

Solving Large Graph Problems Using Tree Decompositions: A Computational Study

Scalable Graph Decompositions & Algorithms to Support the Analysis of Petascale Data

Chris Groër, Blair Sullivan, Dinesh Weerapurage
Oak Ridge National Lab
Computer Science and Mathematics Division
1 Bethel Valley, Oak Ridge TN 37831

Abstract

Tree decompositions provide a way to map an undirected graph onto a tree where each node in this tree represents a subset of vertices from the original graph. The two endpoints of every edge of the graph must live in at least one subset, and all the tree nodes that contain a particular vertex from the graph must form a connected subtree. The largest vertex subset in this tree is known as the width of the decomposition. Dynamic programming algorithms using tree decompositions often provide a way to transform the complexity of a computation into a function that is polynomial in the number of nodes and edges of the graph, but exponential in the decomposition's width. One of the most well-known algorithms of this type allows one to solve an instance of maximum weighted independent set in time $O(2^w n)$ where n is the number of nodes in the original graph and w is the width.

The majority of earlier work in this area is theoretical in nature and sheds little light on the true time complexity or memory requirements of this type of dynamic programming computation. In this talk, we describe some details of a careful implementation of an exact dynamic programming algorithm for maximum weighted independent set. We are able to produce optimal solutions for problem instances where the underlying graph has width much larger than previously thought possible. Moreover, we demonstrate that, along with width, the density of the graph has a great impact on the running time and memory usage of the dynamic programming algorithm.

Finally, we compare the performance and scaling of our implementation with well-established methods such as mixed integer and semi-definite programming. While our dynamic programming algorithm is unable to handle some rather small graphs with high width that are easily solved by the other methods, we demonstrate that, for other types of problems, a dynamic programming algorithm that exploits low width may be the only way to currently generate a provably optimal solution.