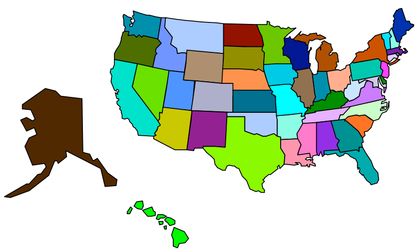
Advanced informed search

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University

Read: *Optimal Winner Determination Algorithms.*
Sandholm, T. 2006.
Chapter 14 of the book Combinatorial Auctions, Cramton, Shoham, and Steinberg, editors, MIT Press.

Example application:
Winner determination in multi-*item* auctions

- Auctioning multiple ***distinguishable*** items when **bidders have preferences over combinations of items: *complementarity & substitutability***
- Example applications
  - Allocation of transportation tasks
  - Allocation of bandwidth
    - Dynamically in computer networks
    - Statically e.g. by FCC



  - Sourcing
  - Electricity markets
  - Securities markets
  - Liquidation
  - Reinsurance markets
  - Retail ecommerce: collectibles, flights-hotels-event tickets
  - Resource & task allocation in operating systems & mobile agent platforms

Auction design for multi-item settings

- **Sequential auctions**
  - **How should rational agents bid (in equilibrium)?**
    - **Full vs. partial vs. no lookahead**
    - **Would need normative deliberation control methods**
  - **Inefficiencies can result from future uncertainties**
- **Parallel auctions**
  - **Inefficiencies can still result from future uncertainties**
  - **Postponing & minimum participation requirements**
    - **Unclear what equilibrium strategies would be**
- **Methods to tackle the inefficiencies**
  - **Backtracking via reauctioning (e.g. FCC** [McAfee&McMillan96]**)**
  - **Backtracking via leveled commitment contracts** [Sandholm&Lesser95,AAAI-96, GEB-01] [Sandholm96] [Andersson&Sandholm98a,b]
    - Breach before allocation
    - Breach after allocation

Auction design for multi-item settings…

- Combinatorial auctions [Rassenti,Smith&Bulfin82]...
  - Bids can be submitted on combinations (bundles) of items
  - Bidder's perspective
    - Avoids the need for lookahead
    - (Potentially $2^{\#items}$ valuation calculations)
  - Auctioneer's perspective:
    - Automated optimal bundling of items
    - Winner determination problem:
      - Label bids as winning or losing so as to maximize sum of bid prices (= revenue ≈ social welfare)
      - Each item can be allocated to at most one bid
    - Exhaustive enumeration is $2^{\#bids}$

Executing the mechanism:
Auctioneer's winner determination problem

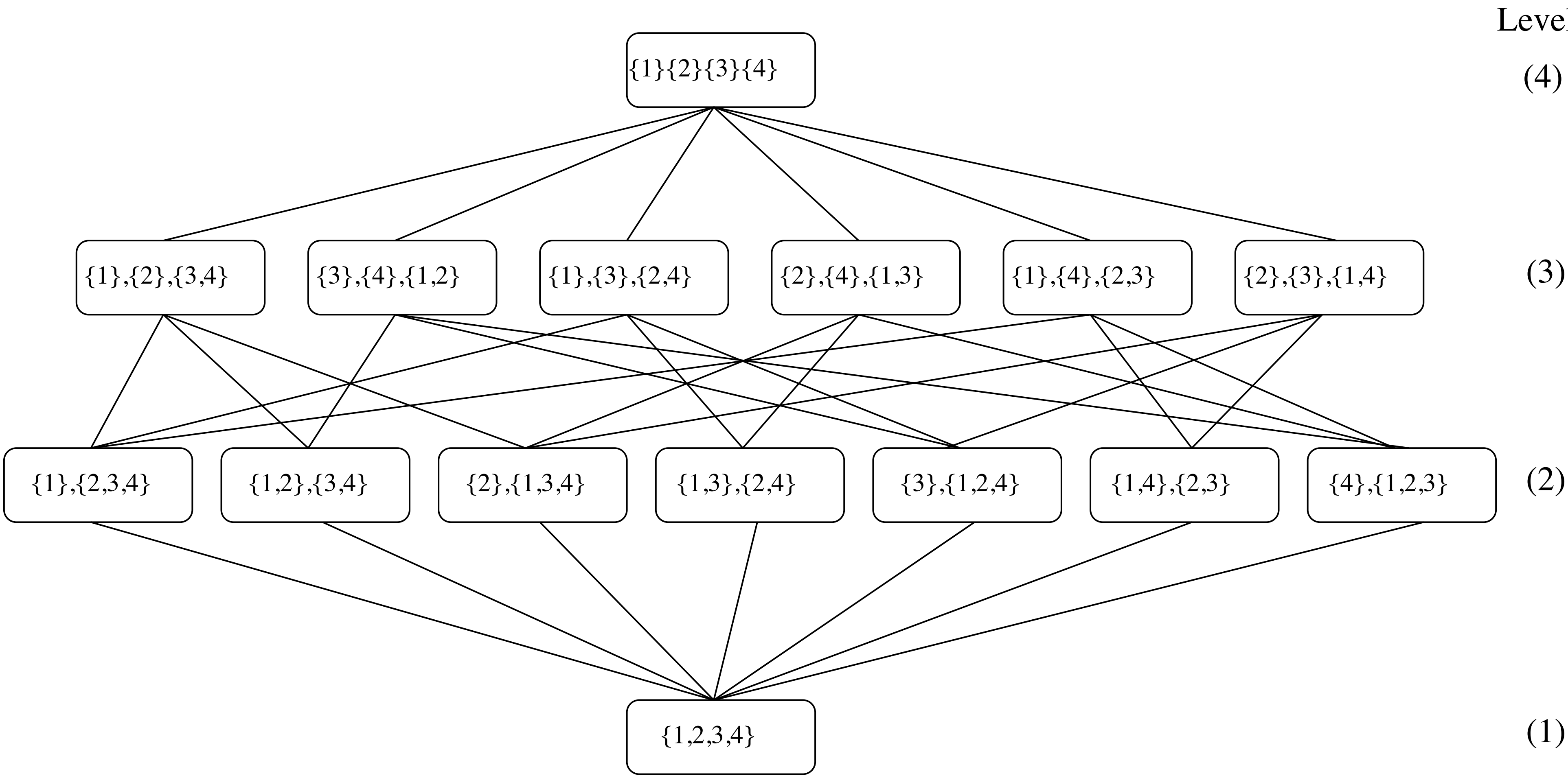
- **Set of items,** $M = \{1, 2, \ldots, \#items\}$
- **Set of bids,** $\mathcal{B} = \{B_1, B_2, \ldots, B_{\#bids}\}$
- $B_j = \langle S_j, p_j \rangle$, **where** $S_j \subseteq M$ **is a set of items and** p_j **is a price**
- $S_j \neq S_k$ **(if multiple bids concern the same set of items, all but the highest bid can be discarded by a preprocessor)**
- **Problem: Label the bids as winning** ($x_j = 1$) **or losing** ($x_j = 0$) **so as to maximize auctioneer's *revenue* such that each item is allocated to at most one bid:**

$$\max \sum_{j=1}^{\#bids} p_j x_j \quad \text{s.t.} \quad \sum_{j | i \in S_j} x_j \leq 1 \qquad i = 1, 2, \ldots, \#items$$

$$x_j \in \{0, 1\}$$

- **Without free disposal,** $\leq$ **becomes** $=$ **(also in generalizations)**

Space of allocations

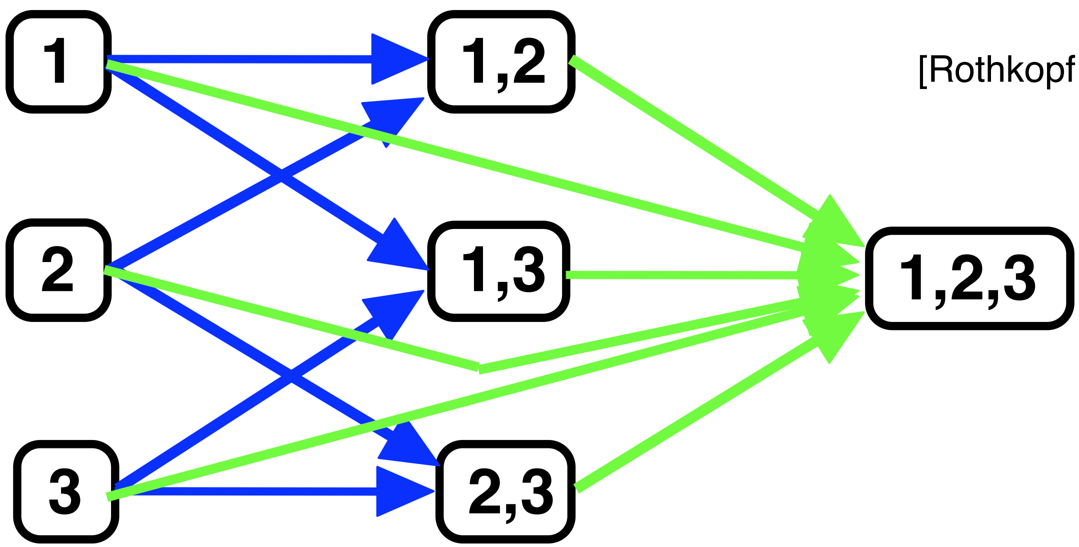


#partitions is $\omega(\#\text{items}^{\#\text{items}/2})$, $O(\#\text{items}^{\#\text{items}})$

[Sandholm et al. AAAI-98, AIJ-99, Sandholm AIJ-02]

Another issue: auctioneer could keep items

Dynamic programming for winner determination



[Rothkopf et al. Mgmt Sci 98]

- Uses $\Omega(2^{\#items})$, $O(3^{\#items})$ operations independent of #bids
  - (Can trivially exclude items that are not in any bid)
  - Does not scale beyond 20-30 items

NP-completeness

- NP-complete [Rothkopf et al Mgmt Sci 98]
  - Weighted set packing [Karp 72]

Polynomial time approximation algorithms with worst case guarantees

$$k = \frac{\textit{value of optimal allocation}}{\textit{value of best allocation found}}$$

General case

- **Cannot be approximated to $k = \#bids^{1-\varepsilon}$ (unless probabilistic polytime = NP)**
  - **Proven in** [Sandholm IJCAI-99, AIJ-02]
  - **Reduction from MAXCLIQUE, which is inapproximable** [Håstad96]
- **Best known approximation gives**

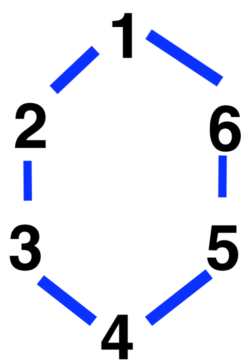
  $k \in O(\#bids / (\log \#bids)^2)$ [Haldorsson98]

Polynomial time approximation algorithms with worst case guarantees

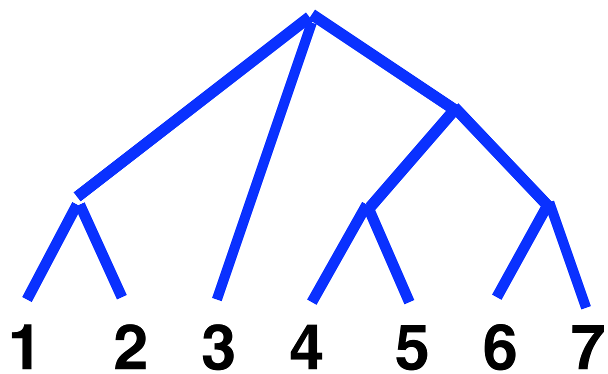
Special cases

- **Let κ be the max #items in a bid: k= 2κ / 3** [Haldorsson SODA-98]
- **Bid can overlap with at most Δ other bids:**
  k= min($\lceil(\Delta+1) / 3\rceil$, $(\Delta+2) / 3$, $\Delta / 2$) [Haldorsson&Lau97;Hochbaum83]
- **k= sqrt(#items)** [Haldorsson99]
- **k= chromatic number / 2** [Hochbaum83]
  - **k=[1 + $\max_{H \in G} \min_{v \in H}$ degree(v)] / 2** [Hochbaum83]
  - **Planar: k=2** [Hochbaum83]
- **So far from optimum that irrelevant for auctions**
- **Probabilistic algorithms?**
- **New special cases, e.g. based on prices** [Lehmann et al. 01, …]

Restricting the allowable combinations that can be bid on to get polytime winner determination [Rothkopf et al. Mgmt Sci 98]



$O(\#items^2)$
or
$O(\#items^3)$



$|set| \le 2$

or $|set| > \#items / c$

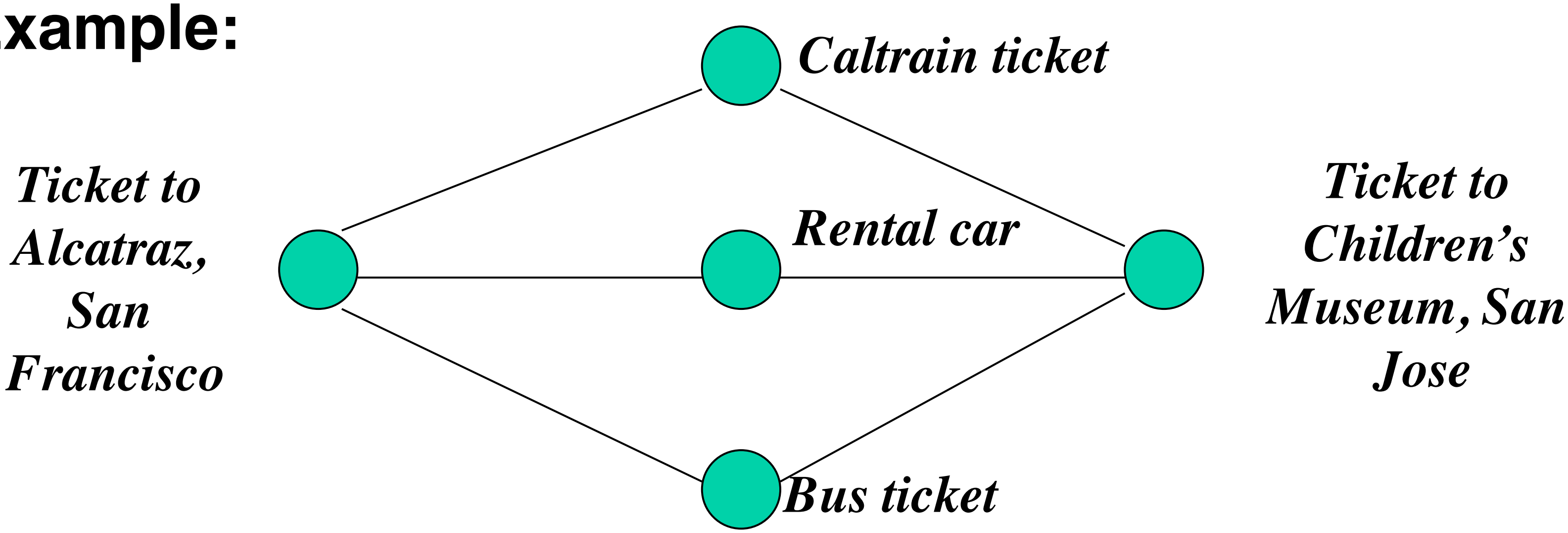
$O(n_{large}^{c-1} \ \#items^3)$

NP-complete already if 3 items per bid are allowed

Gives rise to the same economic inefficiencies that prevail in noncombinatorial auctions

Item *graphs* [Conitzer, Derryberry, Sandholm AAAI-04]

- **Item graph = graph with the items as vertices where every bid is on a connected set of items**
- **Example:**



- **Does not make sense to bid on items in SF and SJ without transportation**
- **Does not make sense to bid on two forms of transportation**

Clearing with item graphs

- *Tree decomposition* of a graph G = a tree T with
  - Subsets of G's vertices as T's vertices; for every G-vertex, set of T-vertices containing it must be a nonempty connected set in T
  - Every neighboring pair of vertices in G occurs in some single vertex of T
- *Width* of T = (max #G-vertices in single T-vertex)-1
  - (For bounded *w*, can construct tree decomposition of width *w* in polynomial time (if it exists))
- **Thrm.** Given an item graph with tree decomposition T (width w), can clear optimally in time $O(|T|^2 (|Bids|+1)^{w+1})$
  - Sketch: for every partial assignment of a T-vertex's items to bids, compute maximum possible value below that vertex (using DP)

Solving the winner determination problem when all combinations can be bid on:

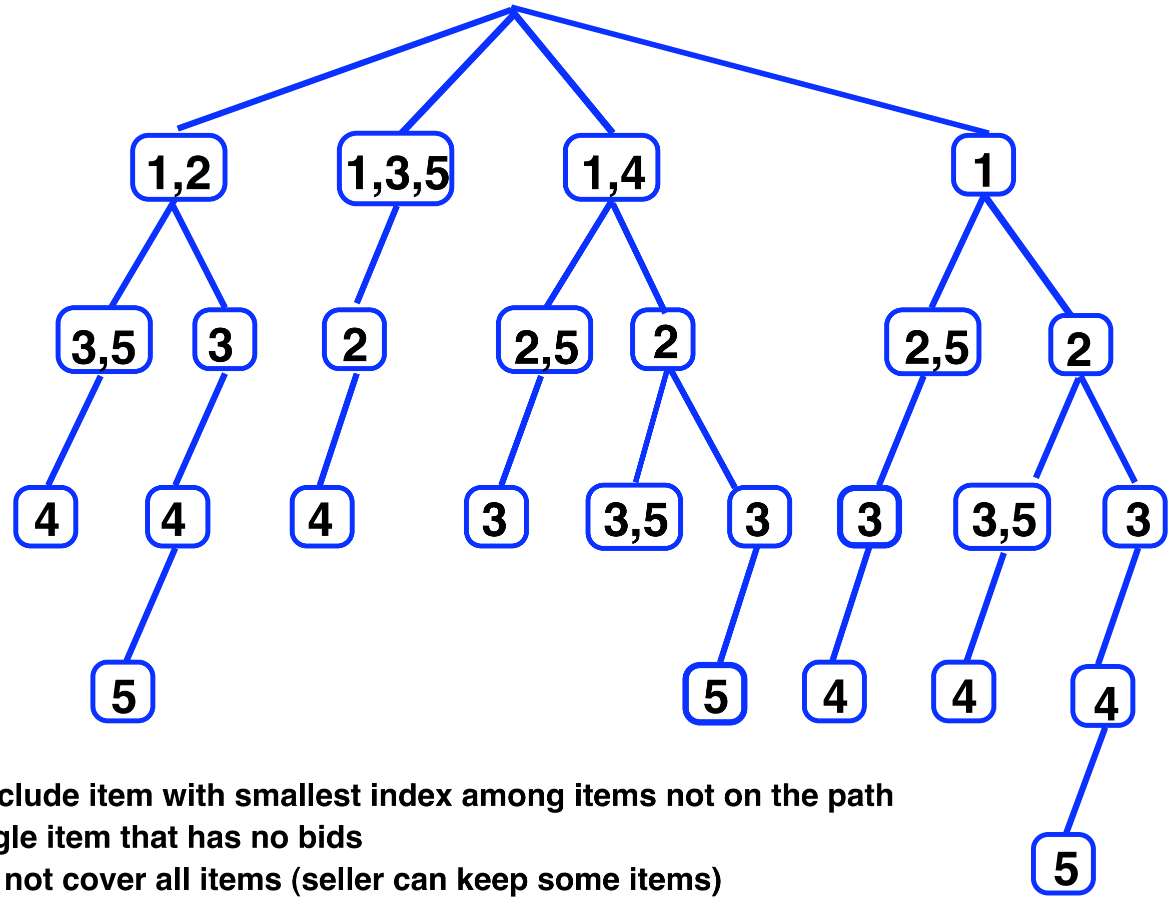
Search algorithms for optimal anytime winner determination

- **Capitalize on sparsely populated space of bids**
- **Generate only populated parts of space of allocations**
- **Highly optimized**
- **1st generation algorithm: branch-on-items formulation** [Sandholm ICE-98, IJCAI-99, AIJ-02; Fujishima, Leyton-Brown & Shoham IJCAI-99]
- **2nd generation algorithm: branch-on-bids formulation** [Sandholm&Suri AAAI-00, AIJ-03, Sandholm et al. IJCAI-01, MgmtSci-05]
- **New ideas, e.g., multivariate branching** [Gilpin & Sandholm IJCAI-07, …]

First generation search algorithms: *branch-on-items formulation*

[Sandholm ICE-98, IJCAI-99, AIJ-02]

Bids:
1
2
3
4
5
1,2
1,3,5
1,4
2,5
3,5



Prop. Need only consider children that include item with smallest index among items not on the path

Insert dummy bid for price 0 for each single item that has no bids

=> allows bid combinations that do not cover all items (seller can keep some items)

Generates each allocation of positive value once, others not generated

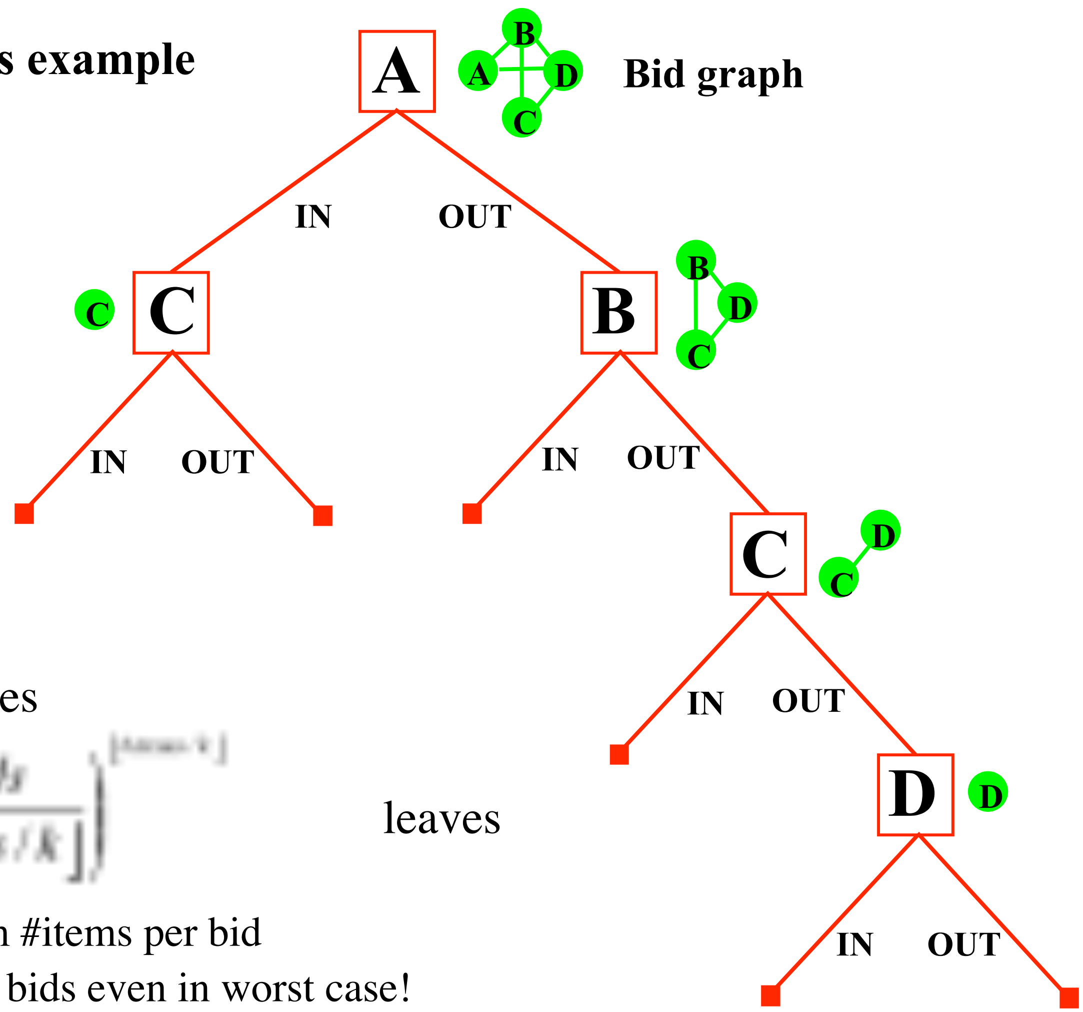
Complexity

- **Prop. #leaves ≤ (#bids/#items)$^{\#items}$**
- **Proof. Let n_i be the number of bids that include item i but no items with smaller index. #leaves ≤ max $n_1 \cdot n_2 \cdot \ldots \cdot n_m$ s.t. $n_1 + n_2 + \ldots + n_m$ = #bids. Max achieved at $n_i = n/m$. Depth at most m. QED**
- **#nodes ≤ #items #leaves**
- **IDA* is 2 orders of magnitude faster than depth first search**
- **Anytime algorithm**

2nd generation algorithm: Combinatorial Auction, *Branch On Bids*

[Sandholm&Suri AAAI-00, AIJ-03]

Bids of this example
A={1,2}
B={2,3}
C={3}
D={1,3}



- Finds an optimal solution
- Naïve analysis: $2^{\#bids}$ leaves
- Thrm. At most $\left(\frac{\#bids}{\lfloor \#items/k \rfloor}\right)^{\lfloor \#items/k \rfloor}$ leaves
  - where k is the minimum #items per bid
  - provably polynomial in bids even in worst case!

Use of h-values (=upper bounds) to prune winner determination search

- f^* = value of best solution found so far
- g = sum of prices of bids that are IN on path
- h = value of LP relaxation of remaining problem
- Upper bounding: Prune the path when $g+h \le f^*$

Linear programming for computing h-values

Linear program of the winner determination problem

LP

$$\max \sum_{j=1}^{n} p_j x_j$$

$$\sum_{j|i \in S_j} x_j \leq 1, \; \forall i \in \{1..m\}$$

$$x_j \geq 0$$

$$x_j \in \mathbb{R}$$

Linear programming

Original problem

maximize $\sum_{j=1}^{n} c_j x_j$

such that

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad (i = 1, 2, \ldots, m)$$

$$x_j \geq 0 \quad (j = 1, 2, \ldots, n)$$

Initial tableau

$$z = \sum_{j=1}^{n} c_j x_j$$

$$x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j \quad (i = 1, 2, \ldots, m)$$

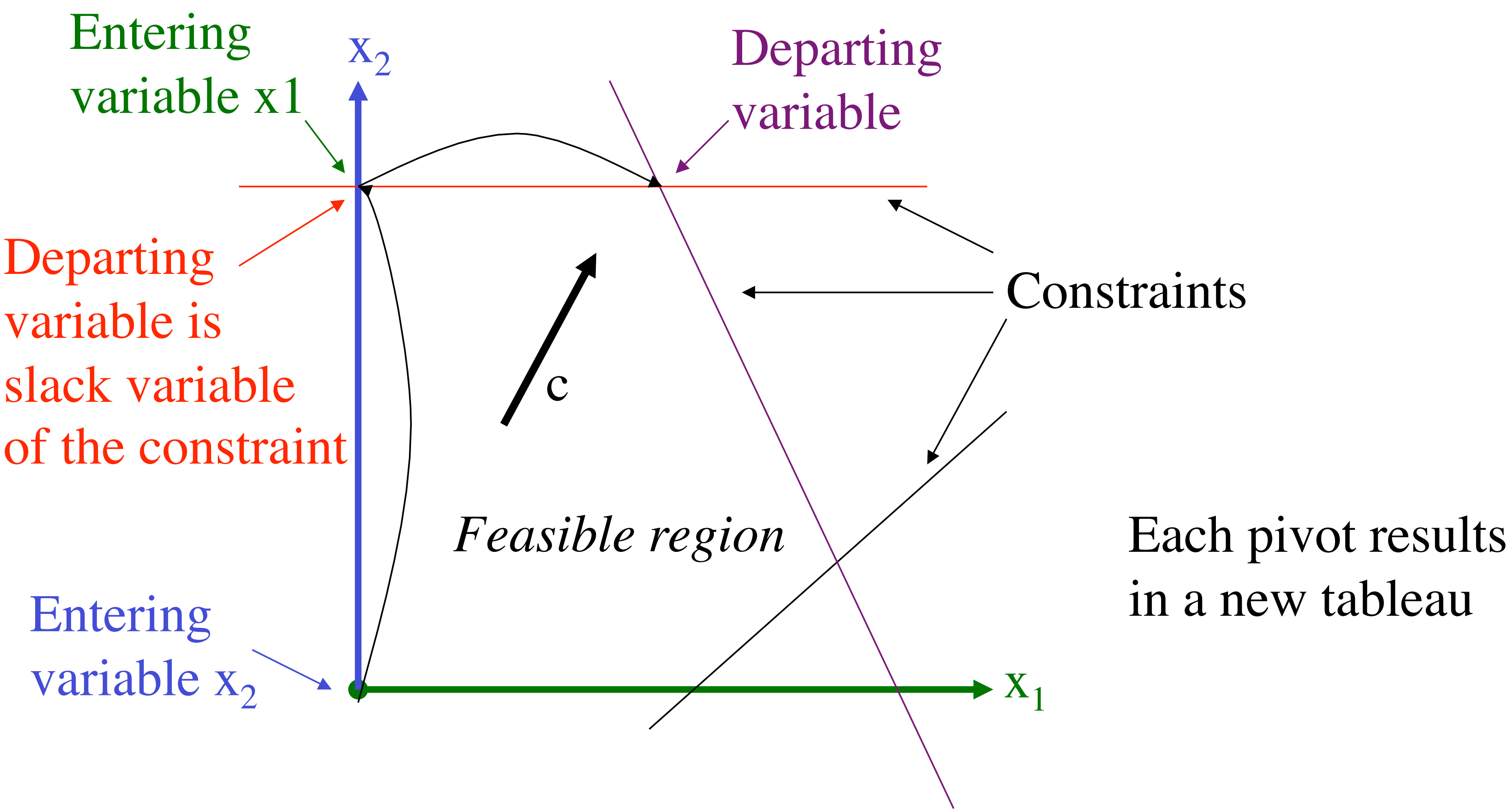
Slack variables

Assume, for simplicity, that origin is feasible (otherwise have to run a different LP to find first feasible and run the main LP in a revised space).

Simplex method "pivots" variables in and out of the tableau

Basic variables are on the left hand side

Graphical interpretation of simplex algorithm for linear programming



Interior point methods are another family of algorithms for linear programming

Speeding up the use of linear programs in search

- If LP returns a solution where all integer variables have integer values, then that is the solution to that node and no further search is needed below that node
- Instead of simplex in the LP, use simplex in the DUAL because after branching, the previous DUAL solution is still feasible and a good starting point for simplex at the new node (see next slide)
  - **Thrm.** LP optimum value = DUAL optimum value

LP

$$\max \sum_{j=1}^{n} p_j x_j$$

$$\sum_{j | i \in S_j} x_j \leq 1, \ \forall i \in \{1..m\}$$

$$x_j \geq 0$$

$$x_j \in \mathbb{R}$$

$DUAL$

$$\min \sum_{i=1}^{m} y_i$$

$$\sum_{i \in S_j} y_i \geq p_j, \ \forall j \in \{1..n\}$$

aka *shadow price* $\longrightarrow$ $y_i \geq 0$

$$y_i \in \mathbb{R}$$

Example showing DUAL is feasible at children

Goods: {1,2,3}, Bids: $<\{1,2\},\$4>$, $<\{1,3\},\$3>$, $<\{2,3\},\$2>$

LP

$$\begin{aligned} \max \quad & 4x_1 + 3x_2 + 2x_3 \\ \text{s.t.} \quad & x_1 + x_2 \le 1 \\ & x_1 + x_3 \le 1 \\ & x_2 + x_3 \le 1 \\ & x_1, x_2, x_3 \ge 0 \end{aligned}$$

$$\boxed{x_1^* = x_2^* = x_3^* = \frac{1}{2}}$$

DUAL

$$\begin{aligned} \min \quad & y_1 + y_2 + y_3 \\ \text{s.t.} \quad & y_1 + y_2 \ge 4 \\ & y_1 + y_3 \ge 3 \\ & y_2 + y_3 \ge 2 \\ & y_1, y_2, y_3 \ge 0 \end{aligned}$$

$$\boxed{y_1^* = \frac{5}{2}, y_2^* = \frac{3}{2}, y_3^* = \frac{1}{2}}$$

$x_2 \le 0$

LP

$$\begin{aligned} \max \quad & 4x_1 + 3x_2 + 2x_3 \\ \text{s.t.} \quad & x_1 + x_2 \le 1 \\ & x_1 + x_3 \le 1 \\ & x_2 + x_3 \le 1 \\ & x_2 \le 0 \\ & x_1, x_2, x_3 \ge 0 \end{aligned}$$

Infeasible ($x_2 > 0$)

DUAL

$$\begin{aligned} \min \quad & y_1 + y_2 + y_3 + 0y_4 \\ \text{s.t.} \quad & y_1 + y_2 \ge 4 \\ & y_1 + y_3 + y_4 \ge 3 \\ & y_2 + y_3 \ge 2 \\ & y_1, y_2, y_3, y_4 \ge 0 \end{aligned}$$

Feasible
(for any y_4)

$x_2 \ge 1$

LP

$$\begin{aligned} \max \quad & 4x_1 + 3x_2 + 2x_3 \\ \text{s.t.} \quad & x_1 + x_2 \le 1 \\ & x_1 + x_3 \le 1 \\ & x_2 + x_3 \le 1 \\ & -x_2 \le -1 \\ & x_1, x_2, x_3 \ge 0 \end{aligned}$$

Infeasible ($x_2 < 1$)

DUAL

$$\begin{aligned} \min \quad & y_1 + y_2 + y_3 - y_4 \\ \text{s.t.} \quad & y_1 + y_2 \ge 4 \\ & y_1 + y_3 - y_4 \ge 3 \\ & y_2 + y_3 \ge 2 \\ & y_1, y_2, y_3, y_4 \ge 0 \end{aligned}$$

Feasible
(for $y_4 = 0$)

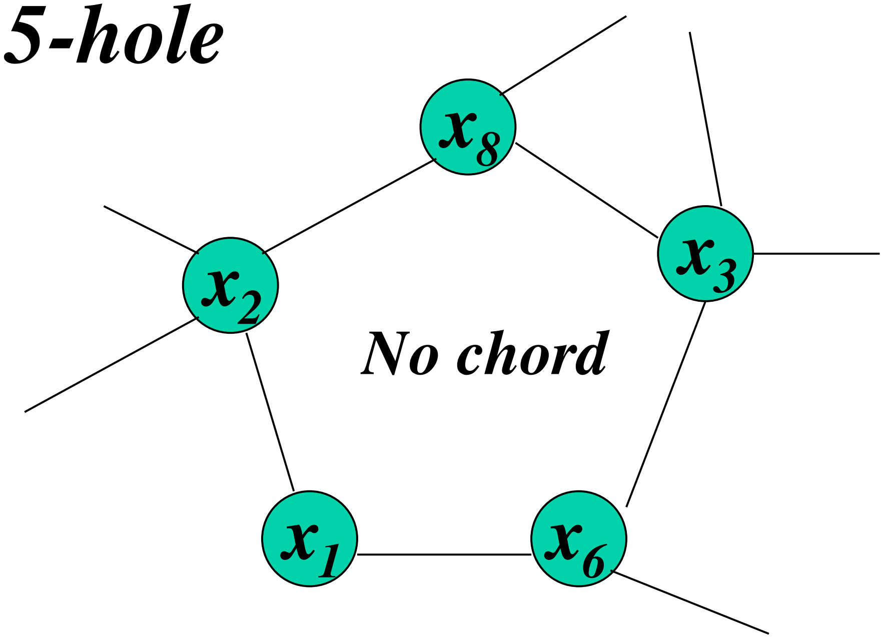
The branch-and-cut approach

Cutting planes (aka cuts)

- Extra linear constraints can be added to the LP to reduce the LP polytope and thus give tighter bounds (less optimistic h-values) if the constraints are guaranteed to not exclude any integer solutions
- Applications-specific vs. general-purpose cuts
- *Branch-and-cut algorithm* = branch-and-bound algorithm that uses cuts
  - A *global cut* is valid throughout the search tree
  - A *local cut* is guaranteed to be valid only in the subtree below the node at which it was generated (and thus needs to be removed from consideration when not in that subtree)

Example of a cut that is valid for winner determination: *Odd hole inequality*

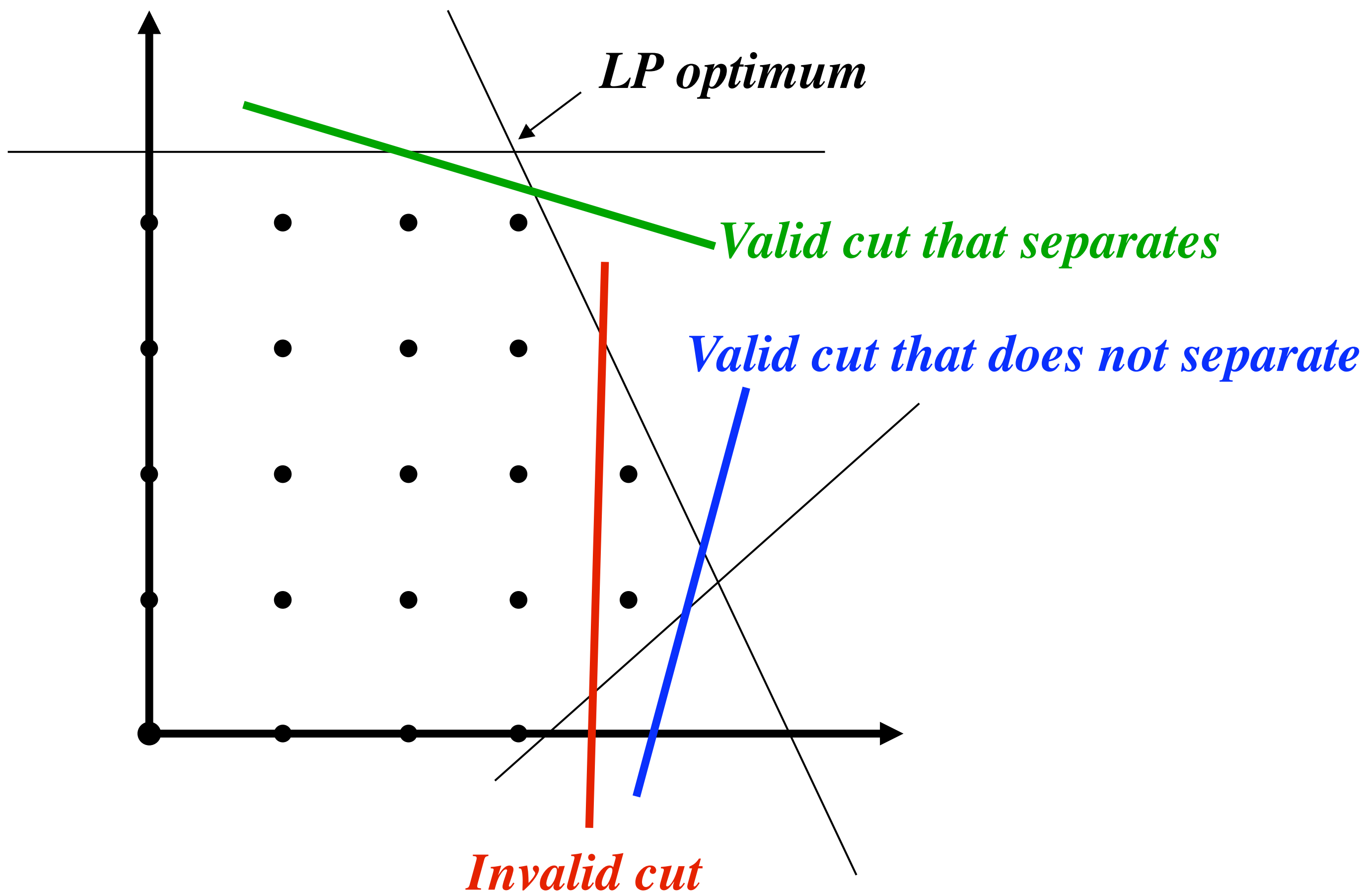
E.g., 5-hole



Edge means that bids share items, so both bids cannot be accepted

$$x_1 + x_2 + x_3 + x_6 + x_8 \leq 2$$

Separation using cuts



How to find cuts that separate?

- For some cut families (and/or some problems), there are polynomial-time algorithms for finding a separating cut
- Otherwise, use:
  - Generate a cut
    - Generation preferably biased towards cuts that are likely to separate
  - Test whether it separates

Gomory mixed integer cut

- Most powerful general-purpose cut for many problems
- Applicable to all problems, where
  - constraints and objective are linear,
  - the problem has integer variables and potentially also real variables
- Cut is generated using the LP optimum so that the cut separates

Interesting tidbit (which we will not use here): Gomory's cutting plane algorithm

*Integer program can be solved with **no search** by an algorithm that generates a finite (potentially exponential) number of these cuts.*

Between the generation of cuts, the (dual) LP is solved.

The LP tableau guides which cut is generated next.

Rules against cycling in the LP solving are needed to guarantee optimality in a finite number of steps (see, e.g., http://www.math.unl.edu/~shartke2/teaching/2008f432/Handout_Gomory.pdf).

While this algorithm has been viewed as a mere curiosity, it has very recently shown promise on some practical problems (the anti-cycling rule is key).

Derivation of Gomory mixed integer cut

Input: one row from optimal tableau: $x_i = a_{i0} + \sum_{j \in J} a_{ij}(-x_j),\ J = J_1 \cup J_2$

Fractional, basic, not a slack, integer variable

Non-basic. Integer. Continuous.

Define: $f_{ij} = a_{ij} - \lfloor a_{ij} \rfloor, \quad j \in J_1 \cup \{0\}$

$J_1^{\leq} = \{j \in J_1 | f_{ij} \leq f_{i0}\}, J_1^{>} = \{j \in J_1 | f_{ij} > f_{i0}\}, J_2^{+} = \{j \in J_2 | a_{ij} \geq 0\}, J_2^{-} = \{j \in J_2 | a_{ij} < 0\}$

Rewrite tableau row: $x_i = \lfloor a_{i0} \rfloor + f_{i0} + \sum_{j \in J_1^{\leq}} \lfloor a_{ij} \rfloor(-x_j) + \sum_{j \in J_1^{\leq}} f_{ij}(-x_j)$

$+ \sum_{j \in J_1^{>}} \lceil a_{ij} \rceil(-x_j) + \sum_{j \in J_1^{>}} (f_{ij}-1)(-x_j) + \sum_{j \in J_2^{+}} a_{ij}(-x_j) + \sum_{j \in J_2^{-}} a_{ij}(-x_j)$

Idea: RHS above has to be integral.
All integer terms add up to integers, so: $\sum_{j \in J_1^{\leq}} f_{ij}x_j + \sum_{j \in J_1^{>}} (f_{ij}-1)x_j + \sum_{j \in J_2^{+}} a_{ij}x_j + \sum_{j \in J_2^{-}} a_{ij}x_j \equiv f_{i0}$

LHS and RHS differ by an integer

$\Rightarrow \text{LHS} \leq f_{i0} - 1 \bigvee \text{LHS} \geq f_{i0} \Rightarrow \text{LHS}^{-} \leq f_{i0} - 1 \bigvee \text{LHS}^{+} \geq f_{i0}$

$\Rightarrow \frac{f_{i0}}{f_{i0}-1}\text{LHS}^{-} \geq f_{i0} \bigvee \text{LHS}^{+} \geq f_{i0} \Rightarrow \frac{f_{i0}}{f_{i0}-1}\text{LHS}^{-} + \text{LHS}^{+} \geq f_{i0}$

$\Rightarrow \sum_{j \in J_1^{\leq}} f_{ij}x_j + \sum_{j \in J_1^{>}} \frac{f_{i0}}{1-f_{i0}}(1-f_{ij})x_j + \sum_{j \in J_2^{+}} a_{ij}x_j + \sum_{j \in J_2^{-}} \frac{f_{i0}}{1-f_{i0}}(-a_{ij})x_j \geq f_{i0}$

Back to search for winner determination…

Formulation comparison

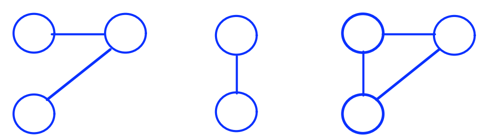
- A branching decision
  - in the branch-on-bids formulation locks in only one bid (and on the IN branch also its neighbors)
  - in the branch-on-items formulation locks in all bids that include that item

- The former follows the *principle of least commitment*
  - More flexibility for further decision ordering (choice of which decision to branch on in light of the newest information)

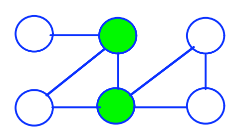
Structural improvements to search algorithms for winner determination

Optimum reached faster & better anytime performance

- **Always branch on a bid j that maximizes e.g. $p_j / |S_j|^{\alpha}$ (presort)**
- **Lower bounding: If g+L>f*, then f*←g+L**
- **Identify decomposition of bid graph in O(|E|+|V|) time & exploit**



  - **Pruning across subproblems (upper & lower bounding) by using f* values of solved subproblems and h values of yet unsolved ones**
- **Forcing decomposition by branching on an articulation bid**



  - **All articulation bids can be identified in O(|E|+|V|) time**
  - **Could try to identify *combinations* of bids that articulate (cutsets)**

Price-based vs. articulation-based bid ordering

Proposition. For any scheme that picks a bid that maximizes $\frac{p_j}{\phi(|S_j|)}$ for any given positive function ϕ (ties can be broken in any way) and any scheme that picks an articulation bid if one exists (ties can be broken in any way), there are instances where the former leads to fewer search nodes, as well as instances where the latter leads to fewer.

Question ordering heuristics

- In depth-first branch-and-bound, it is sometimes best to branch on a question for which the algorithm knows a good answer with high likelihood
  - Best (to date) heuristics for branching on bids [Sandholm et al. IJCAI-01, MgmtSci-05]:
    - A: Branch on bid whose LP value is closest to 1
    - B: Branch on bid with highest normalized shadow surplus:
      $$\frac{p_j - \sum_{i \in S_j} y_i}{\log(\sum_{i \in S_j} y_i)}$$
  - Choosing the heuristic *dynamically* based on remaining subproblem
    - E.g. use A when LP table density > 0.25 and B otherwise
- In A* search, it is usually best to branch on a question whose right answer the algorithm is very uncertain about
  - Traditionally in OR, variable whose LP value is most fractional
  - More general idea [Gilpin&Sandholm 03]: branch on a question that reduces the *entropy* of the LP solution the most
    - Determine this e.g. based on lookahead
    - Applies to multivariate branching too

Branching on more general questions than individual variables [Gilpin&Sandholm 03, IJCAI-07]

- Branching question: "Of these k bids, are more than x winners?"
- Never include bids whose LP values are integers
- Never use a set of bids whose LP values sum to an integer
- Prop. Only one sensible cutoff of x
- Prop. The search space size is the same regardless of which bids (and how many) are selected for branching
- Usually yields smaller search trees than branching on individual bids only

- More generally in MIP, one branch one can branch on hyperplanes: one branch is $\sum_{i \in S} \alpha_i x_i \leq c_1$ and the other branch is $\sum_{i \in S} \alpha_i x_i > c_2$ for some S
  - But how to decide on which hyperplane to branch?
  - For more on this approach, see, e.g., Improved Strategies for Branching on General Disjunctions by Gerard Cornuejols, Leo Liberti and Giacomo Nannicini, July 2008

Other good branching rules (for integer programs)

- Strong branching (= 1-step lookahead)
  - At a node, for each variable (from a set of promising candidate variable) in turn, pretend that you branch on that variable and solve the node's childrens' LPs
    - Sometimes child LPs are not solved to optimality (cap on # of dual pivots) to save time
  - Pick the variable to branch on that leads to tightest child LP bounds
    - Sometimes better and worse child are weighted differently
- Reliability branching
  - Like strong branching, but once lookahead for a certain variable has been conducted at a large enough number of nodes, stop doing lookahead for that variable, and use average reduction in bound in past lookaheads for that variable as that variable's goodness measure
- These could be used when branching on hyperplanes too

Identifying & solving tractable cases at search nodes (so that no search is needed below such nodes)

[Sandholm & Suri AAAI-00, AIJ-03]

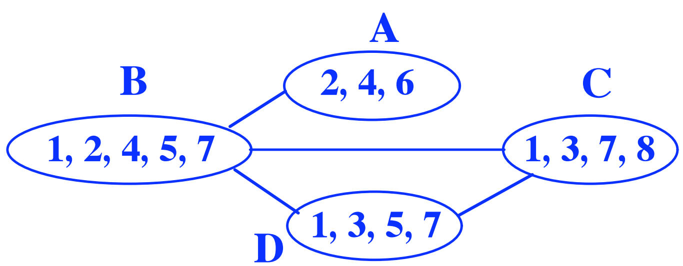
Example 1: “Short” bids

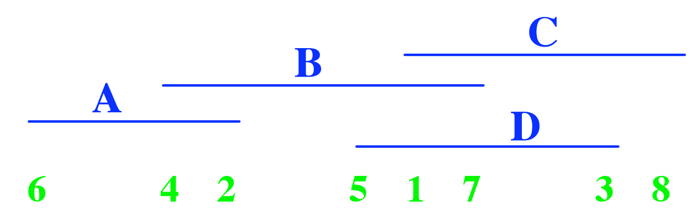
[Sandholm&Suri AAAI-00, AIJ-03]

- **Never branch on *short* bids with 1 or 2 items**
  - At each search node, we solve short bids from bid graph separately
    - O(#short bids ^{3}) time using maximal weighted matching
      - [Edmonds 65; Rothkopf et al 98]
    - NP-complete even if only 3 items per bid allowed
  - Dynamically delete items included in only one bid

Example 2: Interval bids

- **At each search node, use a polynomial algorithm if remaining bid graph only contains *interval bids***
  - Ordered list of items: 1..#items
  - Each bid is for some interval [q, r] of these items
  - [Rothkopf et al. 98] presented O(#items^2) DP algorithm
  - [Sandholm&Suri AAAI-00, AIJ-03] DP algorithm is O(#items + #bids)
    - Bucket sort bids in ascending order of r
    - opt(i) is the optimal solution using items 1..i
    - $\text{opt}(i) = \max_{b \text{ in bids whose last item is } i} \{p_b + \text{opt}(q_b - 1), \text{opt}(i-1)\}$
- **Identifying linear ordering**



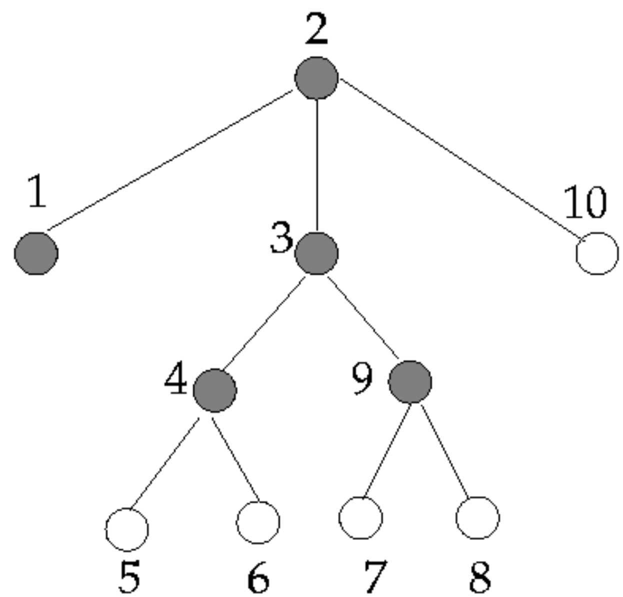


  - Can be identified in O(|E|+|V|) time [Korte & Mohring SIAM-89]
- **Interval bids with wraparound can be identified in O(#bids^2) time** [Spinrad SODA-93] **and solved in O(#items (#items + #bids)) time using our DP while DP of Rothkopf et al. is O(#items^3)**

Example 3:
A more general tractable special case

[Sandholm & Suri AAAI-00, AIJ-03]

- **Set of items structured as a tree**
- **Each bid is for some tree of items**



- **Our $O(\#items \cdot \#bids)$ DP algorithm solves this**
- **Proposition.** If the set of items is structured as a DAG D, and each bid is a tree in D, then winner determination is $\mathcal{NP}$-complete.

Example 3...

- **Thrm.** [Conitzer, Derryberry & Sandholm AAAI-04] An item tree that matches the remaining bids (if one exists) can be *constructed* in time O(|Bids| |#items that any one bid contains|^2 + |Items|^2)
- Algorithm**:**
  - Make a graph with the items as vertices
  - Each edge (*i, j)* gets weight *#(bids with both i and j)*
  - Construct maximum spanning tree of this graph: O(|Items|^2) time
  - **Thrm.** The resulting tree will have the maximum possible weight *#(occurrences of items in bids)* - |Bids| iff it is a valid item tree
- **Complexity of constructing an item graph of treewidth 2 (or 3, or 4, …) is unknown (but complexity of solving any such case *given the item graph* is "polynomial-time" - exponential only in the treewidth)**

Hardness of related questions

- **Constructing the item graph with the fewest edges is NP-complete**
  - **Even when each bid is on at most 5 items, and an item graph of treewidth at most 2 is known to exist; regardless of whether we require the constructed tree to have treewidth 2.**

- **What if a bid can include a few *(say, k)* connected sets rather than just one?**
  - **Clearing is NP-complete even when the graph is a line and $k = 2$**
  - **Deciding whether a line graph exists with $k = 5$ is NP-complete**

Preprocessors [Sandholm IJCAI-99, AIJ-02]

- **Only keep highest bid for each combination that has received bids**
- **Superset pruning**
  - E.g. ⟨{1,2,3,4}, $10⟩ is pruned by ⟨{1,3}, $7⟩ and ⟨{2,4}, $6⟩
  - For each bid (prunee), use same search algorithm as main search, except restrict to bids that are subsets of prunee
  - Terminate the search and prune the prunee if $f^* \geq$ prunee's price
  - Only consider bids with ≤ 30 items as potential prunees
- **Tuple pruning**
  - E.g. ⟨{1,2}, $8⟩ and ⟨{3,4}, $3⟩ are not competitive together given ⟨{1,3}, $7⟩ and ⟨{2,4}, $6⟩
  - Construct virtual prunee from pair of bids with disjoint item sets
  - Use same pruning algorithm as superset pruning
  - If pruned, insert an edge into bid graph between the bids
  - O(#bids^2 cap #items)
  - O(#bids^3 cap #items) for pruning triples, etc.
    - More complex checking required in main search

Generalization: ***substitutability***

[Sandholm IJCAI-99, AIJ-02]

- **What if agent 1 bids**
  - $7 for {1,2}
  - $4 for {1}
  - $5 for {2} ?
- **Bids joined with XOR**
  - Allows bidders to express general preferences
  - Groves-Clarke pricing mechanism can be applied to make truthful bidding a dominant strategy
  - Worst case: Need to bid on all $2^{\#items}$-1 combinations
- **OR-of-XORs bids maintain full expressiveness & are more concise**
  - E.g. (B_2 **XOR** B_3) **OR** (B_1 **XOR** B_3 **XOR** B_4) **OR ...**
  - Our algorithm applies (simply more edges in bid graph => faster)
    - Preprocessors do not apply
    - Short bid technique & interval bid technique do not apply