Hypertree-decomposition via Branch-decomposition[*]

Marko Samer
Institute of Information Systems (DBAI), Vienna University of Technology, Austria
samer@dbai.tuwien.ac.at

Abstract

Hypertree-decomposition is the most general approach in the literature for identifying tractable computation problems encoded as hypergraphs. We show how the heuristic branch-decomposition approach for ordinary graphs of [Cook and Seymour, 2003] can be used for the heuristic construction of hypertree-decompositions.

1 Introduction

Many NP-complete problems (e.g., the constraint satisfaction problem CSP, the homomorphism problem HOM, the Boolean conjunctive query problem BCQ, etc.) can be described by *hypergraphs* in a natural way. It was shown by [Gottlob *et al.*, 2002] that in analogy to tree-width of graphs, hypertree-width of hypergraphs is an appropriate measure for the cyclicity and therefore the tractability of the corresponding computation problems. In particular, they have shown that NP-complete problems become polynomially solvable and even highly parallelizable when restricted to instances with bounded hypertree-width. Moreover, it was shown by [Gottlob *et al.*, 2000] that hypertree-decomposition and the corresponding measure of hypertree-width generalizes all other tractability measures for hypergraphs in the literature.

Recent research focuses on heuristic approaches for fast hypertree-decomposition with small width. In this presentation, we show how the heuristic approach of [Cook and Seymour, 2003] for branch-decomposition (of ordinary graphs) can be used for heuristic hypertree-decomposition.

2 Preliminaries

A *hypergraph* $\mathfrak{H}$ is a tuple $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices (variables) and $\mathcal{E} \subseteq \wp(\mathcal{V})\setminus\{\emptyset\}$ is a set of hyperedges. We define $var(\mathfrak{H}) = \mathcal{V}$ and $edges(\mathfrak{H}) = \mathcal{E}$. A *hypertree for a hypergraph* $\mathfrak{H}$ is a triple $(\mathfrak{T}, \chi, \lambda)$, where $\mathfrak{T} = (\mathcal{V}, \mathcal{E})$ is a tree and $\chi : \mathcal{V} \to var(\mathfrak{H})$ and $\lambda : \mathcal{V} \to edges(\mathfrak{H})$ are labeling functions.

A *hypertree-decomposition* of a hypergraph $\mathfrak{H}$ is a hypertree $(\mathfrak{T}, \chi, \lambda)$ for $\mathfrak{H}$ satisfying four conditions. Due to space restrictions, we refer the interested reader to [Gottlob *et al.*, 2002] for the formal definition of a hypertree-decomposition. The *width of a hypertree-decomposition* $(\mathfrak{T}, \chi, \lambda)$ is given by $\max_{p \in vertices(\mathfrak{T})} |\lambda(p)|$, and the *hypertree-width of a hypergraph* is the minimum width over all its hypertree-decompositions.

A *branch-decomposition of a hypergraph* $\mathfrak{H}$ is a tuple $(\mathfrak{T}, \tau)$, where $\mathfrak{T} = (\mathcal{V}, \mathcal{E})$ is a tree having $|edges(\mathfrak{H})|$ leaves and in which every non-leaf vertex has degree three, and τ is a bijection from the set of leaves of $\mathfrak{T}$ to $edges(\mathfrak{H})$. The *order* of an edge $e \in \mathcal{E}$ is the number of vertices $v \in var(\mathfrak{H})$ such that there are leaves l_1 and l_2 of $\mathfrak{T}$ in different components of $\mathfrak{T}$ when removing e with $v \in \tau(l_1) \cap \tau(l_2)$. The *width of a branch-decomposition* $(\mathfrak{T}, \tau)$ is the maximum order of the edges of $\mathfrak{T}$, and the *branch-width of a hypergraph* is the minimum width over all its branch-decompositions.

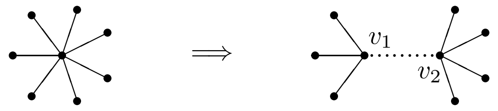

Figure 1: Splitting a vertex

3 The Branch-decomposition Heuristic

In this section, we summarize the basic steps of the heuristic branch-decomposition approach of [Cook and Seymour, 2003]. Note that this approach was developed for ordinary graphs. Its starting point is the construction of the initial data structure which is a star as shown in Fig. 1. This is done in such a way that for each edge in the graph there exists exactly one edge in the star. This edge is then labeled with the set of vertices of the corresponding edge in the graph (i.e., each edge is labeled with exactly two vertices). Afterwards, the vertices of the star are successively split (cf. Fig. 1) according to the decomposition rules below. In each step, the two vertices v_1 and v_2 resulting from the splitting are connected by a new edge which is labeled by the intersection of the set of vertices labeling the edges incident with v_1 and the set of vertices labeling the edges incident with v_2. This process stops when all non-leaf vertices have degree three; the resulting tree is then a *branch-decomposition* of the underlying graph.

The decomposition rules can be divided into two classes: (i) the application of "*safe splits*" and (ii) the application of the *Eigenvector heuristic* [Alon, 1986]. *Safe splits* divide the

[*]This work was supported by the Austrian Science Fund (FWF) project P17222-N04.

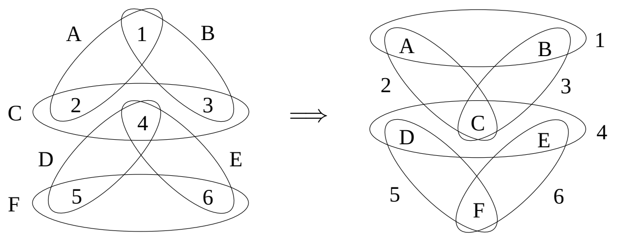

Figure 2: A hypergraph and its dual hypergraph

graph in such a way that it remains *extendible*, i.e., there is some way to (repeatedly) divide the subgraphs to obtain a branch-decomposition of width equal to the branch-width of the original graph. In other words, nothing bad can happen when applying safe splits. There are three kinds of safe splits presented by [Cook and Seymour, 2003]: (i) *pushing*, (ii) *2-separations*, and (iii) *3-separations*. At each stage, these safe splits are applied as long as possible. If none of them is applicable, the Eigenvector heuristic is used.

4 Application to Hypertree-decomposition

We will now show how the branch-decomposition heuristic described in Section 3 can be used for hypertree-decomposition. Recall that this heuristic was developed for ordinary graphs and not for hypergraphs. Thus, our first step is to show that the safe splits as well as the Eigenvector heuristic are also applicable to hypergraphs.

To this aim, recall that the new edge introduced at each vertex splitting (cf. Fig. 1) is labeled with the intersection of two sets of vertices as described above. In general, however, this intersection may contain more than two vertices. Thus, such newly introduced edges represent hyperedges, i.e., the data structure underlying the original branch-decomposition approach starts as an ordinary graph and evolves to a hypergraph during the decomposition process. This can be seen as an intuitive justification for the applicability of the branch-decomposition heuristic to hypergraphs. It is easy to verify that this intuition holds indeed true.

Now, note that given a tree-decomposition[1] of a hypergraph of width k, it is possible to construct a branch-decomposition of width at most $k + 1$, and given a branch-decomposition of a hypergraph of width k, it is possible to construct a tree-decomposition of width at most $3k/2$ [Robertson and Seymour, 1991]. Thus, having small tree-width is equivalent to having small branch-width.

Hence, a simple approach to obtain a hypertree-decomposition via branch-decomposition is to (i) construct a branch-decomposition of the given hypergraph by using the above heuristic, (ii) transform this branch-decomposition into a tree-decomposition, and, in analogy to [McMahan, 2004], (iii) apply set covering heuristics in order to obtain a hypertree-decomposition. In particular, set covering is used to obtain appropriate λ-labels based on the χ-labels given by the tree-decomposition.

[1]Intuitively, a tree-decomposition is a hypertree-decomposition without the labeling function λ.

Another approach we investigate is dual to the above one in the sense that we obtain a hypertree where the λ-labels are given and appropriate χ-labels have to be set. To this aim, let us first introduce the *dual hypergraph* of a hypergraph as exemplified in Fig. 2. The dual hypergraph is simply obtained by swapping the roles of hyperedges and vertices. Our procedure is then to (i) construct a branch-decomposition of the *dual* hypergraph by using the above heuristic, (ii) transform this branch-decomposition into a tree-decomposition, and (iii) interpret the labels of this tree-decomposition as λ-labels of a hypertree and set the χ-labels appropriately in a straight-forward way. The resulting hypertree is then a hypertree-decomposition of the original hypergraph.

The attentive reader may have noticed that there are two problems with the latter approach: First, the branch-width and therefore also the hypertree-width is at least the cardinality of the largest edge in the dual hypergraph which is equal to the maximum number of hyperedges having a common vertex in the original hypergraph. Second, since the λ-labels satisfy the conditions of a tree-decomposition by construction, the hypertree-width may be larger than necessary. However, by introducing heuristic pre- and post-processing steps, we are able to overcome both problems.

5 Conclusion

We have tested our approach on hypergraphs representing adder and bridge circuits connected in a line. For all instances of such adder and bridge hypergraphs, we obtained a hypertree-decomposition of almost optimal width within a few seconds. Natural future work will be a full implementation of our approach, the investigation of further improvements, and a systematic comparison with other hypertree-decomposition heuristics. Moreover, an implementation of our second approach in such a way that the tree-decomposition of the dual hypergraph is directly constructed (e.g., by *bucket elimination*) without the intermediate-step of a branch-decomposition seems to be promising.

References

[Alon, 1986] N. Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

[Cook and Seymour, 2003] W. Cook and P. Seymour. Tour merging via branch-decomposition. *INFORMS Journal on Computing*, 15(3):233–248, 2003.

[Gottlob *et al.*, 2000] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, 124(2):243–282, 2000.

[Gottlob *et al.*, 2002] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, 2002.

[McMahan, 2004] B. McMahan. Bucket eliminiation and hypertree decompositions. Implementation report, Institute of Information Systems (DBAI), TU Vienna, 2004.

[Robertson and Seymour, 1991] N. Robertson and P. Seymour. Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52:153–190, 1991.