Master's Thesis:
Heuristic Search Under a Deadline

Austin Dionne

UNIVERSITY of NEW HAMPSHIRE

Department of Computer Science

`austin.dionne` at `gmail.com`

Acknowledgements

Thanks to:

- Wheeler Ruml (Advisor)
- Jordan T. Thayer (Collaborator)
- NSF (grant IIS-0812141)
- DARPA CSSG program (grant N10AP20029)

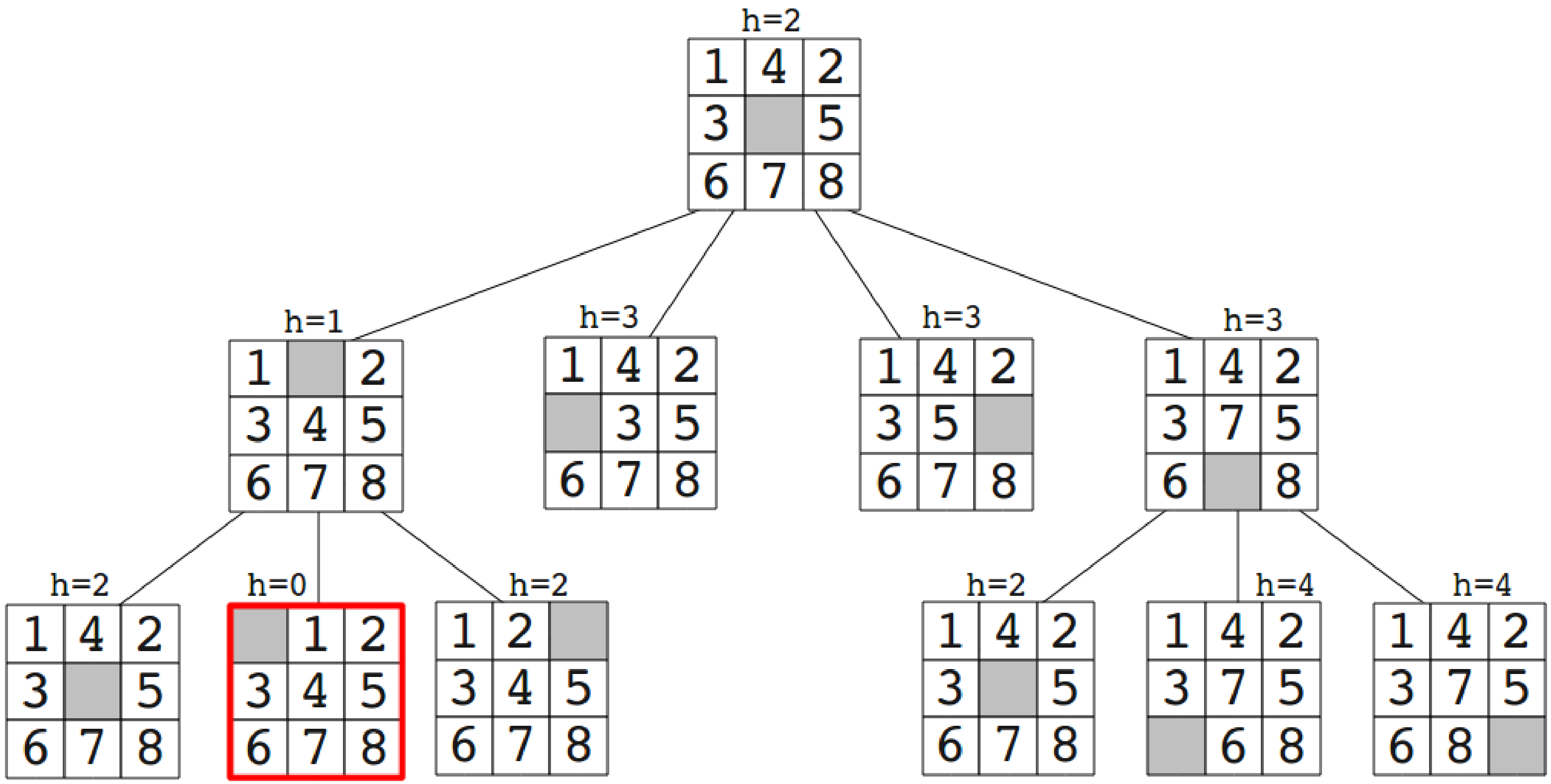
Introduction

Search Is Awesome!



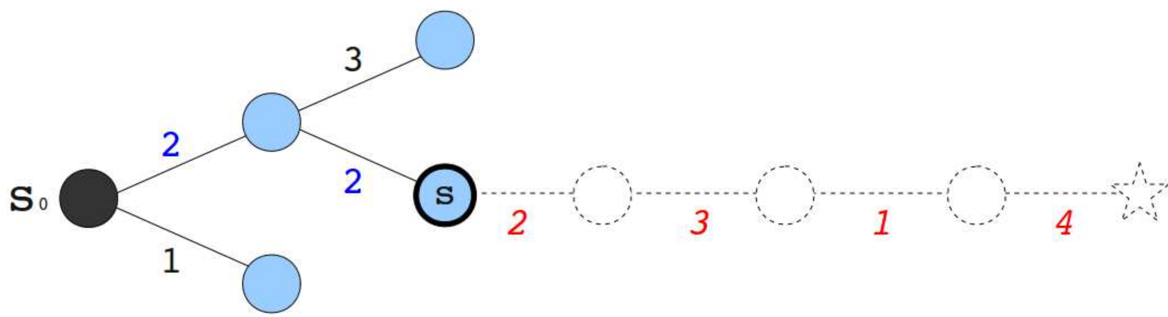


Heuristic Search



Heuristic Search (Continued)

s_0 : starting state
$expand(s)$: returns list of child states (s_c, c)
$goal(s)$: returns true if s is a goal state, false otherwise
$g(s)$: cost accumulated so far on path from s_0 to s
$h^*(s)$: cost of cheapest solution under s
$f^*(s) = g(s) + h^*(s)$: estimated cost of best solution under s
$d^*(s)$: number of steps to cheapest solution under s
$h(s)$, $f(s)$, $d(s)$: heuristic estimators of true values
$\widehat{d}(s)$: unbiased estimator of d^*



$g(s) = 2 + 2 = 4$
$h^*(s) = 2+3+1+4 = 10$
$f^*(s) = g(s) + h^*(s) = 14$

$d^*(s) = 4$

$h(s) = 5$
$f(s) = 4 + 5 = 9$

$d(s) = 2$
$\widehat{d}(s) = 3$

Problem Definition

Given a problem and a **limited amount of computation time**, find the **best solution possible** before the deadline.

- Problem which often occurs in practice
- The current "best" methods do not directly consider the presence of a deadline and waste effort.
- The current "best" methods require off-line tuning for optimal performance.

Thesis Statement

My thesis is that a deadline-cognizant approach which attempts to expend all available search effort towards a single final solution has the potential for outperforming these methods without off-line optimization.

Contributions

In this thesis we have proposed:

- Corrected single-step error model for $\widehat{d}(s)$ and $\widehat{h}(s)$
- Deadline Aware Search (DAS) which can outperform current approaches
- Extended single-step error model for calculating d^* and h^* distributions on-line
- Deadline Decision Theoretic Search (DDT) which is a more flexible and theoretically based algorithm that holds some promise

Related Work

We are not the first to attempt to solve this problem...

- Time Constrained Search *(Hiraishi, Ohwada, and Mizoguchi 1998)*
- Contract Search *(Aine, Chakrabarti, and Kumar 2010)*

Neither of these methods work well in practice!

Related Work (Continued)

Problem with Time Constrained Search:

- Parameters abound! (ϵ_{upper}, ϵ_{lower}, Δw)
- Important questions without answers:
  - When (if ever) should we resort open list?
  - Is a hysteresis necessary for changes in w?

I could not implement a version of this algorithm that worked well!

Related Work (Continued)

Problem with Contract Search:

- Not really applicable to domains with goals at a wide range of depths (tiles/gridworld/robots)
- Takes **substantial** off-line effort to prepare the algorithm for a particular domain and deadline

Jordan Thayer implemented this algorithm and it does not work well!

Currently Accepted Approach

Anytime Search

- Search for a suboptimal initial solution relatively quickly
- Continue searching, finding sequence of improved solutions over time
- Eventually converge to optimal

Problems:

1. Wasted effort in finding sequence of mostly unused solutions
2. Based on bounded suboptimal search, which requires parameter settings
   - May not have time for off-line tuning
   - For some domains different deadlines require different settings

Our Motivation

Our desired deadline-aware approach should:

- Consider the time remaining in ordering state expansion
- Perform consistently well across a full range deadlines (fractions of a second to minutes)
- Be parameterless and general
- Not require significant off-line computation

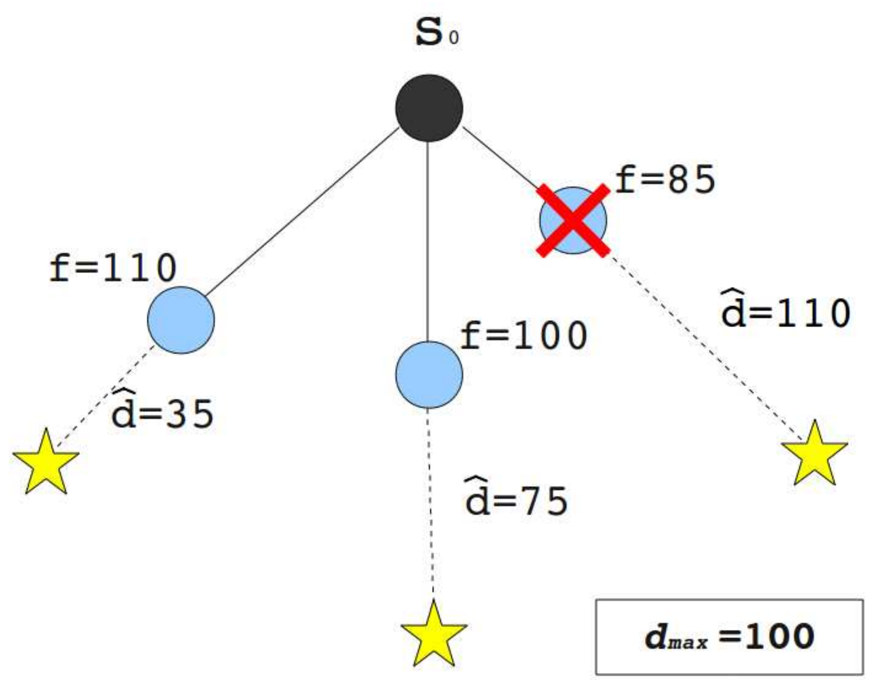
Recap

- Search under deadlines is a difficult and important problem
- Previously proposed approaches don't work
- Currently used approaches are unsatisfying
- We propose an algorithm (DAS) which can outperform these methods **without the use of off-line tuning**

Deadline Aware Search (DAS)

DAS pursues the **best** solution path which is **reachable** within the time remaining in the search.

- **Best** is defined as minimal $f(s)$
- **Reachability** is a function of an estimate distance to a solution $\widehat{d}(s)$ and the current behavior of the search

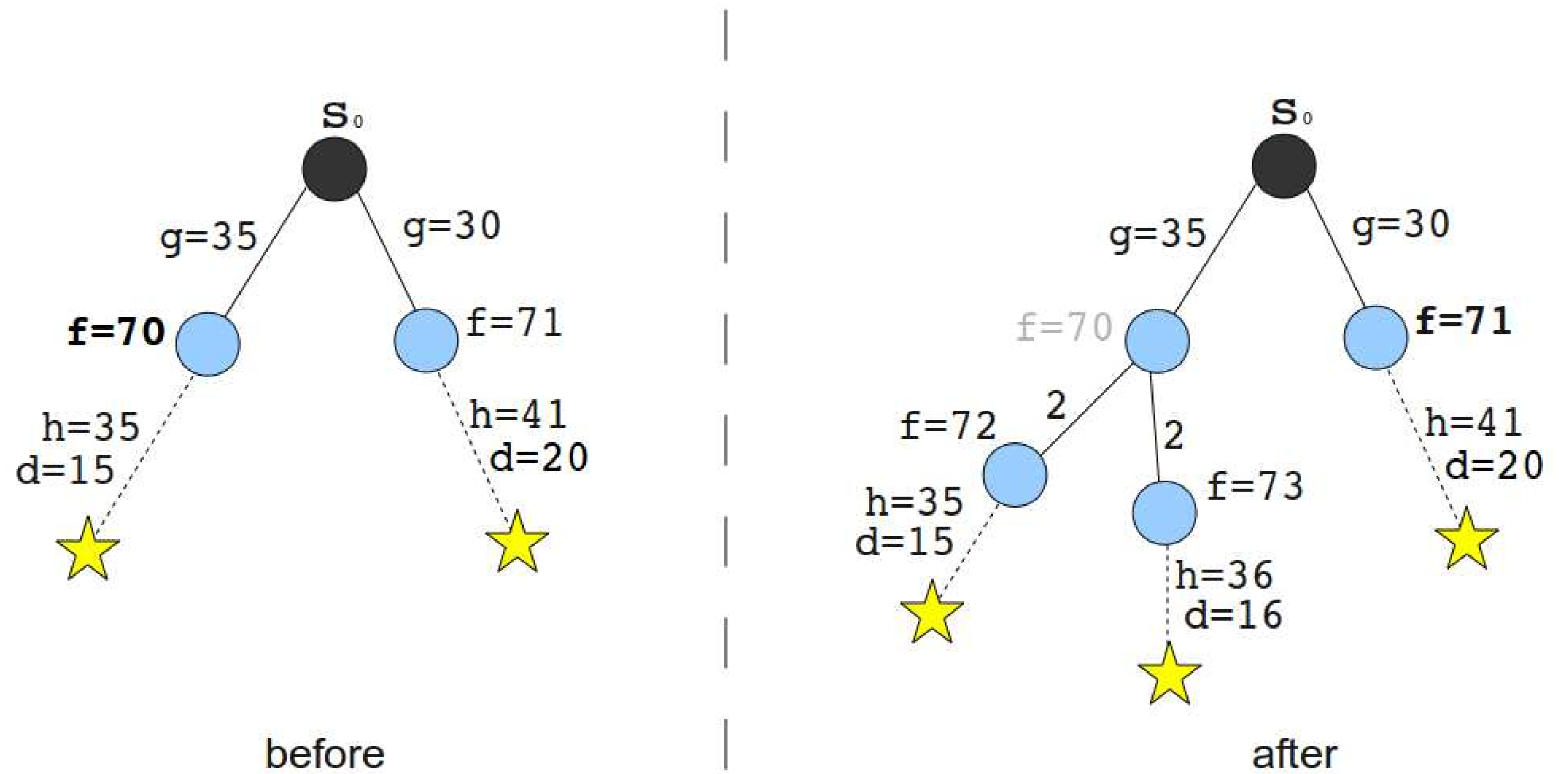


DAS: High-Level Algorithm

While there is time remaining before the deadline:

- Calculate maximum allowable distance d_{max}
- Select node n from open list with minimal $f(n)$
- If $\widehat{d}(n) \leq d_{max}$ (solution is reachable)
  - Expand n, add children to open list
- Otherwise (solution is unreachable)
  - Add n to pruned list

Error in $h(s)$ produces **Search Vacillation**.

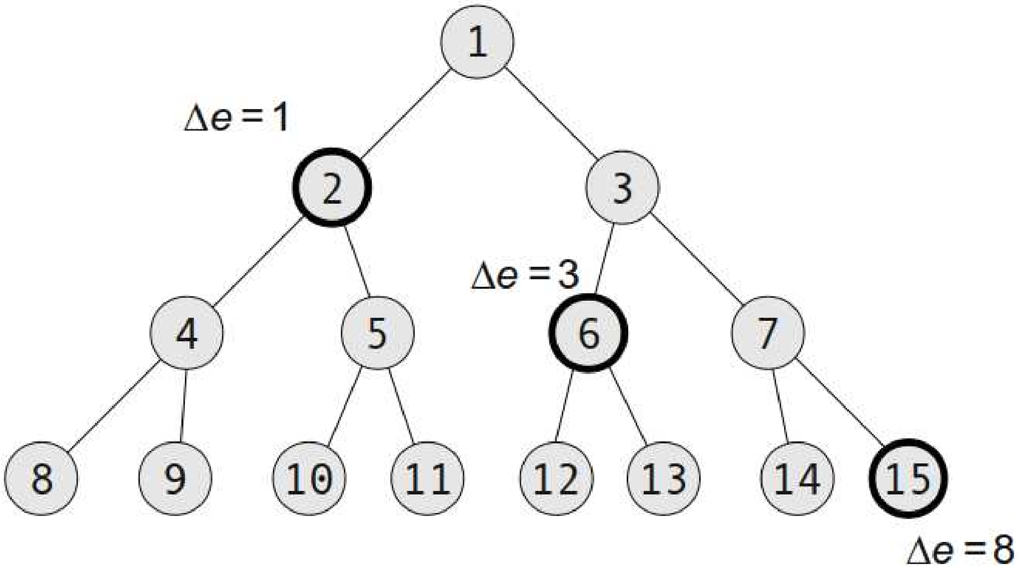


Expansion Delay

Maintain a running expansion counter during search.

At state expansion, define expansion delay as:

$$\Delta e = (\text{current exp counter}) - (\text{exp counter at generation})$$



Expansion Delay

Use mean expansion delay $\overline{\Delta e}$ to calculate d_{max}:

$$d_{max} = \frac{\text{(expansions remaining)}}{\overline{\Delta e}} \tag{1}$$

d_{max} estimates the expected number of steps that will be explored down any particular path in the search space.

DAS: High-Level Algorithm

While there is time remaining before the deadline:

- Calculate maximum allowable distance d_{max}
- Select node n from open list with minimal $f(n)$
- If $\widehat{d}(n) \leq d_{max}$ (solution is reachable)
  - Expand n, add children to open list
- Otherwise (solution is unreachable)
  - Add n to pruned list
- **If open list is empty**
  - **Recover a set of nodes from pruned list with "reachable" solutions**
  - **Reset estimate of d_{max}**

DAS: High-Level Algorithm: Search Recovery

Start again with a set of nodes with “reachable” solutions:

Estimated expansions remaining: **150**

Pruned List:

	f(n)	$\hat{d}$(n)
1.	14	14
2.	24	20
3.	25	22
4.	25	30
5.	40	40
6.	41	34
7.	48	42
8.	55	50
9.	66	56
10.	70	67
	⋮	⋮

Sum of $\hat{d}$(n) ≤ exp remaining

14+20+22+30+40 = 126 ≤ **150**

Recap

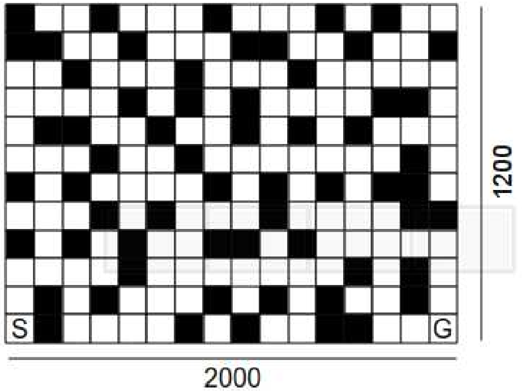
- Search under deadlines is a difficult and important problem
- Previously proposed approaches don't work
- Currently used approaches are unsatisfying
- We propose an algorithm (DAS) which can outperform these methods **without the use of off-line tuning**
    - Uses expansion delay to measure search vacillation
    - Estimates a “reachable” solution distance and prunes nodes

Empirical Evaluation: Domains

15-Puzzle

2 Models:
- Unit-Cost
- Inverse Weighted

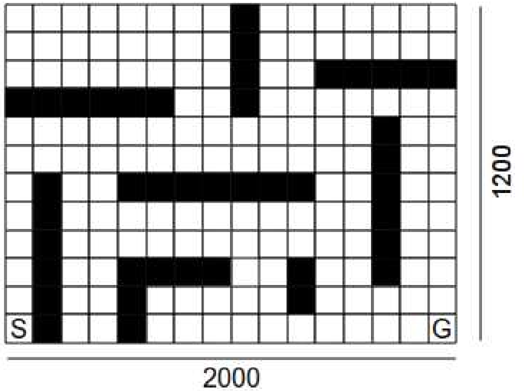
Gridworld



Uniformly Distributed Random Obstacles (p=0.35)

2 Models:
- Unit-Cost
- Life-Cost

Dynamic Robot



75 Randomly Placed Lines

Circular Robot

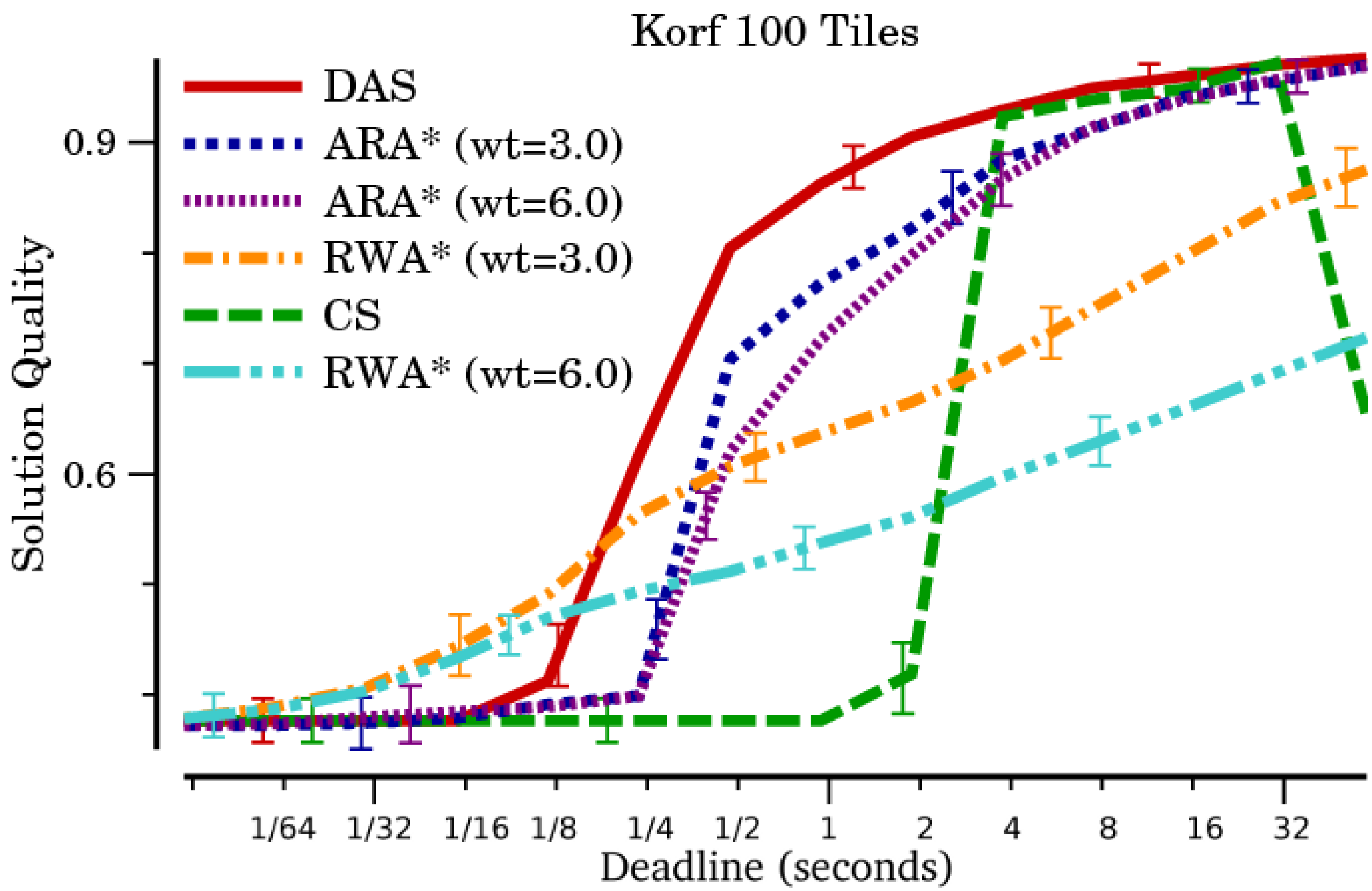
Heading & Velocity

Empirical Evaluation: Methodology

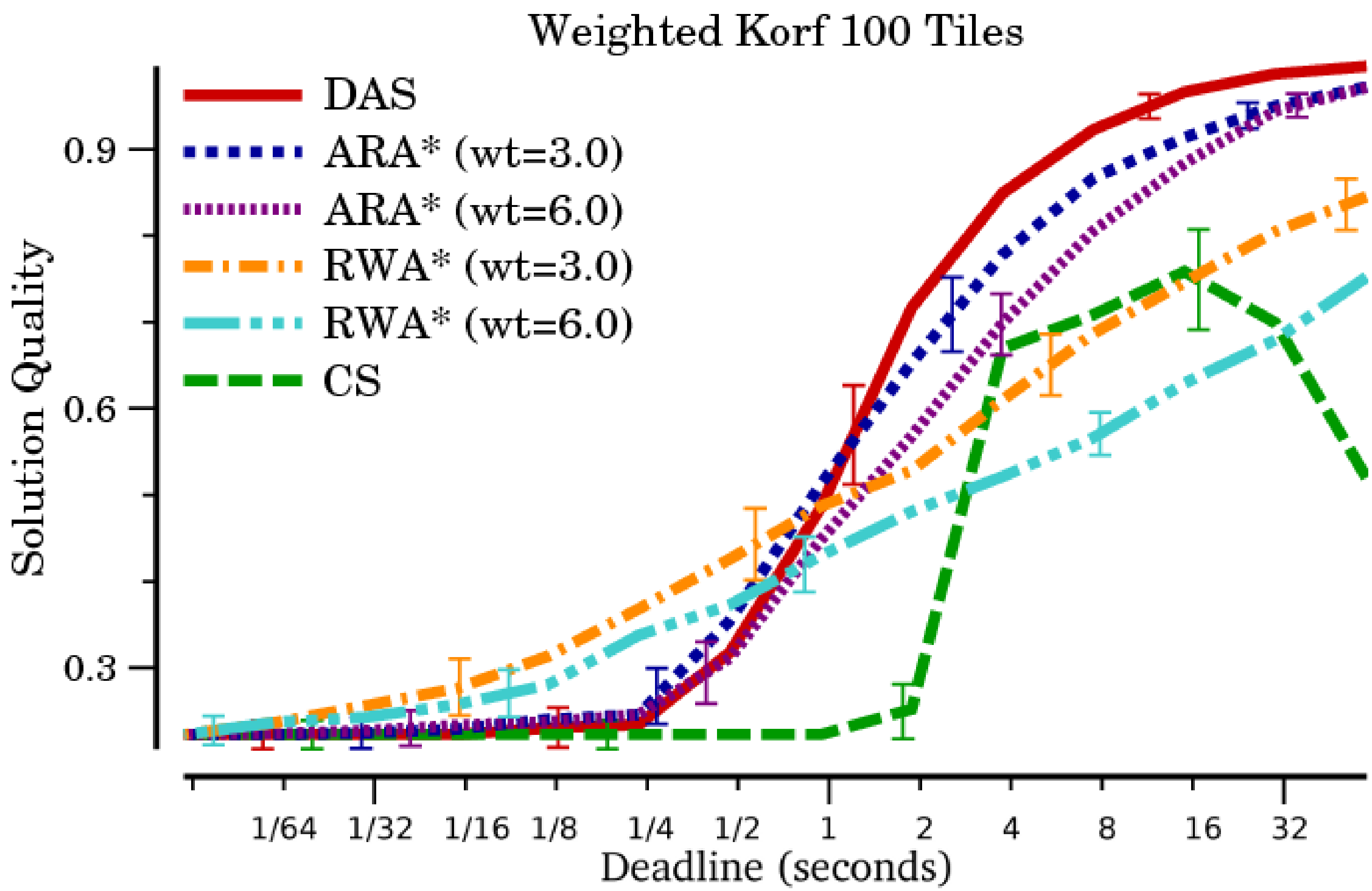
- All algorithms run "Speedier" first to obtain incumbent solution
- Anytime algorithms tested with variety of settings: 1.2, 1.5, 3.0, 6.0, 10.0 (top two performing are displayed)
- Show results for: ARA*, RWA*, CS, DAS
- Deadlines are on a log scale (fractions of second up to minutes)
- Algorithms compared by **solution quality**

solution quality = (best solution cost) / (achieved cost)

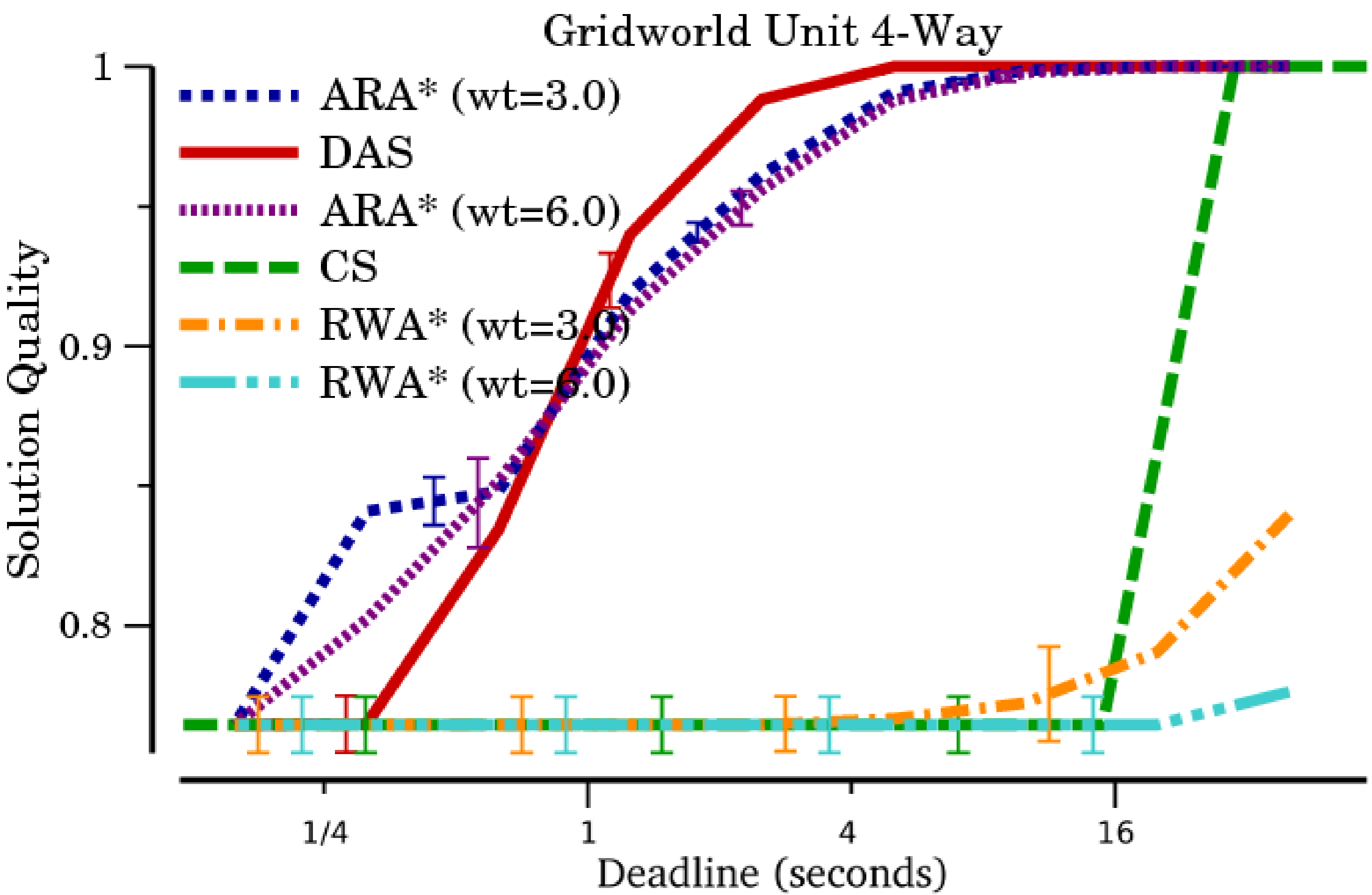
Results: 15-Puzzle



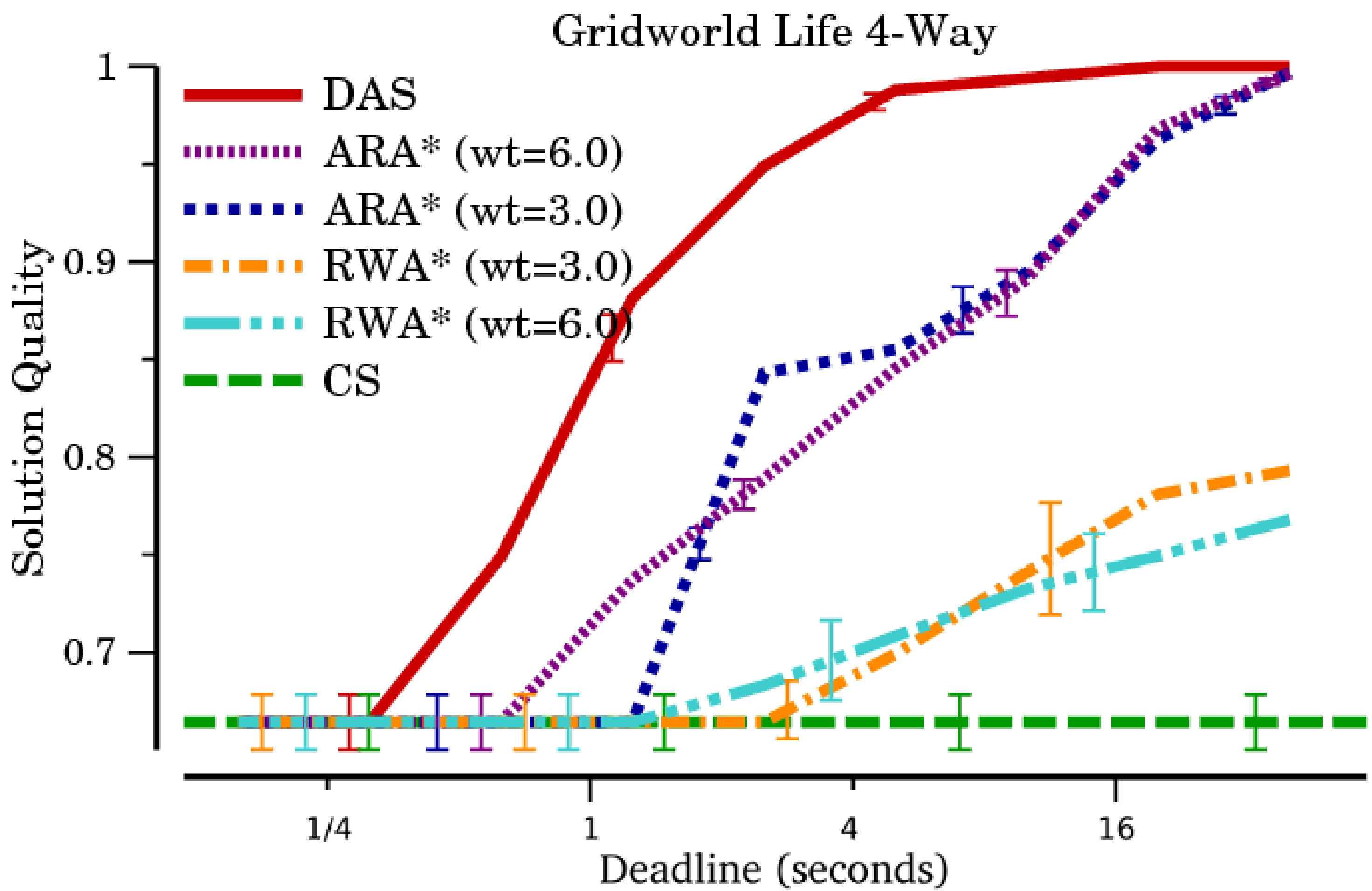
Results: Weighted 15-Puzzle



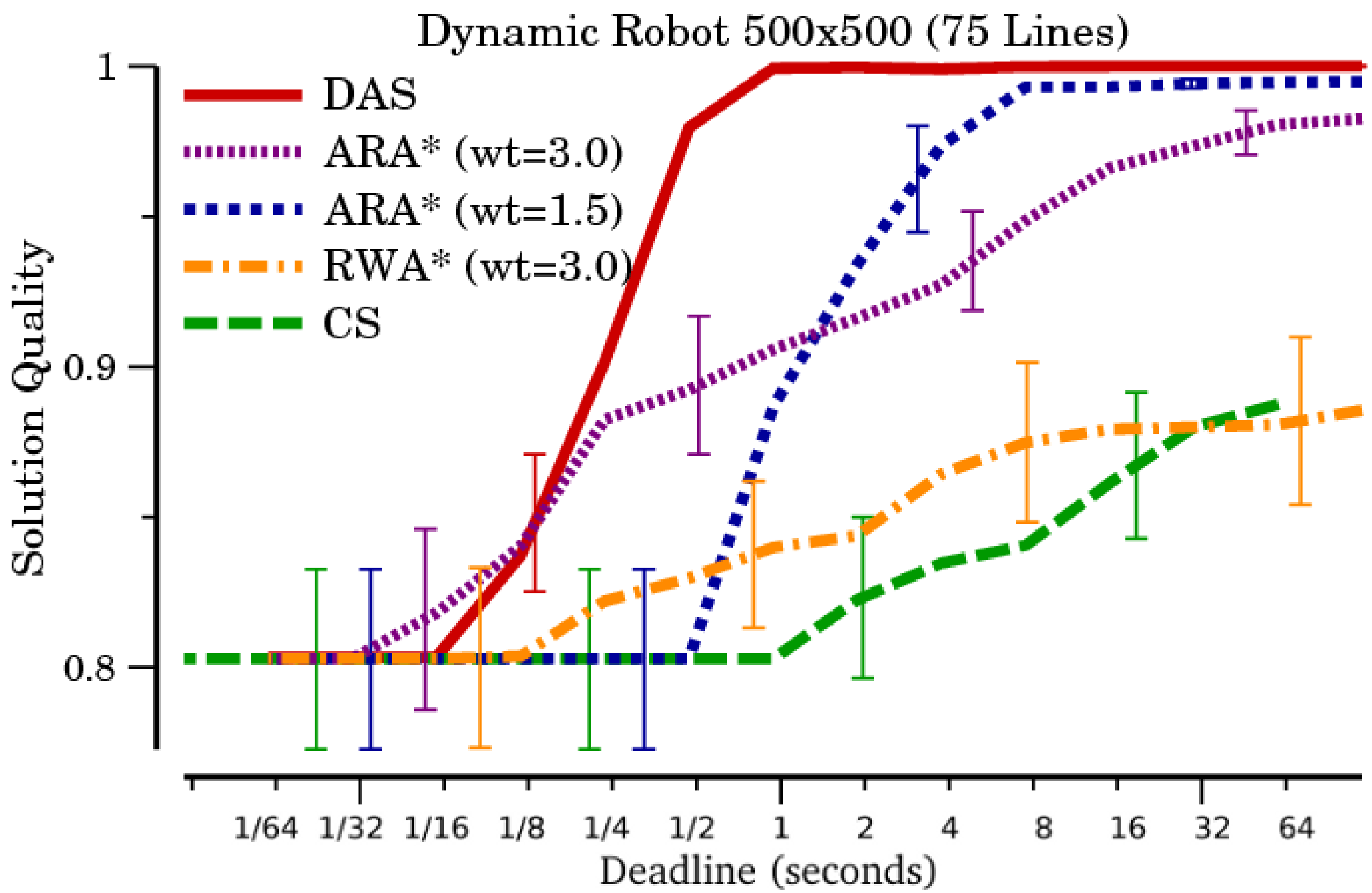
Results: 4-Way 2000x1200 Unit-Cost Gridworld (p=0.35)



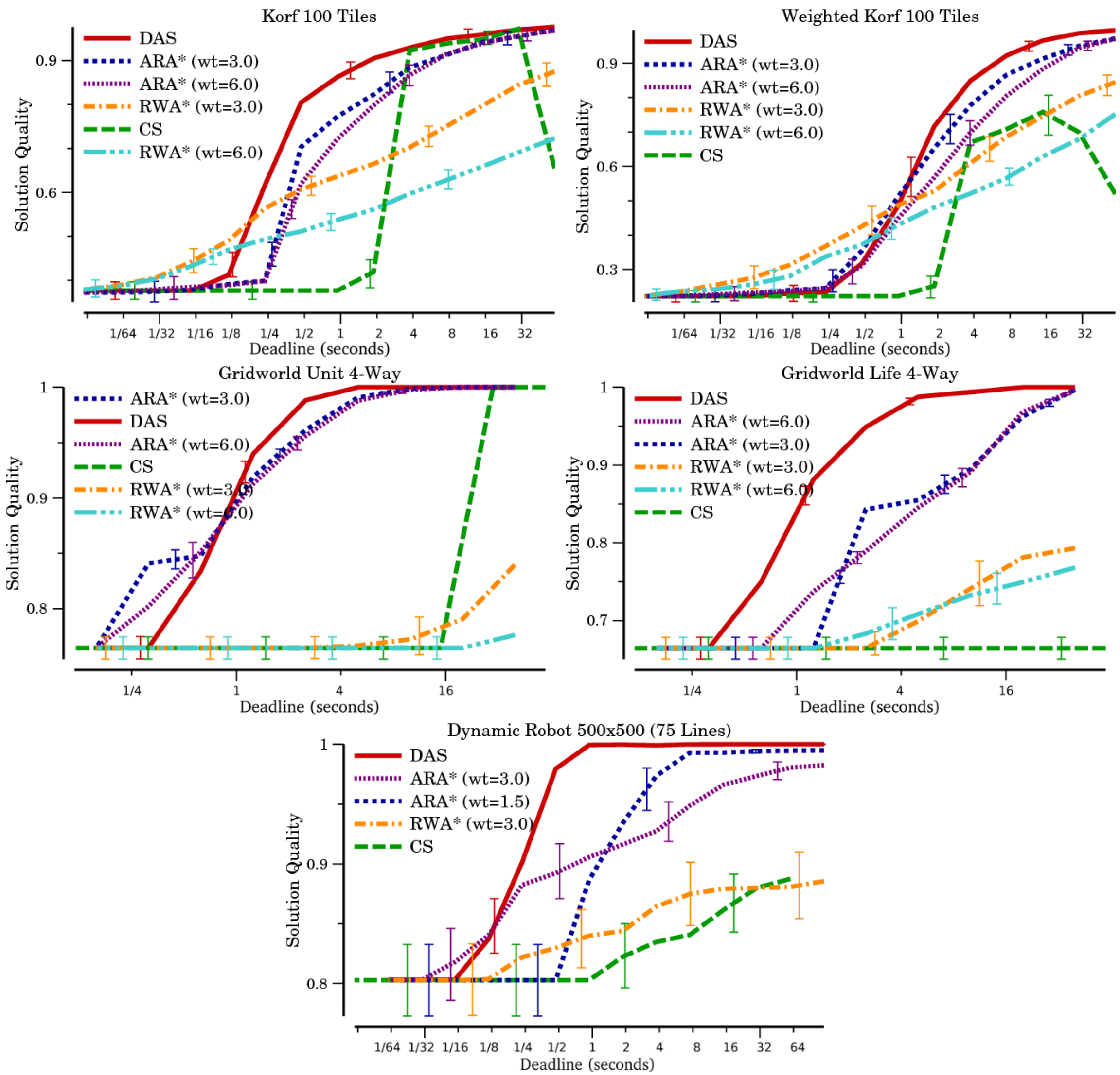
Results: 4-Way 2000x1200 Life-Cost Gridworld (p=0.35)



Results: Dynamic Robot Navigation



Results: Overall



DAS Conclusion

- Parameterless
- Returns optimal solutions for sufficiently large deadlines
- Competitive with or outperforms ARA* for variety of domains

DAS illustrates that an improved deadline-aware approach can be constructed!

Conclusion

Thesis Recap

- Search under deadlines is a difficult and important problem
- Previously proposed approaches don't work
- Currently used approaches are unsatisfying

My thesis is that a deadline-cognizant approach which attempts to expend all available search effort towards a single final solution has the potential for outperforming these methods without off-line optimization.

Contributions

In this thesis we have proposed:

- Corrected single-step error model for $\widehat{d}(s)$ and $\widehat{h}(s)$
- Deadline Aware Search (DAS) which can outperform current approaches
- Extended single-step error model for calculating d^* and h^* distributions on-line
- Deadline Decision Theoretic Search (DDT) which is a more flexible and theoretically based algorithm that holds some promise

DAS illustrates that improvement is possible!

Back-up Slides

DAS Pseudo-Code

Deadline Aware Search(*starting state, deadline*)

```
open ← {starting state}
pruned ← {}
incumbent ← NULL
while (time) < (deadline) and open is non-empty
    d_max ← calculate_d_max()
    s ← remove state from open with minimal f(s)
    if s is a goal and is better than incumbent
        incumbent ← s
    else if d̂(s) < d_max
        for each child s' of state s
            add s' to open
    else
        add s to pruned
    if open is empty
        recover_pruned_states(open, pruned)
return incumbent
```

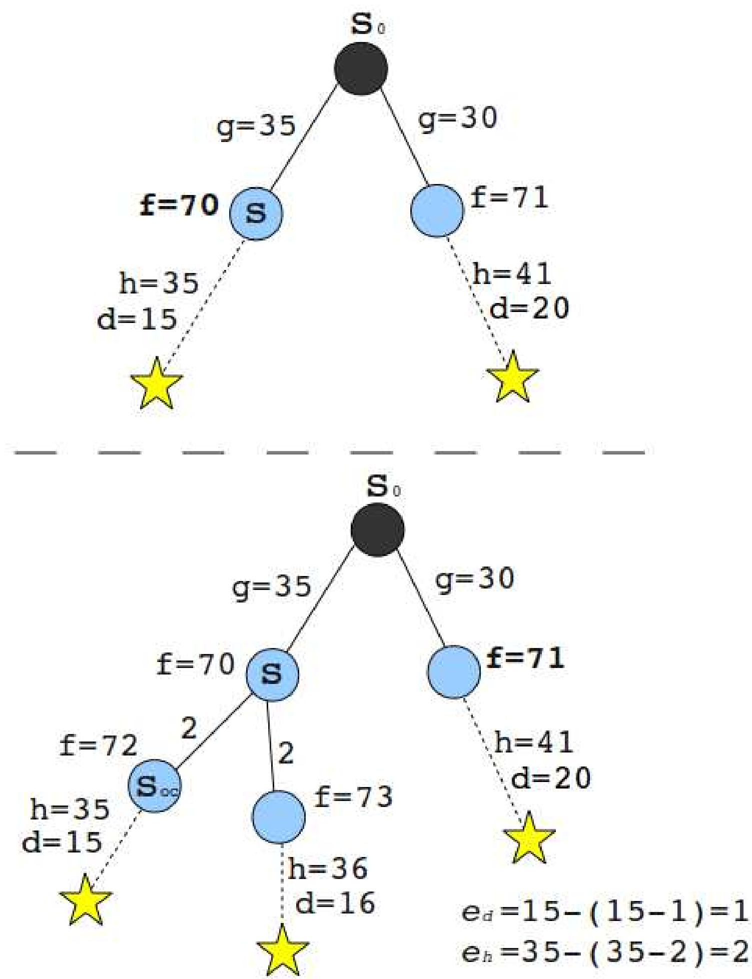
Recover Pruned States(*open*, *pruned*)

18. *exp* $\leftarrow$ *estimated expansions remaining*
19. while *exp* > 0 and *pruned* is non-empty loop
20. $s \leftarrow$ remove state from *pruned* with minimal $f(s)$
21. add s to *open*
23. *exp* $=$ *exp* $-\widehat{d}(s)$

Intention is to replace only a "reachable" set of nodes.

Correcting $d(s)$: Single-Step Error Model

Single-Step Error Model first introduced in BUGSY (Ruml and Do 2007):



$$
\begin{aligned}
e_d &= d(s_{oc}) - (d(s) - 1) \\
e_h &= h(s_{oc}) - (h(s) - c(s, s_{oc}))
\end{aligned}
$$

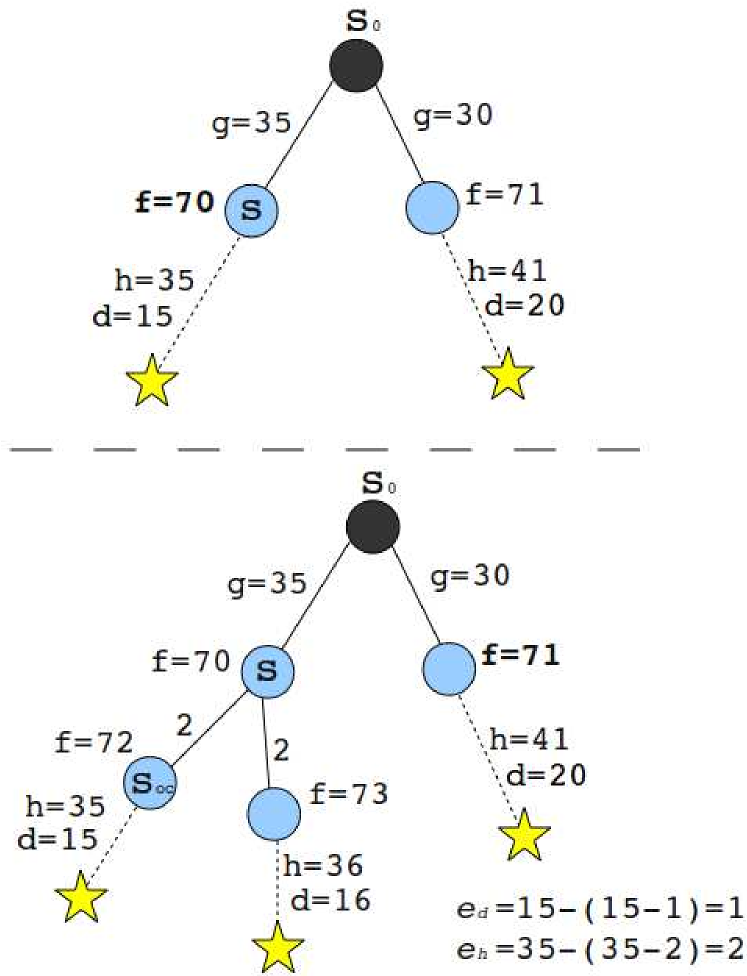
Using average errors $\overline{e_d}$ and $\overline{e_h}$:

$$
\begin{aligned}
\widehat{d}(s) &= d(s) \cdot (1 + \overline{e_d}) \\
\widehat{h}(s) &= h(s) + \overline{e_h} \cdot \widehat{d}(s)
\end{aligned}
$$

s_{oc} is selected as the child state of s with minimal f

Correcting $d(s)$: Single-Step Error Model (Continued)

Our new proposed model is more correct:



$$\begin{aligned} e_d &= d(s_{oc}) - (d(s) - 1) \\ e_h &= h(s_{oc}) - (h(s) - c(s, s_{oc})) \end{aligned}$$

Using average errors $\overline{e_d}$ and $\overline{e_h}$:

$$\begin{aligned} \widehat{d}(s) &= \frac{d(s)}{1 - \overline{e_d}} \\ \widehat{h}(s) &= h(s) + \overline{e_h} \cdot \widehat{d}(s) \end{aligned}$$

s_{oc} is selected as the child state of s with minimal f **excluding the parent of s**

Performs dynamically weighted search on $f'(s) = g(s) + h(s) \cdot w$

- Deadline denoted as T
- Time elapsed denoted as t
- Define $D = h(s_0)$
- Define "desired average velocity" as $V = D/T$
- Define "effective velocity" as $v = (D - h_{min})/t$
- If $v > V + \epsilon_{upper}$, increase w by Δw
- If $v < V - \epsilon_{lower}$, decrease w by Δw

Contract Search

Performs beam-like search, limiting the number of expansions done at each level of the search tree.

- Off-line computation of $k(depth)$ for each level of search tree
- Authors propose models for estimating optimal $k(depth)$ using dynamic programming
- Once $k(depth)$ expansions are made a particular level, that level is disabled

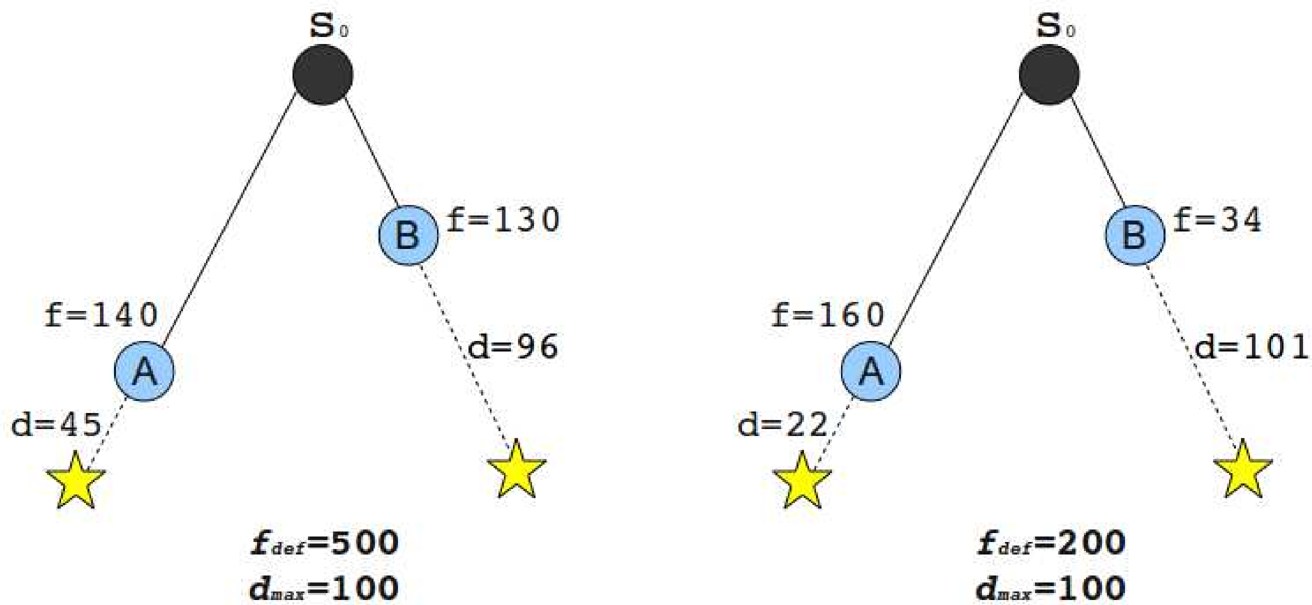
Problems:

- Not applicable to domains where solutions may reside at a wide range of depths
- It takes substantial off-line effort to compute $k(depth)$

Deadline Decision Theoretic Search (DDT)

Motivation

Searching under a deadline involves a great deal of **uncertainty**.



Expected Solution Cost $EC(s)$

f_{def} : cost of default/incumbent solution

f_{exp} : expected value of $f^*(s)$ (if better than incumbent)

P_{goal} : probability of finding solution under s before deadline

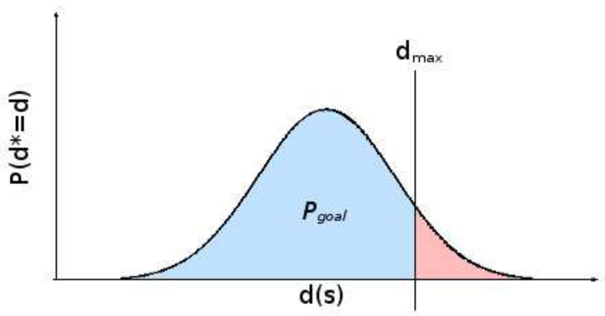
P_{imp} : probability that cost of new solution found under s improves on incumbent

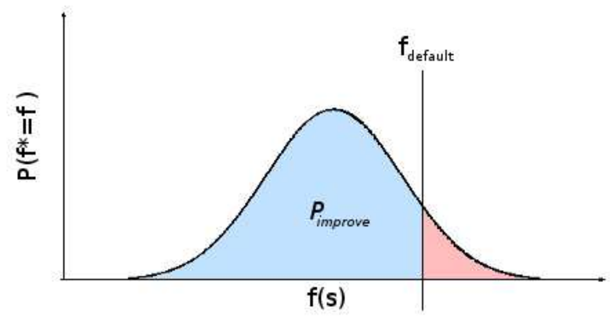
$$EC(s) = \underbrace{P_{goal} \cdot (\overbrace{P_{imp} \cdot f_{exp}}^{\text{Solution found improves on incumbent}} + \overbrace{(1 - P_{imp}) \cdot f_{def}}^{\text{Solution found is worse than incumbent}})}_{\text{Solution found within deadline}} + \underbrace{(1 - P_{goal}) \cdot f_{def}}_{\text{No solution found within deadline}}$$

DDT Search(*initial*, *deadline*, *default solution*)

1. *open* $\leftarrow$ {*initial*}
2. *incumbent* $\leftarrow$ *default solution*
3. while *(time elapsed)* $<$ *(deadline)* loop
5. $s \leftarrow$ remove state from *open* with minimum $EC(s)$
6. if s is a goal and is better than *incumbent*
7. *incumbent* $\leftarrow s$
8. recalculate $EC(s)$ for all s in open and resort
8. otherwise
9. recalculate $EC(s)$
5. $s' \leftarrow$ peek next state from *open* with minimum $EC(s')$
10. if $EC(s) > EC(s')$
11. re-insert s into *open*
12. otherwise
13. expand s, adding child states to *open*
14. return *incumbent*

Off-line Model





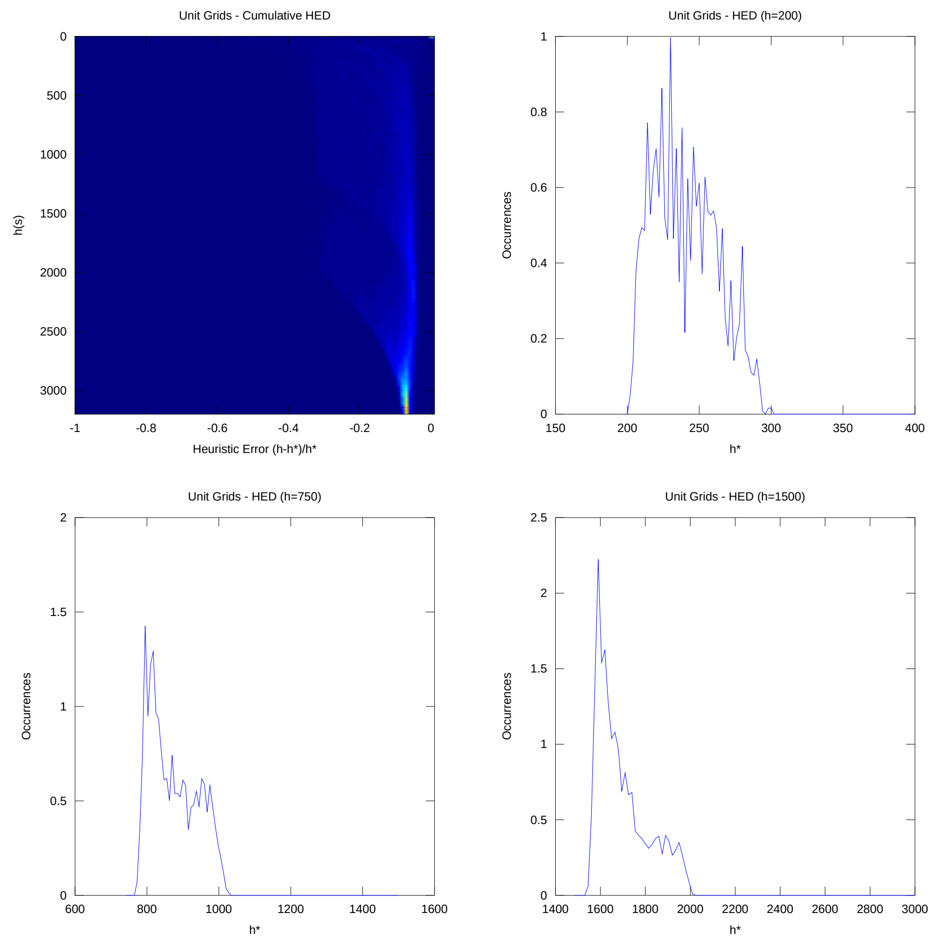
$$P_{goal} = P(d^* \leq d_{max}) \tag{2}$$

$$P_{imp} = P(f^* \leq f_{def}) \tag{3}$$

$$P_{imp} \cdot f_{exp} = \int_{f=0}^{f_{default}} P(f^* = f) \cdot f \tag{4}$$

Off-line Model (Continued)

Measurements on 4-Way 2000x1200 Unit-Cost Gridworld

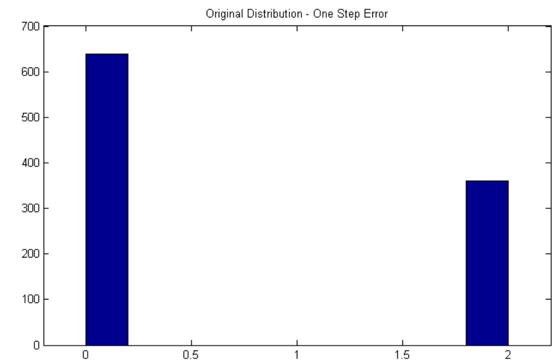


Currently assume h^* and d^* are independant.

On-line Model

Extends one-step error model to support calculation of heuristic distribution functions.

Assume one-step errors are independant identically distributed random variables. See figure for one-step errors in 4-Way Unit-Cost Gridworld.



Then mean one step errors along individual paths are normally distributed according to the Central Limit Theorem with mean and variance:

$$\mu_{\bar{\epsilon}_d} = \mu_{\epsilon_d} \quad (5)$$

$$\sigma^2_{\bar{\epsilon}_d} = \frac{\sigma^2_{\epsilon_d} \cdot (1 - \mu_{\epsilon_d})}{d(s)} \quad (6)$$

On-line Model (Continued)

Using Equations from slide 17 and the assumption that $\bar{\epsilon}_d$ and $\bar{\epsilon}_h$ are normally distributed, we can calculate the CDF for $d^*(s)$:

$$cdf_{d^*}(x) = \frac{1}{2} \cdot \left(1 + \mathrm{ERF}\left(\frac{\left(\frac{x-d(s)}{x} - \mu_\epsilon\right)}{\left(\sqrt{2 \cdot \frac{\sigma_\epsilon^2 \cdot (1-\mu_\epsilon)}{d(s)}}\right)}\right)\right) \tag{7}$$

For a given value of d^* we can assume f^* is normally distributed with mean and variance:

$$\mu_{f^*} = g(s) + h(s) + \mu_{\epsilon_h} \cdot d^*(s) \tag{8}$$

$$\sigma^2_{f^*} = \sigma^2_{\epsilon_h} \cdot (d^*(s)) \tag{9}$$

Details can be found in thesis document.

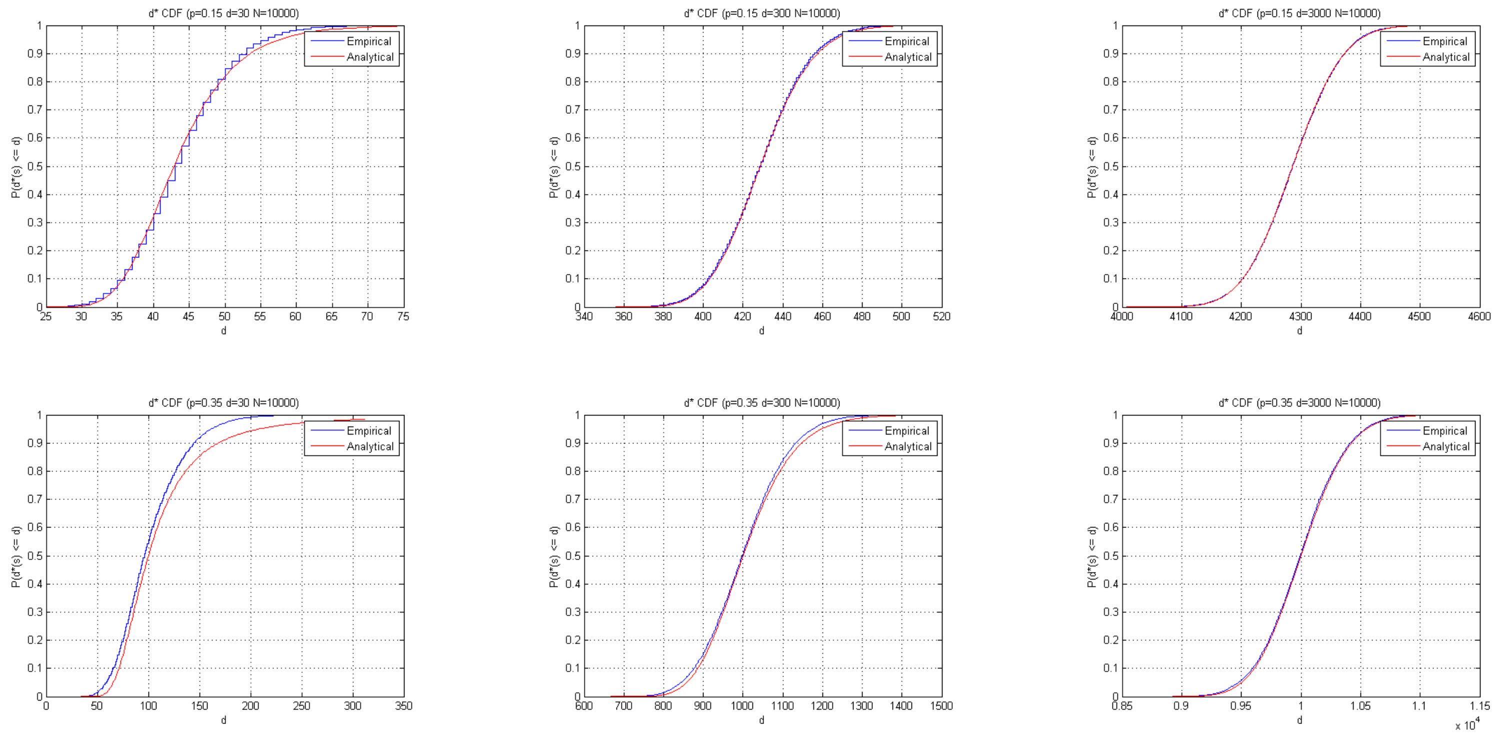
On-line Model (Continued)

Using CDF for d^* and Gaussian PDF for calculating $P(f^* = f|d^* = d)$ we can calculate $EC(s)$ as follows:

$$
\begin{aligned}
P_{imp} &= P(f^* \leq f_{default}|d^* = d) \\
EC(s|d^* = d) &= \left(\int_{f=0}^{f_{default}} P(f^* = f|d^* = d) \cdot f\right) + (1 - P_{imp}) \cdot f_{def} \\
EC(s) &= \left(\int_{d=0}^{d_{max}} EC(s|d^* = d)\right) + (1 - P_{goal}) \cdot f_{def}
\end{aligned}
$$

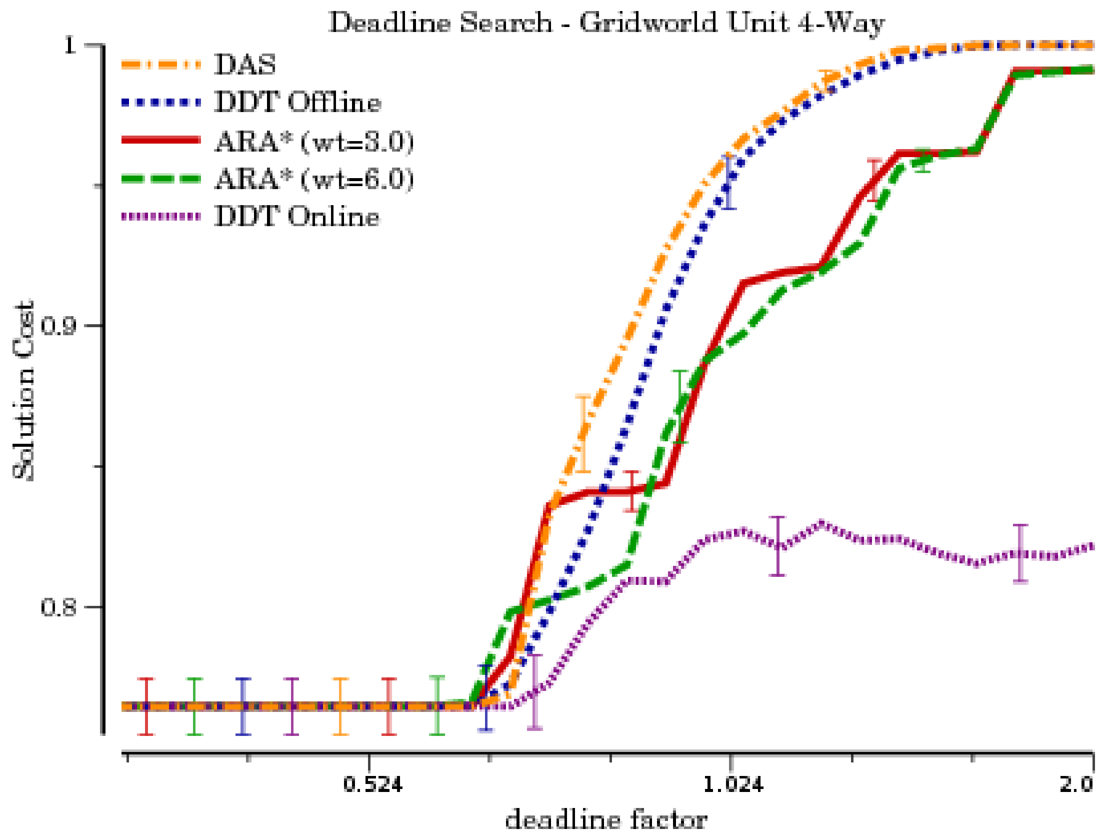
On-line Model Verification

Monte Carlo analysis performed on $d^*(s)$ model using heuristic error from 4-Way Unit-Cost Gridworld.



Model of $d^*(s)$ is accurate unless $\bar{\epsilon}_d$

Results: 4-Way 2000x1200 Unit-Cost Gridworld (p=0.35)



Even in optimistic case DDT does not outperform DAS!

Future Work

- More empirical evaluation of DAS and DDT
- Evaluate other methods of calculating $\widehat{d}(s)$ for DAS
- Evaluate other methods of calculating d_{max} for DAS/DDT
- Evaluate accuracy of probabilistic one-step error model
- Modify Real-Time search to apply to Contract Search