# On The Complexity of Combinatorial Auctions: Structured Item Graphs and Hypertree Decompositions

[Extended Abstract]

Georg Gottlob
Computing Laboratory
Oxford University
OX1 3QD Oxford, UK
georg.gottlob@comlab.ox.ac.uk

Gianluigi Greco
Dipartimento di Matematica
University of Calabria
I-87030 Rende, Italy
ggreco@mat.unical.it

## ABSTRACT

The winner determination problem in combinatorial auctions is the problem of determining the allocation of the items among the bidders that maximizes the sum of the accepted bid prices. While this problem is in general NP-hard, it is known to be feasible in polynomial time on those instances whose associated item graphs have bounded treewidth (called structured item graphs). Formally, an item graph is a graph whose nodes are in one-to-one correspondence with items, and edges are such that for any bid, the items occurring in it induce a connected subgraph. Note that many item graphs might be associated with a given combinatorial auction, depending on the edges selected for guaranteeing the connectedness. In fact, the tractability of determining whether a structured item graph of a fixed treewidth exists (and if so, computing one) was left as a crucial open problem.

In this paper, we solve this problem by proving that the existence of a structured item graph is computationally intractable, even for treewidth 3. Motivated by this bad news, we investigate different kinds of structural requirements that can be used to isolate tractable classes of combinatorial auctions. We show that the notion of hypertree decomposition, a recently introduced measure of hypergraph cyclicity, turns out to be most useful here. Indeed, we show that the winner determination problem is solvable in polynomial time on instances whose bidder interactions can be represented with (dual) hypergraphs having bounded hypertree width. Even more surprisingly, we show that the class of tractable instances identified by means of our approach properly contains the class of instances having a structured item graph.

## Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*; F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

## General Terms

Algorithms, Economics, Theory

## Keywords

Hypergraphs, combinatorial auctions, hypertree decompositions



## 1. INTRODUCTION

**Combinatorial auctions.** Combinatorial auctions are well-known mechanisms for resource and task allocation where bidders are allowed to simultaneously bid on combinations of items. This is desirable when a bidder's valuation of a bundle of items is not equal to the sum of her valuations of the individual items. This framework is currently used to regulate agents' interactions in several application domains (cf., e.g., [21]) such as, electricity markets [13], bandwidth auctions [14], and transportation exchanges [18].

Formally, a *combinatorial auction* is a pair $\langle \mathcal{I}, \mathcal{B} \rangle$, where $\mathcal{I} = \{I_1, ..., I_m\}$ is the set of items the auctioneer has to sell, and $\mathcal{B} = \{B_1, ..., B_n\}$ is the set of bids from the buyers interested in the items in $\mathcal{I}$. Each bid $B_i$ has the form $\langle item(B_i), pay(B_i) \rangle$, where $pay(B_i)$ is a rational number denoting the price a buyer offers for the items in $item(B_i) \subseteq \mathcal{I}$. An outcome for $\langle \mathcal{I}, \mathcal{B} \rangle$ is a subset $\mathbf{b}$ of $\mathcal{B}$ such that $item(B_i) \cap item(B_j) = \emptyset$, for each pair $B_i$ and $B_j$ of bids in $\mathbf{b}$ with $i \neq j$.

**The winner determination problem.** A crucial problem for combinatorial auctions is to determine the outcome $\mathbf{b}^*$ that maximizes the sum of the accepted bid prices (i.e., $\sum_{B_i \in \mathbf{b}^*} pay(B_i)$) over all the possible outcomes. This problem, called *winner determination problem* (e.g., [11]), is known to be intractable, actually NP-hard [17], and even not approximable in polynomial time unless NP = ZPP [19]. Hence, it comes with no surprise that several efforts have been spent to design practically efficient algorithms for general auctions (e.g., [20, 5, 2, 8, 23]) and to identify classes of instances where solving the winner determination problem is feasible in polynomial time (e.g., [15, 22, 12, 21]). In fact, constraining bidder interaction was proven to be useful for identifying classes of tractable combinatorial auctions.

**Item graphs.** Currently, the most general class of tractable combinatorial auctions has been singled out by modelling interactions among bidders with the notion of *item graph*, which is a graph whose nodes are in one-to-one correspondence with items, and edges are such that for any

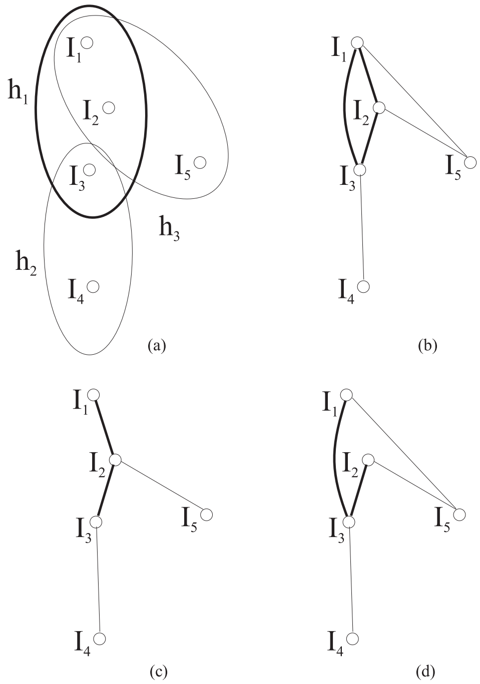


**Figure 1: Example MaxWSP problem: (a) Hypergraph $\mathcal{H}_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle}$, and a packing h for it; (b) Primal graph for $\mathcal{H}_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle}$; and, (c,d) Two item graphs for $\mathcal{H}_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle}$.**

bid, the items occurring in it induce a connected subgraph. Indeed, the winner determination problem was proven to be solvable in polynomial time if interactions among bidders can be represented by means of a *structured* item graph, i.e., a tree or, more generally, a graph having tree-like structure [3]—formally bounded *treewidth* [16].

To have some intuition on how item graphs can be built, we notice that bidder interaction in a combinatorial auction $\langle \mathcal{I},\mathcal{B}\rangle$ can be represented by means of a hypergraph $\mathcal{H}_{\langle \mathcal{I},\mathcal{B}\rangle}$ such that its set of nodes $\mathcal{N}(\mathcal{H}_{\langle \mathcal{I},\mathcal{B}\rangle})$ coincides with set of items $\mathcal{I}$, and where its edges $\mathcal{E}(\mathcal{H}_{\langle \mathcal{I},\mathcal{B}\rangle})$ are precisely the bids of the buyers $\{item(B_i) \mid B_i \in \mathcal{B}\}$. A special item graph for $\langle \mathcal{I},\mathcal{B}\rangle$ is the *primal graph* of $\mathcal{H}_{\langle \mathcal{I},\mathcal{B}\rangle}$, denoted by $G(\mathcal{H}_{\langle \mathcal{I},\mathcal{B}\rangle})$, which contains an edge between any pair of nodes in some hyperedge of $\mathcal{H}_{\langle \mathcal{I},\mathcal{B}\rangle}$. Then, any item graph for $\mathcal{H}_{\langle \mathcal{I},\mathcal{B}\rangle}$ can be viewed as a simplification of $G(\mathcal{H}_{\langle \mathcal{I},\mathcal{B}\rangle})$ obtained by deleting some edges, yet preserving the connectivity condition on the nodes included in each hyperedge.

Example 1. The hypergraph $\mathcal{H}_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle}$ reported in Figure 1.(a) is an encoding for a combinatorial auction $\langle \mathcal{I}_0,\mathcal{B}_0\rangle$, where $\mathcal{I}_0 = \{\mathcal{I}_1, ..., \mathcal{I}_5\}$, and $item(B_i) = h_i$, for each $1 \leq i \leq 3$. The primal graph for $\mathcal{H}_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle}$ is reported in Figure 1.(b), while two example item graphs are reported in Figure 1.(c) and (d), where edges required for maintaining the connectivity for $h_1$ are depicted in bold. $\lhd$

**Open Problem: Computing structured item graphs efficiently.** The above mentioned tractability result on structured item graphs turns out to be useful in practice only when a structured item graph either is given or can be efficiently determined. However, exponentially many item graphs might be associated with a combinatorial auction, and it is not clear how to determine whether a structured item graph of a certain (constant) treewidth exists, and if so, how to compute such a structured item graph efficiently.

Polynomial time algorithms to find the "best" simplification of the primal graph were so far only known for the cases where the item graph to be constructed is a line [10], a cycle [4], or a tree [3], but it was an important open problem (cf. [3]) whether it is tractable to check if for a combinatorial auction, an item graph of treewidth bounded by a fixed natural number $k$ exists and can be constructed in polynomial time, if so.

**Weighted Set Packing.** Let us note that the hypergraph representation $\mathcal{H}_{\langle \mathcal{I},\mathcal{B}\rangle}$ of a combinatorial auction $\langle \mathcal{I},\mathcal{B}\rangle$ is also useful to make the analogy between the winner determination problem and the *maximum weighted-set packing problem* on hypergraphs clear (e.g., [17]).

Formally, a *packing* $\mathbf{h}$ for a hypergraph $\mathcal{H}$ is a set of hyperedges of $\mathcal{H}$ such that for each pair $h, h' \in \mathbf{h}$ with $h \neq h'$, it holds that $h \cap h' = \emptyset$. Letting $w$ be a *weighting function* for $\mathcal{H}$, i.e., a polynomially-time computable function from $\mathcal{E}(\mathcal{H})$ to rational numbers, the weight of a packing $\mathbf{h}$ is the rational number $w(\mathbf{h}) = \sum_{h\in\mathbf{h}} w(h)$, where $w(\{\}) = 0$. Then, the maximum-weighted set packing problem for $\mathcal{H}$ w.r.t. $w$, denoted by $\texttt{MaxWSP}(\mathcal{H}, w)$, is the problem of finding a packing for $\mathcal{H}$ having the maximum weight over all the packings for $\mathcal{H}$. To see that MaxWSP is just a different formulation for the winner determination problem, given a combinatorial auction $\langle \mathcal{I},\mathcal{B}\rangle$, it is sufficient to define the weighting function $w_{\langle \mathcal{I},\mathcal{B}\rangle}(item(B_i)) = pay(B_i)$. Then, the set of the solutions for the weighted set packing problem for $\mathcal{H}_{\langle \mathcal{I},\mathcal{B}\rangle}$ w.r.t. $w_{\langle \mathcal{I},\mathcal{B}\rangle}$ coincides with the set of the solutions for the winner determination problem on $\langle \mathcal{I},\mathcal{B}\rangle$.

Example 2. Consider again the hypergraph $\mathcal{H}_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle}$ reported in Figure 1.(a). An example packing for $\mathcal{H}_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle}$ is $\mathbf{h} = \{h_1\}$, which intuitively corresponds to an outcome for $\langle \mathcal{I}_0,\mathcal{B}_0\rangle$, where the auctioneer accepted the bid $B_1$. By assuming that bids $B_1$, $B_2$, and $B_3$ are such that $pay(B_1) = pay(B_2) = pay(B_3)$, the packing $\mathbf{h}$ is not a solution for the problem $\texttt{MaxWSP}(\mathcal{H}_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle}, w_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle})$. Indeed, the packing $\mathbf{h}^* = \{h_2, h_3\}$ is such that $w_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle}(\mathbf{h}^*) > w_{\langle \mathcal{I}_0,\mathcal{B}_0\rangle}(\mathbf{h})$. $\lhd$

## Contributions

The primary aim of this paper is to identify large tractable classes for the winner determination problem, that are, moreover polynomially recognizable. Towards this aim, we first study structured item graphs and solve the open problem in [3]. The result is very bad news:

- ► It is NP complete to check whether a combinatorial auction has a structured item graph of treewidth 3. More formally, letting $\mathcal{C}(\texttt{ig}, k)$ denote the class of all the hypergraphs having an item tree of treewidth bounded by $k$, we prove that deciding whether a hypergraph (associated with a combinatorial auction problem) belongs to $\mathcal{C}(\texttt{ig}, 3)$ is NP-complete.

In the light of this result, it was crucial to assess whether there are some other kinds of structural requirement that can be checked in polynomial time and that can still be used to isolate tractable classes of the maximum weighted-set packing problem or, equivalently, the winner determination problem. Our investigations, this time, led to very good news which are summarized below:

- ► For a hypergraph $\mathcal{H}$, its dual $\bar{\mathcal{H}} = (V, E)$ is such that nodes in $V$ are in one-to-one correspondence with hyperedges in $\mathcal{H}$, and for each node $x \in \mathcal{N}(\mathcal{H})$, $\{h \mid x \in h \wedge h \in$

$\mathcal{E}(\mathcal{H})\}$ is in $E$. We show that MaxWSP is tractable on the class of those instances whose dual hypergraphs have *hypertree width*[7] bounded by $k$ (short: class $\mathcal{C}(\overline{\mathtt{hw}}, k)$ of hypergraphs). Note that a key issue of the tractability is to consider the hypertree width of the *dual* hypergraph $\bar{\mathcal{H}}$ instead of the auction hypergraph $\mathcal{H}$. In fact, we can show that MaxWSP remains NP-hard even when $\mathcal{H}$ is acyclic (i.e., when it has hypertree width 1), even when each node is contained in 3 hyperedges at most.

► For some relevant special classes of hypergraphs in $\mathcal{C}(\overline{\mathtt{hw}}, k)$, we design a higly-parallelizeable algorithm for MaxWSP. Specifically, if the weighting functions can be computed in logarithmic space and weights are polynomial (e.g., when all the hyperegdes have unitary weights and one is interested in finding the packing with the maximum number of edges), we show that MaxWSP can be solved by a LOGCFL algorithm. Recall, in fact, that LOGCFL is the class of decision problems that are logspace reducible to context free languages, and that LOGCFL $\subseteq$ NC$_2$ $\subseteq$ P (see, e.g., [9]).

► Surprisingly, we show that nothing is lost in terms of generality when considering the hypertree decomposition of dual hypergraphs instead of the treewidth of item graphs. To the contrary, the proposed hypertree-based decomposition method is strictly more general than the method of structured item graphs. In fact, we show that strictly larger classes of instances are tractable according to our new approach than according to the structured item graphs approach. Intuitively, the NP-hardness of recognizing bounded-width structured item graphs is thus not due to its great generality, but rather to some peculiarities in its definition.

► The proof of the above results give us some interesting insight into the notion of structured item graph. Indeed, we show that structured item graphs are in one-to-one correspondence with some special kinds of hypertree decomposition of the dual hypergraph, which we call *strict hypertree decompositions*. A game-characterization for the notion of strict hypertree width is also proposed, which specializes the Robber and Marshals game in [6] (proposed to characterize the hypertree width), and which makes it clear the further requirements on hypertree decompositions.

The rest of the paper is organized as follows. Section 2 discusses the intractability of structured item graphs. Section 3 presents the polynomial-time algorithm for solving MaxWSP on the class of those instances whose dual hypergraphs have bounded hypertree width, and discusses the cases where the algorithm is also highly parallelizable. The comparison between the classes $\mathcal{C}(\mathtt{ig}, k)$ and $\mathcal{C}(\overline{\mathtt{hw}}, k)$ is discussed in Section 4. Finally, in Section 5 we draw our conclusions by also outlining directions for further research.

## 2. COMPLEXITY OF STRUCTURED ITEM GRAPHS

Let $\mathcal{H}$ be a hypergraph. A graph $G = (V, E)$ is an *item graph* for $\mathcal{H}$ if $V = \mathcal{N}(\mathcal{H})$ and, for each $h \in \mathcal{E}(\mathcal{H})$, the subgraph of $G$ induced over the nodes in $h$ is connected. An important class of item graphs is that of *structured* item graphs, i.e., of those item graphs having bounded treewidth as formalized below.

A *tree decomposition* [16] of a graph $G = (V, E)$ is a pair $\langle T, \chi\rangle$, where $T = (N, F)$ is a tree, and $\chi$ is a labelling function assigning to each vertex $p \in N$ a set of vertices $\chi(p) \subseteq V$, such that the following conditions are satisfied: (1) for each vertex $b$ of $G$, there exists $p \in N$ such that $b \in \chi(p)$; (2) for each edge $\{b, d\} \in E$, there exists $p \in N$ such that $\{b, d\} \subseteq \chi(p)$; (3) for each vertex $b$ of $G$, the set $\{p \in N \mid b \in \chi(p)\}$ induces a connected subtree of $T$. The *width* of $\langle T, \chi\rangle$ is the number $\max_{p \in N} |\chi(p) - 1|$. The *treewidth* of $G$, denoted by $tw(G)$, is the minimum width over all its tree decompositions.

The winner determination problem can be solved in polynomial time on item graphs having bounded treewidth [3].

Theorem 1 (cf. [3]). *Assume a $k$-width tree decomposition $\langle T, \chi\rangle$ of an item graph for $\mathcal{H}$ is given. Then,* MaxWSP$(\mathcal{H}, w)$ *can be solved in time* $O(|T|^2 \times (|\mathcal{E}(\mathcal{H})|+1)^{k+1})$.

Many item graphs can be associated with a hypergraph. As an example, observe that the item graph in Figure 1.(c) has treewidth 1, while Figure 1.(d) reports an item graph whose treewidth is 2. Indeed, it was an open question whether for a given constant $k$ it can be checked in polynomial time if an item graph of treewidth $k$ exists, and if so, whether such an item graph can be efficiently computed.

Let $\mathcal{C}(\mathtt{ig}, k)$ denote the class of all the hypergraphs having an item graph $G$ such that $tw(G) \leq k$. The main result of this section is to show that the class $\mathcal{C}(\mathtt{ig}, k)$ is hard to recognize.

Theorem 2. *Deciding whether a hypergraph $\mathcal{H}$ belongs to $\mathcal{C}(\mathtt{ig}, 3)$ is NP-hard.*

The proof of this result relies on an elaborate reduction from the *Hamiltonian path problem* $\mathbf{HP}(s, t)$ of deciding whether there is an Hamiltonian path from a node $s$ to a node $t$ in a directed graph $G = (N, E)$. To help the intuition, we report here a high-level overview of the main ingredients exploited in the proof[1].

The general idea it to build a hypergraph $\mathcal{H}_G$ such that there is an item graph $G'$ for $\mathcal{H}_G$ with $tw(G') \leq 3$ if and only if $\mathbf{HP}(s, t)$ over $G$ has a solution. First, we discuss the way $\mathcal{H}_G$ is constructed. See Figure 2.(a) for an illustration, where the graph $G$ consists of the nodes $s$, $x$, $y$, and $t$, and the set of its edges is $\{e_1 = (s, x), e_2 = (x, y), e_3 = (x, t), e_4 = (y, t)\}$.

**From $G$ to $\mathcal{H}_G$.** Let $G = (N, E)$ be a directed graph. Then, the set of the nodes in $\mathcal{H}_G$ is such that: for each $x \in N$, $\mathcal{N}(\mathcal{H}_G)$ contains the nodes $bs_x$, $bt_x$, $b'_x$, $b''_x$, $bd_x$; for each $e = (x, y) \in E$, $\mathcal{N}(\mathcal{H}_G)$ contains the nodes $ns'_x$, $ns''_x$, $nt'_y$, $nt''_y$, $ns^e_x$ and $nt^e_y$. No other node is in $\mathcal{N}(\mathcal{H}_G)$.
Hyperedges in $\mathcal{H}_G$ are of three kinds:

1) for each $x \in N$, $\mathcal{E}(\mathcal{H}_G)$ contains the hyperedges:
   - $S_x = \{bs_x\} \cup \{ns^e_x \mid e = (x, y) \in E\}$;
   - $T_x = \{bt_x\} \cup \{nt^e_x \mid e = (z, x) \in E\}$;
   - $A^1_x = \{bd_x, b'_x\}$, $A^2_x = \{bd_x, b''_x\}$, and $A^3_x = \{b'_x, b''_x\}$ —notice that these hyperedges induce a clique on the nodes $\{b'_x, b''_x, bd_x\}$;

[1] Detailed proofs can be found in the Appendix, available at www.mat.unical.it/∼ggreco/papers/ca.pdf.

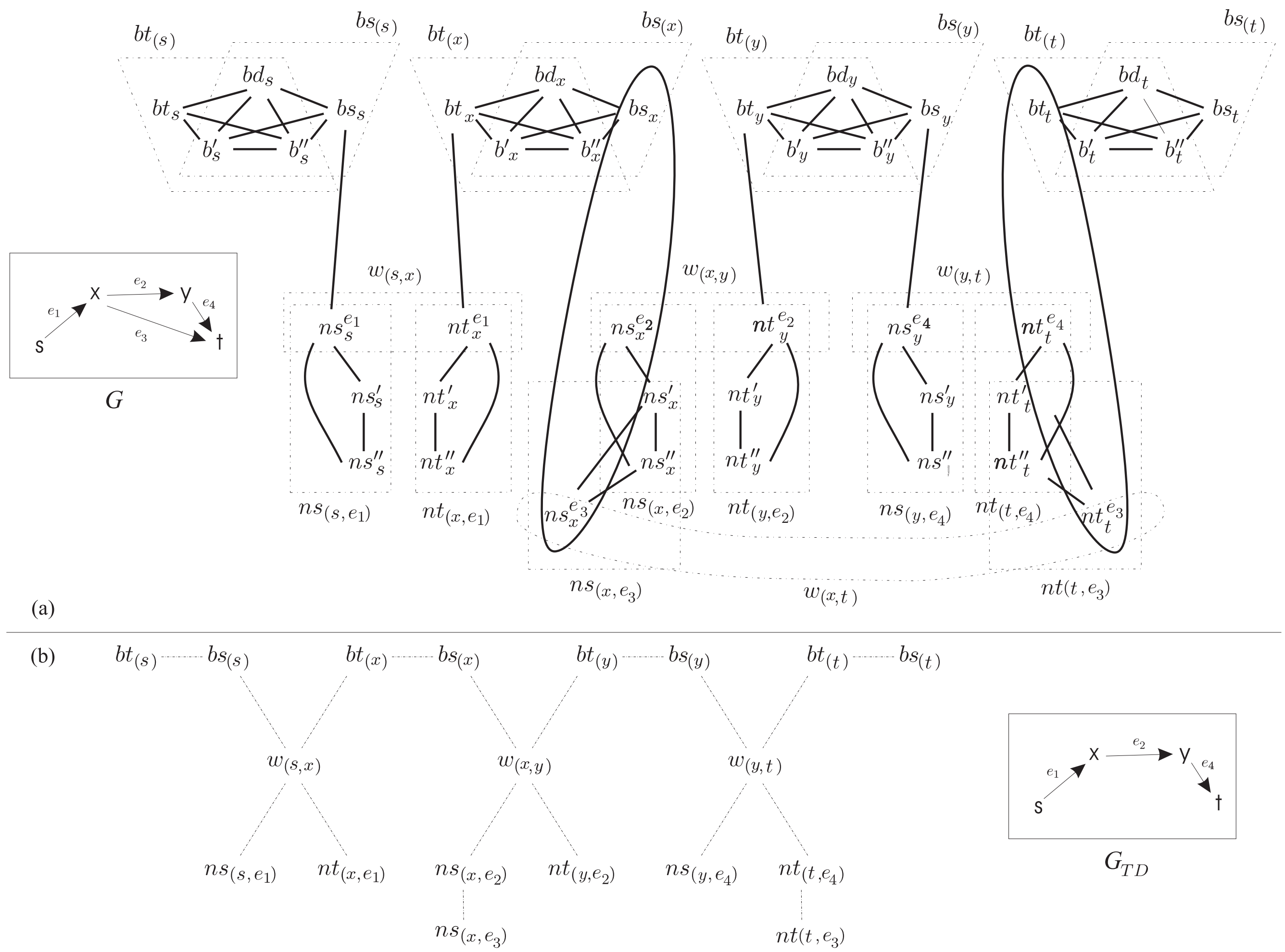


**Figure 2: Proof of Theorem 2: (a) from $G$ to $\mathcal{H}_G$ — hyperedges in 1) and 2) are reported only; (b) a skeleton for a tree decomposition $TD$ for $\mathcal{H}_G$.**

- $SA_x^1 = \{bs_x, b'_x\}$, $SA_x^2 = \{bs_x, b''_x\}$, $SA_x^3 = \{bs_x, bd_x\}$ —notice that these hyperedges plus $A_x^1$, $A_x^2$, and $A_x^3$ induce a clique on the nodes $\{bs_x, b'_x, b''_x, bd_x\}$;
- $TA_x^1 = \{bt_x, b'_x\}$, $TA_x^2 = \{bt_x, b''_x\}$, and $TA_x^3 = \{bt_x, bd_x\}$ —notice that these hyperedges plus $A_x^1$, $A_x^2$, and $A_x^3$ induce a clique on the nodes $\{bt_x, b'_x, b''_x, bd_x\}$;

2) for each $e = (x, y) \in E$, $\mathcal{E}(\mathcal{H}_G)$ contains the hyperedges:

- $SH_x = \{ns'_x, ns''_x\}$;
- $TH_y = \{nt'_y, nt''_y\}$;
- $SE'_e = \{ns'_x, ns^e_x\}$ and $SE''_e = \{ns''_x, ns^e_x\}$ —notice that these two hyperedges plus $SH_x$ induce a clique on the nodes $\{ns'_x, ns''_x, ns^e_x\}$;
- $TE'_e = \{nt'_y, nt^e_y\}$ and $TE''_e = \{nt''_y, nt^e_y\}$ —notice that these two hyperedges plus $TH_y$ induce a clique on the nodes $\{nt'_y, nt''_y, nt^e_y\}$.

Notice that each of the above hyperedges but those of the form $S_x$ and $T_x$ contains exactly two nodes. As an example of the hyperedges of kind 1) and 2), the reader may refer to the example construction reported in Figure 2.(a), and notice, for instance, that $S_x = \{bs_x, ns_x^{e_2}, ns_x^{e_3}\}$ and that $T_t = \{bt_t, nt_t^{e_4}, nt_t^{e_3}\}$.

3) finally, we denote by $\mathcal{D}_G$ the set containing the hyperedges in $\mathcal{E}(\mathcal{H}_G)$ of the third kind. In the reduction we are exploiting, $\mathcal{D}_G$ can be an arbitrary set of hyperedges satisfying the four conditions that are discussed below. Let $\mathcal{P}_G$ be the set of the following $|\mathcal{P}_G| \leq |N| + 3 \times |E|$ pairs: $\mathcal{P}_G = \{(b'_x, b''_x) \mid x \in N\} \cup \{(ns'_x, ns''_x),\ (nt'_y, nt''_y),\ (ns^e_x, nt^e_y) \mid e = (x, y) \in E\}$.
Also, let $\mathcal{I}(v)$ denote the set $\{h \in \mathcal{E}(\mathcal{H}) \mid v \in h\}$ of the hyperedges of $\mathcal{H}$ that are touched by $v$; and, for a set $V \subseteq \mathcal{N}(\mathcal{H})$, let $\mathcal{I}(V) = \bigcup_{v \in V} \mathcal{I}(v)$. Then, $\mathcal{D}_G$ has to be a set such that:

(c1) $\forall (\alpha, \beta) \in \mathcal{P}_G$, $\mathcal{I}(\alpha) \cap \mathcal{I}(\beta) \cap \mathcal{D}_G = \emptyset$;
(c2) $\forall (\alpha, \beta) \in \mathcal{P}_G$, $\mathcal{I}(\alpha) \cup \mathcal{I}(\beta) \supseteq \mathcal{D}_G$;
(c3) $\forall \alpha \in N$ such that $\nexists \beta \in N$ with $(\alpha, \beta) \in \mathcal{P}_G$ or $(\beta, \alpha) \in \mathcal{P}_G$, it holds: $\mathcal{I}(\alpha) \cap \mathcal{D}_G = \emptyset$; and,
(c4) $\forall S \subseteq N$ such that $|S| \leq 3$ and where $\nexists \alpha, \beta \in S$ with $(\alpha, \beta) \in \mathcal{P}_G$, it is the case that: $\mathcal{I}(S) \not\supseteq \mathcal{D}_G$.

Intuitively, the set $\mathcal{D}_G$ is such that each of its hyperedges is touched by exactly one of the two nodes in every pair of $\mathcal{P}_G$ — cf. (c1) and (c2). Moreover, hyperedges in $\mathcal{D}_G$ touch only vertices included in at least a pair of $\mathcal{P}_G$ — cf. (c3); and, any triple of nodes is not capable of touching all the elements of $\mathcal{D}_G$ if none of the pairs that can be built from it belongs to $\mathcal{P}_G$ — cf. (c4).

The reader may now ask whether a set $\mathcal{D}_G$ exists at all satisfying (c1), (c2), (c3) and (c4). In the following lemma, we positively answer this question and refer the reader to its proof for an example construction.

Lemma 1. *A set $\mathcal{D}_G$, with $|\mathcal{D}_G| = 2 \times |\mathcal{P}_G| + 2$, satisfying conditions (c1), (c2), (c3), and (c4) can be built in time $O(|\mathcal{P}_G|^2)$.*

**Key Ingredients.** We are now in the position of presenting an overview of the key ingredients of the proof. Let $G'$ be an arbitrary item graph for $\mathcal{H}_G$, and let $TD = \langle T, \chi \rangle$ be a 3-width tree decomposition of $G'$ (note that, because of the cliques, e.g., on the nodes $\{bs_x, b'_x, b''_x, bd_x\}$, any item graph for $\mathcal{H}_G$ has treewidth 3 at least).

There are three basic observations serving the purpose of proving the correctness of the reduction.

**"Blocks" of $TD$:** First, we observe that $TD$ must contain some special kinds of vertex. Specifically, for each node $x \in N$, $TD$ contains a vertex $bs_{(x)}$ such that $\chi(bs_{(x)}) \supseteq \{bs_x, b'_x, b''_x, bd_x\}$, and a vertex $bt_{(x)}$ such that $\chi(bt_{(x)}) \supseteq \{bt_x, b'_x, b''_x, bd_x\}$. And, for each edge $e = (x, y) \in E$, $TD$ contains a vertex $ns_{(x,e)}$ such that $\chi(ns_{(x,e)}) \supseteq \{ns^e_x, ns'_x, ns''_x\}$, and a vertex $nt_{(y,e)}$ such that $\chi(nt_{(y,e)}) \supseteq \{nt^e_y, nt'_y, nt''_y\}$.

Intuitively, these vertices are required to cover the cliques of $\mathcal{H}_G$ associated with the hyperedges of kind 1) and 2). Each of these vertices plays a specific role in the reduction. Indeed, each directed edge $e = (x, y) \in E$ is encoded in $TD$ by means of the vertices: $ns_{(x,e)}$, representing precisely that $e$ *starts* from $x$; and, $nt_{(y,e)}$, representing precisely that $e$ *terminates* into $y$. Also, each node $x \in N$ is encoded in $TD$ be means of the vertices: $bs_{(x)}$, representing the *starting* point of edges originating from $x$; and, $bt_{(x)}$, representing the *terminating* point of edges ending into $x$. As an example, Figure 2.(b) reports the "skeleton" of a tree decomposition $TD$. The reader may notice in it the blocks defined above and how they are related with the hypergraph $\mathcal{H}_G$ in Figure 2.(a) — other blocks in it (of the form $w_{(x,y)}$) are defined next.

**Connectedness between blocks, and uniqueness of the connections:** The second crucial observation is that in the path connecting a vertex of the form $bs_{(x)}$ (resp., $bt_{(y)}$) with a vertex of the form $ns_{(x,e)}$ (resp., $nt_{(y,e)}$) there is one special vertex of the form $w_{(x,y)}$ such that: $\chi(w_{(x,y)}) \supseteq \{ns^{e'}_x, nt^{e'}_y\}$, for some edge $e' = (x, y) \in E$. Guaranteeing the existence of one such vertex is precisely the role played by the hyperedges in $\mathcal{D}_G$. The arguments for the proof are as follows. First, we observe that $\mathcal{I}(\chi(bs_{(x)})) \cap \mathcal{I}(\chi(ns_{(x,e)})) \supseteq \mathcal{D}_G \cup \{S_x\}$ and $\mathcal{I}(\chi(bt_{(y)})) \cap \mathcal{I}(\chi(nt_{(y,e)})) \supseteq \mathcal{D}_G \cup \{T_y\}$. Then, we show a property stating that for a pair of consecutive vertices $p$ and $q$ in the path connecting $bs_{(x)}$ and $ns_{(x,e)}$ (resp., $bt_{(y)}$ and $nt_{(y,e)}$), $\mathcal{I}(\chi(p) \cap \chi(q)) \supseteq \mathcal{I}(\chi(bs_{(x)})) \cap \mathcal{I}(\chi(ns_{(x,e)}))$ (resp., $\mathcal{I}(\chi(p) \cap \chi(q)) \supseteq \mathcal{I}(\chi(bt_{(x)})) \cap \mathcal{I}(\chi(nt_{(y,e)}))$). Thus, we have: $\mathcal{I}(\chi(p) \cap \chi(q)) \supseteq \mathcal{D}_G \cup \{S_x\}$ (resp., $\mathcal{I}(\chi(p) \cap \chi(q)) \supseteq \mathcal{D}_G \cup \{T_y\}$). Based on this observation, and by exploiting the properties of the hyperedges in $\mathcal{D}_G$, it is not difficult to show that any pair of consecutive vertices $p$ and $q$ must share two nodes of $\mathcal{H}_G$ forming a pair in $\mathcal{P}_G$, and must both touch $S_x$ (resp., $T_y$). When the treewidth of $G'$ is 3, we can conclude that a vertex, say $w_{(x,y)}$, in this path is such that $\chi(w_{(x,y)}) \supseteq \{ns^{e'}_x, nt^{e'}_y\}$, for some edge $e' = (x, y) \in E$ — to this end, note that $ns^{e'}_x \in S_x$, $nt^{e'}_t \in T_y$, and $\mathcal{I}(\chi(w_{(x,y)})) \supseteq \mathcal{D}_G$. In particular, $w_{(x,y)}$ is the only kind of vertex satisfying these conditions, i.e., in the path there is no further vertex of the form $w_{(x,z)}$, for $z \neq y$ (resp., $w_{(z,y)}$, for $z \neq x$).

To help the intuition, we observe that having a vertex of the form $w_{(x,y)}$ in $TD$ corresponds to the selection of an edge from node $x$ to node $y$ in the Hamiltonian path. In fact, given the uniqueness of these vertices selected for ensuring the connectivity, a one-to-one correspondence can be established between the existence of a Hamiltonian path for $G$ and the vertices of the form $w_{(x,y)}$. As an example, in Figure 2.(b), the vertices of the form $w_{(s,x)}$, $w_{(x,y)}$, and $w_{(y,t)}$ are in $TD$, and $G_{TD}$ shows the corresponding Hamiltonian path.

**Unused blocks:** Finally, the third ingredient of the proof is the observation that if a vertex of the form $w_{(x,y)}$, for an edge $e' = (x, y) \in E$ is not in $TD$ (i.e., if the edge $(x, y)$ does not belong to the Hamiltonian path), then the corresponding block $ns_{(x,e')}$ (resp., $nt_{(y,e')}$) can be arbitrarily appended in the subtree rooted at the block $ns_{(x,e)}$ (resp., $nt_{(y,e)}$), where $e$ is the edge of the form $e = (x, z)$ (resp., $e = (z, y)$) such that $w_{(x,z)}$ (resp., $w_{(z,y)}$) is in $TD$.

E.g., Figure 2.(a) shows $w_{(x,t)}$, which is not used in $TD$, and Figure 2.(b) shows how the blocks $ns_{(x,e_3)}$ and $nt_{(t,e_3)}$ can be arranged in $TD$ for ensuring the connectedness condition.

# 3. TRACTABLE CASES VIA HYPERTREE DECOMPOSITIONS

Since constructing structured item graphs is intractable, it is relevant to assess whether other structural restrictions can be used to single out classes of tractable `MaxWSP` instances. To this end, we focus on the notion of *hypertree decomposition* [7], which is a natural generalization of hypergraph acyclicity and which has been profitably used in other domains, e.g, constraint satisfaction and database query evaluation, to identify tractability islands for NP-hard problems.

A *hypertree for a hypergraph $\mathcal{H}$* is a triple $\langle T, \chi, \lambda \rangle$, where $T = (N, E)$ is a rooted tree, and $\chi$ and $\lambda$ are labelling functions which associate each vertex $p \in N$ with two sets $\chi(p) \subseteq \mathcal{N}(\mathcal{H})$ and $\lambda(p) \subseteq \mathcal{E}(\mathcal{H})$. If $T' = (N', E')$ is a subtree of $T$, we define $\chi(T') = \bigcup_{v \in N'} \chi(v)$. We denote the set of vertices $N$ of $T$ by $vertices(T)$. Moreover, for any $p \in N$, $T_p$ denotes the subtree of $T$ rooted at $p$.

Definition 1. A *hypertree decomposition* of a hypergraph $\mathcal{H}$ is a hypertree $HD = \langle T, \chi, \lambda \rangle$ for $\mathcal{H}$ which satisfies all the following conditions:

1. for each edge $h \in \mathcal{E}(\mathcal{H})$, there exists $p \in vertices(T)$ such that $h \subseteq \chi(p)$ (we say that $p$ *covers* $h$);

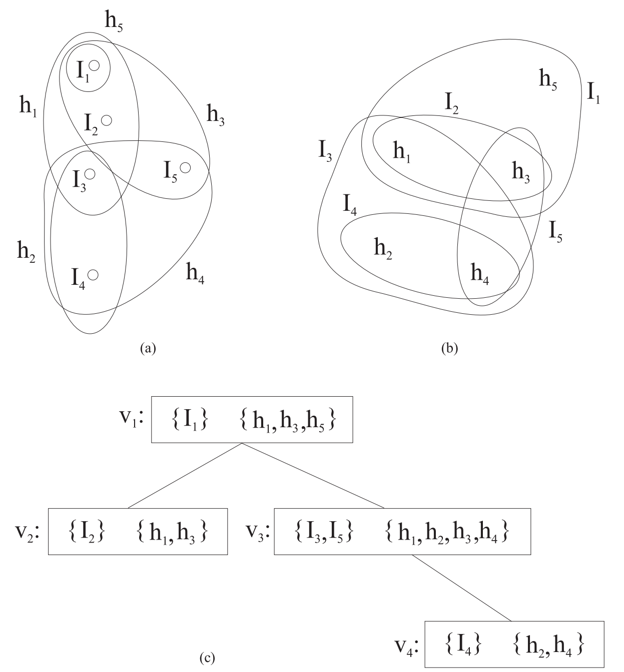


**Figure 3: Example MaxWSP problem: (a) Hypergraph $\mathcal{H}_1$; (b) Hypergraph $\bar{\mathcal{H}}_1$; (b) A 2-width hypertree decomposition of $\bar{\mathcal{H}}_1$.**

2. for each node $Y \in \mathcal{N}(\mathcal{H})$, the set $\{p \in vertices(T) \mid Y \in \chi(p)\}$ induces a (connected) subtree of $T$;
3. for each $p \in vertices(T)$, $\chi(p) \subseteq \mathcal{N}(\lambda(p))$;
4. for each $p \in vertices(T)$, $\mathcal{N}(\lambda(p)) \cap \chi(T_p) \subseteq \chi(p)$.

The *width* of a hypertree decomposition $\langle T, \chi, \lambda \rangle$ is $max_{p \in vertices(T)} |\lambda(p)|$. The HYPERTREE *width* $hw(\mathcal{H})$ of $\mathcal{H}$ is the minimum width over all its hypertree decompositions. A hypergraph $\mathcal{H}$ is acyclic if $hw(\mathcal{H}) = 1$. □

EXAMPLE 3. The hypergraph $\mathcal{H}_{\langle \mathcal{I}_0, \mathcal{B}_0 \rangle}$ reported in Figure 1.(a) is an example acyclic hypergraph. Instead, both the hypergraphs $\mathcal{H}_1$ and $\bar{\mathcal{H}}_1$ shown in Figure 3.(a) and Figure 3.(b), respectively, are not acyclic since their hypertree width is 2. A 2-width hypertree decomposition for $\bar{\mathcal{H}}_1$ is reported in Figure 3.(c).

In particular, observe that $\mathcal{H}_1$ has been obtained by adding the two hyperedges $h_4$ and $h_5$ to $\mathcal{H}_{\langle \mathcal{I}_0, \mathcal{B}_0 \rangle}$ to model, for instance, that two new bids, $B_4$ and $B_5$, respectively, have been proposed to the auctioneer. ◁

In the following, rather than working on the hypergraph $\mathcal{H}$ associated with a MaxWSP problem, we shall deal with its dual $\bar{\mathcal{H}}$, i.e., with the hypergraph such that its nodes are in one-to-one correspondence with the hyperedges of $\mathcal{H}$, and where for each node $x \in \mathcal{N}(\mathcal{H})$, $\{h \mid x \in h \wedge h \in \mathcal{E}(\mathcal{H})\}$ is in $\mathcal{E}(\bar{\mathcal{H}})$. As an example, the reader may want to check again the hypergraph $\mathcal{H}_1$ in Figure 3.(a) and notice that the hypergraph in Figure 3.(b) is in fact its dual.

The rationale for this choice is that issuing restrictions on the original hypergraph is a guarantee for the tractability only in very simple scenarios.

THEOREM 3. *On the class of acyclic hypergraphs,* MaxWSP *is (1) in P if each node occurs into two hyperedges at most; and, (2) NP-hard, even if each node is contained into three hyperedges at most.*

## 3.1 Hypertree Decomposition on the Dual Hypergraph and Tractable Packing Problems

For a fixed constant $k$, let $\mathcal{C}(\overline{\mathtt{hw}}, k)$ denote the class of all the hypergraphs whose dual hypergraphs have hypertree width bounded by $k$. The maximum weighted-set packing problem can be solved in polynomial time on the class $\mathcal{C}(\overline{\mathtt{hw}}, k)$ by means of the algorithm ComputeSetPacking$_k$, shown in Figure 4.

The algorithm receives in input a hypergraph $\mathcal{H}$, a weighting function $w$, and a $k$-width hypertree decomposition $HD = \langle T{=}(N, E), \chi, \lambda \rangle$ of $\bar{\mathcal{H}}$.

For each vertex $v \in N$, let $\mathcal{H}_v$ be the hypergraph whose set of nodes $\mathcal{N}(\mathcal{H}_v) \subseteq \mathcal{N}(\mathcal{H})$ coincides with $\lambda(v)$, and whose set of edges $\mathcal{E}(\mathcal{H}_v) \subseteq \mathcal{E}(\mathcal{H})$ coincides with $\chi(v)$. In an initialization step, the algorithm equips each vertex $v$ with all the possible packings for $\mathcal{H}_v$, which are stored in the set $H_v$. Note that the size of $H_v$ is bounded by $(|\mathcal{E}(\mathcal{H})| + 1)^k$, since each node in $\lambda(v)$ is either left uncovered in a packing or is covered with precisely one of the hyperedges in $\chi(v) \subseteq \mathcal{E}(\mathcal{H})$. Then, ComputeSetPacking$_k$ is designed to filter these packings by retaining only those that "conform" with some packing for $\mathcal{H}_c$, for each children $c$ of $v$ in $T$, as formalized next. Let $\mathbf{h}_v$ and $\mathbf{h}_c$ be two packings for $\mathcal{H}_v$ and $\mathcal{H}_c$, respectively. We say that $\mathbf{h}_v$ *conforms* with $\mathbf{h}_c$, denoted by $\mathbf{h}_v \approx \mathbf{h}_c$ if: for each $h \in \mathbf{h}_c \cap \mathcal{E}(\mathcal{H}_v)$, $h$ is in $\mathbf{h}_v$; and, for each $h \in (\mathcal{E}(\mathcal{H}_c) - \mathbf{h}_c)$, $h$ is not in $\mathbf{h}_v$.

EXAMPLE 4. Consider again the hypertree decomposition of $\bar{\mathcal{H}}_1$ reported in Figure 3.(c). Then, the set of all the possible packings (which are build in the initialization step of ComputeSetPacking$_k$), for each of its vertices, is re-

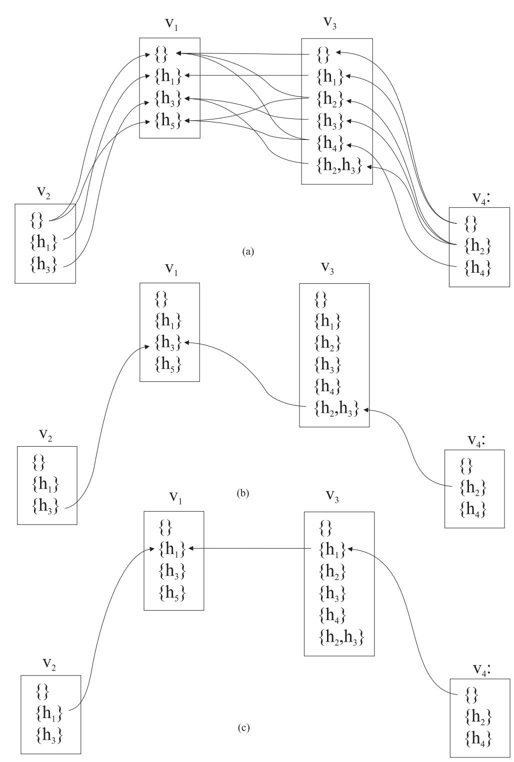

**Figure 5: Example application of Algorithm ComputeSetPacking$_k$.**

**Input**: $\mathcal{H}$, $w$, and a $k$-width hypertree decomposition $HD = \langle T{=}(N,E), \chi, \lambda\rangle$ of $\bar{\mathcal{H}}$;
**Output**: A solution to $\texttt{MaxWSP}(\mathcal{H}, w)$;
**var** $H_v$ : set of packings for $\mathcal{H}_v$, for each $v \in N$; $\mathbf{h}^*$ : packing for $\mathcal{H}$;
$\quad\ell^v_{\mathbf{h}_v}$ : rational number, for each partial packing $\mathbf{h}_v$ for $\mathcal{H}_v$;
$\quad\mathbf{h}_{\mathbf{h}_v,c}$ : partial packing for $\mathcal{H}_c$, for each partial packing $\mathbf{h}_v$ for $\mathcal{H}_v$, and for each $(v,c) \in E$;

---

**Procedure** *BottomUp*;
**begin**
$\quad$ *Done* := the set of all the leaves of $T$;
$\quad$ **while** $\exists v \in T$ such that (i) $v \notin$ *Done*, and (ii) $\{c \mid c$ is child of $v\} \subseteq$ *Done* **do**
$\quad\quad$ **for each** $c$ such that $(v,c) \in E$ **do**
$\quad\quad\quad$ $H_v := H_v - \{\mathbf{h}_v \mid \nexists \mathbf{h}_c \in H_c \text{ s.t. } \mathbf{h}_v \approx \mathbf{h}_c\}$;
$\quad\quad$ **for each** $\mathbf{h}_v \in H_v$ **do**
$\quad\quad\quad$ $\ell^v_{\mathbf{h}_v} := w(\mathbf{h}_v)$;
$\quad\quad\quad$ **for each** $c$ such that $(v,c) \in E$ **do**
$\quad\quad\quad\quad$ $\bar{\mathbf{h}}_c := \arg\max_{\mathbf{h}_c \in H_c \mid \mathbf{h}_v \approx \mathbf{h}_c} \left(\ell^c_{\mathbf{h}_c} - w(\mathbf{h}_c \cap \mathbf{h}_v)\right)$;
$\quad\quad\quad\quad$ $\mathbf{h}_{\mathbf{h}_v,c} := \bar{\mathbf{h}}_c$; `(* set best packing *)`
$\quad\quad\quad\quad$ $\ell^v_{\mathbf{h}_v} := \ell^v_{\mathbf{h}_v} + \ell^c_{\bar{\mathbf{h}}_c} - w(\bar{\mathbf{h}}_c \cap \mathbf{h}_v)$;
$\quad\quad\quad$ **end for**
$\quad\quad$ **end for**
$\quad\quad$ *Done* := *Done* $\cup \{v\}$;
$\quad$ **end while**
**end;**

---

**begin** (* MAIN *)
$\quad$ **for each** vertex $v$ in $T$ **do**
$\quad\quad$ $H_v := \{\mathbf{h}_v$ packing for $\mathcal{H}_v\}$;
$\quad$ *BottomUp*;
$\quad$ let $r$ be the root of $T$;
$\quad$ $\bar{\mathbf{h}}_r := \arg\max_{\mathbf{h}_r \in H_r} \ell^r_{\mathbf{h}_r}$;
$\quad$ $\mathbf{h}^* := \bar{\mathbf{h}}_r$; `(* include packing *)`
$\quad$ $TopDown(r, \mathbf{h}_r)$;
$\quad$ **return** $\mathbf{h}^*$;
**end**.

**Procedure** $TopDown(v$ : vertex of $N$, $\bar{\mathbf{h}}_v \in H_v)$;
**begin**
$\quad$ **for each** $c \in N$ s.t. $(v,c) \in E$ **do**
$\quad\quad$ $\bar{\mathbf{h}}_c := \mathbf{h}_{\bar{\mathbf{h}}_v,c}$;
$\quad\quad$ $\mathbf{h}^* := \mathbf{h}^* \cup \bar{\mathbf{h}}_c$; `(* include packing *)`
$\quad\quad$ $TopDown(c, \bar{\mathbf{h}}_c)$;
$\quad$ **end for**
**end;**

**Figure 4: Algorithm ComputeSetPacking$_k$.**

ported in Figure 5.(a). For instance, the root $v_1$ is such that $H_{v_1} = \{\ \{\},\ \{h_1\},\ \{h_3\},\ \{h_5\}\ \}$.

Moreover, an arrow from a packing $\mathbf{h}_c$ to $\mathbf{h}_v$ denotes that $\mathbf{h}_v$ conforms with $\mathbf{h}_c$. For instance, the reader may check that the packing $\{h_3\} \in H_{v_1}$ conforms with the packing $\{h_2, h_3\} \in H_{v_3}$, but do not conform with $\{h_1\} \in H_{v_3}$. $\lhd$

ComputeSetPacking$_k$ builds a solution by traversing $T$ in two phases. In the first phase, vertices of $T$ are processed from the leaves to the root $r$, by means of the procedure *BottomUp*. For each node $v$ being processed, the set $H_v$ is preliminary updated by removing all the packings $\mathbf{h}_v$ that do not conform with any packing for some of the children of $v$. After this filtering is performed, the weight $\ell_{\mathbf{h}_v}$ is updated. Intuitively, $\ell^v_{\mathbf{h}_v}$ stores the weight of the best partial packing for $\mathcal{H}$ computed by using only the hyperedges occurring in $\chi(T_v)$. Indeed, if $v$ is a leaf, then $\ell^v_{\mathbf{h}_v} = w(\mathbf{h}_v)$. Otherwise, for each child $c$ of $v$ in $T$, $\ell^v_{\mathbf{h}_v}$ is updated with the maximum of $\ell^c_{\mathbf{h}_c} - w(\mathbf{h}_c \cap \mathbf{h}_v)$ over all the packings $\mathbf{h}_c$ that conforms with $\mathbf{h}_v$ (resolving ties arbitrarily). The packing $\bar{\mathbf{h}}_c$ for which this maximum is achieved is stored in the variable $\mathbf{h}_{\mathbf{h}_v,c}$.

In the second phase, the tree $T$ is processed starting from the root. Firstly, the packing $\mathbf{h}^*$ is selected that maximizes the weight equipped with the packings in $H_r$. Then, procedure *TopDown* is used to extend $\mathbf{h}^*$ to all the other partial packings for vertices of $T$. In particular, at each vertex $v$, $\mathbf{h}^*$ is extended with the packing $\mathbf{h}_{\mathbf{h}_v,c}$, for each child $c$ of $v$.

Example 5. Assume that, in our running example, $w(h_1) = w(h_2) = w(h_3) = w(h_4) = 1$. Then, an execution of ComputeSetPacking$_k$ is graphically depicted in Figure 5.(b), where an arrow from a packing $\mathbf{h}_c$ to a packing $\mathbf{h}_v$ is used to denote that $\mathbf{h}_c = \mathbf{h}_{\mathbf{h}_v,c}$. Specifically, the choices made during the computation are such that the packing $\{h_2, h_3\}$ is computed.

In particular, during the bottom-up phase, we have that: (1) $v_4$ is processed, and we set $\ell^{v_4}_{\{h_2\}} = \ell^{v_4}_{\{h_4\}} = 1$ and $\ell^{v_4}_{\{\}} = 0$; (2) $v_3$ is processed, and we set $\ell^{v_3}_{\{h_1\}} = \ell^{v_3}_{\{h_3\}} = 1$ and $\ell^{v_3}_{\{\}} = 0$; (3) $v_2$ is processed, and we set $\ell^{v_2}_{\{h_1\}} = \ell^{v_2}_{\{h_2\}} = \ell^{v_2}_{\{h_3\}} = \ell^{v_2}_{\{h_4\}} = 1$, $\ell^{v_2}_{\{h_2,h_3\}} = 2$ and $\ell^{v_3}_{\{\}} = 0$; (4) $v_1$ is processed and we set $\ell^{v_1}_{\{h_1\}} = 1$, $\ell^{v_1}_{\{h_5\}} = \ell^{v_1}_{\{h_3\}} = 2$ and $\ell^{v_1}_{\{\}} = 0$. For instance, note that $\ell^{v_1}_{\{h_5\}} = 2$ since $\{h_5\}$ conforms with the packing $\{h_4\}$ of $H_{v_2}$ such that $\ell^{v_2}_{\{h_4\}} = 1$.

Then, at the beginning of the top-down phase, ComputeSetPacking$_k$ selects $\{h_3\}$ as a packing for $H_{v_1}$ and propagates this choice in the tree. Equivalently, the algorithm may have chosen $\{h_5\}$.

As a further example, the way the solution $\{h_1\}$ is obtained by the algorithm when $w(h_1) = 5$ and $w(h_2) = w(h_3) = w(h_4) = 1$ is reported in Figure 5.(c). Notice that, this time, in the top-down phase, ComputeSetPacking$_k$ starts selecting $\{h_1\}$ as the best packing for $H_{v_1}$. $\lhd$

Theorem 4. *Let $\mathcal{H}$ be a hypergraph and $w$ be a weighting function for it. Let $HD = \langle T, \chi, \lambda\rangle$ be a complete $k$-width hypertree decomposition of $\bar{\mathcal{H}}$. Then, ComputeSetPacking$_k$ on input $\mathcal{H}$, $w$, and $HD$ correctly outputs a solution for* $\texttt{MaxWSP}(\mathcal{H}, w)$ *in time $O(|T| \times (|\mathcal{E}(\mathcal{H})| + 1)^{2k})$.*

Proof. **[Sketch]** We observe that $\mathbf{h}^*$ (computed by ComputeSetPacking$_k$) is a packing for $\mathcal{H}$. Indeed, consider a pair of hyperedges $h_1$ and $h_2$ in $\mathbf{h}^*$, and assume, for the sake of contradiction, that $h_1 \cap h_2 \neq \emptyset$. Let $v_1$ (resp., $v_2$) be an arbitrary vertex of $T$, for which ComputeSetPacking$_k$ included $h_1$ (resp., $h_2$) in $\mathbf{h}^*$ in the bottom-down computation. By construction, we have $h_1 \in \chi(v_1)$ and $h_2 \in \chi(v_2)$.

Let $I$ be an element in $h_1 \cap h_2$. In the dual hypergraph $\mathcal{H}$, $I$ is a hyperedge in $\mathcal{E}(\bar{\mathcal{H}})$ which covers both the nodes $h_1$ and $h_2$. Hence, by condition (1) in Definition 1, there is a vertex $v \in vertices(T)$ such that $\{h_1, h_2\} \subseteq \chi(v)$. Note that, because of the connectedness condition in Definition 1, we can also assume, w.l.o.g., that $v$ is in the path connecting $v_1$ and $v_2$ in $T$.

Let $\mathbf{h}_v \in H_v$ denote the element added by ComputeSetPacking$_k$ into $\mathbf{h}^*$ during the bottom-down phase. Since the elements in $H_v$ are packings for $\mathcal{H}_v$, it is the case that either $h_1 \in \mathbf{h}_v$ or $h_2 \in \mathbf{h}_v$. Assume, w.l.o.g., that $h_1 \notin \mathbf{h}_v$, and notice that each vertex $w$ in $T$ in the path connecting $v$ to $v_1$ is such that $h_1 \in \chi(w)$, because of the connectedness condition. Hence, because of definition of conformance, the packing $\mathbf{h}_w$ selected by ComputeSetPacking$_k$ to be added at vertex $w$ in $\mathbf{h}^*$ must be such that $h_1 \notin \mathbf{h}_w$. This holds in particular for $w = v_1$. Contradiction with the definition of $v_1$.

Therefore, $\mathbf{h}^*$ is a packing for $\mathcal{H}$. It remains then to show that it has the maximum weight over all the packings for $\mathcal{H}$. To this aim, we can use structural induction on $T$ to prove that, in the bottom-up phase, the variable $\ell^v_{\mathbf{h}_v}$ is updated to contain the weight of the packing on the edges in $\chi(T_v)$, which contains $\mathbf{h}_v$ and which has the maximum weight over all such packings for the edges in $\chi(T_v)$. Then, the result follows, since in the top-down phase, the packing $\mathbf{h}_r$ giving the maximum weight over $\chi(T_r) = \mathcal{E}(\mathcal{H})$ is first included in $\mathbf{h}^*$, and then extended at each node $c$ with the packing $\mathbf{h}_{\mathbf{h}_v,c}$ conformingly with $\mathbf{h}_v$ and such that the maximum value of $\ell^v_{\mathbf{h}_v}$ is achieved.

As for the complexity, observe that the initialization step requires the construction of the set $H_v$, for each vertex $v$, and each set has size $(|\mathcal{E}(\mathcal{H})| + 1)^k$ at most. Then, the function *BottomUp* checks for the conformance between strategies in $H_v$ with strategies in $H_c$, for each pair $(v, c) \in E$, and updates the weight $\ell^v_{\mathbf{h}_v}$. These tasks can be carried out in time $O((|\mathcal{E}(\mathcal{H})| + 1)^{2k})$ and must be repeated for each edge in $T$, i.e., $O(|T|)$ times. Finally, the function $TopDown$ can be implemented in linear time in the size of $T$, since it just requires updating $\mathbf{h}^*$ by accessing the variable $\mathbf{h}_{\mathbf{h}_v,c}$. □

The above result shows that if a hypertree decomposition of width $k$ is given, the `MaxWSP` problem can be efficiently solved. Moreover, differently from the case of structured item graphs, it is well known that deciding the existence of a $k$-bounded hypertree decomposition and computing one (if any) are problems which can be efficiently solved in polynomial time [7]. Therefore, Theorem 4 witnesses that the class $\mathcal{C}(\overline{\texttt{hw}}, k)$ actually constitutes a tractable class for the winner determination problem.

As the following theorem shows, for large subclasses (that depend only on how the weight function is specified), $\texttt{MaxWSP}(\mathcal{H}, w)$ is even highly parallelizeable. Let us call a weighting function *smooth* if it is logspace computable and if all weights are polynomial (and thus just require $O(\log n)$ bits for their representation). Recall that LOGCFL is a parallel complexity class contained in $NC_2$, cf. [9]. The functional version of LOGCFL is $\mathrm{L}^{\mathrm{LOGCFL}}$, which is obtained by equipping a logspace transducer with an oracle in LOGCFL.

Theorem 5. *Let $\mathcal{H}$ be a hypergraph in $\mathcal{C}(\overline{\texttt{hw}}, k)$, and let $w$ be a smooth weighting function for it. Then, $\texttt{MaxWSP}(\mathcal{H}, w)$ is in $\mathrm{L}^{\mathrm{LOGCFL}}$.*

# 4. HYPERTREE DECOMPOSITIONS VS STRUCTURED ITEM GRAPHS

Given that the class $\mathcal{C}(\overline{\texttt{hw}}, k)$ has been shown to be an island of tractability for the winner determination problem, and given that the class $\mathcal{C}(\texttt{ig}, k)$ has been shown not to be efficiently recognizable, one may be inclined to think that there are instances having unbounded hypertree width, but admitting an item graph of bounded tree width (so that the intractability of structured item graphs would lie in their generality).

Surprisingly, we establish this is not the case. The line of the proof is to first show that structured item graphs are in one-to-one correspondence with a special kind of hypertree decompositions of the dual hypergraph, which we shall call *strict*. Then, the result will follow by proving that $k$-width strict hypertree decompositions are less powerful than $k$-with hypertree decompositions.

## 4.1 Strict Hypertree Decompositions

Let $\mathcal{H}$ be a hypergraph, and let $V \subseteq \mathcal{N}(\mathcal{H})$ be a set of nodes and $X, Y \in \mathcal{N}(\mathcal{H})$. $X$ is $[V]$-adjacent to $Y$ if there exists an edge $h \in \mathcal{E}(\mathcal{H})$ such that $\{X, Y\} \subseteq (h - V)$. A $[V]$-path $\pi$ from $X$ to $Y$ is a sequence $X = X_0, \ldots, X_\ell = Y$ of variables such that: $X_i$ is $[V]$-adjacent to $X_{i+1}$, for each $i \in [0...\ell$-1]. A set $W \subseteq \mathcal{N}(\mathcal{H})$ of nodes is $[V]$-connected if $\forall X, Y \in W$ there is a $[V]$-path from $X$ to $Y$. A $[V]$-component is a maximal $[V]$-connected non-empty set of nodes $W \subseteq (\mathcal{N}(\mathcal{H}) - V)$. For any $[V]$-component $C$, let $\mathcal{E}(C) = \{h \in \mathcal{E}(\mathcal{H}) \mid h \cap C \neq \emptyset\}$.

Definition 2. A hypertree decomposition $HD = \langle T, \chi, \lambda \rangle$ of $\mathcal{H}$ is *strict* if the following conditions hold:

1. for each pair of vertices $r$ and $s$ in $vertices(T)$ such that $s$ is a child of $r$, and for each $[\chi(r)]$*-component* $C_r$ s.t. $C_r \cap \chi(T_s) \neq \emptyset$, $C_r$ is a $[\chi(r) \cap \mathcal{N}(\lambda(r) \cap \lambda(s))]$*-component*;
2. for each edge $h \in \mathcal{E}(\mathcal{H})$, there is a vertex $p$ such that $h \in \lambda(p)$ and $h \subseteq \chi(p)$ (we say $p$ strongly covers $h$);
3. for each edge $h \in \mathcal{E}(\mathcal{H})$, the set $\{p \in vertices(T) \mid h \in \lambda(p)\}$ induces a (connected) subtree of $T$.

The *strict hypertree width* $shw(\mathcal{H})$ of $\mathcal{H}$ is the minimum width over all its strict hypertree decompositions. □

The basic relationship between nice hypertree decompositions and structured item graphs is shown in the following theorem.

Theorem 6. *Let $\mathcal{H}$ be a hypergraph such that for each node $v \in \mathcal{N}(\mathcal{H})$, $\{v\}$ is in $\mathcal{E}(\mathcal{H})$. Then, a $k$-width tree decomposition of an item graph for $\mathcal{H}$ exists if and only if $\bar{\mathcal{H}}$ has a $(k+1)$-width strict hypertree decomposition*[2].

Note that, as far as the maximum weighted-set packing problem is concerned, given a hypergraph $\mathcal{H}$, we can always assume that for each node $v \in \mathcal{N}(\mathcal{H})$, $\{v\}$ is in $\mathcal{E}(\mathcal{H})$. In fact, if this hyperedge is not in the hypergraph, then it can be added without loss of generality, by setting $w(\{v\}) = 0$. Therefore, letting $\mathcal{C}(\overline{\texttt{shw}}, k)$ denote the class of all the hypergraphs whose dual hypergraphs (associated with maximum

[2]The term "+1" only plays the technical role of taking care of the different definition of width for tree decompositions and hypertree decompositions.

weighted-set packing problems) have strict hypertree width bounded by $k$, we have that $\mathcal{C}(\overline{\mathtt{shw}}, k+1) = \mathcal{C}(\mathtt{ig}, k)$.

By definition, strict hypertree decompositions are special hypertree decompositions. In fact, we are able to show that the additional conditions in Definition 2 induce an actual restriction on the decomposition power.

THEOREM 7. $\mathcal{C}(\mathtt{ig}, k) = \mathcal{C}(\overline{\mathtt{shw}}, k+1) \subset \mathcal{C}(\overline{\mathtt{hw}}, k+1)$.

**A Game Theoretic View.** We shed further lights on strict hypertree decompositions by discussing an interesting characterization based on the *strict Robber and Marshals Game*, defined by adapting the Robber and Marshals game defined in [6], which characterizes hypertree width.

The game is played on a hypergraph $\mathcal{H}$ by a robber against $k$ marshals which act in coordination. Marshals move on the hyperedges of $\mathcal{H}$, while the robber moves on nodes of $\mathcal{H}$. The robber sees where the marshals intend to move, and reacts by moving to another node which is connected with its current position and through a path in $G(\mathcal{H})$ which does not use any node contained in a hyperedge that is occupied by the marshals before and after their move—we say that these hyperedges are blocked. Note that in the basic game defined in [6], the robber is not allowed to move on vertices that are occupied by the marshals before and after their move, even if they do not belong to blocked hyperedges.

Importantly, marshals are required to play monotonically, i.e., they cannot occupy an edge that was previously occupied in the game, and which is currently not. The marshals win the game if they capture the robber, by occupying an edge covering a node where the robber is. Otherwise, the robber wins.

THEOREM 8. *Let $\mathcal{H}$ be a hypergraph such that for each node $v \in \mathcal{N}(\mathcal{H})$, $\{v\}$ is in $\mathcal{E}(\mathcal{H})$. Then, $\bar{\mathcal{H}}$ has a k-width strict hypertree decomposition if and only if k marshals can win the strict Robber and Marshals Game on $\bar{\mathcal{H}}$, no matter of the robber's moves.*

# 5. CONCLUSIONS

We have solved the open question of determining the complexity of computing a structured item graph associated with a combinatorial auction scenario. The result is bad news, since it turned out that it is NP-complete to check whether a combinatorial auction has a structured item graph, even for treewidth 3. Motivated by this result, we investigated the use of hypertree decomposition (on the dual hypergraph associated with the scenario) and we shown that the problem is tractable on the class of those instances whose dual hypergraphs have bounded hypertree width. For some special, yet relevant cases, a highly parallelizable algorithm is also discussed. Interestingly, it also emerged that the class of structured item graphs is properly contained in the class of instances having bounded hypertree width (hence, the reason of their intractability is not their generality).

In particular, the latter result is established by showing a precise relationship between structured item graphs and restricted forms of hypertree decompositions (on the dual hypergraph), called *query decompositions* (see, e.g., [7]). In the light of this observation, we note that proving some approximability results for structured item graphs requires a deep understanding of the approximability of query decompositions, which is currently missing in the literature.

As a further avenue of research, it would be relevant to enhance the algorithm ComputeSetPacking$_k$, e.g., by using specialized data structures, in order to avoid the quadratic dependency from $(|\mathcal{E}(\mathcal{H})| + 1)^k$.

Finally, an other interesting question is to assess whether the structural decomposition techniques discussed in the paper can be used to efficiently deal with generalizations of the winner determination problem. For instance, it might be relevant in several application scenarios to design algorithms that can find a selling strategy when several copies of the same item are available for selling, and when moreover the auctioneer is satisfied when at least a given number of copies is actually sold.

## Acknowledgement

G. Gottlob's work was supported by the EC3 - *E-Commerce Competence Center* (Vienna) and by a Royal Society Wolfson Research Merit Award. In particular, this Award allowed Gottlob to invite G. Greco for a research visit to Oxford. In addition, G. Greco is supported by ICAR-CNR, and by M.I.U.R. under project TOCAI.IT.

# 6. REFERENCES

[1] I. Adler, G. Gottlob, and M. Grohe. Hypertree-Width and Related Hypergraph Invariants. In *Proc. of EUROCOMB'05*, pages 5–10, 2005.
[2] C. Boutilier. Solving Concisely Expressed Combinatorial Auction Problems. In *Proc. of AAAI'02*, pages 359–366, 2002.
[3] V. Conitzer, J. Derryberry, and T. Sandholm. Combinatorial auctions with structured item graphs. In *Proc. of AAAI'04*, pages 212–218, 2004.
[4] E. M. Eschen and J. P. Sinrad. An $o(n^2)$ algorithm for circular-arc graph recognition. In *Proc. of SODA'93*, pages 128–137, 1993.
[5] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate. In *Proc. of IJCAI'99*, pages 548–553, 1999.
[6] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences*, 66(4):775–808, 2003.
[7] G. Gottlob, N. Leone, and S. Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 63(3):579–627, 2002.
[8] H. H. Hoos and C. Boutilier. Solving combinatorial auctions using stochastic local search. In *Proc. of AAAI'00*, pages 22–29, 2000.
[9] D. Johnson. A Catalog of Complexity Classes. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 67–161. 1990.
[10] N. Korte and R. H. Mohring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM Journal on Computing*, 18(1):68–81, 1989.
[11] D. Lehmann, R. Müller, and T. Sandholm. The Winner Determination Problem. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. MIT Press, 2006.
[12] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient

combinatorial auctions. *J. ACM*, 49(5):577–602, 2002.

[13] R. McAfee and J. McMillan. Analyzing the airwaves auction. *Journal of Economic Perspectives*, 10(1):159175, 1996.

[14] J. McMillan. Selling spectrum rights. *Journal of Economic Perspectives*, 8(3):145–62, 1994.

[15] N. Nisan. Bidding and allocation in combinatorial auctions. In *Proc. of EC'00*, pages 1–12, 2000.

[16] N. Robertson and P. Seymour. Graph minors ii. algorithmic aspects of tree width. *Journal of Algorithms*, 7:309–322, 1986.

[17] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44:1131–1147, 1998.

[18] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proc. of AAAI'93*, pages 256–262, 1993.

[19] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[20] T. Sandholm. Winner determination algorithms. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. MIT Press, 2006.

[21] T. Sandholm and S. Suri. Bob: Improved winner determination in combinatorial auctions and generalizations. *Artificial Intelligence*, 7:33–58, 2003.

[22] M. Tennenholtz. Some tractable combinatorial auctions. In *Proc. of AAAI'00*, pages 98–103, 2000.

[23] E. Zurel and N. Nisan. An efficient approximate allocation algorithm for combinatorial auctions. In *Proc. of EC'01*, pages 125–136, 2001.