RUTCOR
RESEARCH
REPORT

An algorithm for the acyclic hypergraph sandwich problem

Vladimir Gurvich [a] Nysret Musliu [b]
Vladimir Oudalov[c] Marko Samer [d]

RRR 37-2005, December 2005

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/∼rrr

[a]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway NJ 08854-8003; e-mail: gurvich@rutcor.rutgers.edu.

[b]Institut für Informationssysteme, TU Wien, Vienna, Austria; e-mail: musliu@dbai.tuwien.ac.at

[c]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway NJ 08854-8003; e-mail: oudalov@rutcor.rutgers.edu.

[d]Institut für Informationssysteme, TU Wien, Vienna, Austria; e-mail: samer@dbai.tuwien.ac.at

Rutcor Research Report
RRR 37-2005, December 2005

An algorithm for the acyclic hypergraph sandwich problem

Vladimir Gurvich Nysret Musliu Vladimir Oudalov
Marko Samer

Abstract. Given two hypergraphs H and H' on a common vertex-set, we write $H < H'$ if each edge of H is contained in an edge of H'. Given $H < H'$, either find an acyclic hypergraph A between them, $H < A < H'$, or claim that there is no such A. This problem is referred to as the Acyclic Hypergraph Sandwich Problem (AHSP). It generalizes the concept of treewidth as follows. Let $H = G$ be a graph, $|V(G)| = n$, and let $H' = \binom{n}{k}$ consist of all subsets of $V(G)$ of cardinality k. Then the AHSP is solvable if and only if the treewidth of G is strictly less than k, that is $TW(G) \leq k - 1$. Another important special case of the AHSP is $H' = H^k$, that is edges of H' are the unions of all subfamilies of k edges of H. In this case the AHSP generalizes the hypertreewidth of H. In this paper we suggest a simple general algorithm for the AHSP. Though exponential in the worst case, it runs well for many practically important tests with $H' = H^k$.
Key words: acyclic hypergraph, chordal graph, hypertreewidth, sandwich problem, treewidth, triangulated graph

Acknowledgements: This paper was supported by the Austrian Science Fund (FWF) project: *Nr. P17222-N04, Complementary Approaches to Constraint Satisfaction The research of the first author was partially supported by the National Science Foundation, Grant IIS-0118635,*

1 Introduction

In [4] one can find 12 equivalent definitions of the acyclic hypergraphs. We will need 2 of these 12.

Given a hypergraph H with the vertex set $V = V(H)$ and edge set set $E = E(H)$, introduce two operations $O1$ and $O2$ as follows:

$O1$. If $e \subseteq e'$ for a pair of edges $e, e' \in E(H)$ then delete e from $E(H)$.

$O2$. If $deg_H(v) = 1$ for a vertex $v \in V(H)$ then delete v from each edge which contains it.

Recall that $deg_H(v)$, the *degree* of v in H, is the number of edges in H which contain v.

It is not difficult to see that applying recursively $O1$ and $O2$ to H we get a unique (well defined) irreducible hypergraph H_O. In other words, though in some cases we can delete several vertices and/or edges, yet, our choice does not matter and the resulting hypergraph H_O is always the same. The proof can be found in [4, 25], where the above recursive procedure reducing H to H_O is attributed to Marc H. Graham [14].

Definition A. A hypergraph $A = (V, E)$ is *acyclic* if applying recursively $O1$ and $O2$ we can reduce A to nothing, that is A_O is the empty hypergraph.

Acyclic hypergraphs admit the following convenient representation. Given a tree T_0 and a family $\mathcal{T}$ of its subtrees, define a hypergraph $A = A(T_0, \mathcal{T})$ as follows: $V(A) = \mathcal{T}$, that is to each subtree $T \in \mathcal{T}$ we assign a vertex of A, while $E(A) = V(T_0)$, that is to each vertex $v \in T_0$ we assign an edge $e = e_v$ of A such that e consists of all subtrees $T \in \mathcal{T}$ which contain v.

Definition B. A hypergraph $A = (V, E)$ is *acyclic* if $A = A(T_0, \mathcal{T})$ for some T_0 and $\mathcal{T}$.

It is shown in [4] that definitions A and B are equivalent.

Given two hypergraphs H and H' we will write $H < H'$ if for every edge e of H there is an edge e' of H' such that $e \subseteq e'$. Obviously, this relation is transitive. Yet, it is possible that $H < H'$ and $H' < H$. Clearly, it happens if and only if applying $O1$ we obtain the same irreducible hypergraph from H and H'. In this case we will call H and H' *equivalent*.

Acyclic Hypergraph Sandwich Problem (AHSP) is formulated as follows. Given two hypergraphs H and H', either find an acyclic hypergraph A such that $H < A < H'$ or claim that there is no such A. The input of the AHSP is the pair of hypergraphs (H, H'). Obviously, one can assume that $H < H'$, since otherwise the AHSP has no solution for sure. It is also clear that the AHSP belongs to NP.

Remark 1 *The sandwich problems for graphs and hypergraphs are surveyed in [16, 15, 20]. Let us note, however, that the definitions are slightly different. In the cited papers it is assumed that three hypergraphs H, H' and A have the same number of edges which are labelled by the same set of indices $[m] = \{1, \ldots, m\}$ and $e_i(H) \subseteq e_i(A) \subseteq e_i(H')$ for each $i \in [m]$.*

Computing the treewidth [22] is reduced to the AHSP as follows. Let $H = G$ be an arbitrary graph, that is $|e| = 2$ for each $e \in E(H)$, and let $H' = \binom{n}{k}$, that is the edges of

$E(H')$ are all subsets of cardinality k of the given vertex set $V(H') = V(G)$ of cardinality n. Then the AHSP is solvable if and only if the treewidth of the graph G is strictly less than k, that is $TW(G) \leq k-1$. See surveys [6, 7] for this and many other equivalent characterizations of the treewidth.

Given G and k, it is NP-hard to decide whether $TW(G) \leq k-1$, [2]. However, this does not imply that the corresponding AHSP is NP-hard, since the hypergraph $H' = \binom{n}{k}$ may be exponential if k is a part of the input. Yet, if k is bounded then the inequality $TW(G) \leq k-1$ can be verified in polynomial time. The first polynomial algorithm was given in [22]; its performance time was n to a function of k. Then the time was successively reduced to $\emptyset(n^{k+2})$ in [2], $\emptyset(n^2)$ in [24], $\emptyset(n \log^2 n)$ in [3, 9], $\emptyset(n \log n)$ in [21], and finally to $\emptyset(n)$ in [5]. For $k \leq 5$ there are special, simpler, methods; see [7] section 3.3 or [6] section 3 for a survey.

Respectively, the AHSP (H, H') is polynomial when $H' = \binom{n}{k}$.

Another important special case of the AHSP is $H' = H^k$, that is $E(H')$ consists of the unions of all subfamilies of k edges from H. More precisely, $e \in E(H')$ if and only if $e = \cup_{i=1}^{k} e_i$, for some $e_1, \ldots, e_k \in E(H)$. In this case the AHSP generalizes the concept of hypertreewidth, [17, 18].

Remark 2 *Given an AHSP (H, H'), where $H' = H^k$ or $H' = \binom{n}{k}$, both H and H' or respectively H and k may be considered as the input. If k is bounded then these two possible inputs are polynomially equivalent, yet in general, the size of the first one may be exponential in size of the second one.*

In section 2 we suggest a simple general algorithm for the AHSP and in section 3 we report computational results of some tests for the treeweedth, $H' = \binom{n}{k}$, and hypertreewidth, $H' = H^k$.

2 Algorithm

The *neighborhood* $N_H(v)$ of a vertex v in a hypergraph H is defined as the union of all edges of H which contain v, that is $N_H(v) = \bigcup_{e \,|\, v \in e \in E(H)} e$. Let us remark that operation $O1$ keeps all neighborhoods unchanged.

AHSP : Given hypergraphs H and H', either get an acyclic hypergraph A such that $H < A < H'$ or claim that there is no such A.

Let us introduce two lists of hypergraphs $\mathcal{H}$ and $\mathcal{X}$; initialize $X = H$, $\mathcal{H} = \{H\}$ and $\mathcal{X} = \{X\}$. Then order arbitrarily the vertices of $V(H)$ and proceed as follows.

1. Obtain the hypergraph $Y = X_O$ from X applying recursively $O1$ and $O2$.
2. If Y is empty then output $A = H$ and halt.
3. Otherwise, find in $V(Y)$ the first vertex v satisfying property

(P) neighborhood $N_Y(v)$ is contained in an edge of H'.

4. Add $N_Y(v)$ to both sets $E(H)$ and $E(Y)$ as a new edge, denote the obtain two hypergraphs by H and X, and add them to the lists $\mathcal{H}$ and $\mathcal{X}$, respectively; then go to step 1.

5. If in the step 3 there is no v satisfying P then backtrack and reduce respectively the lists $\mathcal{H}$ and $\mathcal{X}$. Proceed with the depth first search; if the search is finished and the step 2 was never performed then claim that the AHSP has no solution.

Theorem 1 *This algorithm is finite and it solves the AHSP.*

Proof . Obviously, if A is output then $H < A < H'$. Indeed, $H < A$, since we only add some new edges to $E(H)$ in step 4, and $A < H'$, since, by property P, every new edge is contained in an edge of H'.

Now let us prove that the algorith is finite. It follows from the fact that between each two successive backtracks the algorithm is strictly monotone. Obviously, H is monotone increasing, $H^t < H^{t+1}$, since with each iteration we just add one new edge to $E(H)$. Clearly this inequality is strict (that is H^t and H^{t+1} are not equivalent), because the new edge e is a neighborhood of a vertex in the hypergraph $Y^t = H^t_O$ and hence e can not be contained in an edge of H^t.

On the contraty, the vertex set $V(X)$ (as well as $V(Y)$) is strictly monotone decreasing between each two successive backtracks. Indeed, after the edge $e = N_Y(v)$ is added to $E(Y)$ all esges containing v , except e , will be deleted from $E(Y)$ in the next step by operation $O1$, since e contains all such edges (and perhaps, some other edges, too). After this the vertex v itself (and perhaps, some other vertices, too) will be deleted by operation $O2$, since $deg(v) = 1$ in the obtained hypergraph. Thus, the vertex-set $V(Y)$ decreases by at least one vertex each round and hence the algorithm can run at most $|V(H)|$ rounds between two successive backtracks.

Let us also remark that the degrees of all vertices of $V(X)$ (as well as of $V(Y)$) are also monotone (not necessarily, strictly) decreasing. Indeed, when we add $e = N_Y(v)$ to $E(Y)$ the degrees of all vertices of e increase by 1; yet, on the next step we delete from $E(Y)$ all edges which contain v and the degrees of all vertices of e decrease by at least 1.

Though H is increasing, while X is decreasing, still, the equalities $H_O = X_O = Y$ hold all the time. It is easy to prove by induction. Indeed, let $H_O = X_O = Y$ and X' and H' are obtained from Y and H by adding an edge e to $E(Y)$ and $E(H)$, respectively. In general, we can not conclude that $H'_O = X'_O$. However, this equality does hold if $e = N_Y(v)$ is a neighborhood of a vertex in Y. Indeed, in this case the operations $O1$ and $O2$ can be applied to H' in exactly the same way they were applied to H in the previous round. The presence of the new edge e will change nothing, since the hypergraph Y is irreducible with respect to $O1$ and $O2$ and e is a neighborhood in Y.

Now it is easy to see that if A is output then it is is acyclic. Indeed, by step 2, we output $A = H$ if and only if the hypergraph Y is empty. Since $H_O = Y$, this means by definition that A is acyclic.

It remains to prove that the algorithm will find out a solution whenever there is one. Let us assume that for the current H^t there exists an acyclic A such that $H^t < A < H'$ and show that

(i) there exists a vertex $vinV(Y^t)$ such that adding $e = N_{Y^t}(v)$ to $E(H^t)$ we get H^{t+1} such that $H^t < H^{t+1} < A < H'$ and the first inequality is strict.

This claim would suffice, since applying it recursively we will get $H^{t+t'} = A$. (Though v satisfying (i) may be not the first in $V(Y^t)$, yet, it will be chosen before the algorithm backtracks from H^t, unless it finds out some other solution earlier.) Consider $Y^t = H^t_O$; by definition, it is irreducible with respect to $O1$ and $O2$. In particular, $deg_{Y^t}(v) \geq 2$. On the contrary, A is acyclic, that is it can be reduced to nothing by $O1$ and $O2$. Let us fix such a reduction sequence, choose the first vertex from $v \in V(Y^t)$ deleted by $O2$, and prove that v satisfies (i). Indeed, since v is deleted first, there should be an edge in $e' \in E(A)$ which contains $e = N_{Y^t}(v)$, otherwise, $deg_A(v)$ can not be reduced to 1, it would remain at least 2. Thus, after we add e to H^t the obtained H^{t+1} is still less than A and it is strictly larger than H^t, as we already know. □

The last arguments make an impression that A will be reached in polynomial time. Yet, this is not the case. Indeed, A is not given, we are just looking for it; so we may choose a "wrong" vertex $v \in Y^t$ and get H^{t+1} such that it is between H^t and H' but not between H^t and A for any acyclic A majorized by H', that is $H^t < H^{t+1} < H'$ holds but $H^{t+1} < A < H'$ holds for no A. Yet, since $H^{t+1} < H'$ we will proceed with our search, perhaps exponentially long time, until we backtrack from H^{t+1} to H^t again. Such situations do appear, as the following example shows.

Example 1 *Let H be an even cycle, that is $V(H) = \{1, \ldots, n\}$, where n is even, and $E(H) = \{(1,2), \ldots, (n-1,n), (n,1)$, and let H' be defined as follows: $V(H') = V(H)$ and $E(H') = \{(1,2,3), (2,3,4), \ldots, (n-1,n,1), (n,1,2); (1,3,\ldots,n-1)\}$. The AHSP (H, H') is solvable: for example, $A = \{(1,2,3), (3,4,5), \ldots, (n-1,n,1); (1,3,\ldots,n-1)\}$ is a solution. Each neighborhood $N_H(i) = \{i-1, i, i+1\}$ is contained in an edge of H'. Hence, the algorithm will delete vertex 1 first and we get a new hypergraph K, where $V(K) = \{2, \ldots, n\}$, and $E(K) = \{(2,3) \ldots, (n-1,n), (n,2)\}$. It is easy to see that now neither vertex n nor 2 can be deleted (since their neighborhoods $N_K(2) = \{n, 2, 3\}$ and $N_K(n) = \{n-1, n, 2\}$ are contained in no edge of H') until the algorithm (after exponentially many backtracs) returns to the first choice, vertex 1, and substitute it by 2. Clearly, it takes exponential time.*

Moreover, even if we will modify the algorithm and choose vertices at random, still the expected running time is exponential. Indeed, we succeed in eliminating all vertices if and only if the elimination order never creates two successive even vertices. In other words, we can delete $2i+1$ only after $2i$ and $2i+2$ are both deleted. (If none or only one of them is deleted then $2i+1$ is not qualified for being deleted.) The number of such orders divided by $n!$ is exponential in $1/n$.

In this example there is exactly one large edge $e = (1, 3, \ldots, n-1)$. Still, $dim(H') = |e| = n/2$ is not bounded by a constant. However, it is not difficult to construct an analogous example in which $dim(H')$ is bounded. To do so we just "triangulate" e. More precisely, we

keep H and $V(H')$ the same and set
$E(H') = (1,2,3),(2,3,4),\ldots,(n-2,n-1,n),(n,1,2);(1,3,n-1),(3,5,n-1),\ldots,(n-5,n-3,n-1)$.
It is not difficult to see that again we succeed in eliminating all vertices if and only if the elimination order never creates two successive even vertices. Hence, the running time is still exponential, though now $dim(H') = 3$.

3 Computing treeweedths and hypertreewidths

As we already mentioned, $H' = \binom{n}{k}$ in case of the treewidth (TW) and $H' = H^k$ for the hypertreewidth HW. In both cases the AHSP (H, H') is solvable if and only if $TW(H)$ and respectively $HW(H)$ is strictly less than k.

First, we consider the case when $H = G_n$ is the planar $n \times n$ grid, that is G_n has n^2 vertices $V(G_n) = \{1,\ldots,n^2\}$ and $2n(n-1)$ edges

$E(G_n) = \{(1,2),(2,3),\ldots,(n-1,n),(n+1,n+2),(n+2,n+3),\ldots,(2n-1,2n),\ldots,$
$(n^2-n+1,n^2-n+2),(n^2-n+2,n^2-n+3),\ldots,(n^2-1,n^2);$
$(1,n+1),(n+1,2n+1),\ldots,(n^2-2n+1,n^2-n+1),$
$(2,n+2),(n+2,2n+2),\ldots,(n^2-2n+2,n^2-n+2),\ldots,$
$(n,2n),(2n,3n),\ldots(n^2-n,n^2)\}$.

For example, if $n = 3$ then

$E(G_n) = \{(1,2),(2,3),(4,5),(5,6),(7,8),(8,9);(1,4),(4,7),(2,5),(5,8),(3,6),(6,9)\}$

For G_n the answers to both AHSPs are well-known: the treewidth $TW(G_n) = n$, [22], and the hypertreewidth $HW(G_n) = \lfloor n/2 \rfloor$. In other words, the AHSPs $(G_n, \binom{n}{k})$ and (G_n, G_n^k) are solvable if and only if $k \geq n+1$ and $k \geq \lfloor n/2 \rfloor + 1 = \lceil (n+1)/2 \rceil$, respectively. In case of equality, $k = k(n) = n+1$ and $k = k(n) = \lfloor n/2 \rfloor + 1$, the hypergraph

$A_n = \{(1,2,\ldots,n,n+1),(2,3\ldots,n+1,n+2),\ldots(n^2-n-1,n^2-n,\ldots,n^2-1,n^2)\}$.
is a common solution to both AHSPs. For example,

$A_3 = \{(1,2,3,4),(2,3,4,5),(3,4,5,6),(4,5,6,7),(5,6,7,8),(6,7,8,9)\}$.

Indeed, it is easy to see that A_n is acyclic and every edge of G_n is contained in an edge of A_n. Furthermore, the cardinality of each edge of A_n is $n+1$ and hence it can be covered by $k = \lfloor n/2 \rfloor + 1$ edges of G_n.

Table 1 represents the results for the treewidths. For each $n \times n$ grid G_n the number of vertices and edges are given in parentheses. The second column (Op. width) shows the treewidth. We get an upper bound as follows. First we apply the algorithm to solve the AHSP $(H, \binom{n}{k})$ and then minimize k for which it is solvable. In fact, we minimize k for which it is *solved* subject to the constraint: CPU-time < 100 seconds.

The performance depends a lot on the order in which the vertices of the grid are numbered. We consider three types of such an ordering: the "natural" order defined above, the random order, and the order of maximum-cadinality search (MCS) defined in ([25]). The results are given in Table 1.

Table 2 represents the corresponding results for the hypertreewidths

Table 1: Treewidth for grids with different order of nodes

		Original order		Random order		MCS order	
Example	Op. width	k	t(sec)	k	t(sec)	k	t(sec)
grid3 (9/12)	3	3	0	3	0	3	0
grid4 (16/24)	4	4	0	4	0	4	0
grid5 (25/40)	5	5	0	5	0	5	0
grid6(36/60)	6	6	0	7	9	6	0
grid7 (49/84)	7	7	1	10	1	8	1
grid8 (64/112)	8	8	2	11	2	9	2
grid9 (81/144)	9	9	5	13	3	11	4
grid10 (100/180)	10	10	9	14	6	11	8
grid11 (121/220)	11	11	15	16	8	12	15
grid12 (144/264)	12	12	23	21	14	14	21
grid13 (169/312)	13	13	35	25	25	15	37
grid14 (196/364)	14	14	53	22	28	16	63
grid15 (225/420)	15	15	77	27	47	17	93

Let us note that in case of the treewidth, $H' = \binom{n}{k}$, the condition (P) of the algorithm, $N_Y(v)$ is contained in an edge of H', becomes trivial: $|N_Y(v)| \leq k$. However, for the hypertreewidth, $H' = H^k$, given H and k, we have to compute $H' = H^k$ first. Though this problem is polynomial when k is bounded, still it is intractable, say, for $|E(H)| = 20$ and $k = 8$. To get an upper bound for the HW $k(n)$ we apply an alternative approach. Given H and k, verifying (P) turns into a set-covering problem. Instead of explicitly computing H^k, we apply set-covering heuristics. First, we apply the greedy algorithm from ([12]). This algorithm picks iteratively the set that covers the largest number of uncovered elements. The results are given in Table 2. Then we experimented with some more sophisticated set cover heuristics based on iterative improvements of the solution obtained by the greedy heuristic. The results for the grids are presented in Table 2.

Further we give the results for hypertreewidth of several real life problems which came from industry. These examples are hypergraph model of of adder circuits, bridge circuits, jet propulsion system. Table table:adderhyp shows that the algorithm can find the hypertreewidth for almost all examples in less than 100 seconds.

Table 2: Hypertreewidth width for grids with different order of nodes

		Original order		Random order		MCS order	
Example	Op. width	k	t(sec)	k	t(sec)	k	t(sec)
grid3 (9/12)	2	2	0	2	0	2	0
grid4 (16/24)	3	3	0	3	0	3	0
grid5 (25/40)	3	3	0	4	0	4	1
grid6(36/60)	4	4	0	5	0	4	1
grid7 (49/84)	4	4	2	6	67	5	2
grid8 (64/112)	5	5	4	7	2	6	3
grid9 (81/144)	5	5	7	9	3	7	4
grid10 (100/180)	6	6	12	10	5	9	8
grid11 (121/220)	6	7	21	12	11	9	12
grid12 (144/264)	7	7	32	15	16	9	23
grid13 (169/312)	7	8	33	14	25	12	37
grid14 (196/364)	8	8	50	15	59	13	63
grid15 (225/420)	8	9	74	18	42	13	91

4 Conclusions

The AHSP (H, H') is NP-complete. However, it is quasi-polynomial if $dim(H')$ is bounded and polynomial if $H' = \binom{n}{k}$ and k is bounded. The AHSP allows a simple general algorithm. Exponential in the worst case, it runs well for practically important tests, for which $H' = H^k$.

Acknowledgments The authors thank Georg Gottlob who suggested the AHSP and outlined its relation to the generalized hypertreewidth, also Endre Boros and Khaled Elbassioni for helpful discussions

References

[1] Richa Agarwala and David Fernandez-Baca, Fast and simple algorithms for perfect phylogeny and triangulating colored graphs, DIMACS Tech Report 94-51, 1994.

[2] Stefan Arnborg, D.G. Corneil, and A. Proskurowsky, Characterization and recognition of partial 3-trees, SIAM J. Alg. Disc. Meth. (1986) 7 : 305-314.

Table 3: Hypertreewidth for examples from industry

Example	Op. width	k	t(sec)
adder10 (71/51)	2	2	0
adder50 (351/251)	2	2	6
adder99 (694/496)	2	2	31
bridge10 (92/92)	2	2	2
bridge50 (452/452)	2	2	56
bridge99 (893/893)	2	2	330
atv-partial-system (125/88)	3	3	1
NewSystem1 (142/84)	3	4	0
NewSystem4 (718/418)	<= 4	4	26

[3] Stefan Arnborg, Jens Lagergren, and Detlef Seese, Easy problems for tree decomposable graphs, J. Algorithms (1991) 12 : 308-340

[4] Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis, On the desirability of database schemes, Journal of the ACM 30(3) : 479-513,(1983).

[5] Hans L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. (1996) 25 : 1305-1317.

[6] Hans L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth, Theoretical Computer Science (1998) 209 : 1-45.

[7] Hans L. Bodlaender, Discovering treewidth, In the Proceedings of the 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2005), LNCS, 3381.

[8] Hans L. Bodlaender, Mike R. Fellows, and Tandy J. Warnow, Two strikes against perfect phylogeny, In the Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP 1992), LNCS, 623 : 273-283.

[9] Hans Bodlaender and Ton Kloks, Better algorithms for path width and tree width of graphs, In the Proceedings of the 18th International Colloquium on Automata, Languages and Programming (1991) p. 544-555.

[10] P. Buneman, A characterization of rigid circuit graphs, Discrete Math. (1974) 9 : 205-212.

[11] R. Castelo and A. Siebes, A characterization of moral transitive directed acyclic graph Markov models as trees and its properties, Tech. Report, CS-2000-44, Univ. of Utrecht, 2000

[12] Vasek Chvatal, A greedy heuristic for the set covering problem, Mathematics of Operations Research, 4 (3) (1979) 233– 235.

[13] Vibhav Gogate and Rina Dechter, A complete anytime algorithm for treewidth, In Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence, UAI-04 (2004) p. 201-208.

[14] Marc H. Graham, On the universal relation, Computer System Research Group Report, University of Toronto, 1979.

[15] Martin Charles Golumbic, Haim Kaplan, and Ron Shamir, Graph sandwich problems. Journal of Algorithms 19(3) 449-473. (1995).

[16] Martin Charles Golumbic and Amir Wasserman, Complexity and algorithms for graph and hypergraph sandwich problems, Graph and Combinatorics (1998) 14 : 223 239.

[17] Georg Gottlob, Nicola Leone, and Francesco Scarcello, Hypertree Decompositions and Tractable Queries, Journal of Computer and System Sciences (JCSS), 64(3) 579-627, (2002).

[18] Georg Gottlob, Nicola Leone, and Francesco Scarcello, Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width, Journal of Computer and System Sciences (JCSS), 66(4) 775-808, (2003).

[19] C.W. Ho and R.C.T. Lee, Counting of clique-trees and computing perfect elimination schemes in parallel, Information Processing Letters (1989) 31 : 61 - 68.

[20] Aviv Lustig and Oded Smueli, Acyclic hypergraph projections, Journal of Algorithms 30(2) 400-422,(1999).

[21] Bruce A. Reed, Finding approximate separators and computing tree width quickly, Proc. 24-th Annual Symposium on Theory of Computing, New York (1992) 221-228; Preprint (1991).

[22] Neil Robertson and Paul D. Seymour, Disjoint paths - a survey, SIAM J. Alg. Disc. Meth. (1985) 6 : 300-305.

[23] Neil Robertson and Paul D. Seymour, Graph minors. II. Algorithmic aspects of tree-width. J. Algorithms (1986) 7 : 309-322.

[24] Neil Robertson and Paul D. Seymour, Graph minors. XIII. The disjoint path problem, J. Comb. Theory, ser. B, (1995) 63 (1) : 65-110; Preprint (1986).

[25] R.E. Tarjan and M. Yannakakis, Simple linear-time algorithm to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce ty8acyclic hypergraphs, SIAM J. Comput. (1984) 13 : 566-579.