

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:

$$\begin{aligned} \text{Max } & c \cdot x \\ \text{s.t. } & Ax \leq b \\ & x \in \mathbb{Z}^n \end{aligned}$$

- One of the most useful and widely applicable optimization techniques



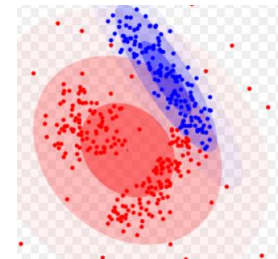
Scheduling



Routing



Combinatorial auctions



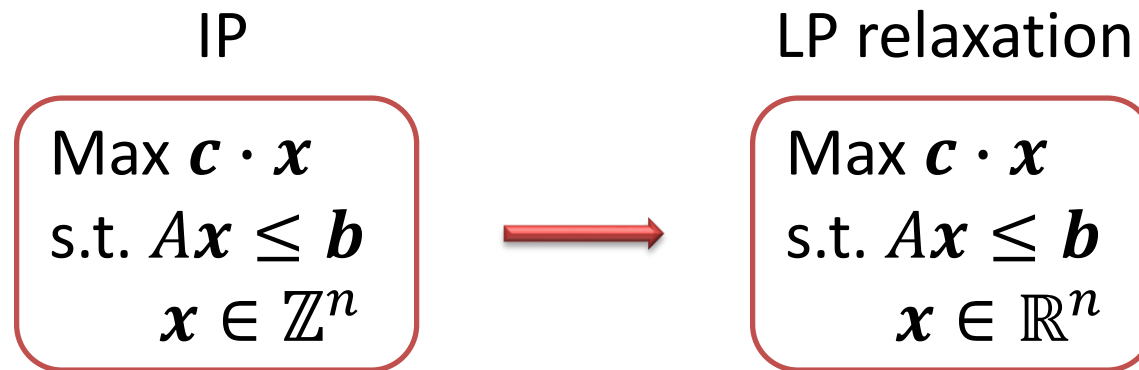
Clustering

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

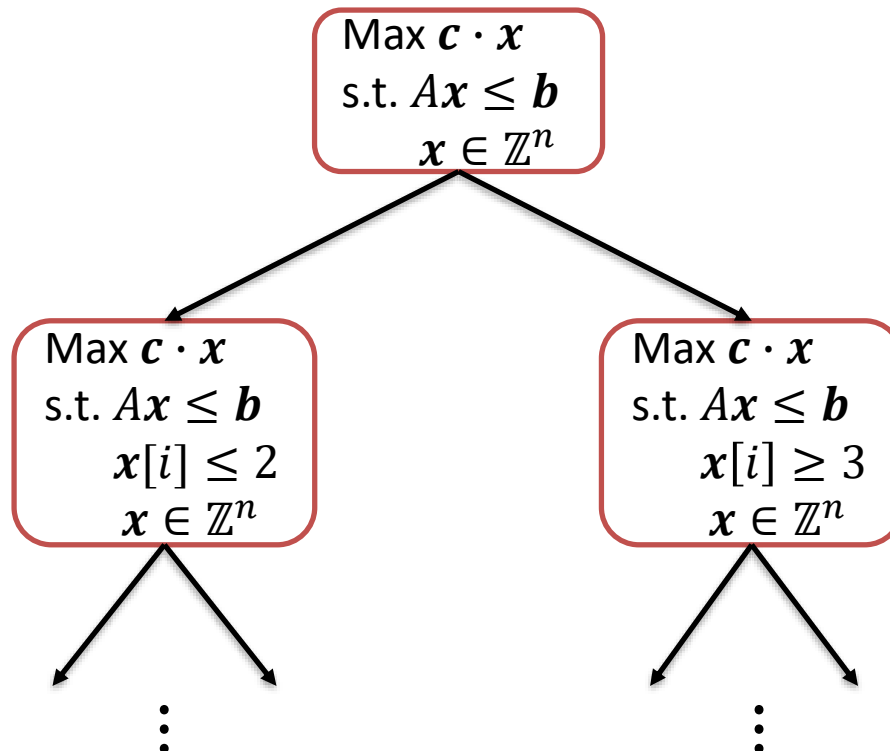
Branch-and-bound

- 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



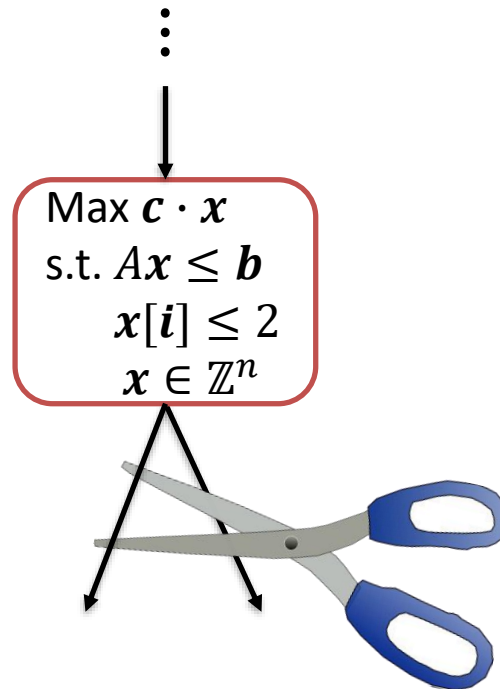
Branch-and-bound: branching

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



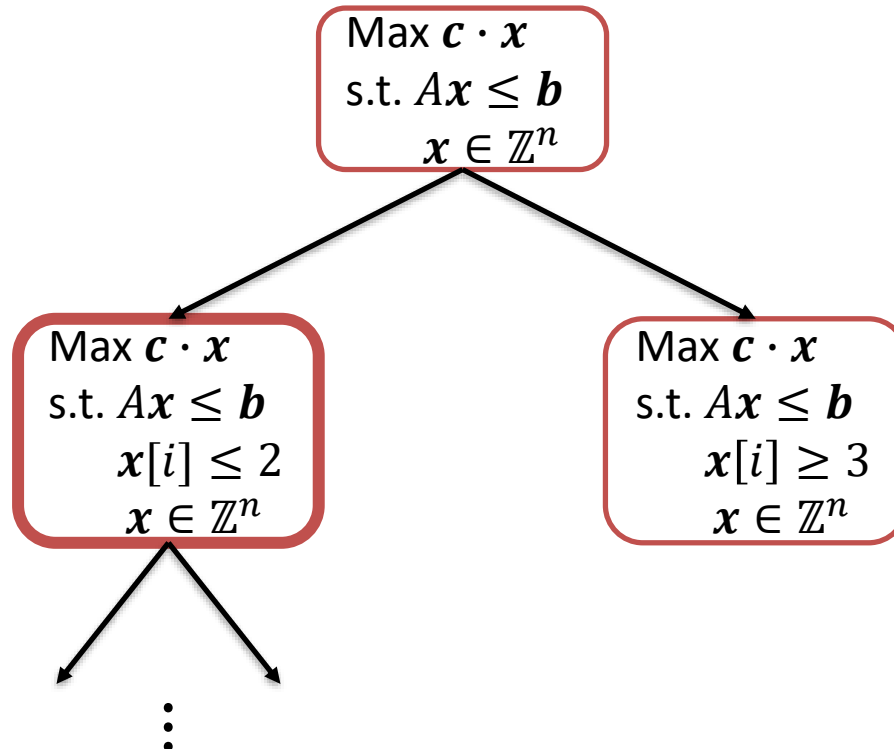
Branch-and-bound: pruning

- 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)

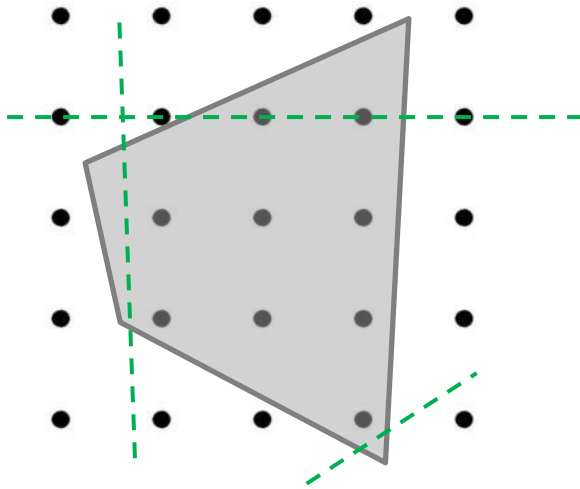


Branch-and-cut

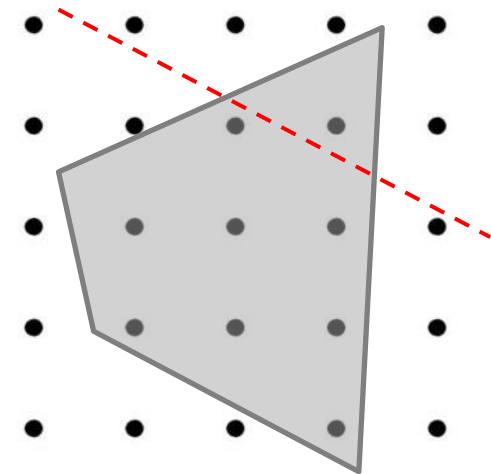
- Branch-and-bound, but at each node may add *cutting planes*
- Method of getting tighter LP relaxation bounds, and thus pruning subtrees sooner

Cutting planes

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



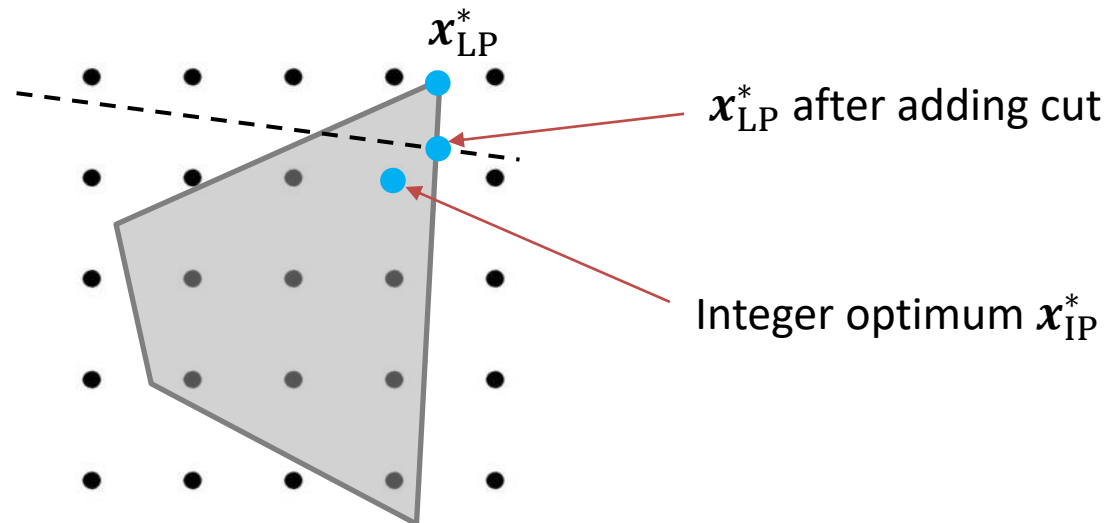
Valid cutting planes



An invalid cutting plane

Cutting planes

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

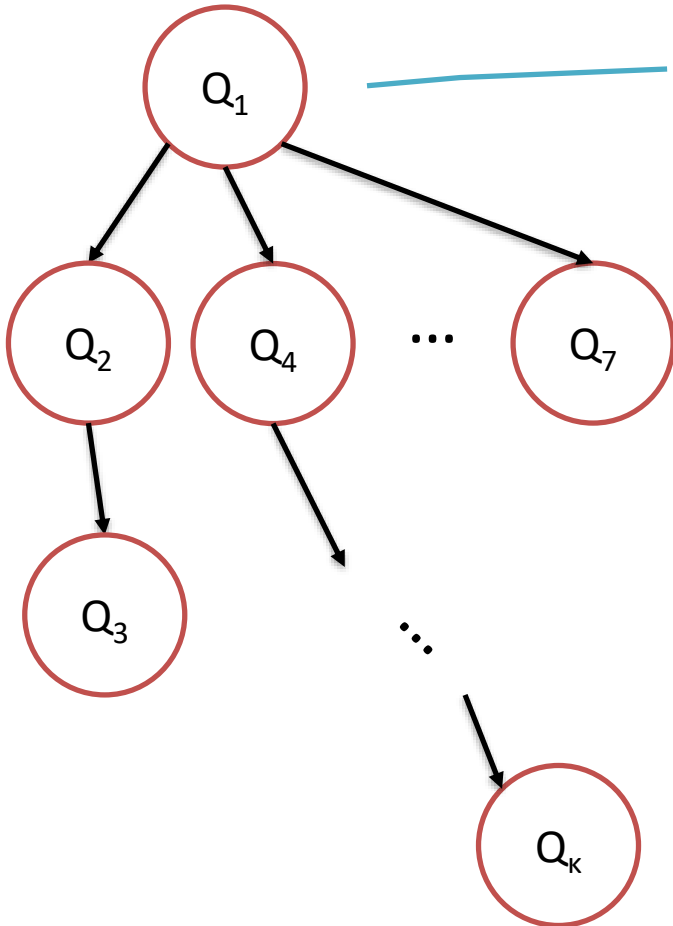


Tuning branch-and-cut

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

CPX_PARAM_NODEFILEIND 100	CPX_PARAM_TRELIM 160	CPX_PARAM_RANDOMSEED 130	CPXPARAM_MIP_Pool_RelGap 148	CPX_PARAM_FLOWCOVERS 70	CPX_PARAM_BRDIR 39
CPX_PARAM_NODELIM 101	CPX_PARAM_TUNINGDETTILIM 160	CPX_PARAM_REDUCE 131	CPXPARAM_MIP_Pool_Replace 151	CPX_PARAM_FLOWPATHS 71	CPX_PARAM_BITTOL 40
CPX_PARAM_NODESEL 102	CPX_PARAM_TUNINGDISPLAY 162	CPX_PARAM_REINV 131	CPXPARAM_MIP_Strategy_Branch 39	CPX_PARAM_FPHEUR 72	CPX_PARAM_CALCOCPDUALS 41
CPX_PARAM_NUMERICALEMPHASIS 102	CPXPARAM_TUNINGMEASURE 163	CPX_PARAM_RELAXPREIND 132	CPXPARAM_MIP_Strategy_MIQCPStrat 93	CPX_PARAM_FRACCAND 73	CPX_PARAM_CLIQUES 42
CPX_PARAM_NZREADLIM 103	CPX_PARAM_TUNINGREPEAT 164	CPX_PARAM_RELOBJDIF 133	CPXPARAM_MIP_Strategy_StartAlgorithm 139	CPX_PARAM_FRACCCUTS 73	CPX_PARAM_CLOCKTYPE 43
CPX_PARAM_OBJDIF 104	CPXPARAM_TUNINGTILIM 165	CPX_PARAM_REPAIRTRIES 133	CPXPARAM_MIP_Strategy_VariableSelect 166	CPX_PARAM_FRACPASS 74	CPX_PARAM_CLONELOG 43
CPX_PARAM_OBJLLIM 105	CPX_PARAM_VARSEL 166	CPX_PARAM_REPEATPRESOLVE 134	CPXPARAM_MIP_SubMIP_NodeLimit 155	CPX_PARAM_GUBCOVERS 75	CPX_PARAM_COEREDIND 44
CPX_PARAM_OBJULIM 105	CPX_PARAM_WORKDIR 167	CPX_PARAM_RINSHEUR 135	CPXPARAM_OptimalityTarget 106	CPX_PARAM_HEURFREQ 76	CPX_PARAM_COLREADLIM 45
CPX_PARAM_PARALLELMODE 108	CPX_PARAM_WORKMEM 168	CPX_PARAM_RLT 136	CPXPARAM_Output_WriteLevel 169	CPX_PARAM_IMPLBD 76	CPX_PARAM_CONFLICTDISPLAY 46
CPX_PARAM_PERIND 110	CPX_PARAM_WRITELEVEL 169	CPX_PARAM_ROWREADLIM 141	CPXPARAM_Preprocessing_Aggregator 19	CPX_PARAM_INTSOLFILEPREFIX 78	CPX_PARAM_COVERS 47
CPX_PARAM_PERLIM 111	CPX_PARAM_ZEROHALFCUTS 170	CPX_PARAM_SCAIND 142	CPXPARAM_Preprocessing_Fill 19	CPX_PARAM_INTSOLLIM 79	CPX_PARAM_CUPUMASK 48
CPX_PARAM_POLISHAFTERDETIME 111	CPXPARAM_Benders_Strategy 30	CPX_PARAM_SCRIND 143	CPXPARAM_Preprocessing_Linear 120	CPX_PARAM_ITLIM 80	CPX_PARAM_CUTLO 51
CPX_PARAM_POLISHAFTEREPAGAP 112	CPXPARAM_Benders_Tolerances_feasibilitycut 35	CPX_PARAM_SIFTALG 143	CPXPARAM_Preprocessing_Reduce 131	CPX_PARAM_LANDPCUTS 82	CPX_PARAM_CUTPASS 52
CPX_PARAM_POLISHAFTEREPGAP 113	CPXPARAM_Benders_Tolerances_optimalitycut 36	CPX_PARAM_SIFTDISPLAY 144	CPXPARAM_Preprocessing_Symmetry 156	CPX_PARAM_LBHEUR 81	CPX_PARAM_CUTSFACOR 52
CPX_PARAM_POLISHAFTERINTSOL 114	CPXPARAM_Conflict_Algorithm 46	CPX_PARAM_SIFITLIM 145	CPXPARAM_Read_DataCheck 54	CPX_PARAM_LPMETHOD 136	CPX_PARAM_CUTUP 53
CPX_PARAM_POLISHAFTERNODE 115	CPXPARAM_CPUmask 48	CPX_PARAM_SIMDISPLAY 145	CPXPARAM_Read_Scale 142	CPX_PARAM_MFCUTS 82	CPX_PARAM_DATACHECK 54
CPX_PARAM_POLISHAFTERTIME 116	CPXPARAM_DistMIP_Rampup_Duration 128	CPX_PARAM_SINGLIM 146	CPXPARAM_ScreenOutput 143	CPX_PARAM_MEMORYEMPHASIS 83	CPX_PARAM_DEPIND 55
CPX_PARAM_POLISHTIME (deprecated) 116	CPXPARAM_LPMethod 136	CPX_PARAM_SOLNPOOLAGAP 146	CPXPARAM_Sifting_Algorithm 143	CPX_PARAM_MIPCBREDLP 84	CPX_PARAM_DETILIM 56
CPX_PARAM_POPULATELIM 117	CPXPARAM_MIP_Cuts_BQP 38	CPX_PARAM_SOLNPOOLCAPACITY 147	CPXPARAM_Sifting_Display 144	CPX_PARAM_MIPDISPLAY 85	CPX_PARAM_DISJCUTS 57
CPX_PARAM_PPRIND 118	CPXPARAM_MIP_Cuts_Locallmplied 77	CPX_PARAM_SOLNPOOLGAP 148	CPXPARAM_Sifting_Iterations 145	CPX_PARAM_MIPEMPHASIS 87	CPX_PARAM_DIVETYPE 58
CPX_PARAM_PREDUAL 119	CPXPARAM_MIP_Cuts_RLT 136	CPX_PARAM_SOLNPOOLINTENSITY 149	CPXPARAM_Simplex_Display 145	CPX_PARAM_MIPINTERVAL 88	CPX_PARAM_EACHCUTLIM 60
CPX_PARAM_PREIND 120	CPXPARAM_MIP_Cuts_ZeroHalfCut 170	CPX_PARAM_SOLNPOOLREPLACE 151	CPXPARAM_Simplex_Limits_Singularity 146	CPX_PARAM_MIPORDIND 90	CPX_PARAM_EPAPG 61
CPX_PARAM_PRELINEAR 120	CPXPARAM_MIP_Limits_CutsFactor 52	CPX_PARAM_SOLUTIONTARGET deprecated: see	CPXPARAM_SolutionType 152	CPX_PARAM_MIPORPTYPE 91	CPX_PARAM_EPINT 62
CPX_PARAM_PREPASS 121	CPXPARAM_MIP_Limits_RampupTimeLimit 127	CPXPARAM_SOLUTIONTYPE 152	CPXPARAM_Threads 157	CPX_PARAM_MIPSEARCH 92	CPX_PARAM_EPMRK 64
CPX_PARAM_PRESLVND 122	CPXPARAM_MIP_Limits_Solutions 79	CPXPARAM_StartALG 139	CPXPARAM_TimeLimit 159	CPX_PARAM_MIQCPSTRAT 93	CPX_PARAM_EPOPT 65
CPX_PARAM_PRICECLIM 123	CPXPARAM_MIP_Limits_StrongCand 154	CPX_PARAM_STRONGCANDLIM 154	CPXPARAM_Tune_DetTimeLimit 160	CPX_PARAM_MIRDCUTS 94	CPX_PARAM_EPPER 65
CPX_PARAM_PROBE 123	CPXPARAM_MIP_Limits_Stronglt 154	CPX_PARAM_STRONGTILIM 154	CPXPARAM_Tune_Display 162	CPX_PARAM_MPSLONGNUM 94	CPX_PARAM_EPRELAX 66
CPX_PARAM_PROBEDETTIME 124	CPXPARAM_MIP_Limits_TreeMemory 160	CPX_PARAM_STRONGTILIM 154	CPXPARAM_Tune_Measure 163	CPX_PARAM_NETDISPLAY 95	CPX_PARAM_EPRESHT 67
CPX_PARAM_PROBETIME 124	CPXPARAM_MIP_OrderType 91	CPX_PARAM_SUBALG 99	CPXPARAM_Tune_Repeat 164	CPX_PARAM_NETDIND 96	CPX_PARAM_EPRESHTMODE 68
CPX_PARAM_QPMAKEPSDIND 125	CPXPARAM_MIP_OrderType 91	CPX_PARAM_SUBMIPNODELIMIT 155	CPXPARAM_Tune_Timelimit 165	CPX_PARAM_NETFIND 97	CPX_PARAM_FILEENCODING 69
CPX_PARAM_QPMETHOD 138	CPXPARAM_MIP_Pool_AbsGap 146	CPX_PARAM_SYMMEYTRY 156	CPXPARAM_WorkDir 167	CPX_PARAM_NETITLIM 98	
CPX_PARAM_QPNZREADLIM 126	CPXPARAM_MIP_Pool_Capacity 147	CPX_PARAM_THREADS 157	CPXPARAM_WorkMem 168	CPX_PARAM_NETPPRIND 98	
	CPXPARAM_MIP_Pool_Intensity 149	CPX_PARAM_TILIM 159	Craind 50		

Abstracting away: tree search



- Select node Q that maximizes node selection rule $n\text{score}(T, Q)$
 - Select action A that maximizes action score $a\text{score}(T, Q, A)$
 - Either prune tree at Q , or add children
 - Continue until all nodes are pruned

Actions chosen using mixture of scoring rules:

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

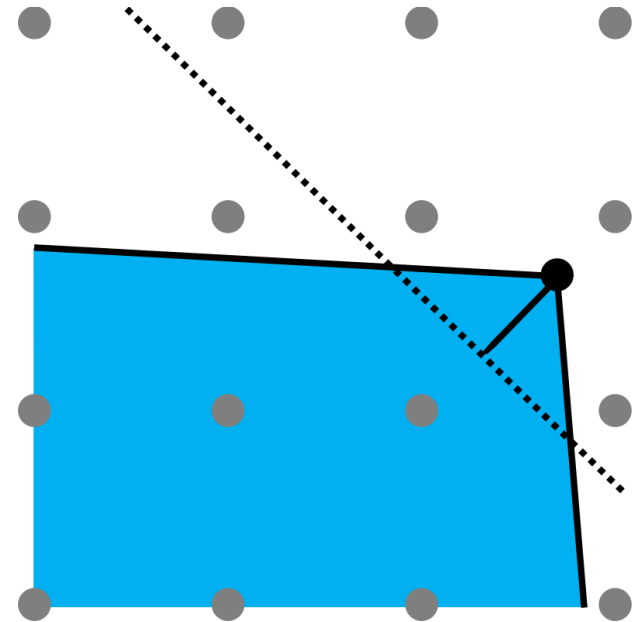
Nodes chosen using mixture of scoring rules:

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

Cut scoring rule example

Efficacy:

distance between cut
and x_{LP}^*

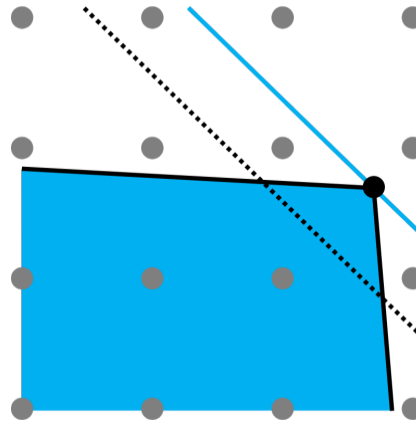


$$\text{score}_1(\alpha^T x \leq \beta) = \frac{\alpha x_{LP}^* - \beta}{\|\alpha\|_2}$$

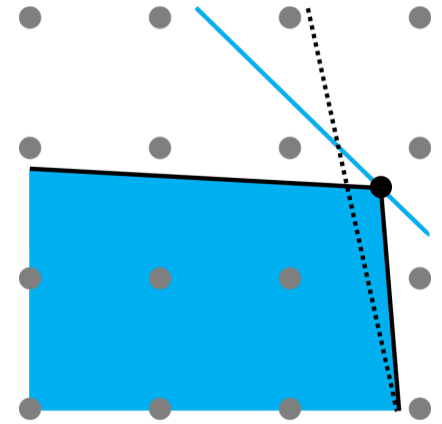
Cut scoring rule example

Parallelism:

angle between cut
and objective



Better parallelism



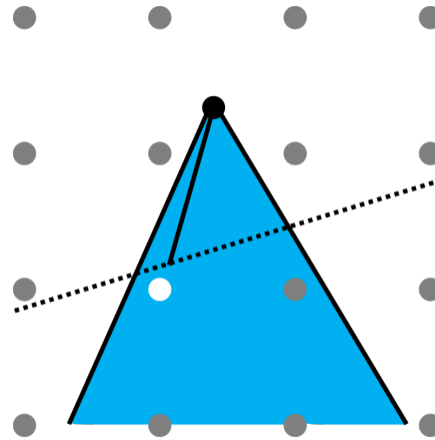
Worse parallelism

$$\text{score}_2(\alpha^T x \leq \beta) = \frac{|c\alpha|}{\|\alpha\|_2 \|\mathbf{c}\|_2}$$

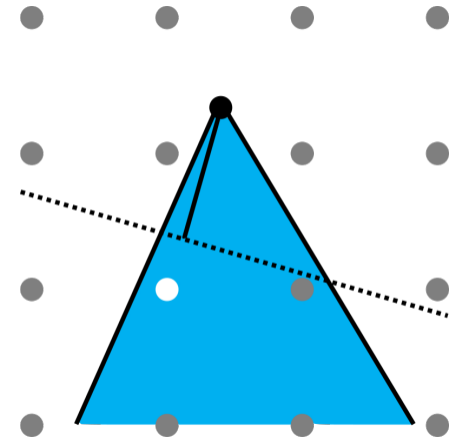
Cut scoring rule example

Directed cutoff:

distance between cut and x_{LP}^* , in direction of current best integer solution



Better directed cutoff



Worse directed cutoff

$$\text{score}_3(\alpha^T x \leq \beta) = \frac{\alpha x_{LP}^* - \beta}{|\alpha(\bar{x} - x_{LP}^*)|} \cdot \|\bar{x} - x_{LP}^*\|_2$$

Pathwise scoring rules

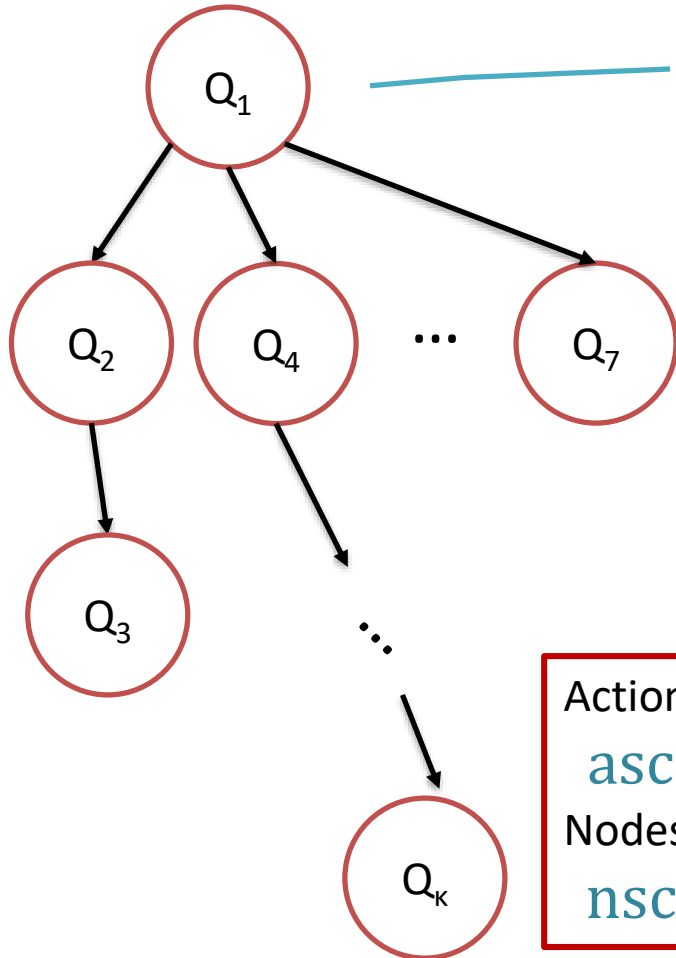
- All the previous scoring rules are *pathwise*: they only depend on the LP information accumulated along the path from the root to the node in question
- Open source solver SCIP uses hard-coded mixture of scores to choose cuts

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

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 $n\text{score}(T, Q)$
 - Select action A that maximizes action score $a\text{score}(T, Q, A)$
 - Either prune tree at Q , or add children
 - Continue until all nodes are pruned

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

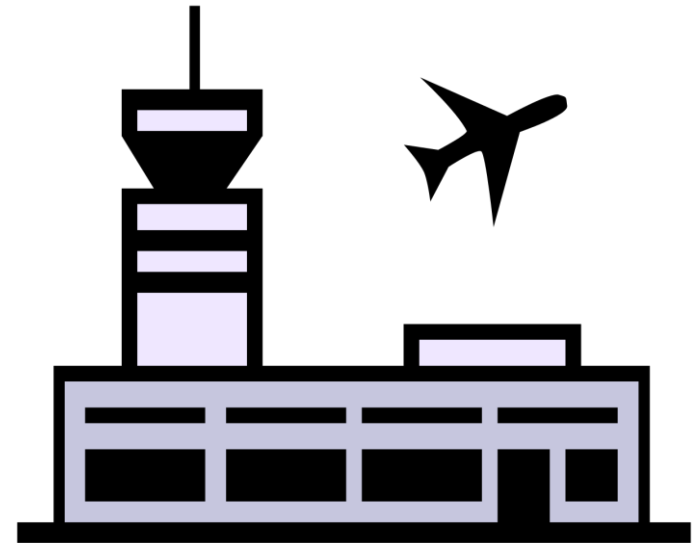
$$a\text{score} = \mu \cdot a\text{score}_1 + (1 - \mu) \cdot a\text{score}_2$$

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

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

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...

$$\begin{array}{ccc} \begin{array}{l} \text{Max } \mathbf{c}_1 \cdot \mathbf{x} \\ \text{s.t. } A_1 \mathbf{x} \leq \mathbf{b}_1 \\ \mathbf{x} \in \mathbb{Z}^n \end{array} & \dots & \begin{array}{l} \text{Max } \mathbf{c}_N \cdot \mathbf{x} \\ \text{s.t. } A_N \mathbf{x} \leq \mathbf{b}_N \\ \mathbf{x} \in \mathbb{Z}^n \end{array} \sim D \\ \text{IP 1} & & \text{IP N} \end{array}$$

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

$$\begin{array}{l} \text{Max } \mathbf{c} \cdot \mathbf{x} \\ \text{s.t. } A \mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \in \mathbb{Z}^n \end{array} \sim D$$

Sample complexity

- Q – domain of input root nodes to tree search
- $F = \{f_{\mu,\lambda}: Q \rightarrow \mathbf{R} \mid \mu, \lambda\}$ 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 $N \geq N_0$:

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

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 $\text{nscore}_1, \text{nscore}_2$ and path-wise action selection scores $\text{ascore}_1, \text{ascore}_2$, 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 $\lambda \cdot \text{nscore}_1 + (1 - \lambda) \cdot \text{nscore}_2$ and action selection score $\mu \cdot \text{ascore}_1 + (1 - \mu) \cdot \text{ascore}_2$ 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

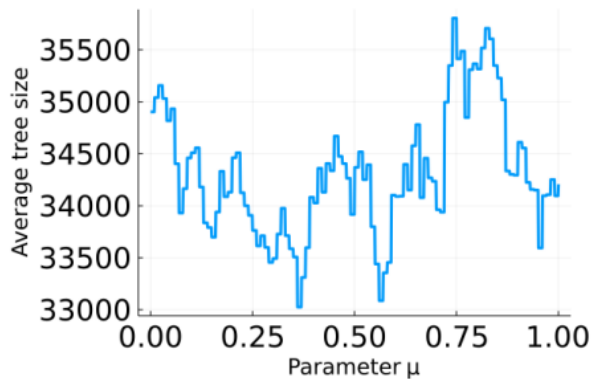
$$\begin{array}{ll} \text{maximize} & \sum_{i \in N} \sum_{k \in K} p_i x_{k,i} \\ \text{subject to} & \sum_{i \in N} w_i x_{k,i} \leq W_k \quad \forall k \in K \\ & \sum_{k \in K} x_{k,i} \leq 1 \quad \forall i \in N \\ & x_{k,i} \in \{0,1\} \quad \forall i \in N, k \in K \end{array}$$

Knapsack cover cuts – an experiment

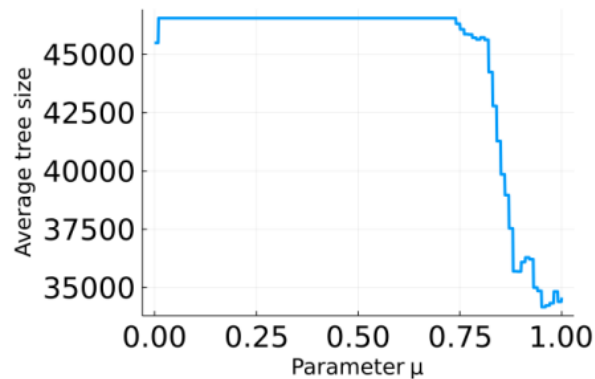
- Cover cut for knapsack k : if $w_1 + w_2 + w_3 \geq W_k$ (items 1, 2, 3 are jointly too heavy for knapsack k), can enforce the constraint $x_{k,1} + x_{k,2} + x_{k,3} \leq 2$
- We tune convex combinations of cut scoring rules to control the addition of cover cuts* throughout the branch-and-cut tree

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

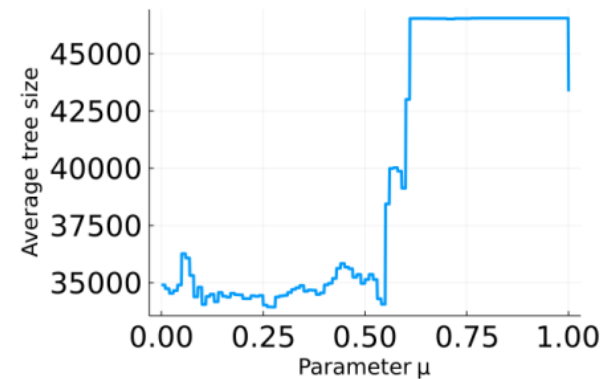
Knapsack cover cuts – an experiment



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

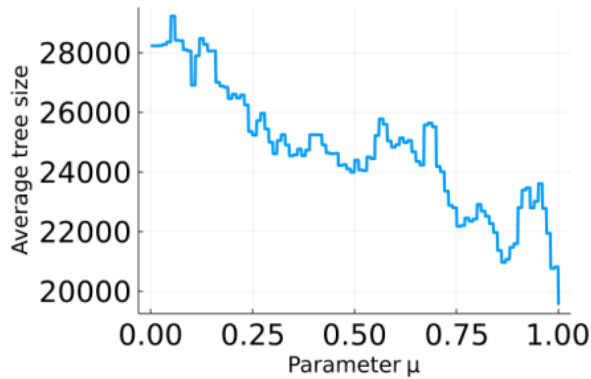


(b) $\mu \cdot E + (1 - \mu) \cdot D$.

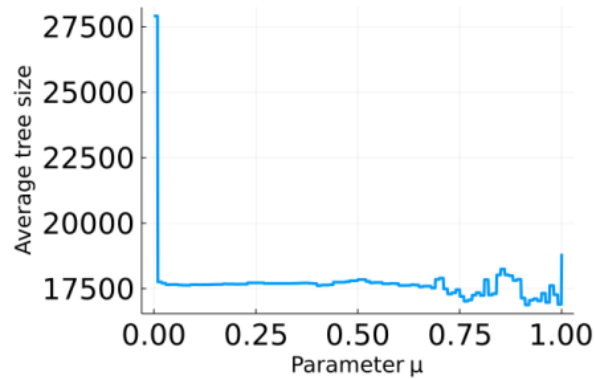


(c) $\mu \cdot D + (1 - \mu) \cdot P$.

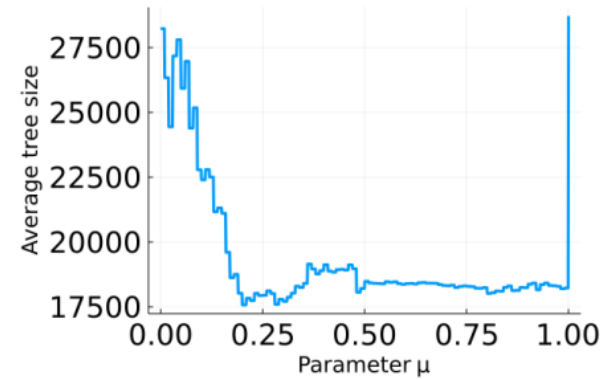
■ **Figure 1** Chvátal distribution with 35 items and 2 knapsacks.



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



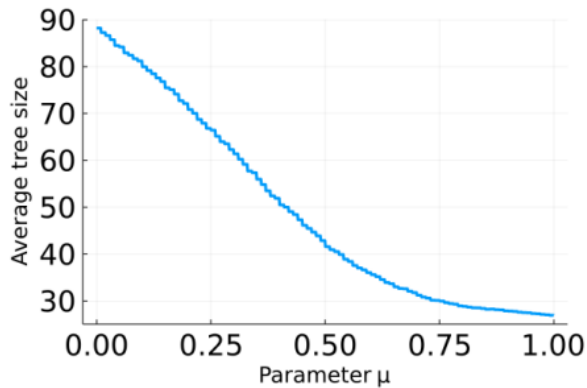
(b) $\mu \cdot E + (1 - \mu) \cdot D$.



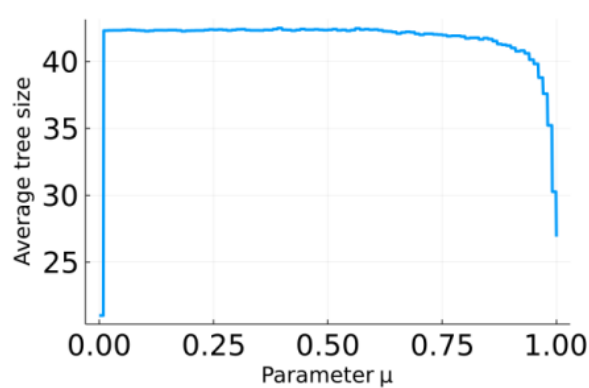
(c) $\mu \cdot D + (1 - \mu) \cdot P$.

■ **Figure 2** Chvátal distribution with 35 items and 3 knapsacks.

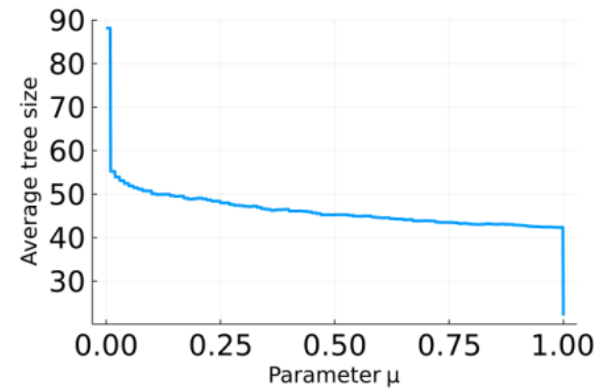
Knapsack cover cuts – an experiment



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

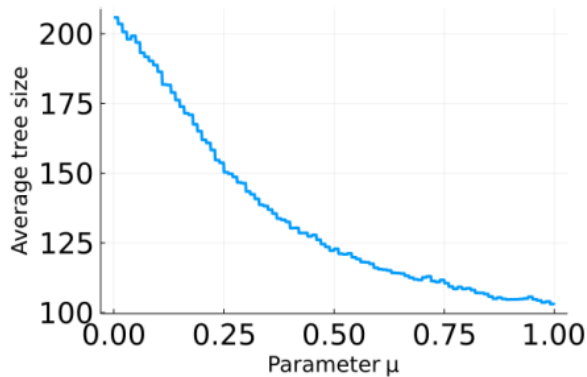


(b) $\mu \cdot E + (1 - \mu) \cdot D.$

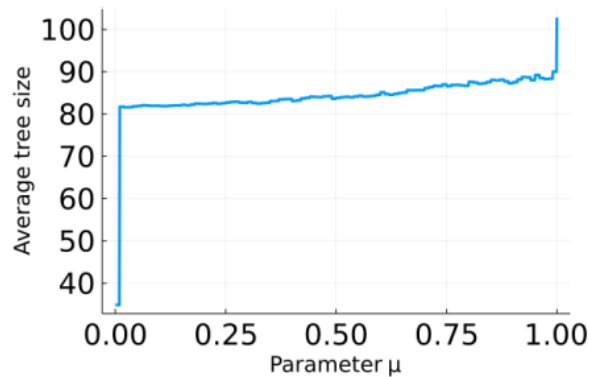


(c) $\mu \cdot D + (1 - \mu) \cdot P.$

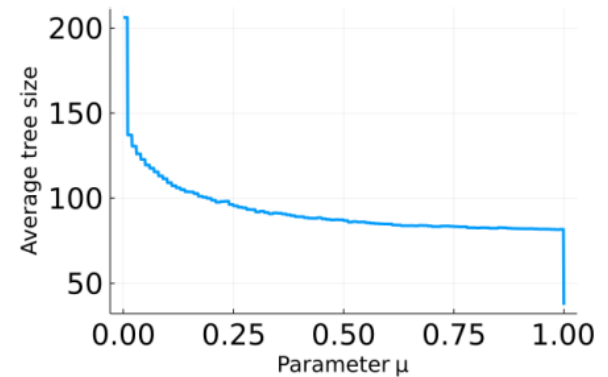
■ **Figure 3** Reverse Chvátal distribution with 100 items and 10 knapsacks.



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



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



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

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