Tel-Aviv University
The Raymond and Beverly Sackler Faculty of Exact Sciences
The Blavatnik School of Computer Science

An Algorithm for Orienting Graphs Based on Cause-Effect Pairs and Its Applications to Orienting Protein Networks

This thesis is submitted in partial fulfillment
of the requirements towards the M.Sc. degree
Tel-Aviv University
School of Computer Science

by
Alexander Medvedovsky

The research work in this thesis has been carried out
under the supervision of Prof. Roded Sharan.

October, 2008

Acknowledgments:

I would like to thank my thesis advisor, Prof. Roded Sharan, for the initial idea and the excellent guidance throughout the research. I would like to thank Prof. Uri Zwick and Prof. Vineet Bafna for substantial contribution to this work and for co-authoring the paper, upon which this thesis is based. I also thank Andreas Beyer and Silpa Suthram for providing the kinase-substrate data, Oved Ourfali for his help with Integer Programming implementation, and Rani Hod for his help with some theoretical issues.

Abstract

In recent years we have seen a vast increase in the amount of protein-protein interaction data. Study of the resulting biological networks can provide us a better understanding of the processes taking place within a cell.

In this work we consider a graph orientation problem arising in the study of biological networks. Given an undirected graph and a list of ordered source-target pairs, the goal is to orient the graph so that a maximum number of pairs will admit a directed path from the source to the target. We show that the problem is NP-hard and hard to approximate to within a constant ratio. We then study restrictions of the problem to various graph classes, and provide an $O(\log n)$ approximation algorithm for the general case. We show that this algorithm achieves very tight approximation ratios in practice and is able to infer edge directions with high accuracy on both simulated and real network data.

This work was presented at the 8th Workshop on Algorithms in Bioinformatics (WABI) [19].

Contents

Chapter 1

Introduction

Recent technological advances allow generation of large-scale protein-protein interaction (PPI) networks. Some of the techniques used for generations of the networks are yeast two-hybrid screening [7, 13, 6] and protein co-immunoprecipitation (Co-IP) paired with mass spectrometry [1, 18]. The analysis of the resulting networks can allow us to better understand the nature of the inner-cellular processes, reflected in protein-protein interactions.

One of the major roles of protein-protein interaction (PPI) networks is to transmit signals within the cell in response to genetic and environmental cues. Technologies for measuring PPIs do not provide information on the direction in which the signal flows. Our goal is to infer the directions of the interactions in a given network by combining causal information on cellular events. One such source of information is perturbation experiments, in which a gene is perturbed and as a result other genes change their expression levels.

In graph theoretic terms, one is given an undirected graph and a list of cause-effect pairs. The goal is to direct the edges of the graph, assigning a single direction to each edge, so that a maximum number of pairs admit a directed path from the cause to the effect. In fact, by contracting cycles in the graph one can easily reduce the problem to that of orienting a tree. Hakimi et al. [9] studied a restricted version of the problem where the list of vertex pairs includes all possible pairs, giving a quadratic time algorithm for it. Another variant of the problem was studied in [3] and [10], where rather than maximizing the total number of pairs, an algorithm was given to decide if one can satisfy *all* given pairs. Other similar graph theoretic problems include Maximum Integral K-Multicommodity Flow On

Trees [8], and Maximum Disjoint Connecting Paths [23, 17].

In this thesis we study the resulting tree orientation problem. We prove that it is NP-hard and hard to approximate to within a constant ratio. We study restrictions of the problem to various graph classes, and show that the problem can be solved in polynomial time on a path, but is still NP-hard for some other simple graph classes. We then provide an $O(\log n)$ approximation algorithm for a general tree, where n is the size of the tree. We show that this algorithm achieves tight approximation ratios in practice and is able to infer edge directions with high accuracy on both simulated and real network data.

The thesis is organized as follows: Chapter 2 gives some biological and computational background. In Chapter 3 the graph orientation problem is presented and its complexity for different graph classes is analyzed, and some related problems are described. Chapter 4 provides exact and approximate algorithms for restrictions of the problem, and an approximation algorithm for the general case. Experimental results of the latter algorithm on simulated and biological data are described in Chapter 5. Finally, a brief summary and possible future research directions appear in Chapter 6.

Chapter 2

Computational and Biological Background

This chapter provides computational and biological background necessary for the presented work. The computational background covers some algorithmic techniques used in this thesis. The biological background provides an introduction to protein-protein interaction networks.

2.1 Computational Background

2.1.1 Linear Programming and Integer Programming

Linear Programming is a technique used for representation and solution of a wide range of optimization problems. A linear program consists of a set of variables $\bar{x} = (x_1, ..., x_n)$, an optimization function $f(x_1, ..., x_n) = c_1x_1 + ... + c_nx_n$, and a set of linear constraints $A\bar{x} \leq \bar{b}$. The goal is to assign values to x_i so that the optimization function is maximized, while maintaining the constraints.

Integer Programming is a type of linear programming, in which all (or part of, in case of *Mixed Integer Programming*) variables are required to be integers. It is usually more useful for solving discreet optimization problems. Unlike classic linear programming, integer programming cannot be solved efficiently in general case, since it is NP-hard. However, there exist many commercial solvers, which can solve integer programs in practical times in many cases.

A version of Integer Programming, in which all variables are required to be 0 or 1, is sometimes called *Binary Integer Programming.* It is one of Richard Karp's famous 21 NP-complete problems [15].

2.1.2 Dynamic Programming

Dynamic Programming is a method of solving computational problems, which have the properties of *overlapping subproblems* and *optimal substructure.*

A problem is said to have overlapping subproblems if it can be broken into subproblems, which can be reused several times in the problem's solution. A common example for such problem is the problem of calculating the nth Fibonacci number. Calculating $F(n)$ is performed by summing $F(n-1)$ and $F(n-2)$. Calculating $F(n-1)$ also involves $F(n-2)$; therefore we can say that the problem has overlapping subproblems structure. A problem has optimal substructure, if an optimal solution can be calculated by dividing the problem into subproblems, and calculating optimal solutions for the subproblems.

The idea behind dynamic programming is to solve the problem by recursively breaking it into subproblems, until a trivial case is reached. Each time a subproblem is solved, the result of the computation is stored, so that when the solution of the subproblem is needed again (because of the overlapping subproblems property), the stored result is used, instead of recalculating the subproblem.

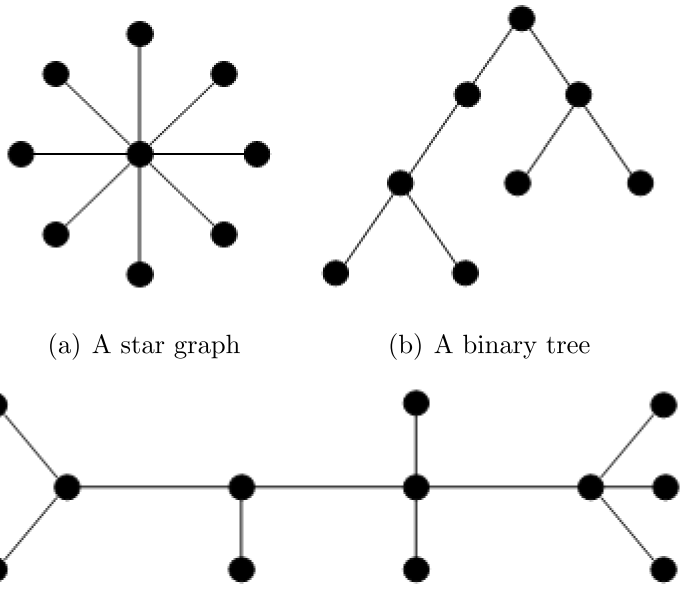
2.1.3 Some Graph Types

We mention several special types of graphs throughout this work. These types of graphs are defined here:

Star Graph A star graph is a tree with depth 1. I.e., all leaves of the graph are connected to the root. See Figure 2.1(a).

Binary Tree A binary tree is a rooted tree, in which each node has at most 2 children. See Figure 2.1(b).

Caterpillar Graph A caterpillar is a tree, in which all leaves are within distance of 1 from a central path. See Figure 2.1(c).


(a) A star graph (b) A binary tree

(c) A caterpillar graph

Figure 2.1: Examples of some graph types.

2.2 Biological Background

2.2.1 Protein-Protein Interactions

Protein-protein interactions (PPI) are physical interactions between proteins within a cell. These interactions are essential in many biological processes. There are different types of interactions. Proteins may bind together and react in biological processes as a complex; or they may interact briefly to transmit a signal within a cell. For example, in the phosphorylation process, a protein (called *kinase* in this context) attaches a phosphate to another protein (called *substrate*). This results in a structural change of the substrate protein, causing it to perform a biological function, which may be in turn a phosphorylation of another protein, or promotion/inhibition of another protein's expression. In this way protein-protein interactions form signaling pathways within cells.

2.2.2 Protein-Protein Interaction Networks

Protein-protein interaction networks are used to analyze data containing multiple proteins and interactions between them. A PPI network is a graph, in which each node represents a protein and each edge represents an interaction between two proteins (see Figure 2.2). These networks are used to study biological processes involving numerous protein-protein interactions, most notably signaling pathways within cells. In recent years new technologies were developed, which allow generation of large scale PPI networks. These technologies have improved our ability to study cellular signaling pathways. However, the techniques used to create large scale PPI networks do not provide information about the directions, in which signals propagate in the network.

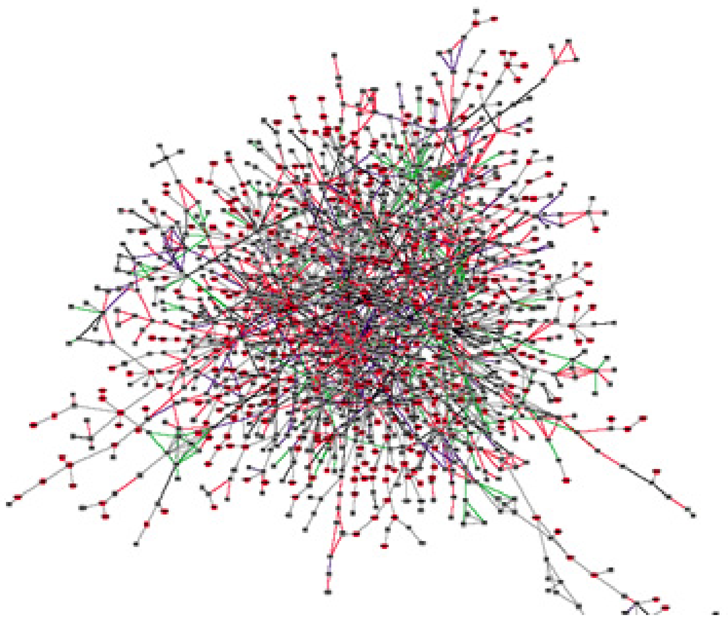


Figure 2.2: A sample protein-protein interaction network. (Source: proteinfunction.net.)

2.2.3 Protein-Protein Interactions Discovery Methods

Currently the main methods for large-scale discovery of protein-protein interactions are Yeast Two-Hybrid screening (Y2H) [7, 13, 6] and Protein Complex Immunoprecipitation (Co-IP) [1, 18]. We will briefly describe these methods here.

Yeast Two Hybrid Screening

The *Yeast Two Hybrid* technique exploits the gene transcription mechanism of the cell. In the regular transcription process a transcription factor binds to the promoter of a target gene via its DNA-binding domain. It then allows binding of RNA polymerase via its transactivation domain, which initiates the transcription.

In the Yeast Two Hybrid technique the transcription factor is divided into two parts. In order to examine whether proteins A and B interact, protein A is attached to the DNA-binding domain of the transcription factor (this hybrid protein is called the *bait*), and protein B is attached to the transactivation domain (forming the *pray*). If A and B interact, the activating domain of the transcription factor is brought near the promoter sequence of the reporting gene, and a transcription occurs. If A and B do not interact, the transcription usually is not initiated. See Figure 2.3.

The Yeast Two Hybrid techniques can be scaled to screen for interactions of a given protein with many other proteins simultaneously. However, an important drawback of the technique is a very high false-positive ratio. For example, in [24] and[4] the authors estimate a false-positive rate of around 50%. Therefore interactions found using Y2H usually need to be verified by other techniques, such as Co-Immunoprecipitation.

Protein Complex Immunoprecipitation

The Complex Immunoprecipitation (Co-IP) process is used to identify protein complexes. In this process a cell extract is prepared, in which proteins occur in their native conformation, and an antibody targeting the protein of interest (the *bait*) is added to the extract. The antibody is then precipitated, together with the bait protein and other proteins that are attached to it. The extracted proteins are then recognized using mass-spectrometry. See Figure 2.4.

The Co-IP technique is considered very reliable in detecting protein complexes. However, it has some drawbacks: it is hard to determine whether a protein interacts directly with the bait protein or whether it interacts via another protein of the complex. Another issue is that the interactions are captured outside the cell (not *in-vivo*); additional tests are required to verify that the interaction also occurs within a cell.

2.2.4 Gene Knockout

In *gene knockout* experiments a gene or a number of genes in an organism are disabled, or "knocked out". This is done by disrupting the DNA sequence in the region of the desired gene. Experiments with the resulting organism can provide insight on biological properties of the knocked-out gene. Of particular interest to us are knock-out experiments, which measure changes of expression of other genes as a result of the knockout. A change of expression of one gene as a result of a knockout of another gene can suggest that the corresponding proteins are part of one signaling pathway. We refer to such pair of genes as a *cause-effect pair*. It is reasonable to assume that the direction of the signaling pathway is from the knocked-out gene (cause) towards the affected gene (effect).

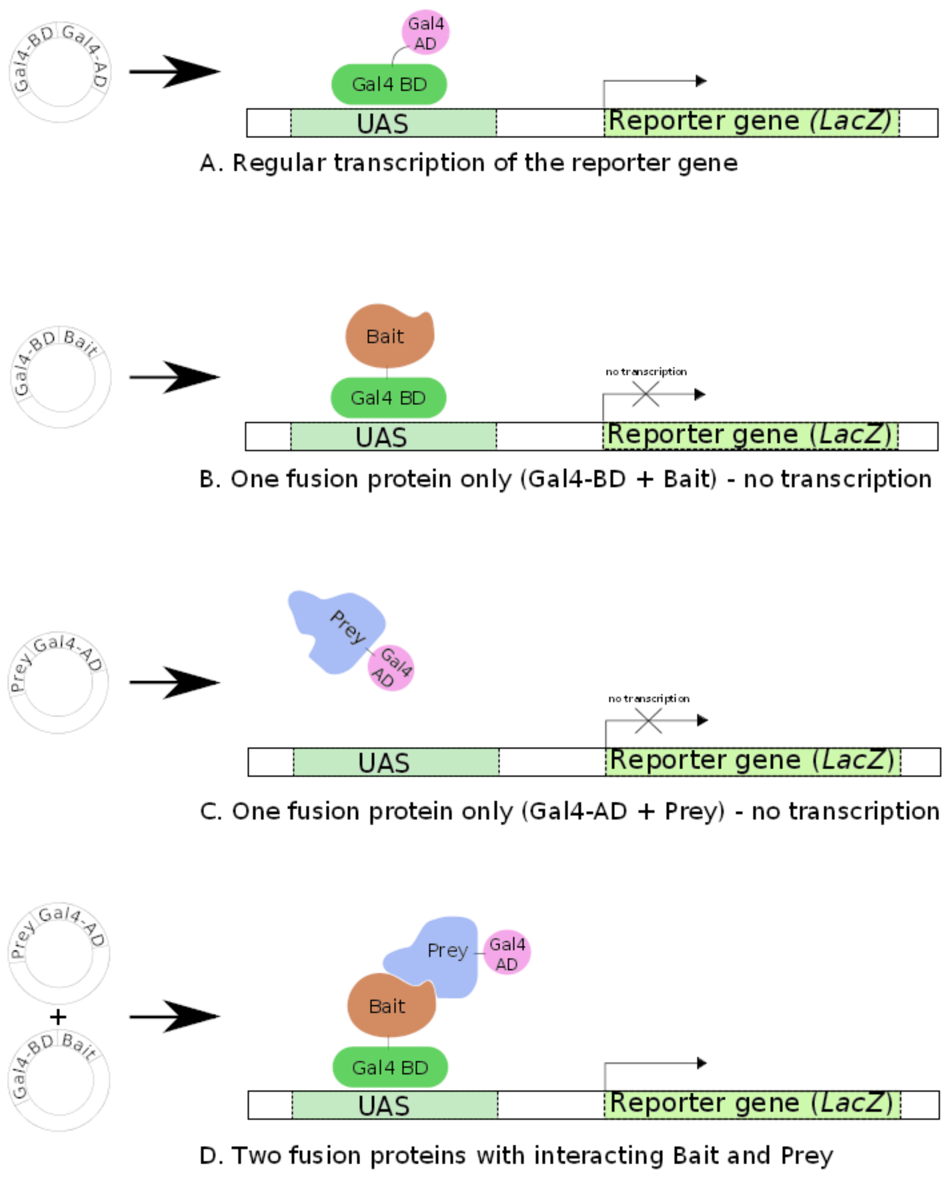


Figure 2.3: The Yeast Two Hybrid technique. **A.** The regular transcription process. **B,C.** Y2H hybridization. The proteins do not interact - no transcription. **D.** The proteins interact, transcription of the reporter gene occurs. (Source: Wikipedia.)

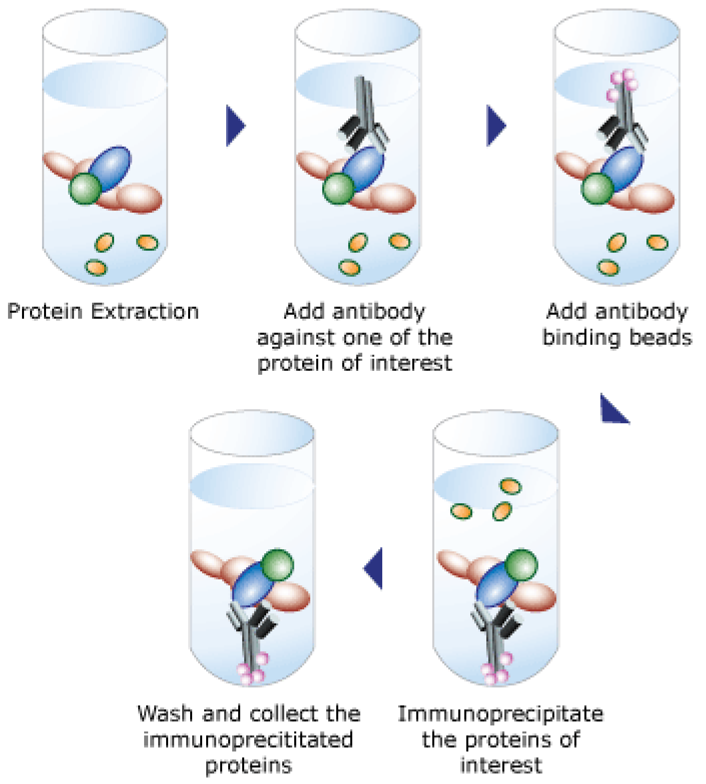

Protein Extraction
Add antibody against one of the protein of interest
Add antibody binding beads
Immunoprecipitate the proteins of interest
Wash and collect the immunoprecititated proteins

Figure 2.4: The Immunoprecipitation process. (Source: www.genwaybio.com.)

Chapter 3

Problem Definition and Related Work

3.1 Maximum Tree Orientation

Let $G = (V, E)$ be an undirected graph. An *orientation* $\vec{G}$ of G is a directed graph obtained from G by orienting each edge $(u, v) \in E$ either from u to v or from v to u. Let $P \subseteq V \times V$ be a set of ordered source-target pairs. A pair $(a, b) \in P$ is satisfied by a given orientation $\vec{G}$ of G if there is a *directed* path from a to b in $\vec{G}$. Our goal is to find an orientation $\vec{G}$ of G that simultaneously satisfies as many pairs from P as possible.

If the graph G contains a cycle C, then it is easy to see that, for any set P, there is an optimal orientation of G in which all the edges of C are oriented in the same direction and, consequently, all pairs that connect two vertices in C are satisfied. The original problem can therefore be solved by *contracting* the cycle C and then solving an equivalent problem on the contracted graph. Thus, the interesting case is when the graph G is a tree.

Definition Maximum Tree Orientation (MTO): Given an undirected tree T and a set P of ordered pairs of vertices, find an orientation of the edges of T that maximizes the number of pairs in P that are satisfied.

In the decision version of the problem, the input includes T, P, and an integer $k \leq |P|$, and the question is whether the edges can be directed so that at least k pairs in P are satisfied. As we show next, the problem is NP-hard even when T is a star or a binary tree.

3.2 Intractability Results

Theorem 3.2.1 *MTO is NP-complete.*

Proof The problem is clearly in NP. We show NP-hardness by reduction from MAX DI-CUT [14], which is defined as follows: given a directed graph $G = (V, E)$ and an integer $k \leq |E|$, is there a cut $A \subset V$ such that there are at least k edges $e = (u, v)$, with $u \in A$ and $v \in V \backslash A$.

We map an instance (G, k) of MAX DI-CUT into an instance $(T = (V', E'), P, k)$ of MTO in the following way: $V' = V \cup \{O\}$, $E' = \{(v, O) : v \in V\}$ and $P = E$.

Given a cut $A \subset V$ with k crossing edges, it is easy to see that each pair corresponding to such an edge can be satisfied: for all $v \in A$ direct the edge (v, O) toward O, and direct all other edges away from O.

On the other hand, suppose that we have directed the edges of T so that k pairs are satisfied. Note that if (u, v) is satisfied then u is directed toward O, and no pair (v', u) can be satisfied. Therefore, the cut defined by $A = \{u \mid (u, O) \text{ is directed toward } O\}$, is of size k. ■

Corollary 3.2.2 *MTO is NP-complete even on stars.*

As MAX DI-CUT is hard to approximate to within a factor of $\frac{11}{12} \simeq 0.9166$ (Håstad [11]), and the reduction is approximation preserving, we conclude:

Corollary 3.2.3 *It is NP-hard to approximate MTO to within a factor of $\frac{11}{12}$.*

Theorem 3.2.4 *MTO is NP-complete on binary trees.*

Proof The problem is clearly in NP. We prove NP-hardness by a reduction from MAX 2-SAT, where each clause is assumed to contain exactly two literals. Suppose f is a 2-SAT formula with variables $x_1, ..., x_n$. Create a binary tree T with subtrees T_s and T_t, so that T_s has a leaf s_i, and T_t has a leaf t_i for each variable x_i. Create two child nodes s_i^t and s_i^f for each s_i, and t_i^t and t_i^f for each t_i (see Figure 3.1).

To complete the reduction we need to specify a set of pairs to be satisfied. This set will be composed of two subsets: P_1, forcing the choice of a truth value for each variable, and P_2, relating these truth values to the clauses in f.

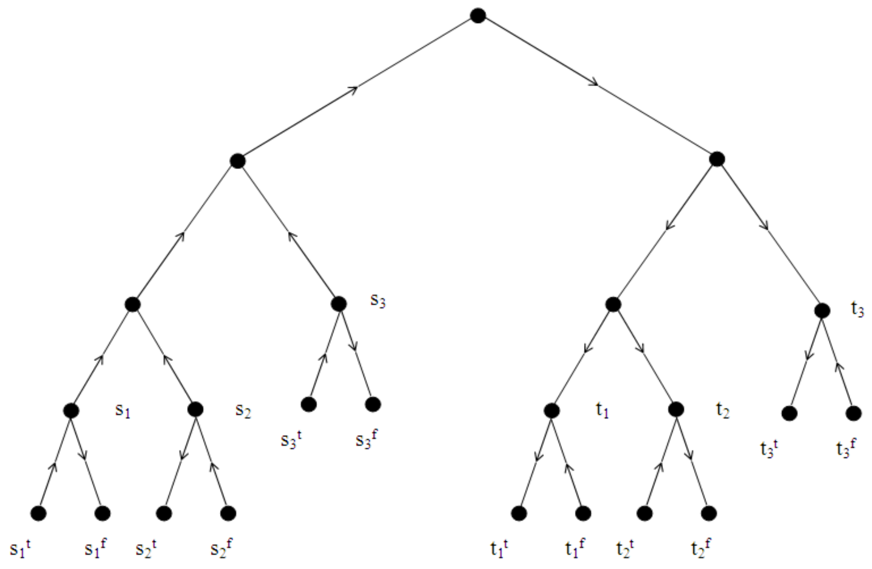


Figure 3.1: An example of the reduction from MAX-2-SAT to MTO. The input 2-SAT formula is $(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_2)$; x_1 and x_3 are assigned a *True* value, and x_2 is assigned a *False* value.

The truth value of a variable will be set by forcing a directed path between s_i^t and s_i^f. If the path is directed from s_i^t to s_i^f we will interpret it as assigning the value *True* to x_i; if it is directed the other way, we will associate the value *False* with x_i. To this end, for every variable x_i participating in n_i clauses, we will add $3n_i$ pairs (s_i^t, s_i^f) and $3n_i$ pairs (s_i^f, s_i^t) to P_1. Similarly, we will force a path between t_i^t and t_i^f, indicating the truth value of x_i, but this time a path from t_i^t to t_i^f will indicate *False* and the opposite direction will indicate *True.* Again, this will be done by adding $3n_i$ pairs (t_i^t, t_i^f) and $3n_i$ pairs (t_i^f, t_i^t) to P_1. Finally, to force the consistency of truth association in T_s and T_t, we will force a directed path from s_i^t to t_i^t or from s_i^f to t_i^f by adding $3n_i$ pairs (s_i^t, t_i^t) and $3n_i$ pairs (s_i^f, t_i^f) to P_1.

The complementing subset of pairs is defined as follows:

$$\begin{aligned}
P_2 \;=\; & \{(s_i^t, t_j^t), (s_i^t, t_j^f), (s_i^f, t_j^t) | (x_i \vee x_j) \in f\} \cup \\
& \{(s_i^f, t_j^t), (s_i^f, t_j^f), (s_i^t, t_j^t) | (\neg x_i \vee x_j) \in f\} \cup \\
& \{(s_i^t, t_j^t), (s_i^t, t_j^f), (s_i^f, t_j^f) | (x_i \vee \neg x_j) \in f\} \cup \\
& \{(s_i^f, t_j^t), (s_i^f, t_j^f), (s_i^t, t_j^f) | (\neg x_i \vee \neg x_j) \in f\}
\end{aligned}$$

Define $P = P_1 \cup P_2$. We claim that c clauses in f can be satisfied iff $18n + c$ pairs in P can be satisfied, where n is the total number of clauses in f.

Suppose that there is a truth assignment that satisfies c clauses in f. Direct the edges (s_i, s_i^t), (s_i, s_i^f), (t_i, t_i^t) and (t_i, t_i^f) according to the assignment. Direct all other edges in T_s upwards, and edges in T_t downwards. For each i, there are $9n_i$ satisfied pairs in P_1. Since $\sum_i n_i = 2n$, the number of satisfied pairs in P_1 is $18n$. Clearly, for every satisfied clause there is a satisfied pair from P_1. Thus, $18n + c$ pairs of P can be satisfied.

Conversely, suppose we have an orientation of T so that $18n + c$ pairs of P are satisfied. For each i there are at most $9n_i$ satisfied pairs in P_1. If the total number of satisfied pairs in P_1 is less than $18n$, then for some i there are less than $9n_i$ satisfied pairs (out of the ones associated with it). This implies that the directions of the edges (s_i, s_i^t), (s_i, s_i^f), (t_i, t_i^t), (t_i, t_i^f) are inconsistent. Thus, either $6n_i$, $3n_i$ or 0 of the corresponding pairs are satisfied. However, if we make these edge directions consistent, we add at least $3n_i$ satisfied pairs from P_1 and lose at most $3n_i$ pairs involving one of $s_i^t, s_i^f, t_i^t, t_i^f$ from P_2. Thus, w.l.o.g., we can assume that these edges are directed consistently, implying exactly $18n$ satisfied pairs from P_1. In addition, we have c satisfied pairs from P_2. Moreover, due to the consistency assumption, each clause can have at most one associated pair satisfied. It follows that c clauses can be satisfied in f. ∎

In the next chapter we show that MTO can be solved in polynomial time on a path. It is therefore interesting to understand where exactly the tractability boundary lies.

Theorem 3.2.5 *MTO is NP-complete on caterpillars.*

Proof Recall that a *caterpillar* is a graph in which all vertices are on a central path or at most one edge away from it. As in the previous proof, we will prove by a reduction from Max 2-SAT. We will use the same notation as in the previous proof. Suppose that a formula f contains variables $x_1, ..., x_n$. Build a path $s_1, s_1', ..., s_n, s_n', t_1, t_1', ..., t_n, t_n'$. For each i, connect a new vertex s_i^t to s_i, s_i^f to s_i', t_i^t to t_i, and t_i^f to t_i'. For each i the set P_1 will contain the pairs $\{(s_i^t, s_i^f), (s_i, s_i^t), (t_i^t, t_i^f), (t_i^f, t_i'), (s_i^t, t_i^t), (s_i^f, t_i^f)\}$ $3n_i$ times. The set P_2 will be defined as previously. The rest of the proof is the same as for binary trees.

Corollary 3.2.6 *MTO is NP-complete even on caterpillars of degree 3.*

3.3 Related Work

Although we could not find a previous work that aims to solve the same problem that we have worked on, there are some problems that may be considered related to the work in this thesis. Some of these works are presented here.

3.3.1 Graph Reachability

In [9] Hakimi et al. studied the problem of finding a graph orientation $\vec{G}$, that maximizes the reachability of a graph G. The reachability of a graph is the number of vertex pairs (x, y), such that there is a directed path from x to y in $\vec{G}$. As we can see this problem is a restricted case of MTO, in which the set of pairs contains all the vertex pairs of G. The authors show that this problem can be solved in $O(n^2)$ time.

3.3.2 Pair Connectivity

In [3] Arkin and Hassin study the problem of pair connectivity on mixed graphs: given a mixed graph (a graph, in which some of the edges are directed) and a set vertex pairs, decide whether there is an orientation, such that there is a directed path between all given vertex pairs. The authors show that the problem is NP-Complete. Notice that in case of undirected graphs the problem can be solved in polynomial time: the graph can be transformed into a tree by cycle contraction, after which it is easy to verify whether there are pairs with conflicting paths.

3.3.3 Maximum Integral K-Multicommodity Flow On Trees

In [8] N. Garg et al study the problem of k-multicommodity flow on trees. The problem is defined as follows: given a tree $T = (V, E)$, a capacity function $c : E \rightarrow N$ and k pairs of vertices (s_i, t_i), assign a flow $f_i \in N$ for each pair (s_i, t_i), so that the total flow through each edge does not exceed its capacity, and $\sum f_i$ is maximized. The authors show that the problem is APX-Complete, but can be approximated within 2.

3.3.4 SPINE

A related work from biological point of view is presented in [20]. The authors aim to explain gene expression changes in knockout experiments by searching signaling pathways in protein-protein interaction networks, and assigning activation/repression attributes to interactions. The authors solve the problem using an Integer Programming formulation.

Chapter 4

Exact and Approximation Algorithms for MTO

We have shown that MTO is NP-hard in the general case, as well as in some special cases. In this chapter we will describe exact and approximation algorithms for special cases and for the general case of the problem.

4.1 Exact Algorithms

We start by providing an integer programming (IP) formulation of the problem that will be useful for studying the practical performance of the algorithms we propose for MTO.

4.1.1 An Integer Program Formulation

Since every two vertices in a tree are connected by a unique path, MTO can be solved using the following integer program:

1. For each vertex pair $p \in P$ introduce a Boolean variable $y(p)$, indicating whether it is satisfied or not.

2. For each edge $e = (u, v) \in T$, where $u < v$, introduce a Boolean variable $x(e)$, indicating its direction (1 if it is directed from u to v, and 0 otherwise).

3. For each pair $p = (a, b) \in P$ and every tree edge $e = (u, v) \in T$, where $u < v$: if the path from a to b in T uses e in the direction from u to v, introduce a constraint $y(p) \leq x(e)$, and if it uses the edge in the direction from v to u, introduce a constraint $y(p) \leq 1 - x(e)$.

4. Maximize the objective function $\sum_{p \in P} y(p)$.

It is possible to consider an LP-relaxation of the above integer programming, but it is not very useful as a value of $|P|/2$ can always be obtained by setting $x(e) = y(p) = \frac{1}{2}$ for every $e \in T$ and $p \in P$.

4.1.2 Solving MTO on Paths

In this section we present a simple dynamic programming algorithm that solves MTO on a path in polynomial time.

Assume that the vertices on the path are numbered consecutively from 1 to n. The edges of the path are $(i, i+1)$, for $1 \leq i < n$. We think of vertex i as lying to the *left* of vertex $i+1$. We also let $[i, j] = \{i, i+1, \ldots, j\}$.

Let P be the input set of pairs. For every $1 \leq i < j \leq n$, let $v^+_{ij} = |\{(a, b) \in P \mid i \leq a < b \leq j\}|$ and $v^-_{ij} = |\{(b, a) \in P \mid i \leq a < b \leq j\}|$. In other words, v^+_{ij} is the number of pairs of P with both endpoints in the interval $[i, j]$ that are satisfied when the edges $(i, i+1), \ldots, (j-1, j)$ are all oriented to the right, while v^-_{ij} is the number of such pairs satisfied when the edges are oriented to the left. Let v_{ij} be the maximal number of pairs of P with both endpoints in $[i, j]$ that can be simultaneously satisfied using *any* orientation of the edges in the interval $[i, j]$. We claim:

Lemma 4.1.1 *For every $1 \leq i < j \leq n$ we have $v_{ij} = \max\{\, v^+_{ij}, v^-_{ij}, \max\limits_{i<k<j} v_{ik} + v_{kj}\}$.*

Proof The proof that $v_{ij} \geq \max\{\, v^+_{ij}, v^-_{ij}, \max_{i<k<j} v_{ik} + v_{kj}\}$ is straightforward. We prove, therefore, the opposite inequality. Consider the orientation of $[i, j]$ that achieves the maximal value of v_{ij}. If all the edges in this orientation are oriented to the right, then $v_{ij} = v^+_{ij}$ and

we are done. Similarly, if they are all oriented to the left, then $v_{ij} = v_{ij}^-$. Otherwise, there is a vertex $i < k < j$ for which the edges $(k-1, k)$ and $(k, k+1)$ have opposite orientations. It follows that no pair (a, b) with $a < k < b$ or $b < k < a$ can be satisfied by such an orientation. Hence, all edges satisfied by this orientation lie in either $[i, k]$ or $[k, j]$; thus, $v_{ij} = v_{ik} + v_{kj}$, as required. ■

As an immediate corollary we get:

Theorem 4.1.2 *MTO on paths of length n can be solved in $O(n^3)$ time.*

4.2 Approximation Algorithms

4.2.1 Approximating MTO on Stars

In this section we describe an approximation algorithm for MTO on stars that will also serve as a building block in our approximation algorithms for general trees.

Lemma 4.2.1 *If T is a star then at least $1/4$ of all pairs can be satisfied.*

Proof Choose a random orientation. Each pair is then satisfied with a probability of at least 1/4. ■

It is easy to use the method of conditional expectations to obtain a deterministic linear time algorithm that produces an orientation of a star that satisfies at least 1/4 of the pairs. This immediately gives us a 1/4-approximation algorithm for the problem. As the MTO problem on stars is equivalent to the Max-DI-Cut problem, an 0.874-approximation for the problem can be obtained using the semidefinite programming based approximation algorithms [5, 16].

Lemma 4.2.1 can be improved as follows:

Lemma 4.2.2 *If T is a star, and $k = \max_x |\{y|(x, y) \in P\}|$, then at least*

$$\frac{k+1}{4k+2}$$

of all pairs can be satisfied.

Proof The lemma follows directly from the result in [2], in which authors show that a digraph with m edges and maximum outdegree of k has a directed cut of size at least $\frac{k+1}{4k+2}m$. ■

4.2.2 Approximating MTO on Caterpillars

Recall that MTO is NP-complete even for caterpillars with maximum degree 3 (see 3.2.6). We show the following:

Lemma 4.2.3 *Let T be a caterpillar. At least $1/8$ of all pairs can be satisfied.*

Proof Partition edges into 'path' edges which lie on the caterpillar path, and 'bush' edges which "stick" from it. Direct the path edges in a single direction by choosing one of the two at random. Also, randomly direct each of the bush edges. Note that each pair of vertices (u, v) is connected by at most two bush edges and a sub-path. Therefore, the probability that (u, v) is satisfied by the random assignment is at least $1/8$. The claim follows. ■

4.2.3 Approximating MTO on Bounded-Depth Trees

In this section we present approximation algorithms for rooted, bounded-depth trees that make use of the approximation algorithm for stars. All the results can be extended to unrooted, bounded-diameter trees by rooting them at a "central" vertex so that their depth is bounded by roughly half the diameter.

Consider a tree T with d levels (and depth $d-1$). We denote the vertices at level i by L_i, starting from the root at level 1. For a node v, denote by T_v the subtree rooted at v. Two notions of separation will be useful to us in the approximation algorithms that we design in the sequel.

Definition A node w in a tree *separates* a pair (u, v), if w is on the path between u and v. w is called the *lowest common ancestor (LCA)* of u and v if in addition it lies on the lowest possible level in the tree.

Lemma 4.2.4 *For any vertex v in T, at least $1/4$ of the pairs separated by v can be satisfied.*

Proof Re-root the tree at v, and denote its subtrees by $T_1, \dots, T_l$. We will direct all edges in T_i either toward v or away from v, consistent with the edge between v and T_i. To direct the edges between v and T_i, construct an instance of MTO on a star T' as follows: the root is v, and there are l leaves $v_1, \dots, v_l$. For each (u, w) separated by v, where $u \in T_x, w \in T_y$, add (v_x, v_y) to P'. By Lemma 4.2.1, 1/4 of the pairs in P' can be satisfied. The edge directions in T' are used to direct the edges of the tree. It is easy to see that any pair (u, w) separated by v, where $u \in T_x, w \in T_y$, is satisfied iff (v_x, v_y) is satisfied in T'. The claim follows. ■

Corollary 4.2.5 *For any vertex v in T, at least $1/4$ of the pairs whose LCA is v can be satisfied.*

Definition For a tree T rooted at a vertex v, we denote by StarMTO(T, P, v) the star-based solution of the MTO problem on T, as described in the proof of Lemma 4.2.4.

Lemma 4.2.6 *Let T be a rooted tree with d levels. At least $1/(4d)$ of the pairs in P can be satisfied.*

Proof As there are d levels, there must be a level j that contains at least $|P|/d$ LCAs of the pairs in P. Compute StarMTO(T_v, P, v) for each node $v \in L_j$. By Corollary 4.2.5, at least $|P|/(4d)$ of the pairs are satisfied. ■

Theorem 4.2.7 *Lemma 4.2.6 is tight up to a constant factor.*

Proof Let T_n be a complete binary tree with n levels and root R. Create a set of pairs between different leaves of T_n with weights $w(u, v)$, so that $w(u, v) = 2^{l(u,v)}$, where $l(u, v)$ is the 0-based level number of the lowest common ancestor of u and v.

For a tree with n levels, denote $v[n, k]$ the maximal weight of satisfied pairs in all orientations, in which exactly k leaves have a directed path to R. Denote $v[n] = \max_k v[n, k]$.

Note that since all pairs are symmetrical, reversing all edges of the tree does not change the weight of the satisfied pairs. Thus $v[n, k]$ also denotes the maximal weight of satisfied pairs in all orientations, in which exactly k leaves have a directed path *from* R.

Proposition 4.2.8

$$v[n, k] = \begin{cases} \frac{4^n + 2\bar{k}^2}{3} - \bar{k}k, & k \geq 1 \\ \frac{4^n - 4}{3}, & k = 0 \end{cases} \tag{4.1}$$

where $\bar{k} = 2^{\lfloor \log k \rfloor}$

Proof The equation is easy to verify for $n = 1$ (a tree with 2 leaves). We assume correctness for n, and prove for $n + 1$. Let us check the relation between $v[n + 1, k]$ and $v[n, k]$.

First verify eq. (4.1) for $k = 0$. Note that since $\bar{k} \leq k$, $v[n, k] \leq \frac{4^n-1}{3} = v[n, 1]$. Since none of the leaves have a path to R, only pairs within the two main subtrees can be satisfied. Since each subtree of T_{n+1} is of the form T_n with double weights, the maximal total weight of satisfied pairs is

$$v[n + 1, 0] = 4 \cdot \max_k v[n, k] = 4 \cdot \frac{4^n - 1}{3} = \frac{4^{n+1} - 4}{3} .$$

Next, assume $k \geq 1$. Define $\Delta v[n, k] = v[n, k] - v[n, k+1]$. For simplicity, define $\bar{0} = -1$.

Lemma 4.2.9 $\Delta v[n, k] = \bar{k}$.

Proof For $k = 0$,

$$\Delta v[n, 0] = v[n, 0] - v[n, 1] = \frac{4^n - 4}{3} - \frac{4^n - 1}{3} = -1$$

For $k > 0$,

$$\begin{aligned} \Delta v[n, k] &= \frac{4^n + 2\bar{k}^2}{3} - \bar{k}k - \frac{4^n + 2\overline{(k+1)}^2}{3} + \overline{(k+1)}(k+1) \\ &= \frac{2}{3}\left(\bar{k}^2 - \overline{(k+1)}^2\right) + \overline{(k+1)}(k+1) - \bar{k}k \end{aligned}$$

If $\overline{(k+1)} = \bar{k}$ then $\Delta v[n, k] = \bar{k}(k + 1 - k) = \bar{k}$; Otherwise, $\overline{(k+1)} = k + 1 = 2\bar{k}$, and

$$\Delta v[n, k] = \frac{2}{3}\left(\bar{k}^2 - 4\bar{k}^2\right) + 4\bar{k}^2 - \bar{k}k = \bar{k}(2\bar{k} - k) = \bar{k} .$$

■

Back to T_{n+1}: if we examine the two topmost edges, there are two cases:

1. Both edges are directed upwards;

2. The edges are oppositely directed.

Define $v_1[n + 1, k]$ and $v_2[n + 1, k]$ as maximal weight of satisfied edges in each of the two cases. Naturally, $v[n + 1, k] = \max\{v_1[n + 1, k], v_2[n + 1, k]\}$.

Examine the first case. If we define $v_1[n+1,k,r]$ as the maximal number of satisfied pairs, when one of the subtrees main has r leaves with a path to R, then

$$\begin{aligned} v_1[n+1,k,r] &= 2v[n,r]+2v[n,k-r] \\ v_1[n+1,k] &= \max_{0\le r\le k/2} v_1[n+1,k,r] \end{aligned}$$

For the second case, define $v_2[n+1,k,r]$ as the maximal number of satisfied pairs, when one subtree has k leaves with a path to R, and the other has r leaves with a path from R. Then

$$\begin{aligned} v_2[n+1,k,r] &= 2v[n,r]+2v[n,k]+kr \\ v_2[n+1,k] &= \max_{0\le r\le 2^n} v_2[n+1,k,r]\,. \end{aligned}$$

For $i=1,2$ define:

$$\Delta v_i[n+1,k,r] = v_i[n+1,k,r]-v_i[n+1,k,r+1]$$

Observe that

$$\begin{aligned} \Delta v_1[n+1,k,r] &= v_1[n+1,k,r]-v_1[n+1,k,r+1] \\ &= 2(v[n,r]+v[n,k-r])-2(v[n,r+1]+v[n,k-r-1]) \\ &= 2\Delta v[n,r]-2\Delta v[n,k-r-1] = 2(\bar r-\overline{(k-r-1)}) \end{aligned}$$

- For $r=0$, $\Delta v_1[n+1,k,r] = -2-2\overline{(k-1)} \le 0$;
- For $r=k-1$, $\Delta v_1[n+1,k,r] = 2(\overline{(k-1)}+1) \ge 0$;
- If $0<r<k-1$, then $\Delta v_1[n+1,k,r] = 2(\bar r-\overline{(k-r-1)})$ is monotonically increasing.

Thus, $v_1[n+1,k,r]$ is maximized when $\Delta v_1[n+1,k,r]$ crosses 0. If $k=\bar k$ is a power of two, then this happens at $r=\bar r = k/2$, that is,

$$\begin{aligned} v_1[n+1,k,r] &= 2v[n,r]+2v[n,k-r] = 4v[n,r] = 4\cdot\frac{4^n-r^2}{3} = \frac{4^{n+1}-k^2}{3} \\ &= \frac{4^{n+1}+2\bar k^2}{3}-\bar k k \end{aligned}$$

Otherwise $k = 2^t + d$, $0 < d < 2^t$. Define r as:

$$r = \begin{cases} 2^{t-1}, & d < 2^{t-1} \\ d, & d \geq 2^{t-1} \end{cases}$$

Then $\bar{r} = \overline{(k-r-1)} = \frac{1}{2}\bar{k}$, and

$$\begin{aligned} v_1[n+1,k,r] &= 2v[n,r] + 2v[n,k-r] \\ &= 2v[n,r] - 2\Delta v[n,k-r-1] + 2v[n,k-r-1] \\ &= 2\left[\frac{4^n + 2\bar{r}^2}{3} - \bar{r}r\right] - 2\overline{(k-r-1)} \\ &\quad +2\left[\frac{4^n + 2\overline{(k-r-1)}^2}{3} - \overline{(k-r-1)}(k-r-1)\right] \\ &= \frac{4^{n+1}}{3} + \frac{8}{3}\frac{\bar{k}^2}{4} - \bar{k}(r+k-r-1+1) = \frac{4^{n+1}+2\bar{k}^2}{3} - \bar{k}k\,. \end{aligned}$$

Now examine $\Delta v_2[n+1,k,r]$.

$$\begin{aligned} \Delta v_2[n+1,k,r] &= v_2[n+1,k,r] - v_2[n+1,k,r+1] \\ &= (2v[n,r] + 2v[n,k] + kr) - (2v[n,r+1] + 2v[n,k] + k(r+1)) \\ &= 2\Delta v[n,r] - k = 2\bar{r} - k \end{aligned}$$

Since $\Delta v_2[n+1,k,r]$ is increasing with r, $v_2[n+1,k,r]$ is maximized at the first r such that $2\bar{r} > k$, i.e., $r = \bar{k}$. Therefore,

$$\begin{aligned} v_2[n+1,k] &= 2v[n,k] + 2v[n,\bar{k}] + k\bar{k} \\ &= 2\left(\frac{4^n + 2\bar{k}^2}{3} - k\bar{k}\right) + 2\left(\frac{4^n + 2\bar{k}^2}{3} - \bar{k}^2\right) + k\bar{k} \\ &= \frac{4^{n+1} - 2\bar{k}^2}{3} - k\bar{k}\,. \end{aligned}$$

Therefore, eq. (4.1) holds. ■

It is easy to see that $v[n,k]$ is maximal for $k = 1$, hence the maximal weight of satisfied pairs for the tree T_n is $v[n] = \frac{4^n-1}{3}$. Since the total weight of pairs in T_n is $2n4^{n-1}$, at most

$$\frac{(4^n-1)/3}{2n4^{n-1}} = \frac{1}{6n}\frac{4^n-1}{4^{n-1}} < \frac{2}{3n}$$

of the pairs can be satisfied. ■

The above lemma provides us with a lower bound on the number of pairs that can be satisfied. It implies an approximation algorithm to MTO with a ratio of $1/(4d)$, but the latter ratio can be improved as we show next:

Lemma 4.2.10 *Let T be a tree with d levels. MTO can be approximated to within a factor of $1/(2d)$ on T.*

Proof We form a d-partite graph G_d, in which each node corresponds to a pair in P. The i-th layer is the set of pairs whose LCAs lie on L_i. We connect two vertices (in two layers) by an edge if the two pairs cannot be simultaneously satisfied. Clearly, the maximum number of pairs that can be satisfied is no more than a maximum sized independent set I in G_d.

For the algorithm, find an independent set I' in G_d. Next, solve StarMTO starting with the root level of the tree, and going down the tree. Specifically, for each vertex $v \in L_i$, solve StarMTO(T_v, I', v) on the pairs in the independent set I'. Note that some of the edge directions have been pre-set by previous levels. However, as I' is an independent set, the edge directions set by previous levels do not contradict any of the pairs in the current level. By Lemma 4.2.1, at least $|I'|/4$ pairs are satisfied. The approximation ratio is therefore

$$\frac{|I'|}{4|I|} = \frac{\alpha_d}{4}$$

where $\alpha_d = 2/d$ is the approximation ratio achievable for independent sets on a d-partite graph [12]. Overall we get an approximation ratio of $1/(2d)$. ■

Lemma 4.2.10 implies a $1/(2 \lg n)$ approximation algorithm for a complete binary tree, as it contains $\lg n$ levels.

4.2.4 Approximating MTO on General Trees

Lemma 4.2.11 *In every tree of size n there is a* centroid *node whose removal breaks the tree into components of size at most $n/2$.*

Our approximation algorithm for general trees is as follows:

Algorithm 1 MTO(T,P)

```
1: Find a centroid $v$,with resulting subtrees $T_1, \ldots, T_l$.
2: Let $A_s = \text{StarMTO}(T, P, v)$.
3: For all $1 \le j \le l$, let $P_j = \text{MTO}(T_j, P)$.
4: return $\max\{A_s, \sum_j P_j\}$.
```

Theorem 4.2.12 *Let T be a tree with n nodes. For any set P, MTO(T, P) finds an orientation that satisfies at least $1/(4\lg n)$ of the pairs.*

Proof Let $R(n)$ be a lower bound on the fraction of the pairs satisfied by the orientation produced by the algorithm when run on a tree with n vertices. We show by induction that $R(n) \ge \frac{1}{4\lg n}$.

At the base of the induction $n = 2$. In this case, at least 1/2 of the pairs are satisfied, and $R(n) \ge 1/2$. Suppose now that $R(k) \ge 1/(4\lg k)$ for any $k < n$. Let $P(n) = |P|R(n)$ denote the minimum number of pairs satisfied by running MTO on an input of size n. Also, let A be the subset of pairs separated by the centroid v. By the induction assumption, $\sum_j P(n_j) \ge \frac{|P|-|A|}{4\lg(n/2)}$. Therefore, applying to the recursion

$$P(n) = \max\left\{\frac{|A|}{4}, \sum_j P(n_j)\right\} \ge \max\left\{\frac{|A|}{4}, \frac{|P|-|A|}{4\lg(n/2)}\right\}$$

The two sub-expressions are balanced for $|A| = |P|/\lg n$, implying that

$$R(n) = \frac{P(n)}{|P|} \ge \frac{1}{4\lg n}$$

■

Chapter 5

Experimental Results

We implemented the general approximation algorithm described above and tested it on simulated and real network data. To evaluate its performance we also implemented the integer-program algorithm which provides an optimal solution to MTO.

5.1 Performance on Simulated Data

Pairs Satisfaction

As a first test of the algorithm, we measured its approximation ratio performance on random trees, by comparing the solutions it obtained to those obtained by the IP algorithm. The random trees contained 1,000 vertices and were generated by iteratively adding a vertex, and connecting it to one of the already existing vertices uniformly at random. The cause-effect pairs were generated randomly with uniform distribution. We tested the algorithm's performance when varying the number of cause-effect pairs from 20 to 1000. The percentage of satisfied pairs along with the implied approximation ratio are displayed in Figure 5.1. Evidently, in case of evenly distributed pairs the algorithm attains very high approximation ratios in practice reaching up to 0.9 and higher ratios on instances with 500 or more pairs.

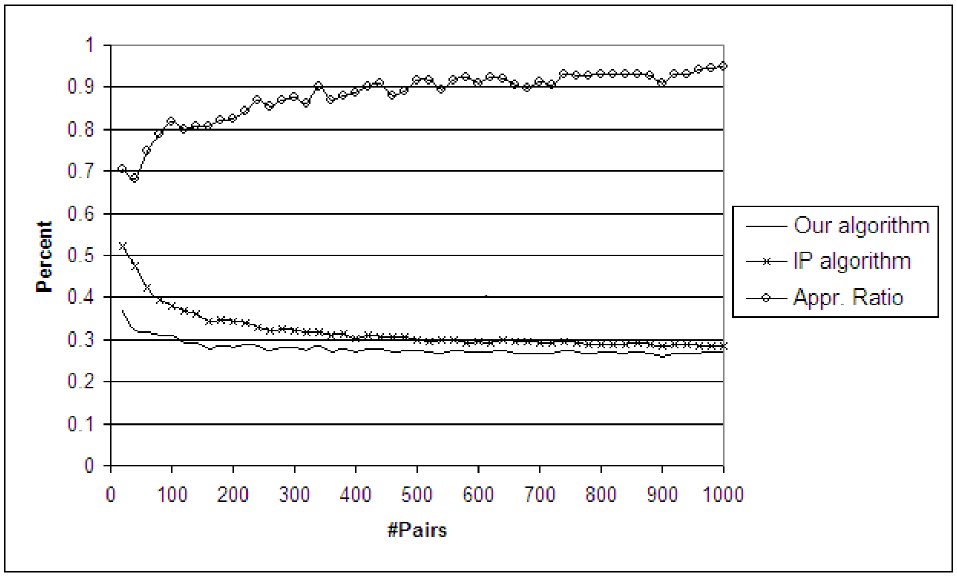


Figure 5.1: Performance on simulated data - pair satisfaction. Percents of satisfied pairs are displayed for both an optimal solution (based on IP) and the approximation algorithm's solution. A third plot depicts the implied approximation ratio.

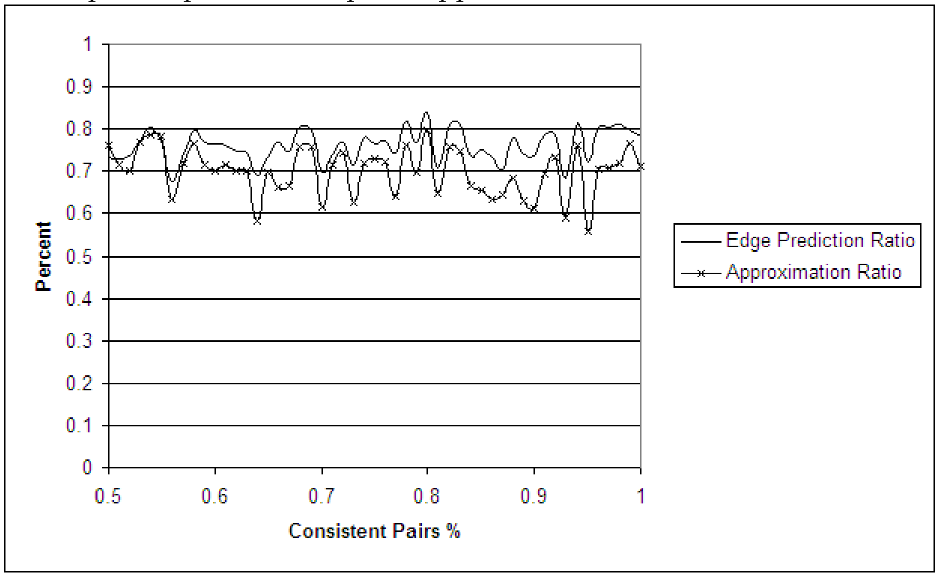


Figure 5.2: Performance on simulated data - edge directions prediction accuracy. The plot displays the percentage of correctly predicted edge directions for different noise levels, together with the approximation ratio achieved (in terms of pair satisfaction).

Edge Directions Prediction

To test the utility of the algorithm in predicting edge directions, we simulated input with known edge directions as follows: we generated 1,000 pairs of vertices, so that at least k of the pairs could be satisfied simultaneously. The remaining pairs were generated randomly, to simulate noise. We randomly chose 50 edges on the paths of the "correct" pairs, and tested the algorithm's accuracy in predicting their direction. We varied the percent of consistent pairs, $k/1000$, from 0.5 to 1. The resulting edge prediction ratio, together with the corresponding approximation ratio (the number of satisfied pairs / maximal number of satisfied pairs) and the total number of satisfied pairs are displayed in Figure 5.2. Noticeably, the percent of correctly predicted directions does not strongly depend on noise level, but is highly correlated with the approximation ratio.

5.2 Biological Data

Next, we tested the performance of our algorithm on real data. To this end, we used a yeast protein-protein interaction (PPI) network consisting of 15,147 protein-protein interactions obtained from the Database of Interacting Proteins [22]. We complemented this network by additional 596 (kinase-substrate) PPIs from [21] for which the direction of signal flow is known (from the kinase to the substrate), represented as undirected edges in the constructed network. For cause-effect pairs, we used knockout data obtained from [25]. The data set contained 24,457 pairs of a knocked out gene (cause) and an affected gene (effect), out of which 14,295 are pairs of proteins from the network.

After removal of small disconnected components (of size ≤ 3) and cycle-contraction, we obtained a tree with 2,027 vertices and 3,370 cause-effect pairs. Interestingly, about 90% of the vertices in the contracted tree were aligned in a star form. Applying our approximation algorithm to the tree yielded an orientation that satisfied 3,262 of the 3,370 pairs. The optimal solution, obtained using integer programming, satisfied 3,295 pairs, implying a practical approximation ratio of 0.99. This tight ratio matches the ratios observed in the simulations (Figure 5.1).

The orientation produced by the algorithm provided predictions for 3,880 interaction directions. 148 of these interactions were from the kinase-substrate data set and, hence, their true directions were known. Remarkably, 147 of these 148 directions were predicted correctly. Notably, none of the kinase-substrate interactions were also cause-effect pairs, but rather lied on paths connecting such pairs.

Chapter 6

Conclusions

In this paper we have studied the problem of orienting a graph so as to satisfy a maximum number of ordered pairs. We have given exact and approximate algorithm to certain restrictions of the problem, and an $O(\log n)$ approximation algorithm for the general case. The algorithm was shown to yield very tight approximation ratios in practice, and attained remarkable accuracy in predicting edge directions on a real protein network.

Several open problems that require further investigation include: (i) closing the gap between the guaranteed approximation ratio in the general case and the approximation hardness result and (ii) tackling the graph orientation problem when some of the edge directions are pre-set (in the biological context this happens when there is prior biological knowledge on directionality or when considering other types of interactions such as transcriptional regulatory ones).

Bibliography

[1] R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422(6928):198–207, 2003.

[2] Noga Alon, Béla Bollobás, András Gyárfás, Jenő Lehel, and Alex Scott. Maximum directed cuts in acyclic digraphs. *J. Graph Theory*, 55(1):1–13, 2007.

[3] E. M. Arkin and R. Hassin. A note on orientations of mixed graphs. *Discrete Applied Mathematics*, 116(3):271–278, 2002.

[4] C. M. Deane, Ł. Salwiński, I. Xenarios, and D. Eisenberg. Protein interactions: two methods for assessment of the reliability of high throughput observations. *Mol Cell Proteomics*, 1(5):349–356, May 2002.

[5] U. Feige and M. X. Goemans. Aproximating the value of two prover proof systems, with applications to MAX 2-SAT and MAX DI-CUT. In *ISTCS*, pages 182–189, 1995.

[6] S. Fields. High-throughput two-hybrid analysis. the promise and the peril. *FEBS Journal*, 272(21):5391–5399, 2005.

[7] S. Fields and O. Song. A novel genetic system to detect protein-protein interactions. *Nature*, 340(6230):245–246, 1989.

[8] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees . *Algorithmica*, 18(1):3–20, 1997.

[9] S. L. Hakimi, E. Schmeichel, and N. E. Young. Orienting graphs to optimize reachability. *Information Processing Letters*, 63(5):229–235, 1997.

[10] R. Hassin and N. Megiddo. On orientations and shortest paths. *Linear Algebra and its applications*, 114/115:589–602, 1989.

[11] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[12] D. S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics*, 6:243–254, 1983.

[13] T. Ito, T. Chiba, and M.Yoshida. Exploring the yeast protein interactome using comprehensive two-hybrid projects. *Trends in Biotechnology*, 19(10):23–27, 2001.

[14] V. Kann, J. Lagergren, and A. Panconesi. Approximability of maximum splitting of k-sets and some other apx-complete problems. *Information Processing Letters*, 58(3):105–110, 1996.

[15] K M Karp. Reducibility among combinatorial problems. In *In Complexity of Computer Computations*. Plenum Press, 1972.

[16] M. Lewin, D. Livnat, and U. Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 67–82, London, UK, 2002. Springer-Verlag.

[17] B. Ma and L. Wang. On the inapproximability of disjoint paths and minimum steiner forest with bandwidth constraints. *Journal of Computer and Systems Sciences*, 60:1–12, 2000.

[18] M. Mann., R.C. Hendrickson, and A. Pandey. Analysis of proteins and proteomes by mass spectrometry. *Annual review of biochemistry*, 70:437–73, 2001.

[19] Alexander Medvedovsky, Vineet Bafna, Uri Zwick, and Roded Sharan. An algorithm for orienting graphs based on cause-effect pairs and its applications to orienting protein networks. In *WABI*, pages 222–232, 2008.

[20] Oved Ourfali, Tomer Shlomi, Trey Ideker, Eytan Ruppin, and Roded Sharan. Spine: a framework for signaling-regulatory pathway inference from cause-effect experiments. In *ISMB/ECCB (Supplement of Bioinformatics)*, volume 23, pages 359–366, 2007.

[21] J. Ptacek, G. Devgan, G. Michaud, H. Zhu, X. Zhu, J. Fasolo, H. Guo, G. Jona, A. Breitkreutz, R. Sopko, R. R. McCartney, M. C. Schmidt, N. Rachidi, S. J. Lee, A. S. Mah, L. Meng, M. J. Stark, D. F. Stern, C. De Virgilio, M. Tyers, B. Andrews, M. Gerstein, B. Schweitzer, P. F. Predki, and M. Snyder. Global analysis of protein phosphorylation in yeast. *Nature 438*, pages 679–84, 2005.

[22] L. Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit, J. U. Bowie, , and D. Eisenberg. The database of interacting proteins: 2004 update. *Nucleic Acids Research, 32*, page D449, 2004.

[23] A. Srinivasan. Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems . *Proc. 38th Ann. IEEE Symp. on Foundations of Comput. Sci.*, IEEE Computer Society:416–425, 1997.

[24] Peter Uetz, Loic Giot, Gerard Cagney, Traci A. Mansfield, Richard S. Judson, James R. Knight, Daniel Lockshon, Vaibhav Narayan, Maithreyan Srinivasan, Pascale Pochart, Alia Qureshi-Emili, Ying Li, Brian Godwin, Diana Conover, Theodore Kalbfleisch, Govindan Vijayadamodar, Meijia Yang, Mark Johnston, Stanley Fields, and Jonathan M. Rothberg. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. *Nature*, 403(6770):623–627, February 2000.

[25] C.-H. Yeang, T. Ideker, and T. Jaakkola. Physical network models. *Journal of Computational Biology*, 11(2/3):243–262, 2004.