ECAI 2008
M. Ghallab et al. (Eds.)
IOS Press, 2008

doi:10.3233/978-1-58603-891-5-913

On the Practical Significance of Hypertree vs. Tree Width

Rina Dechter[1] **Lars Otten**[1] and **Radu Marinescu**[2]

Abstract. The recently introduced notion of hypertree width has been shown to provide a broader characterization of tractable constraint and probabilistic networks than the tree width. This paper demonstrates empirically that *in practice* the bounding power of the tree width is still superior to the hypertree width for many benchmark instances of both probabilistic and deterministic networks.

1 INTRODUCTION

Inference in graphical models is known to be time and space exponential in the problem graph's tree width. In practice, however, this measure is often inaccurate, since it ignores the effects of determinism in problem solving, which can, for instance, lead to pruning of large parts of the search space. To that end, in 2000 [3] introduced a parameter called hypertree width and showed that for constraint networks it is more effective in capturing tractable classes. In [4], its applicability was extended to inference algorithms over general graphical models having relational function specification.

In this paper we explore the practical significance of the hypertree width against the tree width from a more practical angle. We show empirically, on probabilistic and deterministic benchmarks, that in most cases the tree width yields a far better predictor of instance-based complexity than the hypertree width, except when the problem has substantial determinism.

The outline of this paper is as follows: Section 2 gives a brief overview of the two decomposition schemes. Section 3 provides the empirical results, and Section 4 concludes.

2 DECOMPOSITION SCHEMES

We assume the usual definitions of directed and undirected graphs, hypergraphs, primal and dual graphs, and hypertrees. A graphical model is typically defined to be a set of real-valued functions $F = \{f_1, \ldots, f_l\}$ over a set of variables $X = \{x_1, \ldots, x_n\}$ with domains $D = \{D_1, \ldots, D_n\}$, together with a combination operator like summation or multiplication. The scope of a function f_j, denoted $scope(f_j)$, is the set of variables on which f_j is defined.

A common approach to solving graphical model problems is to cluster variables and functions such that the resulting decomposition exhibits tree structure:

DEFINITION **1** *A **tree decomposition** of a graphical model is a triple $\mathcal{T} = \langle T, \chi, \psi \rangle$, where $T = (V, E)$ is a tree and χ and ψ are labeling functions that associate with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq F$, that satisfy the following conditions:*

1. *For each $f_j \in F$, there is at least one $v \in V$ such that $f_j \in \psi(v)$.*
2. *If $f_j \in \psi(v)$, then $scope(f_j) \subseteq \chi(v)$.*
3. *For each $x_i \in X$, the set $\{v \in V | x_i \in \chi(v)\}$ induces a connected subtree of T.*

*The **tree width** of $\mathcal{T}$ is $w = max_{v \in V} |\chi(v)| - 1$. $\mathcal{T}$ is also a **hypertree decomposition** if it satisfies the following additional condition:*

4. *For each $v \in V$, $\chi(v) \subseteq \bigcup_{f_j \in \psi(v)} scope(f_j)$.*

*In this case, the **hypertree width** of $\mathcal{T}$ is $hw = max_{v \in V} |\psi(v)|$.*

Finding tree and hypertree decompositions of minimal width is known to be NP-complete, therefore heuristic algorithms are employed in practice [4, 2]. Once a tree or hypertree decomposition is available, it can be processed by the suitable version of a message passing algorithm like *Cluster-Tree Elimination (CTE)* [4]. Allowing a probabilistic function to be placed in more than one node will lead to incorrect processing by CTE for any graphical model other than constraint networks. To remedy this we modify multiple showings of a function by *flattening* all but one of them into a 0/1-valued constraint.

Complexity bounds

The time complexity of algorithm CTE, when executed on a tree decomposition $\mathcal{T}$ with tree width w, has been shown to be

$$O((r + m) \cdot deg \cdot k^{w+1}) \ , \tag{1}$$

where r is the number of functions in the problem, m the number of clusters in $\mathcal{T}$, and deg the maximum degree in $\mathcal{T}$. The space complexity is $\mathcal{O}(m \cdot k^w)$ [4].

By virtue of using a tree decomposition, however, the hypergraph structure of the problem is completely ignored. Bound (1) does therefore not account for any determinism that might be present in the function specifications. To that end, if $\mathcal{T}$ is also a hypertree decomposition, algorithm CTE can be adapted to exploit the hypergraph structure. Assuming $\mathcal{T}$ has hypertree width hw, the time complexity of applying CTE can be shown to be

$$O(m \cdot deg \cdot hw \cdot \log t \cdot t^{hw}) \ , \tag{2}$$

where t bounds the size of the relational representation of each function in the problem (i.e., the number of zero cost tuples in CSPs and the number of non-zero probability tuples in belief networks). Space complexity is $\mathcal{O}(t^{hw})$ [3, 4].

We note that bound (2) indeed takes determinism into account by using the parameter t, which denotes the number of relevant tuples in a function table. It is clear that $t \leq k^r \leq k^w$, where r is the maximum function arity. Hence bound (2) can only yield tighter results when the problem instance possesses a high degree of determinism. While it has been shown that hypertree decompositions are strictly more general than tree decompositions, it is unclear how the asymptotic bounds compare for practical problem instances.

[1] Bren School of Information and Computer Sciences, University of California, Irvine, CA 92697-3435. Email: {dechter,lotten}@ics.uci.edu
[2] Cork Constraint Computation Centre, Department of Computer Science, University College Cork, Ireland. Email: r.marinescu@4c.ucc.ie

instance	n	k	r	t	w	hw	R
Genetic linkage							
pedigree1	334	4	5	32	16	13	9.934
pedigree18	1184	5	5	50	22	18	15.204
pedigree20	437	5	4	50	24	16	10.408
pedigree23	402	5	4	50	29	15	5.214
pedigree25	1289	5	5	50	27	19	13.408
pedigree30	1289	5	5	50	25	18	13.107
pedigree33	798	4	5	32	31	21	12.944
pedigree37	1032	5	4	32	22	13	4.190
pedigree38	724	5	4	50	18	10	4.408
pedigree39	1272	5	4	50	25	18	13.107
pedigree42	448	5	4	50	24	16	10.408
pedigree50	514	6	4	72	18	10	4.567
pedigree7	1068	4	4	32	40	23	10.536
pedigree9	1118	7	4	50	31	21	9.480
pedigree13	1077	3	4	18	35	29	19.704
pedigree19	793	5	5	50	27	21	16.806
pedigree31	1183	5	5	50	34	29	25.505
pedigree34	1160	5	4	32	32	25	15.262
pedigree40	1030	7	5	98	31	24	21.591
pedigree41	1062	5	5	50	35	25	18.010
pedigree44	811	4	5	32	28	22	16.256
pedigree51	1152	5	4	50	44	33	25.311
Mastermind puzzle game (WCSP)							
mm_03_08_03	1220	2	3	4	21	14	2.107
mm_03_08_04	2288	2	3	4	31	20	2.709
mm_03_08_05	3692	2	3	4	40	25	3.010
mm_04_08_03	1418	2	3	4	26	17	2.408
mm_04_08_04	2616	2	3	4	38	24	3.010
mm_10_08_03	2606	2	3	4	56	34	3.612

instance	n	k	r	t	w	hw	R
Coding networks							
BN_126	512	2	5	16	56	21	8.429
BN_127	512	2	5	16	55	22	9.934
BN_128	512	2	5	16	50	20	9.031
BN_129	512	2	5	16	54	21	9.031
BN_130	512	2	5	16	53	21	9.332
BN_131	512	2	5	16	53	21	9.332
BN_132	512	2	5	16	52	21	9.633
BN_133	512	2	5	16	56	21	8.429
BN_134	512	2	5	16	55	21	8.730
Dynamic Bayesian networks							
BN_21	2843	91	4	208	7	4	-4.441
BN_23	2425	91	4	208	5	3	-2.841
BN_25	1819	91	4	208	5	2	-5.159
BN_27	3025	5	7	3645	10	2	0.134
BN_29	24	10	6	999999	6	2	6.000
Digital circuits							
c432.isc	432	2	10	512	28	22	51.175
c499.isc	499	2	6	32	25	25	30.103
s386.scan	172	2	5	16	19	8	3.913
s953.scan	440	2	5	16	66	38	25.889
Radio frequency assignment (WCSP)							
CELAR6-SUB0	16	44	2	1302	8	4	-0.689
CELAR6-SUB1-24	14	24	2	301	10	5	-1.409
CELAR6-SUB1	14	44	2	928	10	5	-1.597
CELAR6-SUB2	16	44	2	928	11	6	-0.273
CELAR6-SUB3	18	44	2	928	11	6	-0.273
CELAR6-SUB4-20	22	20	2	396	12	6	-0.026
CELAR6-SUB4	22	44	2	1548	12	6	-0.583

Table 1. Selected experimental results comparing the tree width and hypertree width based bounds.

3 EXPERIMENTAL RESULTS

We evaluated empirically the tree width and hypertree width bounds on 112 practical probabilistic networks and 30 constraint networks. Problem instances were obtained from various sources; all of them are available online[3].

To obtain a tree decomposition of a problem, we perform bucket elimination along a minfill ordering (random tie breaking, optimum over 20 iterations). The tree decomposition is then extended to a hypertree decomposition by the method described in [2], where variables in a decomposition cluster are greedily covered by functions.

For each problem instance we collected the following statistics: the number of variables n, the maximum domain size k, the maximum function arity r, and the maximum function tightness t. We also report the best tree width and hypertree width found in the experiments described above.

We define the measure $R := \log_{10}(\frac{t^{hw}}{k^w})$. This compares the two dominant factors of the w bound (1) and the hw bound (2). If R is positive, it signifies how many orders of magnitude tighter the w bound is when compared to the hw bound, and vice versa for negative values of R. Some select instances are shown in Table 1, the full set of results is available in an extended version of this paper [1].

Out of the 112 belief networks, the hw bound was only superior for 5 instances, and not by many orders of magnitude. On the other hand, for genetic linkage instances with considerable determinism in their CPTs, the hw bound is significantly worse, as is the case for most other belief networks. This situation does not change much for constraint problems, except for radio frequency assignment, where the hw bound fares somewhat better, but only by a small margin.

In summary we can review that, in order for the hypertree width bound to be competitive with, or even superior to, the tree width bound, problem instances need to comply with several conditions; foremost among these are very tight function specifications. The latter is promoted by large variable domains and high function arity, which we found to be not the case for the majority of practical problem instances.

[3] Repository at http://graphmod.ics.uci.edu/

4 CONCLUSION

The contribution of this paper is in exploring empirically the practical benefit of the hypertree width compared with the tree width in bounding the complexity of algorithms over given problem instances. Statistics collected over 112 Bayesian networks instances and 30 weighted CSPs provided interesting, yet somewhat sobering information. We confirmed that while the hypertree is always smaller than the tree width, the complexity bound it implies is often inferior to the bound suggested by the tree width. Only when problem instances possess substantial determinism and when the functions have large arity, the hypertree can provide bounds that are tighter and therefore more informative than the tree width.

The above empirical observation raises doubts regarding the need to obtain good hypertree decompositions beyond the already substantial effort into the search of good tree-decompositions, that has been ongoing for three decades now.

ACKNOWLEDGEMENTS

This work was partially supported by NSF grant IIS-0713118 and NIH grant R01-HG004175-02.

REFERENCES

[1] R. Dechter, L. Otten, and R. Marinescu: 'On the Practical Significance of Hypertree vs. Tree Width', *Technical Report, University of California, Irvine*, (2008).

[2] G. Gottlob, M. Grohe, N. Musliu, M. Samer, and F. Scarcello, Hypertree Decompositions: 'Structure, Algorithms, and Applications', *International Workshop on Graph-Theoretic Concepts in Computer Science*, (2005).

[3] G. Gottlob, N. Leone, and F. Scarcello, 'A comparison of structural CSP decomposition methods', *Artificial Intelligence*, (2000).

[4] K. Kask, R. Dechter, J. Larrosa, and A. Dechter, 'Unifying tree-decompositions for reasoning in graphical models', *Artificial Intelligence*, (2005).