Iterative-deepening Search with On-line Tree Size Prediction

Ethan Burns and Wheeler Ruml

University of New Hampshire

{eaburns, ruml} at cs.unh.edu

Thanks to the NSF and the DARPA CSSG program for support

Take Away

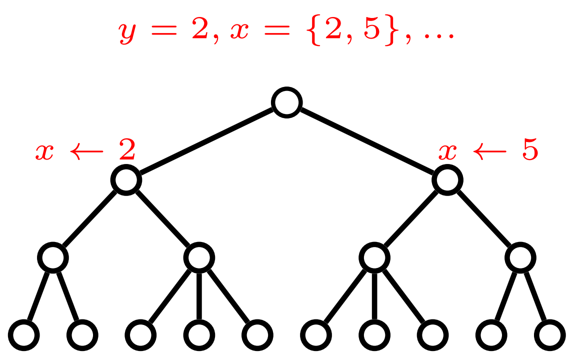
On-line learning for shortest-path search

- Conventional IDA* uses a cost bound parameter
- Simple bound setting can fail on trivial problems
- Learn a model of search space and adjust the bound online

Learning search space structure to control algorithms on-line works for shortest-path search, too!

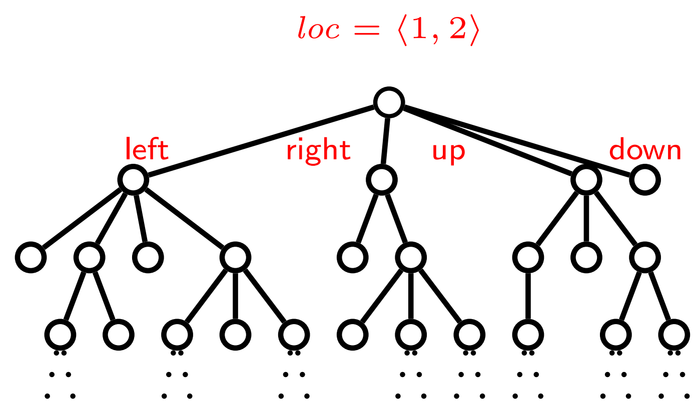
Tree Search Versus Shortest Path Search

Tree Search: Optimization



- Nodes: partial assignment
- Edges: assign values

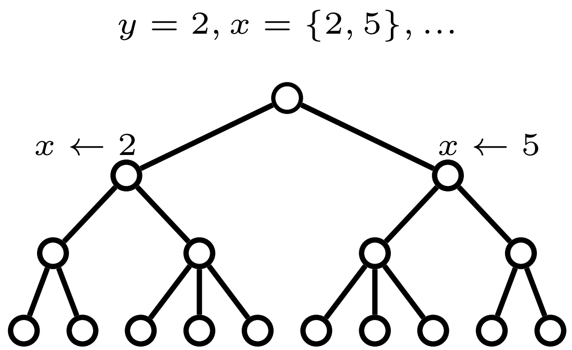
Heuristic Search: Planning



- Nodes: world states
- Edges: actions

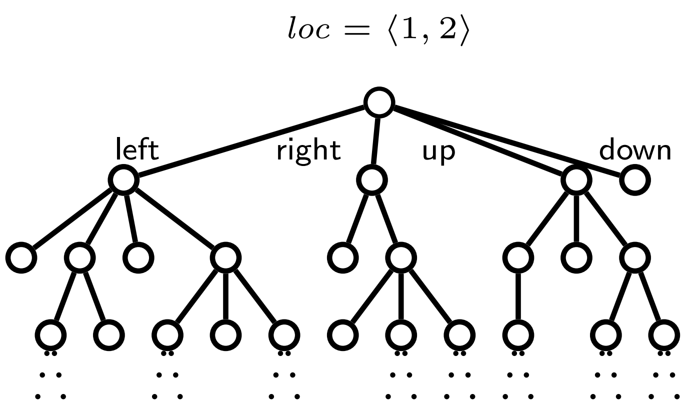
Tree Search Versus Shortest Path Search

Tree Search: Optimization



- Find minimum cost assignment

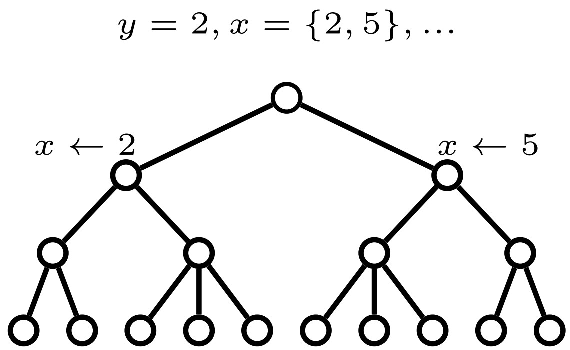
Heuristic Search: Planning



- Find cheapest plan that achieves goal

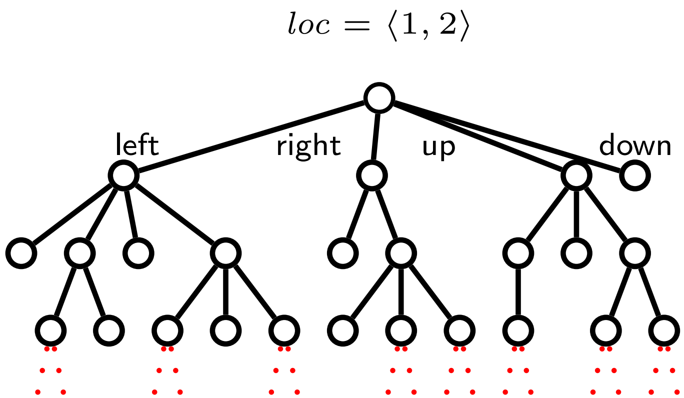
Tree Search Versus Shortest Path Search

Tree Search: Optimization



- Bounded depth
- Depth-first branch-and-bound

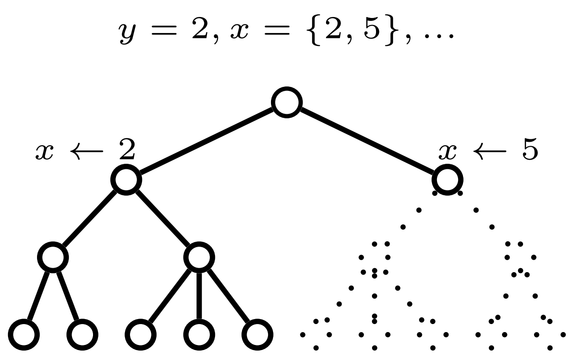
Heuristic Search: Planning



- Unbounded depth
- How do we bound the search?

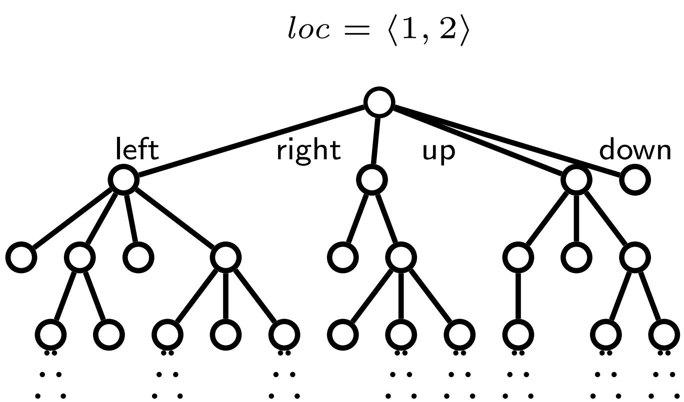
Tree Search Versus Shortest Path Search

Tree Search: Optimization



- Nodes have lower bounds
- Incumbent provides upper bound for pruning

Heuristic Search: Planning



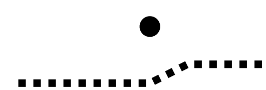
- Artificial upper bound to limit the tree
- Increase bound when search fails

Iterative Deepening A* (IDA*, Korf, 1985)

- Compute a lower bound at each node: f
- Use depth-first search and an upper bound to limit the tree
- Raise the bound and restart when the search fails

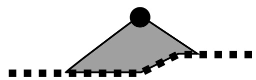
Iterative Deepening A* (IDA*, Korf, 1985)

- Compute a lower bound at each node: f
- Use depth-first search and an upper bound to limit the tree
- Raise the bound and restart when the search fails



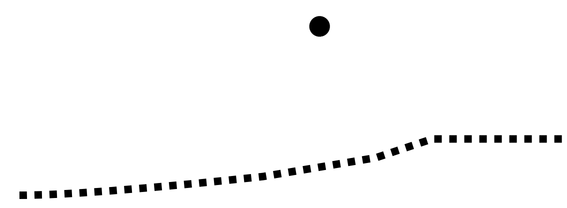
Iterative Deepening A* (IDA*, Korf, 1985)

- Compute a lower bound at each node: f
- Use depth-first search and an upper bound to limit the tree
- Raise the bound and restart when the search fails



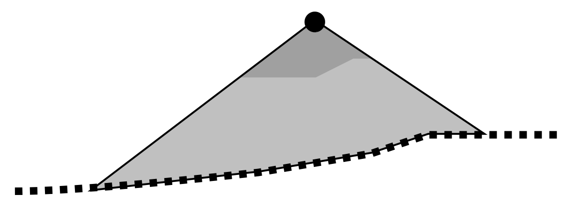
Iterative Deepening A* (IDA*, Korf, 1985)

- Compute a lower bound at each node: f
- Use depth-first search and an upper bound to limit the tree
- Raise the bound and restart when the search fails



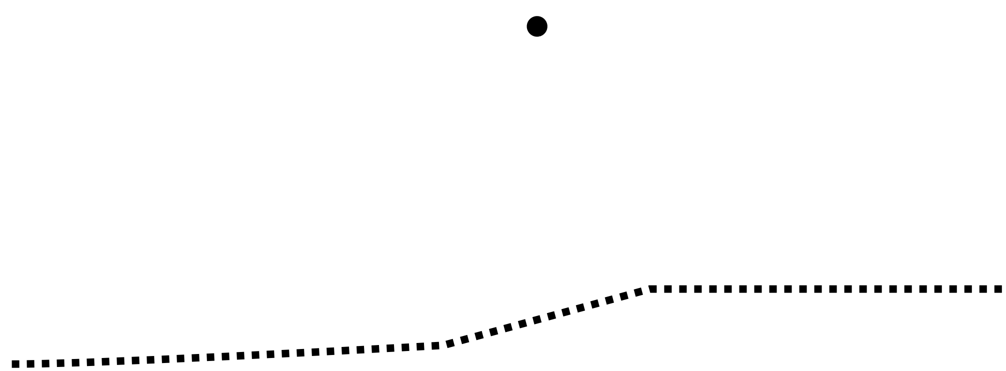
Iterative Deepening A* (IDA*, Korf, 1985)

- Compute a lower bound at each node: f
- Use depth-first search and an upper bound to limit the tree
- Raise the bound and restart when the search fails



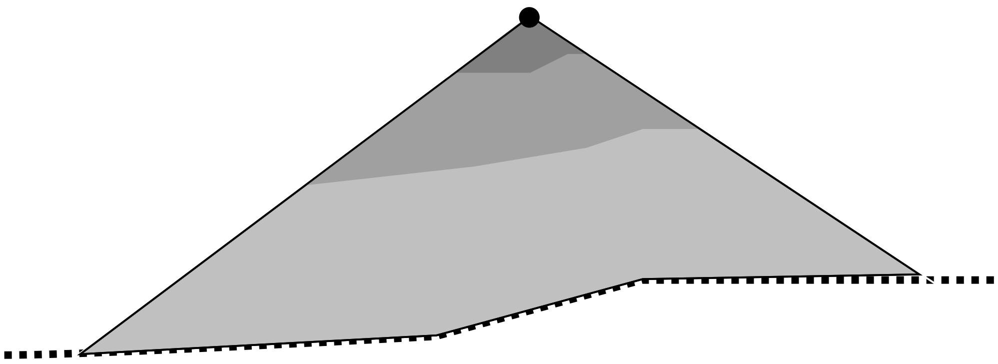
Iterative Deepening A* (IDA*, Korf, 1985)

- Compute a lower bound at each node: f
- Use depth-first search and an upper bound to limit the tree
- Raise the bound and restart when the search fails



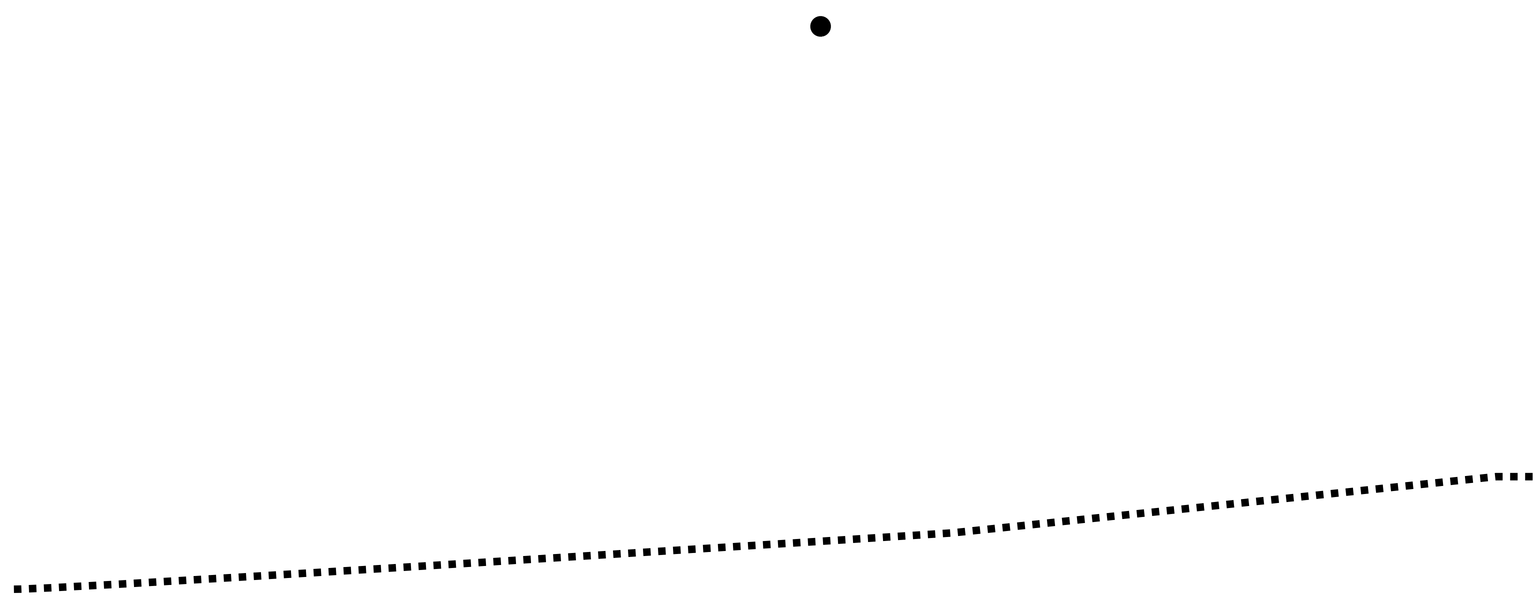
Iterative Deepening A* (IDA*, Korf, 1985)

- Compute a lower bound at each node: f
- Use depth-first search and an upper bound to limit the tree
- Raise the bound and restart when the search fails



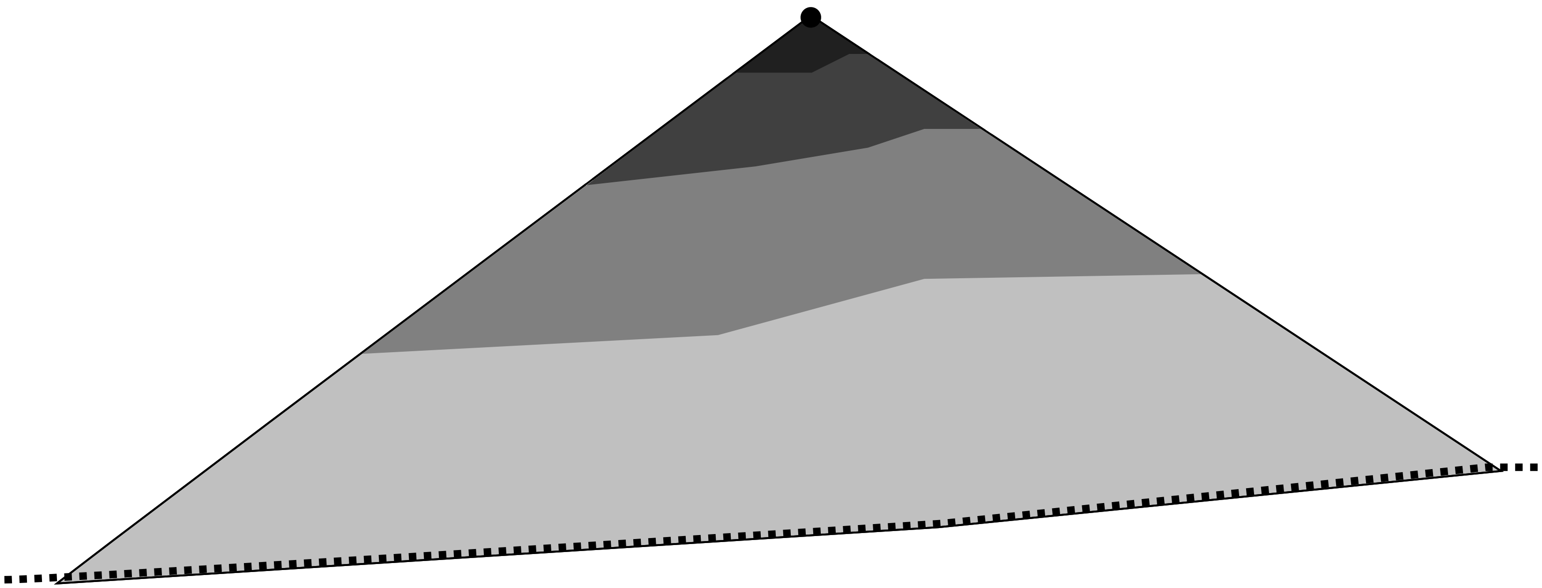
Iterative Deepening A* (IDA*, Korf, 1985)

- Compute a lower bound at each node: f
- Use depth-first search and an upper bound to limit the tree
- Raise the bound and restart when the search fails



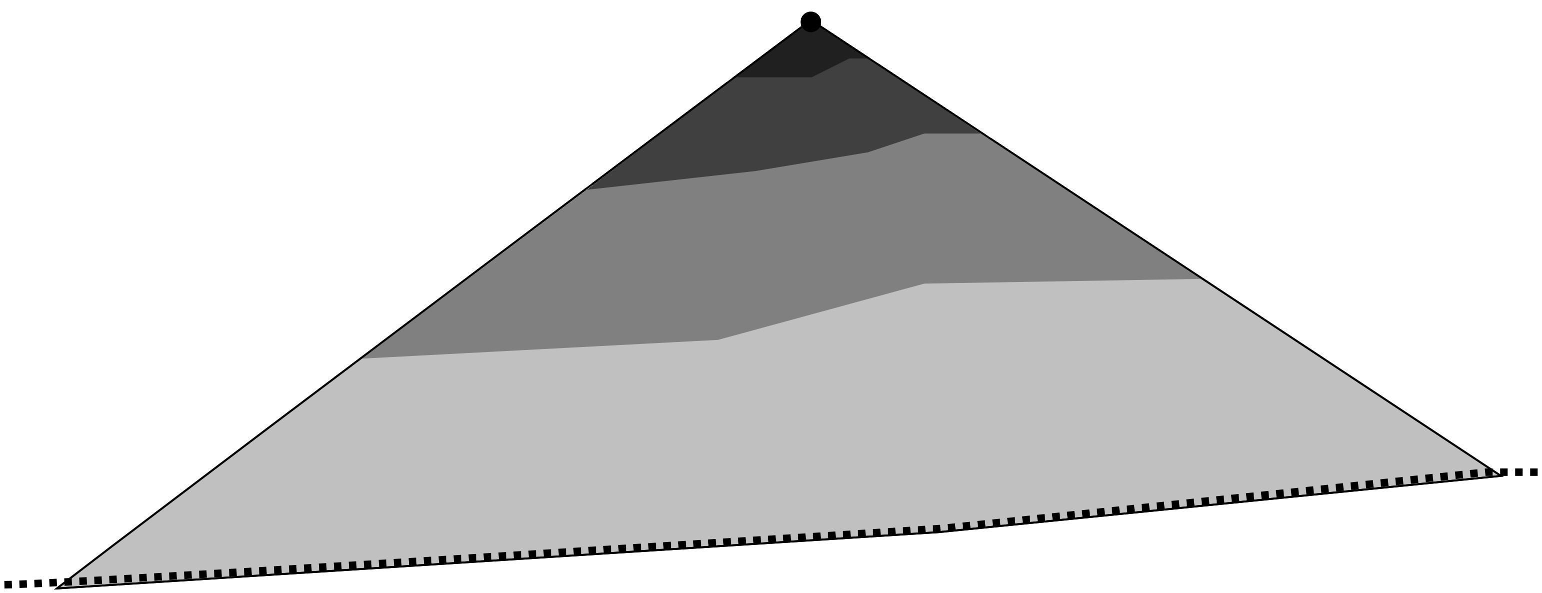
Iterative Deepening A* (IDA*, Korf, 1985)

- Compute a lower bound at each node: f
- Use depth-first search and an upper bound to limit the tree
- Raise the bound and restart when the search fails



Iterative Deepening A* (IDA*, Korf, 1985)

- Compute a lower bound at each node: f
- Use depth-first search and an upper bound to limit the tree
- Raise the bound and restart when the search fails



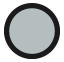
How do we set the bound?

IDA* Growth

If the new bound yields exponential growth then the re-expansion overhead is small

IDA* Growth

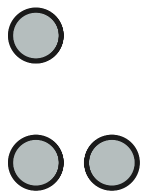
If the new bound yields exponential growth then the re-expansion overhead is small

$f < 10$ 

If the new bound yields exponential growth then the re-expansion overhead is small

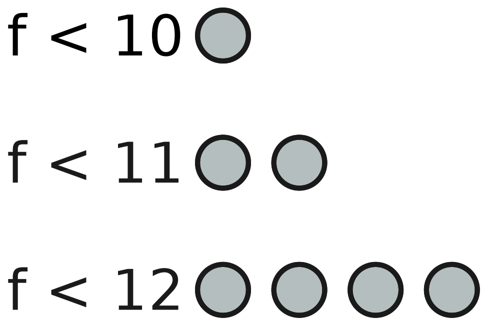
$f < 10$

$f < 11$



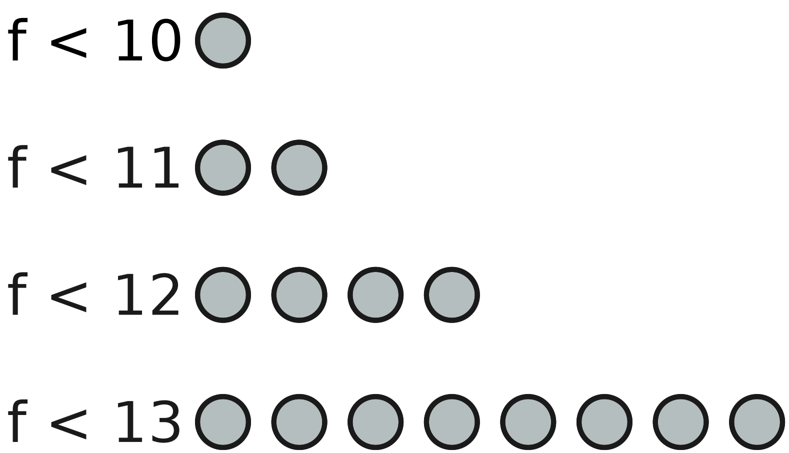
IDA* Growth

If the new bound yields exponential growth then the re-expansion overhead is small



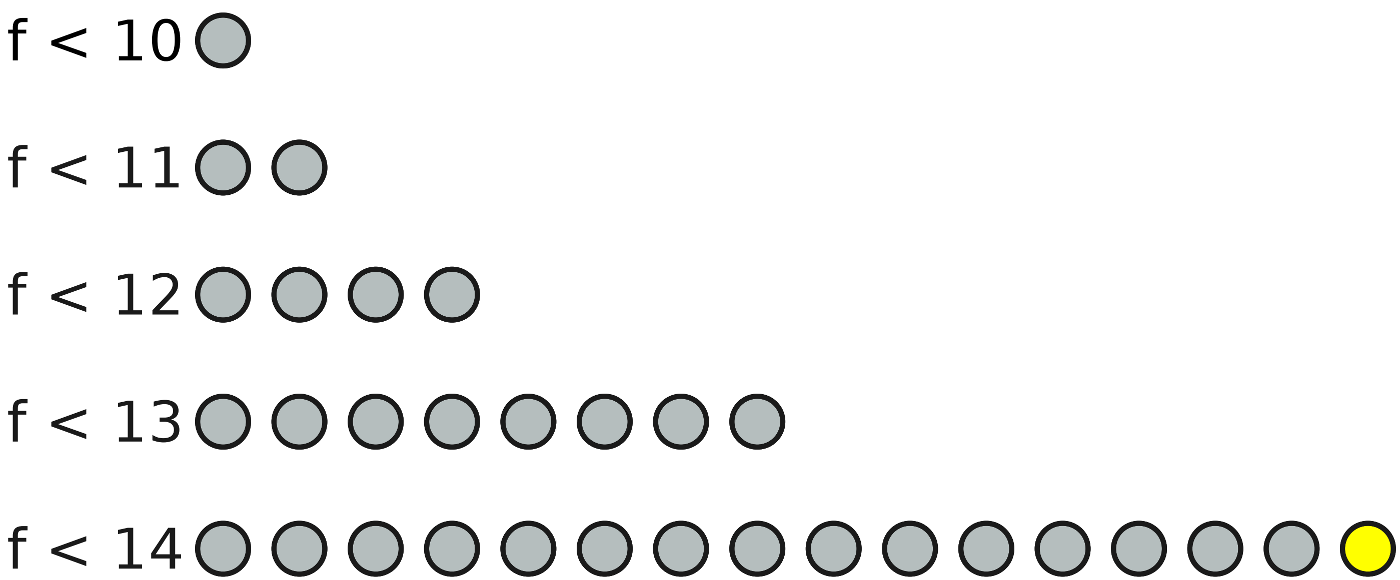
IDA* Growth

If the new bound yields exponential growth then the re-expansion overhead is small



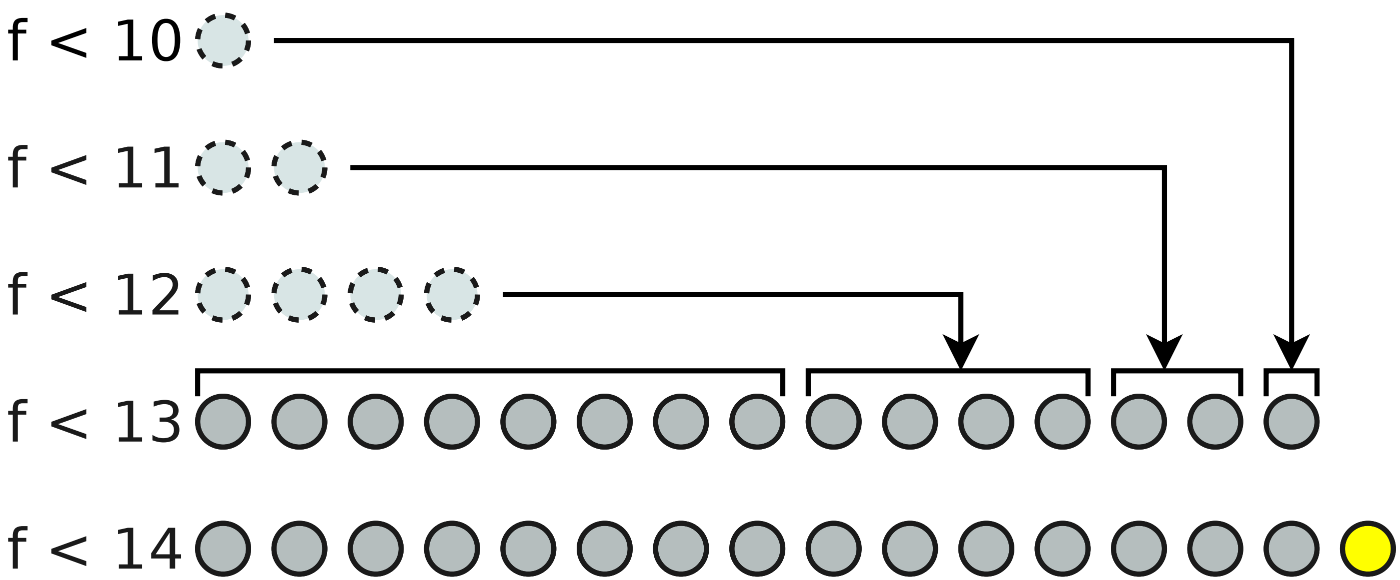
IDA* Growth

If the new bound yields exponential growth then the re-expansion overhead is small



IDA* Growth

If the new bound yields exponential growth then the re-expansion overhead is small

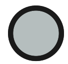


Problems with IDA*

If the new bound yields little growth then the re-expansion overhead dominates

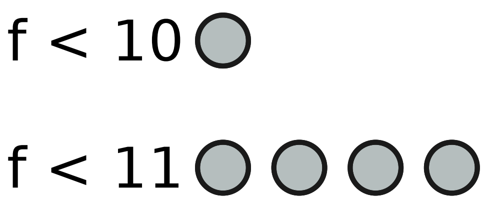
Problems with IDA*

If the new bound yields little growth then the re-expansion overhead dominates

$f < 10$ 

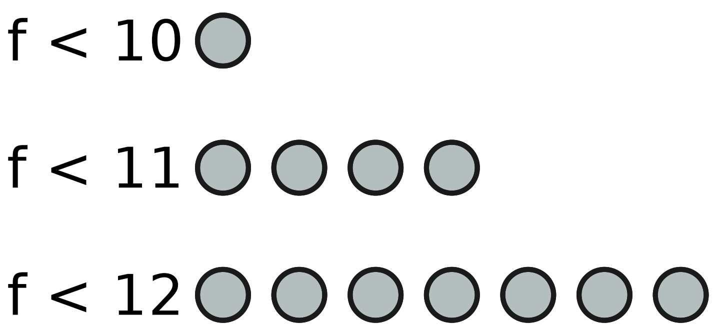
Problems with IDA*

If the new bound yields little growth then the re-expansion overhead dominates



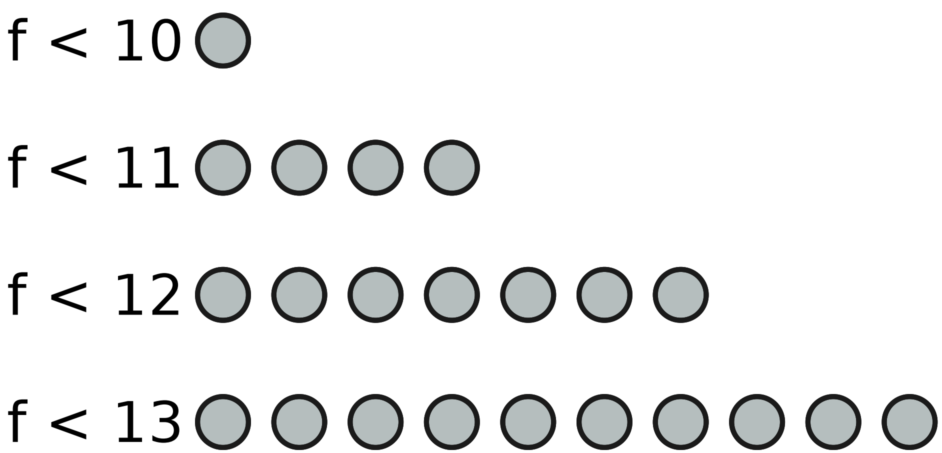
Problems with IDA*

If the new bound yields little growth then the re-expansion overhead dominates



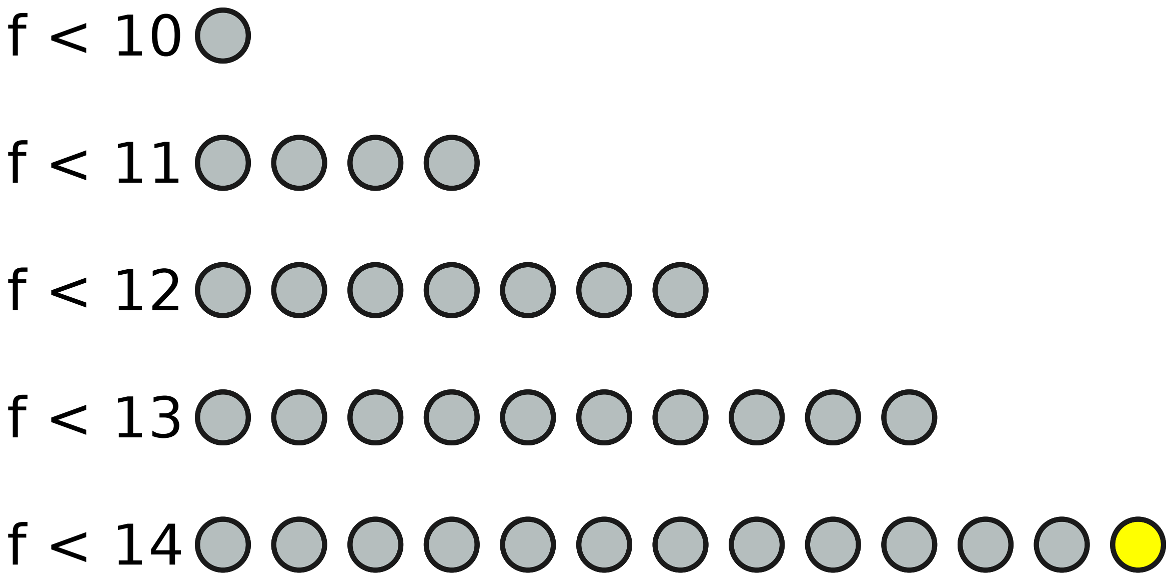
Problems with IDA*

If the new bound yields little growth then the re-expansion overhead dominates



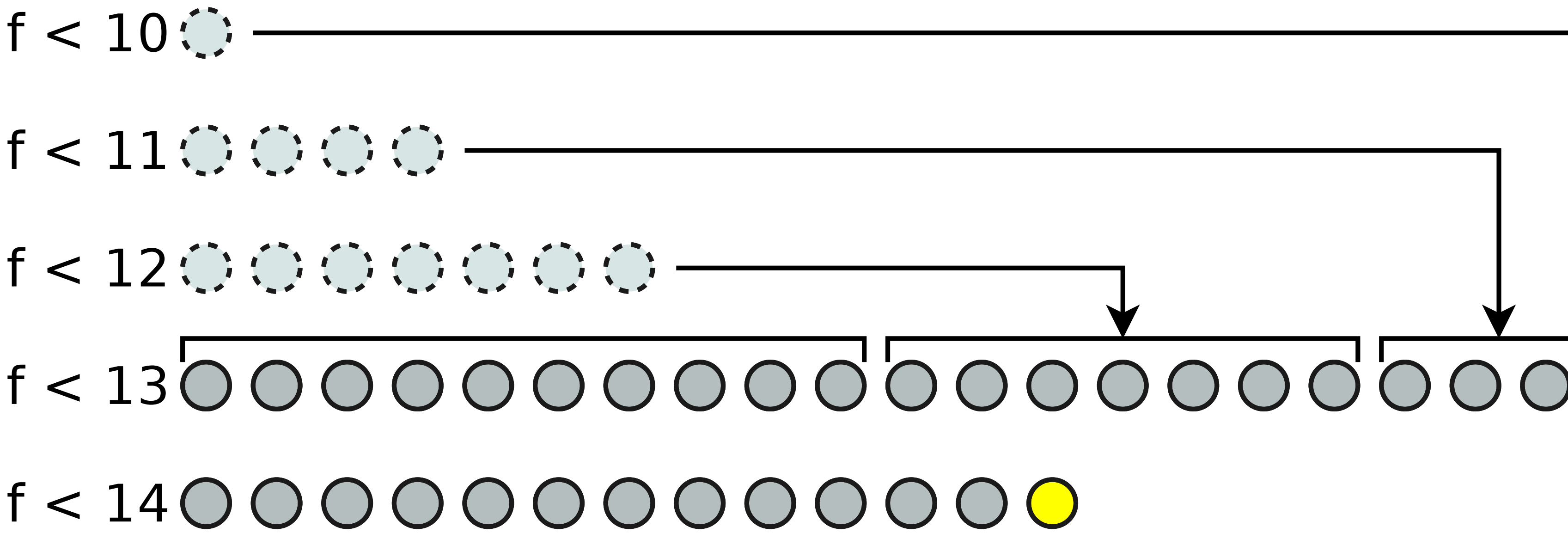
Problems with IDA*

If the new bound yields little growth then the re-expansion overhead dominates



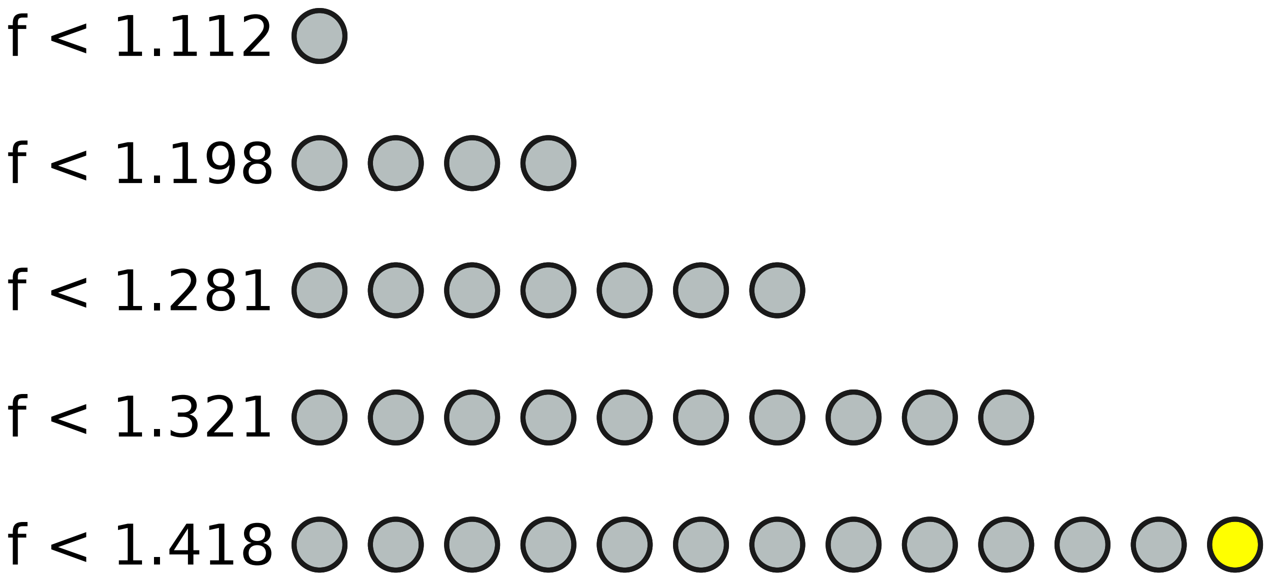
Problems with IDA*

If the new bound yields little growth then the re-expansion overhead dominates



Real Valued Costs

Real valued costs tend to have sub-exponential growth



How Do We Set The Bound?

- If the bound is set too low there is too much re-expansion overhead (original IDA*)
- If the bound is set too high then the tree becomes intractably large

How Do We Set The Bound?

- If the bound is set too low there is too much re-expansion overhead (original IDA*)
- If the bound is set too high then the tree becomes intractably large

Answer: learn a model of the space on-line

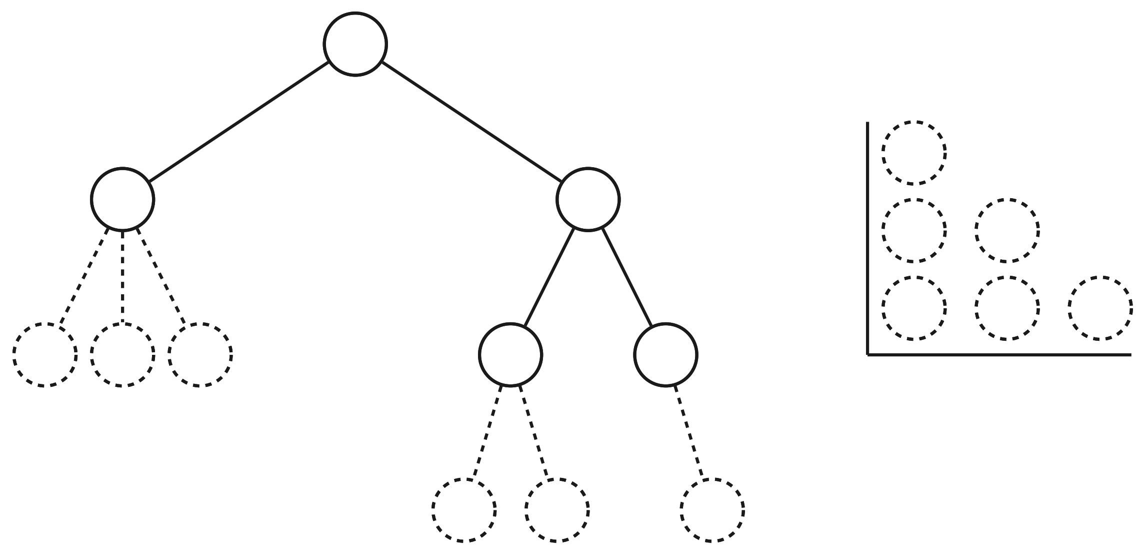
Previous Work

IDA* with Controlled Re-expansion (IDA*$_{\text{CR}}$, Sarkar et al., 1991)

Goal: Try to select an f bound for the next iteration that doubles the number of nodes

Goal: Try to select an f bound for the next iteration that doubles the number of nodes

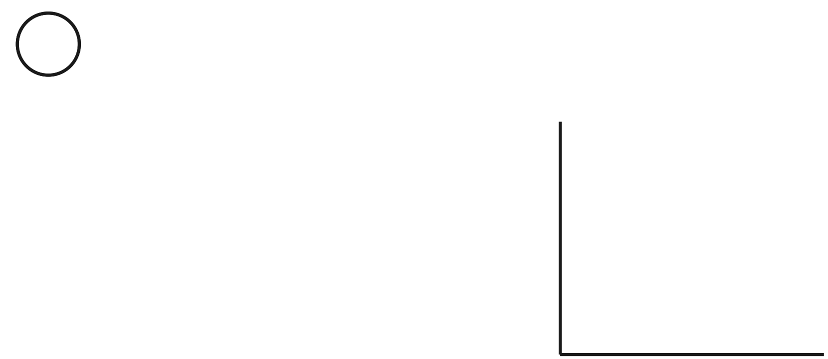
- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration



IDA* with Controlled Re-expansion (IDA*$_{CR}$, Sarkar et al., 1991)

Goal: Try to select an f bound for the next iteration that doubles the number of nodes

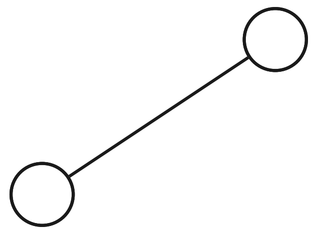
- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration



IDA* with Controlled Re-expansion (IDA*$_{CR}$, Sarkar et al., 1991)

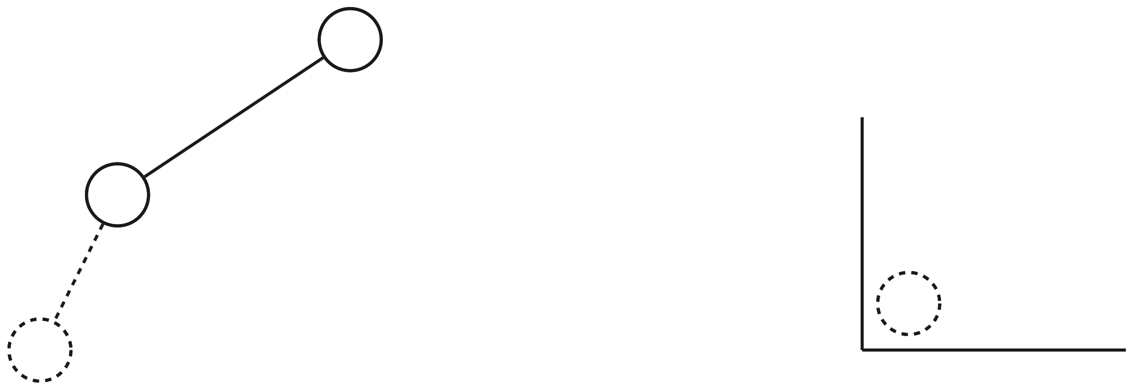
Goal: Try to select an f bound for the next iteration that doubles the number of nodes

- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration



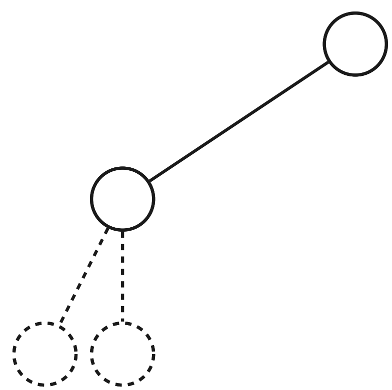
Goal: Try to select an f bound for the next iteration that doubles the number of nodes

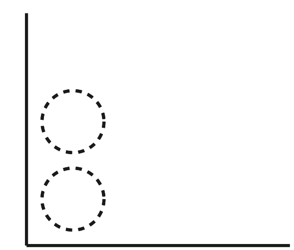
- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration



Goal: Try to select an f bound for the next iteration that doubles the number of nodes

- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration

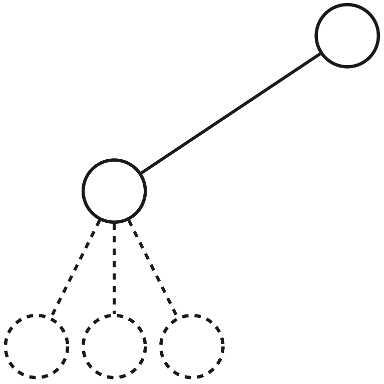


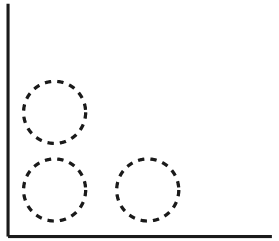


IDA* with Controlled Re-expansion (IDA*$_{CR}$, Sarkar et al., 1991)

Goal: Try to select an f bound for the next iteration that doubles the number of nodes

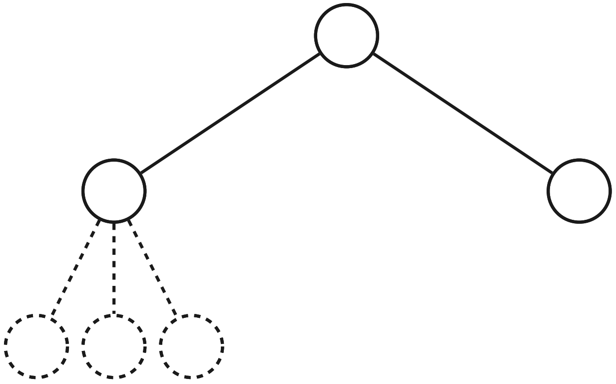
- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration

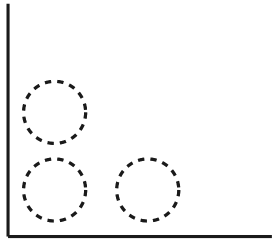




Goal: Try to select an f bound for the next iteration that doubles the number of nodes

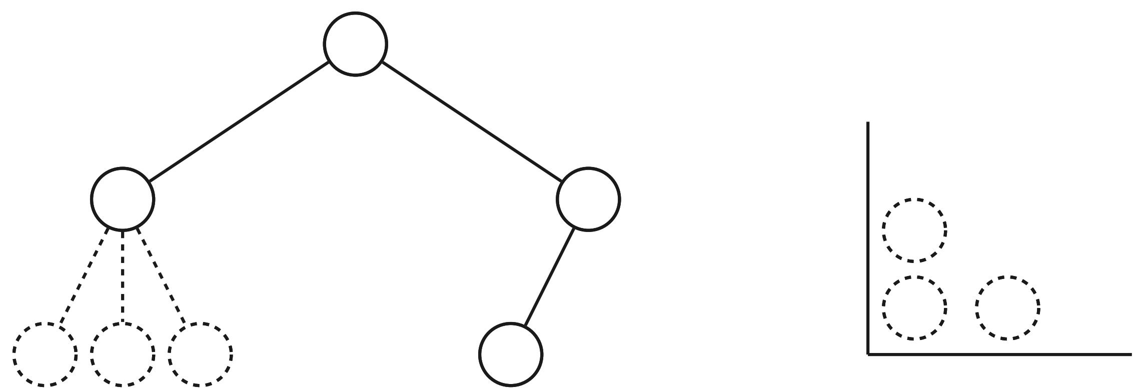
- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration





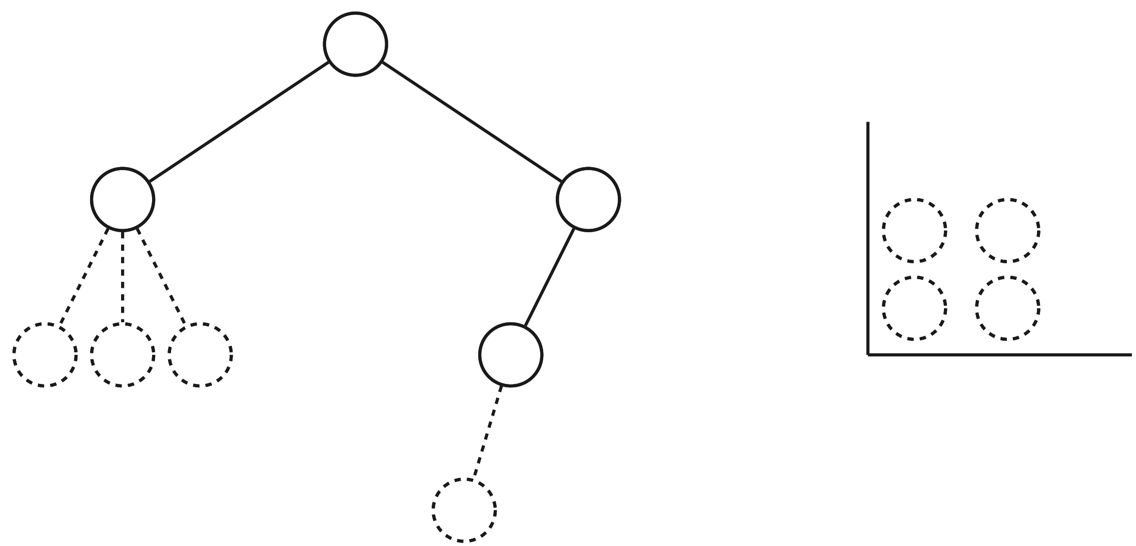
Goal: Try to select an f bound for the next iteration that doubles the number of nodes

- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration



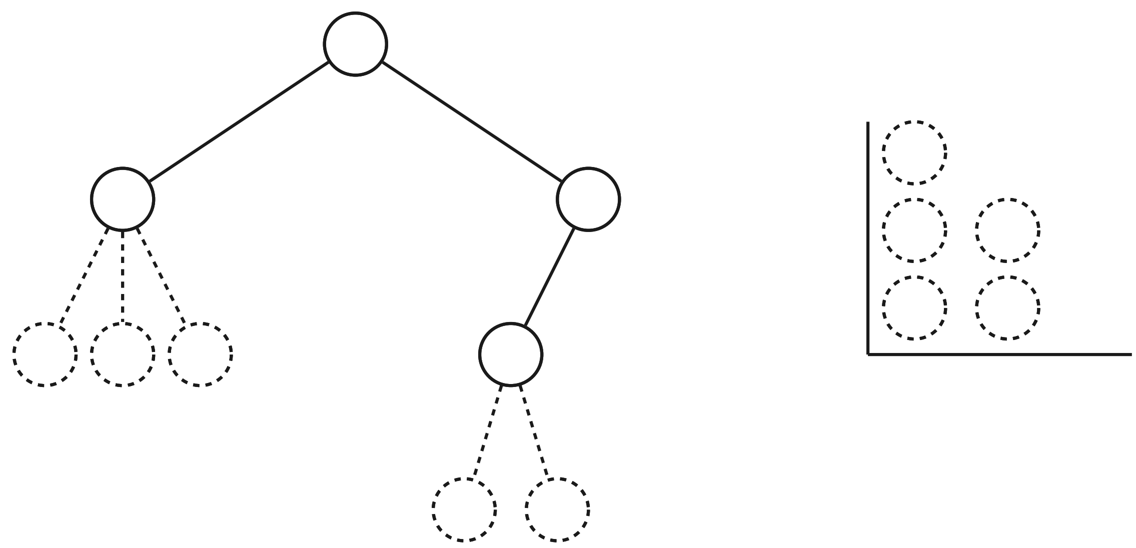
Goal: Try to select an f bound for the next iteration that doubles the number of nodes

- Histogram of f values of pruned nodes
    - These nodes are candidates for the next iteration



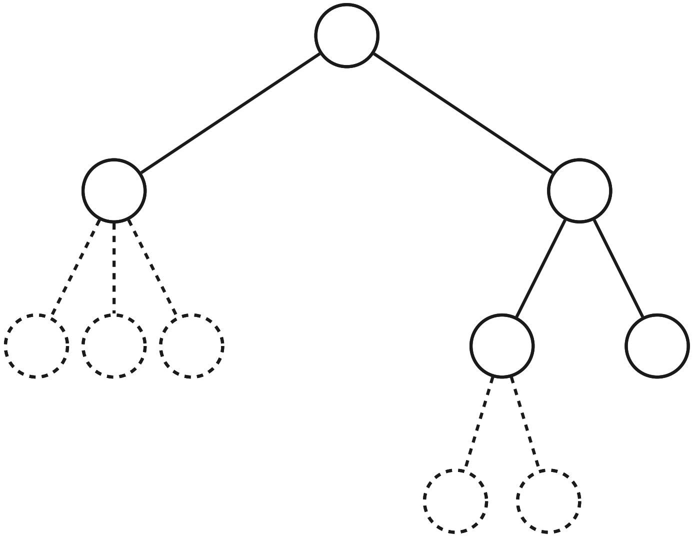
Goal: Try to select an f bound for the next iteration that doubles the number of nodes

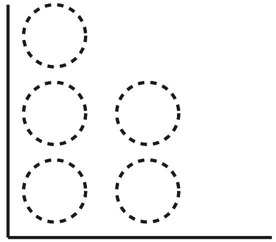
- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration



Goal: Try to select an f bound for the next iteration that doubles the number of nodes

- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration

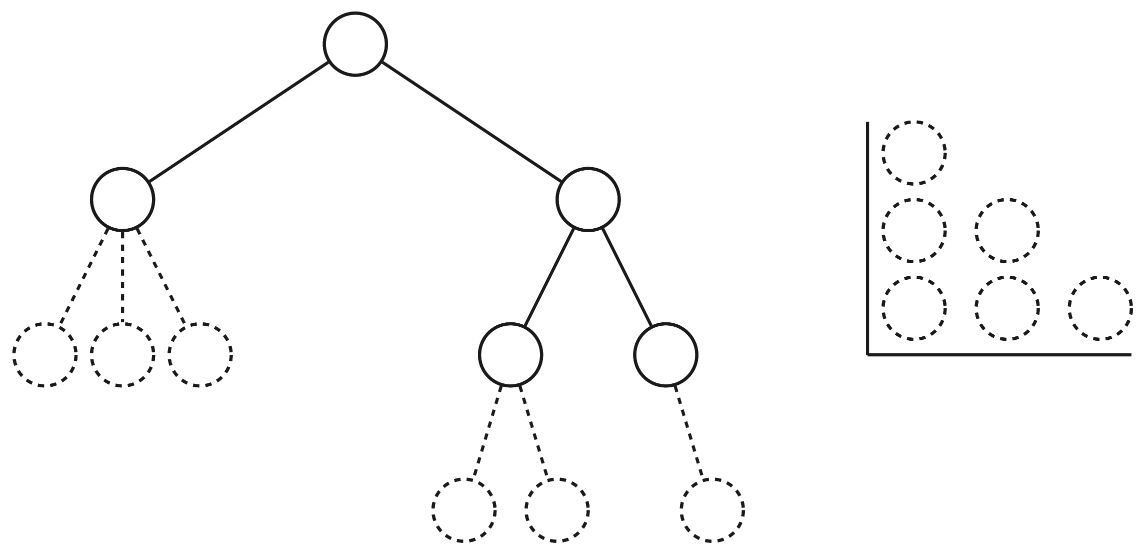




IDA* with Controlled Re-expansion (IDA*$_{CR}$, Sarkar et al., 1991)

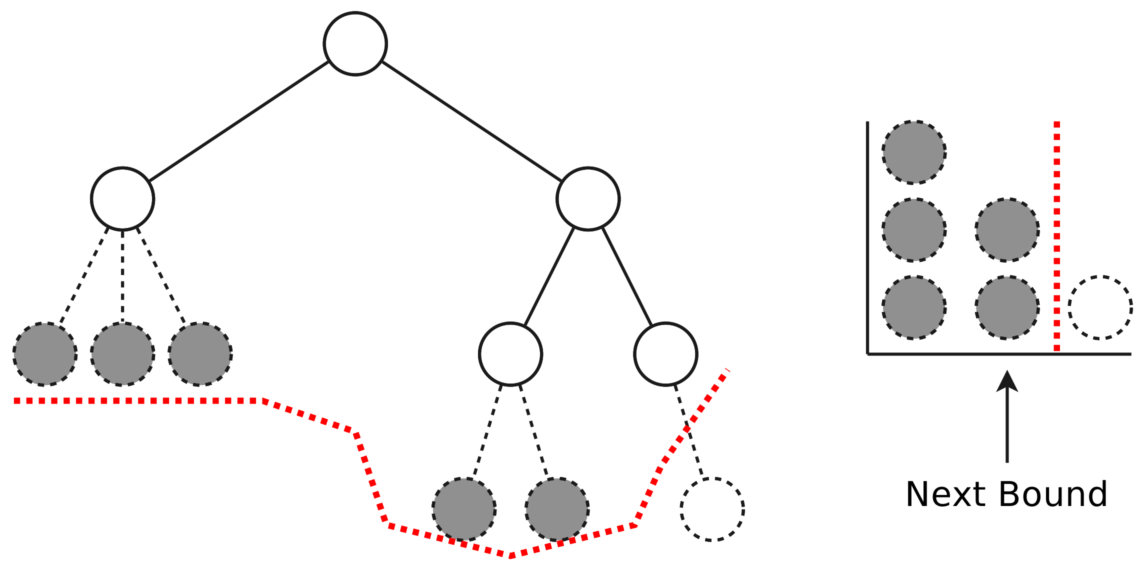
Goal: Try to select an f bound for the next iteration that doubles the number of nodes

- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration



Goal: Try to select an f bound for the next iteration that doubles the number of nodes

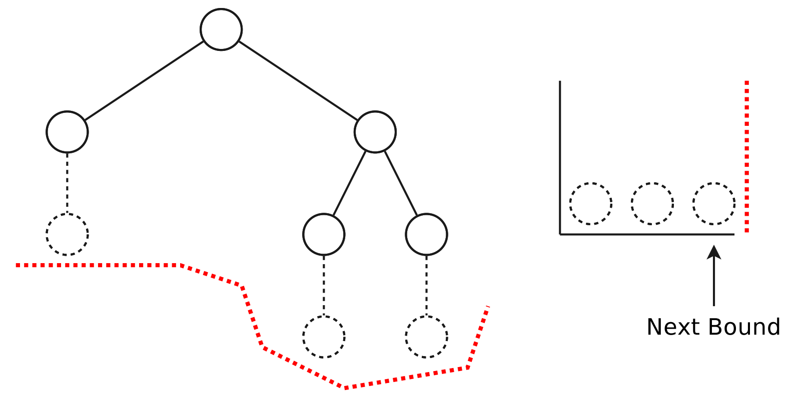
- Histogram of f values of pruned nodes
  - These nodes are candidates for the next iteration



- Find a bound using this histogram

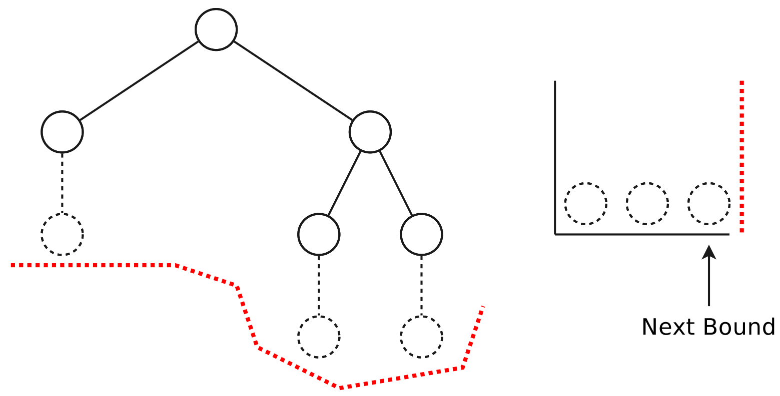
Two Problems with IDA*$_{CR}$

1. What if there are too few pruned nodes?

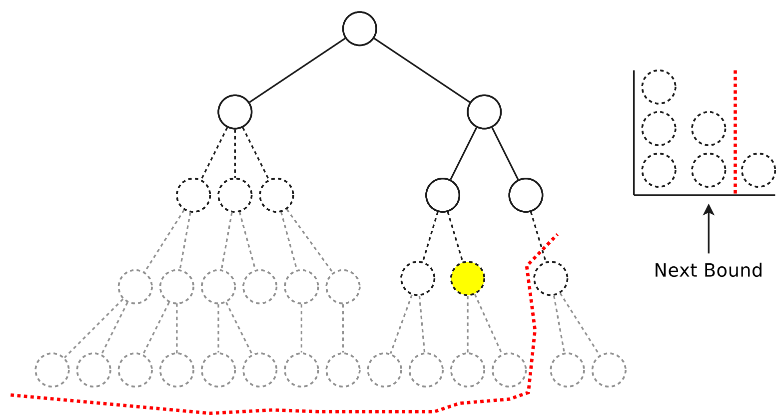


Two Problems with IDA*$_{CR}$

1. What if there are too few pruned nodes?

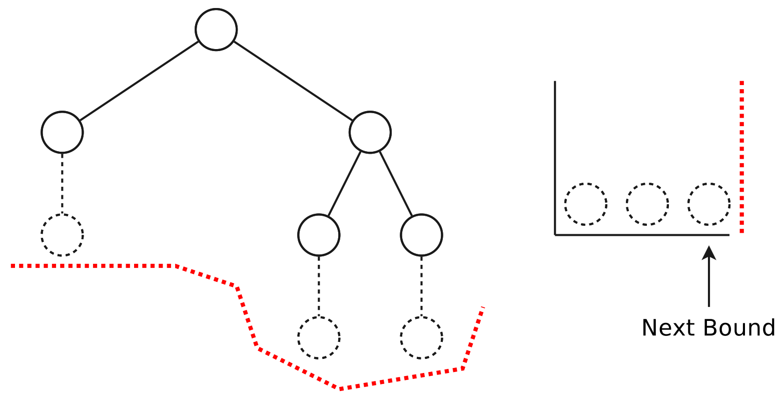


2. What if there are too many new nodes?

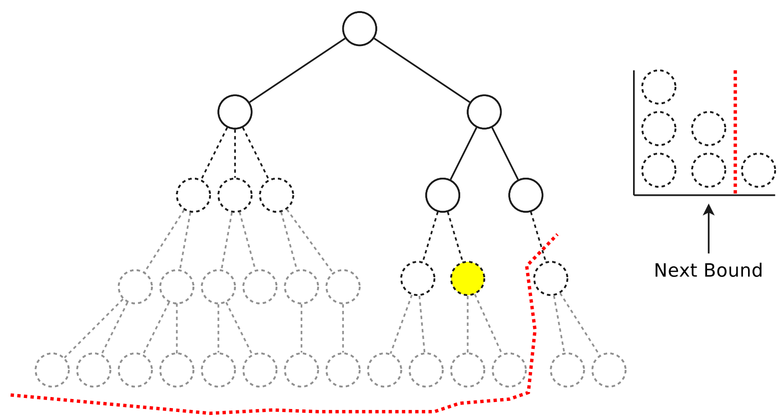


Two Problems with IDA*$_{CR}$

1. What if there are too few pruned nodes?



2. What if there are too many new nodes?



This simple model only looks ahead one layer

IDA* with Online Learning

IDA*$_{IM}$: New Approach for Setting IDA* Bounds

- Learn how costs change during previous iterations of search
- Incremental: allows for extrapolation

IDA*$_{\text{IM}}$: New Approach for Setting IDA* Bounds

- Learn how costs change during previous iterations of search
- Incremental: allows for extrapolation

- Between iterations, use the model to build a histogram
- $f(i+1) = f(i) + \Delta f$

See the paper for extensive details

IDA*$_{\text{IM}}$: New Approach for Setting IDA* Bounds

- Learn how costs change during previous iterations of search
- Incremental: allows for extrapolation

- Between iterations, use the model to build a histogram
- $f(i+1) = f(i) + \Delta f$

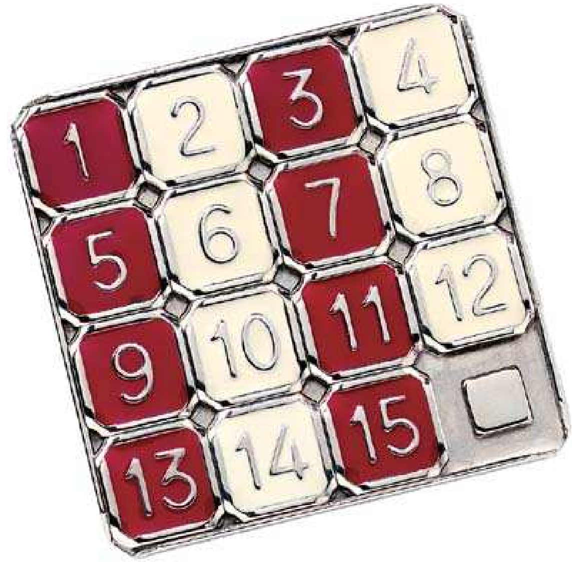
See the paper for extensive details

The model is flexible:

- Can predict a bound to give a desired number of nodes
- Can predict a number of nodes within a given bound

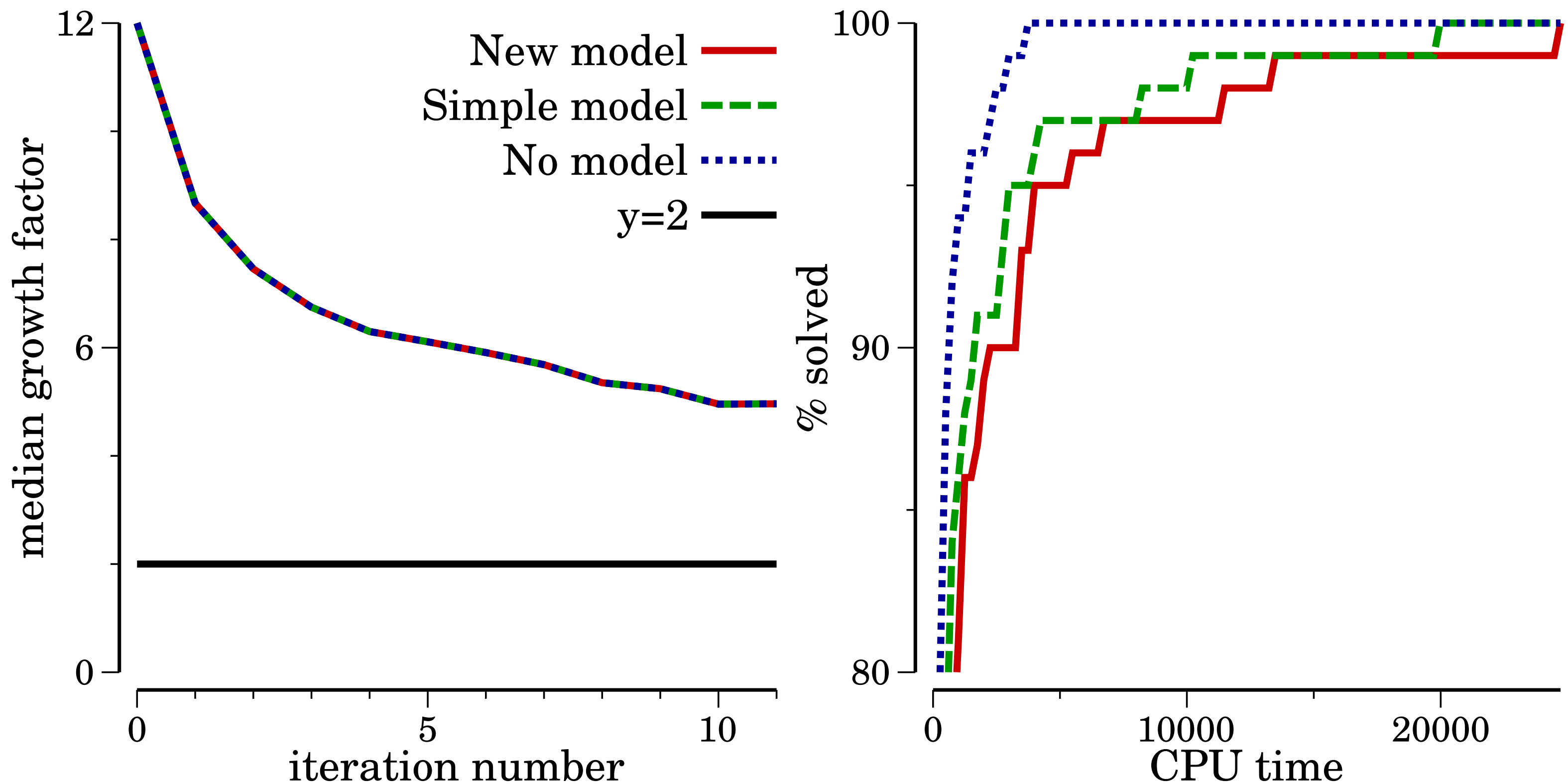
Empirical Evaluation

Sliding Tile Puzzle



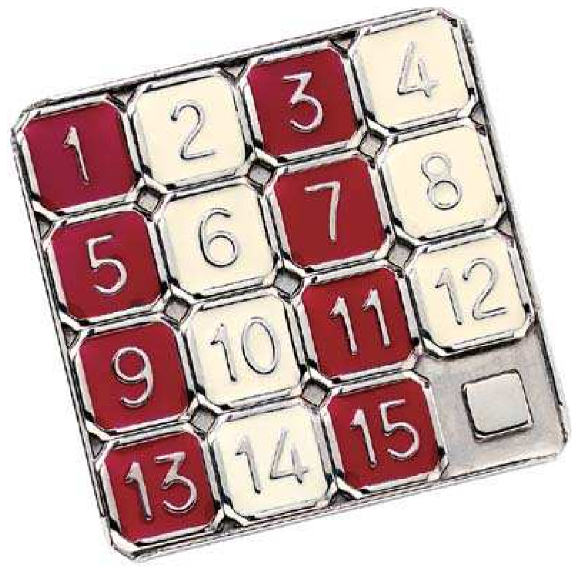
- A classic planning benchmark
- Easy to implement
- Models constrained logistics
- Exponential growth: IDA* performs well

Goal: Double growth between iterations



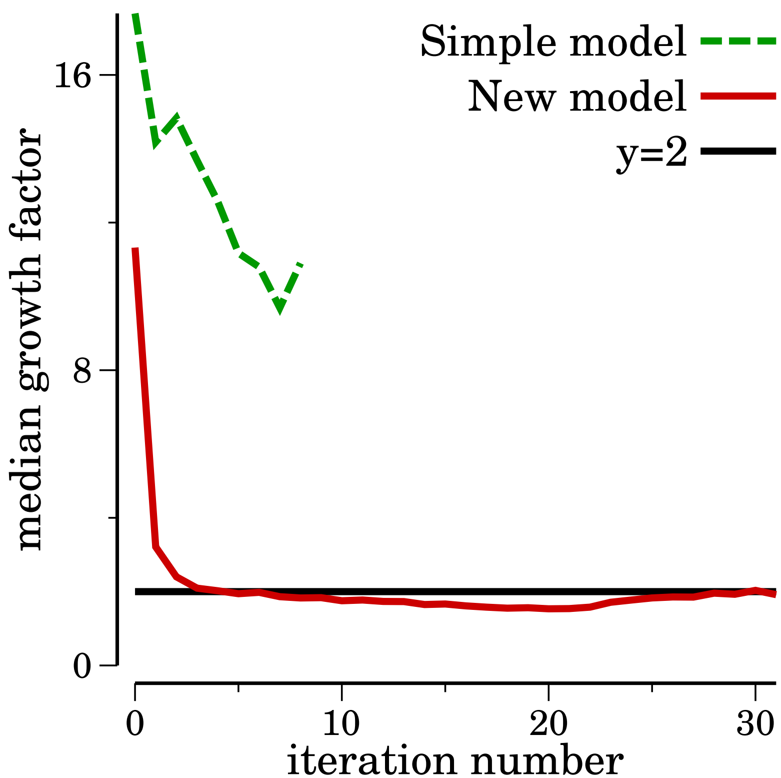
All algorithms give correct behavior but differ in overhead

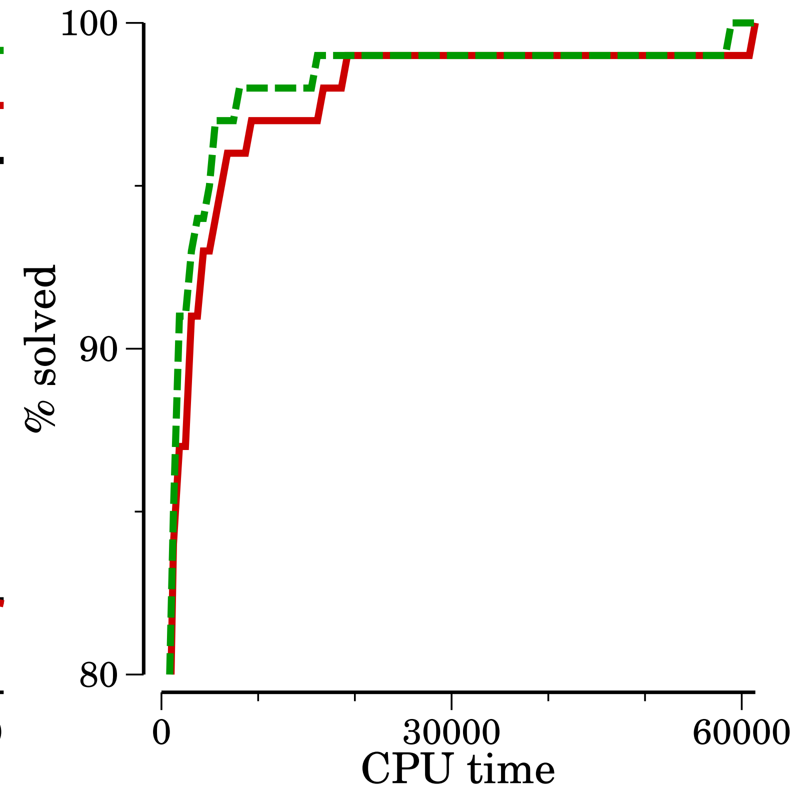
Square Root Sliding Tile Puzzle



- New action cost: each move costs $\sqrt{t}$, for each tile t
- Real-value edge costs
- Non-exponential growth: IDA* performs poorly

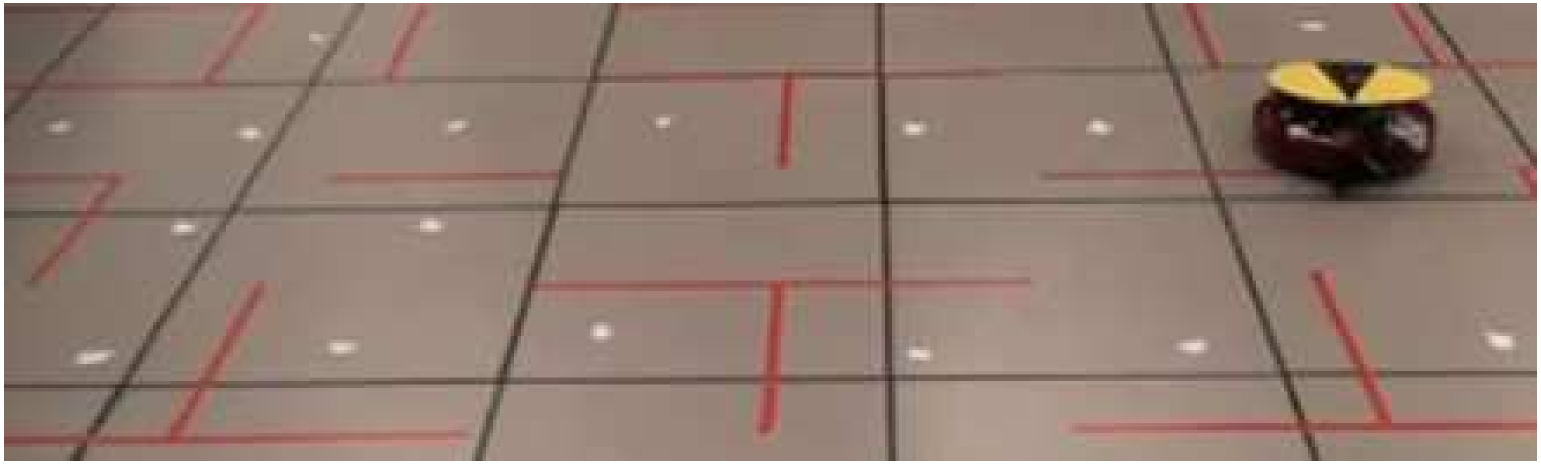
Controlling IDA* Search: On-line learning





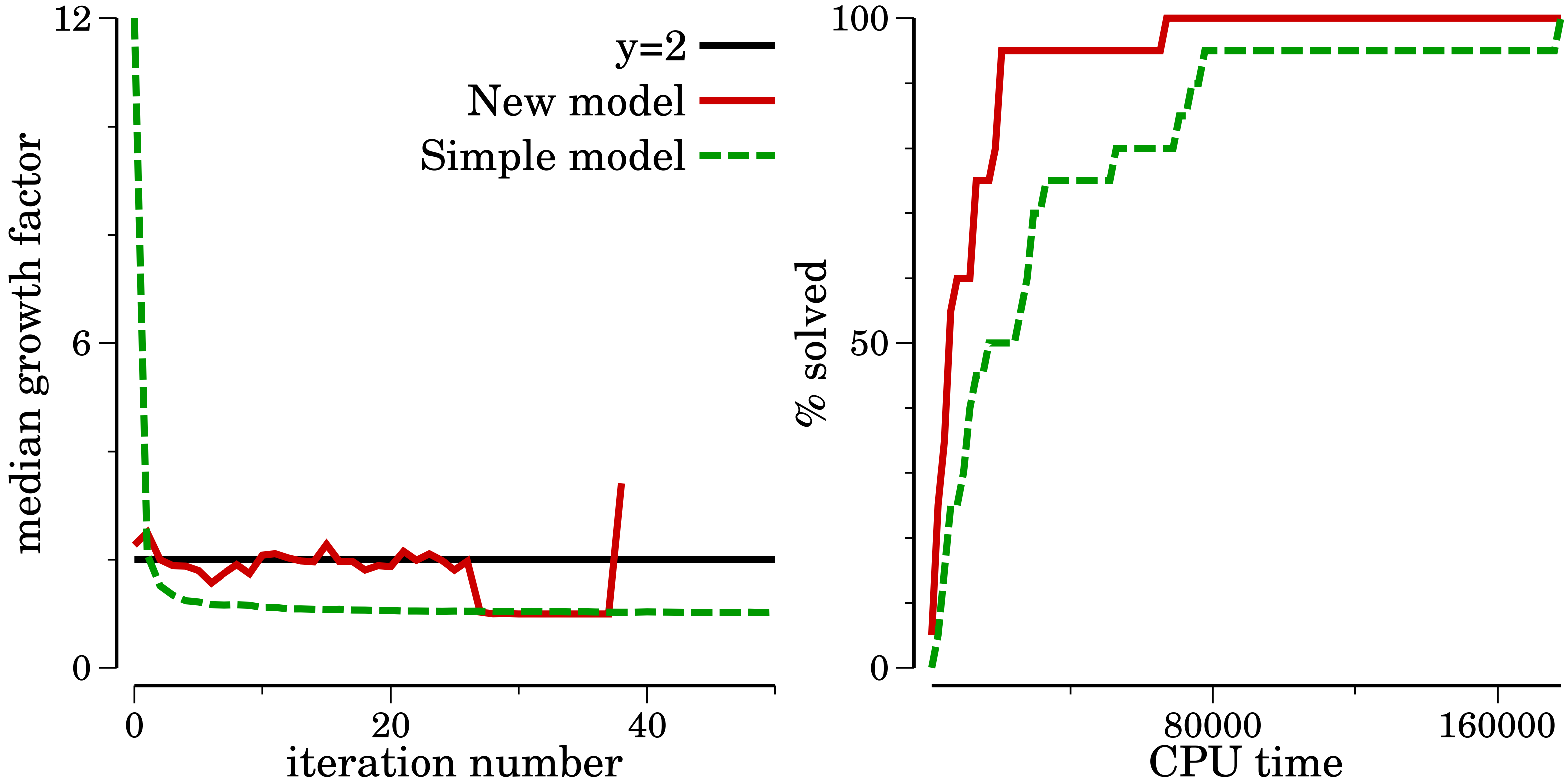
The new model predicts accurately, the simple one does not

Vacuum Maze



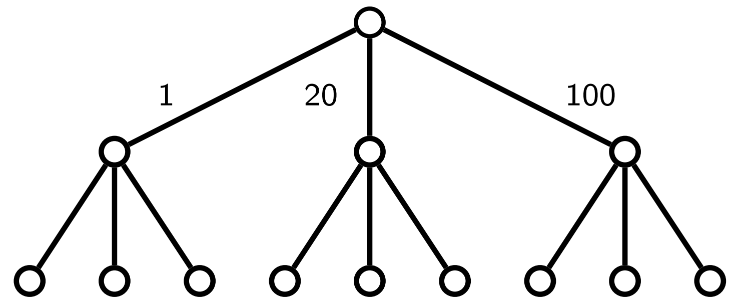
- Find and cleanup pieces of dirt in a maze
- TSP combined with maze navigation
- Non-exponential growth: IDA* performs poorly
- Very few pruned successors: simple model performs poorly

Controlling IDA* Search: On-line learning

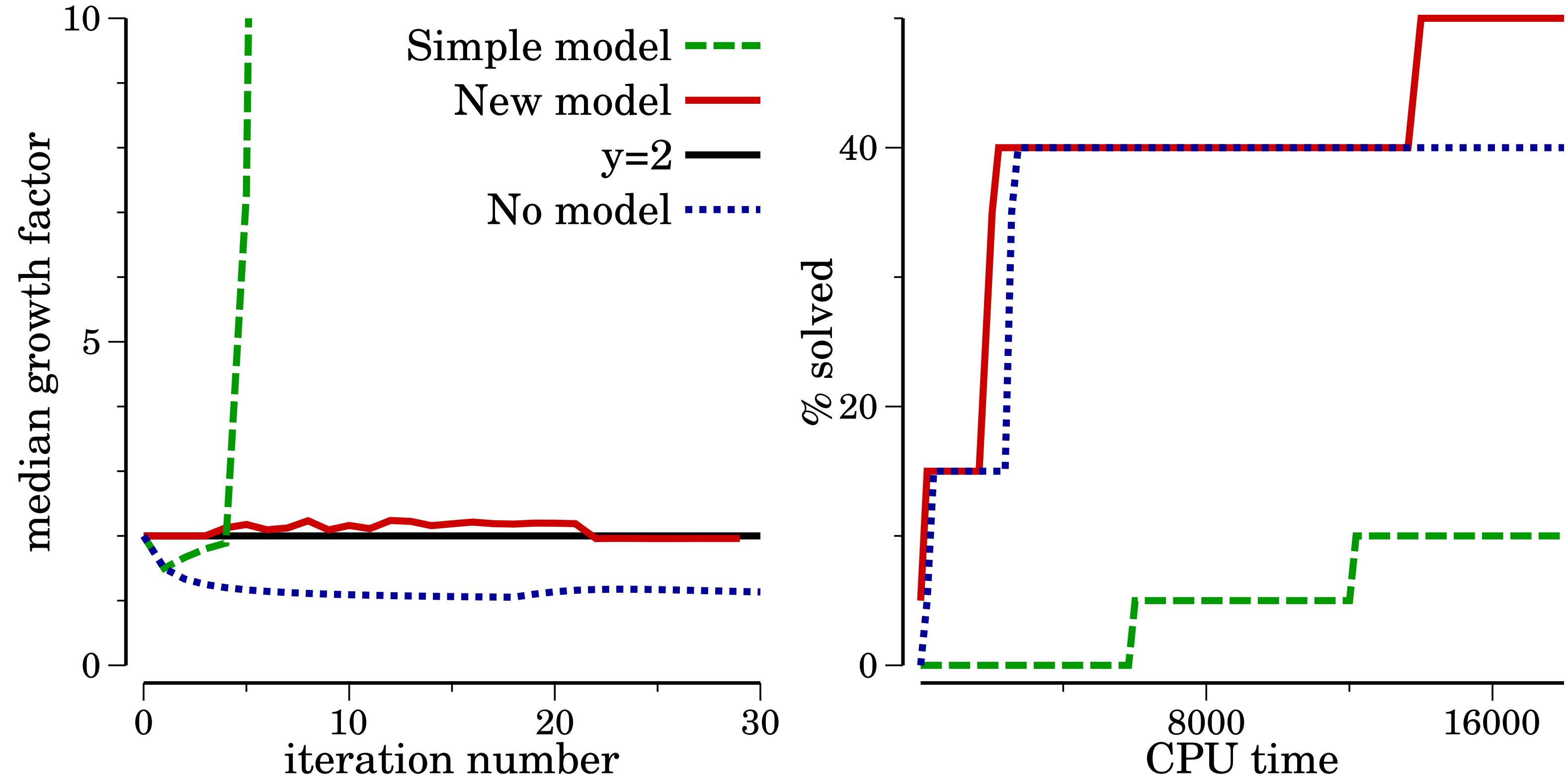


Too few successors so the simple model fails

Synthetic Uniform Tree



- Each node has 3 children with costs 1, 20 and 100
- Solution is at depth 22
- Solution is path is random 1 and 20 cost branches
- Non-exponential growth: IDA* performs poorly
- Lots of cheap actions: simple model performs poorly



Too many successors, the simple model fails

Conclusion

Tree search for Planning

- Infinite tree: requires a cost bound
- How do we set the bound?

New incremental model:

- Works with real-value costs
- Can be trained on-line

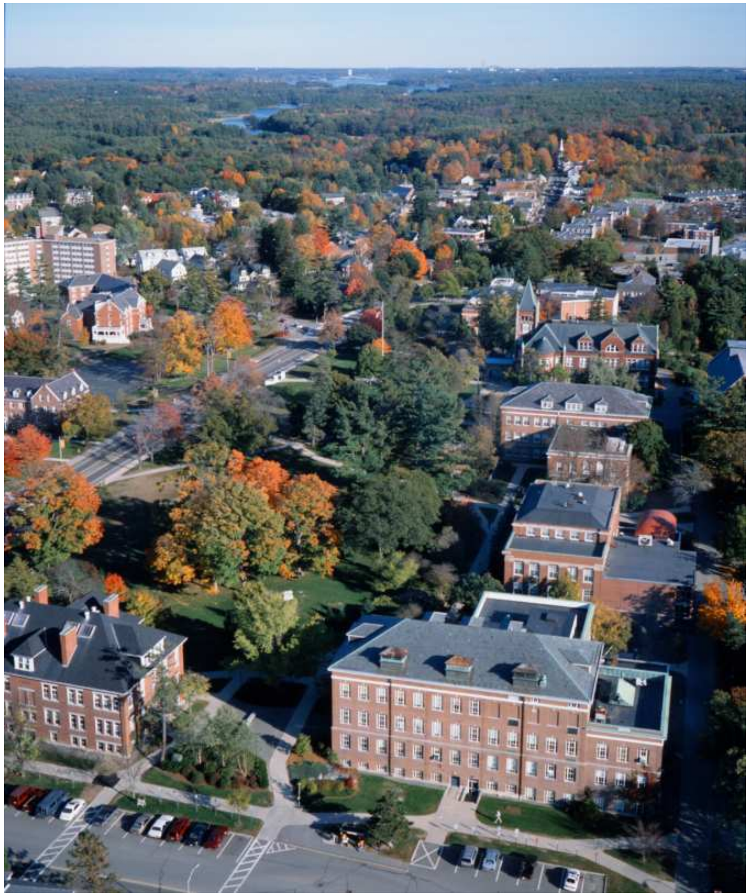
IDA*$_{\text{IM}}$:

- Uses the incremental model to choose bounds
- Performs well on all domains tested

On-line learning to control search applies to shortest-path search, too!

The University of New Hampshire

Tell your students to apply to grad school in CS at UNH!



- friendly faculty
- funding
- individual attention
- beautiful campus
- easy access to Boston
- strong in AI, infoviz, networking

Additional Slides

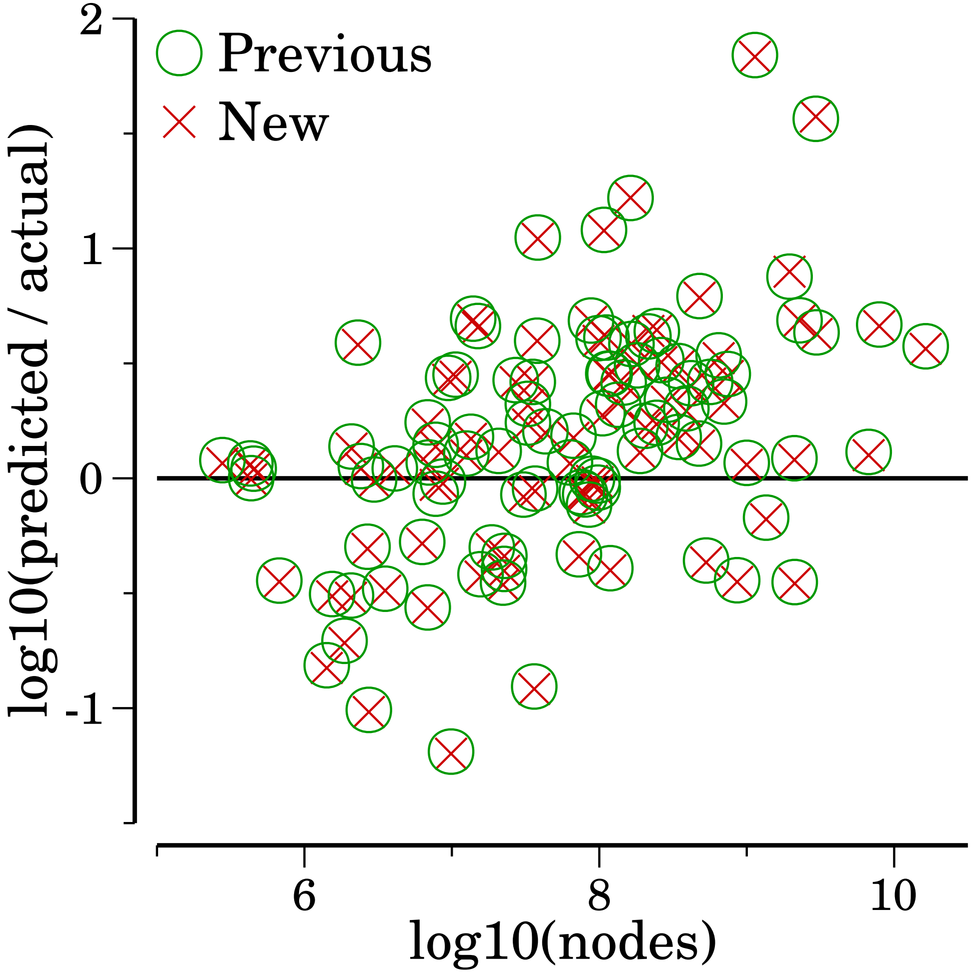
Off-line Training

The model can be used off-line too: estimate node expansions to solve a problem

- Previous technique is off-line only
- Previous technique works with integer costs only

When restricted to integer costs, how does our new approach compare?

Predict the number of nodes visited



IDA*$_{IM}$: New Approach for Setting IDA* Bounds

- Learn how costs change during previous iterations of search
- Incremental: allows for extrapolation

IDA*$_{IM}$: New Approach for Setting IDA* Bounds

- Learn how costs change during previous iterations of search
- Incremental: allows for extrapolation

- Between iterations, use the model to build a histogram
- $f(i+1) = f(i) + \Delta f$

See the paper for extensive details

IDA*$_{IM}$: New Approach for Setting IDA* Bounds

- Learn how costs change during previous iterations of search
- Incremental: allows for extrapolation

- Between iterations, use the model to build a histogram
- $f(i+1) = f(i) + \Delta f$

See the paper for extensive details

The model is flexible:

- Can predict a bound to give a desired number of nodes
- Can predict a number of nodes within a given bound

A High-level Description of the Model

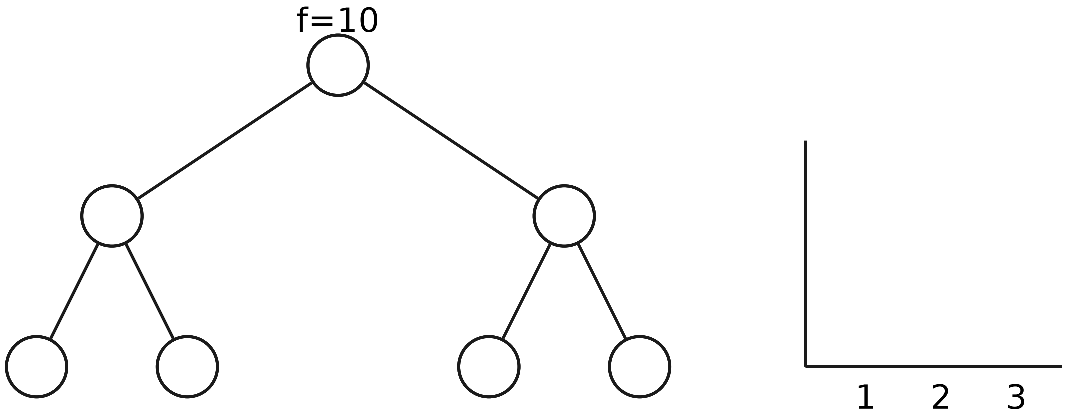
Learn how f tends to change from parent to child

- Use a histogram to track distribution of Δf

A High-level Description of the Model

Learn how f tends to change from parent to child

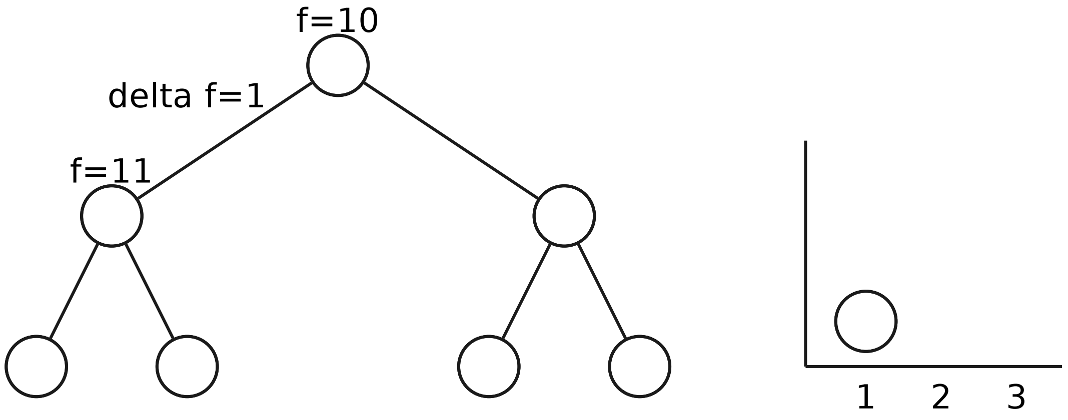
- Use a histogram to track distribution of Δf



A High-level Description of the Model

Learn how f tends to change from parent to child

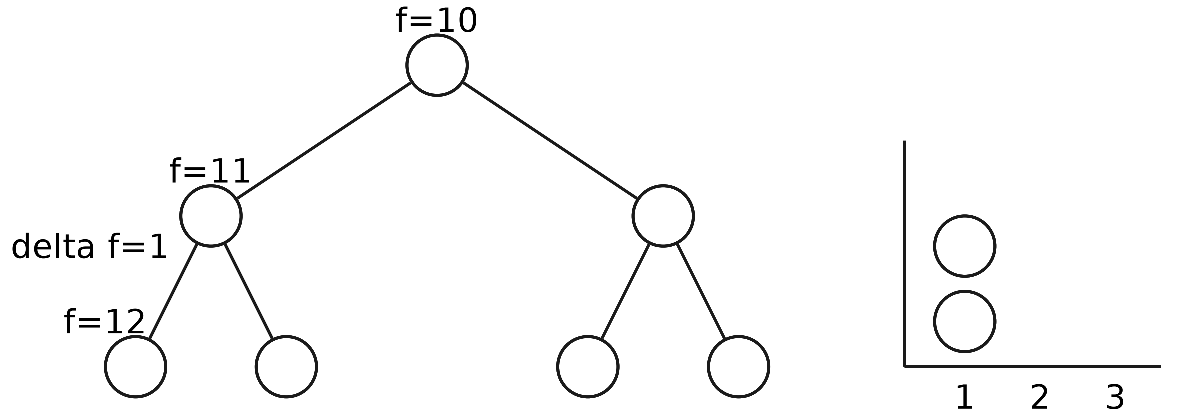
- Use a histogram to track distribution of Δf



A High-level Description of the Model

Learn how f tends to change from parent to child

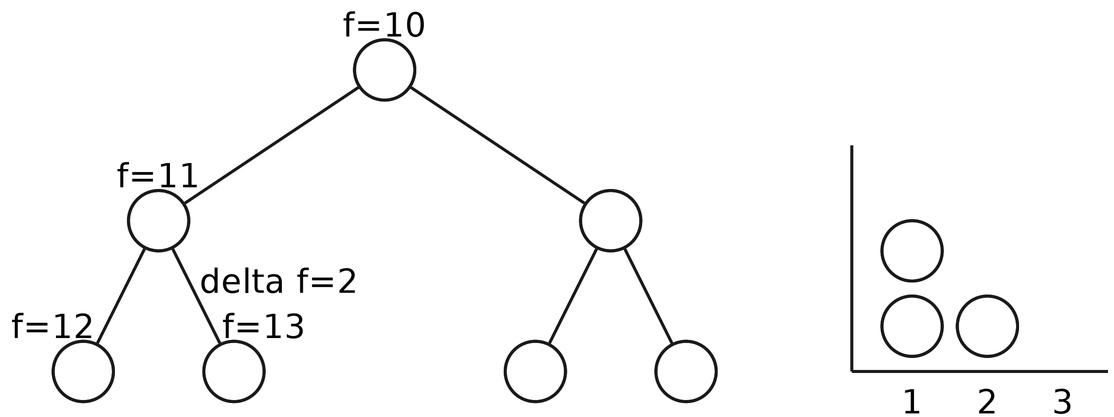
- Use a histogram to track distribution of Δf



A High-level Description of the Model

Learn how f tends to change from parent to child

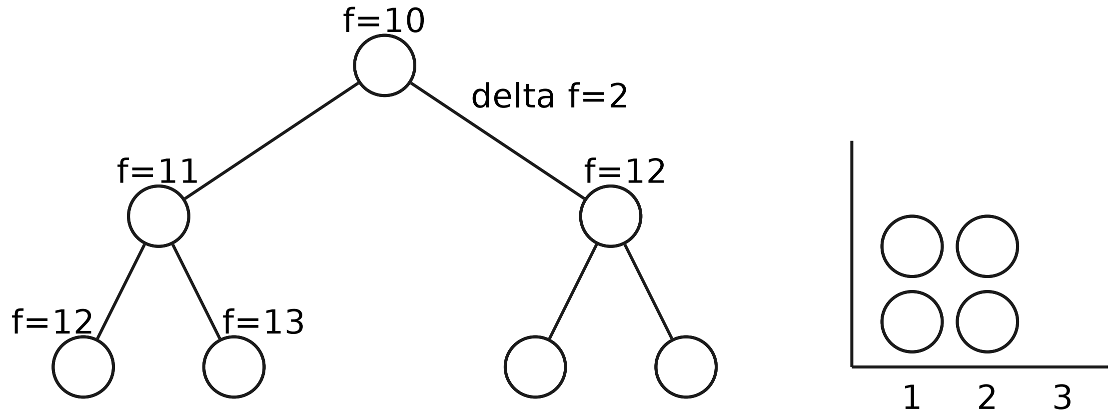
- Use a histogram to track distribution of Δf



A High-level Description of the Model

Learn how f tends to change from parent to child

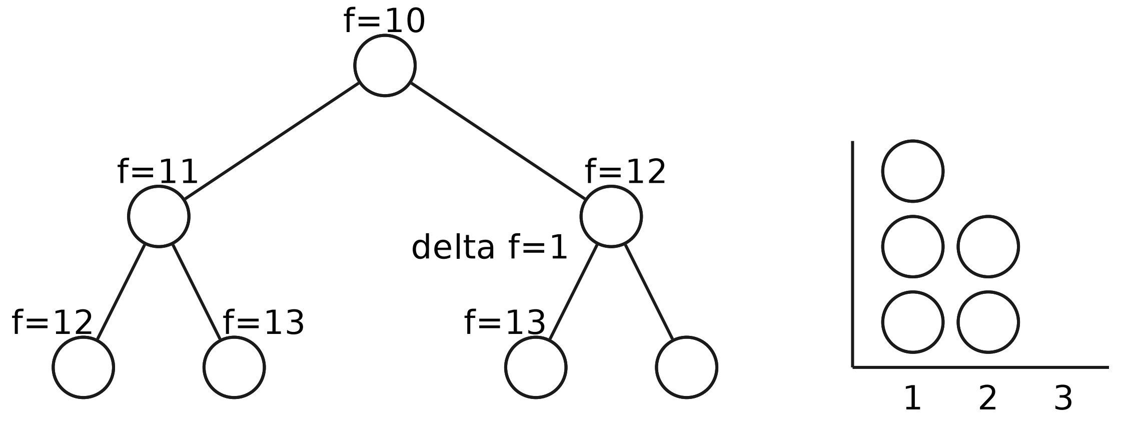
- Use a histogram to track distribution of Δf



A High-level Description of the Model

Learn how f tends to change from parent to child

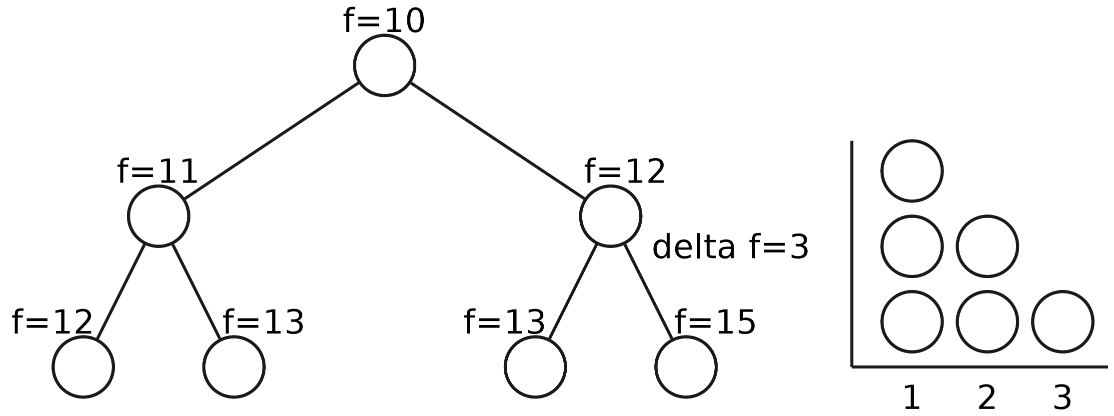
- Use a histogram to track distribution of Δf



A High-level Description of the Model

Learn how f tends to change from parent to child

- Use a histogram to track distribution of Δf



A High-level Description of the Model

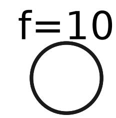
Between iterations, estimate f distribution in the search space

- Beginning from the initial node's f value
- Extrapolate f values of successors using the Δf distrobution

A High-level Description of the Model

Between iterations, estimate f distribution in the search space

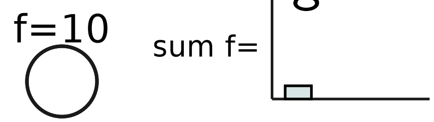
- Beginning from the initial node's f value
- Extrapolate f values of successors using the Δf distrobution



A High-level Description of the Model

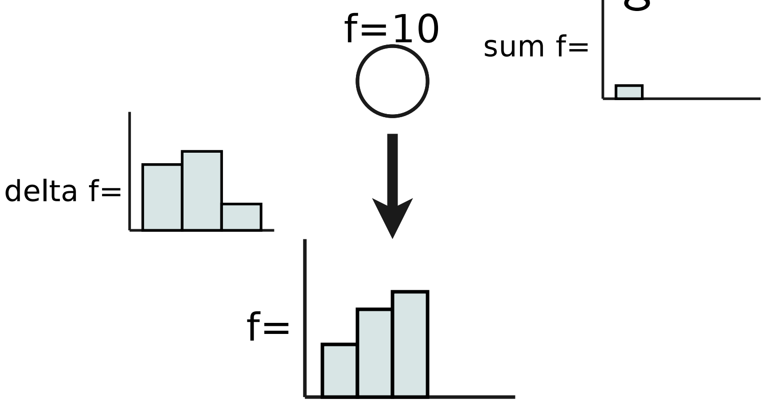
Between iterations, estimate f distribution in the search space

- Beginning from the initial node's f value
- Extrapolate f values of successors using the Δf distrobution



Between iterations, estimate f distribution in the search space

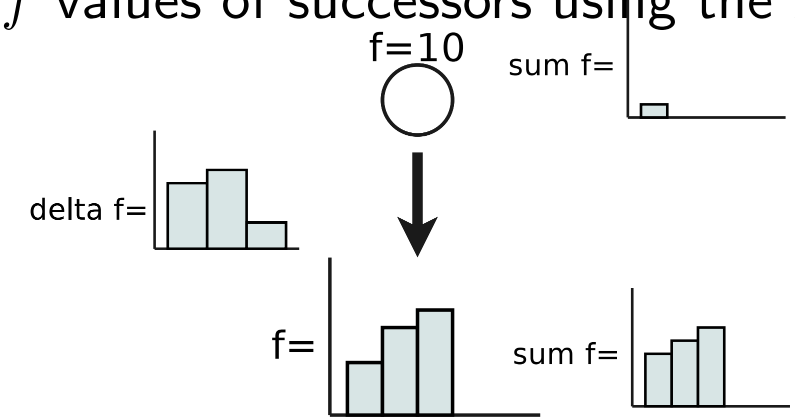
- Beginning from the initial node's f value
- Extrapolate f values of successors using the Δf distrobution



A High-level Description of the Model

Between iterations, estimate f distribution in the search space

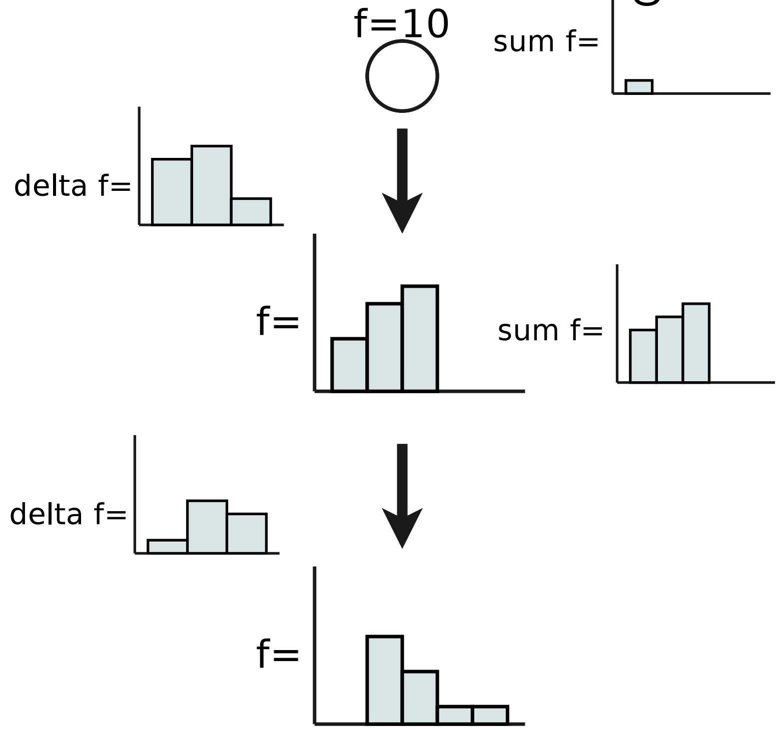
- Beginning from the initial node's f value
- Extrapolate f values of successors using the Δf distrobution



A High-level Description of the Model

Between iterations, estimate f distribution in the search space

- Beginning from the initial node's f value
- Extrapolate f values of successors using the Δf distrobution



A High-level Description of the Model

Between iterations, estimate f distribution in the search space

- Beginning from the initial node's f value
- Extrapolate f values of successors using the Δf distrobution