Combinatorial Search Algorithms as Rational Agents

Wheeler Ruml
Palo Alto Research Center
ruml@parc.com

Motivation

Research goal: "What algorithm to run?"

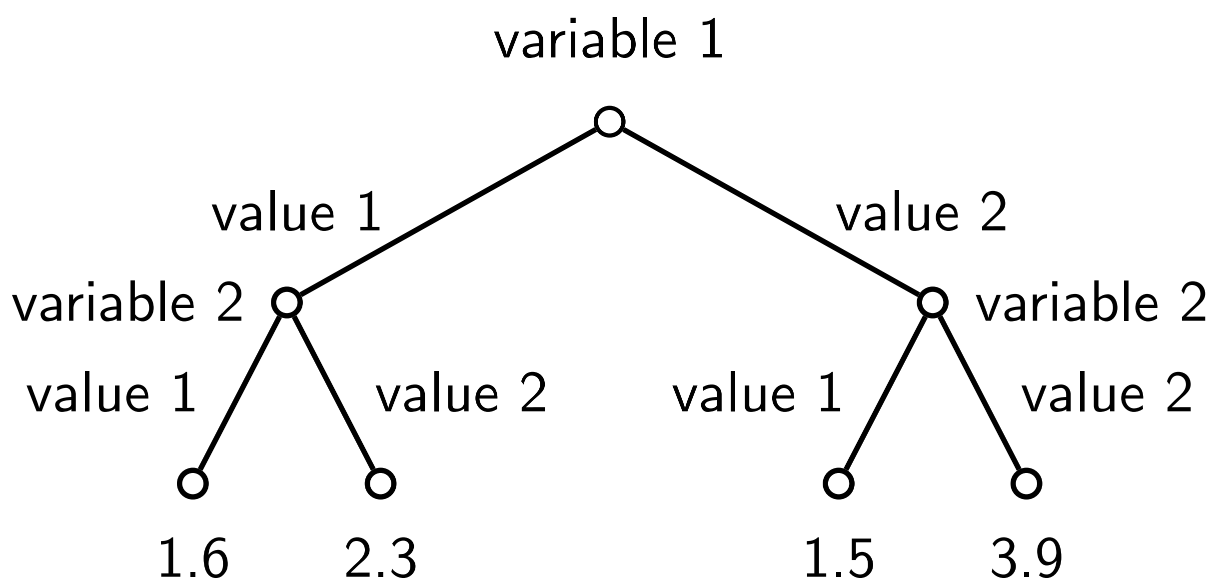
- fundamental properties of various algorithms
- fundamental properties of problems

How to best use available information in a tree search?

Combinatorial Optimization

Given:	set of variables
	possible values for each variable
	objective function over assignments
Find:	assignment that minimizes objective function

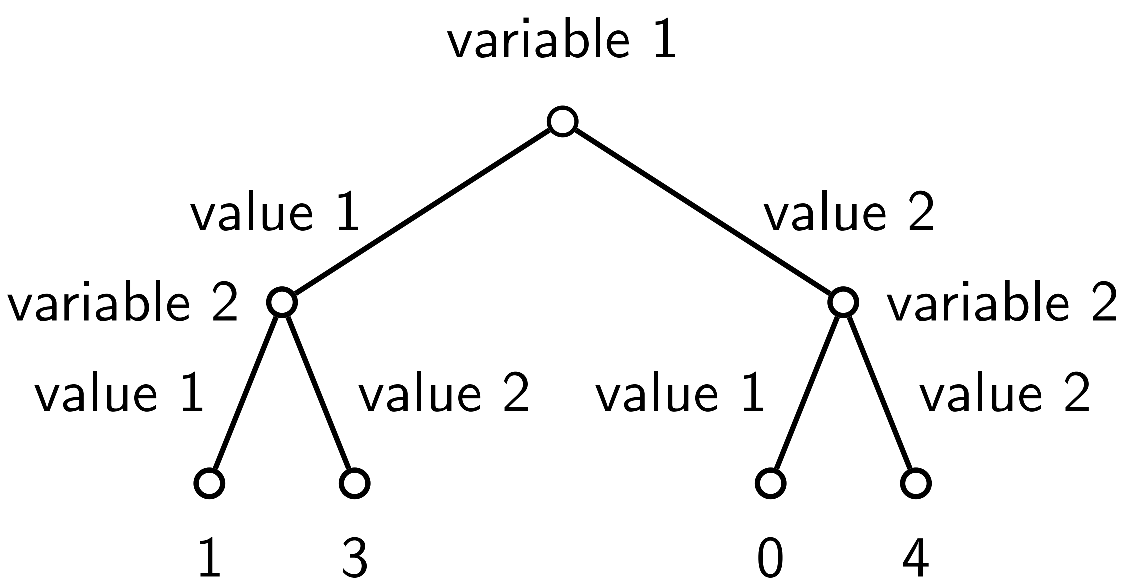
One approach: search tree for best leaf



Constraint Satisfaction

Given: set of variables
possible values for each variable
set of constraints between variables
Find: complete and feasible assignment

Treat as combinatorial optimization:



Types of Search Problems

Shortest path: find shallowest node that is a goal

eg, shortest plan

Constraint satisfaction: find any leaf node that is a goal

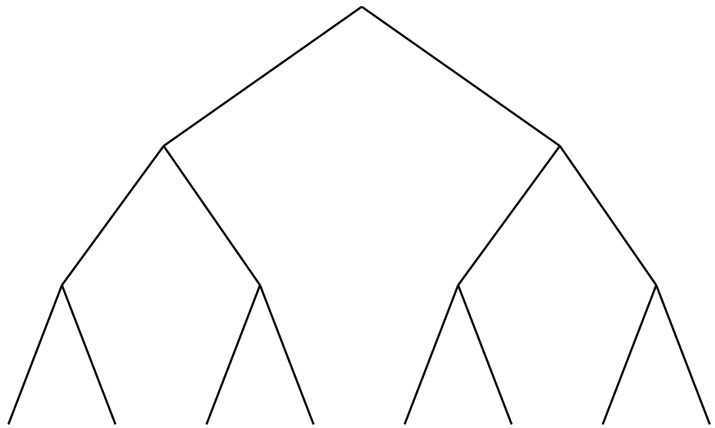
eg, valid configuration

Combinatorial optimization: find best-scoring leaf node

eg, balanced partitioning

Adversarial search: find best-scoring leaf we can surely reach

eg, chess



Types of Search Problems

Shortest path: find shallowest node that is a goal

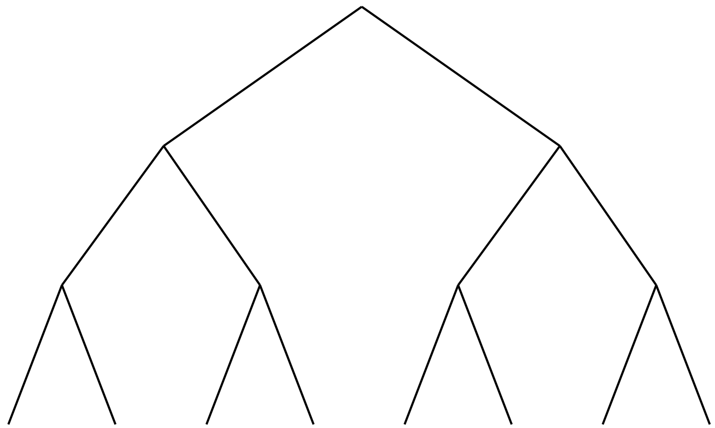
eg, shortest plan

Combinatorial optimization: find best-scoring leaf node

eg, balanced partitioning

Adversarial search: find best-scoring leaf we can surely reach

eg, chess



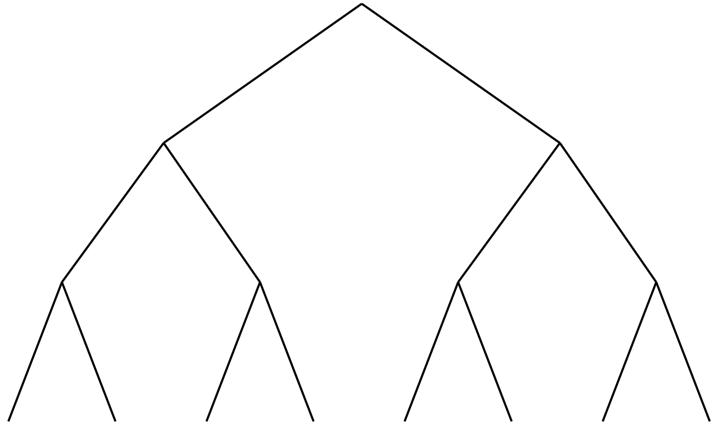
Types of Search Problems

Shortest path: find shallowest node that is a goal

eg, shortest plan

Adversarial search: find best-scoring leaf we can surely reach

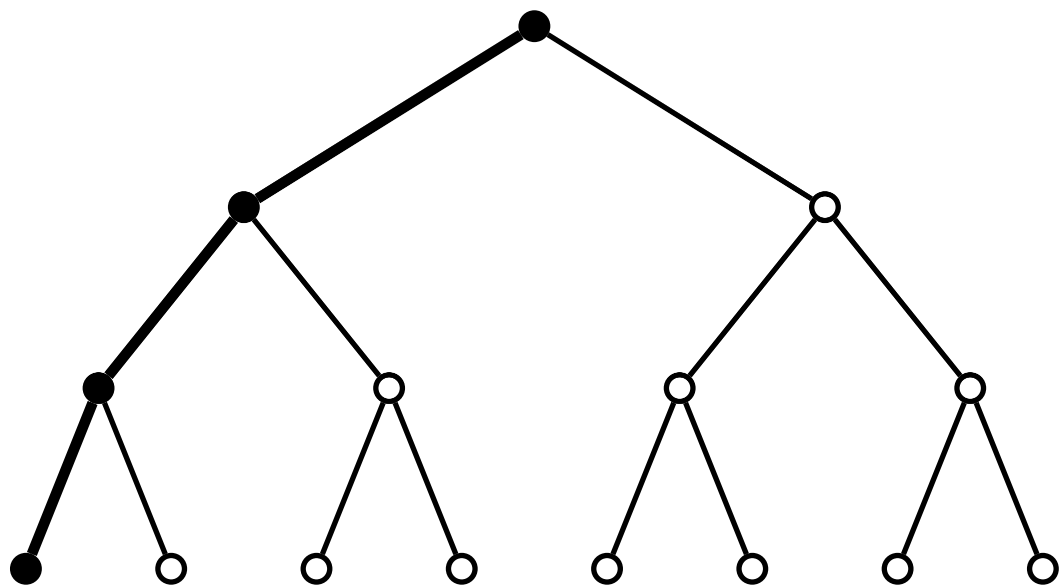
eg, chess



The Problem

For large problems or when optimum is recognizable, search order matters.

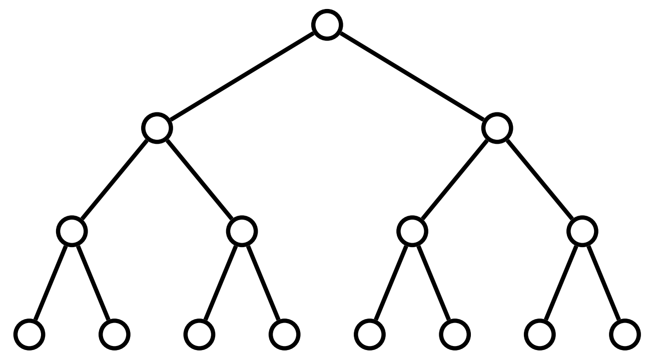
Where was the mistake?



Truncated depth-first is not necessarily optimal!

The Central Idea

Where to backtrack first?



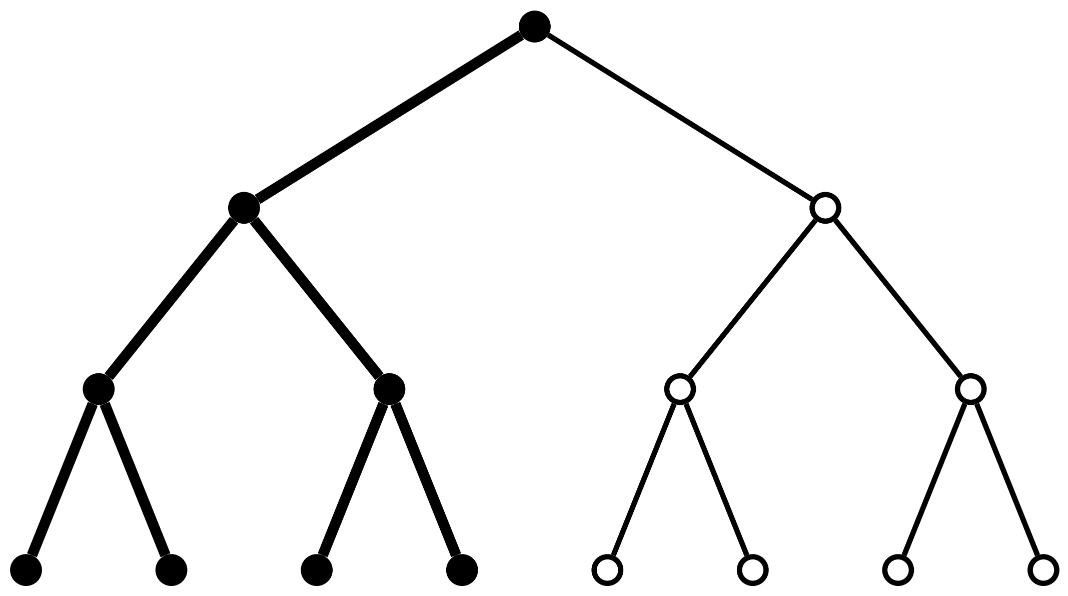
Predetermined order = strong assumptions = ad hoc = brittle

Use a model of leaf costs on-line to guide search.

[Ruml, 2001; Boyan, 1998; Baluja, 1996]

Previous Approaches

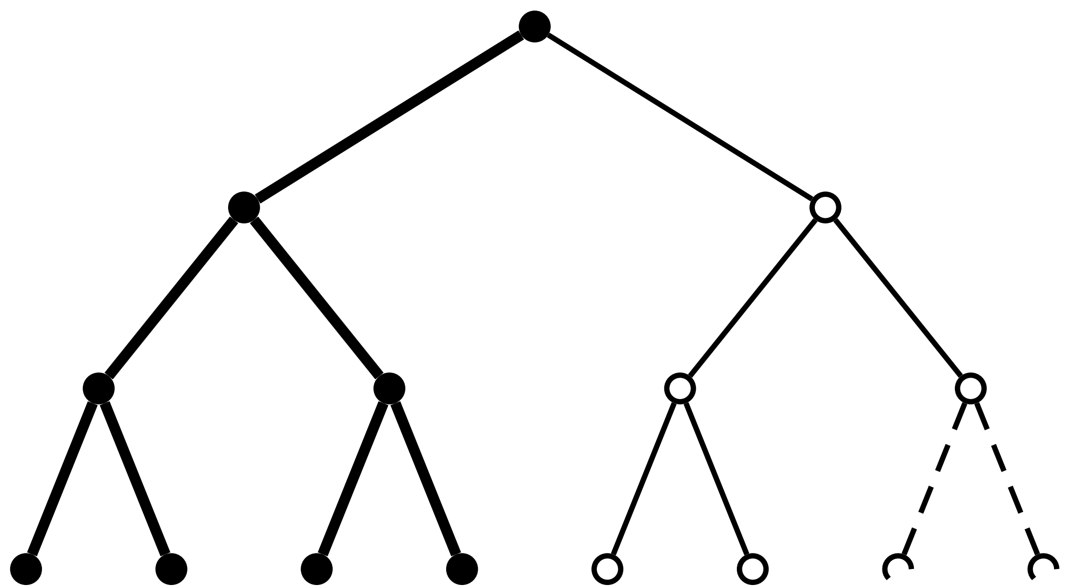
Depth-First Search (DFS)



1. Prune provably bad nodes (branch and bound)
2. Sort children left to right using a heuristic ordering function h

Assumes penalty at top is enormous.

Depth-First Search (DFS)



1. Prune provably bad nodes (branch and bound)
2. Sort children left to right using a heuristic ordering function h

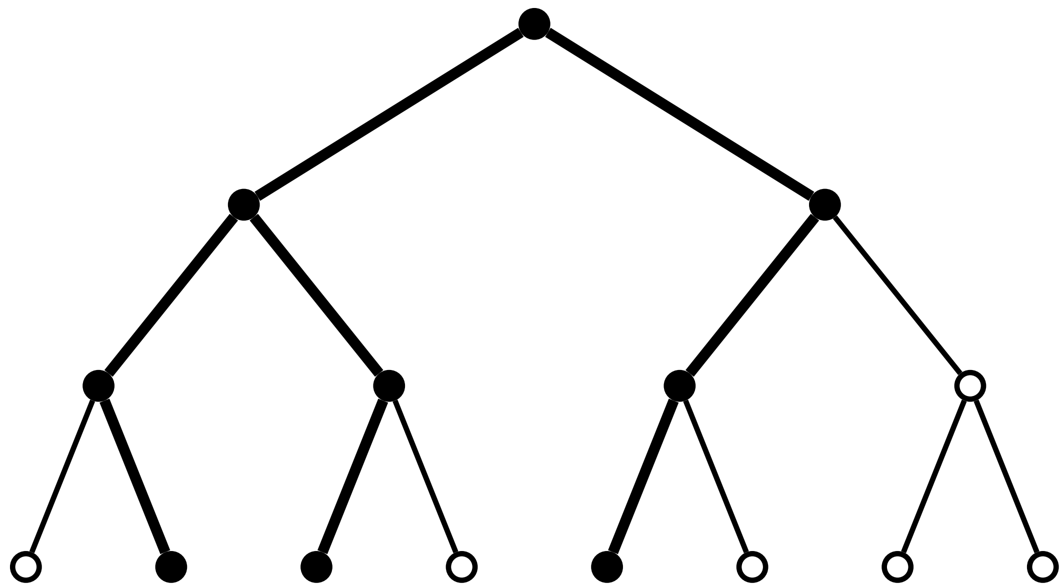
Assumes penalty at top is enormous.

Discrepancy Search

Harvey and Ginsberg (1995): Limited Discrepancy Search

discrepancy: a choice against the heuristic ordering

Explore all paths with k discrepancies before any with $k + 1$.



Korf (1996): ILDS

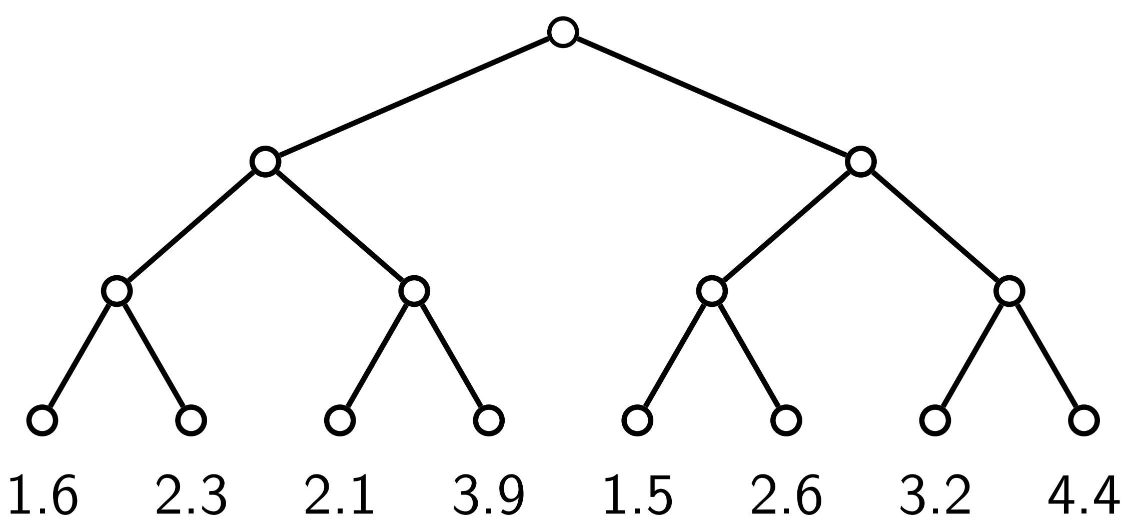
Also Walsh (1997), Ginsberg and Harvey (1992), Meseguer (1997)

Fixed order $\leftrightarrow$ fixed predictions for leaf costs
Want predicted costs to match current problem

Use run-time heuristic information to help make predictions.

Use predictions to guide search:

Rational order: increasing predicted leaf cost = best-first

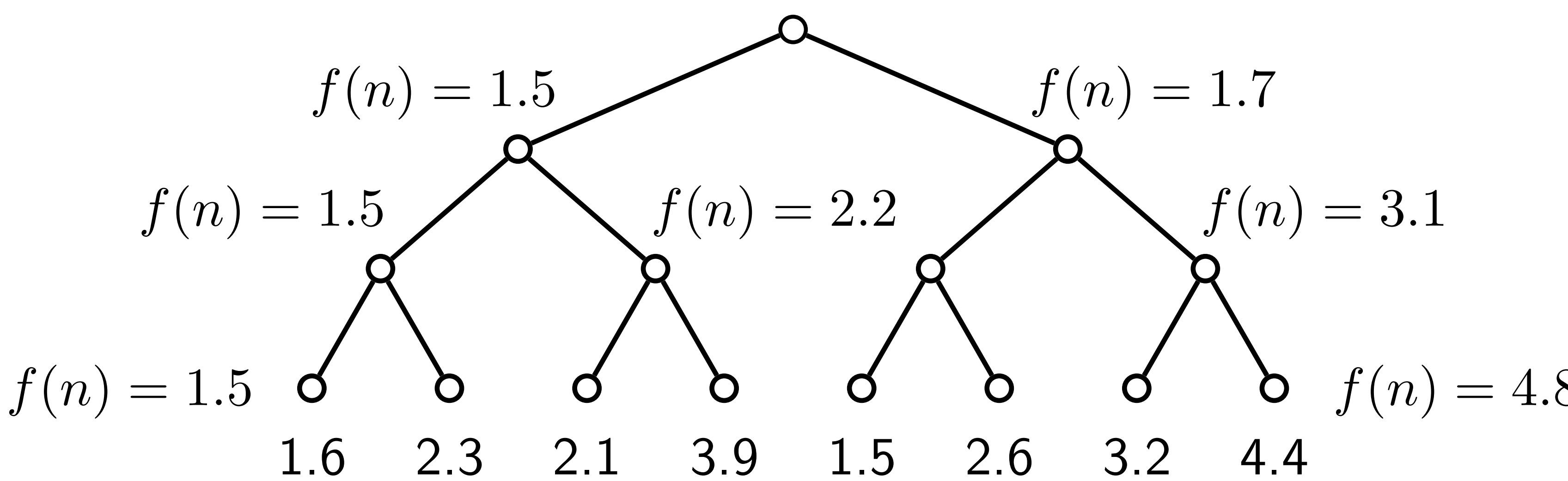


[Ruml, 200

Want to visit leaves in increasing order of predicted cost.

Where are they?

- $f(n)$ = predicted cost of best leaf at or below n
- can use any info at n or on path from root
- want $f(n)$ consistent



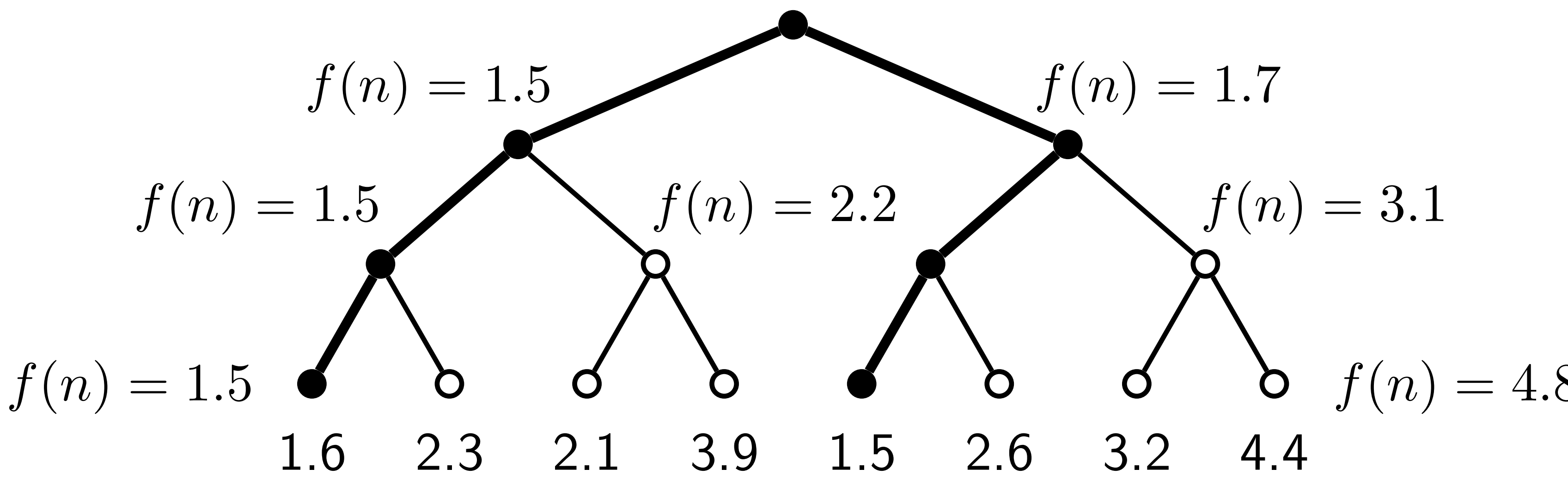
Avoid Bookkeeping

Want to visit leaves in increasing order of predicted cost.

How to keep track of them?

- don't — allow slight misordering
- use iteratively increasing cost bound

Cost bound = 2

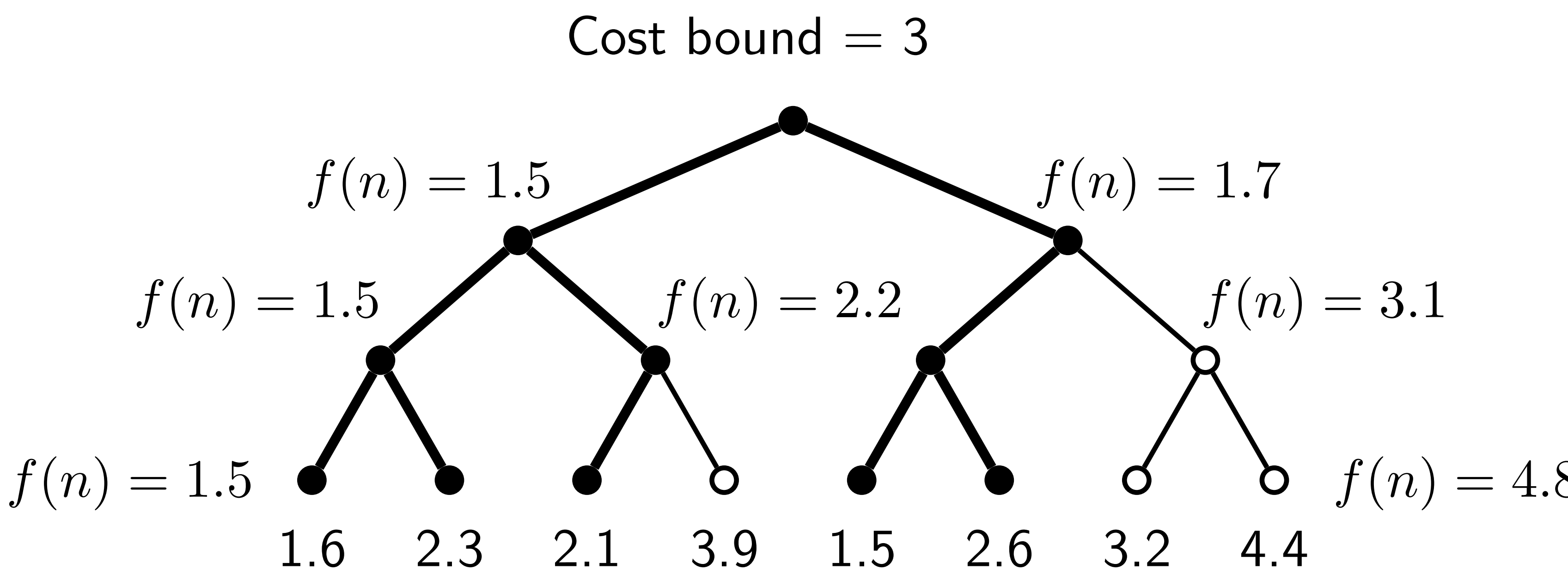


Avoid Bookkeeping

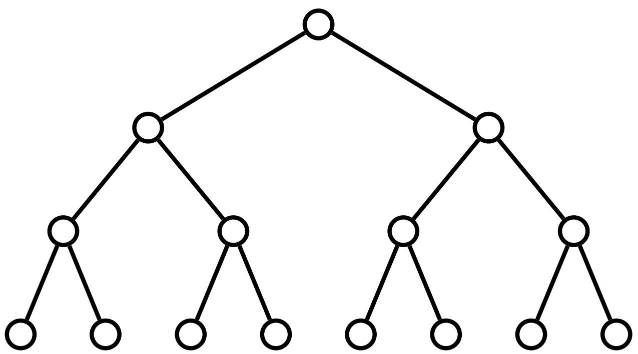
Want to visit leaves in increasing order of predicted cost.

How to keep track of them?

- don't — allow slight misordering
- use iteratively increasing cost bound



BLFS(*root*)
Visit a few leaves
Nodes-desired $\leftarrow$ number of nodes visited so far
Loop until time runs out:
    Double *nodes-desired*
    Estimate cost bound that visits *nodes-desired* nodes
    BLFS-expand(*root*, *bound*)

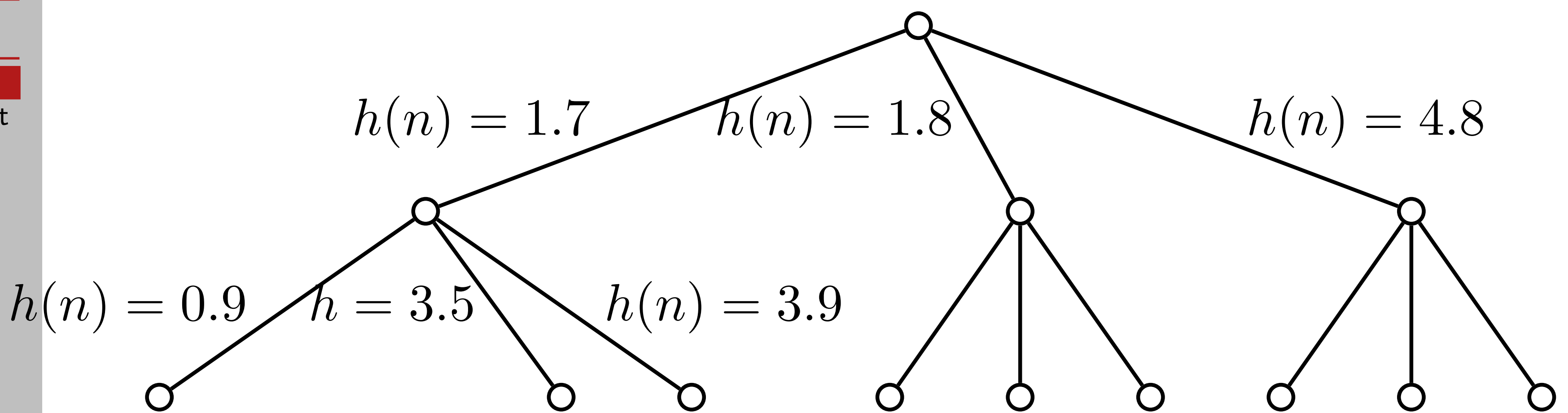


BLFS-expand(*node*, *bound*)
If leaf(*node*), visit(*node*)
else, for each *child* of *node*:
    If best-completion(*child*) $\leq$ *bound*
        BLFS-expand(*child*, *bound*)

Basic BLFS

Indecision Search

Many domains have a quantitative child ordering heuristic:



Fixed model:

- Cost of child $i = h(\text{child } i) - h(\text{child } 0)$
- $f(\textit{leaf}) =$ predicted leaf cost $=$ maximum cost along path

$f(n) =$ maximum cost so far, because child 0 always costs zero

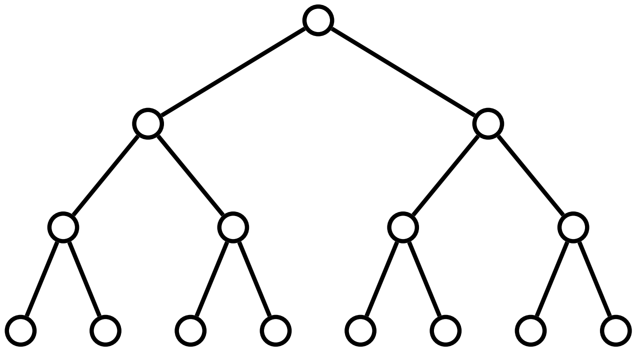
Choosing the Cost Bound

Start by visiting all leaves with predicted cost 0
Estimate cost bound that yield *nodes-desired* nodes

1. Assume independence, estimate branching factor at each level
2. Estimate node cost distributions from costs seen on previous iteration
3. Simulate growth of tree from level to level
4. Implemented using histograms

Best-Leaf-First Search (BLFS)

BLFS(*root*)
Visit a few leaves
Nodes-desired ← number of nodes visited so far
Loop until time runs out:
    Double *nodes-desired*
    Estimate cost bound that visits *nodes-desired* nodes
    BLFS-expand(*root*, *bound*)



BLFS-expand(*node*, *bound*)
If leaf(*node*), visit(*node*)
else, for each *child* of *node*:
    If best-completion(*child*) $\leq$ *bound*
        BLFS-expand(*child*, *bound*)

Test Domains

Constraint satisfaction:

1. Latin square completion (Gomes & Selman, . . .)

1	2	3
3	1	2
2	3	1

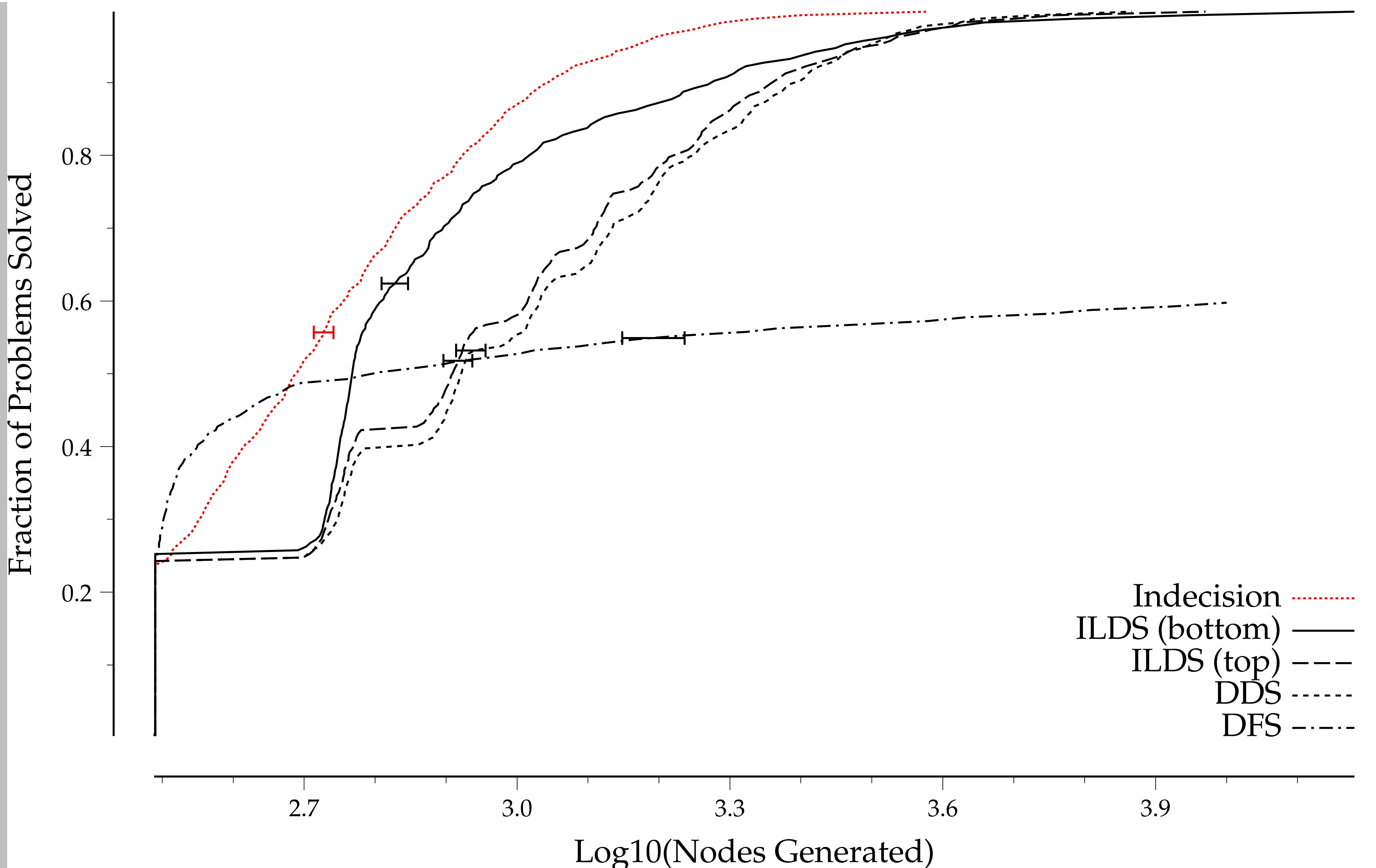
Structure plus random constraints (30% filled)

2. Binary CSPs (Smith, . . .)

   Canonical form

   Random with known characteristics

21 × 21 Latin Squares



Latin Squares

95th percentile of nodes generated to solve instances of each class.

n	DFS	Indec.	ILDS	DDS	Indec / ILDS
11	7,225	188	**183**	206	1.03
13	888,909	**298**	303	357	.983
15	∞	**402**	621	642	.647
17	∞	**648**	1,047	1,176	.619
19	∞	**908**	1,609	1,852	.564
21	∞	**1,242**	2,812	3,077	.442

Random Binary CSPs

95th percentile of nodes generated to solve instances of each class.

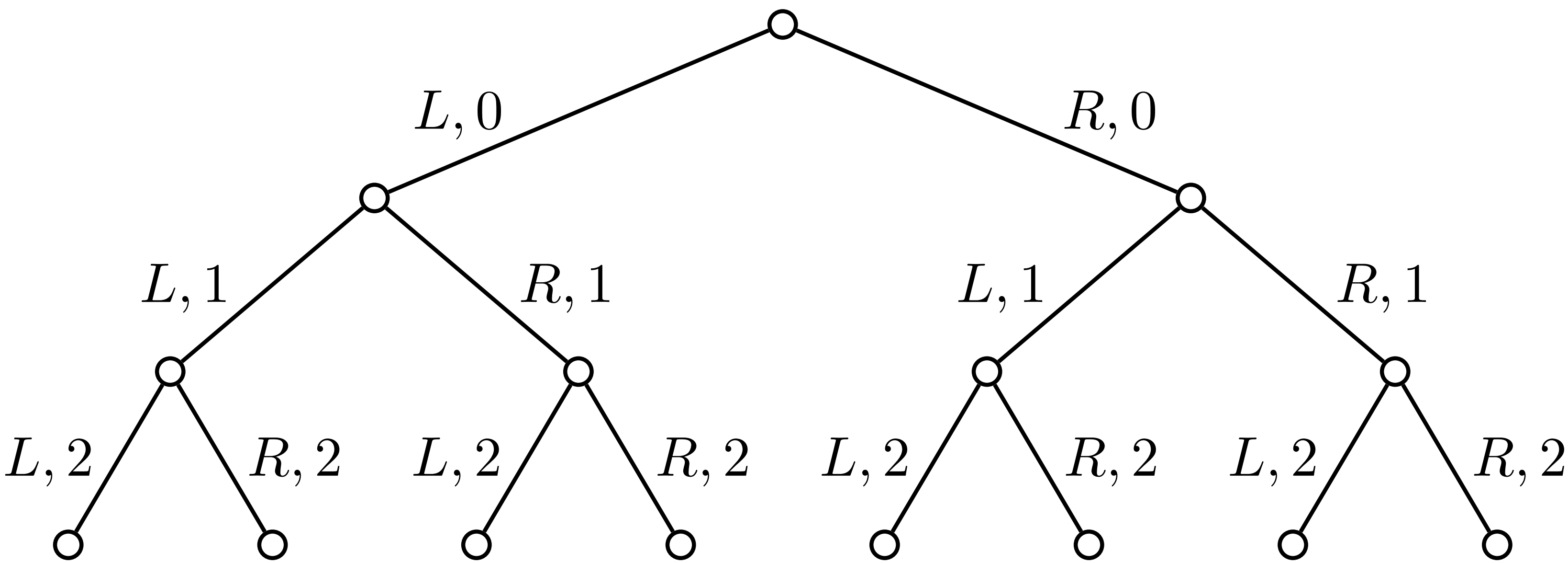
$\langle n, m, p_1, p_2 \rangle$	DFS	Indec.	ILDS	DDS
$\langle 30, 15, .4, .320 \rangle$	1,119	**884**	1,122	1,115
$\langle 30, 15, .4, .347 \rangle$	42,025	**28,294**	30,996	100,387
$\langle 30, 15, .4, .360 \rangle$	**103,878**	536,716	309,848	1,642,806
$\langle 50, 12, .2, .319 \rangle$	1,450	**984**	1,271	1,301
$\langle 50, 12, .2, .347 \rangle$	**22,852**	28,630	52,491	187,856
$\langle 50, 12, .2, .361 \rangle$	**352,788**	387,432	554,036	3,546,588
$\langle 100, 6, .06, .333 \rangle$	31,910	**3,344**	4,012	11,845
$\langle 100, 6, .06, .361 \rangle$	208,112	**70,664**	127,712	2,048,320

BLFS with Learning

Modeling Leaf Costs

Assume cost of leaf is sum of costs of actions along its path.
Assume cost of k-th child at level d depends only on k and d:

$$\textit{leaf} = \sum_d \textit{cost}_{k,d}$$



Paths form linear equations:

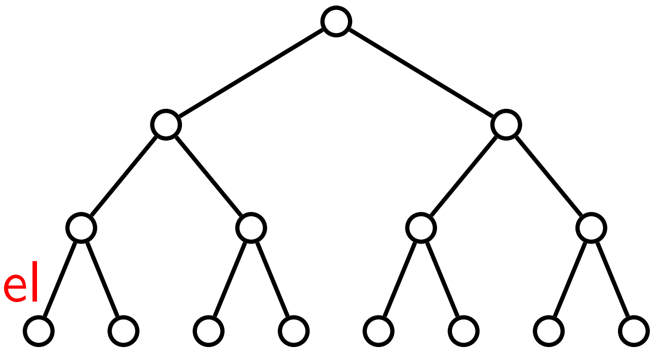
$$\begin{array}{cccccccl} c_{L,0} & & + & c_{L,1} & & + & & c_{R,2} = \mathit{leaf}_1 \\ c_{L,0} & & + & & c_{R,1} & + & c_{L,2} & = \mathit{leaf}_2 \\ & c_{R,0} & + & c_{L,1} & & + & c_{L,2} & = \mathit{leaf}_3 \end{array}$$

Solve for mean costs of actions via on-line least-squares regression (Widrow and Hoff, 1960; Murata et al., 1997)

To aid learning, we enforce $c_{L,d} < c_{R,d}$.

$f(n)$ is sum of actions so far plus best possible in future..

BLFS(*root*)
Visit a few leaves
Initialize model
Nodes-desired $\leftarrow$ number of nodes visited so far
Loop until time runs out:
    Double *nodes-desired*
    Estimate cost bound that visits *nodes-desired* nodes
    Make static copy of current model
    BLFS-expand(*root*, *bound*)



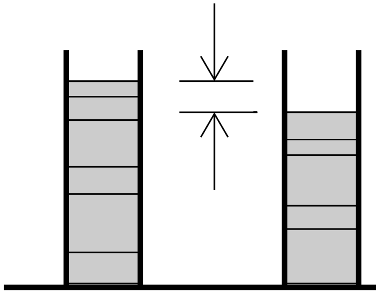
BLFS-expand(*node*, *bound*)
If leaf(*node*), visit(*node*) and update model
else, for each *child* of *node*:
    If best-completion(*child*) $\leq$ *bound*
        BLFS-expand(*child*, *bound*)

Must be able to:

1. Predict cost of best leaf in subtree
   - With linear model, can be precomputed and cached
2. Estimate cost bound that yields *nodes-desired* nodes
   - As before, predict number of nodes for given bound
   - Use binary search over values for bound

Number Partitioning: Given n numbers $w_1, \ldots, w_n$.

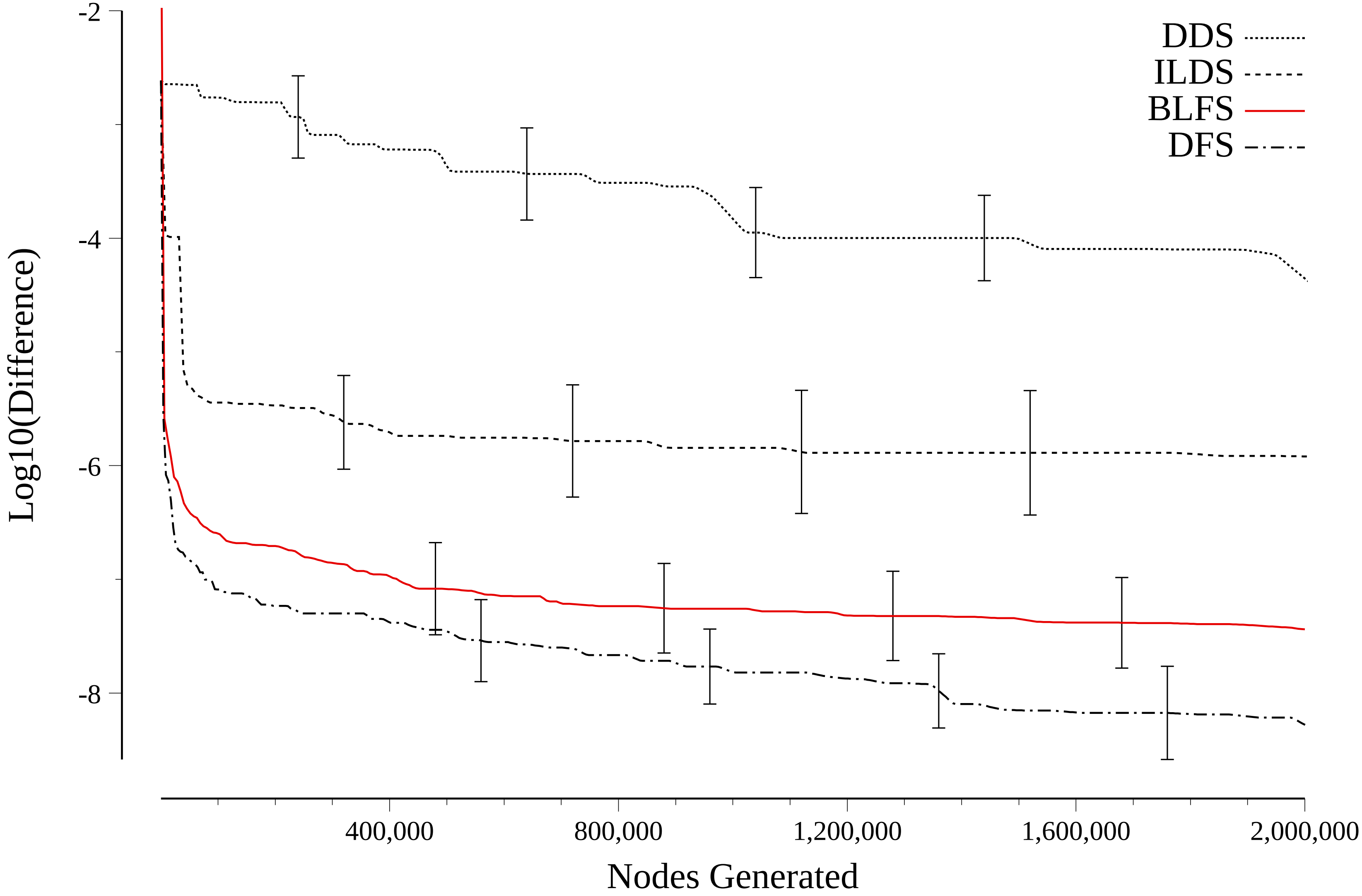
Find partition into A and B to minimize $\left| \sum_{w \in A} w - \sum_{w \in B} w \right|$



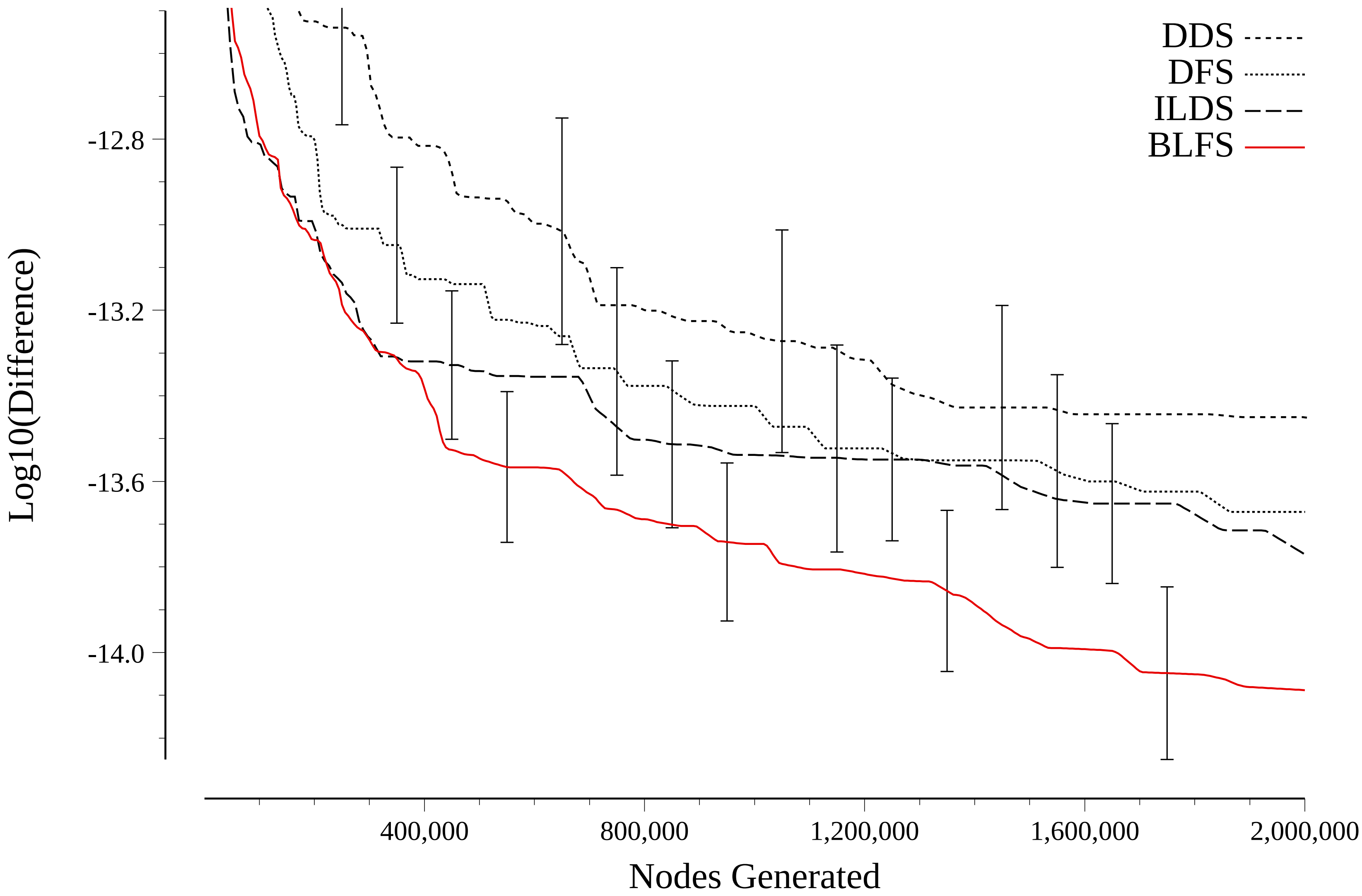
1. Basic Representation (Johnson et al, . . .)
   branch on placement of largest remaining

2. CKK Representation (Korf, . . .)
   branch on type of constraint for two largest remaining

Basic Space (256 #s, 82 digits)



CKK Space (256 #s, 82 digits)



Preliminary Results

Competitive or superior in all domains:

1. Constraint satisfaction
   (a) Latin square completion: *Fixed BLFS superior*
   (b) Binary CSPs: *Fixed BLFS competitive*
2. Optimization
   (a) Basic number partitioning: *Learning BLFS competitive*
   (b) CKK number partitioning: *Learning BLFS superior*
3. Related methods (Ruml, 2001)
   (a) Harvey-Ginsberg abstract CSP trees
   (b) Boolean satisfiability

Relationship to IDA*

Both visit all nodes within an increasing $f(n)$ bound.

	BLFS	IDA*
$f(n)$ semantics	best leaf below n	best path through n
$f(n)$ source	from user or learned	$= g(n) + h(n)$
$g(n)$ source	not necessary	from problem
$h(n)$ source	not necessary	from user
$f(n)$ property	consistent	non-overestimating
additive model	convenient	required
updating bound	estimation	add ϵ
	rational	optimal

Best-first tree search using a model of leaf cost

1. Adapts backtracking to current tree
2. Complete
3. Explicit modeling assumptions
4. Easy use of prior knowledge from similar problems
5. Allows investigation of heuristic knowledge
    - Which kinds are most powerful?
    - How can they be combined?
6. Allows comparison of constructive and improvement search

Principles should apply equally well to improvement search

Extra slides

Rationalizes Previous Work

1. Discrepancy search (Harvey, Ginsberg; Korf; Walsh), Iterative broadening (Ginsberg, Harvey)
   - assumes *ad hoc* action costs
2. Randomized restarts (Gomes, Selman, Kautz; Walsh;...)
   - randomly reorders children with scores $< \epsilon$
3. GRASP (Feo and Resende,...)
   - randomly reorders top k children
4. Heuristic-biased stochastic sampling (Bresina)
   - fixed bias for preferred child
5. Adaptive Probing (Ruml)
   - *ad hoc* exploration policy

Help!

1. Applications
   - DFS is lousy
   - significant computation per node
2. Visualizers
   - trees with 2^{100} nodes
3. Models and methods for on-line learning
   - estimation error from on-line regression
4. New problems
   - anytime shortest-path

Predicting Nodes for Bound

Consider cost bound as allowance being spent

- Compute expected number of affordable branches at each level (costs are known)
- Compute expected distribution of remaining allowance (truncating subtractive convolution):

$$p_{\mathit{new}}(x) = \begin{cases} \int (p_{\mathit{child}}(y) \times p_{\mathit{old}}(x+y))dy & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Robustness

	best	near	poor	pathological
BLFS	7	3	1	
DFS	4	4	2	1
ILDS		9	2	
DDS		3	8	

No other tree search algorithm is as robust.

Incomplete Tree Search

Constructive vs improvement search

- Often confused with complete vs incomplete
- What are their fundamental properties?
- What about designing for incompleteness?

Constructive methods easily exploit knowledge

- variable and value choice heuristics
- lower bounds, constraint propagation