Winner Determination for Bidding Languages using Weighted Formulas

Joel Uckelman Ulle Endriss

Institute for Logic, Language, and Computation
University of Amsterdam
{juckelma,ulle}@illc.uva.nl

3rd MARA Get-Together
Amsterdam, 6 June 2008

Overview

Background	Combinatorial Auctions &c.
Weighted Formulas	Describe *weighted formulas* and *goal bases*.
The WDP	The Winner Determination Probelm for CAs
Branch & Bound	A method for solving the WDP using heuristics.
B&B for OR	Ways to do B&B for the OR language.
WF Languages	Bidding languages using weighted formulas.
B&B for WF	How B&B for WF differs from B&B for OR.
Design Parameters	What knobs B&B offers the designer, and how we turned them.
Experimental Results	How our heurstics performed.
Integer Programming	An Integer Programming formulation of the WDP for weighted formulas.

Combinatorial Auctions

Combinatorial auctions:

- Simultaneous, not sequential. All items auctioned at once.
- Bids can be for subsets of items, not just single items.

Bidding for subsets explicitly is inefficient. We need *bidding languages*.

OR Language

Bids are sets of items. Multiple nonintersecting atomic bids may be accepted.

E.g.:

$$(\{a, b\}, 1) \text{ OR } (\{b, c\}, 2) \text{ OR } (\{c, d\}, 3)$$

This bidder will pay 4 for bundle $\{a, b, c, d\}$ but only 2 for $\{a, b, c\}$.

Weighted Formulas and Goal Bases

Definitions

- A *weighted formula* is a pair (φ, w), where φ is a propositional formula and $w \in \mathbb{R}$.
- A *goal base* is a set of weighted satisfiable formulas.

Examples

Goal bases:

$$\emptyset \qquad \{(p, 42)\} \qquad \{(\top, -2)\} \qquad \{(a, 1), (a \wedge a, 1)\}$$

$$\{(a \wedge b, -5), (\neg a \vee d, 13)\}$$

Not a goal base:

$$\{(\bot, 3)\}$$

Weighted Formulas and Goal Bases

Definitions

- A *weighted formula* is a pair (φ, w), where φ is a propositional formula and $w \in \mathbb{R}$.
- A *goal base* is a set of weighted satisfiable formulas.

Examples

Goal bases:

$$\emptyset \qquad \{(p, 42)\} \qquad \{(\top, -2)\} \qquad \{(a, 1), (a \wedge a, 1)\}$$

$$\{(a \wedge b, -5), (\neg a \vee d, 13)\}$$

Not a goal base:

$$\{(\cancel{\bot, 3})\}$$

Goal Bases and Utility Functions

Definitions

- $\mathcal{PS}$ is a finite set of propositional variables.
- A *utility function* is a mapping $u : 2^{\mathcal{PS}} \to \mathbb{R}$.
- A *model* is a set $M \subseteq \mathcal{PS}$ (i.e., just the true atoms).
- Every goal base G generates a unique utility function u_G:

$$u_G(M) = \sum \{w : (\varphi, w) \in G \text{ and } M \models \varphi\}$$

Goal Bases Form Bidding Languages

A goal base language is a class of goal bases meeting given conditions. E.g.:

$\mathcal{L}($	*positive cubes*,	*positive*$)$
language of	conjunctions of atoms	with positive weights

Restrictions on formulas correspond to properties of utility functions. For more on this, and issues of succinctness and computational complexity of queries see (Chevalrye, Endriss, & Lang, KR-2006), (U&E AIPref-2007), (U&E KR-2008).

A goal base can be submitted as an agent's bid in a combinatorial auction, so a goal base language can serve as a bidding language.

Goal Bases Form Bidding Languages

A goal base language is a class of goal bases meeting given conditions. E.g.:

$\mathcal{L}($	*positive cubes*,	*positive*$)$
language of	conjunctions of atoms	with positive weights

Restrictions on formulas correspond to properties of utility functions. For more on this, and issues of succinctness and computational complexity of queries see (Chevalrye, Endriss, & Lang, KR-2006), (U&E AIPref-2007), (U&E KR-2008).

A goal base can be submitted as an agent's bid in a combinatorial auction, so a goal base language can serve as a bidding language.

Goal Bases Form Bidding Languages

A goal base language is a class of goal bases meeting given conditions. E.g.:

$\mathcal{L}($	*positive cubes*,	*positive*$)$
language of	conjunctions of atoms	with positive weights

Restrictions on formulas correspond to properties of utility functions. For more on this, and issues of succinctness and computational complexity of queries see (Chevalrye, Endriss, & Lang, KR-2006), (U&E AIPref-2007), (U&E KR-2008).

A goal base can be submitted as an agent's bid in a combinatorial auction, so a goal base language can serve as a bidding language.

Goal Bases Form Bidding Languages

A goal base language is a class of goal bases meeting given conditions. E.g.:

$\mathcal{L}($	*positive cubes*,	*positive*$)$
language of	conjunctions of atoms	with positive weights

Restrictions on formulas correspond to properties of utility functions. For more on this, and issues of succinctness and computational complexity of queries see (Chevalrye, Endriss, & Lang, KR-2006), (U&E AIPref-2007), (U&E KR-2008).

A goal base can be submitted as an agent's bid in a combinatorial auction, so a goal base language can serve as a bidding language.

The Winner Determination Problem

The WDP: finding an optimal allocation of items in a combinatorial auction.

Complexity of the WDP depends on the biding language/utility functions epxressible in it. NP-hard in most cases.

We need heuristic methods for finding exact solutions rapidly.

The Branch & Bound Algorithm

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- We're building a tree form the solution space.
- S_i is dominated if S_i's upper bound $<$ some S_j's lower bound
- $\emptyset \subset S_{c_j} \subset S_i$ ensures termination; covering so we don't omit solutions
- Result is all optimal solutions.

The Branch & Bound Algorithm

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- We're building a tree form the solution space.
- S_i is dominated if S_i's upper bound $<$ some S_j's lower bound
- $\emptyset \subset S_{c_j} \subset S_i$ ensures termination; covering so we don't omit solutions
- Result is all optimal solutions.

The Branch & Bound Algorithm

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- We're building a tree form the solution space.
- S_i is dominated if S_i's upper bound $<$ some S_j's lower bound
- $\emptyset \subset S_{c_j} \subset S_i$ ensures termination; covering so we don't omit solutions
- Result is all optimal solutions.

The Branch & Bound Algorithm

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- We're building a tree form the solution space.
- S_i is dominated if S_i's upper bound $<$ some S_j's lower bound
- $\emptyset \subset S_{c_j} \subset S_i$ ensures termination; covering so we don't omit solutions
- Result is all optimal solutions.

The Branch & Bound Algorithm

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- We're building a tree form the solution space.
- S_i is dominated if S_i's upper bound $<$ some S_j's lower bound
- $\emptyset \subset S_{c_j} \subset S_i$ ensures termination; covering so we don't omit solutions
- Result is all optimal solutions.

The Branch & Bound Algorithm

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} do
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- We're building a tree form the solution space.
- S_i is dominated if S_i's upper bound $<$ some S_j's lower bound
- $\emptyset \subset S_{c_j} \subset S_i$ ensures termination; covering so we don't omit solutions
- Result is all optimal solutions.

The Branch & Bound Algorithm

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- We're building a tree form the solution space.
- S_i is dominated if S_i's upper bound $<$ some S_j's lower bound
- $\emptyset \subset S_{c_j} \subset S_i$ ensures termination; covering so we don't omit solutions
- Result is all optimal solutions.

Design Choices for B&B

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- How to specify subsets of the solution space?
- How to choose which nondominated leaf to expand next?
- How to build the children of the node to expand?
- How to calculate the bounds?

Design Choices for B&B

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \dots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- How to specify subsets of the solution space?
- How to choose which nondominated leaf to expand next?
- How to build the children of the node to expand?
- How to calculate the bounds?

Design Choices for B&B

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- How to specify subsets of the solution space?
- How to choose which nondominated leaf to expand next?
- How to build the children of the node to expand?
- How to calculate the bounds?

Design Choices for B&B

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \dots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- How to specify subsets of the solution space?
- How to choose which nondominated leaf to expand next?
- How to build the children of the node to expand?
- How to calculate the bounds?

Design Choices for B&B

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- How to specify subsets of the solution space?
- How to choose which nondominated leaf to expand next?
- How to build the children of the node to expand?
- How to calculate the bounds?

Design Choices for B&B

let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \ldots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- How to specify subsets of the solution space?
- How to choose which nondominated leaf to expand next?
- How to build the children of the node to expand?
- How to calculate the bounds?

Design Choices for B&B

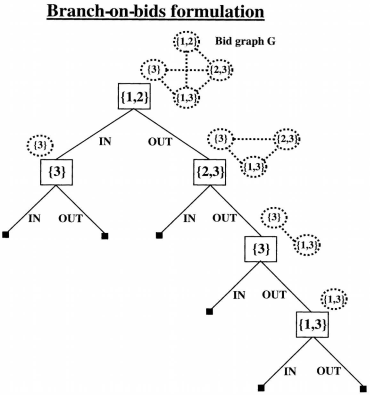
let S_0 be the whole solution space
while no weakly dominant leaf S_i is a singleton **do**
  choose such a leaf S_i // i.e., no leaf S_j s.t. $h(S_i) < g(S_j)$
  build children $S_{c_1}, \dots, S_{c_n}$ s.t. the S_{c_j} cover S_i and each $\emptyset \subset S_{c_j} \subset S_i$
  for all S_{c_j} **do**
    calculate $g(S_{c_j})$ // lower bound on quality of solutions in S_{c_j}
    calculate $h(S_{c_j})$ // upper bound on quality of solutions in S_{c_j}
  end for
end while
the first weakly dominant singleton S_i is optimal

- How to specify subsets of the solution space?
- How to choose which nondominated leaf to expand next?
- How to build the children of the node to expand?
- How to calculate the bounds?

Branch & Bound for the OR Language

(Sandholm 2006) suggests branch-on-bids for OR:

- Branch by accepting or rejecting a bid.
- Keep a conflict graph of bids to knock out unacceptable bids.
- Question: Should we focus on high-conflict bids first?



Many, many, many options are given for deciding what to branch on, how to order bids, whether to search depth-first, etc.

How to set the branch and bound parameters for weighted formulas?

What Should Our Nodes and Branches Be?

Branch-on-bids seems good for OR (see discussion in CA book)

B-on-B is more complex for formulas: φ as a node means checking $\varphi \models \psi$ (in the accpting branch) and $\psi \models \varphi$ (in the rejecting branch).

For us: nodes are partial allocations, branch on single-good extensions.

B&B tree is wider and shallower this way: a depth-$|\mathcal{PS}|$ $|\mathcal{A}|$-ary tree, instead of a depth-$\sum_{i \in \mathcal{A}} |G_i|$ binary tree.

Upper and Lower Bounds: How?

Tighter bounds = more pruning, but more time calculating the bounds.

Lower Bound

- An easy, fast lower bound: Attained value of the partial allocation.
- Hypothetical allocation of all remaining items to a designated agent.
- Random allocation of all remaining items to a designated agent.
- ...

Upper Bound

- Sum all (positive) weights of remaining unsatisfied formulas? Not very tight.
- Calculate "optimistic" values for each agent.
- Solve relaxations (allocate remaining items among $2, 3, \ldots, n$ agents.
- Do something else to find conflicts?
- ...

Branching Policy: What item to allocate next?

Definition

A *branching policy* b maps partial allocations to goods unallocated in them.

- The *lexical* branching policy: $b(A) = p$, where p is the lexically least good not allocated by partial allocation A.
- The *best-estimate first* branching policy: $b(A) = p$, where p is the lexically least good such that $h^p(A) = \max_{a \in \mathcal{PS}} h^a(A)$.

Notes

- With random test data, lexical = fixing a random order on goods before starting B&B.
- Lexical means that at every depth-n node was reached by allocating the same items (e.g., a, followed by b, followed by c). Best-estimate first may allocate items in a different order along different branches.
- Best-estimate first *should* result in fewer nodes built in general.

Expansion Policy: What node to expand next?

Definition

An *expansion policy* e choooses a partial allocation from a set of undominated partial allocatoins.

- The *best-upper-bound first* expansion policy:

$$e(\{A_1, \ldots, A_n\}) = \operatorname*{argmax}_{A_i} h(A_i)$$

- Some other expansion policy?

Notes

Why should we ever choose to expand a node which our upper-bound heurstic tells us has less potential than some other node?

Two Awful Upper Bound Hueristics

A lot of freedom here: Two extremes.

$h(A) = \infty$ is:

- a correct heuristic: Never an underestimate.
- a loose heuristic: No pruning will ever occur.
- a fast heuristic: It's a constant function!

$h(A) = \infty$ is worse *than a brute force search*:

$\rightsquigarrow$ only 2^n leaves, but $2^{n+1} - 1$ nodes!

$h(A) =$ "the true value of allocation A" is

- a correct heuristic: Never an underestimate.
- a tight heuristic: It's the WDP!
- a slow heuristic: It's the WDP!

$\boxed{\text{A good upper bound heuristic must balance tightness with speed.}}$

Some Notation...

- $\mathcal{A}$ is the set of agents
- A is a partial allocation of items to agents
- M_i^A is the model induced by allocation A for agent i, i.e.,

$$M_i^A = \{a \in \mathcal{PS} : A(a) = i\}$$

- $M_i^A \models \varphi$ iff allocation A gives agent i enough items to make φ true.
- $M_i^A \mathrel{?} \varphi$ iff allocation A does not yet determine φ.
- $\text{und}(A, \varphi)$ is the set of atoms not yet allocated by A in the formula φ.

Some Notation...

- $\mathcal{A}$ is the set of agents
- A is a partial allocation of items to agents
- M_i^A is the model induced by allocation A for agent i, i.e.,

$$M_i^A = \{a \in \mathcal{PS} : A(a) = i\}$$

- $M_i^A \models \varphi$ iff allocation A gives agent i enough items to make φ true.
- $M_i^A \mathrel{?} \varphi$ iff allocation A does not yet determine φ.
- $\mathrm{und}(A, \varphi)$ is the set of atoms not yet allocated by A in the formula φ.

Some Notation...

- $\mathcal{A}$ is the set of agents
- A is a partial allocation of items to agents
- M_i^A is the model induced by allocation A for agent i, i.e.,

$$M_i^A = \{a \in \mathcal{PS} : A(a) = i\}$$

- $M_i^A \models \varphi$ iff allocation A gives agent i enough items to make φ true.
- $M_i^A \,?\, \varphi$ iff allocation A does not yet determine φ.
- $\text{und}(A, \varphi)$ is the set of atoms not yet allocated by A in the formula φ.

Some Notation...

- $\mathcal{A}$ is the set of agents
- A is a partial allocation of items to agents
- M_i^A is the model induced by allocation A for agent i, i.e.,

$$M_i^A = \{a \in \mathcal{PS} : A(a) = i\}$$

- $M_i^A \models \varphi$ iff allocation A gives agent i enough items to make φ true.
- $M_i^A \, ? \, \varphi$ iff allocation A does not yet determine φ.
- $\mathrm{und}(A, \varphi)$ is the set of atoms not yet allocated by A in the formula φ.

Some Notation...

- $\mathcal{A}$ is the set of agents
- A is a partial allocation of items to agents
- M_i^A is the model induced by allocation A for agent i, i.e.,

$$M_i^A = \{a \in \mathcal{PS} : A(a) = i\}$$

- $M_i^A \models \varphi$ iff allocation A gives agent i enough items to make φ true.
- $M_i^A \, ? \, \varphi$ iff allocation A does not yet determine φ.
- $\text{und}(A, \varphi)$ is the set of atoms not yet allocated by A in the formula φ.

Some Notation...

- $\mathcal{A}$ is the set of agents
- A is a partial allocation of items to agents
- M_i^A is the model induced by allocation A for agent i, i.e.,

$$M_i^A = \{a \in \mathcal{PS} : A(a) = i\}$$

- $M_i^A \models \varphi$ iff allocation A gives agent i enough items to make φ true.
- $M_i^A \,?\, \varphi$ iff allocation A does not yet determine φ.
- $\mathrm{und}(A, \varphi)$ is the set of atoms not yet allocated by A in the formula φ.

Some Notation...

- $\mathcal{A}$ is the set of agents
- A is a partial allocation of items to agents
- M_i^A is the model induced by allocation A for agent i, i.e.,

$$M_i^A = \{a \in \mathcal{PS} : A(a) = i\}$$

- $M_i^A \models \varphi$ iff allocation A gives agent i enough items to make φ true.
- $M_i^A\, ?\, \varphi$ iff allocation A does not yet determine φ.
- $\mathrm{und}(A, \varphi)$ is the set of atoms not yet allocated by A in the formula φ.

A Better Upper Bound Heuristic for Positive Cubes, I

For $\mathcal{L}(\textit{positive cubes}, \textit{positive})$, this heuristic works:

$$h(A) = \sum_{p \in \mathcal{PS}} h^p(A)$$

$$h^p(A) = \max_{i \in \mathcal{A}} h_i^p(A) \qquad h_i^p(A) = \sum_{(\varphi, w) \in G_i} h_i^p(A, \varphi)$$

$$h_i^p(A, \varphi) = \begin{cases} \frac{w}{|\mathrm{und}(A,\varphi)|} & \text{if } (\varphi, w) \in G_i, p \in \mathrm{und}(A, \varphi), M_i^A \,?\, \varphi \\ 0 & \text{otherwise} \end{cases}$$

Optimistic value: The share of overall value an item has for an agent, assuming that the agent is allocated all remaining items.

> Intuition: Calculate the optimistic value of each item for each agent, and "award" the items so as to maximize the overall optimistic value.

A Better Upper Bound Heuristic for Positive Cubes, II

For $\mathcal{L}(\textit{positive cubes}, \textit{positive})$, this heuristic works:

$$h(A) = \sum_{p \in \mathcal{PS}} h^p(A)$$

$$h^p(A) = \max_{i \in \mathcal{A}} h_i^p(A) \qquad h_i^p(A) = \sum_{(\varphi, w) \in G_i} h_i^p(A, \varphi)$$

$$h_i^p(A, \varphi) = \begin{cases} \frac{w}{|\mathrm{und}(A,\varphi)|} & \text{if } (\varphi, w) \in G_i, p \in \mathrm{und}(A, \varphi), M_i^A \mathrel{?} \varphi \\ 0 & \text{otherwise} \end{cases}$$

- p is an atom, i is an agent, A is a partial allocation
- M_i^A is the model induced for agent i by allocation A
- $h_i^p(A, \varphi)$ is the optimistic value of p for agent i just from formula φ
- $h_i^p(A)$ is the optimistic value of p for agent i over all formulas
- $h^p(A)$ is the optimistic value of p to the agent who optimistically values it most
- $h(A)$ is the optimistic value of all unallocated atoms

A Better Upper Bound Heuristic for Positive Cubes, II

For $\mathcal{L}(\textit{positive cubes}, \textit{positive})$, this heuristic works:

$$h(A) = \sum_{p \in \mathcal{PS}} h^p(A)$$

$$h^p(A) = \max_{i \in \mathcal{A}} h_i^p(A) \qquad h_i^p(A) = \sum_{(\varphi, w) \in G_i} h_i^p(A, \varphi)$$

$$h_i^p(A, \varphi) = \begin{cases} \frac{w}{|\text{und}(A,\varphi)|} & \text{if } (\varphi, w) \in G_i, p \in \text{und}(A, \varphi), M_i^A \,?\, \varphi \\ 0 & \text{otherwise} \end{cases}$$

- p is an atom, i is an agent, A is a partial allocation
- M_i^A is the model induced for agent i by allocation A
- $h_i^p(A, \varphi)$ is the optimistic value of p for agent i just from formula φ
- $h_i^p(A)$ is the optimistic value of p for agent i over all formulas
- $h^p(A)$ is the optimistic value of p to the agent who optimistically values it most
- $h(A)$ is the optimistic value of all unallocated atoms

A Better Upper Bound Heuristic for Positive Cubes, II

For $\mathcal{L}(\textit{positive cubes}, \textit{positive})$, this heuristic works:

$$h(A) = \sum_{p \in \mathcal{PS}} h^p(A)$$

$$h^p(A) = \max_{i \in \mathcal{A}} h^p_i(A) \qquad h^p_i(A) = \sum_{(\varphi, w) \in G_i} h^p_i(A, \varphi)$$

$$h^p_i(A, \varphi) = \begin{cases} \frac{w}{|\mathrm{und}(A,\varphi)|} & \text{if } (\varphi, w) \in G_i, p \in \mathrm{und}(A, \varphi), M^A_i \,?\, \varphi \\ 0 & \text{otherwise} \end{cases}$$

- p is an atom, i is an agent, A is a partial allocation
- M^A_i is the model induced for agent i by allocation A
- $h^p_i(A, \varphi)$ is the optimistic value of p for agent i just from formula φ
- $h^p_i(A)$ is the optimistic value of p for agent i over all formulas
- $h^p(A)$ is the optimistic value of p to the agent who optimistically values it most
- $h(A)$ is the optimistic value of all unallocated atoms

A Better Upper Bound Heuristic for Positive Cubes, II

For $\mathcal{L}(\mathit{positive\ cubes}, \mathit{positive})$, this heuristic works:

$$h(A) = \sum_{p \in \mathcal{PS}} h^p(A)$$

$$h^p(A) = \max_{i \in \mathcal{A}} h^p_i(A) \qquad h^p_i(A) = \sum_{(\varphi, w) \in G_i} h^p_i(A, \varphi)$$

$$h^p_i(A, \varphi) = \begin{cases} \frac{w}{|\mathrm{und}(A,\varphi)|} & \text{if } (\varphi, w) \in G_i, p \in \mathrm{und}(A, \varphi), M_i^A \text{ ? } \varphi \\ 0 & \text{otherwise} \end{cases}$$

- p is an atom, i is an agent, A is a partial allocation
- M_i^A is the model induced for agent i by allocation A
- $h^p_i(A, \varphi)$ is the optimistic value of p for agent i just from formula φ
- $h^p_i(A)$ is the optimistic value of p for agent i over all formulas
- $h^p(A)$ is the optimistic value of p to the agent who optimistically values it most
- $h(A)$ is the optimistic value of all unallocated atoms

A Better Upper Bound Heuristic for Positive Cubes, II

For $\mathcal{L}(\textit{positive cubes}, \textit{positive})$, this heuristic works:

$$h(A) = \sum_{p \in \mathcal{PS}} h^p(A)$$

$$h^p(A) = \max_{i \in \mathcal{A}} h^p_i(A) \qquad h^p_i(A) = \sum_{(\varphi,w) \in G_i} h^p_i(A,\varphi)$$

$$h^p_i(A,\varphi) = \begin{cases} \frac{w}{|\mathrm{und}(A,\varphi)|} & \text{if } (\varphi,w) \in G_i, p \in \mathrm{und}(A,\varphi), M_i^A \not\models \varphi \\ 0 & \text{otherwise} \end{cases}$$

- p is an atom, i is an agent, A is a partial allocation
- M_i^A is the model induced for agent i by allocation A
- $h^p_i(A,\varphi)$ is the optimistic value of p for agent i just from formula φ
- $h^p_i(A)$ is the optimistic value of p for agent i over all formulas
- $h^p(A)$ is the optimistic value of p to the agent who optimistically values it most
- $h(A)$ is the optimistic value of all unallocated atoms

A Better Upper Bound Heuristic for Positive Cubes, II

For $\mathcal{L}(\mathit{positive\ cubes}, \mathit{positive})$, this heuristic works:

$$h(A) = \sum_{p \in \mathcal{PS}} h^p(A)$$

$$h^p(A) = \max_{i \in \mathcal{A}} h_i^p(A) \qquad h_i^p(A) = \sum_{(\varphi, w) \in G_i} h_i^p(A, \varphi)$$

$$h_i^p(A, \varphi) = \begin{cases} \frac{w}{|\mathrm{und}(A,\varphi)|} & \text{if } (\varphi, w) \in G_i, p \in \mathrm{und}(A, \varphi), M_i^A \mathrel{?} \varphi \\ 0 & \text{otherwise} \end{cases}$$

- p is an atom, i is an agent, A is a partial allocation
- M_i^A is the model induced for agent i by allocation A
- $h_i^p(A, \varphi)$ is the optimistic value of p for agent i just from formula φ
- $h_i^p(A)$ is the optimistic value of p for agent i over all formulas
- $h^p(A)$ is the optimistic value of p to the agent who optimistically values it most
- $h(A)$ is the optimistic value of all unallocated atoms

A Better Upper Bound Heuristic for Positive Cubes, II

For $\mathcal{L}(\textit{positive cubes}, \textit{positive})$, this heuristic works:

$$h(A) = \sum_{p \in \mathcal{PS}} h^p(A)$$

$$h^p(A) = \max_{i \in \mathcal{A}} h_i^p(A) \qquad h_i^p(A) = \sum_{(\varphi, w) \in G_i} h_i^p(A, \varphi)$$

$$h_i^p(A, \varphi) = \begin{cases} \frac{w}{|\mathrm{und}(A,\varphi)|} & \text{if } (\varphi, w) \in G_i, p \in \mathrm{und}(A, \varphi), M_i^A \mathrel{?} \varphi \\ 0 & \text{otherwise} \end{cases}$$

- p is an atom, i is an agent, A is a partial allocation
- M_i^A is the model induced for agent i by allocation A
- $h_i^p(A, \varphi)$ is the optimistic value of p for agent i just from formula φ
- $h_i^p(A)$ is the optimistic value of p for agent i over all formulas
- $h^p(A)$ is the optimistic value of p to the agent who optimistically values it most
- $h(A)$ is the optimistic value of all unallocated atoms

A Better Upper Bound Heuristic for Positive Cubes, II

For $\mathcal{L}(\textit{positive cubes}, \textit{positive})$, this heuristic works:

$$h(A) = \sum_{p \in \mathcal{PS}} h^p(A)$$

$$h^p(A) = \max_{i \in \mathcal{A}} h_i^p(A) \qquad h_i^p(A) = \sum_{(\varphi, w) \in G_i} h_i^p(A, \varphi)$$

$$h_i^p(A, \varphi) = \begin{cases} \frac{w}{|\mathrm{und}(A,\varphi)|} & \text{if } (\varphi, w) \in G_i, p \in \mathrm{und}(A, \varphi), M_i^A \,?\, \varphi \\ 0 & \text{otherwise} \end{cases}$$

- p is an atom, i is an agent, A is a partial allocation
- M_i^A is the model induced for agent i by allocation A
- $h_i^p(A, \varphi)$ is the optimistic value of p for agent i just from formula φ
- $h_i^p(A)$ is the optimistic value of p for agent i over all formulas
- $h^p(A)$ is the optimistic value of p to the agent who optimistically values it most
- $h(A)$ is the optimistic value of all unallocated atoms

A Better Upper Bound Heuristic for Positive Cubes, II

For $\mathcal{L}(\textit{positive cubes}, \textit{positive})$, this heuristic works:

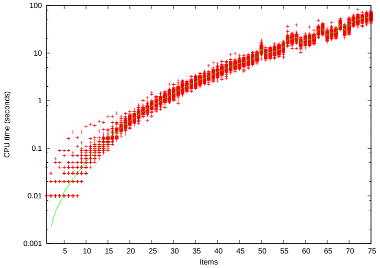
$$h(A) = \sum_{p \in \mathcal{PS}} h^p(A)$$

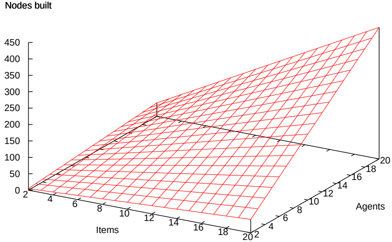
$$h^p(A) = \max_{i \in \mathcal{A}} h_i^p(A) \qquad h_i^p(A) = \sum_{(\varphi, w) \in G_i} h_i^p(A, \varphi)$$

$$h_i^p(A, \varphi) = \begin{cases} \frac{w}{|\mathrm{und}(A,\varphi)|} & \text{if } (\varphi, w) \in G_i, p \in \mathrm{und}(A, \varphi), M_i^A \mathrel{?} \varphi \\ 0 & \text{otherwise} \end{cases}$$

- p is an atom, i is an agent, A is a partial allocation
- M_i^A is the model induced for agent i by allocation A
- $h_i^p(A, \varphi)$ is the optimistic value of p for agent i just from formula φ
- $h_i^p(A)$ is the optimistic value of p for agent i over all formulas
- $h^p(A)$ is the optimistic value of p to the agent who optimistically values it most
- $h(A)$ is the optimistic value of all unallocated atoms

Experimental Results: How well does this work?





CPU time used by our `PCubeBF` solver (best-estimate-first branching), 20 agents, goods range from 2 to 75. At 75 goods, approx. 1500 atomic bids generated.

Avg. number of nodes built by `PCubeLex` for various numbers of items and agents. Full tree at $(20, 20)$ has $5.5 \times 10^2 4$ nodes. `PCubeLex` built about 450.

`PCubeLex` and `PCubeBF` perfromed almost the same; we expected `PCubeBF` to be *much* better. Our guess: `PCubeBF` would pull ahead if the bounds were tighter.

An IP formulation of the WDP for $\mathcal{L}(\mathit{positive\ cubes}, \mathit{all})$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} \leq 1$$

$$\text{For all } i, j \quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\mathit{positive\ cubes}, \mathit{all})$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} \leq 1$$

$$\text{For all } i, j \quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\textit{positive cubes}, \textit{all})$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} \leq 1$$

$$\text{For all } i, j \quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\textit{positive cubes}, \textit{all})$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} \leq 1$$

$$\text{For all } i, j \quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(positive\ cubes, all)$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} \leq 1$$

$$\text{For all } i, j \quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\textit{positive cubes}, \textit{all})$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} \leq 1$$

$$\text{For all } i, j \quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\textit{positive cubes}, \textit{all})$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} \leq 1$$

$$\text{For all } i, j \quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\mathit{positive\ cubes}, \mathit{all})$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} \leq 1$$

$$\text{For all } i, j \quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\textit{positive cubes}, \textit{all})$

Total revenue $\Longrightarrow$ Maximize $\sum_{ij} p_{ij} x_{ij}$ subject to:

For all i, r $\quad \sum_{i} y_{ir} \leq 1$

For all i, j $\quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\textit{positive cubes}, \textit{all})$

$$\text{Maximize} \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

Preemption $\Longrightarrow$

$$\text{For all } i, r \quad \sum_{i} y_{ir} \leq 1$$

$$\text{For all } i, j \quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\textit{positive cubes}, \textit{all})$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} \leq 1$$

Bid satisfaction $\Longrightarrow$ $$\text{For all } i, j \quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\textit{positive cubes}, \textit{all})$

Total revenue $\Longrightarrow$ Maximize $\sum_{ij} p_{ij} x_{ij}$ subject to:

Preemption $\Longrightarrow$ For all i, r $\quad \sum_{i} y_{ir} \leq 1$

Bid satisfaction $\Longrightarrow$ For all i, j $\quad x_{ij} \leq \min_{r \in S_{ij}} y_{ir}$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(\mathit{positive\ clauses}, \mathit{all})$

Total revenue $\Longrightarrow$ Maximize $\sum_{ij} p_{ij} x_{ij}$ subject to:

Preemption $\Longrightarrow$ For all $i, r \quad \sum_{i} y_{ir} \leq 1$

Bid satisfaction $\Longrightarrow$ For all $i, j \quad x_{ij} \leq \max_{r \in S_{ij}} y_{ir}$

- i indexes bidders
- j indexes formulas
- r indexes goods
- S_{ij} is the set of atoms appearing in the jth formula of the ith bidder,
- p_{ij} is the weight of that formula,
- x_{ij} is true iff that bid is accepted;
- y_{ir} is true iff bidder i is awarded item r.

An IP formulation of the WDP for $\mathcal{L}(cubes, all)$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} = 1$$

$$\text{For all } i, j \quad x_{ij} \leq \max_{r \in S_{ij}^{+}} y_{ir}$$

$$\text{For all } i, j \quad x_{ij} \leq 1 - \max_{r \in S_{ij}^{-}} y_{ir}$$

- S_{ij}^{+} is the set of *positive* literals in the jth formula of the ith bidder.
- S_{ij}^{-} is the set of *negative* literals in the jth formula of the ith bidder.

An IP formulation of the WDP for $\mathcal{L}(cubes, all)$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} = 1$$

$$\text{For all } i, j \quad x_{ij} \leq \max_{r \in S_{ij}^{+}} y_{ir}$$

$$\text{For all } i, j \quad x_{ij} \leq 1 - \max_{r \in S_{ij}^{-}} y_{ir}$$

- S_{ij}^{+} is the set of *positive* literals in the jth formula of the ith bidder.
- S_{ij}^{-} is the set of *negative* literals in the jth formula of the ith bidder.

An IP formulation of the WDP for $\mathcal{L}(cubes, all)$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} = 1$$

$$\text{For all } i, j \quad x_{ij} \leq \max_{r \in S^+_{ij}} y_{ir}$$

$$\text{For all } i, j \quad x_{ij} \leq 1 - \max_{r \in S^-_{ij}} y_{ir}$$

- S^+_{ij} is the set of *positive* literals in the jth formula of the ith bidder.
- S^-_{ij} is the set of *negative* literals in the jth formula of the ith bidder.

An IP formulation of the WDP for $\mathcal{L}(cubes, all)$

Total revenue $\Longrightarrow$ Maximize $\sum_{ij} p_{ij} x_{ij}$ subject to:

For all i, r $\quad \sum_{i} y_{ir} = 1$

For all i, j $\quad x_{ij} \leq \max_{r \in S_{ij}^{+}} y_{ir}$

For all i, j $\quad x_{ij} \leq 1 - \max_{r \in S_{ij}^{-}} y_{ir}$

- S_{ij}^{+} is the set of *positive* literals in the jth formula of the ith bidder.
- S_{ij}^{-} is the set of *negative* literals in the jth formula of the ith bidder.

An IP formulation of the WDP for $\mathcal{L}(cubes, all)$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

Preemption $\Longrightarrow$ $$\text{For all } i, r \quad \sum_{i} y_{ir} = 1$$

$$\text{For all } i, j \quad x_{ij} \leq \max_{r \in S_{ij}^{+}} y_{ir}$$

$$\text{For all } i, j \quad x_{ij} \leq 1 - \max_{r \in S_{ij}^{-}} y_{ir}$$

- S_{ij}^{+} is the set of *positive* literals in the jth formula of the ith bidder.
- S_{ij}^{-} is the set of *negative* literals in the jth formula of the ith bidder.

An IP formulation of the WDP for $\mathcal{L}(\mathit{cubes}, \mathit{all})$

$$\text{Maximize} \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} = 1$$

Bid satisfaction $(+) \Longrightarrow$ $$\text{For all } i, j \quad x_{ij} \leq \max_{r \in S_{ij}^{+}} y_{ir}$$

$$\text{For all } i, j \quad x_{ij} \leq 1 - \max_{r \in S_{ij}^{-}} y_{ir}$$

- S_{ij}^{+} is the set of *positive* literals in the jth formula of the ith bidder.
- S_{ij}^{-} is the set of *negative* literals in the jth formula of the ith bidder.

An IP formulation of the WDP for $\mathcal{L}(cubes, all)$

$$\text{Maximize } \sum_{ij} p_{ij} x_{ij} \text{ subject to:}$$

$$\text{For all } i, r \quad \sum_{i} y_{ir} = 1$$

$$\text{For all } i, j \quad x_{ij} \leq \max_{r \in S_{ij}^{+}} y_{ir}$$

Bid satisfaction $(-) \Longrightarrow$ $$\text{For all } i, j \quad x_{ij} \leq 1 - \max_{r \in S_{ij}^{-}} y_{ir}$$

- S_{ij}^{+} is the set of *positive* literals in the jth formula of the ith bidder.
- S_{ij}^{-} is the set of *negative* literals in the jth formula of the ith bidder.

An IP formulation of the WDP for $\mathcal{L}(cubes, all)$

Total revenue $\Longrightarrow$ Maximize $\sum_{ij} p_{ij} x_{ij}$ subject to:

Preemption $\Longrightarrow$ For all i, r $\quad \sum_{i} y_{ir} = 1$

Bid satisfaction $(+)$ $\Longrightarrow$ For all i, j $\quad x_{ij} \leq \max_{r \in S_{ij}^{+}} y_{ir}$

Bid satisfaction $(-)$ $\Longrightarrow$ For all i, j $\quad x_{ij} \leq 1 - \max_{r \in S_{ij}^{-}} y_{ir}$

- S_{ij}^{+} is the set of *positive* literals in the jth formula of the ith bidder.
- S_{ij}^{-} is the set of *negative* literals in the jth formula of the ith bidder.

An IP formulation of the WDP for $\mathcal{L}(cubes, all)$

Total revenue $\Longrightarrow$ Maximize $\sum_{ij} p_{ij} x_{ij}$ subject to:

Preemption $\Longrightarrow$ For all i, r $\quad \sum_{i} y_{ir} = 1$

Bid satisfaction $(+)$ $\Longrightarrow$ For all i, j $\quad x_{ij} \leq \max_{r \in S_{ij}^{+}} y_{ir}$

Bid satisfaction $(-)$ $\Longrightarrow$ For all i, j $\quad x_{ij} \leq 1 - \max_{r \in S_{ij}^{-}} y_{ir}$

- S_{ij}^{+} is the set of *positive* literals in the jth formula of the ith bidder.
- S_{ij}^{-} is the set of *negative* literals in the jth formula of the ith bidder.

Note: Use $=$ instead of $\leq$ here to prevent free disposal.

. . . and more general still:

$$\mathcal{L}(\mathit{clauses}, \mathit{all}) \implies \text{For all } i, j \quad x_{ij} \leq \max_{r \in S^+_{ij}} y_{ir} + 1 - \min_{r \in S^-_{ij}} y_{ir}$$

$$\mathcal{L}(\mathit{CNF}, \mathit{all}) \implies$$

$$\text{For all } i, j\text{: } 1, \ldots, k \begin{cases} x_{ij} \leq \max_{r \in S^+_{ij1}} y_{ir} + 1 - \min_{r \in S^-_{ij1}} y_{ir} \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ x_{ij} \leq \max_{r \in S^+_{ijk}} y_{ir} + 1 - \min_{r \in S^-_{ijk}} y_{ir} \end{cases}$$

where φ_{ij} has k conjuncts

IP is Fast... but B&B Could Be Improved

E.g., a representative instance with 57 goods and 20 agents solved by `PCubeBF` in 135s was solved by CPLEX in 16s.

`PCubeBF` gets runtime within a factor of 10 here:

- without using lower bounds aggressively
- without considering any conflicts among agents
- using only 2000 lines of code

We could get more pruning (= better times) by:

- doing tighter lower-bounding
- calculating upper bounds by summing over groups rather than single agents
- branching on bids?
- ...

Future Work

- Investigate branching on bids.
- Find better heuristics for languages other than $\mathcal{L}(pcubes, pos)$.
- Preprocessing of bids?
- Use more realistic test data.

Future Work

- Investigate branching on bids.
- Find better heuristics for languages other than $\mathcal{L}(pcubes, pos)$.
- Preprocessing of bids?
- Use more realistic test data.

Thank you!