On the Practical Significance of Hypertree vs Tree Width

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

Abstract. In 2000, [4] presented a new graph parameter, the hypertree width, and showed that it provides a broader characterization of tractable constraint networks than the treewidth. In 2005, [5] extended this result to general probabilistic graphical models, showing that the hypertree width yields bounds on inference algorithms when functions are expressed relationally.

The main contribution of this paper is in demonstrating empirically that *in practice* the bounding power of the treewidth is still superior to the hypertree width for many benchmark instances of both probabilistic and deterministic networks. Specifically, we show that the treewidth yields a far tighter bound on the algorithm's performance when the graphical model has a low level of determinism. A secondary theoretical contribution of the paper is in showing that the hypertree width bound is also relevant to search algorithms and to functions which are specified via decision trees.

1 Introduction

Bayesian networks, constraint networks, Markov random fields and influence diagrams, commonly referred to as graphical models, are all languages for knowledge representation that use graphs to capture conditional independencies between variables. These independencies allow both the concise representation of knowledge and the use of efficient graph-based algorithms for query processing.

Algorithms for processing graphical models fall into two general types: inference-based algorithms and search-based algorithms. Inference-based algorithms (e.g., variable-elimination, join-tree clustering) exploit the independencies captured by the underlying graphical model. They are known to be time and space exponential in the treewidth of the graph. It was recently shown that search algorithms can also be bounded exponentially by the treewidth if they traverse an AND/OR search graph, called the context-minimal graph [2].

However, often graphical models algorithms are far more efficient than what is predicted by the treewidth, especially when the problem instance possesses a significant amount of determinism. Indeed, the treewidth is a measure which is completely blind to the specific representation of the functions and in fact it assumes that a function defined on r variables must take $O(k^r)$ to specify, when k bounds the variables' domain size.

In 2000, [4] introduced another parameter called hypertree width and showed that for constraint networks it is more effective at capturing tractable classes. In [5] the applicability of the hypertree width was extended to *inference* algorithms over general graphical models having relational function specification.

[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

In this paper we explore the practical significance of the hypertree width against the treewidth from a more practical angle. We ask if the hypertree width can yield tighter bounds on algorithms' performance, given a problem instance, than those provided by the treewidth.

We show empirically, on probabilistic and deterministic benchmarks, that the answer is negative: in most cases the treewidth yields a far better predictor of instance-based complexity than the hypertree width, except when the problem has substantial determinism.

Sections 2 provides preliminaries. Section 3 gives an overview of work on tree-decomposition and hypertree-decomposition. Sections 4 and 5 extend the hypertree width bound to search algorithms and decision trees specifications. Section 6 provides the empirical evaluation and Section 7 concludes.

2 Background

We assume the usual definitions of directed and undirected graphs.

DEFINITION **1 (hypergraph, primal and dual graph)** *A* **hypergraph** *is a pair $H = (V, S)$ where $S = \{S_1, ..., S_t\}$ is a set of subsets of V, called hyper-edges. The* **primal graph** *of a hypergraph $H = (V, S)$ is an undirected graph $G = (V, E)$ such that there is an edge $(u, v) \in E$ for any two vertices $u, v \in V$ that appear in the same hyper-edge (namely, there exists S_i, s.t., $u, v \in S_i$). The* **dual graph** *of a hypergraph $H = (V, S)$ is an undirected graph $G = (S, E)$ that has a vertex for each hyper-edge, and there is an edge $(S_i, S_j) \in E$ when the corresponding hyper-edges share a vertex ($S_i \cap S_j \neq \emptyset$).*

DEFINITION **2 (hypertree)** *A hypergraph is a* **hypertree**, *also called* **acyclic hypergraph**, *if and only if its dual graph has an edge subgraph that is a tree, such that all the nodes in the dual graph that contain a common variable form a connected subgraph.*

A graphical model is defined by a set of real-valued functions F over a set of variables X, conveying probabilistic or deterministic information, whose structure can be captured by a graph.

DEFINITION **3 (graphical model)** *A* graphical model $\mathcal{R}$ *is a 4-tuple, $\mathcal{R} = \langle X, D, F, \otimes \rangle$, where: (1) $X = \{X_1, \ldots, X_n\}$ is a set of variables; (2) $D = \{D_1, \ldots, D_n\}$ is the set of their respective finite domains of values; (3) $F = \{f_1, \ldots, f_r\}$ is a set of discrete real-valued functions, each defined over a subset of variables $S_i \subseteq X$, called its scope, and sometimes denoted by $scope(f_i)$; (4) $\otimes_i f_i \in \{\prod_i f_i, \sum_i f_i, \bowtie_i f_i\}$ is a combination operator.*

The graphical model represents the combination of all its functions: $\otimes_{i=1}^{r} f_i$. A reasoning task is based on a marginalization (elimination) operator $\Downarrow$ and is defined by: $\Downarrow_{Z_1} \otimes_{i=1}^{r} f_i, \ldots, \Downarrow_{Z_t} \otimes_{i=1}^{r} f_i$, where $Z_i \subseteq X$.

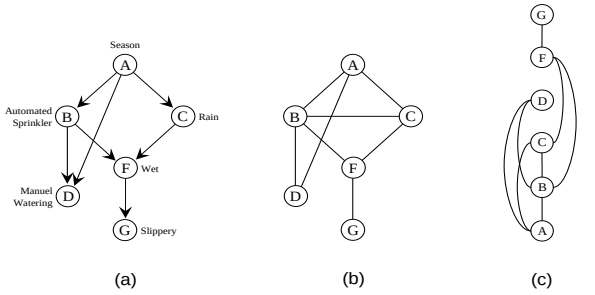

Figure 1. (a) Belief network $P(g, f, d, c, b, a)$, (b) its moral graph and (c) its induced graph.

DEFINITION **4 (acyclic graphical model)** *The set of variables and the scopes of a graphical model (X, S) defines the graphical model's hypergraph. If this hypergraph is a hypertree the graphical model is called* acyclic.

The two special cases of reasoning tasks which we have in mind are constraint networks, belief networks or mixed networks that combine both [2]. The primary tasks for constraint networks are finding a solution and counting solutions. The common function specification is relational, using relational join and project as the combination and marginalization operators, respectively. The primary tasks over belief networks are belief updating and finding the most probable explanation. They are defined using conditional probability functions defined on each variable and its parents in a given directed acyclic graph, and use multiplication and summation or maximization as the combination and marginalization operators [5].

Example 1 Consider a belief network in Figure 1a. It contains variables A, B, C, D, F, G and functions $f(A, B)$, $f(A, C)$, $f(B, C, F)$, $f(A, B, D)$, $f(F, G)$ which are conditional probability tables between a child node and its parents. For example $f(B, C, F) = P(F|B, C)$. Figure 1c gives the induced-graph along the ordering $d = A, B, C, D, F, G$.

3 Tree and Hypertree Decompositions

Tree clustering schemes have been widely used for constraint processing and for probabilistic reasoning.

3.1 Tree Decomposition

The most popular variants are join-tree and junction-tree algorithms. The schemes vary somewhat in their graph definitions as well as in the way tree-decompositions are processed [3, 6, 4, 8]. However, they all involve a decomposition of a hypergraph into a hypertree.

DEFINITION **5 (tree-decomposition)** *[5] Let $P = \langle \mathcal{R} \Downarrow, \{Z_i\} \rangle$ be a reasoning problem over a graphical model $\langle X, D, F, \bigotimes \rangle$. A* **tree-decomposition** *for P is a triple $\langle T, \chi, \psi \rangle$, 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 function $f_i \in F$, there is* **exactly one** *vertex $v \in V$ such that $f_i \in \psi(v)$.*
2. *If $f_i \in \psi(v)$, then $scope(f_i) \subseteq \chi(v)$.*
3. *For each variable $X_i \in X$, the set $\{v \in V | X_i \in \chi(v)\}$ induces a connected subtree of T. This is also called the running intersection or the connectedness property.*

DEFINITION **6 (treewidth, separator)** *The treewidth w of a tree-decomposition $\langle T, \chi, \psi \rangle$ is $\max_{v \in V} |\chi(v)| - 1$. Given two adjacent vertices u and v of a tree-decomposition, a separator of u and v is defined as $sep(u, v) = \chi(u) \cap \chi(v)$.*

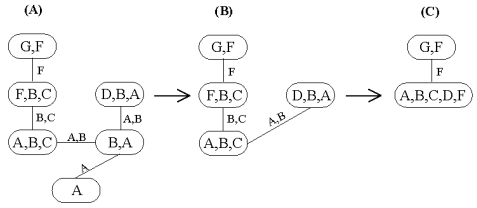

Figure 2. Several tree-decompositions of the same belief network

Example 2 Consider the belief network in Figure 1a whose primal graph is given in Figure 1b. Any of the trees in Figure 2 is a tree-decomposition for this problem, where the labeling χ is the set of variables in each node. The functions can be assigned to nodes whose χ variables contain their scopes. For example, in the tree-decomposition of Figure 2c, any function with scope $\{G\}$ must be placed in vertex 1 because vertex 1 is the only vertex that contains variable G. Any function with scope $\{A, B, C, D\}$ or its subset must be placed in vertex 2, and any function with scope $\{F\}$ can be placed either in vertex 1 or 2.

Once a tree-decomposition is given, it can be processed by a message passing algorithm where each vertex of the tree sends a function to each of its neighbors. If the tree contains m edges, then a total of $2m$ messages will be sent as follows: for each neighbor v, node u takes all the functions in $\psi(u)$ and all the messages received by u from all adjacent nodes and generates their *combined* function which is *marginalized* over the separator between u and v, the result is sent to v. The algorithm can be applied with any style of function specification over which the combination and marginalization operators are well defined. For a discussion of various styles of algorithms such as *join-tree clustering (JTC)* and the more generic version of *Cluster Tree Elimination (CTE)* see [5]. We will use *CTE* as a generic name for a message passing algorithm over a tree-decomposition.

Theorem 1 (treewidth-based complexity of CTE) *[5] Given a reasoning problem $\mathcal{P} = \langle X, D, F, \bigotimes, \Downarrow_Y \rangle$, and a tree-decomposition $\langle T, \chi, \psi \rangle$ let m be the number of vertices in the tree decomposition, w its treewidth, sep its maximum separator size, r the number of input functions in F, deg the maximum degree in T, and k the maximum domain size of a variable. The time complexity of CTE is*

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

and its space complexity is $O(m \cdot k^{sep})$.

3.2 Hypertree Decomposition

One problem with the treewidth is its sole dependence on the primal graph, ignoring its hypergraph structure completely. For example, an

acyclic problem whose scopes have high arity would have a high treewidth, even though it can be processed in linear time. In particular, Bayesian networks which are *polytrees* are acyclic, yet they have treewidth equal to the maximum family size, which can be arbitrarily large [7].

The hypertree width introduced by [4] for constraint networks and extended in [5] for general graphical models, relies on the notion of *hypertree decompositions*. It provides a stronger indicator of tractability than the treewidth. Hypertree decompositions are a subclass of tree-decompositions: on the one hand it has to satisfy the additional restriction that the variables labeling each node in the decomposition node are covered by the combined scopes of all its labeling functions while on the other, a function can be placed in more than a single node and therefore cover variables in several clusters.

DEFINITION **7 (hypertree decomposition)** *(adapted from [4]) Let $\mathcal{T} = \langle T, \chi, \psi \rangle$, where $T = (V, E)$, be a tree-decomposition of $P = \langle X, D, F, \bigotimes, \Downarrow \rangle$. $\mathcal{T}$ is a hypertree decomposition of P if condition 1 of definition 5 is relaxed to allow that a function is placed in more than one node:*

1. *For each function $f_i \in F$, there is* **at least one** *vertex $v \in V$ such that $f_i \in \psi(v)$.*

and if the following additional condition is satisfied: lets $\psi(v)$ be all the functions in the problem whose scopes are included in $\chi(v)$ Then,

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

The hypertree width of a hypertree decomposition is $hw = max_v|\psi(v)|$. The hypertree-width of a hypergraph is the minimum hypertree width over all hypertree decompositions.

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 should modify multiple showings of a function by *flattening* the function into a 0/1-valued constraint. With this modification we can show that the algorithm is guaranteed to work properly.

It was shown in [4] that a hypertree decomposition of a constraint problem can be processed in time exponential in the hypertree width. [5] showed that this complexity bound can be extended straightforwardly to any general graphical model with relational specification that is absorbing relative to 0. A graphical model is absorbing relative to a 0 element if its combination operator has the property that $x \bigotimes 0 = 0$; for example, multiplication has this property while summation does not. In summary,

Theorem 2 *[5] A hypertree decomposition of a reasoning problem that is absorbing relative to 0 can be processed in time*[3]

$$O(m \cdot deg \cdot hw \cdot logt \cdot t^{hw}) \quad (2)$$

and space $O(t^{hw})$, where m is the number of edges in the hypertree decomposition, hw its hypertree width, and t bounds the size of the relational representation of each function in $\mathcal{R}$.

4 Hypertree Bounds for Search and for Decision Tree Specification

AND/OR search spaces were recently introduced for graphical models, showing that problem decomposition can be captured explicitly in the search space using AND nodes. When caching of nodes that root identical search subspace is utilized via context-identical nodes, we get the *context-minimal* AND/OR search space. It was shown that the size of the context-minimal AND/OR search graph is $O(nk^{w+1})$ when w is the tree-width of the tree-decomposition that guides the AND/OR search [2].

We next show that the hypertree width bound is also capable of bounding the AND/OR search graph if functions are specified relationally, and discuss extensions to additional specifications of functions. It has been shown:

Proposition 1 *[2] For any acyclic graphical model $\mathcal{P}$, having r, relationally-specified functions absorbing relative to 0, the AND/OR context-minimal search graph is bounded by $O(r \cdot t)$, when t bounds the size of the relational specification of each function.*

We can now extend the above theorem to any cyclic graphical model using hypertree decompositions and hypertree width as follows. Given a hypertree decomposition T, we can consider the acyclic graphical model that can be generated by *combining* (e.g., multiplying) all the functions in each node, yielding functions of size at most $O(t^{hw})$. As in the acyclic case, it can be argued that the number of different contexts in an AND/OR search graph derived from T will not be larger than the number of tuples in the generated function in each cluster, which is bounded by t^{hw}. In summary:

Theorem 3 *Given a graphical model $\mathcal{R}$, and a hypertree-decomposition having hypertree width hw, then there exists a tree $\mathcal{T}$ such that the context minimal AND/OR search graph based on $\mathcal{T}$ is $O(m \cdot t^{hw})$, when t bounds the function relational specification size and m is the number of vertices in the decomposition.*

Corollary 1 *Thus, if we have a hypertree decomposition and a tree $\mathcal{T}$ that can drive an AND/OR search graph, then algorithms such as AND/OR depth-first or best-first search, that perform context-based caching along $\mathcal{T}$, are bounded exponentially by the hypertree width.*

The question is, what happens when a function is specified more compactly? For example, what if functions are specified via decision trees or decision diagrams? It is easy to show that,

Theorem 4 *For a graphical model whose functions are specified as* decision trees *whose sizes is bounded by t, both inference (e.g., CTE) and search (AND/OR search) algorithms have time and space exponential in the hypertree width, namely $O(m \cdot t^{hw})$.*

5 Experimental Results

Note that if $t = k^w$ then Equation 2 becomes: $m \cdot deg \cdot hw \cdot w \cdot logk \cdot (k^w)^{hw}$. In this case, for any hypertree decomposition, the treewidth-based bound is clearly superior than the one provided by its hypertree width. The question we therefore ask is, under what conditions would the complexity bound generated by the hypertree width ("hw bound") be tighter than the bound generated by the treewidth ("w bound")? And how often are those conditions met in practice? A simple algebraic manipulation of Equations 1 and 2 yields:

Theorem 5 (comparing bounds) *Given a hypertree decomposition of treewidth w and hypertree width hw, the hw bound is tighter than the w bound iff*

$$t < \frac{k^{\frac{w+1}{hw}}}{\sqrt[hw]{w \cdot hw}},$$

[3] The algorithms for processing decompositions assumed in [4] and [5] are slightly different

instance	n	k	r	t	w	hw	R
Grid networks							
90-10-1	100	2	3	8	14	7	2.107
90-14-1	196	2	3	8	20	11	3.913
90-16-1	256	2	3	8	24	12	3.612
90-24-1	576	2	3	8	36	20	7.225
90-24-1e20	576	2	3	8	37	19	6.021
90-26-1e40	676	2	3	8	41	22	7.526
90-30-1e60	900	2	3	8	47	24	7.526
90-34-1e80	1156	2	3	8	56	29	9.332
90-38-1e120	1444	2	3	8	62	33	11.138
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
Grid networks							
BN_31	1156	2	3	8	52	30	11.439
BN_33	1444	2	3	8	61	34	12.342
BN_35	1444	2	3	8	61	34	12.342
BN_37	1444	2	3	8	60	34	12.643
BN_39	1444	2	3	8	61	34	12.342
BN_41	1444	2	3	8	62	34	12.041
Digital circuits							
BN_48	661	2	5	16	46	23	13.847
BN_50	661	2	5	16	46	23	13.847
BN_52	661	2	5	16	46	23	13.847
BN_54	561	2	5	16	53	32	22.577
BN_56	561	2	5	16	53	32	22.577
BN_58	561	2	5	16	53	32	22.577
BN_60	540	2	5	16	55	30	19.567
BN_62	667	2	5	16	46	23	13.847
BN_64	540	2	5	16	55	30	19.567
BN_66	440	2	5	16	69	36	22.577
BN_68	440	2	5	16	68	38	25.287
CPCS medical diagnosis							
BN_79	54	2	10	1024	14	6	13.847
BN_81	360	2	12	2048	21	4	6.924
BN_83	360	2	12	2048	21	4	6.924
BN_85	360	2	12	4096	21	4	8.128
BN_87	422	2	18	131072	23	4	13.546
BN_89	422	2	18	131072	23	4	13.546
BN_91	422	2	18	131072	23	4	13.546
BN_93	422	2	18	261408	23	5	20.163
Randomly generated belief networks							
BN_95	53	3	4	81	20	8	5.725
BN_97	54	3	4	81	20	8	5.725
BN_99	57	3	4	81	23	10	8.111
BN_101	58	3	4	81	22	9	6.680
BN_103	76	3	4	81	28	11	7.634
Randomly generated partial k-trees with forced determinism							
BN_105	50	2	21	20540	25	3	5.412
BN_107	50	2	21	327471	25	3	9.020
BN_109	50	2	20	163866	25	3	8.118
BN_111	50	2	20	81910	25	3	7.214
BN_113	50	2	21	327546	25	3	9.020
Randomly generated partial k-trees without forced determinism							
BN_115	50	2	19	131072	25	3	7.827
BN_117	50	2	20	65536	25	3	6.924
BN_119	50	2	19	32768	25	3	6.021
BN_121	50	2	19	131072	25	3	7.827
BN_123	50	2	20	32768	25	3	6.021
BN_125	50	2	18	131072	25	3	7.827
Digital circuits (WCSP)							
c432	432	2	10	512	28	21	48.466
c499	499	2	6	32	25	25	30.103
c880	880	2	5	16	25	25	22.577
s1196	561	2	5	16	55	30	19.567
s1238	540	2	5	16	55	30	19.567
s1423	748	2	5	16	24	19	15.654
s1488	667	2	5	16	46	23	13.847
s1494	661	2	5	16	46	23	13.847
s386	172	2	5	16	19	8	3.913
s953	440	2	5	16	66	39	27.093
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
CPCS medical diagnosis							
cpcs54	54	2	10	256	14	6	10.235
cpcs179	179	4	9	8192	9	3	6.322
cpcs360b	360	2	12	4096	21	4	8.128
cpcs422b	422	2	18	261408	23	4	14.746
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
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
Various networks							
Barley	48	67	5	40320	8	4	3.813
Diabetes	413	21	3	2040	5	5	9.937
hailfinder	56	11	5	1181	5	3	4.010
insurance	27	5	4	156	8	3	0.988
Mildew	35	100	4	14849	5	3	2.515
Munin1	189	21	4	276	12	6	-1.221
Munin2	1003	21	4	276	8	8	8.950
Munin3	1044	21	4	276	8	6	4.068
Munin4	1041	21	4	276	9	5	0.305
Pigs	441	3	3	15	11	8	4.160
Water	32	4	6	1454	11	4	6.028
Genetic linkage							
fileEA0	381	4	4	16	8	5	1.204
fileEA1	836	5	4	20	12	6	-0.581
fileEA2	979	5	4	20	14	10	3.225
fileEA3	1122	5	4	20	16	10	1.827
fileEA4	1231	5	4	20	16	10	1.827
fileEA5	1515	5	4	20	15	11	3.827
fileEA6	1816	5	4	20	17	12	3.730
Satellite scheduling (WCSP)							
29	82	4	2	13	15	8	-0.119
42b	190	4	2	13	19	10	-0.300
54	67	4	3	31	12	6	1.723
404	100	4	3	15	20	10	-0.280
408b	200	4	2	13	25	13	-0.570
503	143	4	3	63	10	5	2.976
505b	240	4	2	15	17	9	0.350
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. Results for experiments with 112 Bayesian networks and 30 weighted CSP instances

where t bounds the number of tuples in the relational specification.

A necessary condition for the hypertree width to yield a better bound, given a specific hypertree decomposition, is that $t < k^{\frac{w+1}{hw}}$.

In the remainder of this section we evaluate empirically the treewidth and hypertree width bounds on 112 practical probabilistic networks such as coding networks, dynamic Bayesian networks, genetic linkage instances, and CPCS networks used in medical diagnosis. We also look at 30 constraint networks such as radio frequency assignment and satellite scheduling problems. The problem instances were obtained from various sources, including the UAI'06 evaluation repository; all of them are made available online[4].

Since finding a tree or hypertree decomposition of minimal width is NP-complete, one usually resorts to heuristic methods. To obtain a tree decomposition of a given problem, we perform bucket elimination along a minfill ordering (random tie breaking, optimum over 20 iterations), a wide-spread and established method. The resulting tree decomposition is then extended to a hypertree decomposition by the method described in [1], where variables in a decomposition cluster are greedily covered by functions (by means of a set covering heuristic).

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 – defined as the number of zero cost tuples in CSPs and the number of non-zero probability tuples in belief networks. We also report the best treewidth and hypertree width found in the experiments described above. Lastly, we compute the following measure:

$$R = \log_{10}\left(\frac{t^{hw}}{k^{w}}\right)$$

This compares the two dominant factors in Equation 2 and 1. 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. The complete set of results is shown in Table 1.

Going over the numbers for belief networks, we note that out of the 112 problem instances, the hw bound is only superior for 5 instances, and not by many orders of magnitude. In particular, three of these are dynamic Bayesian network instances with low tightness $t = 208$ (out of $k^r = 91^4 = 68,574,961$ possible tuples) and small hypertree width.

For other instances that exhibit determinism, such as pedigree genetic linkage problems and partial k-trees that were randomly generated with determinism enforced, the hw bound is significantly worse, often by more than ten orders of magnitude. For the linkage instance pedigree42, for example, we have $k^w = 5^{24} = 59,604,644,775,390,625$. On the other hand, $t^{hw} = 50^{16} = 1,525,878,906,250,000,000,000,000,000$, which is a difference of ten orders of magnitude.

Several other belief network instances (all grid networks and most CPCS instances, for example) show little to no determinism ($t \approx k^r$), and the hw bound is inferior, as it was to be expected.

Moving to weighted CSP instances, the situation barely changes: Only for two instances of radio frequency assignment is the hw bound more than one order of magnitude better. Again, these problems exhibit a significant level of determinism in their function specifications. For satellite scheduling problems both bounds seem to fare roughly equally well overall, while the hw bound is again inferior on Mastermind game instances and digital circuit problems, on the latter often by more than 20 orders of magnitude.

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

In summary we can review that, in order for the hypertree width bound to be competitive with, or even superior to, the treewidth bound, problem instances need to comply with several conditions, foremost among these 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.

6 Conclusion

It is well known that a graph's treewidth provides bounds for many computational tasks over graphical models (e.g., satisfiability, counting, belief updating, finding the most likely explanation.). All these tasks are bounded exponentially by the graph treewidth. Also, compiled data-structures such as AND/OR BDDs and OBDDs are bounded exponentially by their underlying graph's treewidth and pathwidth respectively.

In this paper we demonstrated that the hypertree width bound, shown to provide a broader tractability characterization for constraint networks and for inference algorithms for general graphical models, is applicable to search algorithms as well when functions are specified as *relations* and as *decision trees*.

The primary contribution of the paper, however, is in exploring empirically the practical benefit of the hypertree width compared with the treewidth 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 treewidth, the complexity bound it implies is often inferior to the bound suggested by the treewidth. 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 treewidth. This demonstrates the added sensitivity of the hypertree width to the hypergraph structure and to the functions' specification.

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 two decades now.

REFERENCES

[1] G. Gottlob, B. McMahan, N. Musliu, A. Dermaku, T. Ganzow, and M. Samen, 'Heuristic methods for hypertree decompositions', in *Technical Report, Technische Universitaet Wien*, (2005).

[2] R. Dechter and R. Mateescu, 'And/or search spaces for graphical models', *Artificial Intelligence*, 73–106, (2007).

[3] R. Dechter and J. Pearl, 'Tree clustering for constraint networks', *Artificial Intelligence*, 353–366, (1989).

[4] N. Leone, G. Gottlob, and F. Scarcello, 'A comparison of structural CSP decomposition methods', *Artificial Intelligence*, 243–282, (2000).

[5] K. Kask, R. Dechter, J. Larrosa, and A. Dechter, 'Unifying tree-decompositions for reasoning in graphical models', *Artificial Intelligence*, **166(1-2)**, 165–193, (2005).

[6] S.L. Lauritzen and D.J. Spiegelhalter, 'Local computation with probabilities on graphical structures and their application to expert systems', *Journal of the Royal Statistical Society, Series B*, **50(2)**, 157–224, (1988).

[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, 1988.

[8] P.P. Shenoy, 'Binary join trees for computing marginals in the shenoy-shafer architecture', *International Journal of approximate reasoning*, 239–263, (1997).