A Note on the Parameterized Complexity of Unordered Maximum Tree Orientation

Sebastian Böcker
Lehrstuhl für Bioinformatik
Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 2, 07743 Jena, Germany
sebastian.boecker@uni-jena.de

.

Peter Damaschke*
Department of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
ptr@chalmers.se

Abstract

In the UNORDERED MAXIMUM TREE ORIENTATION problem, a set P of paths in a tree and a parameter k is given, and we want to orient the edges in the tree such that all but at most k paths in P become directed paths. This is a more difficult variant of a well-studied problem in computational biology where the directions of paths in P are already given. We show that the parameterized complexity of the unordered version is between EDGE BIPARTIZATION and VERTEX BIPARTIZATION, and we give a characterization of orientable path sets in trees by forbidden substructures, which are cycles of a certain kind.

Keywords: graph orientation, directed path, parameterized reduction, bipartization

1 Introduction

Consider an undirected graph. To orient an edge uv means to decide on one direction, from u to v or vice versa. Edges can be oriented independently. Given an orientation of all edges, a path in the graph is called directed if we can traverse this path in one direction, following the orientations of its edges. For convenience we will use the terms *node* and *vertex* in a tree and in a general graph, respectively. Define the following problem:

MAXIMUM GRAPH ORIENTATION: Given is an undirected graph, a set P of n ordered pairs of vertices, and a parameter $k < n$. Delete at most k pairs in P and orient the edges in the graph such that, for each remaining pair $(u, v) \in P$, some directed path goes from u to v.

P can also be a multiset, that is, a pair (u, v) may appear several times in P. Then we do not demand several distinct paths from u to v, but the multiplicity of (u, v) serves as a weight

*Corresponding author. Tel. 0046-31-772-5405. Fax 0046-31-772-3663.

when it comes to deletions. For notational convenience we will speak of a set P, although all considerations carry over to multisets.

A motivation of the problem is the inference of signal transmissions in protein-protein interaction networks, based on cause-effect pairs. For example, if an experiment that changes the amount of a protein u causes also a change at v, some information must flow from u to v. Since a cycle can always be oriented, we can successively shrink cycles to new vertices until a tree remains, thereby preserving the solution space. Thus it suffices to consider the problem for trees, see [8].

Maximum Tree Orientation: Given is a tree, a set P of n ordered pairs of nodes of the tree, and a parameter $k < n$. Delete at most k pairs in P and orient the edges in the tree such that, for each remaining pair $(u, v) \in P$, some directed path goes from u to v.

Maximum Tree Orientation is NP-hard [8], and several approximation results have also been derived there. Another standard direction of research on hard problems is the computation of optimal solutions by mildly exponential or parameterized algorithms. We state a few basic notions and refer to [4, 9] for general introductions. In exponential time bounds we use the O^* notation that omits polynomial factors. A problem is fixed-parameter tractable (FPT) if it can be solved in $p(n) \cdot f(k) = O^*(f(k))$ time where p is a polynomial and f any computable function.

Maximum Tree Orientation can be reduced to the classical FPT problem Vertex Cover [3]: Construct a graph $G = (P, E)$ with P as vertex set, and with edges between any two vertices whose corresponding tree paths in P have conflicting orientations; equivalence is obvious. Hence Maximum Tree Orientation is in FPT, and it can be solved in "Vertex Cover time" which is much faster than $O^*(2^k)$, see [2] for the latest bound.

In the present paper we study a more difficult problem:

Unordered Maximum Graph Orientation: Given is an undirected graph, a set P of n unordered pairs of vertices, and a parameter $k < n$. Delete at most k pairs in P and orient the edges in the graph such that, for each remaining pair $(u, v) \in P$, some directed path goes from u to v, or in the opposite direction.

Here we have to decide not only on the deleted pairs but also on the orientations of pairs that are kept. This unordered version comes into mind naturally, and it has a plausible motivation as well: In the protein-protein interaction network application, the data may be expression profiles, such that one can observe that certain pairs are in correlation, but one cannot see what is the cause and the effect.

Still the unordered problem inherits some basic properties of the ordered counterpart. Exactly as in the ordered case, we can always orient cycles and successively shrink them to new vertices, thereby preserving the solution space. Therefore, the actual problem to consider is the following.

Unordered Maximum Tree Orientation: Given is a tree T, a set P of n unordered pairs of nodes of T, and a parameter $k < n$. Delete at most k pairs in P and orient the edges in T such that, for each remaining pair $(u, v) \in P$, the (u, v)-path in T is a directed path from u to v, or in the opposite direction.

Since any pair of nodes in a tree is connected by a unique path, we will henceforth consider P as a set of paths, and the problem is to orient all but k paths in P by edge orientations.

Trivially, UNORDERED MAXIMUM TREE ORIENTATION can be solved in $O^*(3^n)$ time: Decide for each path in P one orientation or decide not to orient the path, and then check consistency. A little more thinking yields:

Proposition 1 UNORDERED MAXIMUM TREE ORIENTATION *is solvable in* $O^*(2^n)$ *time.*

Proof. We process the paths one-by-one in a certain order and decide for each path whether to orient it or not. It always suffices to consider one orientation: For the first path, both orientations lead to symmetric solutions, hence we can take either one. In the following steps we always take a path from P that intersects paths that are already oriented, hence only one possible orientation is consistent. If no such path exists, then we start a new connected component, and again the first orientation is arbitrary. $\diamond$

Essentially the same argument shows:

Proposition 2 UNORDERED MAXIMUM TREE ORIENTATION *with* $k = 0$ *is solvable in polynomial time.* $\diamond$

Triviality for $k = 0$ provokes the question whether UNORDERED MAXIMUM TREE ORIENTATION is also fixed-parameter tractable in parameter k, like the ordered version. In this paper we give an affirmative answer. More specifically, we show that its complexity is between two established FPT graph problems:

EDGE BIPARTIZATION: Given a graph and a parameter k, delete at most k edges such that the graph becomes bipartite.

VERTEX BIPARTIZATION: Given a graph and a parameter k, delete at most k vertices such that the graph becomes bipartite.

The following statements are well known to be equivalent:

- G is a bipartite graph.
- The vertices of G can be colored white and black such that any two adjacent vertices have distinct colors.
- G does not contain odd cycles.

Due to the last property, solutions to VERTEX BIPARTIZATION are also known as *odd cycle transversals.*

We will give simple parameterized reductions from EDGE BIPARTIZATION to UNORDERED MAXIMUM TREE ORIENTATION, and from the latter problem to VERTEX BIPARTIZATION. Since VERTEX BIPARTIZATION is solvable in $O^*(3^k)$ time [11, 7], so is UNORDERED MAXIMUM TREE ORIENTATION. The best known FPT algorithm for EDGE BIPARTIZATION, based on iterative compression, runs in $O^*(2^k)$ time [6]. Besides these implicit complexity bounds for UNORDERED MAXIMUM TREE ORIENTATION, we also give structural characterizations of path sets in trees that can be oriented.

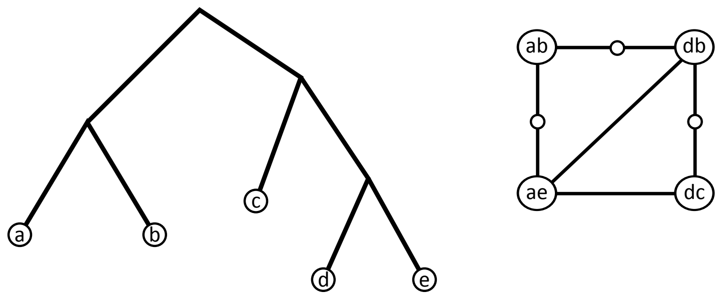

Figure 1: Transformation of an Unordered Maximum Tree Orientation instance into a graph. The pairs from $P = \{ab, ae, db, dc\}$ have been arbitrarily ordered as $a \to b, d \to b, a \to e, d \to c$. Dummy vertices have been inserted between the vertices in G representing non-conflicting paths. The resulting graph is not bipartite, indicating that P is not orientable.

2 Unordered Maximum Tree Orientation versus Bipartization Problems

We call a set P of paths in a tree *orientable* if Unordered Maximum Tree Orientation has a solution with $k = 0$. Proposition 2 says that this property can be checked in polynomial time. Now we give a characterization of orientable path sets in terms of an auxiliary graph. A similar construction was used in [1] for a graph drawing problem. Interestingly, the auxiliary graph is not "canonical" but depends on some arbitrary initial decisions, however this will not cause any problems for its use.

Given a set P of paths in a tree, we construct a graph $G = (V, E)$ as follows. Every path in P is represented by a vertex of V. (We will sometimes identify paths in P and their corresponding vertices in V, without risk of confusion.) We assign arbitrary (!) orientations to the paths, that is, each path with end nodes u and v, say, is directed either from u to v, or from v to u. This should induce an orientation on the tree edges in an obvious sense, however, it is also obvious that the given orientations can be in conflict. Note that, for any two paths in P that intersect in at least one edge, the orientations on *all* edges in the intersection either accord or conflict; a mixed case is not possible. Now, if two intersecting paths are in conflict, we connect them in G by an edge. If two intersecting paths are in accordance, we connect them in G by an edge as well, but we subdivide this edge by a dummy vertex. We call the non-dummy vertices *regular*. See Fig. 1 for an example.

Lemma 3 *The set P is orientable if and only if G (constructed from any initial orientation of the paths in P) is bipartite.*

Proof. Consider an orientation of the tree edges that induces an orientation of P, in the obvious sense. We color a path white if its orientation is the original one, and we color a path black if its orientation has been switched. We claim that this yields a white-black coloring of G, hence G is bipartite. For any two regular vertices of V that are adjacent, the edge between them was created because they were in conflict. But since P is now oriented, exactly one of the two orientations has been switched, hence the two vertices are white and black. Any further edge in G connects a pair of regular vertices, that represent according paths in P, with their dummy vertex. Since the two vertices are still in accordance, either none or both of the orientations have been switched, thus they have the same color, and we can assign the other color to the dummy vertex.

For the reverse direction consider a white-black coloring of G. We switch the orientation of all black paths, and we claim that this resolves all orientation conflicts. Any two intersecting paths that were in conflict are joined by an edge in G, hence they have different colors, hence exactly one orientation is switched. Any two intersecting paths that were in accordance have distance 2 as vertices in G, hence they have the same color, hence none or both of their orientations are switched, and no new conflict is introduced. $\diamond$

Theorem 4 *There is a parameter-preserving reduction from* Unordered Maximum Tree Orientation *to* Vertex Bipartization*, thus the former problem can be solved in $O^*(3^k)$ time.*

Proof. Given a set P of paths in a tree and parameter k, we construct a graph G as explained prior to Lemma 3. Then we run the $O^*(3^k)$ time Vertex Bipartization algorithm from [11] on G. Suppose it finds a solution with at most k vertex deletions. Wherever the solution deletes a dummy vertex, we re-insert it and delete a neighbored regular vertex instead. Obviously this does not increase the number of vertex deletions, and the resulting graph remains bipartite: A graph is bipartite if and only if no induced odd cycles exist, moreover, all odd cycles destroyed by removal of a dummy vertex are destroyed as well by removal of a neighbor. Now only regular vertices are deleted. Finally we remove from P the, at most k, corresponding paths. Since the rest of G is bipartite, the rest of P is orientable due to Lemma 3, and we can construct an orientation in polynomial time due to Proposition 2. Conversely, if P minus k selected paths is orientable, then G minus k selected vertices is bipartite, also by Lemma 3. $\diamond$

As an implicit lower bound we establish:

Proposition 5 *There is a parameter-preserving reduction from* Edge Bipartization *to* Maximum Tree Orientation*.*

Proof. Given a graph with p vertices and parameter k, i.e., an instance of Edge Bipartization, we construct a tree that is merely a star, consisting of a central node and p leaves attached. These leaves represent the p vertices, and every edge of the graph is represented as a path between the leaves. We show that the graph is bipartite if and only if this path system P is orientable. Clearly, this also establishes equivalence of the parameterized problems. It suffices to observe the following chain of equivalent statements:

- The graph is bipartite.
- The vertices can be colored white and black, such that adjacent vertices have distinct colors.
- The tree edges can be oriented outwards and inwards such that every path in P is composed of an outwards and an inwards edge.
- P is orientable. $\diamond$

3 An Explicit Characterization of Orientable Path Sets

Lemma 3 gave us an FPT algorithm for Maximum Tree Orientation, but the characterization itself may appear somewhat unsatisfactory, as the auxiliary graph has some arbitrariness. Here we give a more canonical "forbidden substructure" characterization of orientable path sets, in terms of the paths only. It is not only aesthetic but may also serve as a basis for alternative FPT algorithms in future research (although, admittedly, we cannot provide an algorithmic application right now).

Definition 6 *An inner node v of a path p splits p in two subpaths that we call the* rays *of p, with respect to this fixed v. (An inner node of a path is any node except the two endnodes.) Consider a cyclic sequence $(p_0, \ldots, p_{k-1})$ of paths in a tree. In the following, addition and subtraction of indices is understood modulo k, hence $(k-1)+1=0$. We call $(p_0, \ldots, p_{k-1})$ a* star cycle *with center v if:*

- *Node v is an inner node of every p_i.*
- *For all i, the intersection $p_i \cap p_{i+1}$ of paths p_i and p_{i+1} is a path (of at least one edge) with v as one end node.*
- *For all i, the intersections $p_i \cap p_{i-1}$ and $p_i \cap p_{i+1}$ are contained in the two distinct rays of p_i.*
- *For all i, j that are not cyclic neighbors, that is, neither $j = i+1$ nor $j = i-1$ modulo k, only the center v is in $p_i \cap p_j$. ("The cycle is induced.")*

Theorem 7 *P is orientable if and only if P has no odd star cycles.*

Proof. An odd star cycle is not orientable, since the rays must be directed towards the center and away from the center alternatingly, which is impossible in an odd cycle.

For the reverse direction, ironically, we use Lemma 3. So let G be a graph as used there, and P not orientable. By Lemma 3, G is not bipartite, hence G contains an odd cycle C. A chord in a cycle C is an edge connecting two vertices being not consecutive in C. Note that vertices involved in a chord are always regular. A pseudochord is a subdivided edge, i.e., path of two edges linked by a dummy vertex, connecting two regular vertices being not consecutive in C. If C has a chord or pseudochord, we can split C in two cycles one of which, denoted C', is odd again. In the case of a chord, C' is shorter than C. In the case of a pseudochord, C' may still be as long as C, but then C' has one more dummy vertex than C, since the vertex in $C - C'$ was regular. (Any two regular vertices have at most one dummy vertex between them.) Thus, after finitely many iterations we obtain an odd cycle C^* with neither chords nor pseudochords. For the tree paths (regular vertices) in C^* this means that only consecutive regular vertices in C^* (possibly with one dummy vertex in between) intersect as paths. This property together with the geometry of a tree implies that C^* is also a star cycle.

To see this last conclusion, first observe that C^* has at least three regular vertices. If C^* has exactly $k=3$ regular vertices, then the three corresponding tree paths either form a star cycle or they have a common edge. But in the latter case every possible orientation makes C^* an even cycle (thanks to the dummy vertices), which contradicts the choice of C^*. Now consider any cyclic sequence $(p_0, \ldots, p_{k-1})$ of $k > 3$ paths in the tree, such that any two cyclic neighbors (and only cyclic neighbors) intersect in a path, that is, share at least one edge. Observe that p_0, p_1, p_2 must have a common node v which is also an endpoint of both $p_0 \cap p_1$ and $p_1 \cap p_2$, and these two intersections are in different rays of p_1. Otherwise some edge e of p_1 would separate p_0 and p_2, but then the remaining paths p_i, which must not contain e, would not be able to connect p_0 and p_2. Similarly p_1, p_2, p_3 have a common node, and this is the same v, otherwise p_3 would be separated from p_0; and so on. Traversing the cyclic sequence we thus establish the structure as described in Definition 6. $\diamond$

4 Further Research

Vertex Bipartization, and hence Unordered Maximum Tree Orientation, is solvable in $O^*(3^k)$ time. In Section 1 we also mentioned a trivial $O^*(2^n)$ time algorithm for Unordered Maximum Tree Orientation. Due to the smaller base this would be faster than the FPT algorithm when k/n is above some threshold. But the obvious question is whether such problem instances exist at all. If k denotes the optimal solution value, how large can k/n be? An observation is that k/n can be $0.5 - \epsilon$ for arbitrarily small $\epsilon > 0$. Just consider stars with p leaves pairwise connected by $\binom{p}{2}$ paths. But are there also computationally hard instances with large k/n? And is there a general upper bound for k/n, especially, some constant $c < 1$ such that $k/n \leq c$ for all path sets in all trees?

The $O^*(2^n)$ time algorithm in Section 1 also enumerates all possible orientations. Is it possible to enumerate all orientations with at most k deletions in FPT time?

On another front, the simple $O^*(2^n)$ time bound for Unordered Maximum Tree Orientation can be slightly improved. Using the general technique from [10] (their Theorem 16), one can combine the trivial algorithm and the FPT algorithm to reduce the base 2 slightly. One could even try to get much faster exact algorithms for Unordered Maximum Tree Orientation via the reduction to Vertex Bipartization. Note that Vertex Bipartization can be solved in $O^*(1.62^n)$ time [10], but the catch is that n is the number of vertices in the graph, which can be much higher than the given number of tree paths, as we may have many dummy vertices. Also, it seems that the method in [10] (enumeration of maximal independent sets) cannot be easily adapted to our reduction graphs with many dummy vertices, without destroying the time bound. Anyhow, it remains open whether there exist instances where any such improvement would be relevant, hence the relation of k and n is the primary question.

Next, Unordered Maximum Tree Orientation (parameterized) could be easier than Vertex Bipartization in general, as our reduction graphs may form a special class of graphs with algorithmically useful properties. Additional parameters (as for the ordered case in [3]) and mixed graphs where some edges are already oriented [5] could be interesting as well.

Acknowledgments

The second author has been supported by the Swedish Research Council (Vetenskapsrådet), grant 2010-4661, "Generalized and fast search strategies for parameterized problems". We are indebted to Falk Hüffner and Rolf Niedermeier for drawing our attention to their paper [3] and pointing out a mishap in an early draft.

References

[1] S. Böcker, F. Hüffner, A. Truss, M. Wahlström: A faster fixed-parameter approach to drawing binary tanglegrams, in: J. Chen, F.V. Fomin (Eds.), IWPEC 2009, LNCS 5917, Springer, Heidelberg, 2009, pp. 38–49.

[2] J. Chen, I.A. Kanj, G. Xia: Improved upper bounds for vertex cover, Theor. Comput. Sci. 411 (2010) 3736–3756.

[3] B. Dorn, F. Hüffner, D. Krüger, R. Niedermeier, J. Uhlmann, Exploiting bounded signal flow for graph orientation based on cause-effect pairs, Algor. for Mol. Biol. 6:21 (2011).

[4] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer, New York, 1999.

[5] M. Elberfeld, D. Segev, C.R. Davidson, D Silverbush, R. Sharan: Approximation algorithms for orienting mixed graphs, in: R. Giancarlo, G. Manzini (Eds.), CPM 2011, LNCS 6661, Springer, Heidelberg, 2011, pp. 416–428.

[6] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, S. Wernicke, Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization, J. Comp. Syst. Sci. 72 (2006) 1386–1396.

[7] D. Lokshtanov, S. Saurabh, S. Sikdar, Simpler parameterized algorithm for OCT, in: J. Fiala, J. Kratochvíl, M. Miller (Eds.), IWOCA 2009, LNCS 5874, Springer, Heidelberg, 2009, pp. 380–384.

[8] A. Medvedovsky, V. Bafna, U. Zwick, R. Sharan, An algorithm for orienting graphs based on cause-effect pairs and its applications to orienting protein networks, in: K.A. Crandall, J. Lagergren (Eds.), WABI 2008, LNCS 5251, Springer, Heidelberg, 2008, pp. 222–232.

[9] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford Lecture Series in Math. and its Appl., Oxford Univ. Press, 2006.

[10] V. Raman, S. Saurabh, S. Sikdar, Efficient exact algorithms through enumerating maximal independent sets and other techniques, Theory Comput. Syst. 41 (2007) 563–587.

[11] B. Reed, K. Smith, A. Vetta, Finding odd cycle transversals, Oper. Res. Letters 32 (2004) 299–301.