Optimally Orienting Physical Networks

Dana Silverbush[⋆,1], Michael Elberfeld[⋆,2], and Roded Sharan[1]

[1] Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel, {danasilv,roded}@post.tau.ac.il

[2] Institute of Theoretical Computer Science, University of Lübeck, 23538 Lübeck, Germany, elberfeld@tcs.uni-luebeck.de

Abstract. In a network orientation problem one is given a mixed graph, consisting of directed and undirected edges, and a set of source-target vertex pairs. The goal is to orient the undirected edges so that a maximum number of pairs admit a directed path from the source to the target. This problem is NP-complete and no approximation algorithms are known for it. It arises in the context of analyzing physical networks of protein-protein and protein-DNA interactions. While the latter are naturally directed from a transcription factor to a gene, the direction of signal flow in protein-protein interactions is often unknown or cannot be measured en masse. One then tries to infer this information by using causality data on pairs of genes such that the perturbation of one gene changes the expression level of the other gene. Here we provide a first polynomial-size ILP formulation for this problem, which can be efficiently solved on current networks. We apply our algorithm to orient protein-protein interactions in yeast and measure our performance using edges with known orientations. We find that our algorithm achieves high accuracy and coverage in the orientation, outperforming simplified algorithmic variants that do not use information on edge directions. The obtained orientations can lead to better understanding of the structure and function of the network.

Key words: network orientation, protein-protein interaction, protein-DNA interaction, integer linear program, mixed graph

1 Introduction

High-throughoutput technologies are routinely used nowadays to detect physical interactions in the cell, including chromatin immuno-precipitation experiments for measuring protein-DNA interactions (PDIs) [10], and yeast two-hybrid assays [6] and co-immunoprecipitation screens [8] for measuring protein-protein interactions (PPIs). These networks serve as the scaffold for signal processing in the cell and are, thus, key to understanding cellular response to different genetic or environmental cues.

⋆ These authors contributed equally to this work.

While PDIs are naturally directed (from a transcription factor to its regulated genes), PPIs are not. Nevertheless, many PPIs transmit signals in a directional fashion, with kinase-substrate interactions (KPIs) being one of the prime examples. These directions are vital to understanding signal flow in the cell, yet they are not measured by most current techniques. Instead, one tries to infer these directions from perturbation experiments. In these experiments, a gene (cause) is perturbed and as a result other genes change their expression levels (effects). Assuming that each cause-effect pair should be connected by a directed pathway in the physical network, one can predict an orientation (direction assignments) to the undirected part of the network that will best agree with the cause-effect information.

The resulting combinatorial problem can be formalized by representing the network as a mixed graph, where undirected edges model interactions with unknown causal direction, and directed edges represent interactions with known directionality. The cause-effect pairs are modeled by a collection of *source-target vertex pairs*. The goal is to orient (assign single directions to) the undirected edges so that a maximum number of source-target pairs admit a directed path from the source to the target.

Previous work on this and related problems can be classified into theoretical and applied work. On the theoretical side, Arkin and Hassin [1] studied the decision problem of orienting a mixed graph to admit directed paths for a given set of source-target vertex pairs and showed that this problem is NP-complete. The problem of finding strongly connected orientations of graphs can be solved in polynomial time [3, 5]. For a comprehensive discussion of the various kinds of graph orientations (not necessarily reachability preserving), we refer to the textbook of Bang-Jensen and Gutin [2].

For the special case of an undirected network (with no pre-directed edges), the orientation problem was shown to be NP-complete and hard to approximate to within a constant factor of 11/12 [12]. On the positive side, Medvedovsky et al. [12] provided an ILP-based algorithm, and showed that the problem is approximable to within a ratio of $O(1/\log n)$, where n is the number of vertices in the network. The approximation ratio was recently improved to $O(\log\log n/\log n)$ [7]. The authors considered also the more general problem on mixed graphs, but the polylogarithmic approximation ratio attained was not satisfying as its power depends on some properties of the actual paths.

On the practical side, several authors studied the orientation problem and related annotation problems using statistical approaches [16, 13].

However, these approaches rely on enumerating all paths up to a certain length between a pair of nodes, making them infeasible on large networks.

Our main contribution in this paper is a first efficient ILP formulation of the orientation problem on mixed graphs, leading to an optimal solution of the problem on current networks. We implemented our approach and applied it to a large data set of physical interactions and knockout pairs in yeast. We collected interaction and cause-effect pair information from different publications and integrated them into a physical network with 3,658 proteins, 4,000 PPIs, 4,095 PDIs, along with 53,809 knockout pairs among the molecular components of the network. We carried out a number of experiments to measure the accuracy of the orientations produced by our method for different input scenarios. In particular, we studied how the portion of undirected interactions and the number of cause-effect pairs affect the orientations. We further compared our performance to that of two layman approaches that are based on orienting undirected networks, ignoring the edge directionality information. We demonstrate that our method retains more information to guide the search, achieving higher numbers of correctly oriented edges.

The paper is organized as follows: In the next section we provide preliminaries and define the orientation problem. In Section 3 we present an ILP-based algorithm to solve the orientation problem on mixed graphs. In Section 4 we discuss our implementation of this algorithm and in Section 5 we report on its application to orient physical networks in yeast. For lack of space, some proofs are shortened or omitted.

2 Preliminaries

We focus on simple graphs with no loops or parallel edges. A *mixed graph* is a triple $G = (V, E_{\mathrm{U}}, E_{\mathrm{D}})$ that consists of a set of vertices V, a set of *undirected edges* $E_{\mathrm{U}} \subseteq \{e \subseteq V \mid |e| = 2\}$, and a set of *directed edges* $E_{\mathrm{D}} \subseteq V \times V$. We assume that every pair of vertices is either connected by a single edge of a specific type (directed or undirected) or not connected. For convenience, we also use the notations $V(G)$, $E_{\mathrm{U}}(G)$, and $E_{\mathrm{D}}(G)$ to refer to the sets V, E_{U}, and E_{D}, respectively.

Let G_1 and G_2 be two mixed graphs. The graph G_1 is a *subgraph* of G_2 iff the relations $V(G_1) \subseteq V(G_2)$, $E_{\mathrm{U}}(G_1) \subseteq E_{\mathrm{U}}(G_2)$, and $E_{\mathrm{D}}(G_1) \subseteq E_{\mathrm{D}}(G_2)$ hold; in this case we also write $G_1 \subseteq G_2$. Similarly, an *induced subgraph* $G[V']$ is a subset $V' \subseteq V$ of the graph's vertices and all their pairwise relations (directed and undirected edges).

A *path* in a mixed graph G of length m is a sequence $p = v_1, v_2, \ldots, v_m, v_{m+1}$ of distinct vertices $v_i \in V(G)$ such that for every $i \in \{1, \ldots, m\}$, we have $\{v_i, v_{i+1}\} \in E_U(G)$ or $(v_i, v_{i+1}) \in E_D(G)$. It is a *cycle* iff $v_1 = v_{m+1}$. Given $s \in V(G)$ and $t \in V(G)$, we say that t *is reachable from* s iff there exists a path in G that goes from s to t. In this case we also say that G *satisfies* the pair (s, t). The *transitive closure* $C(G)$ of a mixed graph G is the set of all its satisfied vertex pairs. A mixed graph with no cycles is called a *mixed acyclic graph* (MAG).

Let G be a mixed graph. An *orientation* of G is a directed graph $G' = (V(G), \emptyset, E_{\mathrm{D}}(G'))$ over the same vertex set whose edge set contains all the directed edges of G and a single directed instance of every undirected edge, but nothing more. We are now ready to state the main optimization problem that we tackle:

*Problem 2.1 (*MAXIMUM-MIXED-GRAPH-ORIENTATION*).*

Input: A mixed graph G, and a set of vertex pairs $P \subseteq V(G) \times V(G)$.
Output: An orientation G' of G that satisfies a maximum number of pairs from P.

3 An ILP Algorithm for Orienting Mixed Graphs

In this section we present an integer linear program (ILP) for optimally orienting a mixed graph. The inherent difficulty in developing such a program is that a direct approach, which represents every possible path in the graph with a single variable (indicating whether, in a given orientation, this path exists or not), leads to an exponential program. Below we will work toward a polynomial size program.

Many algorithms for problems on directed graphs first solve the problem for the graph's strongly connected components independently and, then, work along the directed acyclic graph (DAG) of strongly connected components to produce a solution for the whole instance. Our ILP-based approach for orienting mixed graphs has the same high level structure: In Section 3.1 we define a generalization of strongly connected components to mixed graphs, called strongly orientable components, and show how the computation of a solution for the orientation problem can be reduced to the mixed acyclic graph of strongly orientable components. For MAGs, in turn, we present (in Section 3.2) a polynomial-size ILP that optimally solves the orientation problem.

3.1 A Reduction to a Mixed Acyclic Graph

Let G be a mixed graph. The graph G is *strongly orientable* iff it has a strongly connected orientation. The *strongly orientable components* of G are its maximal strongly orientable subgraphs. It is straightforward to prove that a graph can be partitioned into its strongly orientable components (by noting that if the vertex sets of two strongly orientable graphs intersect, then their union is also strongly orientable). The *strongly orientable component graph*, or *component graph*, G_{SOC} of G is a mixed graph that is defined as follows: Its vertices are the strongly orientable components $C_1,\ldots,C_n$ of G. Its edges are constructed as follows: There is a directed edge (C_i, C_j) in G_{SOC} iff $(v, w) \in E_{\text{D}}(G)$ for some $v \in V(C_i)$ and $w \in V(C_j)$. There is an undirected edge $\{C_i, C_j\}$ in G_{SOC} iff $\{v, w\} \in E_{\text{U}}(G)$ for some $v \in V(C_i)$ and $w \in V(C_j)$. Note that G_{SOC} must be acyclic. The strongly orientable components of a mixed graph G and, hence, the graph G_{SOC}, can be computed in polynomial time as follows: Repeatedly identify cycles in the graph and orient their undirected edges in a consistent direction. After orienting all cycles the strongly connected components that are made up by the directed edges are exactly the strongly orientable components of the initial graph.

To complete the reduction we need to specify the new set of source-target pairs. This also involves a slightly more general definition of the orientation problem where the collection of input pairs is allowed to be a multi-set. Let P be the input multi-set for the original graph G. The multi-set P_{SOC} for the reduced graph is constructed as follows: for every pair $(s, t) \in P$ we insert a pair (C, C') into P_{SOC}, where C and C' are the strongly orientable components that contain s and t, respectively. The following lemma establishes the correctness for the reduction from instances (G, P) to $(G_{\text{SOC}}, P_{\text{SOC}})$.

Lemma 3.1. *Let G be a mixed graph and P a set of vertex pairs from G. For every $k \in \mathbb{N}$ there exists an orientation G' of G that satisfies k pairs from P iff there exists an orientation G'_{SOC} of G_{SOC} that satisfies k pairs from P_{SOC}.*

A mixed acyclic graph $G_{\text{SOC}} = (V, E_{\text{U}}, E_{\text{D}})$ is, in general, neither a forest nor a directed acyclic graph. Its structure inherits from both of these concepts: The undirected graph $(V, E_{\text{U}}, \emptyset)$ is a forest whose trees are connected by the directed edges E_{D} without producing cycles. This observation gives rise to the following definition of topological sortings for mixed graphs: A mixed graph G *admits a topological sorting* if (1) the connected components of $(V, E_\mathrm{U}, \emptyset)$ are trees and (2) they can be arranged in a linear order $T_1, \ldots, T_n$, such that directed edges from E_D can only go from a vertex in T_i to a vertex in T_j if $i < j$. The linear order $T_1, \ldots, T_n$ of the trees is called a *topological sorting* of G. Note that the definition of topological sortings for MAGs also works for DAGs – with every tree being a single vertex. Moreover, similar to DAGs, every MAG admits a topological sorting.

3.2 An ILP Formulation for Mixed Acyclic Graphs

Given an instance of a MAG G and a multiset of vertex pairs P, our ILP consists of a set of binary *orientation* variables, describing the edge orientations, and binary *closure* variables, describing reachability relations in the oriented graph. The objective of satisfying a maximum number of vertex pairs can then be phrased as summing over closure variables for all pairs from P.

The ILP relies on a topological sorting $T_1, \ldots, T_n$ of the input MAG, which allows formulating constraints that force a consistent assignment of values to the orientation and closure variables. The formulation is built iteratively on growing parts of the graph following the topological sorting. Specifically, for every $i \in \{1, \ldots, n\}$, we define $G_i = G[V(T_1) \cup \cdots \cup V(T_i)]$ and $P_i = P \cap (V(G_i) \times V(G_i))$, and for every $i \in \{2, \ldots, n\}$, we define $E_i = E_\mathrm{D}(G) \cap (V(G_{i-1}) \times V(T_i))$. We will first define the variables of the ILP and discuss their intuitive meaning. Then we will define the constraints and the objective function of the ILP, followed by a discussion about the correctness. The ILP I for G and P is made up by the variable set variables(I) that is the union of the binary variables:

$$\left\{o_{(v,w)} \mid \{v, w\} \in E_\mathrm{U}(G)\right\} \tag{1}$$

$$\left\{c_{(v,w)} \mid (v, w) \in V(G) \times V(G)\right\} \tag{2}$$

$$\begin{aligned}\{p_{(v,v',w',w)} \mid \exists\, 2 \le i \le n : (v, w) \in V(G_{i-1}) \times V(T_i) \wedge \\ (v', w') \in E_i\}\end{aligned} \tag{3}$$

The orientation variables (1) are used to encode orientations of the edges: an assignment of 1 to $o_{(v,w)}$ means that the undirected edge $\{v, w\}$ is oriented from v to w. The closure variables (2) are used to represent which vertex pairs of the graph are satisfied: an assignment of 1 to $c_{(v,w)}$ will imply that there exists a directed path from v to w in the constructed orientation. During the construction we will set closure variables

$c_{(v,w)}$ with $(v,w) \in E_{\mathrm{D}}(G)$ to 1, and closure variables $c_{(v,w)}$ where w is not reachable from v in G to 0. *Path variables* are used to describe the satisfaction of a vertex pair (v,w) by using an intermediate directed edge (v',w'): an assignment of 1 to $p_{(v,v',w',w)}$ will imply that there exists a directed path from v to w that goes through the directed edge (v',w').

The ILP contains the constraints

$$o_{(v,w)} + o_{(w,v)} = 1 \quad \text{for all } \{v,w\} \in E_{\mathrm{U}}(G) \tag{4}$$

$$c_{(v,w)} \leq o_{(x,y)} \quad \begin{array}{l} \text{for all } v,w \in V(T_i), \text{ and all } x,y \in V(T_i) \\ \text{where } y \text{ comes directly after } x \text{ on the} \\ \text{unique path from } v \text{ to } w \text{ in } T_i, 1 \leq i \leq n \end{array} \tag{5}$$

$$c_{(v,w)} \leq \sum_{(v',w') \in E_i} p_{(v,v',w',w)} \qquad \text{for all } (v,w) \in V(G_{i-1}) \times V(T_i), 2 \leq i \leq n \tag{6}$$

$$p_{(v,v',w',w)} \leq c_{(v,v')}, c_{(w',w)} \qquad \text{for all } (v,w) \in V(G_{i-1}) \times V(T_i), (v',w') \in E_i, 2 \leq i \leq n \tag{7}$$

and the objective

$$\text{maximize} \sum_{(s,t) \in P} c_{(s,t)} \tag{8}$$

Constraints (4) force that each undirected edge is oriented in exactly one direction. The remaining constraints (5) to (7) are used to connect closure variables to the underlying orientation variables. They force that every closure variable $c_{(v,w)}$ can only be set to 1 if the orientation variables describe a graph that has a directed path from v to w. Whenever v and w are in the same undirected component (which is a tree since the whole graph is a MAG), they can only be connected via an orientation of the unique undirected path between them. For vertex pairs of these kind constraint (5) ensures the above property. Next we consider the case where v and w are in different components T_i and T_j with $i < j$. We need to associate $c_{(v,w)}$ with all possible paths from v to w; this is done by using the path variables: If there is a path from v to w then it must visit a directed edge (v',w') that starts in some component that precedes T_j and ends at T_j (Constraint (6)). Path variables are, in turn, constrained by (7). The objective function maximizes the number of closure variables with assignment 1 that correspond to pairs from P. The above discussion contains the basic ideas to prove the following lemma, which formally implies the correctness of the ILP.

Lemma 3.2. *The following properties hold:*

Completeness: *For every orientation G' of G there exists an assignment $a : \text{variables}(I) \to \{0,1\}$ with $\{(v,w) \in V(G) \times V(G) \mid a(c_{(v,w)}) = 1\} = C(G')$ that satisfies the constraints* (4) *to* (7)*.*

Soundness: *For every assignment $a : \text{variables}(I) \to \{0,1\}$ that satisfies the constraints* (4) *to* (7) *there exists an orientation G' of G with $\{(v,w) \in V(G) \times V(G) \mid a(c_{(v,w)}) = 1\} \subseteq C(G')$.*

The ILP has polynomial size and can be constructed in polynomial time: The construction starts by sorting the MAG topologically. Constant length constraints (4) are constructed for all undirected edges. For every ordered pair (v,w) of vertices v and w that are inside the same undirected component T_i, we construct at most $|E_{\text{U}}|$ constraints of type (5) using reachability queries to T_i. The sum constraints (6) are constructed for all ordered vertex pairs (v,w) where the undirected component of v comes before the undirected component of w in the topological sorting of the MAG. Each sum iterates over the directed edges that lead into the component of w. Thus, each sum's length is bounded by $O(|E_{\text{D}}|)$ and it can be written down in polynomial time. The constraints (7) of constant length are constructed by iterating over the same vertex pairs and directed edges. In total, the size of the ILP is asymptotically bounded by $O(|V(G)|^2(|E_{\text{D}}| + |E_{\text{U}}|))$.

One may ask if it is possible to apply the ILP construction to general mixed graphs instead of MAGs. The MAG-based construction explores the graph iteratively by using a topological sorting. It relies on the fact that connecting paths in MAGs are either unique (inside the undirected components) or can only go from a component T_i to a component T_j if $i < j$. In a mixed graph $G = (V, E_{\text{U}}, E_{\text{D}})$ that contains cycles, connecting paths in (V, E_{U}) are, in general, not unique and there may be directed edges going back and forth between components of (V, E_{U}). This prevents the iterative construction and implies a construction that needs to revise already constructed parts of the formulation instead of just appending new constraints at each step. We are not aware of any method that directly produces polynomial size ILP formulations for general graphs.

4 Implementation Details

Our implementation is written in C++ using BOOST C++ libraries (version number 1.43.0) and the commercial IBM ILOG CPLEX optimizer (version number 12) to solve ILPs. The input of our program consists a mixed graph $G = (V, E_{\mathrm{U}}, E_{\mathrm{D}})$ and a collection P of vertex pairs from G. The program predicts an orientation G' for G that satisfies a maximum number of pairs from P.

The program starts by computing strongly connected orientations for all strongly orientable components of the input graph. This can be done in polynomial time, as described in Section 3.2. Our program implements a linear time approach for this step that is based on combined ideas from [15] and [5]. Next, the program computes the acyclic component graph G_{SOC} of G and transforms the collection of pairs P into the collection of pairs P_{SOC}. Finally, the program computes an optimal orientation for the resulting instance $(G_{\mathrm{SOC}}, P_{\mathrm{SOC}})$ via the ILP approach from Section 3.2. This results in an orientation for all undirected edges that are not inside strongly orientable components and the number of satisfied pairs, which is optimal. Altogether, the program outputs an optimal orientation for the input instance and, if desired, the satisfied pairs and their number.

Due to the combinatorial nature of our approach, there is possibly more than one orientation that results in an optimal number of satisfied pairs. To determine if an undirected edge $e = \{v, w\}$ has the same orientation in all maximum solutions, one can utilize our computational pipeline as follows: First compute the number of satisfied pairs in an optimal solution s_{opt}. Let (v, w) be the orientation of e in this solution. Then run the experiment again, but this time with $\{v, w\}$ replaced by (w, v) in the input network. After that set a *confidence value* $c_e = s_{\mathrm{opt}} - s_e$, where s_e is the maximum number of satisfied pairs for the modified instance. The edge e is said to be oriented with *confidence* iff $c_e \geq 1$; in this case its direction is the same in all optimal orientations of the input.

5 Experimental Results

5.1 Data acquisition and integration

We gathered physical interactions (PPIs, PDIs, and KPIs) and cause-effect pair information for *Saccharomyces cerevisiae* from different sources. We used the PPI data set "Y2H-union" from Yu et al. [17], which contains 2,930 highly-reliable undirected interactions between 2,018 proteins. The PDI data were taken from MacIsaac et al. [11], an update of which can be found at http://fraenkel.mit.edu/improved_map/. We used the collection of PDIs with $p < 0.001$ conserved over at least two other yeast species, which consists of 4,113 unique PDIs spanning 2,079 proteins. The KPIs were collected from Breitkreutz et al. [4] by taking the directed kinase-substrate interactions out of their data set. This results in 1361 KPIs among 802 proteins. A set of 110,487 knockout pairs among 6,228 proteins where taken from Reimand et al. [14].

We integrated the data to obtain a physical network of undirected and directed interactions. We removed self loops and parallel interactions; for the latter, whenever both a directed and an undirected edge were present between the same pair of vertices, we maintained the former only. Pairs of edges that are directed in opposite directions were maintained, and will be contracted into single vertices in later phases of the preprocessing. The resulting physical network, which we call the *integrated network* spans 3,658 proteins, 2,639 PPIs, 4,095 PDIs and 1,361 KPIs. For some of the following experiments we want to control the amount of directed edges better, to investigate their contribution in a purified manner. To this end we will also use the subnetwork of 2,579 proteins of the integrated network that is obtained by taking only the directed PDIs and PKIs, leaving the PPIs out; we call it the *refined network*. To orient the physical networks, we use the set of 110,487 knockout pairs and consider the subset of pairs with endpoints being in the physical network. The integrated network contains 53,809 of the pairs; the refined network contains 34,372 of the pairs.

5.2 Application and performance evaluation

To study the behavior and properties of our algorithm, we apply it to the physical networks and monitor properties of the instance from the intermediate steps and the resulting orientations. For the former, we examine the contraction step, monitoring the size of the component graph obtained (number of vertices, directed edges and undirected edges), and the number of cause-effect pairs after the contraction. For the latter, we run the algorithm in a cross-validation setting, hiding the directions of some of the edges and testing our success in orienting them.

The component graph for the integrated network contains 763 undirected edges and 2,910 directed edges among 2,569 vertices. We filter from the corresponding set of pairs P_{SOC} those pairs that have the same source and target vertices; these pairs lie inside strongly orientable components and are already satisfied. About 85% (44,825) of the initial pairs from the large knockout pair data remain in the contracted graph and can be used to guide the orientation produced by our ILP algorithm. Considering the whole integrated network with the large set of knockout pairs, the component orientation and component contraction steps take 3 seconds and the solution of the ILP takes 70 seconds. Considering only the refined network, the preprocessing as elaborated takes 5 seconds, as there are eventually more components in the contracted graph, and 57 seconds for the solution of the ILP, as there are less choices to be made. Computing confidence scores for undirected edges requires rerunning the steps of the computational pipeline for each of these edges, resulting in about 3.4 hours for the integrated network and 5 hours for the refined network (in which more test edges remain after the cycle contraction).

Next, we wished to evaluate the orientations suggested by our algorithm. To this end, we defined a subset of the directed edges in the input graph (KPIs or PDIs) as undirected *test edges.* Guided by the set of knockout pairs, our program computes orientations for all undirected edges, including the test edges. In the evaluation of the orientation we focus on test edges that survive the contraction and *remain* in the component graph, as the orientation of the other test edges depends only on the cycles they lie in and not on the input cause-effect pairs. We further focus on confident orientations, as other orientations are arbitrary. We define the *coverage* of an orientation as the percent of remaining test edges that are oriented with confidence. The *accuracy* of an orientation is the percent of confidently oriented test edges whose orientation is correct.

When using the integrated network and all 1,361 KPIs as test edges, 166 (12%) of them remain after cycle contraction. The algorithm covers 158 (95%) of the remaining test edges, orienting correctly 137 (86%) of the covered ones. In the refined network 290 (21%) of the KPIs remain, 264 (91%) of them are covered, and 228 (86%) of those are oriented correctly. When using the integrated network and the 4,095 PDIs as test edges 712 (17%) of the test edges remain, 634 (89%) of them are covered, and 614 (96%) of those are oriented correctly. In the refined network 996 (24%) of the PDIs remain, 895 (90%) of them are covered, and 868 (97%) of those are oriented correctly. Expectedly, more test edges remain in the refined network; coverage and accuracy are high in all these experiments.

Effects of the portion of undirected edges. The previous results hint that to obtain a higher percentage of remaining edges, it is helpful to consider networks with a smaller number of undirected edges and larger number of directed edges. To test the effect of the portion of undirected edges more systematically, we focused on the refined network and used different portions (chosen at random) of KPIs as test sets. All KPIs that are not test edges are deleted from the network. The results are depicted in Figure 1(a), demonstrating that the percentage of remaining test edges increases when we consider small fractions of them. This stems from the fact that a smaller number of test edges gives rise to fewer component contractions in the input graph. Interestingly, the coverage and accuracy go up when considering larger amounts of test edges. The reason is that while many parts of the graph are contracted, the initial large number of input test edges leads to a large number of test edges after the contractions. As a result, there is more information to guide the orientation compared to the smaller test sets.

Effects of the amount of cause-effect pairs. Next, we wished to study the effect of the amount of cause-effect pairs on the orientation. We used as input the integrated network, with the KPIs serving as test edges, and applied the algorithm with increasing portions (chosen at random) of pairs. Out of the 166 test edges that remain after contraction, different numbers of covered and accurate edges were attained depending on the input pairs. As evident from Figure 1(b), the more pairs the higher the number of covered edges, albeit with similar accuracy (86-90%), supporting our use of the cause-effect pairs to guide the orientation. Although this is not our objective, it is interesting to note that a high percentage (approximately 85%) of the knockout pairs are satisfied throughout the experiments. The much smaller fraction of unsatisfied pairs may be due to noise in the expression data, incomplete interaction data or molecular events that are not covered by the physical interactions considered here.

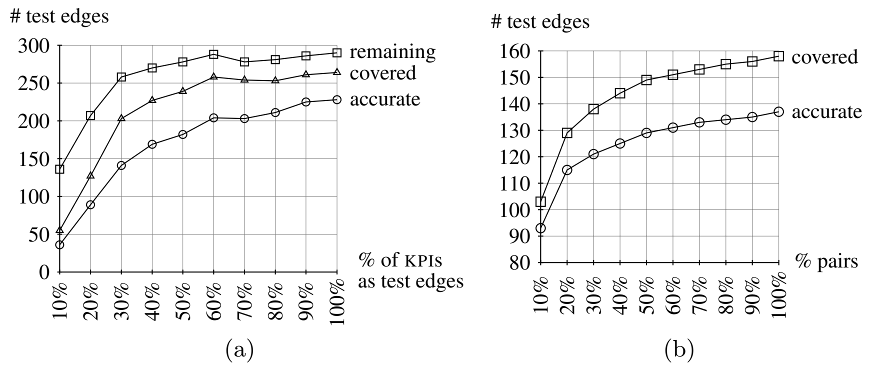

Fig. 1. (a) Remaining, covered and accurate test edges as a function of the percentage of input test edges. x-axis: percentage of KPIs that are used as test edges. y-axis: numbers of test edges that remain (squares), are covered (triangles) and accurately oriented (circles). (b) The number of covered and accurately oriented test edges as a function of the percentage of cause-effect pairs guiding the orientation.

5.3 Comparison to layman approaches

To the best of our knowledge, there exists no previous method to orient mixed graphs, but one can try to adapt methods for undirected graphs to the mixed graph case. The only previous method for orienting large undirected graphs is the one from Medvedovsky et al. [12]. In our terminology, it first computes the graph's component graph, which is a tree for undirected input graphs. It then applies an ILP-formulation, using the fact that there is at most one path between any two vertices. We consider two ways of transforming mixed graphs into undirected graphs to which this method can be applied. Both approaches take their action after the construction of the component graph for the mixed input graph. While our approach, which we call MIXED, uses an ILP at this point, the DELETION approach removes all directed edges from the component graph, yielding a forest of its undirected components to which an ILP is applied. The UNDIRECTED approach considers all directed edges as being undirected and applies a second component contraction step to produce a forest to which the ILP is applied. The same forest can be obtained by starting from the input graph, making all directed edges undirected, and applying a single contraction step.

The behaviors of the intermediate steps of the three approaches when applied to the integrated network are shown in Table 1(a). In comparison to UNDIRECTED, MIXED maintains a higher number of vertices in the component graph, as less cycles are contracted. In comparison to DELETION, MIXED maintains a much higher amount (6 fold) of pairs that are satisfied in the component graph and, therefore, potentially affect the orientation process. This is due to the fact that the edge deletion separates large parts of the graph. Overall, one can see that MIXED retains more information for the ILP step in the form of vertices in the component graph and causal information from the knockout pairs.

To compare the orientations produced by the three approaches, we applied them to the refined network using the KPIs as test edges and different portions of the cause-effect pairs. As the baseline for computing the coverage of the three approaches should be the same – the number of test edges after the initial contraction – we report in the following the absolute numbers of covered (confidently oriented) and correctly oriented interactions, rather than the relative coverage and accuracy measures. Table 1(b) present these results, comparing the numbers of remaining, covered and correctly oriented test edges among the three approaches. Evidently, MIXED yields higher numbers of test edges, covered edges, and correctly oriented edges.

(a)

	DELETION	UNDIRECTED	MIXED
# of undirected edges in the input	2639	8089	2639
# of directed edges in the input	5450	0	5450
# of vertices in the component graph	2569	1483	2569
# of undirected edges in the component graph	763	1445	763
# of directed edges in the component graph	0	0	2910
# of pairs between different vertices in P_{SOC}	44825	24423	44825
# of pairs between different vertices in P_{SOC} that are satisfied in G_{SOC}	4705	23587	29792

(b)

	100% cause-effect pairs			10% cause-effect pairs		
	DELETION	UNDIRECTED	MIXED	DELETION	UNDIRECTED	MIXED
# of test edges						
that remain	290	226	290	290	226	290
that are covered	240	215	265	102	133	144
that are accurate	212	187	229	87	112	121

Table 1. (a) Properties of the intermediate steps of the three orientation approaches. (b) A comparison of the three orientation approaches with cross-validation experiments using different fractions of cause-effect pairs.

6 Conclusions

We presented an ILP algorithm that efficiently computes optimal orientations for mixed graph inputs. We implemented the method and applied it to the orientation of physical interaction networks in yeast. Depending on the input the method yields very high coverage and accuracy in the orientation. Our experiments further show that the algorithm works very fast in practice and produces orientations that cover (accurately) larger portions of the network compared to the ones produced by previous approaches that ignore the directionality information and operate on undirected versions of the networks.

While in this paper we concentrated on the computational challenges in network orientation, the use of the obtained orientations to gain biological insights on the pertaining networks is of great importance. As demonstrated by [9], the directionality information facilitates pathway inference. It may also contribute to module detection; in particular, it is intriguing in this context to map the correspondence between contracted edges (under our method) and known protein complexes.

Acknowledgements

M.E. was supported by a research grant from the Dr. Alexander und Rita Besser-Stiftung. R.S. was supported by a research grant from the Israel Science Foundation (grant no. 385/06).

References

1. E. M. Arkin and R. Hassin. A note on orientations of mixed graphs. *Discrete Applied Mathematics*, 116(3):271–278, 2002.
2. J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications.* Springer, 2nd edition, 2008.
3. F. Boesch and R. Tindell. Robbins's theorem for mixed multigraphs. *The American Mathematical Monthly*, 87(9):716–719, 1980.
4. A. Breitkreutz et al. A global protein kinase and phosphatase interaction network in yeast. *Science*, 328(5981):1043–1046, May 2010.
5. F. R. K. Chung, M. R. Garey, and R. E. Tarjan. Strongly connected orientations of mixed multigraphs. *Networks*, 15(4):477–484, 1985.
6. S. Fields and O.-k. Song. A novel genetic system to detect protein-protein interactions. *Nature*, 340(6230):245–246, July 1989.
7. I. Gamzu, D. Segev, and R. Sharan. Improved orientations of physical networks. In *Proceedings of the 10th International Workshop on Algorithms in Bioinformatics (WABI 2010)*, volume 6293 of *LNCS*, pages 215–225. Springer, 2010.
8. A. Gavin et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415(6868):141–147, Jan. 2002.
9. A. Gitter, J. Klein-Seetharaman, A. Gupta, and Z. Bar-Joseph. Discovering pathways by orienting edges in protein interaction networks. *Nucleic Acids Research*, 2010. doi: 10.1093/nar/gkq1207.
10. T. I. Lee et al. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298(5594):799–804, Oct. 2002.
11. K. MacIsaac et al. An improved map of conserved regulatory sites for saccharomyces cerevisiae. *BMC Bioinformatics*, 7(1):113, 2006.
12. A. Medvedovsky, V. Bafna, U. Zwick, and R. Sharan. An algorithm for orienting graphs based on cause-effect pairs and its applications to orienting protein networks. In *Proceedings of the 8th International Workshop on Algorithms in Bioinformatics (WABI 2008)*, volume 5251 of *LNCS*, pages 222–232. Springer, 2008.
13. O. Ourfali, T. Shlomi, T. Ideker, E. Ruppin, and R. Sharan. SPINE: a framework for signaling-regulatory pathway inference from cause-effect experiments. *Bioinformatics*, 23(13):i359–i366, 2007.
14. J. Reimand, J. M. Vaquerizas, A. E. Todd, J. Vilo, and N. M. Luscombe. Comprehensive reanalysis of transcription factor knockout expression data in saccharomyces cerevisiae reveals many new targets. *Nucleic Acids Research*, 38(14):4768–4777, 2010.
15. R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
16. C. Yeang, T. Ideker, and T. Jaakkola. Physical network models. *Journal of Computational Biology*, 11(2-3):243–262, 2004.
17. H. Yu et al. High-Quality binary protein interaction map of the yeast interactome network. *Science*, 322(5898):104–110, Oct. 2008.