Faster than Weighted A*: An Optimistic Approach to Bounded Suboptimal Search

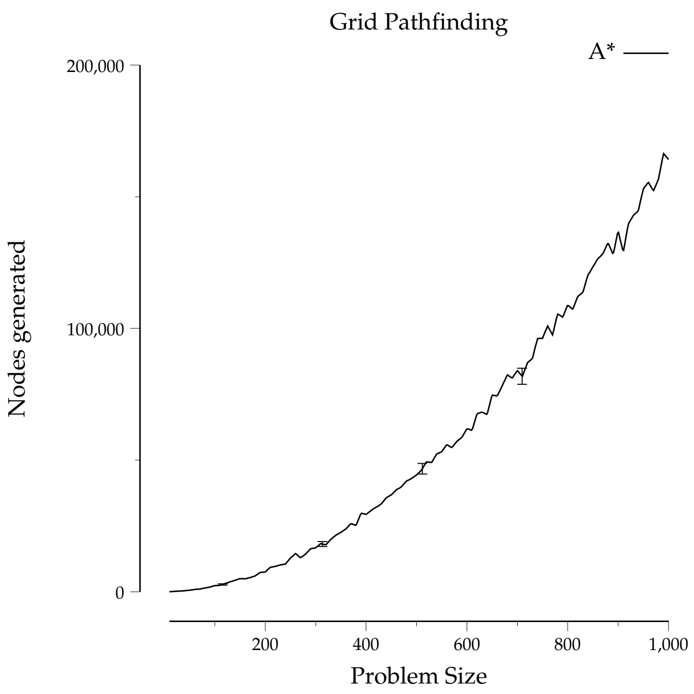
Jordan Thayer and Wheeler Ruml

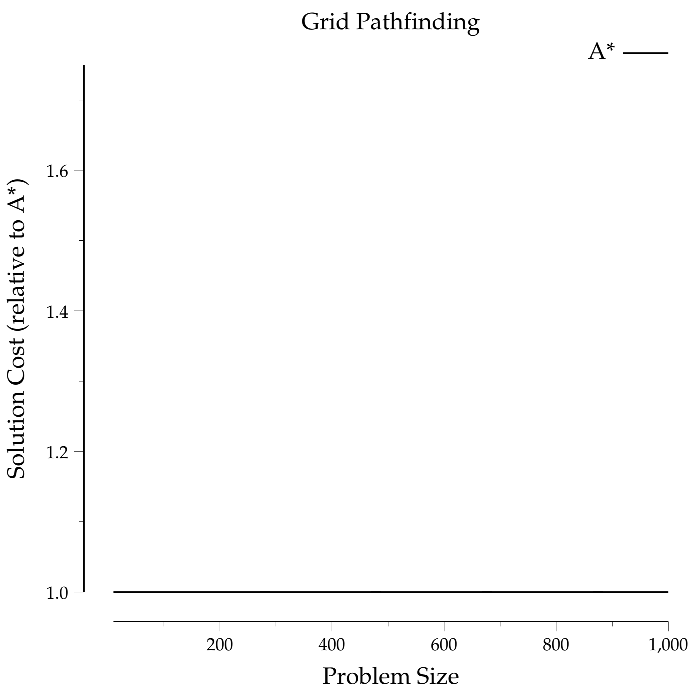
University of New Hampshire

{jtd7, ruml} at cs.unh.edu

Motivation

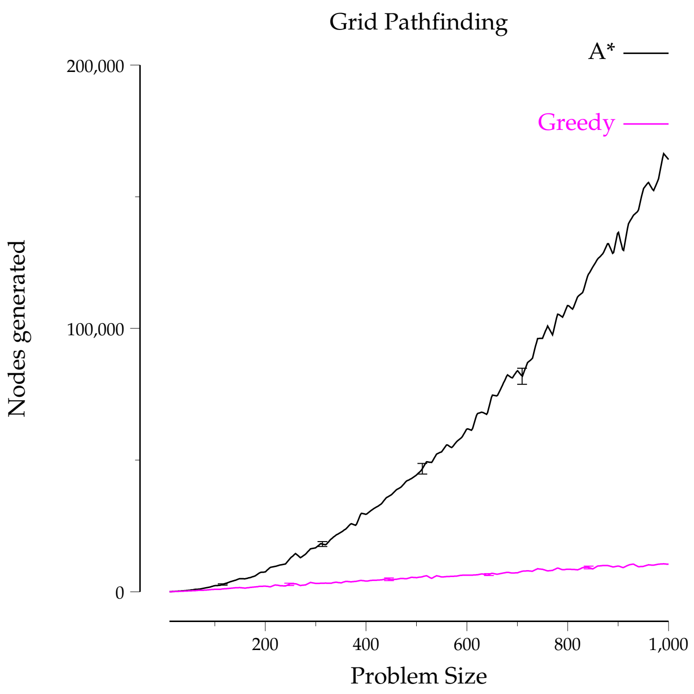
- Finding optimal solutions is prohibitively expensive.

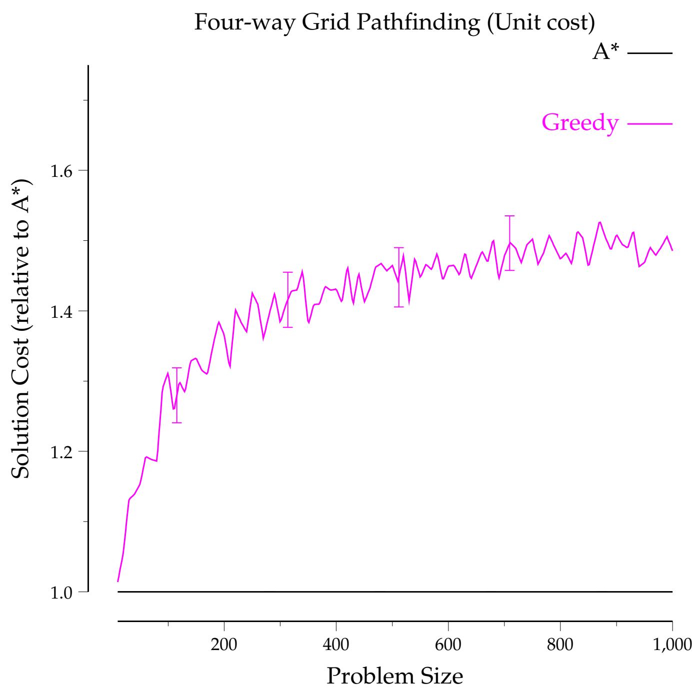




Motivation

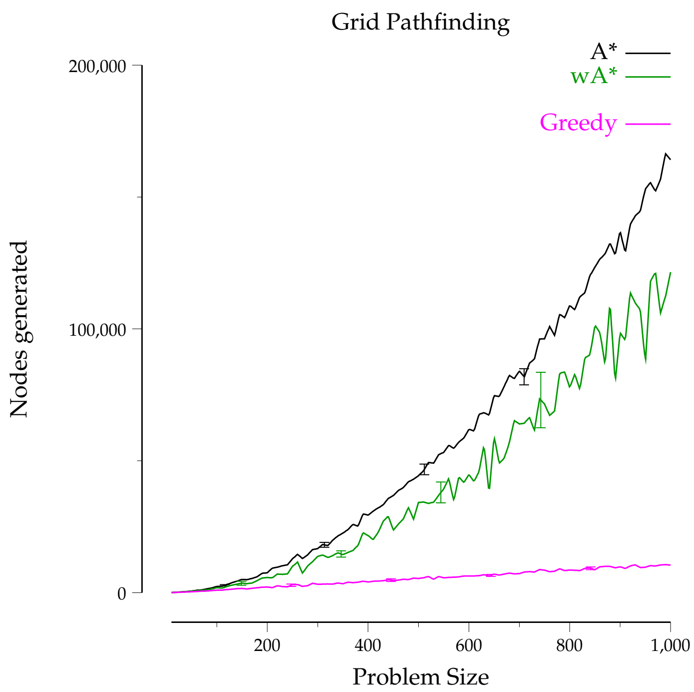
- Finding optimal solutions is prohibitively expensive.
- Greedy solutions can be arbitrarily bad.

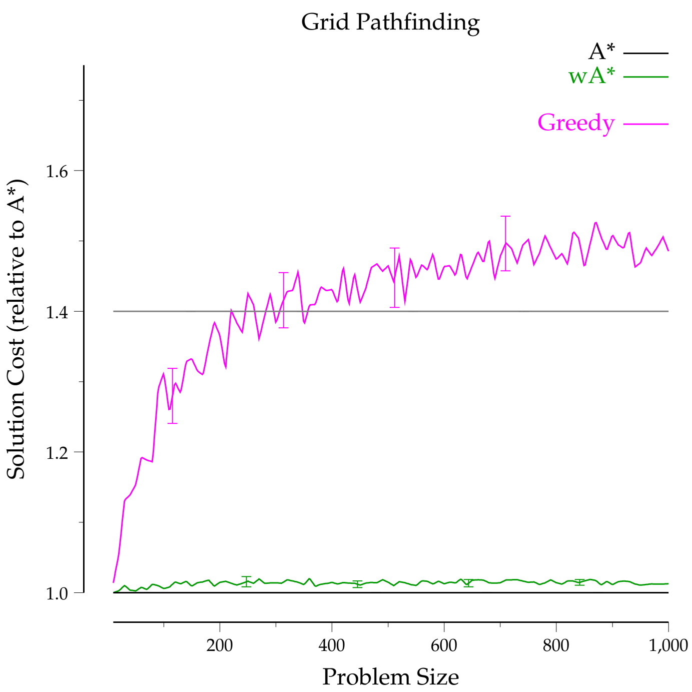




Motivation

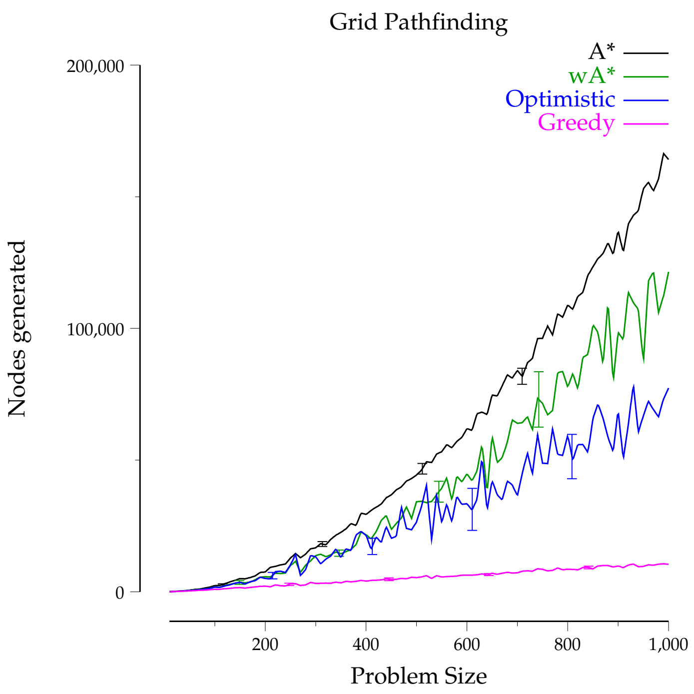
- Finding optimal solutions is prohibitively expensive.
- Greedy solutions can be arbitrarily bad.
- Weighted A* bounds suboptimality.

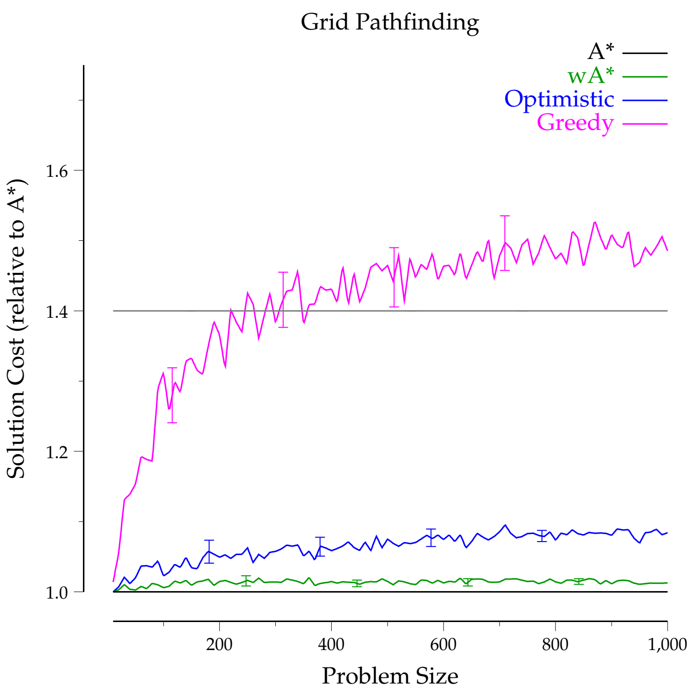




Motivation

- Finding optimal solutions is prohibitively expensive.
- Greedy solutions can be arbitrarily bad.
- Weighted A* bounds suboptimality.
- Optimistic Search: faster search within the same bound.





Algorithm Overview

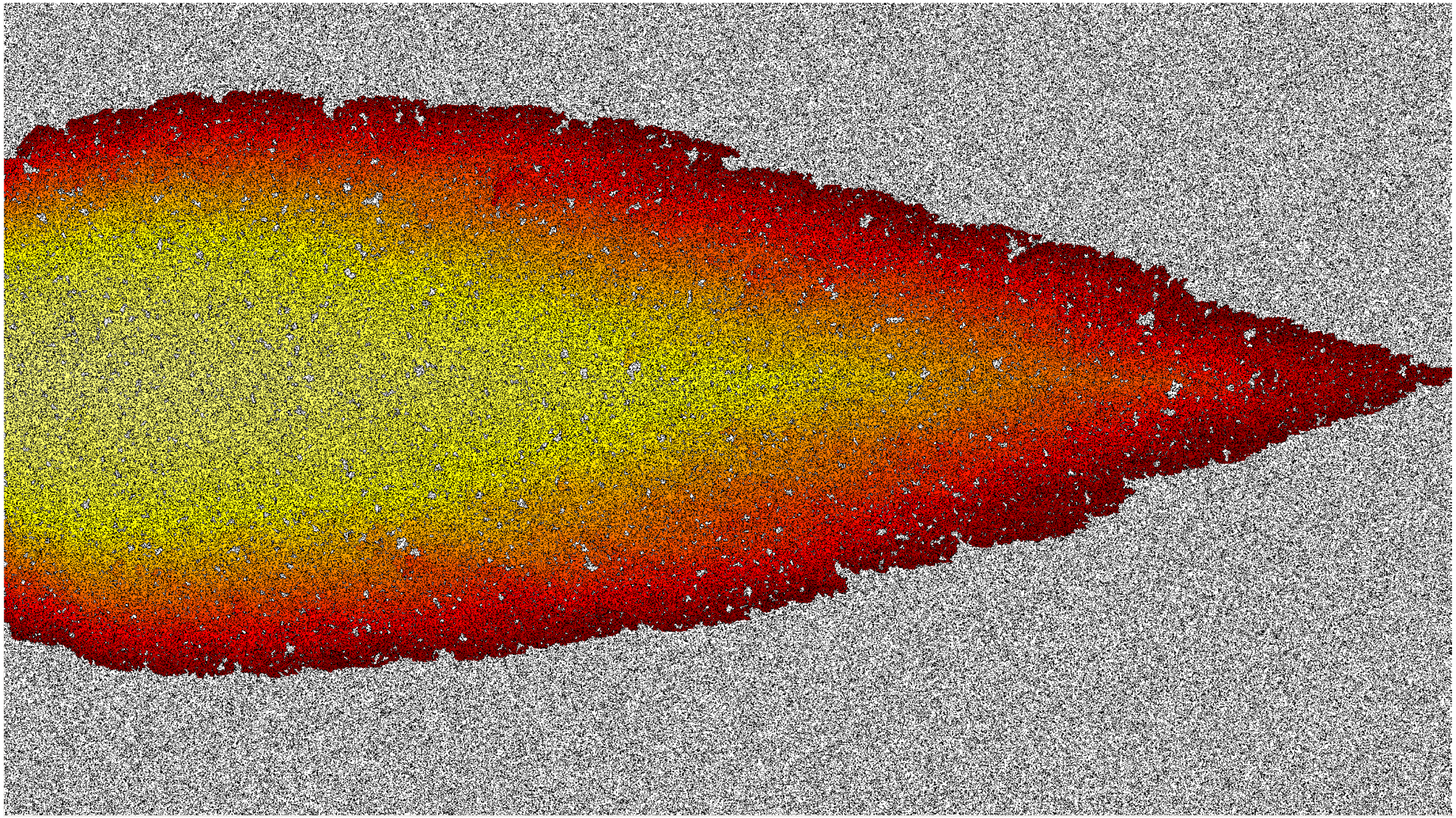
Talk Outline

- Algorithm Overview
  - Run weighted A^* with a weight higher than the bound.
  - Expand additional nodes to prove solution quality.
- The Greedy Search Phase
- The Cleanup Phase
- Empirical Evaluation
- Further Observations

Previous Algorithms: A^*

- A best first search expanding nodes in f order.
- $f(n) = g(n) + h(n)$

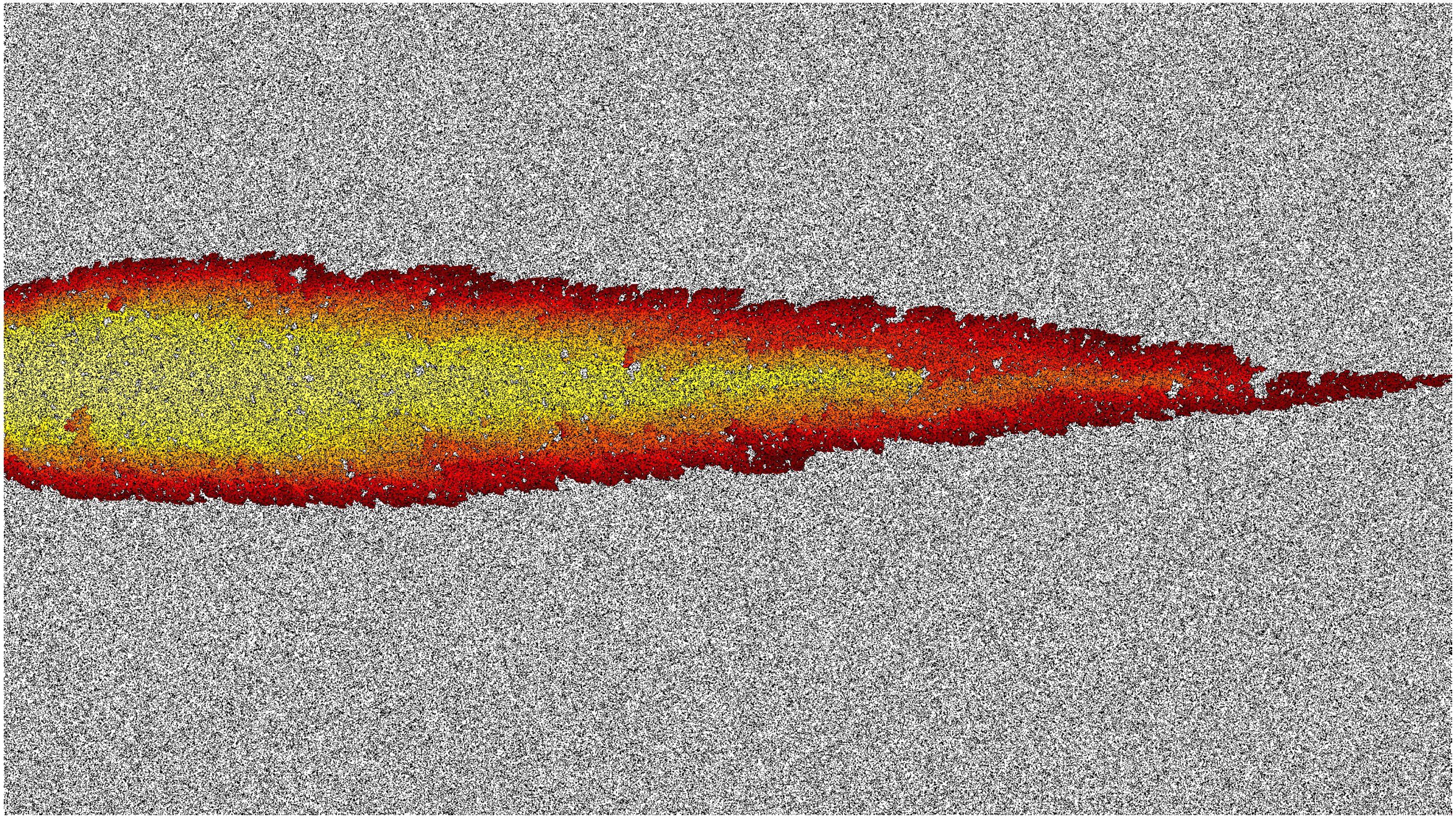
  If $h(n)$ is admissible, returns optimal solution.



Previous Algorithms: Weighted A^*

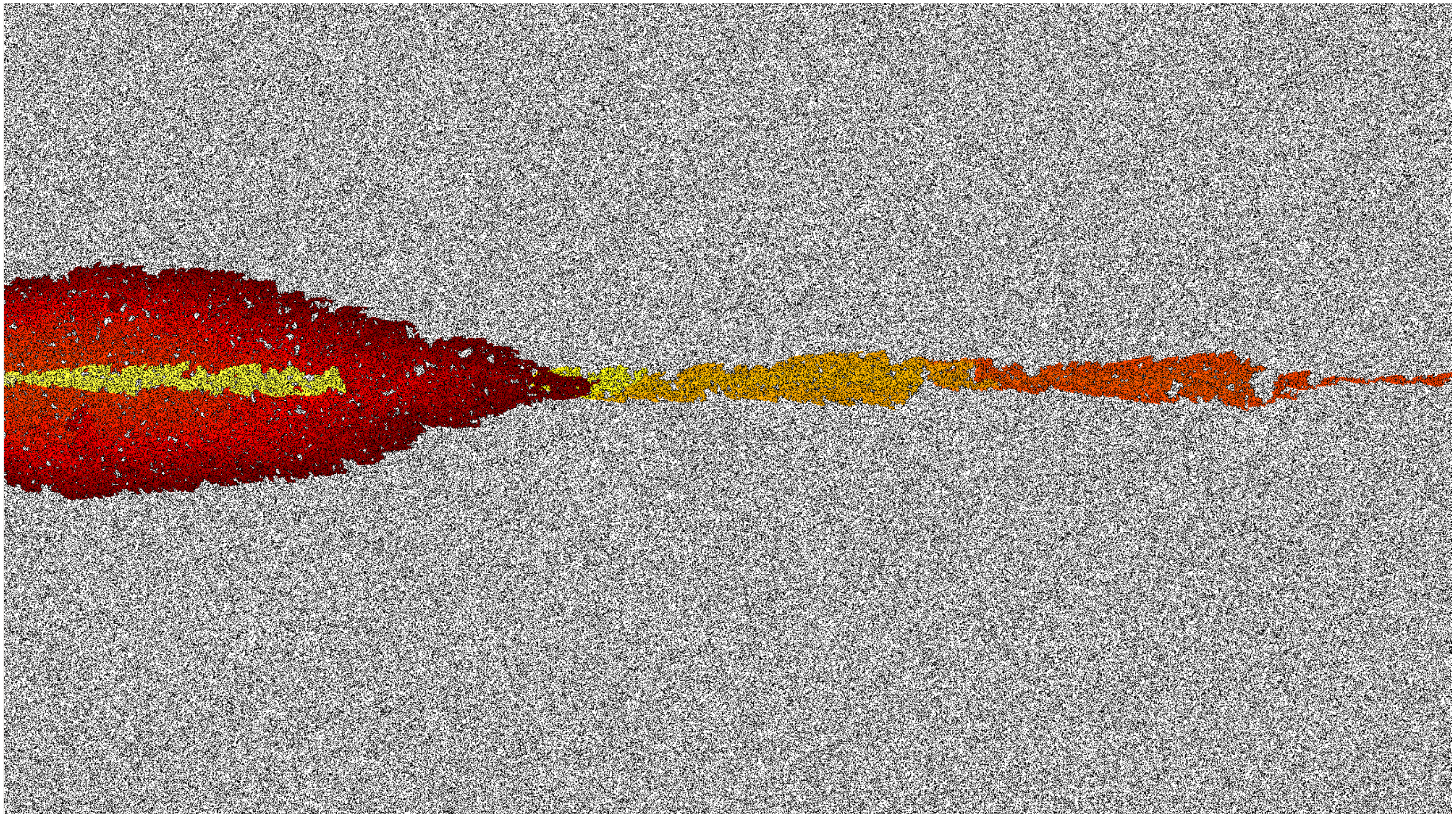
- A best first search expanding nodes in f' order.
- $f'(n) = g(n) + w \cdot h(n)$

  Solution quality bounded by w for admissible $h(n)$.



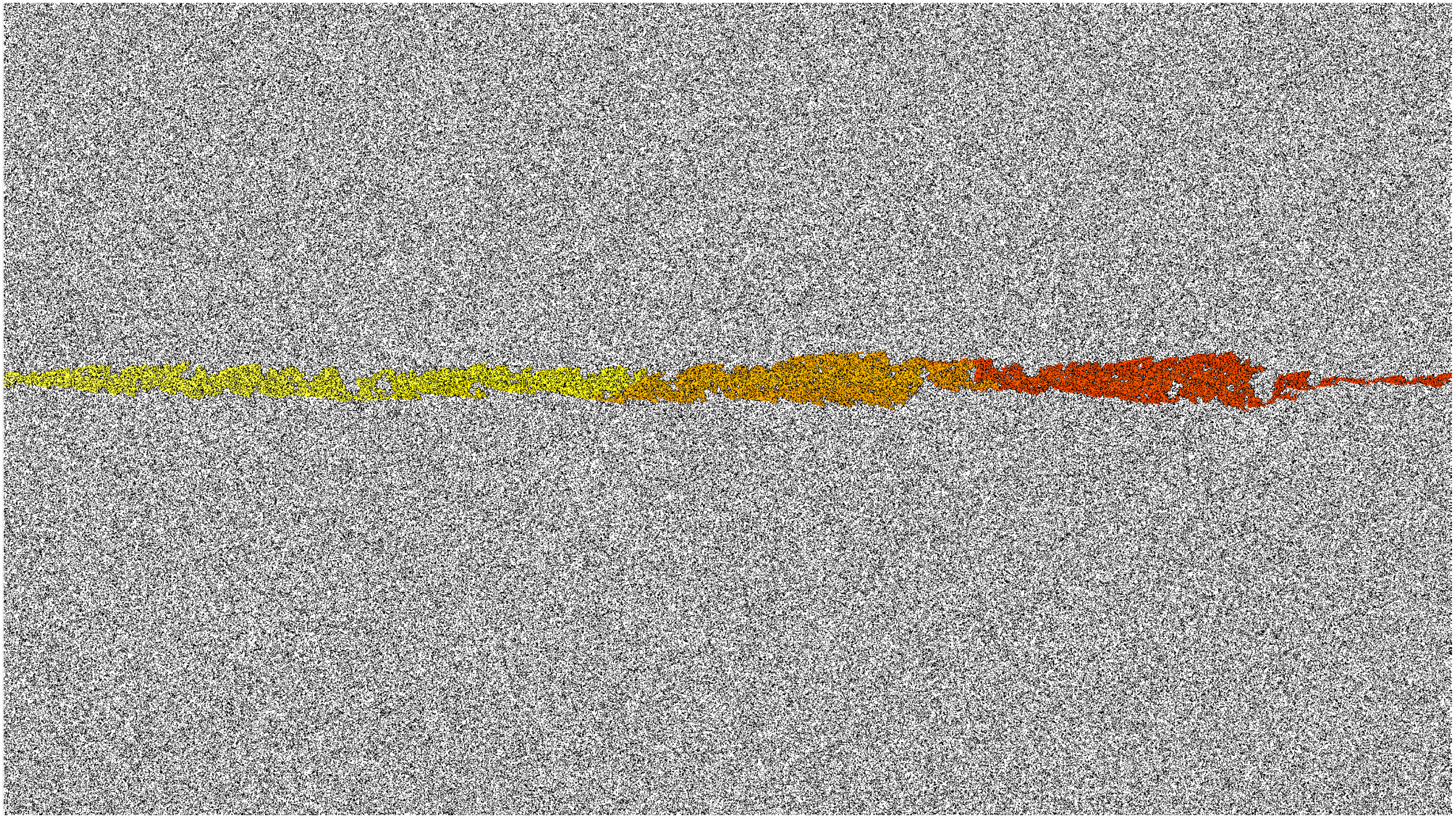
Optimistic Search: The Basic Idea

1. Run weighted A^* with a high weight.
2. Expand node with lowest f value after a solution is found.
   Continue until $w \cdot f_{min} > f(sol)$
   This 'clean up' guarantees solution quality.



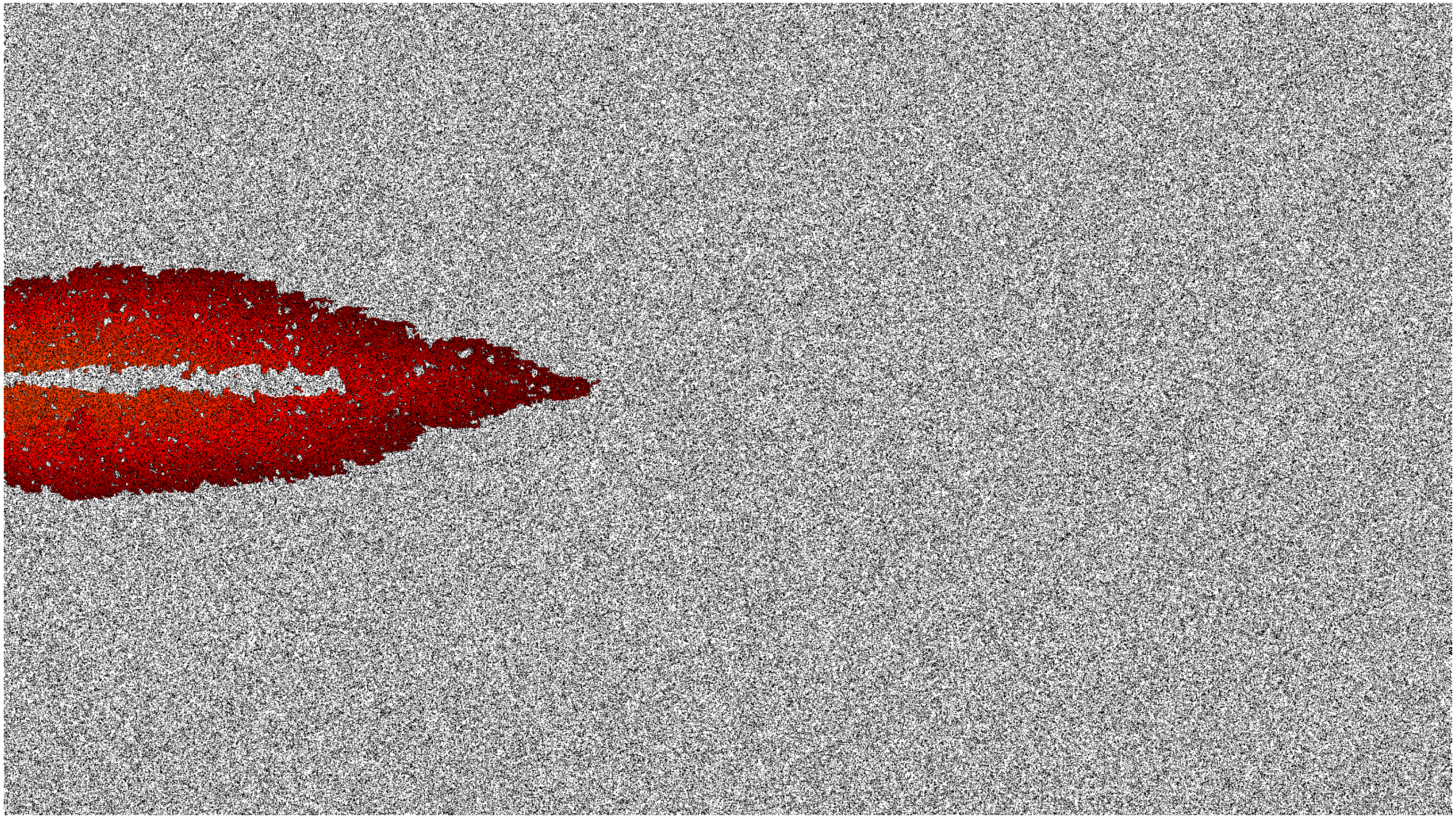
Optimistic Search: The Basic Idea

1. Run weighted A^* with a high weight.
2. Expand node with lowest f value after a solution is found.
   Continue until $w \cdot f_{min} > f(sol)$
   This 'clean up' guarantees solution quality.

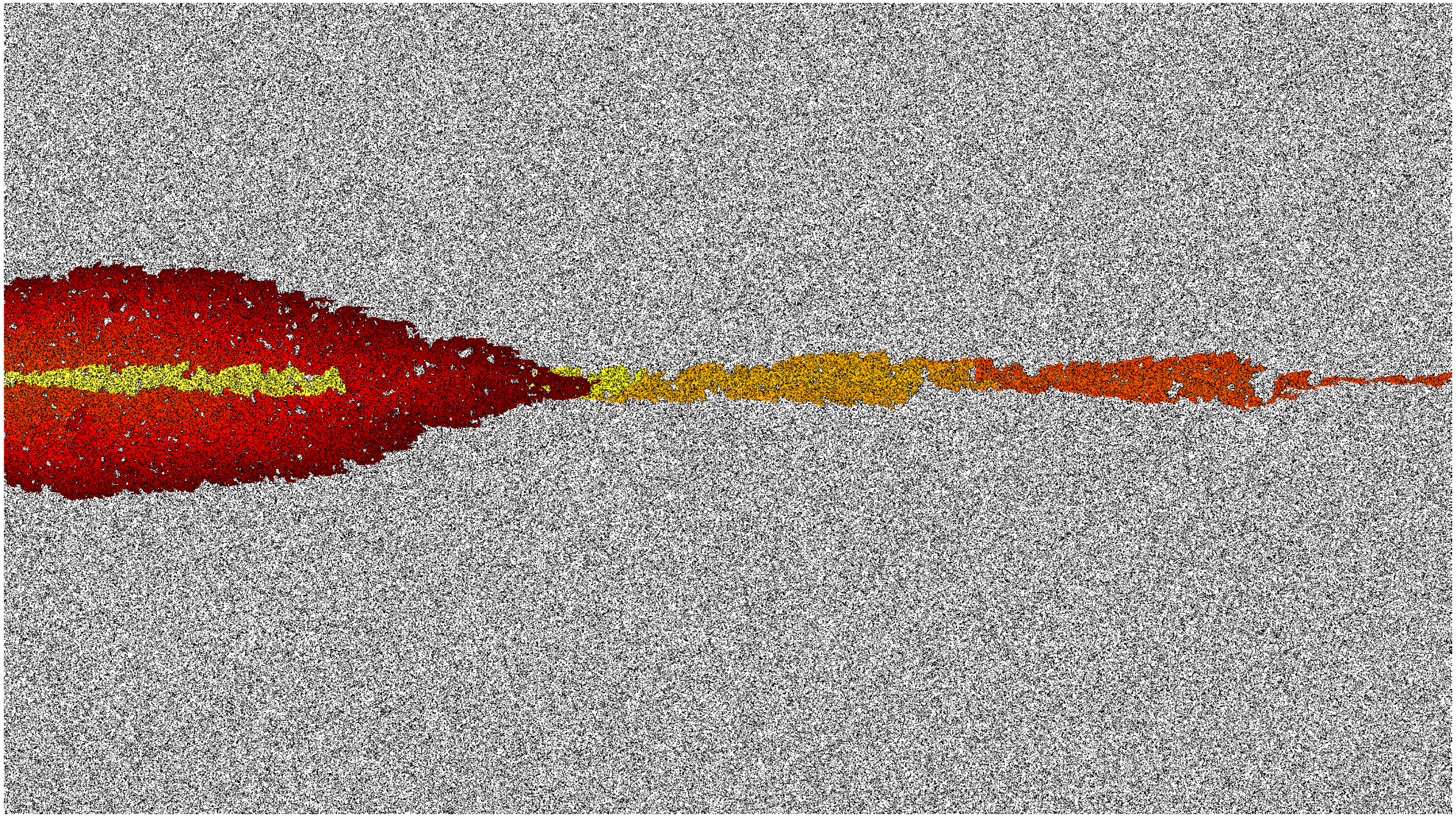


Optimistic Search: The Basic Idea

1. Run weighted A^* with a high weight.
2. Expand node with lowest f value after a solution is found.
   Continue until $w \cdot f_{min} > f(sol)$
   This 'clean up' guarantees solution quality.



1. Run weighted A^* with a high weight.
2. Expand node with lowest f value after a solution is found.
   Continue until $w \cdot f_{min} > f(sol)$
   This 'clean up' guarantees solution quality.



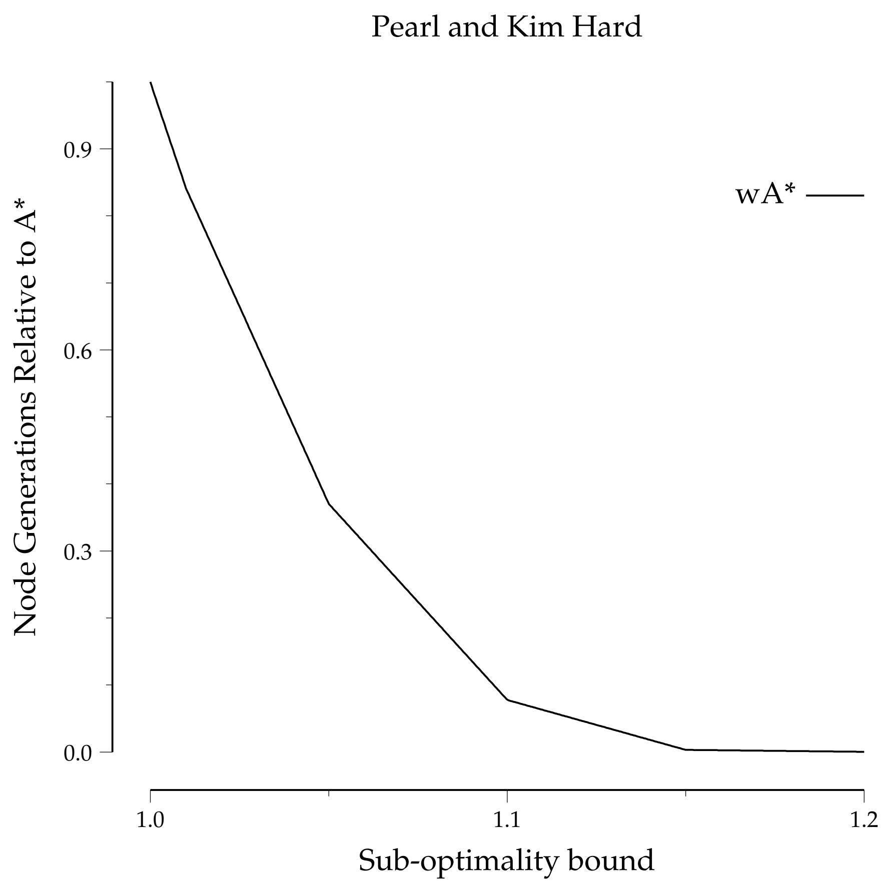
1: Greedy Phase

Talk Outline

- Algorithm Overview
- The Greedy Search Phase
  - Weighted A^* becomes faster as the bound grows.
  - Weighted A^* is often better than the bound.
- The Cleanup Phase
- Empirical Evaluation
- Further Observations

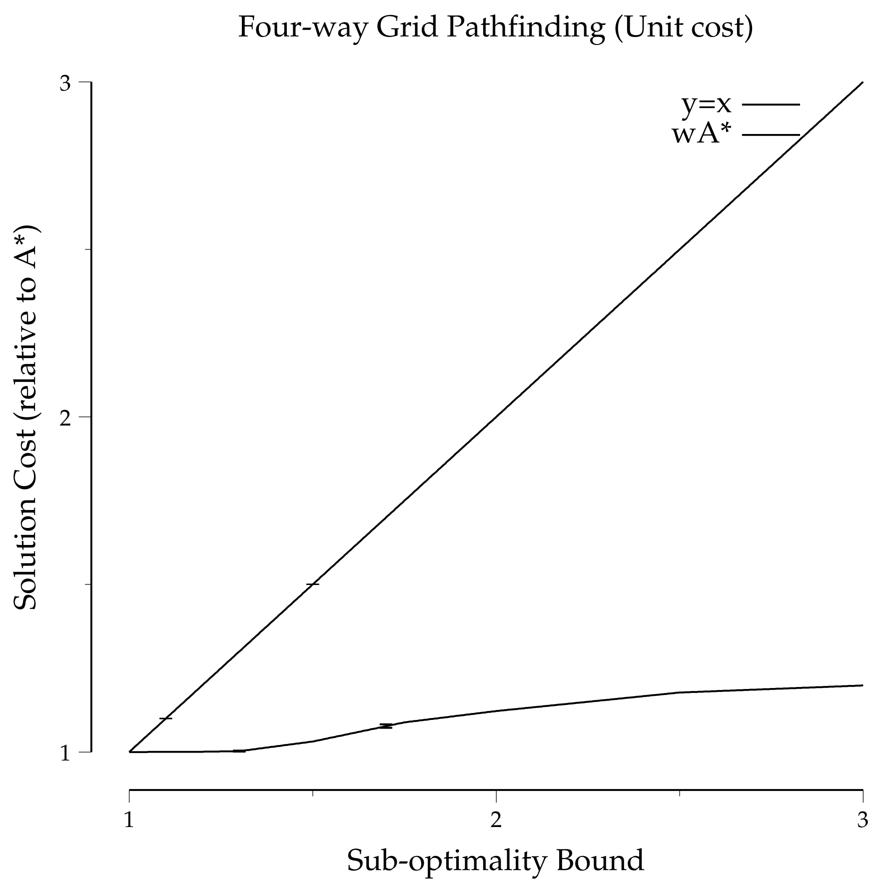
Large Bounds, Faster Solution

- wA^* returns solutions faster as the bound increases.



Weighted A^* is often better than the bound

- wA^* returns solutions better than the bound.

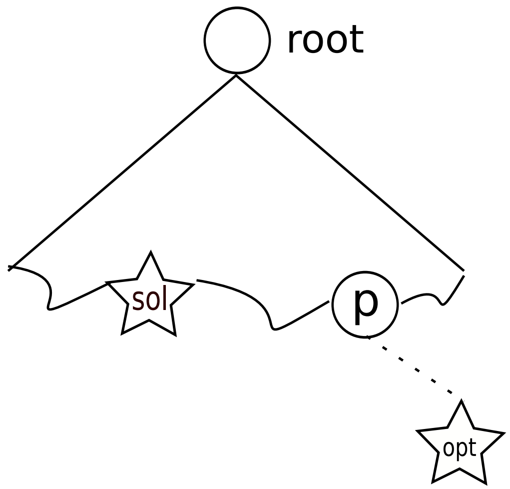


2: Cleanup Phase

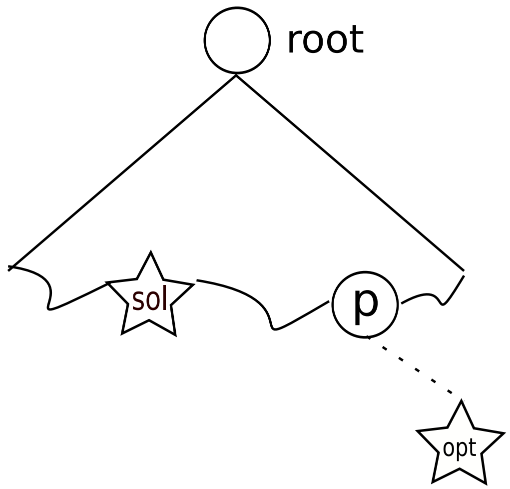
Talk Outline

- Algorithm Overview
- The Greedy Search Phase
- The Cleanup Phase
  - Expand additional nodes in f order.
  - Quit when the solution is provably within the bound.
- Empirical Evaluation
- Further Observations

Proving w-Admissibility



- p is the deepest node on an optimal path to opt.
- f_{min} is the node with the smallest f value.



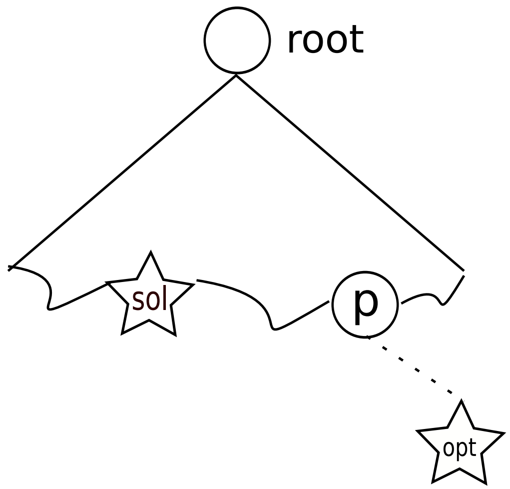
- p is the deepest node on an optimal path to opt.
- f_{min} is the node with the smallest f value.

$$
\begin{aligned}
f(p) &\leq f(opt) \\
f(f_{min}) &\leq f(p)
\end{aligned}
$$

f_{min} provides a lower bound on solution cost.

Determine f_{min} by priority queue sorted on f

Proving w-Admissibility



- p is the deepest node on an optimal path to opt.
- f_{min} is the node with the smallest f value.

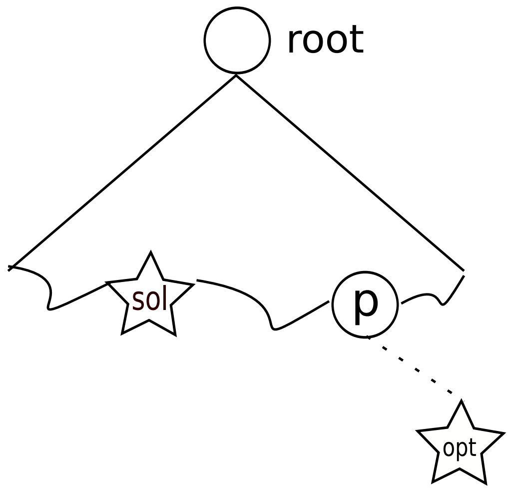
$$
\begin{aligned}
f(p) &\leq f(opt) \\
f(f_{min}) &\leq f(p)
\end{aligned}
$$

f_{min} provides a lower bound on solution cost.

Determine f_{min} by priority queue sorted on f

Optimistic Search: Run a greedy search
Expand f_{min} until $w \cdot f_{min} \geq f(sol)$

Proving w-Admissibility



- p is the deepest node on an optimal path to opt.
- f_{min} is the node with the smallest f value.

$$
\begin{aligned}
f(p) &\leq f(opt) \\
f(f_{min}) &\leq f(p)
\end{aligned}
$$

f_{min} provides a lower bound on solution cost.

Determine f_{min} by priority queue sorted on f

Optimistic Search: Run a greedy search

Expand f_{min} until $w \cdot f_{min} \geq f(sol)$

Empirical Evaluation

Talk Outline

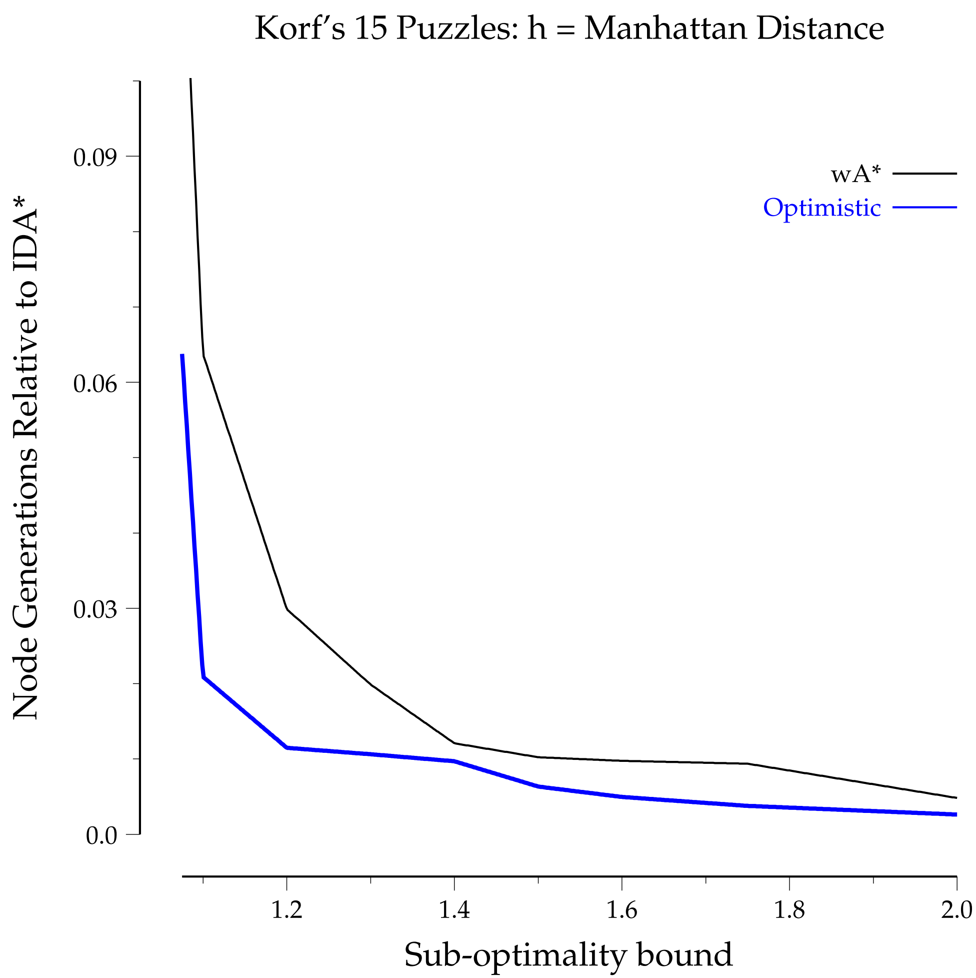
- Algorithm Overview
- The Greedy Search
- Guaranteeing solution quality
- Empirical Evaluation
  - Results in several domains.
- Further Observations

Empirical Evaluation

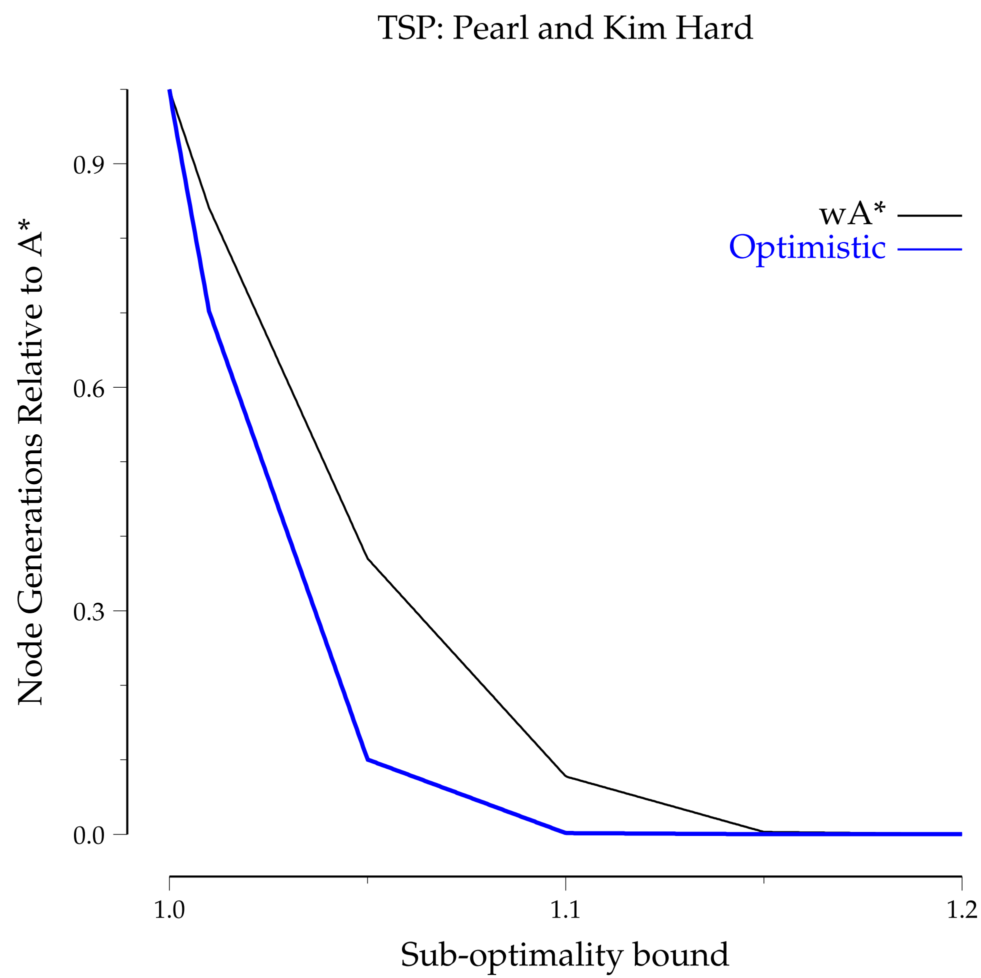
- Sliding Tile Puzzles
  - Korf's 100 15-puzzle instances (add date)
- Traveling Salesman
  - Unit Square
  - Pearl and Kim Hard (add date)
- Grid world path finding
  - Four-way and Eight-way Movement
  - Unit and Life Cost Models
  - 25%, 30%, 35%, 40%, 45% obstacles
- Temporal Planning
  - Blocksworld, Logistics, Rover, Satellite, Zenotravel

See paper for additional plots.

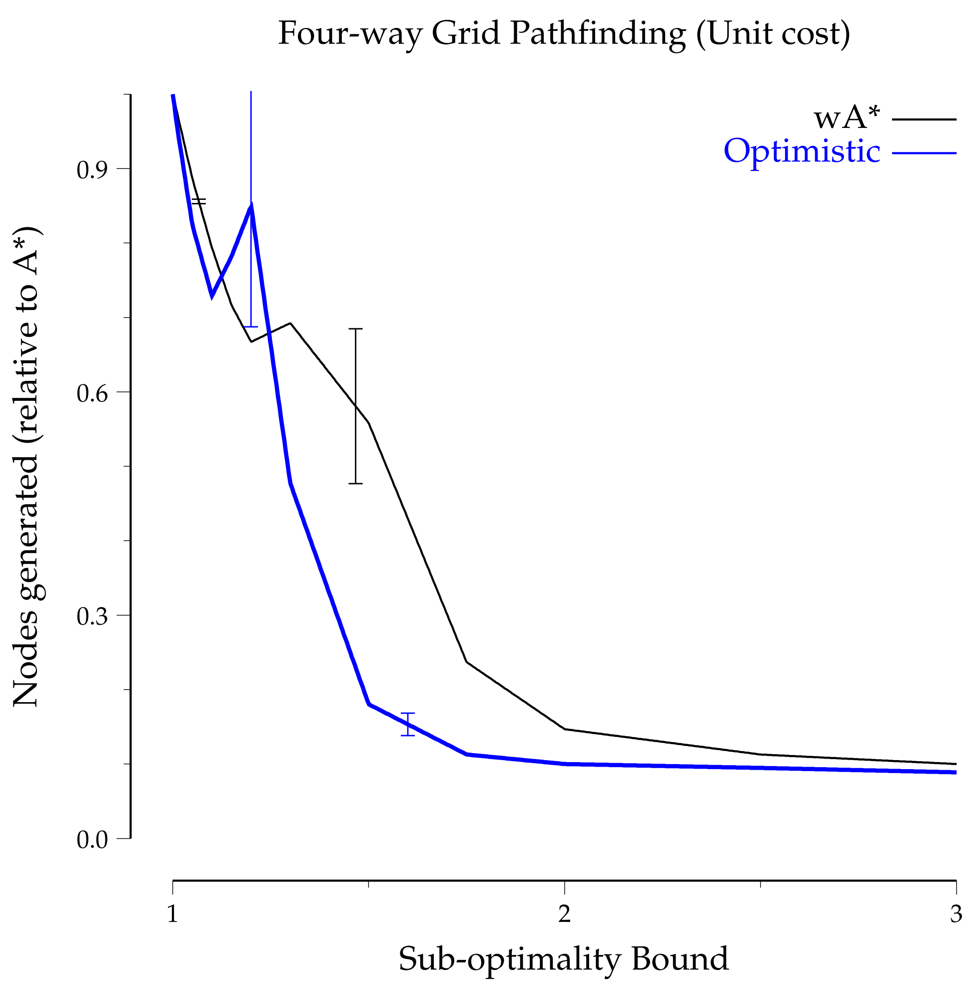
Performance of Optimistic Search



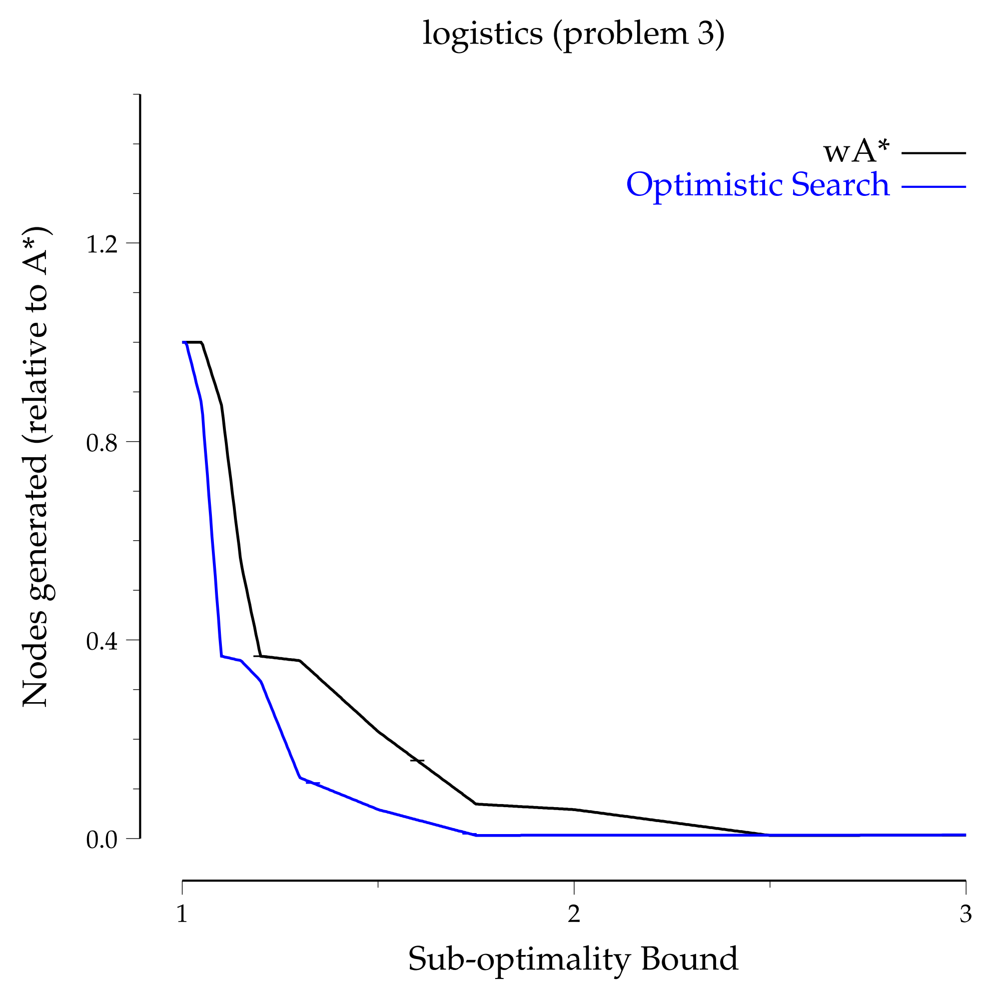
Performance of Optimistic Search



Performance of Optimistic Search



Performance of Optimistic Search



Further Observations

Talk Outline

- Algorithm Overview
- The Greedy Search
- Guaranteeing solution quality
- Empirical Evaluation
- Further Observations
  - Strict vs. Loose Expansion Policy
  - Bounded Anytime Weighted A^*

Strict Expansion Order:

- Algorithms like
  wA^*, A^*_ϵ, Dynamically Weighted A^*
- Any expanded node can be shown to be within the bound at the time of their expansion
- Quality bound comes from this

Loose Expansion Order:

- Algorithms like
  Optimistic Search
- No restriction on the nodes expanded initially.
- Quality bound requires node expansion beyond the initial solution.

Bounded Anytime Weighted A^*

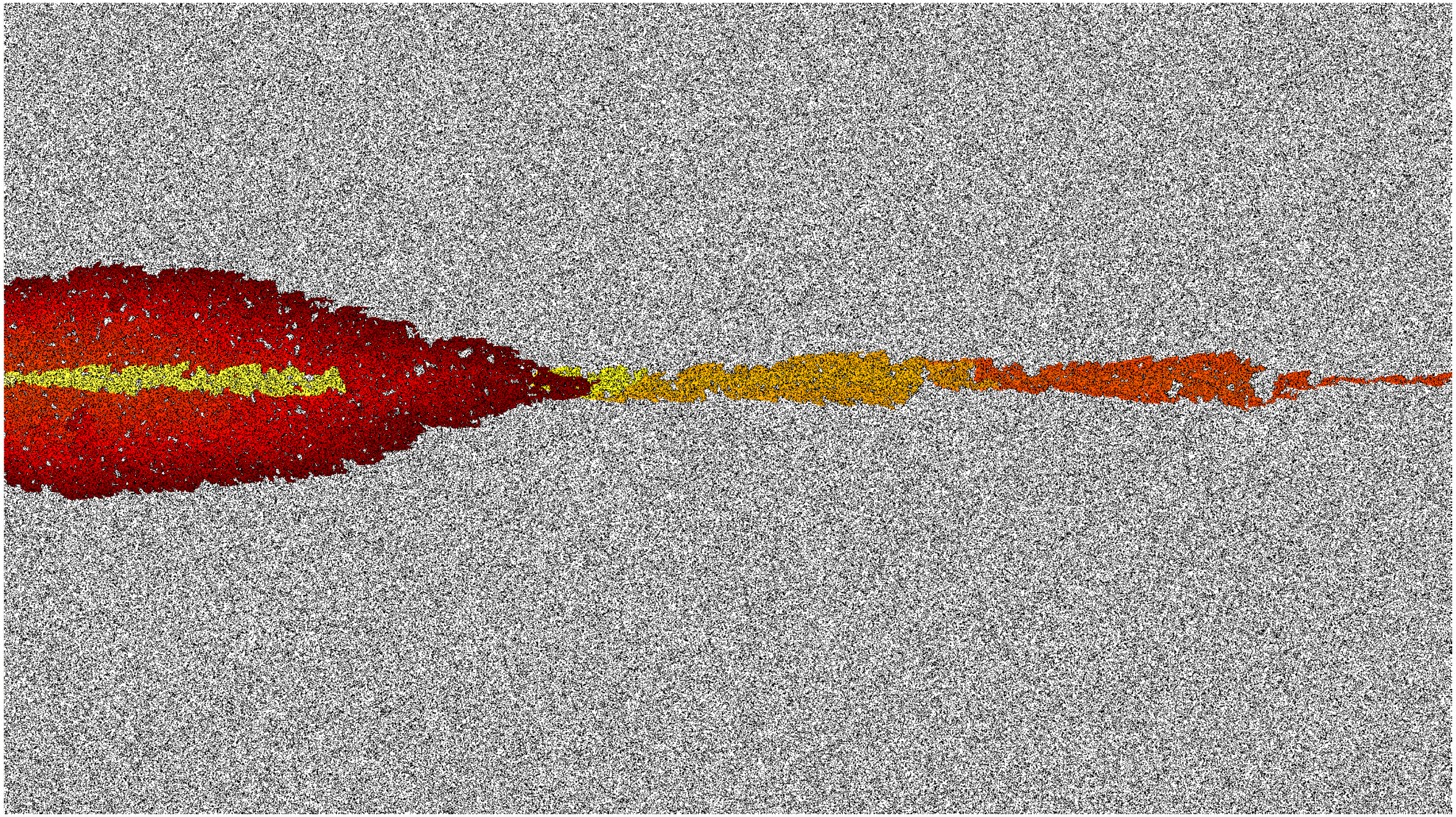
- Anytime Heuristic Search:
  - Running weighted A^* with a high weight
  - Continue node expansions after a solution is found

Bounded Anytime Weighted A^*

- Anytime Heuristic Search:
  - Running weighted A^* with a high weight
  - Continue node expansions after a solution is found
- Bounded Anytime Weighted A^*:
  - Running weighted A^* with a high weight
  - Continue node expansions after a solution is found
  - Add a second priority queue allows us to converge on a bound instead of on optimal.

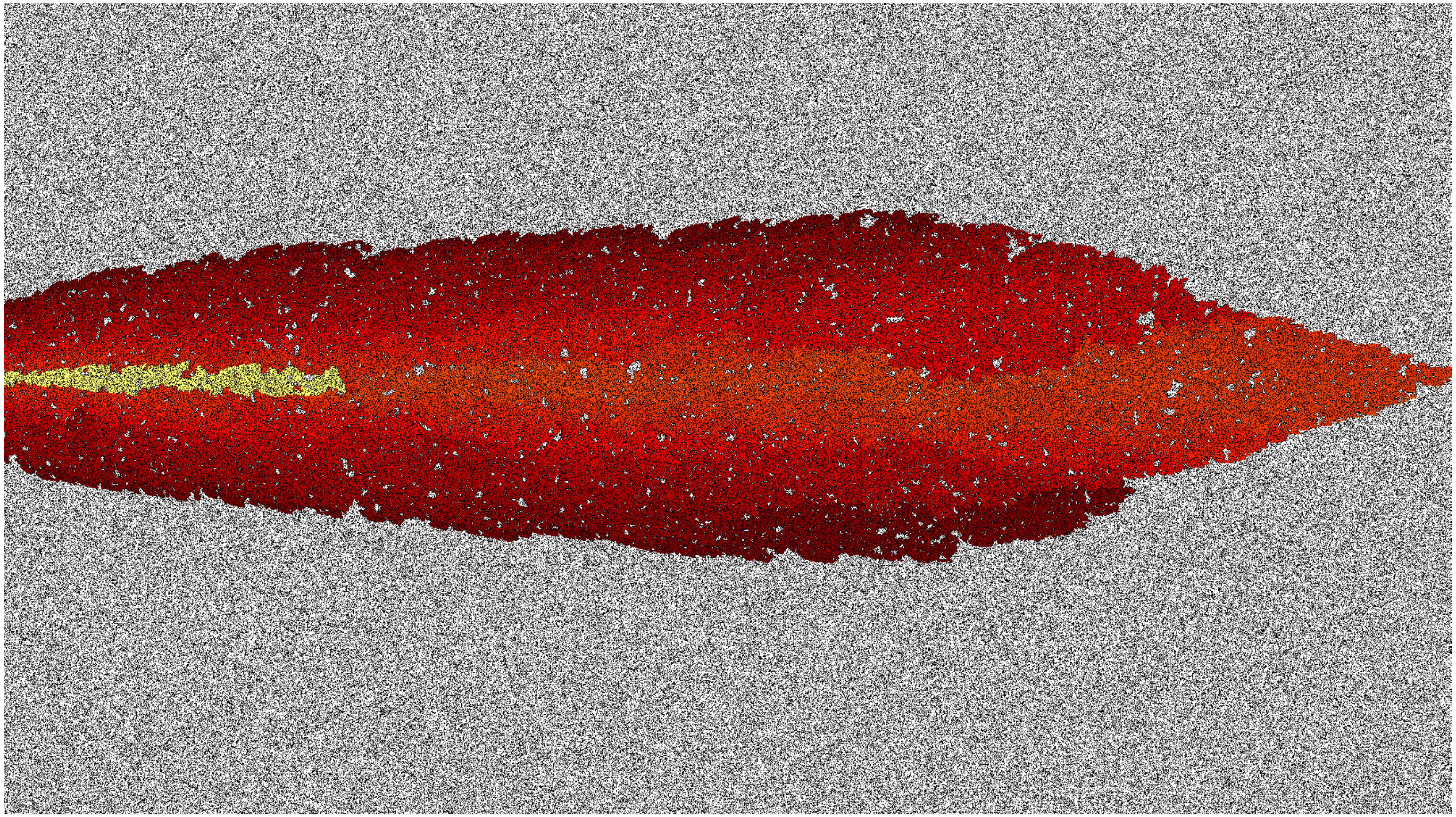
Optimistic Search expansions

1. Run weighted A^* with a high weight.
2. Expand node with lowest f value after a solution is found.
   Continue until $w \cdot f_{min} > f(sol)$
   This 'clean up' guarantees solution quality.

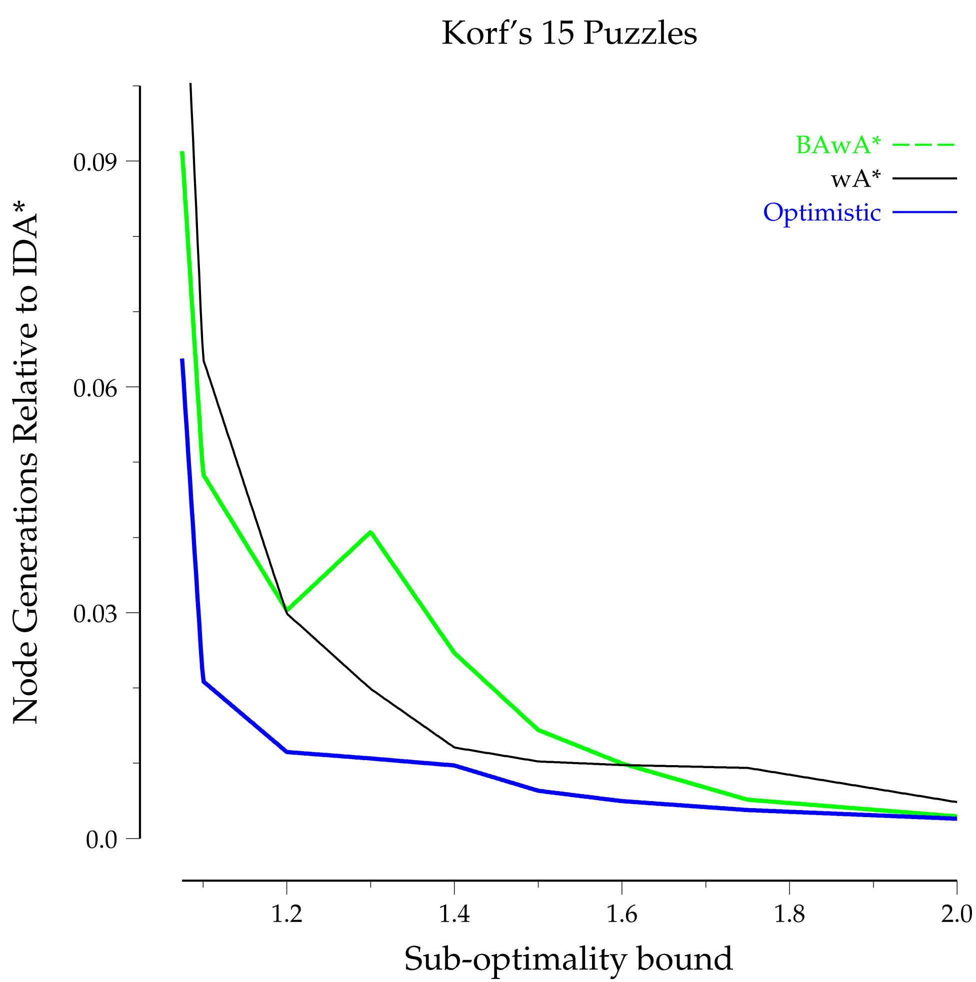


Bounded Anytime Weighted A^* Expansions

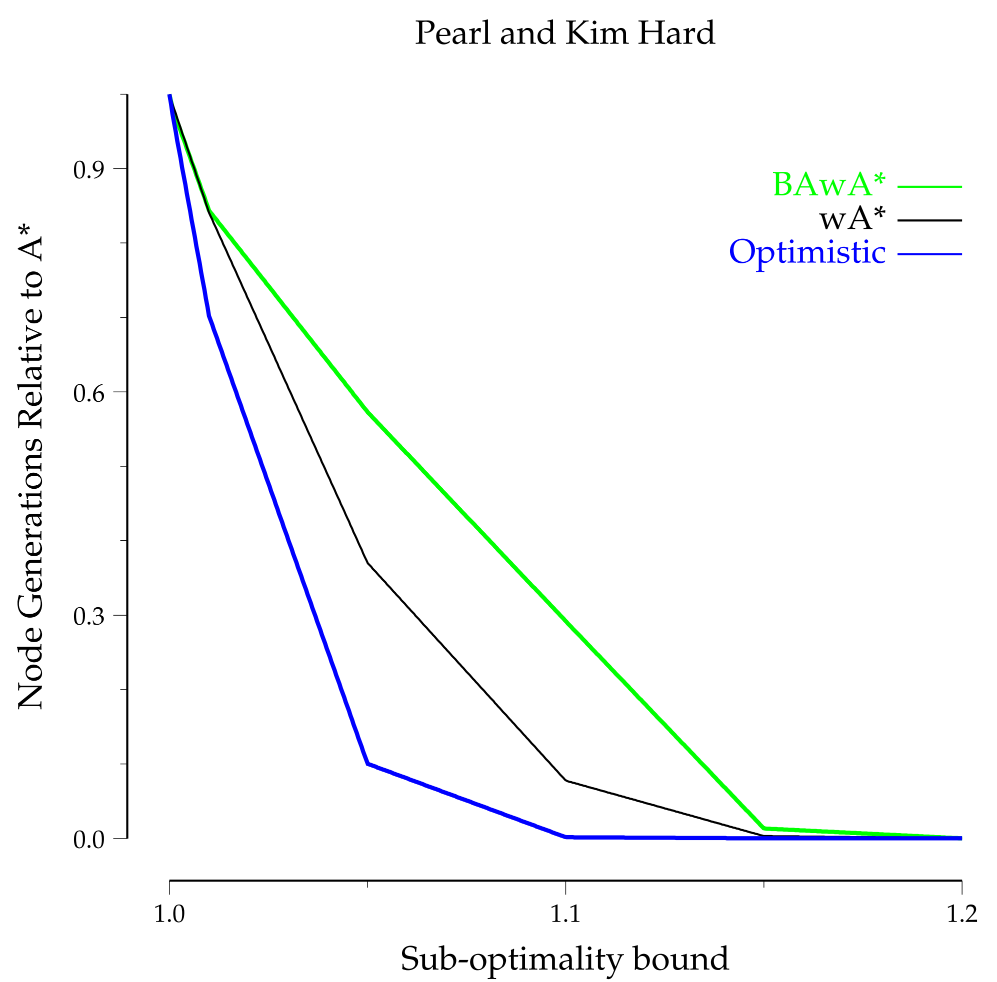
1. Run weighted A^* with a high weight.
2. Expand node with lowest f' value after a solution is found.
   Continue until $w \cdot f_{min} > f(sol)$
   This 'clean up' guarantees solution quality.



Bounded Anytime Weighted A*



Bounded Anytime Weighted A*



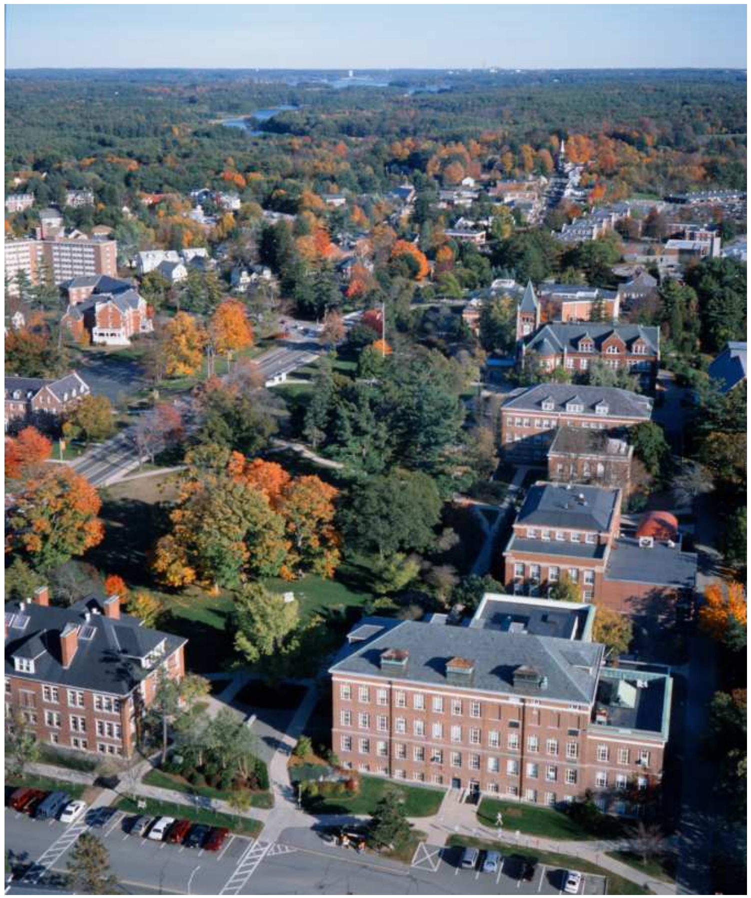
Conclusion

Optimistic Search:

- Simple to implement.
- Performance is predictable.
- Current results are good, tuning could help.
  Optimal greediness is still an open question.
- Consistently better than Weighted A^*
  If you currently use wA^*, you should use Optimistic Search.

The University of New Hampshire

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

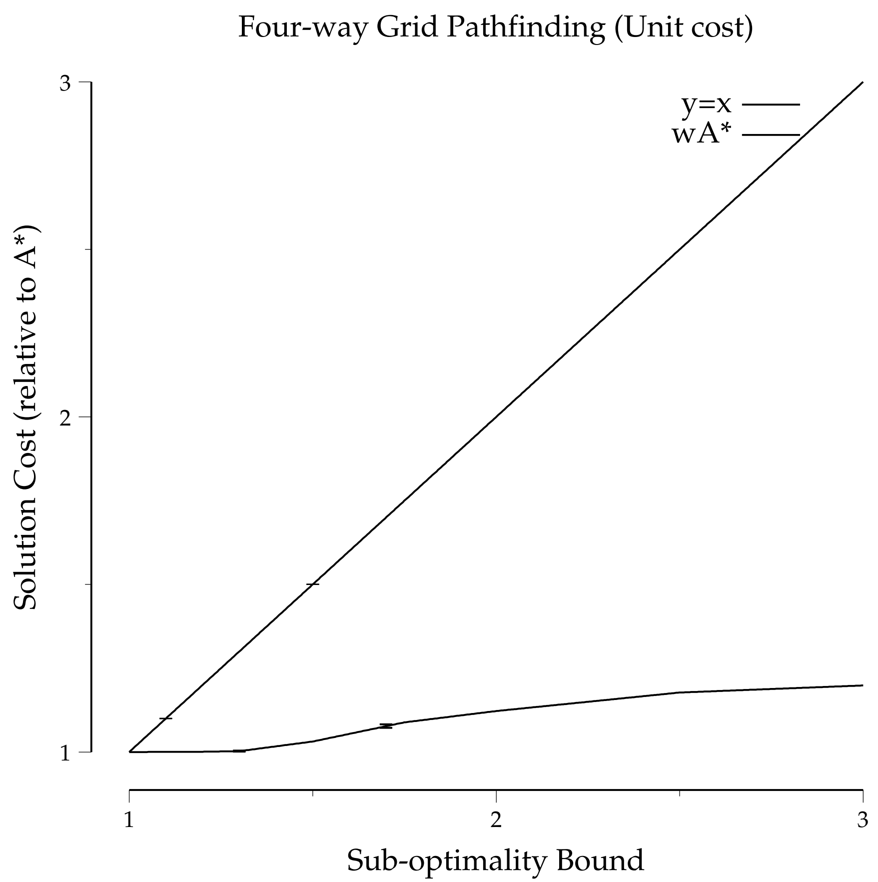


- friendly faculty
- funding
- individual attention
- beautiful campus
- low cost of living
- easy access to Boston, White Mountains
- strong in AI, infoviz, networking, systems, bioinformatics

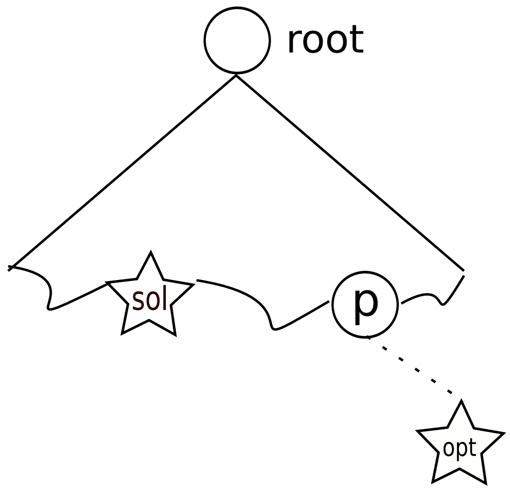
Additional Slides

Weighted A^* is often better than the bound

- wA^* returns solutions better than the bound.



Weighted A^* Respects a Bound

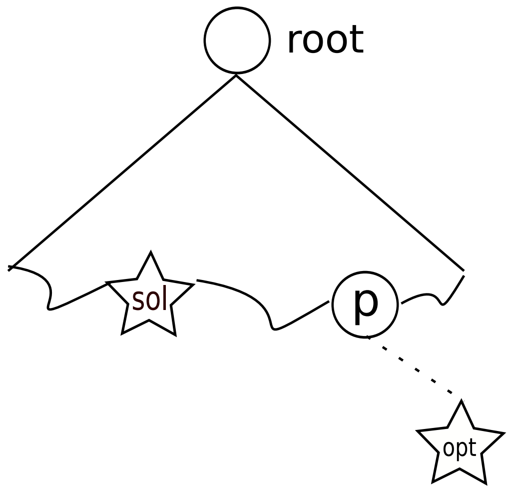


$$f(n) = g(n) + h(n)$$
$$f'(n) = g(n) + w \cdot h(n)$$

$$\begin{array}{lllllll} g(sol) & & & & & & \\ f'(sol) & \leq & f'(p) & & & & \\ & & g(p) + w \cdot h(p) & \leq & w \cdot (g(p) + h(p)) & & \\ & & & & w \cdot f(p) & \leq & w \cdot f(opt) \\ & & & & & & w \cdot g(opt) \end{array}$$

Therefore, $g(sol) \leq w \cdot g(opt)$

Weighted A^* Respects the Bound and Then Some

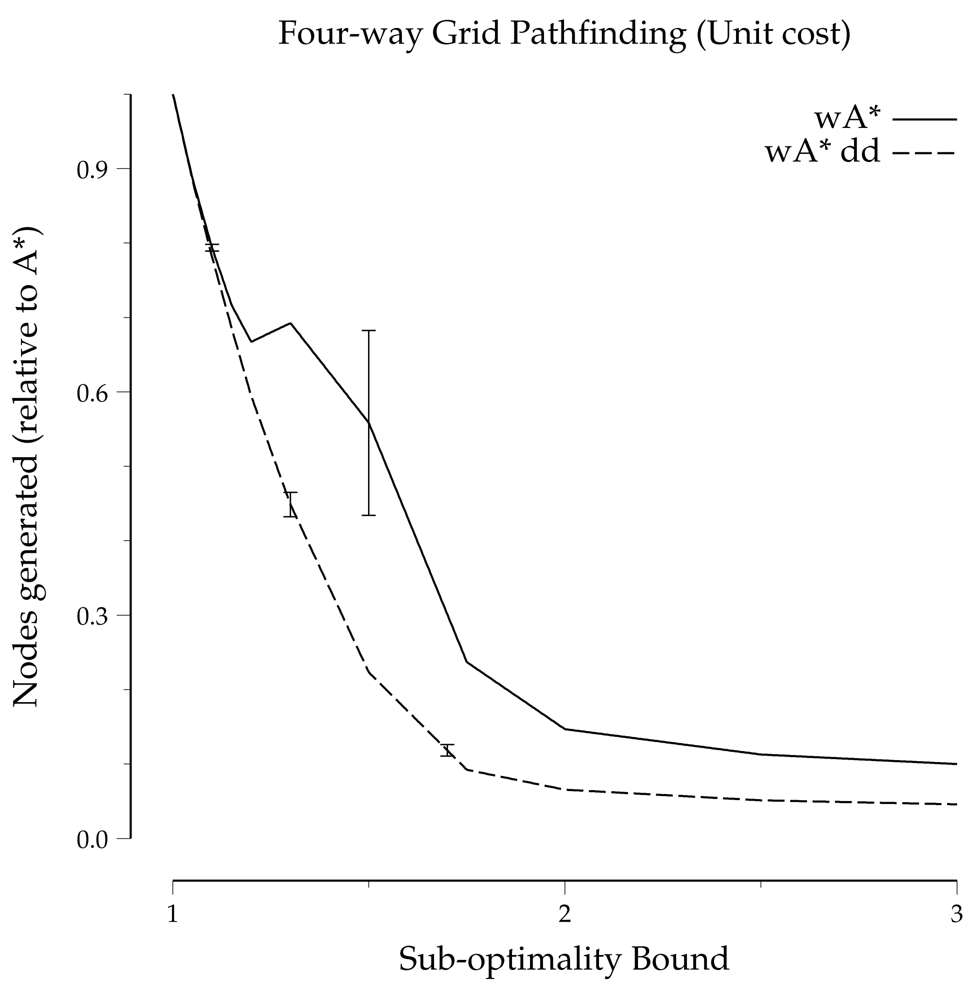


$$f(n) = g(n) + h(n)$$
$$f'(n) = g(n) + w \cdot h(n)$$

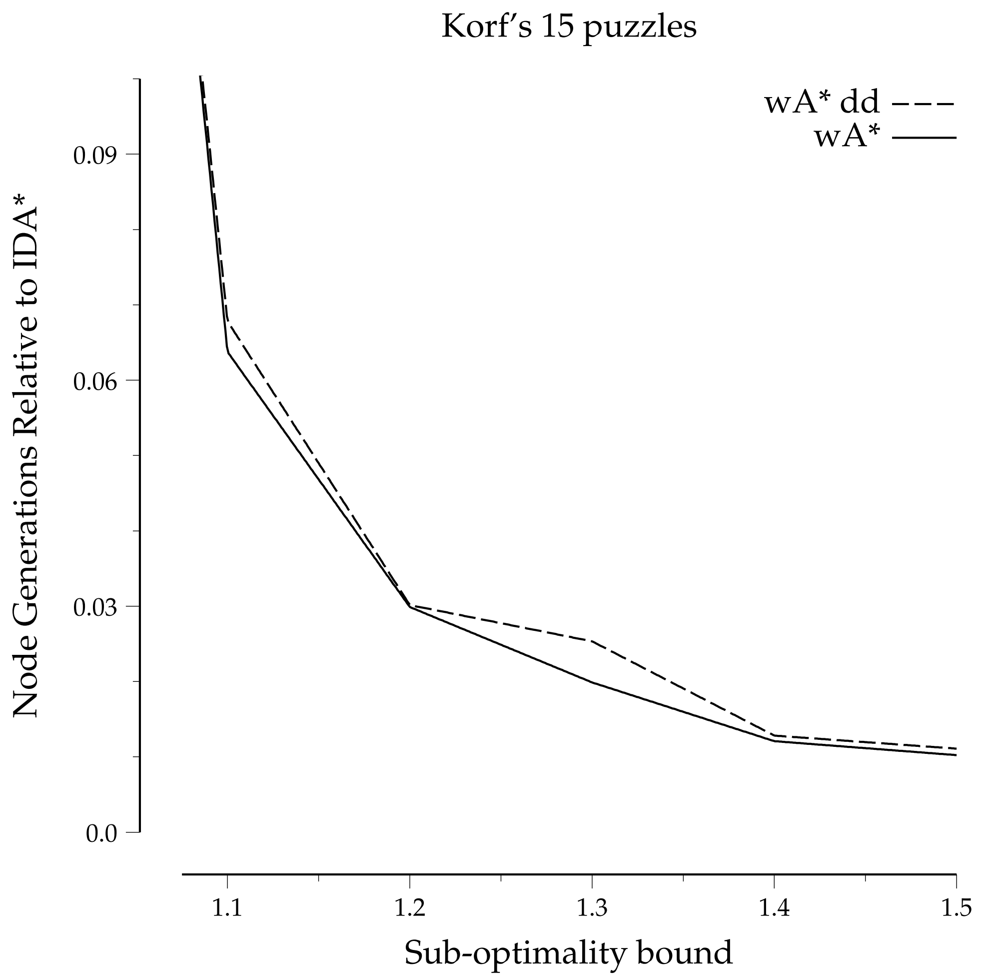
$$g(sol)$$
$$f'(sol) \leq f'(p)$$
$$g(p) + w \cdot h(p) \leq w \cdot (g(p) + h(p))$$
$$w \cdot f(p) \leq w \cdot f(opt)$$
$$w \cdot g(opt)$$

$$g(p) + w \cdot h(p) \leq w \cdot g(p) + w \cdot h(p)$$

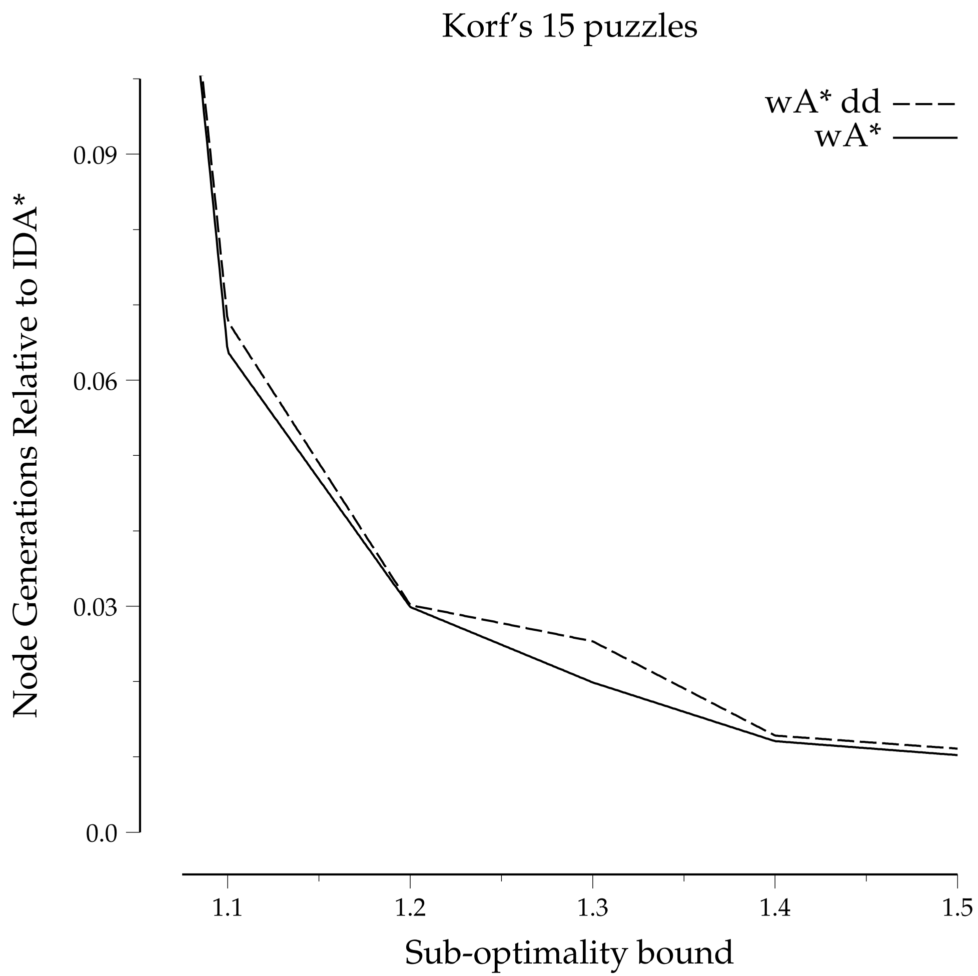
Duplicate Dropping can be Important



Sometimes it isn't



Duplicates can be delayed during the greedy search phase.



Pseudo Code

Optimistic Search(*initial*, *bound*)

$open_f \leftarrow \{initial\}$
$open_{\widehat{f}} \leftarrow \{initial\}$
incumbent $\leftarrow \infty$
repeat until *bound* $\cdot f$(first on $open_f$) $\geq f(incumbent)$:
    if $\widehat{f}$(first on $open_{\widehat{f}}$) $< \widehat{f}(incumbent)$ then
        $n \leftarrow$ remove first on $open_{\widehat{f}}$
        remove n from $open_f$
    else $n \leftarrow$ remove first on $open_f$
        remove n from $open_{\widehat{f}}$
    add n to *closed*
    if n is a goal then
        incumbent $\leftarrow n$
    else for each child c of n
        if c is duplicated in $open_f$ then
            if c is better than the duplicate then
                replace copies in $open_f$ and $open_{\widehat{f}}$
        else if c is duplicated in *closed* then
            if c is better than the duplicate then
                add c to $open_f$ and $open_{\widehat{f}}$
        else add c to $open_f$ and $open_{\widehat{f}}$