Improved Sample Complexity Bounds for Branch-and-Cut

Nina Balcan Carnegie Mellon University

Siddharth Prasad Carnegie Mellon University

Tuomas Sandholm Carnegie Mellon University, Optimized Markets, Inc., Strategic Machine, Inc., Strategy Robot, Inc.

Ellen Vitercik Stanford University

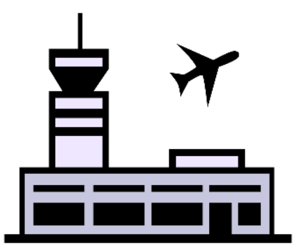
CP 2022

Integer programming

- Integer program (IP) in standard form:

$$\boxed{\begin{aligned} \text{Max } & \boldsymbol{c} \cdot \boldsymbol{x} \\ \text{s.t. } & A\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{x} \in \mathbb{Z}^n \end{aligned}}$$

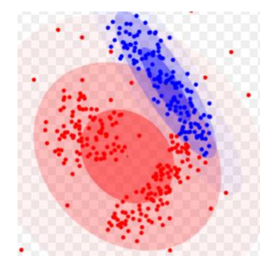
- One of the most useful and widely applicable optimization techniques



Scheduling

Routing

Combinatorial auctions



Clustering

Branch-and-cut

- Powerful tree-search algorithm used by fastest solvers to solve IPs in practice

- Our contribution: improved theory for using machine learning to tune (1) general model of tree search and (2) any-and-all aspects of branch-and-cut

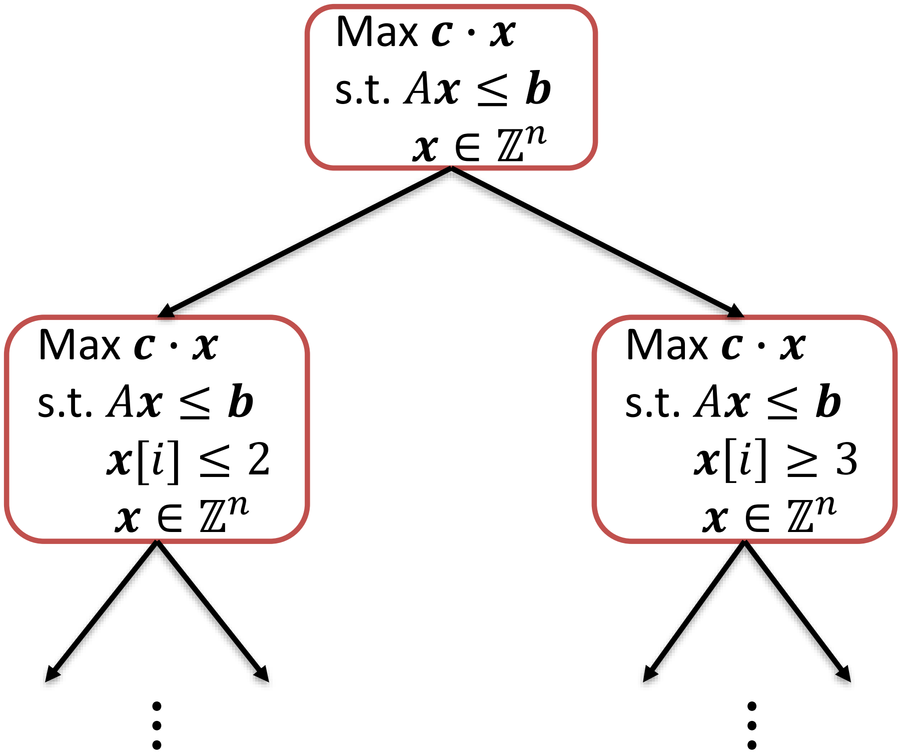
Branch-and-bound

- Powerful tree-search algorithm used to solve IPs in practice
- Uses the linear programming (LP) relaxation to do an informed search through the set of feasible integer solutions

IP		LP relaxation
Max $\boldsymbol{c} \cdot \boldsymbol{x}$ s.t. $A\boldsymbol{x} \leq \boldsymbol{b}$ $\boldsymbol{x} \in \mathbb{Z}^n$	$\longrightarrow$	Max $\boldsymbol{c} \cdot \boldsymbol{x}$ s.t. $A\boldsymbol{x} \leq \boldsymbol{b}$ $\boldsymbol{x} \in \mathbb{R}^n$

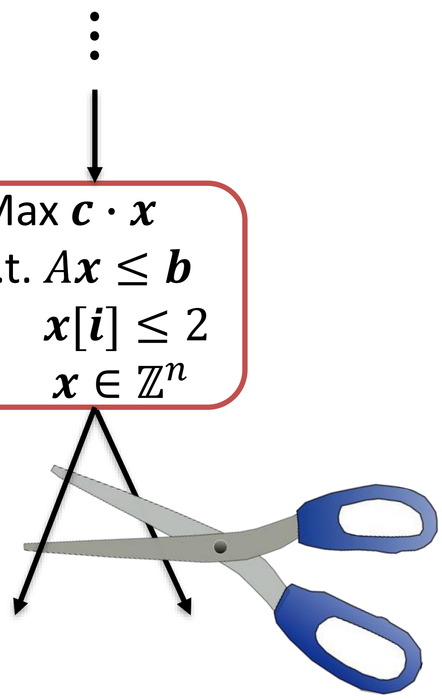
Branch-and-bound: branching

- Choose variable *i* to branch on.
- Generate one subproblem with $\boldsymbol{x}[i] \leq \lfloor \boldsymbol{x}^*_{\mathrm{LP}}[i] \rfloor$ another with $\boldsymbol{x}[i] \geq \lceil \boldsymbol{x}^*_{\mathrm{LP}}[i] \rceil$



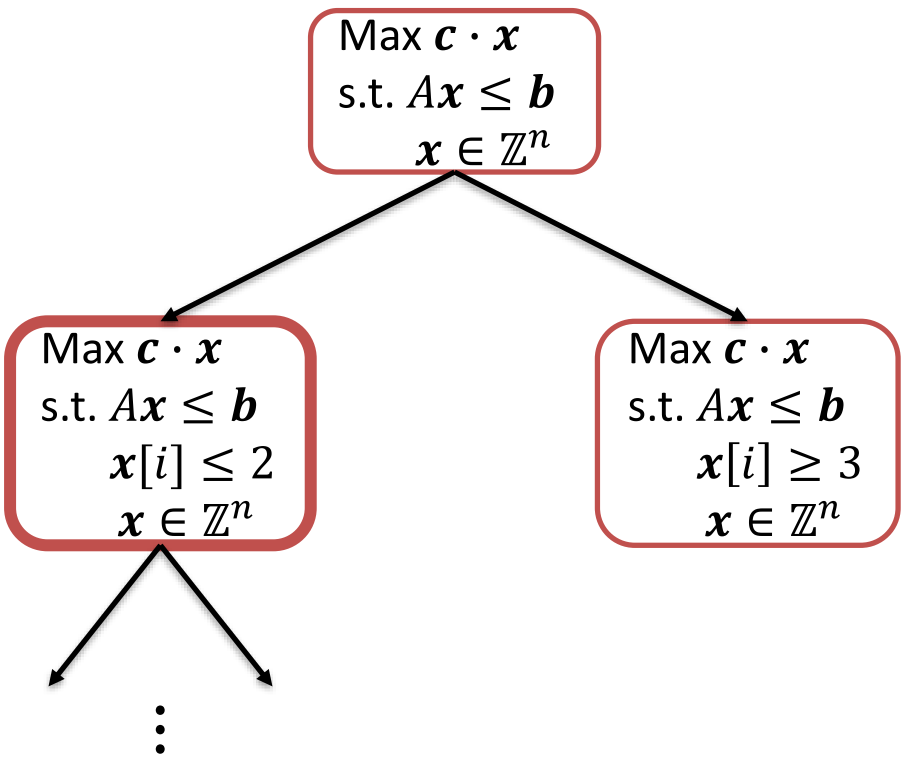
Branch-and-bound: pruning

- Prune subtrees if
  - LP relaxation at a node is integral, infeasible, or
  - (Bounding) LP optimal *worse* than best feasible integer solution found so far



Branch-and-bound: node selection

- At every stage, need to choose a leaf to explore further
- Variety of heuristics (e.g. *best-bound-first* chooses the node with the smallest LP objective)



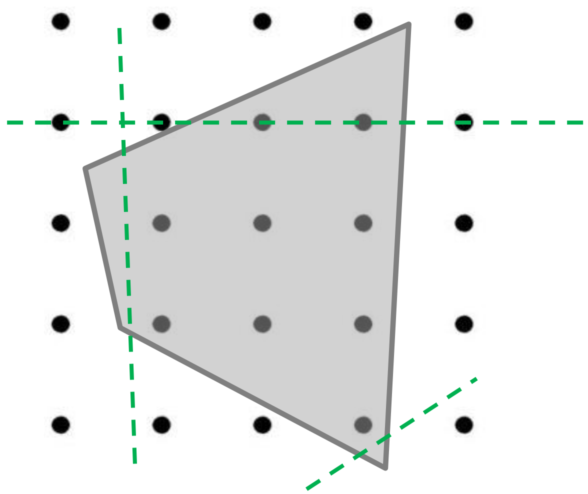
Branch-and-cut

- Branch-and-bound, but at each node may add *cutting planes*

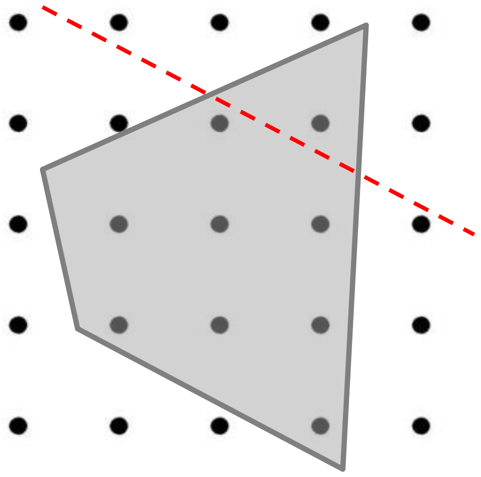
- Method of getting tighter LP relaxation bounds, and thus pruning subtrees sooner

Cutting planes

- Constraint $\boldsymbol{\alpha x} \leq \boldsymbol{\beta}$ is a *valid cutting plane* if it does not cut off any integer feasible points



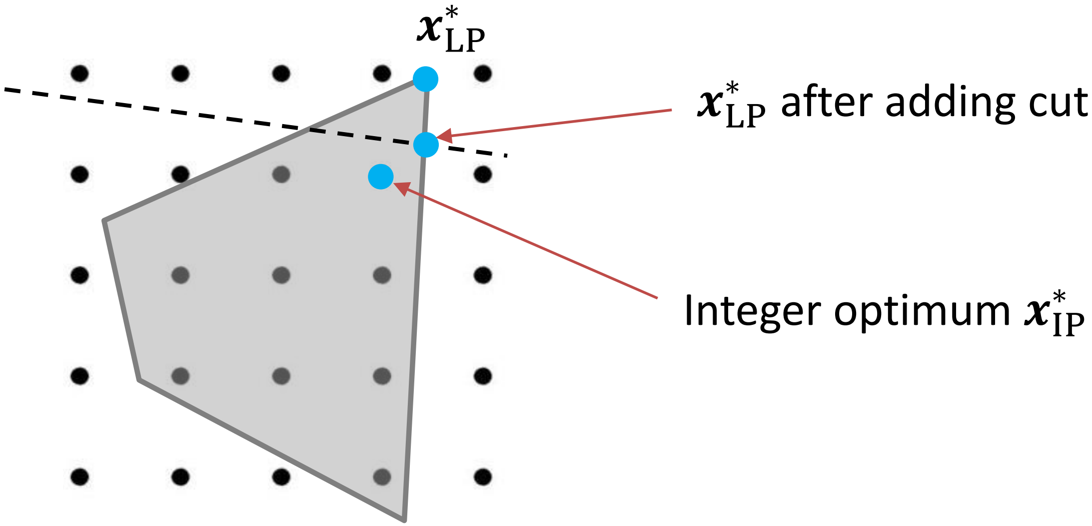
Valid cutting planes



An invalid cutting plane

Cutting planes

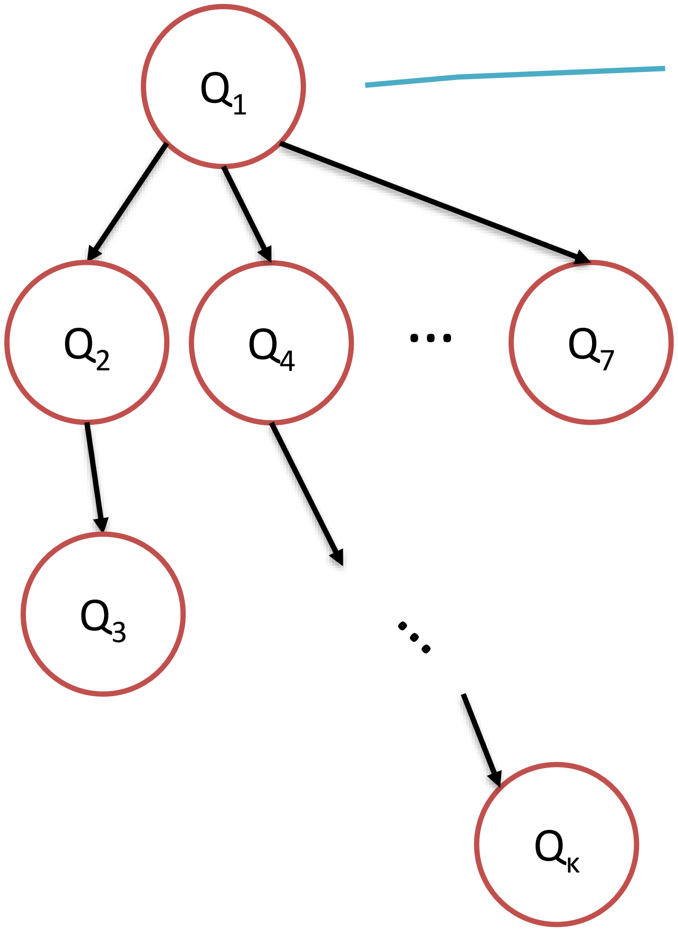
- If $\boldsymbol{\alpha x} \leq \beta$ is valid and separates the LP optimum, can speed up B&C by pruning nodes sooner



Tuning branch-and-cut

- Solvers like CPLEX, Gurobi have *numerous* parameters that control various aspects of the search (CPLEX has 170 page manual describing 172 parameters)

Abstracting away: tree search



- Select node Q that maximizes node selection rule nscore(T, Q)
  - Select action A that maximizes action score ascore(T, Q, A)
  - Either prune tree at Q, or add children
  - Continue until all nodes are pruned

Actions chosen using mixture of scoring rules:

$$\text{ascore} = \mu \cdot \text{ascore}_1 + (1 - \mu) \cdot \text{ascore}_2$$

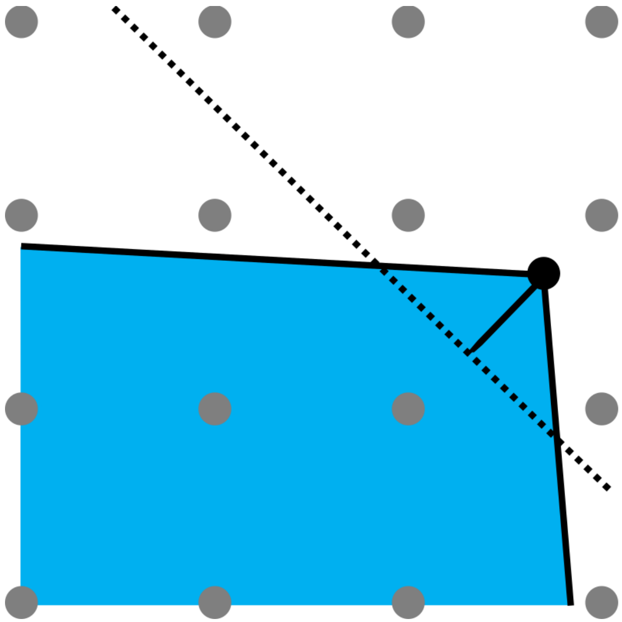
Nodes chosen using mixture of scoring rules:

$$\text{nscore} = \lambda \cdot \text{nscore}_1 + (1 - \lambda) \cdot \text{nscore}_2$$

Cut scoring rule example

Efficacy:

distance between cut and $\boldsymbol{x}_{\mathrm{LP}}^{*}$

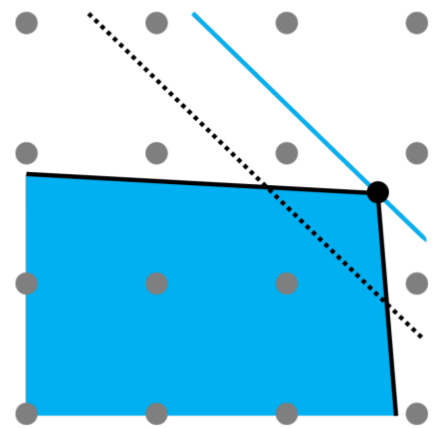


$$\mathrm{score}_1(\boldsymbol{\alpha}^T\boldsymbol{x} \leq \beta) = \frac{\boldsymbol{\alpha}\boldsymbol{x}_{\mathrm{LP}}^{*} - \beta}{\|\boldsymbol{\alpha}\|_2}$$

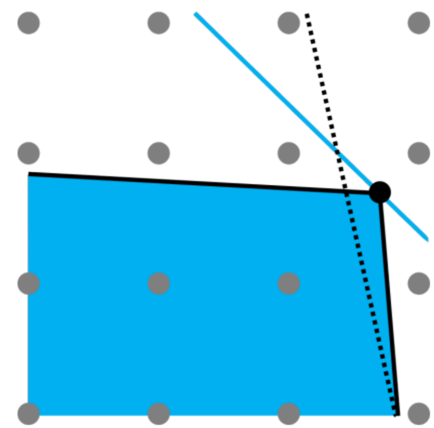
Cut scoring rule example

Parallelism:

angle between cut and objective



Better parallelism



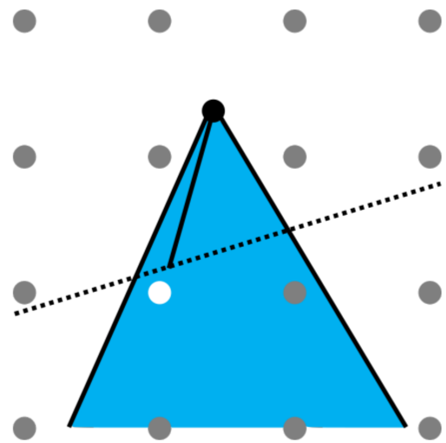
Worse parallelism

$$\mathrm{score}_2(\boldsymbol{\alpha}^T\boldsymbol{x} \le \beta) = \frac{|\boldsymbol{c}\boldsymbol{\alpha}|}{\|\boldsymbol{\alpha}\|_2\|\boldsymbol{c}\|_2}$$

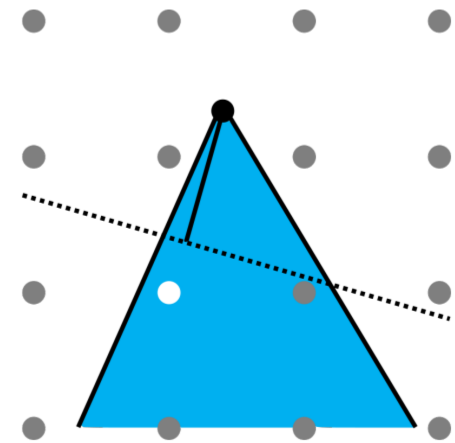
Cut scoring rule example

Directed cutoff:

distance between cut and $\boldsymbol{x}_{\mathrm{LP}}^*$, in direction of current best integer solution



Better directed cutoff



Worse directed cutoff

$$\mathrm{score}_3(\boldsymbol{\alpha}^T\boldsymbol{x} \le \beta) = \frac{\boldsymbol{\alpha}\boldsymbol{x}_{\mathrm{LP}}^* - \beta}{|\boldsymbol{\alpha}(\overline{\boldsymbol{x}} - \boldsymbol{x}_{\mathrm{LP}}^*)|} \cdot \|\overline{\boldsymbol{x}} - \boldsymbol{x}_{\mathrm{LP}}^*\|_2$$

Pathwise scoring rules

- All the previous scoring rules are *pathwise*: they only depend on the LP information accumulated along the path from the root to the node in question

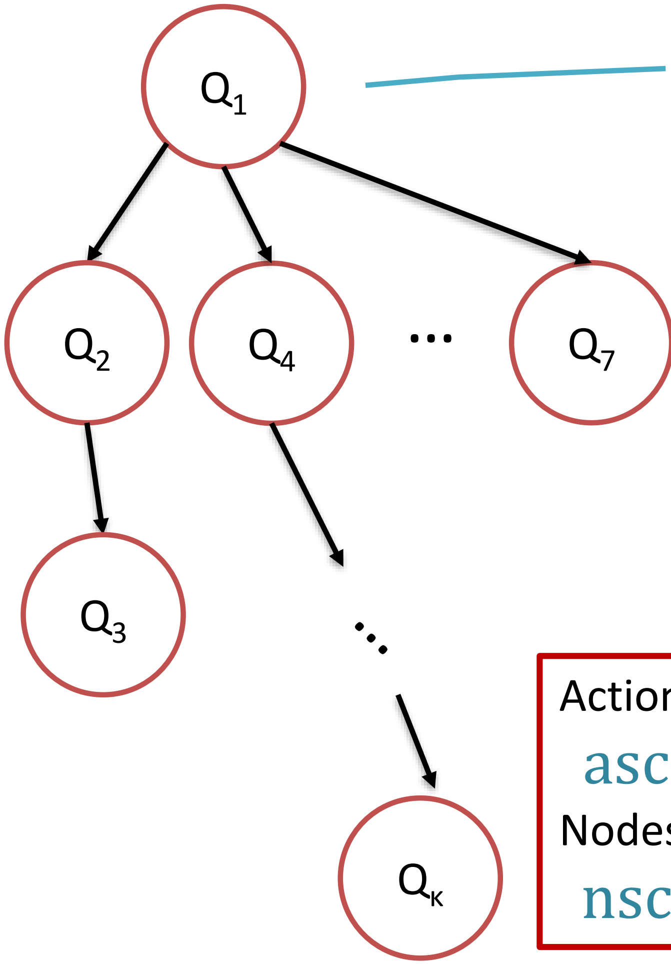
- Open source solver SCIP uses hard-coded mixture of scores to choose cuts

$$\frac{3}{5}\mathrm{score}_1 + \frac{1}{10}\mathrm{score}_2 + \frac{1}{2}\mathrm{score}_3 + \frac{1}{10}\mathrm{score}_4$$

Generalization guarantees for tree search and branch-and-cut

Distribution-dependent parameter selection of μ, λ

Parameterized tree search



- Select node Q that maximizes node selection rule nscore(T, Q)
  - Select action A that maximizes action score ascore(T, Q, A)
  - Either prune tree at Q, or add children
  - Continue until all nodes are pruned

Actions chosen using mixture of **pathwise** scoring rules:

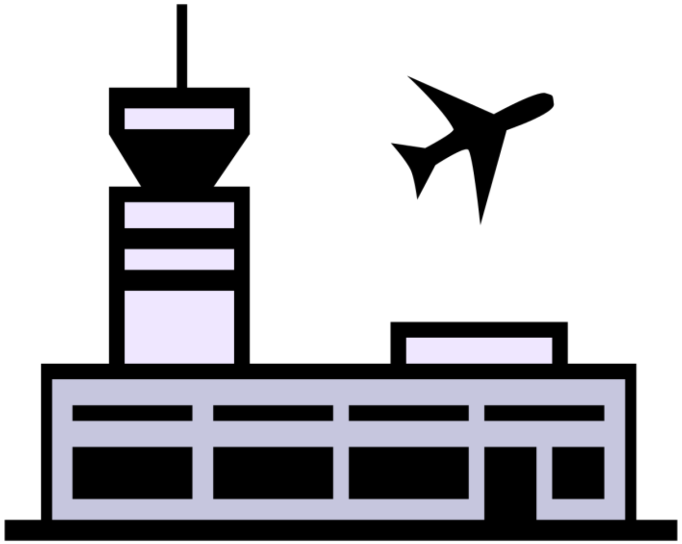
$$\text{ascore} = \boldsymbol{\mu} \cdot \text{ascore}_1 + (1 - \boldsymbol{\mu}) \cdot \text{ascore}_2$$

Nodes chosen using mixture of **pathwise** scoring rules:

$$\text{nscore} = \boldsymbol{\lambda} \cdot \text{nscore}_1 + (1 - \boldsymbol{\lambda}) \cdot \text{nscore}_2$$

Learning to tune tree search

Best parameters for airline-scheduling IPs…



…might not be useful for combinatorial-auction IPs solved by a sourcing firm

Learning to tune branch-and-cut

If a certain set of parameters yields small average branch-and-cut tree size over IP samples…

IP 1		IP N	
Max $\boldsymbol{c_1} \cdot \boldsymbol{x}$ s.t. $A_1\boldsymbol{x} \leq \boldsymbol{b_1}$ $\boldsymbol{x} \in \mathbb{Z}^n$	$\bullet\bullet\bullet$	Max $\boldsymbol{c}_N \cdot \boldsymbol{x}$ s.t. $A_N\boldsymbol{x} \leq \boldsymbol{b_N}$ $\boldsymbol{x} \in \mathbb{Z}^n$	$\sim D$

…is it likely to yield a small branch-and-cut tree on a fresh IP?

Max $\boldsymbol{c} \cdot \boldsymbol{x}$
s.t. $A\boldsymbol{x} \leq \boldsymbol{b}$
$\boldsymbol{x} \in \mathbb{Z}^n$ $\quad \sim D$

Sample complexity

- Q – domain of input root nodes to tree search
- $F = \{f_{\mu,\lambda} : Q \to \mathbf{R} \mid \mu, \lambda\}$ class of functions (e.g. tree size)
- Sample complexity $N_F(\varepsilon, \delta)$ is the minimum $N_0 \in \mathbf{N}$ such that for any $N \geq N_0$:

$$\Pr_{Q_1,\ldots,Q_N \sim D}\left(\sup_{f \in F}\left|\frac{1}{N}\sum_{i=1}^{N} f(Q_i) - \mathbf{E}_{Q\sim D}[f(Q)]\right| \leq \varepsilon\right) \geq 1 - \delta$$

for any distribution D on Q.

Sample complexity of tuning tree search

Theorem [BPSV CP'22]: For all μ, λ, the number of samples so that the difference between average training performance and expected performance when μ, λ is used to select actions and nodes throughout the tree is (whp) at most ε is

$$\tilde{O}\left(\frac{H^2}{\varepsilon^2}(\Delta^2 \log k + \Delta \log b)\right)$$

Δ = tree depth
k = tree branching factor
b = # actions available at each node
H = cap on size of tree

First guarantee that handles multiple critical aspects of branch-and-cut:
Node selection, branching, and cutting plane selection

Generalization guarantee for tree search

Theorem [BPSV CP'22]: For all μ, λ, difference between average training performance and expected performance when μ, λ is used to select actions and nodes throughout the tree is (whp)

$$\tilde{O}\left(H\sqrt{\frac{\Delta^2 \log k + \Delta \log b}{N}}\right)$$

Δ = tree depth
k = tree branching factor
b = # actions available at each node
H = cap on size of tree

Holds for any (unknown) distribution over tree-search problem instances

First guarantee that handles multiple critical aspects of branch-and-cut:
Node selection, branching, and cutting plane selection

Tree search guarantees

- Main challenge: performance functions (e.g. size of tree) are highly discontinuous
  - Miniscule change in parameters can lead to exponential difference in tree size
- We prove that parameterized tree search is structured
- Allows us to bound the *intrinsic complexity* (pseudo-dimension from learning theory) of the class of performance functions parameterized by (μ, λ), which implies our sample complexity bounds

Tree search structure

Theorem [BPSV CP'22]:

Fix path-wise node selection scores $\mathrm{nscore}_1, \mathrm{nscore}_2$ and path-wise action selection scores $\mathrm{ascore}_1, \mathrm{ascore}_2$, and the input node Q.

There are $\leq k^{\Delta(9+\Delta)} b^{\Delta}$ rectangles partitioning $[0,1]^2$ such that for any rectangle R, the node-selection score $\lambda \cdot \mathrm{nscore}_1 + (1-\lambda) \cdot \mathrm{nscore}_2$ and action selection score $\mu \cdot \mathrm{ascore}_1 + (1-\mu) \cdot \mathrm{ascore}_2$ result in the same tree for all $(\mu, \lambda) \in R$.

Δ = tree depth
k = tree branching factor

Back to branch-and-cut

- Our result implies polynomial bounds for:
  - Branching: single-variable, multi-variable, branching on general disjunctions with bounded coefficients,…
  - Cutting planes: cover cuts, clique cuts, any cuts derived from simplex tableau (Chvátal cuts, Gomory mixed integer cuts)
  - Allows node selection to be tuned simultaneously
- Prior work
  - [Balcan et al. ICML’18] studied single-variable branching with pathwise scoring rules (our result recovers theirs)
  - [Balcan, **Prasad,** Vitercik, Sandholm NeurIPS’21] studied Chvátal cuts, but obtained a much weaker bound when these are applied throughout the tree due to not using pathwise assumption

Knapsack cover cuts – an experiment

- Set of items N, item $i \in N$ has value $p_i \geq 0$ and weight $w_i \geq 0$
- Set of knapsacks K, knapsack $k \in K$ has capacity $W_k \geq 0$
- *Goal:* find feasible packing of maximum weight

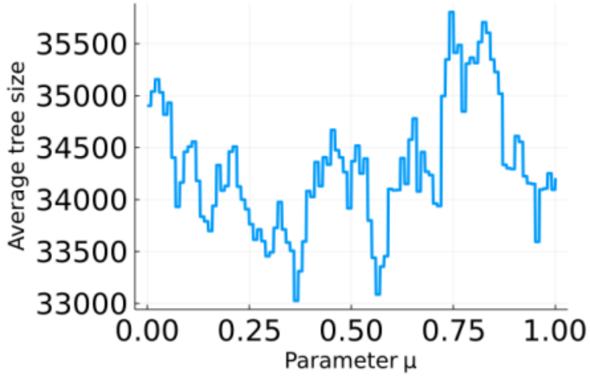
$$
\begin{array}{lll}
\text{maximize} & \Sigma_{i \in N} \Sigma_{k \in K} p_i x_{k,i} & \\
\text{subject to} & \Sigma_{i \in N} w_i x_{k,i} \leq W_k & \forall k \in K \\
 & \Sigma_{k \in K} x_{k,i} \leq 1 & \forall i \in N \\
 & x_{k,i} \in \{0,1\} & \forall i \in N, k \in K
\end{array}
$$

Knapsack cover cuts – an experiment

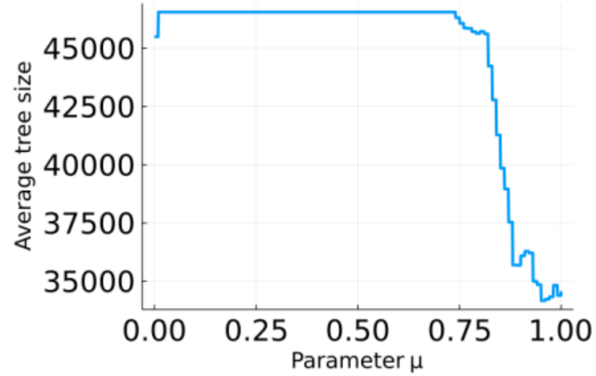
- Cover cut for knapsack k: if $w_1 + w_2 + w_3 \geq W_k$ (items 1, 2, 3 are jointly too heavy for knapsack k), can enforce the constraint $x_{k,1} + x_{k,2} + x_{k,3} \leq 2$
- We tune convex combinations of cut scoring rules to control the addition of cover cuts* throughout the branch-and-cut tree

*actually a special kind of cover cut: *extended minimal cover cuts*

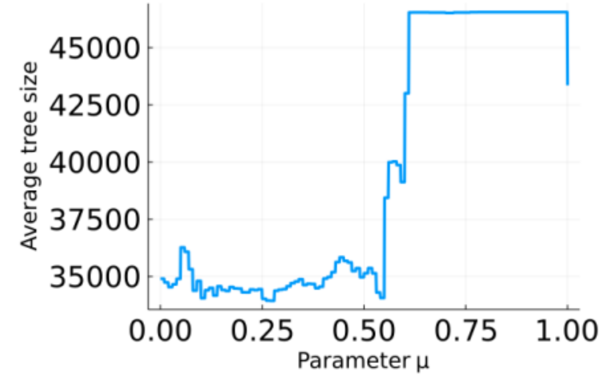
Knapsack cover cuts – an experiment



(a) $\mu \cdot \mathtt{E} + (1 - \mu) \cdot \mathtt{P}$.

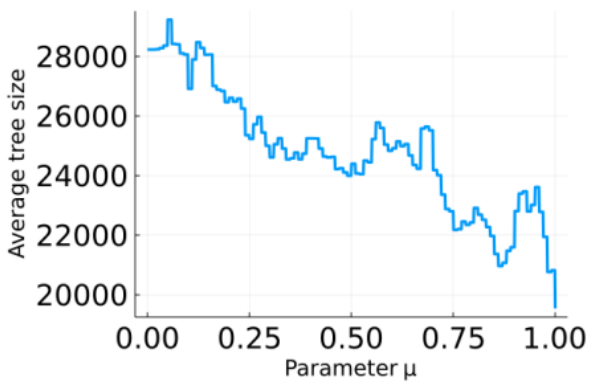


(b) $\mu \cdot \mathtt{E} + (1 - \mu) \cdot \mathtt{D}$.

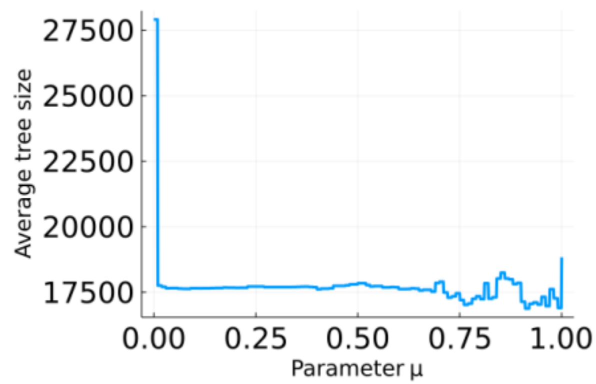


(c) $\mu \cdot \mathtt{D} + (1 - \mu) \cdot \mathtt{P}$.

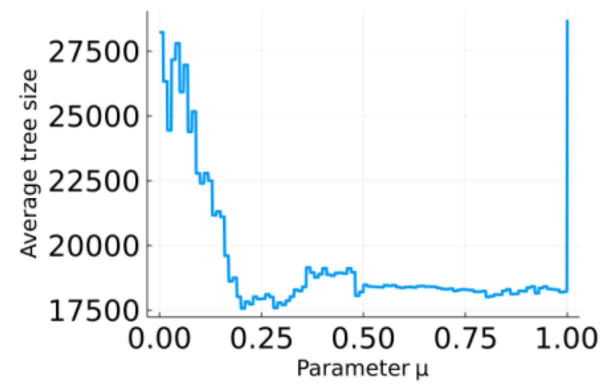
Figure 1 Chvátal distribution with 35 items and 2 knapsacks.



(a) $\mu \cdot \mathtt{E} + (1 - \mu) \cdot \mathtt{P}$.



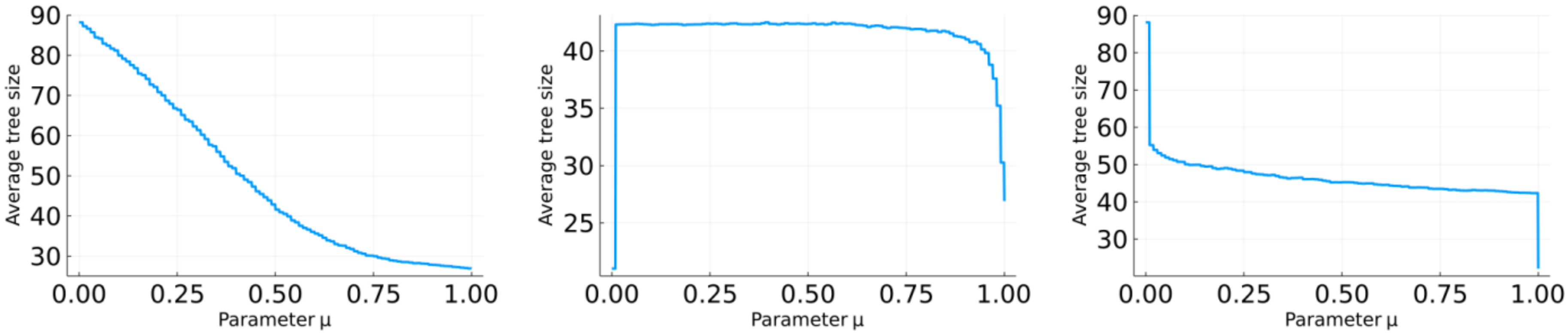
(b) $\mu \cdot \mathtt{E} + (1 - \mu) \cdot \mathtt{D}$.



(c) $\mu \cdot \mathtt{D} + (1 - \mu) \cdot \mathtt{P}$.

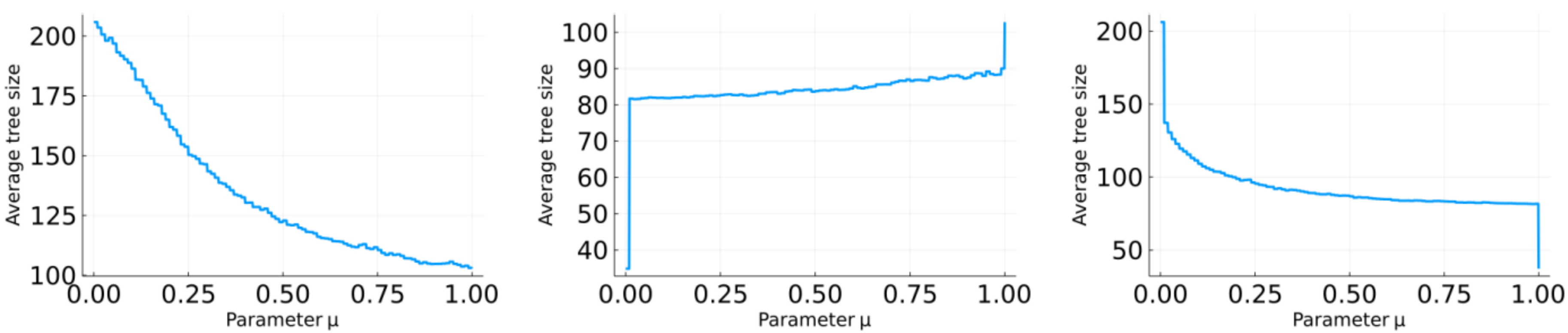
Figure 2 Chvátal distribution with 35 items and 3 knapsacks.

Knapsack cover cuts – an experiment



(a) $\mu \cdot \mathtt{E} + (1-\mu) \cdot \mathtt{P}$. **(b)** $\mu \cdot \mathtt{E} + (1-\mu) \cdot \mathtt{D}$. **(c)** $\mu \cdot \mathtt{D} + (1-\mu) \cdot \mathtt{P}$.

Figure 3 Reverse Chvátal distribution with 100 items and 10 knapsacks.



(a) $\mu \cdot \mathtt{E} + (1-\mu) \cdot \mathtt{P}$. **(b)** $\mu \cdot \mathtt{E} + (1-\mu) \cdot \mathtt{D}$. **(c)** $\mu \cdot \mathtt{D} + (1-\mu) \cdot \mathtt{P}$.

Figure 4 Reverse Chvátal distribution with 100 items and 15 knapsacks.