Theoretical Computer Science 592 (2015) 49–58

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On finding optimal polytrees[☆]

Serge Gaspers[a], Mikko Koivisto[b], Mathieu Liedloff[c], Sebastian Ordyniak[d,*], Stefan Szeider[e]

[a] The University of New South Wales and NICTA, Australia
[b] University of Helsinki, Finland
[c] Université d'Orléans, France
[d] Masaryk University, Botanická 68a, 60200 Brno, Czech Republic
[e] Institute of Information Systems, Vienna University of Technology, Austria

ARTICLE INFO

Article history:
Received 20 August 2014
Received in revised form 20 March 2015
Accepted 9 May 2015
Available online 14 May 2015
Communicated by V.Th. Paschos

Keywords:
Directed acyclic graphs
Branchings
Polytrees
Parameterized complexity
Matroids
Probabilistic networks

ABSTRACT

We study the NP-hard problem of finding a directed acyclic graph (DAG) on a given set of nodes so as to maximize a given scoring function. The problem models the task of inferring a probabilistic network from data, which has been studied extensively in the fields of artificial intelligence and machine learning. Several variants of the problem, where the output DAG is constrained in several ways, are NP-hard as well, for example when the DAG is required to have bounded in-degree, or when it is required to be a polytree. 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.

In this paper, we generalize this polynomial-time result to polytrees that can be turned into a branching by deleting a constant number of arcs. Our algorithm stems from a matroid intersection formulation. As the order of the polynomial time bound depends on the number of deleted arcs, the algorithm does not establish fixed-parameter tractability when parameterized by that number. We show that certain additional constraints on the sought polytree render the problem fixed-parameter tractable. 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. In this setting, we are given a set of nodes and a scoring function that assigns a real number to every directed graph (digraph) on the node set, and the goal is to find (or construct) a directed *acyclic* graph (DAG) that maximizes the score. As the problem is NP-hard in general [6], researchers have sought for natural problem variants that would be computationally tractable. One approach to this end is to tighten the combinatorial constraints for the sought DAG. This approach has two main advantages: (1) the found more constrained DAG provides a lower bound on the optimum score over less constrained DAGs, which can be used to guide the search in the less constrained space; and (2) a more constrained

[☆] This paper appeared in a preliminary and shortened form in the Proceedings of AAAI 2012, the 26th Conference on Artificial Intelligence [21].
[*] Corresponding author.
E-mail addresses: sergeg@cse.unsw.edu.au (S. Gaspers), mikko.koivisto@cs.helsinki.fi (M. Koivisto), mathieu.liedloff@univ-orleans.fr (M. Liedloff), sordyniak@gmail.com (S. Ordyniak), stefan@szeider.net (S. Szeider).

http://dx.doi.org/10.1016/j.tcs.2015.05.012
0304-3975/© 2015 Elsevier B.V. All rights reserved.

DAG often simplifies the reasoning tasks that will be performed on the learned probabilistic network. It is important to note that the constraints in question restrict the output of the problem, not the input. Thus, tighter constraints do not necessarily amount to easier problem variants. This makes a systematic search for positive and negative results more challenging than in the case where the restrictions concern the input.

The first positive result in this direction was given by Edmonds [10], who gave an efficient algorithm for the class of *branchings*. A branching is a directed forest with in-degree at most one, equivalently, a DAG with in-degree at most one. The algorithm was discovered independently by Chu and Liu [7], and it has been later simplified and expedited by others [1,3,17–19,25,34]. On the other hand, Chickering [6] showed—by a reduction from the Feedback Arc Set problem—that the problem becomes NP-hard as soon as the in-degree bound on the DAG is raised to two (or any larger constant). Motivated by this gap, Dasgupta [8] asked for a class of DAGs that is more general than branchings yet admitting provably good structure-learning algorithms. In particular, he studied *polytrees*, that is, digraphs whose underlying undirected graph is a tree. The results were, however, rather negative, showing that the optimization problem is NP-hard when the in-degree bound is two (or any larger constant).

Given the recent advances in exact exponential algorithms in general (see, e.g., the dedicated book [14]), and in finding optimal DAGs in particular, it is natural to ask, whether "fast" exponential-time algorithms exist for finding optimal polytrees. For unconstrained DAGs the fastest known algorithms run in time within a polynomial factor of 2^n, where n is the number of nodes [26,30,31,33]. Currently, we do not know whether 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 labeled trees $n^{n-2}2^{n-1}$ [4]. Do significantly faster algorithms exist? What if we restrict our search for DAGs that are "almost branchings": Does the problem become easier if only a small number of nodes are allowed an in-degree larger than one?

Results

This article takes a 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. In the problem formulation, we make the assumption—which is standard in the context of learning probabilistic networks—that the score of a DAG is obtained as the sum of local scores, each of which is a single real number assigned to a node and its in-neighbors; we give precise definitions in the next section.

Our main result is an algorithm that finds an optimal k-branching in polynomial time for every constant k. 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 a 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.

While our algorithm for finding an optimal k-branching is polynomial for fixed k, the degree of the polynomial depends on k. Thus the algorithm does not scale well with respect to k. We therefore investigate variants of the problem that admit *fixed-parameter tractability* in the sense of Downey and Fellows [9]: the running time is given by $O(f(k)p(n))$, where p is a polynomial depending only on the input size n and f is a computable function that depends only on the parameter k. In particular, we show that finding an optimal k-branching is NP-hard but fixed-parameter tractable under the following additional constraint on the sought k-branching: the set of arcs incident to nodes with more than one parent are required to form a connected polytree with exactly one sink; in addition, we assume that each node has a bounded number of possible incoming arcs. We complement the fixed-parameter tractability result by showing that finding an optimal polytree is not fixed-parameter tractable with respect to another natural parameter, subject to complexity theoretic assumptions. Namely, we show that the problem is not fixed-parameter tractable when parameterized by the *number of nodes* whose deletion produces a branching. We want to point out that deleting nodes instead of arcs enlarges the class of potential solution networks and therefore potentially produces higher quality networks. This is what makes this parameterization interesting in its own right.

2. The *k*-Branching problem

A (directed) probabilistic network, also called a Bayesian network or directed graphical model, is a multivariate probability distribution that obeys a structural representation in terms of a directed acyclic graph (DAG) and a collecion of univariate conditional probability distributions, one for each node of the graph. In the structure learning problem we consider, the DAG will be the protagonist, whereas the conditional distributions will only play an implicit role in defining the scoring function [27,5,22,8].

We use the following terminology and notation. A DAG is a pair (N, A), where N is the *node set* and $A \subseteq N \times N$ is the *arc set*. We shall identify the graph with the arc set A when there is no ambiguity about the node set. We call a node u a *parent* of v in the graph if (u, v) is an arc in A, and denote by A_v the set of parents of v, that is, the in-neighborhood of v. When our interest is in the undirected structure of the graph, we may denote by $\overline{A}$ the *skeleton* of A, that is, the set of *edges* $\{\{u, v\} : (u, v) \in A\}$. Specifically, we call A a *polytree* if $\overline{A}$ is acyclic, and a *branching* if additionally each node has at most one parent.