

On Finding Optimal Polytrees

Serge Gaspers

The University of New South Wales and
Vienna University of Technology
gaspers@kr.tuwien.ac.at

Mikko Koivisto

University of Helsinki
mikko.koivisto@cs.helsinki.fi

Mathieu Liedloff

Université d'Orléans
mathieu.liedloff@univ-orleans.fr

Sebastian Ordyniak

Vienna University of Technology
ordyniak@kr.tuwien.ac.at

Stefan Szeider

Vienna University of Technology
stefan@szeider.net

Abstract

Inferring probabilistic networks from data is a notoriously difficult task. Under various goodness-of-fit measures, finding an optimal network is NP-hard, even if restricted to polytrees of bounded in-degree. Polynomial-time algorithms are known only for rare special cases, perhaps most notably for branchings, that is, polytrees in which the in-degree of every node is at most one. Here, we study the complexity of finding an optimal polytree that can be turned into a branching by deleting some number of arcs or nodes, treated as a parameter. We show that the problem can be solved via a matroid intersection formulation in polynomial time if the number of deleted arcs is bounded by a constant. The order of the polynomial time bound depends on this constant, hence the algorithm does not establish fixed-parameter tractability when parameterized by the number of deleted arcs. We show that a restricted version of the problem allows fixed-parameter tractability and hence scales well with the parameter. We contrast this positive result by showing that if we parameterize by the number of deleted nodes, a somewhat more powerful parameter, the problem is not fixed-parameter tractable, subject to a complexity-theoretic assumption.

1 Introduction

There has been extensive research on learning probabilistic networks from data by maximizing some suitable scoring function. Edmonds (1967) gave an efficient algorithm for the class of *branchings*, that is, directed forests with in-degree at most one; the algorithm was discovered independently by Chu and Liu (1965), and it has been later simplified and expedited by others (Bock 1971; Camerini, Fratta, and Maffioli 1979; Fulkerson 1974; Gabow et al. 1986; Gabow, Galil, and Spencer 1989; Karp 1971; Tarjan 1977). Chickering (1996) showed that for general directed acyclic graphs, DAGs, the problem is NP-hard even if the in-degree is at most two. Motivated by this gap, Dasgupta (1999) asked for a network class that is more general than branchings yet admitting provably good structure-learning algorithms; his findings concerning *polytrees*, that is, DAGs without undirected cycles, were however rather negative, showing that the optimization problem is NP-hard even if the in-degree is at most two.

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Given the recent advances in exact exponential algorithms in general (see, e.g., the book by Fomin and Kratsch (2010)), and in finding optimal DAGs in particular, it is natural to ask, whether “fast” exponential-time algorithms exist for finding optimal polytrees. For general DAGs the fastest known algorithms run in time within a polynomial factor of 2^n , where n is the number of nodes (Koivisto and Sood 2004; Ott and Miyano 2003; Parviainen and Koivisto 2009; Silander and Myllymäki 2006). However, it is not clear, whether even these bounds can be achieved for polytrees; a brute-force algorithm would visit each polytree one by one, whose number scales as the number of directed labelled trees $n^{n-2}2^{n-1}$ (Cayley 1889). Do significantly faster algorithms exist? Does the problem become easier if only a small number of nodes are allowed an in-degree larger than one?

In this work, we take a first step towards answering these questions by considering polytrees that differ from branchings by only a few arcs. More precisely, we study the problem of finding an optimal *k-branching*, defined as a polytree that can be turned into a branching by deleting k arcs. We make the standard assumption that the scoring function decomposes into a sum of local scores; see the next section for precise definitions. We note that *k-branchings* generalize branchings in a different direction than the Tree-augmented Naive Bayes classifier (TAN) due to Friedman, Geiger, and Goldszmidt (1997). Namely, in a TAN the in-degree of each node is at most two, and there is a designated class node of in-degree zero, removing of which leaves a spanning tree; the tree is undirected in the sense that the symmetric conditional mutual information is employed to score arcs.

Polynomial-time result for *k-branchings* Our main result is an algorithm that finds an optimal *k-branching* in polynomial time for every constant k . (See the next section for a formal definition of the problem.) Our overall approach is straightforward: we search exhaustively over all possible sets of at most k “extra arcs”, fix the guessed arcs, and solve the induced optimization problem for branchings. Implementing this seemingly innocent algorithm, however, requires successful treatment of certain complications that arise when applying the existing matroid machinery for finding optimal branchings. In particular, one needs to control the interaction of the extra arcs with the solution from the induced subproblem.

Fixed-parameter tractability Our algorithm for the k -branching is polynomial for fixed k , but the degree of the polynomial depends on k , hence the algorithm does not scale well in k . We therefore investigate variants of the k -branching problem that admit *fixed-parameter tractability* in the sense of Downey and Fellows (1999): the running time bound is given by a polynomial whose degree is independent of the parameter, the parameter contributing a constant factor to the bound.

In particular, we show that the k -branching problem is fixed-parameter tractable if the set of arcs incident to nodes with more than one parent form a connected polytree with exactly one sink, and each node has a bounded number of potential parent sets. This result is interesting as we show that the k -branching problem remains NP-hard under these restrictions.

We complement the fixed-parameter tractability result by showing that more general variants of the k -branching problem are not fixed-parameter tractable, subject to complexity theoretic assumptions. In particular, we show that the k -branching problem is not fixed-parameter tractable when parameterized by the *number of nodes* whose deletion produces a branching.

2 The k -branching problem

A probabilistic network is a multivariate probability distribution that obeys a structural representation in terms of a directed graph and a corresponding collection of univariate conditional probability distributions. For our purposes, it is crucial to treat the directed graph explicitly, whereas the conditional probabilities will enter our formalism only implicitly. Such a graph is formalized as a pair (N, A) , where N is the *node set* and $A \subseteq N \times N$ is the *arc set*; we identify the graph with the arc set A when there is no ambiguity about the node set. A node u is said to be a *parent* of v in the graph if the arc (u, v) is in A ; we denote by A_v the set of parents of v . When our interest is in the undirected structure of the graph, we may denote by $\bar{A}$ the *skeleton* of A , that is, the set of *edges* $\{\{u, v\} : (u, v) \in A\}$. For instance, we call A a *polytree* if $\bar{A}$ is acyclic, and a *branching* if additionally each node has at most one parent.

When learning a probabilistic network from data it is customary to introduce a scoring function that assigns each graph A a real-valued score $f(A)$ that measures how well A fits the data. While there are plenty of alternative scoring functions, derived under different statistical paradigms and assumptions (Lam and Bacchus 1994; Chickering 1995; Heckerman, Geiger, and Chickering 1995; Dasgupta 1999), the most popular ones share one important property: they are *decomposable*, that is,

$$f(A) = \sum_{v \in N} f_v(A_v),$$

with some “local” scoring functions f_v . The generic computational problem is to maximize the scoring function over some appropriate class of graphs given the local scoring functions as input. Note that the score $f_v(A_v)$ need not be a sum of any individual arc weights, and that the parent set

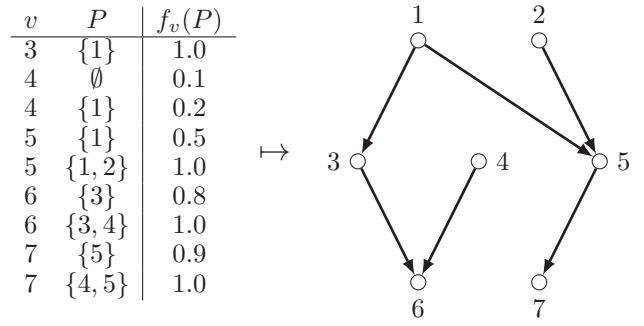


Figure 1: An optimal polytree for a given scoring function.

A_v may be empty. Figure 1 shows a table representing a local scoring function f , together with an optimal polytree.

We study this problem by restricting ourselves to a graph class that is a subclass of polytrees but a superclass of branchings. We call a polytree A a k -branching if there exists a set of at most k arcs $D \subseteq A$ such that in $A \setminus D$ every node has at most one parent. Note that any branching is a 0-branching. The k -branching problem is to find a k -branching A that maximizes $f(A)$, given the values $f_v(A_v)$ for each node v and some collection of possible parent sets $A_v \subseteq N \setminus \{v\}$.

3 An algorithm for the k -branching problem

Throughout this section we consider a fixed instance of the k -branching problem, that is, a node set N and scoring functions f_v for each $v \in N$. Thus all arcs will refer to elements of $N \times N$. We will use the following additional notation. If A is an arc set, then $H(A)$ denotes the *heads* of the arcs in A , that is, the set $\{v : (u, v) \in A\}$. If C is a set of edges, then $N(C)$ denotes the induced node set $\{u, v : \{u, v\} \in C\}$.

We present an algorithm that finds an optimal k -branching by implementing the following approach. First, we guess an arc set D of size at most k . Then we search for an optimal polytree A that contains D such that in $A \setminus D$ every node has at most one parent; in other words, $B = A \setminus D$ is an optimal branching with respect to an induced scoring function. Clearly, the set $\bar{D}$ must be acyclic. The challenge is in devising an algorithm that finds an optimal branching B that is disjoint from D while guaranteeing that the arcs in D will not create undirected cycles in the union $B \cup D$. To this end, we will employ an appropriate weighted matroid intersection formulation that extends the standard formulation for branchings.

We will need some basic facts about matroids. A *matroid* is a pair $(E, \mathcal{I})$, where E is a set of *elements*, called the *ground set*, and $\mathcal{I}$ is a collection of subsets of E , called the *independent sets*, such that

(M1) $\emptyset \in \mathcal{I}$;

(M2) if $A \subseteq B$ and $B \in \mathcal{I}$ then $A \in \mathcal{I}$; and

(M3) if $A, B \in \mathcal{I}$ and $|A| < |B|$ then there exists an $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

The *rank* of a matroid is the cardinality of its maximal independent sets. Any subset of E that is not independent

is called *dependent*. Any minimal dependent set is called a *circuit*.

The power of matroid formulations is much due to the availability of efficient algorithms (Brezovec, Cornuéjols, and Glover 1986; Edmonds 1970; 1979; Frank 1981; Iri and Tomizawa 1976; Lawler 1976) for the *weighted matroid intersection problem*, defined as follows. Given two matroids $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$, and a weight function $w: E \rightarrow \mathbb{R}$, find an $I \subseteq E$ that is independent in both matroids and maximizes the total weight of I , that is, $w(I) = \sum_{e \in I} w(e)$. The complexity of the fastest algorithm we are aware of (for the general problem) is summarized as follows.

Theorem 1 (Brezovec, Cornuéjols, and Glover, 1986). *The weighted matroid intersection problem can be solved in $O(mr(r+c+\log m))$ time, where $m = |E|$, r is the minimum of the ranks of M_1 and M_2 , and c is the time needed for finding the circuit of $I \cup \{e\}$ in both M_1 and M_2 where $e \in E$ and I is independent in both M_1 and M_2 .*

We now proceed to the specification of two matroids, $M_1(S) = (N \times N, \mathcal{I}_1(S))$ and $M_2(N \times N, \mathcal{I}_2(S))$, parametrized by an arbitrary arc set S such that $\bar{S}$ is acyclic.

The *in-degree matroid* $M_1(S)$: Let $\mathcal{I}_1(S)$ consist of all arc sets B such that no arc in B has a head in $H(S)$ and every node outside $H(S)$ is the head of at most one arc in B .

The *acyclicity matroid* $M_2(S)$: Let $\mathcal{I}_2(S)$ consist of all arc sets B such that $\bar{B} \cup S$ is acyclic.

We observe that the standard matroid intersection formulation of branchings is obtained as the special case of $S = \emptyset$: then an arc set is seen to be branching if and only if it is independent in both the in-degree matroid and the acyclicity matroid.

The next two lemmas show that $M_1(S)$ and $M_2(S)$ are indeed matroids whenever $\bar{S}$ is acyclic.

Lemma 2. *$M_1(S)$ is a matroid.*

Proof. Fix the arc set S and denote $\mathcal{I}_1(S)$ by $\mathcal{I}_1$ for short. Clearly, $\emptyset \in \mathcal{I}_1$ and if $A \subseteq B$ and $B \in \mathcal{I}_1$ then also $A \in \mathcal{I}_1$. Consequently, $M_1(S)$ satisfies (M1) and (M2). To see that $M_1(S)$ satisfies (M3) let $A, B \in \mathcal{I}_1$ with $|A| < |B|$. Because of the definition of $M_1(S)$ the sets A and B contain at most one arc with head v , for every $v \in N \setminus H(S)$. Because $|A| < |B|$ there is a node $v \in N \setminus H(S)$ such that v is the head of an arc in B but v is not the head of an arc in A . Let $e \in B$ be the arc with head v . Then $e \in B \setminus A$ and $A \cup \{e\} \in \mathcal{I}_1$. Hence, $M_1(S)$ satisfies (M3). $\square$

Lemma 3. *$M_2(S)$ is a matroid.*

Proof. Fix the arc set S and denote $\mathcal{I}_2(S)$ by $\mathcal{I}_2$ for short. Because the skeleton $\bar{S}$ is acyclic and acyclicity is a hereditary property (a graph property is called hereditary if it is closed under taking induced subgraphs) it follows that $\emptyset \in \mathcal{I}_2$ and if $A \subseteq B$ and $B \in \mathcal{I}_2$ then also $A \in \mathcal{I}_2$. Consequently, $M_2(S)$ satisfies (M1) and (M2). To see that $M_2(S)$ satisfies (M3) let $A, B \in \mathcal{I}_2$ with $|A| < |B|$. Consider the sets $A' = \overline{A \cup S}$ and $B' = \overline{B \cup S}$. Let C be a connected

subset of A' . Because both A' and B' are acyclic, it follows that the number of edges of B' with both endpoints in $N(C)$ is at most the number of edges of A' with both endpoints in $N(C)$. Because every edge in $A' \setminus \bar{S}$ corresponds to an arc in A and similarly every edge in $B' \setminus \bar{S}$ corresponds to an arc in B and $|A| < |B|$, it follows that there is an arc $e \in B \setminus A$ whose endpoints are contained in two distinct components of A' . Consequently, the set $A' \cup \{e\}$ is acyclic and hence $A \cup \{e\} \in \mathcal{I}_2$. $\square$

We now relate the common independent sets of these two matroids to k -branchings. If A is a k -branching, we call an arc set D a *deletion set* of A if D is a subset of A , contains at most k arcs, and in $A \setminus D$ every node has at most one parent.

Lemma 4. *Let A be an arc set and D a subset of A of size at most k such that no two arcs from $D' = \{ (u, v) \in A \setminus D : v \in H(D) \}$ have the same head and such that $\bar{S}$ is acyclic, where $S = D \cup D'$. We have that A is a k -branching with deletion set D if and only if $A \setminus S$ is independent in both $M_1(S)$ and $M_2(S)$.*

Proof. ($\Rightarrow$): Suppose A is a k -branching with deletion set D . Then $A \setminus D$ is a branching, which shows that every node v outside $H(S)$ has in-degree at most one in $A \setminus S$. Since by definition all arcs with a head in $H(S)$ are contained in S , no arc in $A \setminus S$ has a head in $H(S)$. Therefore, $A \setminus S$ is independent in $M_1(S)$. Since every k -branching is a polytree, $\overline{(A \setminus S) \cup S} = \bar{A}$ is acyclic, and therefore $A \setminus S$ is independent in $M_2(S)$.

($\Leftarrow$): Since $A \setminus S$ is independent in $M_2(S)$, we have that $\overline{(A \setminus S) \cup S} = \bar{A}$ is acyclic. Thus, A is a polytree. As $A \setminus S$ is independent in $M_1(S)$, every node outside $H(S)$ has in-degree at most one in $A \setminus S$ and every node from $H(S)$ has in-degree zero in $A \setminus S$. Since the head of every arc from D' is in $H(S)$ and no two arcs from D' have a common head, $(A \setminus S) \cup D' = A \setminus D$ has maximum in-degree at most one. Because $|D| \leq k$, we have that A is a k -branching with deletion set D . $\square$

The characterization of Lemma 4 enables the following algorithm for the k -branching problem. Define the weight function by letting $w(u, v) = f_v(\{u\}) - f_v(\emptyset)$ for all arcs (u, v) . Guess the arc sets D and D' , put $S = D \cup D'$, check that $\bar{S}$ is acyclic, find a maximum-weight set B that is independent in both $M_1(S)$ and $M_2(S)$; output a k -branching $A = B \cup S$ that yields the maximum weight over all guesses D and D' , where the weight of $B \cup S$ is obtained as

$$w(B) + \sum_{v \in H(S)} (f_v(S_v) - f_v(\emptyset)).$$

It is easy to verify that maximizing this weight is equivalent to maximizing the score $f(A)$. Figure 2 illustrates the algorithm for the scoring function of Figure 1.

It remains to analyze the complexity of the algorithm. Denote by n the number of nodes. For a moment, consider the arc set S fixed. To apply Theorem 1, we bound the associated key quantities: the size of the ground set is $O(n^2)$; the rank of both matroids is clearly $O(n)$; circuit detection can be performed in $O(n)$ time, by a depth-first search

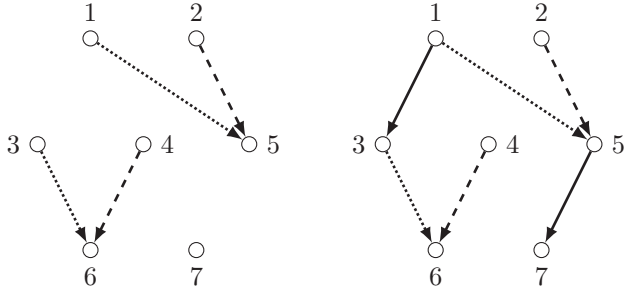


Figure 2: Left: the two guessed arc sets D (dotted) and D' (dashed). Right: the arc set A (solid) that is a heaviest common independent set of the two matroids $M_1(S)$ and $M_2(S)$.

for $M_1(S)$ and by finding a node that has higher in-degree than it is allowed to have in $M_2(S)$. Thus, by Theorem 1, a maximum-weight set that is independent in both matroids can be found in $O(n^4)$ time. Then consider the number of possible choices for the set $S = D \cup D'$. There are $O(n^{2k})$ possibilities for choosing a set D of at most k arcs such that $\overline{D}$ is acyclic. For a fixed D , there are $O(n^k)$ possibilities for choosing a subset $D' \subseteq N \times H(D)$ such that $\overline{D \cup D'}$ is acyclic and no two arcs from D' have the same head. Thus there are $O(n^{3k})$ relevant choices for the set S .

We have shown the following.

Theorem 5. *The k -branching problem can be solved in $O(n^{3k+4})$ time.*

4 Fixed-parameter tractability

Theorem 5 shows that the k -branching problem can be solved in “non-uniform polynomial time” as the order of the polynomial time bound depends on k . In this section we study the question of whether one can get k “out of the exponent” and obtain a uniform polynomial-time algorithm.

The framework of *Parameterized Complexity* (Downey and Fellows 1999) offers the suitable tools and methods for such an investigation, as it allows us to distinguish between uniform and non-uniform polynomial-time tractability with respect to a parameter. An instance of a parameterized problem is a pair (I, k) where I is the *main part* and k is the *parameter*; the latter is usually a non-negative integer. A parameterized problem is *fixed-parameter tractable* if there exist a computable function f and a constant c such that instances (I, k) of size n can be solved in time $O(f(k)n^c)$. FPT is the class of all fixed-parameter tractable decision problems. Fixed-parameter tractable problems are also called *uniform polynomial-time tractable* because if k is considered constant, then instances with parameter k can be solved in polynomial time where the order of the polynomial is independent of k (in contrast to non-uniform polynomial-time running times such as n^k).

Parameterized complexity offers a completeness theory similar to the theory of NP-completeness. One uses *parameterized reductions* which are many-one reductions where the parameter for one problem maps into the parameter for the

other. More specifically, problem L reduces to problem L' if there is a mapping R from instances of L to instances of L' such that (i) (I, k) is a yes-instance of L if and only if $(I', k') = R(I, k)$ is a yes-instance of L' , (ii) $k' \leq g(k)$ for a computable function g , and (iii) R can be computed in time $O(f(k)n^c)$ where f is a computable function, c is a constant, and n denotes the size of (I, k) . The parameterized complexity class W[1] is considered as the parameterized analog to NP. For example, the parameterized Maximum Clique problem (given a graph G and a parameter $k \geq 0$, does G contain a complete subgraph on k vertices?) is W[1]-complete under parameterized reductions. Note that there exists a trivial non-uniform polynomial-time n^k algorithm for the Maximum Clique problems that checks all sets of k vertices. $\text{FPT} \neq \text{W}[1]$ is a widely accepted complexity theoretic assumption (Downey and Fellows 1999). For example, $\text{FPT} = \text{W}[1]$ implies the (unlikely) existence of a $2^{o(n)}$ algorithm for n -variable 3SAT (Impagliazzo, Paturi, and Zane 2001; Flum and Grohe 2006). A first parameterized analysis of probabilistic network structure learning using structural parameters such as treewidth has recently been carried out by Ordyniak and Szeider (2010).

The algorithm from Theorem 5 considers $O(n^{3k})$ relevant choices for the set $S = D \cup D'$, and for each fixed choice of S the running time is polynomial. Thus, for restrictions of the problem for which the enumeration of all relevant sets S is fixed parameter tractable, one obtains an FPT algorithm. One such restriction requires that $S = D \cup D'$ is an in-tree, i.e., a directed tree where every arc is directed towards a designated root, and each node has a bounded number of potential parent sets.

Theorem 6. *The k -branching problem is fixed-parameter tractable if we require that (i) the set $S = D \cup D'$ of arcs is an in-tree and (ii) each node has a bounded number of potential parent sets.*

Proof. To compute a k -branching A , the algorithm guesses its deletion set D and the set $D' = \{(u, v) \in A \setminus D : v \in H(D)\}$. As A is a k -branching, $|D| \leq k$ and for every $v \in H(D)$ there is at most one arc in D' with head v . The algorithm first guesses the root r for the in-tree S . Then it goes over all possible choices for D and D' as follows, until D has at least k arcs.

Guess a leaf ℓ of S (initially, r is the unique leaf of S), and guess a non-empty parent set P for ℓ in A . If $|D| + |P| + 1 > k$, then backtrack. Otherwise, choose at most one arc (p, ℓ) to add to D' , where $p \in P$, and add all other arcs from a node from P to ℓ to D (if $|P| = 1$, no arc is added to D'). Now, check whether the current choice for $S = D \cup D'$ leads to a k -branching by checking whether $\overline{S}$ is acyclic and using the matroids $M_1(S)$ and $M_2(S)$ as in Theorem 5.

There are at most n choices for r . The in-tree S is expanded in at most k steps, as each step adds at least one arc to D . In each step, ℓ is chosen among at most $k + 1$ leaves, there is a constant number of choices for its parent set P and at most $k + 2$ choices for adding (or not) an arc (p, ℓ) , with $p \in P$, to D' (as $|P| \leq k + 1$). The acyclicity check for $\overline{S}$ and the weighted matroid intersection can be computed in time $O(n^4)$, leading to a total running time of $O(k^{2k} c^k n^5)$,

where c is such that every node has at most c potential parent sets. $\square$

Condition (i) in Theorem 6 may be replaced by other conditions requiring the connectivity of D or a small distance between arcs from D , giving other fixed-parameter tractable restrictions of the k -branching problem.

The following theorem shows that an exponential dependency on k or some other parameter is necessary since the k -branching problem remains NP-hard under the restrictions given above.

Theorem 7. *The k -branching problem is NP-hard even if we require that (i) the set $S = D \cup D'$ of arcs is an in-tree and (ii) each node has at most 3 potential parent sets.*

Proof. We devise a polynomial reduction from 3-SAT-2 a version of 3-SATISFIABILITY where every literal occurs at most in two clauses. 3-SAT-2 is well known to be NP-hard (Garey and Johnson 1979). Our reduction uses the same ideas as the proof of Theorem 6 in (Dasgupta 1999). Let Φ be an instance of 3-SAT-2 with clauses $C_1, \dots, C_m$ and variables $x_1, \dots, x_n$. We define the set N of nodes as follows. For every variable x_i in Φ the set N contains the nodes $p_i, x_i, x_i^1, x_i^2, \bar{x}_i^1$ and $\bar{x}_i^2$. Furthermore, for every clause C_j the set N contains the nodes p_{n+j} and C_j . Let $1 \leq i \leq n$, $1 \leq j \leq m$, and $1 \leq l \leq 2$. We set $f(C_j, x_i^l) = 1$ if the clause C_j is the l -th clause that contains the literal x_i . Similarly, we set $f(C_j, \bar{x}_i^l) = 1$ if the clause C_j is the l -th clause that contains the literal $\bar{x}_i$. We set $f(x_i, \{x_i^1, x_i^2\}) = f(x_i, \{\bar{x}_i^1, \bar{x}_i^2\}) = 1$, $f(p_1, \{x_1\}) = f(p_i, \{x_i, p_{i-1}\}) = 1$ for every $1 < i \leq n$, and $f(p_{n+j}, \{C_j, p_{n+j-1}\}) = 1$ for every $1 \leq j \leq m$. Furthermore, we set $f(v, P) = 0$ for all the remaining combinations of $v \in N$ and $P \subseteq N$. This completes the construction of N and f . Observe that every node of N has at most 3 potential parent sets. This completes our construction. We will have shown the theorem after showing the following claim.

Claim: Φ is satisfiable if and only if there is a $2n + m$ -branching D such that $f(D) \geq 2(m + n)$, the set $S = D \cup D'$ of arcs is an in-tree, and each node of N has at most 3 potential parent sets.

($\Rightarrow$) : Suppose that the formula Φ is satisfiable and let β be a satisfying assignment for Φ . Furthermore, for every $1 \leq j \leq m$ let l_j be a literal of C_j that is set to true by β . We construct a $2n + m$ -branching D as follows. For every $1 \leq j \leq m$ the digraph D contains an arc (x_i^l, C_j) if $l_j = x_i$ and C_j is the l -th clause that contains x_i and an arc $(\bar{x}_i^l, C_j)$ if $l_j = \bar{x}_i$ and C_j is the l -th clause that contains $\bar{x}_i$ for some $1 \leq i \leq n$ and $1 \leq l \leq 2$. Furthermore, for every $1 \leq i \leq n$ the digraph D contains the arcs (x_i^1, x_i) and (x_i^2, x_i) if $\beta(x_i) = \text{false}$ and the arcs $(\bar{x}_i^1, x_i)$ and $(\bar{x}_i^2, x_i)$ if $\beta(x_i) = \text{true}$. Last but not least D contains the arcs (x_i, p_i) , (C_j, p_{n+j}) and (p_l, p_{l+1}) for every $1 \leq i \leq n$, $1 \leq j \leq m$, and $1 \leq l < m + n$. Figure 3 shows an optimal $2n + m$ -branching D for some 3-SAT-2 formula. It is easy to see that D is a $2n + m$ -branching such that $f(D) = 2(m + n)$ and the set $S = D \cup D'$ of arcs is an in-tree.

($\Leftarrow$) : Suppose there is a $2n + m$ -branching D such that $f(D) \geq 2(m + n)$. Because $f(D) \geq 2(m + n)$ it fol-

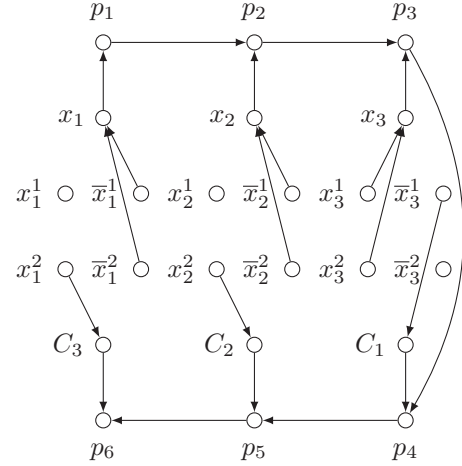


Figure 3: An optimal $2n + m$ -branching D for the formula $\Phi = C_1 \wedge C_2 \wedge C_3$ with $C_1 = x_1 \vee x_2 \vee \bar{x}_3$, $C_2 = \bar{x}_1 \vee x_2 \vee x_3$, and $C_3 = x_1 \vee \bar{x}_2 \vee \bar{x}_3$ according to the construction given in the proof of Theorem 7.

lows that every node of N achieves its maximum score in D . Hence, D has to contain the arcs (x_i, p_i) , (C_j, p_{n+j}) , (p_l, p_{l+1}) , for every $1 \leq i \leq n$, $1 \leq j \leq m$, and $1 \leq l < m + n$. For the same reasons D has to contain either the arcs (x_i^1, x_i) and (x_i^2, x_i) or the arcs $(\bar{x}_i^1, x_i)$ and $(\bar{x}_i^2, x_i)$ for every $1 \leq i \leq n$. Furthermore, for every $1 \leq j \leq m$ the $2n + m$ -branching D has to contain one arc of the form (x_i^l, C_j) or $(\bar{x}_i^l, C_j)$ where C_j is the l -th clause that contains x_i or $\bar{x}_i$, respectively, for some $1 \leq i \leq n$ and $1 \leq l \leq 2$. Let $1 \leq i \leq n$, $1 \leq j \leq m$, and $1 \leq l \leq 2$. We first show that whenever D contains an arc (x_i^l, x_i) then D contains no arc of the form (x_i^l, C_j) and similarly if D contains an arc $(\bar{x}_i^l, x_i)$ then D contains no arc of the form $(\bar{x}_i^l, C_j)$. Suppose for a contradiction that D contains an arc (x_i^l, x_i) together with an arc (x_i^l, C_j) or an arc (x_i^l, x_i) together with an arc $(\bar{x}_i^l, C_j)$. In the first case D contains the undirected cycle $(x_i^l, x_i, p_i, \dots, p_{n+j}, C_j, x_i^l)$ and in the second case D contains the cycle $(\bar{x}_i^l, x_i, p_i, \dots, p_{n+j}, C_j, \bar{x}_i^l)$ contradicting our assumption that D is a $2n + m$ -branching. It now follows that the assignment β with $\beta(x_i) = \text{true}$ if D does not contain the arcs (x_i^1, x_i) and (x_i^2, x_i) and $\beta(x_i) = \text{false}$ if D does not contain the arcs $(\bar{x}_i^1, x_i)$ and $(\bar{x}_i^2, x_i)$ is a satisfying assignment for Φ . $\square$

So far we have measured the difference of a polytree to branchings in terms of the number of arcs to be deleted. Next we investigate the consequences of measuring the difference by the number of nodes to be deleted. We call a polytree A a k -node branching if there exists a set of at most k nodes $X \subseteq A$ such that $A \setminus X$ is a branching. The k -node branching problem is to find a k -node branching A that maximizes $f(A)$. Clearly every k -branching is a k -node branching, but the reverse does not hold. In other words, the k -node branching problem generalizes the k -branching problem.

In the following we show that the k -node branching prob-

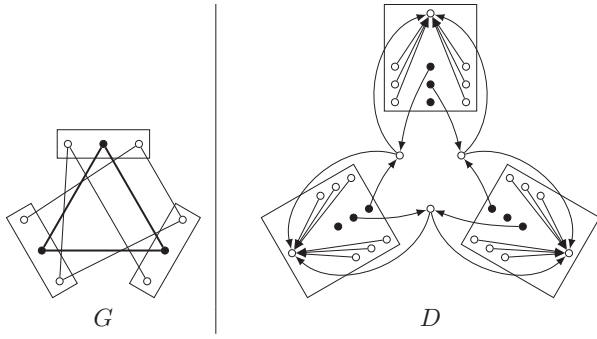


Figure 4: An example graph G ($k = 3$) together with an optimal k' -node branching D with $f(D) \geq s$ according to the construction given in the proof of Theorem 8.

lem is hard for the parameterized complexity class $W[1]$; this provides strong evidence that the problem is not fixed-parameter tractable.

Theorem 8. *The k -node branching problem is $W[1]$ -hard.*

Proof. We devise a parameterized reduction from the following problem, called Partitioned Clique, which is well-known to be $W[1]$ -complete for parameter k (Pietrzak 2003). The Instance is a k -partite graph $G = (V, E)$ with partition $V_1, \dots, V_k$ such that $|V_i| = n$ for every $1 \leq i \leq k$. The question is whether there are nodes $v_1, \dots, v_k$ such that $v_i \in V_i$ for $1 \leq i \leq k$ and $\{v_i, v_j\} \in E$ for $1 \leq i < j \leq k$? (The graph $K = (\{v_1, \dots, v_k\}, \{\{v_i, v_j\} : 1 \leq i < j \leq k\})$ is a k -clique of G .)

Let $G = (V, E)$ be an instance of this problem with partition $V_1, \dots, V_k$, $|V_1| = \dots = |V_k| = n$, and parameter k . Let $k' = \binom{k}{2} + k$, $\alpha = 1$, and $s = k'\alpha$. Let $A = \{a_{ij} : 1 \leq i < j \leq k\}$ and $A_i = \{a_{lk} \in A : l = i \text{ or } k = i\}$ for every $1 \leq i \leq k$. Then N is defined as $N = A \cup \{c_1, \dots, c_k\} \cup \{v^1, \dots, v^k : v \in V\}$. Let $V_i^w = \{v^1, \dots, v^k : v \in V_i \text{ and } v \neq w\}$. We define the score function f as follows. We set $f(c_i, A_i \cup V_i^w) = \alpha$ for every $1 \leq i \leq k$ and $w \in V_i$, and $f(a_{ij}, \{u^j, w^i\}) = \alpha$ for every $1 \leq i < j \leq k$, $u \in V_i$, $w \in V_j$, and $\{u, w\} \in E(G)$. Furthermore, we set $f(v, P) = 0$ for all the remaining combinations of v and P . This completes our construction. We will have the theorem after showing the following claim.

Claim: G has a k -clique if and only if there is a k' -node branching D such that $f(D) \geq s$.

($\Rightarrow$) : Suppose that G has a k -clique K . Then it is easy to see that the DAG D on N defined by the arc set $\{(v^j, a_{ij}), (v^j, a_{ji}) : v \in V(K) \cap V_i \text{ and } 1 \leq i, j \leq k\} \cup \{(v^i, a_{ij}), (v^i, a_{ji}) : v \in V(K) \cap V_j \text{ and } 1 \leq i, j \leq k\} \cup \{(a_{ij}, c_i) : 1 \leq i < j \leq k\} \cup \{(a_{ij}, c_j) : 1 \leq i < j \leq k\} \cup \{(v^j, c_i) : v \in V_i \setminus (\bigcup_{e \in E(K)} e) \text{ and } 1 \leq i, j \leq k\}$ is a k' -node branching and $f(D) = s$. Figure 4 shows an optimal k' -node branching D constructed from an example graph G .

($\Leftarrow$) : Suppose there is a k' -node branching D with $f(D) \geq s$. It follows that every node of D achieves its maximum score. In particular, for every $1 \leq i \leq k$ the

nodes c_i must have score α in D and hence there is a node $w_i \in V_i$ such that c_i is adjacent to all nodes in $V_i^{w_i} \cup A_i$. Furthermore, for every $1 \leq i < j \leq k$ the node a_{ij} is adjacent to exactly one node in V_i and to exactly one node in V_j . Let v_i^l be the unique node in V_i adjacent to a_{ij} and similarly let v_j^m be the unique node in V_j that is adjacent to a_{ij} for every $1 \leq i < j \leq k$. Then $w_i = v_i$ and $w_j = v_j$ because otherwise the skeleton of D would contain the cycle (v_i, a_{ij}, c_i) or the cycle (v_j, a_{ij}, c_j) . Consequently, the edges represented by the parents of a_{ij} in D for all $1 \leq i < j \leq k$ form a k -clique in G . $\square$

5 Concluding remarks

We have studied a natural approach to extend the known efficient algorithms for branchings to polytrees that differ from branchings in only a few extra arcs. At first glance, one might expect this to be achievable by simply guessing the extra arcs and solving the remaining problem for branchings. However, we do not know whether such a reduction is possible in the strict sense. Indeed, we had to take a slight detour and modify the two matroids in a way that guarantees a control for the interactions caused by the presence of high-in-degree nodes. As a result, we got an algorithm that runs in time polynomial in the input size: namely, there can be more than $\binom{n-1}{k+1}$ relevant input values for each of the n nodes; so, the runtime of our algorithm is less than cubic in the size of the input, supposing the local scores are given explicitly. While this answers one question in the affirmative, it also raises several further questions, some of which we give in the next paragraphs.

Our complexity analysis relied on a result concerning the general weighted matroid intersection problem. Do significantly faster algorithms exist when restricted to our two specific matroids? One might expect such algorithms exist, since the related problem for branchings can be solved in $O(n^2)$ time by the algorithm of (Tarjan 1977).

Even if we could solve the matroid intersection problem faster, our algorithm would remain practical only for very small values of k . Can one find an optimal k -branching significantly faster, especially if allowing every node to have at most two parents? As the current algorithm makes around n^{3k} mutually overlapping guesses, there might be a way to considerably reduce the time complexity. Specifically, we ask whether the restricted problem is fixed-parameter tractable with respect to the parameter k , that is, solvable in $O(f(k)p(n))$ time for some computable function f and polynomial p (Downey and Fellows 1999). The fixed-parameter algorithm given in Section 4 can be seen as a first step towards an answer to this question. Can we find other restrictions under which the k -branching problem becomes fixed-parameter tractable?

Can we use a similar approach for the more general k -node branching problem, i.e., is there a polynomial time algorithm for the k -node branching problem for every fixed k ? Likewise, we do not know whether the problem is easier or harder for polytrees than for general DAGs: Do similar techniques apply to finding maximum-score DAGs that can be turned into branchings by deleting some k arcs?

Acknowledgments Serge Gaspers, Sebastian Ordyniak, and Stefan Szeider acknowledge support from the European Research Council (COMPLEX REASON, 239962). Serge Gaspers acknowledges support from the Australian Research Council (DE120101761). Mikko Koivisto acknowledges the support from the Academy of Finland (Grant 125637). Mathieu Liedloff acknowledges the support from the French Agence Nationale de la Recherche (ANR AGAPE ANR-09-BLAN-0159-03).

References

- Bock, F. C. 1971. An algorithm to construct a minimum directed spanning tree in a directed network. In Avi-Itzak, B., ed., *Developments in Operations Research*. Gordon and Breach. 29–44.
- Brezovec, C.; Cornuéjols, G.; and Glover, F. 1986. Two algorithms for weighted matroid intersection. *Mathematical Programming* 36(1):39–53.
- Camerini, P. M.; Fratta, L.; and Maffioli, F. 1979. A note on finding optimum branchings. *Networks* 9:309–312.
- Cayley, A. 1889. A theorem on trees. *Quart. J. Math.* 23:376–378.
- Chickering, D. M. 1995. A transformational characterization of equivalent Bayesian network structures. In *Uncertainty in artificial intelligence (UAI 1995)*.
- Chickering, D. M. 1996. Learning Bayesian networks is NP-complete. In *Learning from data (Fort Lauderdale, FL, 1995)*, volume 112 of *Lecture Notes in Statist.* Springer Verlag. 121–130.
- Chu, Y. J., and Liu, T. H. 1965. On the shortest arborescence of a directed graph. *Science Sinica* 14:1396–1400.
- Dasgupta, S. 1999. Learning polytrees. In *Uncertainty in Artificial Intelligence (UAI 1999)*.
- Downey, R. G., and Fellows, M. R. 1999. *Parameterized Complexity*. Monographs in Computer Science. New York: Springer Verlag.
- Edmonds, J. R. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards* 71B(4):233–240.
- Edmonds, J. R. 1970. Submodular functions, matroids and certain polyhedra. In *Combinatorial Structures and their Applications*, 69–87.
- Edmonds, J. R. 1979. Matroid intersection. *Annals of Discrete Mathematics* 4:39–49.
- Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Berlin: Springer Verlag.
- Fomin, F. V., and Kratsch, D. 2010. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer.
- Frank, A. 1981. A weighted matroid intersection algorithm. *Journal of Algorithms* 2:328–336.
- Friedman, N.; Geiger, D.; and Goldszmidt, M. 1997. Bayesian network classifiers. *Machine Learning* 29:131–163.
- Fulkerson, D. R. 1974. Packing rooted directed cuts in a weighted directed graph. *Mathematical Programming* 6:1–13.
- Gabow, H. N.; Galil, Z.; Spencer, T.; and Tarjan, R. E. 1986. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica* 6(2):109–122.
- Gabow, H. N.; Galil, Z.; and Spencer, T. H. 1989. Efficient implementation of graph algorithms using contraction. *Journal of the ACM* 36(3):540–572.
- Garey, M. R., and Johnson, D. R. 1979. *Computers and Intractability*. San Francisco: W. H. Freeman and Company, New York.
- Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning* 20(3):197–243.
- Impagliazzo, R.; Paturi, R.; and Zane, F. 2001. Which problems have strongly exponential complexity? *J. of Computer and System Sciences* 63(4):512–530.
- Iri, M., and Tomizawa, N. 1976. An algorithm for finding an optimal 'independent' assignment. *Journal of the Operations Research Society of Japan* 19:32–57.
- Karp, R. M. 1971. A simple derivation of Edmonds' algorithm for optimum branchings. *Networks* 1(3):265–272.
- Koivisto, M., and Sood, K. 2004. Exact Bayesian structure discovery in Bayesian networks. *J. Mach. Learn. Res.* 5:549–573.
- Lam, W., and Bacchus, F. 1994. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence* 10:269–293.
- Lawler, E. L. 1976. *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart and Winston.
- Ordyniak, S., and Szeider, S. 2010. Algorithms and complexity results for exact Bayesian structure learning. In *Uncertainty in Artificial Intelligence (UAI 2010)*.
- Ott, S., and Miyano, S. 2003. Finding optimal gene networks using biological constraints. *Genome Informatics* 14:124–133.
- Parviainen, P., and Koivisto, M. 2009. Exact structure discovery in Bayesian networks with less space. In *Uncertainty in Artificial Intelligence (UAI 2009)*, 436–443.
- Pietrzak, K. 2003. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. of Computer and System Sciences* 67(4):757–771.
- Silander, T., and Myllymäki, P. 2006. A simple approach for finding the globally optimal Bayesian network structure. In *Uncertainty in Artificial Intelligence (UAI 2006)*, 445–452.
- Tarjan, R. E. 1977. Finding optimum branchings. *Networks* 7:25–35.