'' \xrightarrow{x_2 \oplus_p s \ominus_p t} w' \overset{x_3}{\Rightarrow} w$ 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 $u \overset{x_1}{\Rightarrow} u' \overset{x_2}{\Rightarrow} v' \overset{x_3}{\Rightarrow} w$ where u' and v' are the first and the last vertices that belong to C. But, then $u \overset{x_1}{\Rightarrow} u'$ and $v' \overset{x_3}{\Rightarrow} w$ 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.

Now we turn our attention to the verification of (P4). We know that in $G_{\min}$ Invariant (P2) is not violated for any $(\mathbf{C_{j,x,u}})'$ or $(\mathbf{C_{j,x,u}})''$. Since all the gadget edges are selected in any valid solution for G', to verify (P4) one needs to consider the following cases.

(I) $[i]'' \xrightarrow{x} w$ is in G' for $w \notin \bigcup_{i=0}^{p-1} \{ [i]', [i]'' \}$. By Lemma 11(a) this implies that $E_{\text{opt}}(G)$ contains $u \xrightarrow{x \oplus_p i \ominus_p j} w$ for some $u \in [j]$. 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 10 $w' \in [j \oplus_p y_1]$. 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 $[j \oplus_p y_1]'' \overset{y_2}{\Longrightarrow} w''$ $\overset{x \oplus_p i \ominus_p j \ominus_p y_1 \ominus_p y_2}{\Longrightarrow}$ w. By Lemma 11(a) the edge $[j \oplus_p y_1]'' \overset{y_2}{\Longrightarrow} w''$ implies that the edge $[i]''$ $\overset{j \oplus_p y_1 \ominus_p i \oplus_p y_2}{\Longrightarrow}$ w'' also exists and, by Invariant **(P2)**, selected in $G_{\min}$. Then the path $[i]''$ $\overset{j \oplus_p y_1 \ominus_p i \oplus_p y_2}{\Longrightarrow}$ w'' $\overset{x \oplus_p i \ominus_p j \ominus_p y_1 \ominus_p y_2}{\Longrightarrow}$ w is of parity x.

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

(III) $w_1 \overset{x_1}{\rightarrow} [i] \overset{j \ominus_p i}{\rightarrow} [j] \overset{x_2}{\rightarrow} w_2$ is in G' for $w_1, w_2 \notin \bigcup_{i=0}^{p-1} \{ [i]', [i]'' \}$. $w_1 \overset{x_1}{\Rightarrow} [i]$ and $[j] \overset{x_2}{\Rightarrow} w_4$ exist in $G_{\min}$ by **(II)** and **(I)**, respectively, and $[i] \overset{j \ominus_p i}{\rightarrow} [j]$ is provided by one of the gadget edges for C.

Acknowledgments

We wish to thank the anonymous reviewer for very helpful comments that led to significant improvements in the presentation of the paper. 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*, 5th International Workshop Experimental Algorithms, LNCS 4007, pp. 253-264, Springer-Verlag, 2006.

[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*, Discrete Applied Mathematics, 69 (3), pp. 281-289, 1996.

[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 dynamic model of guard cell signaling*, preprint, 2005, submitted for publication.

[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.