# Exploiting Bounded Signal Flow for Graph Orientation Based on Cause–Effect Pairs*

Britta Dorn[1], Falk Hüffner[2], Dominikus Krüger[3], Rolf Niedermeier[4], and Johannes Uhlmann[4,**]

[1] Fakultät für Mathematik und Wirtschaftswissenschaften, Universität Ulm, Helmholtzstr. 18, D-89081 Ulm, Germany, britta.dorn@uni-ulm.de
[2] Institut für Informatik, Humboldt-Universität zu Berlin, D-10099 Berlin, Germany, hueffner@informatik.hu-berlin.de
[3] Institut für Theoretische Informatik, Universität Ulm, James-Franck-Ring O27, D-89081 Ulm, Germany, dominikus.krueger@uni-ulm.de
[4] Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, D-10857 Berlin, Germany, {rolf.niedermeier, johannes.uhlmann}@tu-berlin.de

**Abstract.** We consider the following problem: Given an undirected network and a set of sender–receiver pairs, direct all edges such that the maximum number of "signal flows" defined by the pairs can be routed respecting edge directions. This problem has applications in communication networks and in understanding protein interaction based cell regulation mechanisms. Since this problem is NP-hard, research so far concentrated on polynomial-time approximation algorithms and tractable special cases. We take the viewpoint of parameterized algorithmics and examine several parameters related to the maximum signal flow over vertices or edges. We provide several fixed-parameter tractability results, and in one case a sharp complexity dichotomy between a linear-time solvable case and a slightly more general NP-hard case. We examine the value of these parameters for several real-world network instances. For many relevant cases, the NP-hard problem can be solved to optimality. In this way, parameterized analysis yields both deeper insight into the computational complexity and practical solving strategies.

## 1 Introduction

Consider a communication network, with a given list of one-way connection request pairs. Each link between two network nodes can only be used in one direction. The task is now to orient the links such that as many communication requests as possible can be fulfilled. We formalize this as follows.

* Main work done while BD and DK were with the Universität Tübingen, and RN and JU were with the Universität Jena.

** Supported by the Deutsche Forschungsgemeinschaft (DFG), research project PABI (NI 369/7).

*Problem Formalization.* Let $G = (V, E)$ be an undirected graph. An *orientation* $\vec{G}$ of $G$ is a directed graph $\vec{G} = (V, \vec{E})$ obtained from $G$ by replacing every undirected edge $\{u, v\} \in E$ by a directed one, i. e., either by $(u, v) \in \vec{E}$ or by $(v, u) \in \vec{E}$. Let $P \subseteq V \times V$ be a set of ordered source–target pairs, which we sometimes refer to as "signals". In order to distinguish pairs from edges or arcs, we use the notation $[a, b] \in P$ to denote the pair starting in $a$ and ending in $b$. We say that a pair $[a, b] \in P$ is *satisfied* by a given orientation $\vec{G}$ if there exists a directed path from $a$ to $b$ in $\vec{G}$. The central problem considered in this work is to find an orientation of a given graph maximizing the number of satisfied pairs. As pointed out by Medvedovsky et al. [9], we can assume that the given graph is a tree: it is clearly optimal to orient the edges of a cycle to form a directed cycle, and, hence, one can contract each cycle to a single vertex, obtaining a tree. Thus, formalized as a decision problem, Maximum Tree Orientation is defined as follows.

> Maximum Tree Orientation (MTO)
> Given an undirected tree $T$, a set $P$ of ordered pairs of vertices of $T$, and an integer $k \leq |P|$, is it possible to find an orientation of $T$ such that at most $k$ pairs in $P$ are not satisfied?

We also consider the weighted version, called Weighted Maximum Tree Orientation (W-MTO), where every pair $[a, b] \in P$ is associated with a rational weight $\omega([a, b]) \geq 1$, and the goal is to maximize the sum of weights of the satisfied pairs.

MTO also has applications in network biology [1,13], more specifically, in the inference of causal relations in biological networks. Often experimental techniques do not yield (enough) information concerning causal relations. This is particularly true for protein–protein interaction (PPI) networks: current technologies like two-hybrid screening can find many protein interactions, but cannot decide the direction of the interaction. Medvedovsky et al. [9] introduced a graph-theoretic model to study signal transmission in PPI networks and the corresponding inference of causal relations. Roughly speaking, the challenge is to orient a given network by combining causal information on cellular events. Medvedovsky et al. [9] formalized this as MTO.

*Previous Work.* MTO was introduced by Medvedovsky et al. [9]; they showed that the problem is NP-complete even when the underlying tree is a star (that is, a diameter-two tree) or a tree with maximum vertex degree three. Moreover, they provided a cubic-time algorithm for MTO restricted to paths. Seeing MTO as the task to maximize the number of satisfied pairs, Medvedovsky et al. also provided polynomial-time approximation algorithms with approximation factor $1/4$ in the case of stars and $O(1/\log n)$ in the case of general $n$-vertex trees. The latter approximation factor was recently improved to $O(\log\log n/\log n)$ by Gamzu et al. [6], who furthermore extended the studies of MTO to "mixed graphs" where some of the edges are already oriented based on causal relations known in advance. Besides these theoretical investigations, Medvedovsky et al. [9] also

provided some experimental results based on a yeast PPI network and some synthetic data. Silverbush et al. [14] very recently did experiments on mixed graphs using integer linear programming. In earlier work Hakimi et al. [8] studied the special case of MTO where the list of pairs to be satisfied contains *all* possible pairs; they developed a quadratic-time algorithm for this case.

*Our Contributions.* We mainly continue and complement the so far mostly theoretical studies on MTO [9,6] by starting a parameterized and multivariate complexity analysis of MTO. That is, we try to better understand the border between tractable and intractable cases of MTO while sticking to optimal (instead of approximate) solutions. In particular, our focus is on the "amount of signal flow" over vertices and edges, respectively, and how this influences the computational complexity of MTO. First, we show that W-MTO can be solved in $O(2^{m_v} \cdot |P| + n^3)$ time on an $n$-vertex tree, where $m_v$ denotes the maximum number of connections paths (one-to-one corresponding to the input vertex pairs) over any tree vertex. In other words, W-MTO is fixed-parameter tractable with respect to the parameter $m_v$. Second, we introduce the concept of cross pairs and show that cross-pair-free instances of W-MTO can be solved in quadratic time, as a corollary also improving the cubic-time algorithm of Medvedovsky et al. [9] for MTO on paths to quadratic time. Third, we additionally show that W-MTO is fixed-parameter tractable with respect to the parameter $q_v$ which is the maximum number of cross pairs over any vertex; namely, it can be solved in $O(2^{q_v} \cdot n^2 \cdot q_v)$ time. Fourth, shifting the focus from "maximum vertex signal flow" to "maximum edge signal flow", we show a sharp complexity dichotomy: W-MTO can be solved in linear time if no tree edge has to carry more than two signals, but if this maximum edge signal flow is three, MTO already becomes NP-hard. Finally, we briefly discuss some practical aspects of exactly solving the so far very few considered real-world instances and conclude that these can be already solved to optimality within milliseconds (via at least three different strategies). However, we also make the point that with the future availability of further real-world data, our new algorithms can be of significant practical relevance beyond so far known or straightforward approaches.

Because of space constraints, some proofs and details are deferred to the full version of this paper.

## 2 Preliminaries, Basic Facts, and Simple Observations

For ease of presentation, for a W-MTO instance $(T, P, \omega)$, we always assume that $\omega([s,t]) = 0$ for all pairs $s, t \in V$ with $[s,t] \notin P$. Moreover, subsequently mostly referring to MTO, the presented concepts and definitions clearly apply to W-MTO as well. Note that in a tree $T = (V, E)$, for each ordered pair $[a, b]$ of vertices, there exists a uniquely determined path connecting these vertices. We will therefore often write *the path defined by the pair* $[a, b]$ when we refer to the unique path in the tree starting in vertex $a$ and ending in vertex $b$, or talk about pairs and paths interchangeably. Sometimes, we also talk about paths in

the tree which do not necessarily correspond to pairs. We denote the undirected path connecting vertices $v$ and $w$ in $T$ by $\text{path}_T(v, w)$. Moreover, $P_v := \{[s, t] \in P \mid v \in V(\text{path}_T(s, t))\}$ denotes the set of paths *passing through a vertex* $v$. An MTO instance is called *rooted* if the underlying tree $T$ is rooted. In a rooted tree $T = (V, E)$, if vertex $a \in V$ is an ancestor of vertex $b \in V$, then we use the notation $a \prec b$. The subtree of $T$ rooted at $v \in V$ is denoted $T_v$.

Let $(T = (V, E), P)$ be an MTO instance, and let $x, y \in P$ be two pairs. We say that $x$ *conflicts with* $y$ if there exists no orientation of $T$ for which both $x$ and $y$ are satisfied. From an $n$-vertex MTO instance, we build in $O(n^3)$ time a *conflict graph* in which each vertex corresponds to an input pair of the MTO instance, and where there is an edge between two pairs if and only if they conflict with each other. More formally, given an MTO instance $(T = (V, E), P)$, the corresponding conflict graph $G_c(T, P)$ is defined as $G_c(T, P) := (P, E_c)$ where $E_c := \{\{u, v\} \mid u, v \in P \wedge u \text{ conflicts with } v\}$.

Clearly, for an orientation of $(T, P)$, in $G_c$ there are no edges (that is, conflicts) between the vertices corresponding to the satisfied source–target pairs, and hence the vertices corresponding to the non-satisfied source–target pairs form a vertex cover for $G_c$, that is, a vertex set $V' \subseteq P$ such that for every edge $e \in E_c$ at least one endpoint of $e$ is in $V'$. This yields the following useful observation.

**Proposition 1.** *Finding a minimum-weight vertex cover in the conflict graph* $G_c(T, P)$ *one-to-one corresponds to determining a minimum-weight set of pairs that cannot be satisfied in* $(T, P)$.

Parameterized complexity is a two-dimensional framework for the analysis of computational complexity [4,5,10]. One dimension is the input size $n$, and the other one is the *parameter* (usually a positive integer). A problem is called *fixed-parameter tractable* (fpt) with respect to a parameter $k$ if it can be solved in $f(k) \cdot n^{O(1)}$ time, where $f$ is a computable function only depending on $k$. For instance, it is well-known that finding an optimal (weighted) vertex cover is NP-hard but fixed-parameter tractable with respect to the parameter "solution size". Due to Proposition 1 we can immediately conclude that MTO and W-MTO are fixed-parameter tractable with respect to the parameters "number of unsatisfied vertex pairs" or "total weight of unsatisfied vertex pairs", respectively (parameter $k$).

## 3 Bounded Signal Flow Over Vertices

In this section, we investigate how the vertex-wise structure of the source–target pairs influences the computational complexity of Maximum Tree Orientation. More specifically, first we consider the parameter $m_v$ denoting the maximum number of source–target paths passing through a vertex. We show that MTO can be solved in $O(2^{m_v} \cdot |P| + n^3)$ time. In other words, MTO is fixed-parameter tractable with respect to the parameter $m_v$. Motivated by this positive result, we explore in more depth the structure of the source–target paths

that pass through a vertex. To this end, we introduce the concept of "cross pairs" and show that for cross-pair-free instances MTO can be solved in $O(n^2)$ time. Informally speaking, an instance is cross-pair-free if the input tree can be rooted such that for each source–target pair one endpoint is an ancestor of the other one. Then, for a rooted MTO instance a cross pair is a source–target pair such that none of its endpoints is the ancestor of the other endpoint. By refining the solving strategy for cross-pair-free instances, we show that Maximum Tree Orientation can be solved in $O(2^{q_v} \cdot n^2 \cdot q_v)$ time, where $q_v$ denotes the maximum number of cross pairs passing through a vertex.

All algorithms in this section are based on dynamic programming, and, hence, since source–target pair weights can easily be incorporated, extend to W-MTO.

### 3.1 Parameter "Maximum Number of Pairs Per Vertex"

Here, we show that W-MTO is fixed-parameter tractable for the parameter $m_v$ denoting the maximum number of source–target pairs passing through a vertex. To this end, we construct in polynomial time a tree decomposition of the conflict graph of treewidth at most $m_v$ (proof omitted). Informally speaking, the treewidth [10] measures the "tree-likeness" of a graph, and a tree decomposition is the "embedding" of a graph into a tree depicting the tree-like structure of the graph. Recall that (weighted) MTO is equivalent to (weighted) Vertex Cover on the conflict graph (see Proposition 1). Thus, the running time follows by the fact that (weighted) Vertex Cover can be solved in $O(2^{\mathrm{tw}}n)$ time, given a tree decomposition of width tw [10].

**Theorem 1.** *On $n$-vertex trees,* Weighted Maximum Tree Orientation *is solvable in $O(2^{m_v} \cdot |P| + n^3)$ time, where $m_v$ denotes the maximum number of source–target pairs passing through a vertex.*

### 3.2 Cross Pairs

In the previous subsection, we have shown that W-MTO is fixed-parameter tractable with respect to the parameter $m_v$. In the following two subsections, we will strengthen this result by showing that W-MTO is fixed-parameter tractable with respect to the parameter "number of a special type of source–target pairs (the so-called cross pairs) passing through a vertex". The idea in the next two subsections is to identify a "trivial" (that is, polynomial-time solvable) special case of the problem and then to investigate instances that are close to these trivial instances, their closeness measured in terms of a certain parameter which is referred to as *distance from triviality* [7,11].

In the following, we will always consider *rooted* trees. Informally speaking, a cross-pair-free instance only contains source–target pairs whose corresponding paths are directed either towards the root or towards the leaves, but do not change their direction. Cross-pair-free instances of W-MTO are of special interest since they constitute our "trivial instances".

**Definition 1.** *Let $(T = (V, E), P, \omega)$ be an instance of W-MTO where $T$ is a rooted tree. A source–target pair $p = [a, b] \in P$ is called* cross pair *if neither $a$ is an ancestor of $b$ nor $b$ an ancestor of $a$. An instance of W-MTO is called* cross-pair-free *if $T$ can be rooted such that $P$ does not contain any cross pairs.*

### 3.3 Cross-pair-free Instances

Now, we devise a dynamic-programming-based algorithm solving W-MTO in quadratic time on cross-pair-free instances.

**Theorem 2.** *On $n$-vertex trees,* Weighted Maximum Tree Orientation *for cross-pair-free instances with given root can be solved in $O(n^2)$ time.*

*Proof.* We present a dynamic programming algorithm with quadratic running time solving a cross-pair-free W-MTO instance $(T = (V, E), P, \omega)$ with root $r$. For the presentation of the algorithm, we use the following notation. For all $v, w \in V$ with $v \prec w$ (that is, $v$ is an ancestor of $w$) let $T_w^v$ denote the subtree of $T$ induced by $V_w^v := V(T_w) \cup V(\mathrm{path}_T(v, w))$. For ease of presentation, let $V_w^w := V(T_w)$. Moreover, let $P_w^v := \{[s, t] \in P \mid s, t \in V_w^v\}$. That is, $T_w^v$ is the tree consisting of the path $\mathrm{path}_T(v, w)$ and the subtree $T_w$ rooted at $w$, and $P_w^v$ are the pairs with both endpoints in $T_w^v$. Finally, the *weight* of an orientation $\vec{T}_w^v$ of $(T_w^v, P_w^v)$ is the sum of the weights of the pairs in $P_w^v$ satisfied by $\vec{T}_w^v$.

The algorithm maintains an $n \times n$ dynamic programming table $S$, containing for each $v, w \in V$ with $v \prec w$ or $v = w$ the two entries $S(v, w)$ and $S(w, v)$. The goal of the dynamic programming procedure is to fill $S$ in accordance with the following definition.

For all $v, w \in V$ with $v \prec w$, entry $S(v, w)$ is the maximum weight of an orientation of $(T_w^v, P_w^v)$ among all orientations of $(T_w^v, P_w^v)$ orienting the path between $v$ and $w$ from $v$ to $w$ (that is, away from the root). Analogously, $S(w, v)$ is the maximum weight of an orientation of $(T_w^v, P_w^v)$ among all orientations of $(T_w^v, P_w^v)$ orienting the path between $v$ and $w$ from $w$ to $v$ (that is, towards the root). Note that in the case $v = w$, we have that $S(v, v)$ is the weight of an optimal orientation of the subtree rooted at $v$.

Next, we describe how our algorithm computes the entries of $S$ in accordance with this definition. The weight of an optimal orientation of $(T, P)$ can then be found in $S(r, r)$.

To compute the entries of $S$, visit all vertices $w \in V$ in a bottom-up traversal. Then, for each $w$ consider all vertices $v \in V$ with $v = w$ or $v \prec w$ and set (omit the sum if $w$ is a leaf):

$$S(v, w) := A(v, w) + \sum_{u \text{ is a child of } w} \max\{S(u, w), S(v, u) - A(v, w)\},$$

$$S(w, v) := A(w, v) + \sum_{u \text{ is a child of } w} \max\{S(w, u), S(u, v) - A(w, v)\}.$$

Herein, $A(v, w)$ denotes the sum of the weights of the source–target pairs with both endpoints on $\mathrm{path}_T(v, w)$ that are satisfied when orienting the path between $v$ and $w$ from $v$ to $w$, that is,

$$A(v, w) := \omega(\{[s, t] \in P \mid s, t \in V(\mathrm{path}_T(v, w)) \wedge s \prec t\}).$$

Analogously, $A(w, v) := \omega(\{[s, t] \in P \mid s, t \in V(\mathrm{path}_T(v, w)) \wedge t \prec s\})$. Moreover, for ease of presentation we assume that $A(v, v) = 0$.

For the correctness of the algorithm note the following. For a leaf $w$ and an ancestor $v$ of $w$, the tree $T_w^v$ is identical to the path $\mathrm{path}_T(v, w)$. Hence, the sum of the weights of pairs that can be satisfied by orienting the path either from $v$ to $w$ or from $w$ to $v$ is $A(v, w)$ and $A(w, v)$, respectively. Next, consider the case that $w$ is an inner vertex and let $v$ be an ancestor of $w$. Moreover, let $u_1, \ldots, u_\ell$ denote the children of $w$. We argue that the maximum weight of an orientation of $(T_w^v, P_w^v)$ orienting the edges on $\mathrm{path}_T(v, w)$ towards $w$ equals

$$A(v, w) + \sum_{i=1}^{\ell} \max\left\{S(u_i, w), S(v, u_i) - A(v, w)\right\}, \tag{1}$$

and, hence, $S(v, w)$ is computed correctly. To this end, consider a maximum-weight orientation $\vec{T}_w^v$ of $(T_w^v, P_w^v)$ orienting the edges on $\mathrm{path}_T(v, w)$ towards $w$. If, for a child $u_i$, $\vec{T}_w^v$ contains the arc $(u_i, w)$, then the contribution of the source–target pairs in $P_w^v$ with at least one endpoint in $T_{u_i}$ to the weight of $\vec{T}_w^v$ is $S(u_i, w)$; note that no source–target pair of $P_w^v$ with exactly one endpoint in $T_{u_i}^w$ is satisfied by $\vec{T}_w^v$, and, thus, the contribution of these pairs is $S(u_i, w)$ (a smaller contribution would contradict the optimality of $\vec{T}_w^v$). Moreover, if for a child $u_i$ the oriented tree $\vec{T}_w^v$ contains the arc $(w, u_i)$, then it follows by a similar argument that the contribution of the paths in $P_w^v$ with at least one endpoint in $V(T_{u_i})$ is $S(v, u_i) - A(v, w)$. The only difference is that the contribution of the source–target pairs with both endpoints in $V(\mathrm{path}_T(v, w))$ is already considered in the above formula, and, hence, must be subtracted from $S(v, u_i)$.

We omit the proof of the running time. □

Note that if the root of a cross-pair-free W-MTO instance is not known, it can be calculated in $O(n|P|)$ time by trying all roots and then checking for each pair if the least common ancestor is one of the two endpoints.

As an immediate consequence of Theorem 2, we can improve the cubic-time algorithm for MTO on paths by Medvedovsky et al. [9] to quadratic time. Herein, we use that every path rooted at one of its endpoints results in a cross-pair-free instance of MTO.

**Corollary 1.** Weighted Maximum Tree Orientation *on $n$-vertex paths can be solved in $O(n^2)$ time.*

### 3.4 Parameter "Maximum Number of Cross Pairs Passing Through a Vertex"

Next, we show that W-MTO is fixed-parameter tractable with respect to the parameter $q_v$ by extending the dynamic programming algorithm for cross-pair-free instances. Formally, $q_v$ is defined as follows. For a rooted W-MTO instance $(T = (V, E), P)$ with root $r$, let $Q$ denote the set of cross pairs. Moreover, for $v \in V$ let $Q_v := P_v \cap Q$ be the set of cross pairs passing through $v$. With respect to the root $r$ the maximum number $q_v(r)$ of cross pairs passing through a vertex is given by $\max_{v \in V} |Q_v|$. Then, $q_v$ is the minimum value of $q_v(r)$ over all possible choices $r$ to root $T$.

**Theorem 3.** *On $n$-vertex trees,* Weighted Maximum Tree Orientation *with given root can be solved in $O(2^{q_v} \cdot q_v \cdot n^2)$ time, where $q_v$ denotes the maximum number of cross pairs passing through a vertex.*

The basic idea of the algorithm is to incorporate the cross pairs by trying for every vertex all possibilities to realize the cross pairs passing through this vertex. To this end, we extend the matrix $S$ by an additional dimension. As a consequence, the dynamic programming update step becomes significantly more intricate. The details are omitted for space constraints.

## 4 Bounded Signal Flow Over Edges

We now consider MTO instances where the number $m_e$ of paths that pass through an edge is limited. We show that the problem is linear-time solvable for $m_e \leq 2$, but NP-hard for $m_e \geq 3$, thereby establishing a dichotomy on the complexity of MTO with respect to $m_e$.

First, we note that if $m_e \leq 2$, then the conflict graph has treewidth at most two (proof omitted). Since width-two tree decompositions can be constructed in linear time [2] and weighted Vertex Cover can be solved in linear time on graphs with constant treewidth [10], this yields linear-time solvability for Weighted Maximum Tree Orientation with $m_e \leq 2$.

**Theorem 4.** *If $m_e \leq 2$, then* Weighted Maximum Tree Orientation *can be solved in linear time.*

We can further prove that for $m_e \geq 3$, MTO is NP-hard even on stars, that is, on trees where all leaves are attached to the same vertex. The proof is by reduction from MaxDiCut.

**Theorem 5.** Maximum Tree Orientation *on stars with $m_e \geq 3$ is NP-complete.*

## 5 Observations on Protein Networks

The goal in this section is to explore the space of practically meaningful parameterizations, here focusing on biological applications. We first performed experiments based on the same data as used by Medvedovsky et al. [9]. The network is a yeast protein–protein interaction network from the Database of Interacting Proteins (DIP) [12], containing 4 737 vertices and 15 147 edges. The cause–effect pairs were obtained from gene knockout experiments by Yeang et al. [15] and contain 14 502 pairs. After discarding small connected components and contracting cycles, we obtained a tree with 1 278 vertices and 5 569 pairs.[5]

The resulting tree is, as already observed by Medvedovsky et al. [9], very star-like: there is one vertex of degree 1151 and 1048 degree-one vertices attached to it. The remaining 229 vertices have degree 1 to 4. All paths connecting cause–effect pairs pass through the central vertex.

We first note that this MTO instance is actually fairly easy to solve exactly. The Integer Linear Program (ILP) by Medvedovsky et al. [9, Sect. 3.1] and Vertex Cover on the conflict graph (see Section 2) solved by either an ILP or a simple branching strategy with data reduction all solve the instance in less than a second.[6] The branching strategy finds a vertex $v$ of maximum degree and branches into the two cases of taking $v$ into the vertex cover or taking all neighbors of $v$ into the vertex cover. Before each branch, degree-1 vertices are eliminated by taking their neighbor into the vertex cover. The search in the second branch is cut short when the accumulated vertex cover is larger than that of the first branch.

The reason that these strategies work so well is probably due to the low value of the parameter $k$: only 77 cause–effect pairs cannot be satisfied. This limits the size of the branch-and-bound tree that underlies all three methods.

In Table 1, we examine several other parameters. Since there are still $p = 5\,569$ pairs left, using this parameter for a fixed-parameter algorithm seems infeasible. Unfortunately, since all paths run through a single vertex, the parameter $m_v$ is not any more useful. Only about 5 % of the pairs are cross pairs, so $q$ is already a more promising parameter. However, with a value of $q = 417$, direct application of Theorem 3, with a worst-case running time bound of $O(n^3 + 2^q \cdot (|P| + n^2))$ seems not practical. Even if we eliminate pairs that do not conflict with any other pairs, leaving only $n_c = 1\,287$ pairs, we still find at least 306 cross pairs (parameter $q'$). Again, because all paths run through a single vertex, considering cross pairs per vertex does not help. In summary, for this particular instance the number of unsatisfiable pairs $k$ is clearly the most useful parameter.

[5] These numbers differ slightly from the ones stated by Medvedovsky et al. [9]. We do not use the additional kinase–substrate data, which is only meaningful to evaluate the orientations obtained, and requires an arbitrary parameter choice not documented by Medvedovsky et al. [9].

[6] The running times are 0.09 s, 0.02 s, and 0.13 s, respectively, on a 2.67 GHz Intel Xeon W3520 machine, using GLPK 4.44 for the ILPs, and with the branching strategy implemented in Objective Caml.

**Table 1.** Values for various parameters for the protein interaction network instance from Medvedovsky et al. [9].

| Parameter | | Value |
|---|---|---|
| $n$ | Number of network vertices | 4 654 |
| $m$ | Number of network edges | 15 104 |
| $p$ | Number of pairs | 14 155 |
| $n_t$ | Vertices in MTO instance | 1 278 |
| $p_t$ | Number of pairs in MTO instance | 5 569 |
| $n^*$ | Number of vertices in star | 1 049 |
| $m_v$ | Max. number of pairs per vertex | 5 569 |
| $m_e$ | Max. number of pairs per edge | 371 |
| $q$ | Number of cross pairs | 417 |
| $q_v$ | Max. number of cross pairs per vertex | 417 |
| $q'$ | Number of cross pairs after data reduction | 306 |
| $q'_v$ | Max. number of cross pairs per vertex after data reduction | 306 |
| $n_c$ | Number of vertices in conflict graph | 1 287 |
| $m_c$ | Number of edges in conflict graph | 4 626 |
| $k$ | Number of unsatisfiable pairs | 77 |

To examine the effect of the sparseness of the input instance on the various parameters, we investigated another yeast protein interaction network assembled by Nir Yosef from various sources (see references in [3]). In this network, each edge is annotated with a probability of interaction. Thus, by thresholding, we can obtain graphs of different sparseness. The results are shown in Table 2.

We see that, here, the parameter $k$ is not always a clear winner. When the network becomes sparser, the components that will be shrunk to a single vertex by the cycle contraction will be smaller, leaving fewer pairs with both endpoints on the same tree vertex, and thereby increasing the number of potential conflicts. Only for very high thresholds, the parameter becomes small again, since then the original instance is already much smaller. Still, all instances can be solved in less than one second by the three algorithms mentioned above, which exploit low values of $k$.

We also see that for denser graphs, the parameter values based on the number of cross pairs are quite low, e. g. $q'_v = 3$ for the whole graph. Thus, it seems very likely that these instances can be quickly solved by the algorithm from Theorem 3, running in $O(2^{q'_v} \cdot n^2 \cdot q'_v)$ time. One possible explanation for the low value for these parameters is that the networks exhibit a linear structure. For example, if each protein can be assigned a distance to the nucleus, and interactions mostly transport information to or from the nucleus, then we would expect to have only few cross pairs.

The parameter $m_v$ could be expected to be not too high in biological networks, since otherwise this would make the network less robust, since elimination of one vertex would disrupt too many paths. However, one vertex in the tree under consideration can actually correspond to a very large component in

**Table 2.** Parameters for the largest connected component of the protein interaction network assembled by Nir Yosef [3] with different thresholds for the edge probability. The uneven gaps in the sizes of the instances are because many edges have identical weights.

| threshold | $n$ | $m$ | $p$ | $n_t$ | $p_t$ | $n^*$ | $m_v$ | $m_e$ | $q$ | $q_v$ | $q'$ | $q'_v$ | $n_c$ | $m_c$ | $k$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.000000 | 5385 | 39921 | 14393 | 799 | 2014 | 750 | 2014 | 59 | 7 | 7 | 3 | 3 | 115 | 292 | 17 |
| 0.154420 | 4530 | 35041 | 11522 | 747 | 2203 | 705 | 2203 | 298 | 27 | 27 | 20 | 20 | 475 | 1632 | 40 |
| 0.371369 | 4254 | 32135 | 10740 | 796 | 2443 | 749 | 2443 | 275 | 47 | 47 | 35 | 35 | 528 | 2424 | 46 |
| 0.573290 | 3871 | 27128 | 9445 | 777 | 2225 | 704 | 2225 | 268 | 32 | 32 | 13 | 13 | 140 | 311 | 32 |
| 0.573313 | 2546 | 8977 | 5279 | 638 | 2311 | 477 | 2310 | 208 | 252 | 252 | 151 | 151 | 561 | 2394 | 68 |
| 0.830093 | 2206 | 7136 | 4346 | 643 | 2206 | 449 | 2206 | 192 | 304 | 304 | 193 | 193 | 727 | 4017 | 83 |
| 0.886308 | 1407 | 3646 | 1607 | 441 | 787 | 260 | 785 | 45 | 106 | 106 | 88 | 88 | 311 | 1876 | 75 |
| 0.943001 | 1135 | 3069 | 920 | 361 | 464 | 195 | 463 | 32 | 57 | 57 | 42 | 42 | 179 | 801 | 44 |
| 0.954421 | 1039 | 2504 | 843 | 350 | 489 | 175 | 461 | 45 | 85 | 73 | 71 | 61 | 215 | 3001 | 81 |
| 0.957338 | 895 | 2060 | 681 | 304 | 405 | 119 | 375 | 39 | 64 | 54 | 58 | 50 | 240 | 3092 | 89 |
| 0.965986 | 874 | 2018 | 666 | 299 | 477 | 103 | 411 | 165 | 90 | 78 | 85 | 75 | 358 | 12284 | 110 |
| 0.984753 | 668 | 1676 | 312 | 206 | 163 | 95 | 162 | 20 | 7 | 7 | 6 | 6 | 55 | 222 | 15 |
| 0.989212 | 581 | 1322 | 188 | 192 | 167 | 69 | 161 | 86 | 24 | 24 | 24 | 24 | 141 | 1088 | 32 |
| 0.989233 | 307 | 681 | 71 | 121 | 70 | 32 | 66 | 36 | 21 | 21 | 11 | 11 | 52 | 219 | 7 |
| 0.990409 | 294 | 666 | 28 | 114 | 27 | 26 | 26 | 21 | 2 | 2 | 2 | 2 | 9 | 8 | 2 |

the original graph, which weakens this effect. Therefore, this parameter is more useful in sparser graphs, where not too many graph vertices are joined into a tree vertex. However, for the given instances, it seems small enough to be exploited only for fairly small instances, where other parameters would give good results, too.

The parameter $m_e$ could similarly be expected to be low in sparse networks; however, the NP-hardness result already for $m_e \geq 3$ (Theorem 5) makes practical use of this parameter unlikely.

## 6 Conclusion

We started a parameterized complexity analysis of (Weighted) Maximum Tree Orientation, obtaining a more fine-grained view on the computational complexity of this NP-hard problem. In this line, there are still several challenges for future investigations. For instance, in the spirit of "distance-from-triviality parameterization" [7,11] it would be interesting to study the parameterized complexity of MTO with respect to the parameter "number of all possible pairs minus the number of input pairs"—recall that for parameter value zero MTO is polynomial-time solvable [8]. MTO restricted to stars is still NP-hard, but then at least one quarter of all input pairs can always be satisfied [9]. Hence, it would be interesting to study above guarantee parameterization [10,11] with respect to the number of satisfied pairs. MTO can be translated into a vertex covering problem (see Proposition 1) on a graph class that is $K_4$-free—this motivates to study

whether vertex covering on this graph class can be done faster than on general graphs. Clearly, MTO brings along numerous further parameters and parameter combinations which can make a more comprehensive multivariate complexity analysis [11] very attractive. Often, it is desirable to not only list a single solution, but to enumerate all optimal solutions. Our dynamic-programming-based algorithms seem suitable for this. Following Gamzu et al. [6] and extending the studies for MTO as pursued here to the more general case of mixed graphs with partially already oriented edges is of high interest. First steps in this direction have very recently been undertaken by Silverbush et al. [14]. Finally, it seems promising to examine the parameters based on cross pairs in other networks such as communication networks, and to try to apply the concept to other hard network problems.

## References

1. Alm, E., Arkin, A.P.: Biological networks. Current Opinion in Structural Biology 13(2), 193–202 (2003)
2. Arnborg, S., Proskurowski, A.: Characterization and recognition of partial 3-trees. SIAM Journal on Algebraic and Discrete Methods 7(2), 305–314 (1986)
3. Bruckner, S., Hüffner, F., Karp, R.M., Shamir, R., Sharan, R.: Topology-free querying of protein interaction networks. Journal of Computational Biology 17(3), 237–252 (2010)
4. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
5. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
6. Gamzu, I., Segev, D., Sharan, R.: Improved orientations of physical networks. In: Proc. 10th WABI. LNBI, vol. 6293, pp. 215–225. Springer (2010)
7. Guo, J., Hüffner, F., Niedermeier, R.: A structural view on parameterizing problems: distance from triviality. In: Proc. 1st IWPEC. LNCS, vol. 3162, pp. 162–173. Springer (2004)
8. Hakimi, S.L., Schmeichel, E.F., Young, N.E.: Orienting graphs to optimize reachability. Information Processing Letters 63(5), 229–235 (1997)
9. Medvedovsky, A., Bafna, V., Zwick, U., Sharan, R.: An algorithm for orienting graphs based on cause-effect pairs and its applications to orienting protein networks. In: Proc. 8th WABI. LNBI, vol. 5251, pp. 222–232. Springer (2008)
10. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. No. 31 in Oxford Lecture Series in Mathematics and Its Applications, Oxford University Press (2006)
11. Niedermeier, R.: Reflections on multivariate algorithmics and problem parameterization. In: Proc. 27th STACS. Leibniz International Proceedings in Informatics, vol. 5, pp. 17–32. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2010)
12. Salwinski, L., Miller, C.S., Smith, A.J., Pettit, F.K., Bowie, J.U., Eisenberg, D.: The database of interacting proteins: 2004 update. Nucleic Acids Research 32(Database issue), D449–D451 (2004)
13. Sharan, R., Ideker, T.: Modeling cellular machinery through biological network comparison. Nature Biotechnology 24, 427–433 (April 2006)
14. Silverbush, D., Elberfeld, M., Sharan, R.: Optimally orienting physical networks. In: Proc. 15th RECOMB. LNCS, Springer (2011), to appear.
15. Yeang, C.H., Ideker, T., Jaakkola, T.: Physical network models. Journal of Computational Biology 11(2–3), 243–262 (2004)