Master’s Thesis:
Heuristic Search Under a Deadline

Austin Dionne

University of New Hampshire
Department of Computer Science
austin.dionne at gmail.com
Thanks to:

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

Introduction
- Heuristic Search
- Problem Def.
- Thesis Statement
- Contributions

Related Work

DAS

Conclusion

DDT
Heuristic Search (Continued)

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

---

**Example**

\[ g(s) = 2 + 2 = 4 \]
\[ h'(s) = 2 + 3 + 1 + 4 = 10 \]
\[ f'(s) = g(s) + h'(s) = 14 \]
\[ d'(s) = 4 \]

\[ h(s) = 5 \]
\[ f(s) = 4 + 5 = 9 \]
\[ d(s) = 2 \]
\[ \hat{d}(s) = 3 \]
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.
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.
In this thesis we have proposed:

- Corrected single-step error model for $\hat{d}(s)$ and $\hat{h}(s)$
- 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!
Problem with Time Constrained Search:

- Parameters abound! \((\epsilon_{upper}, \epsilon_{lower}, \Delta w)\)

- 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!
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 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
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 \( \hat{d}(s) \) and the current behavior of the search
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 \( \hat{d}(n) \leq d_{max} \) (solution is reachable)
  - Expand \( n \), add children to open list
- Otherwise (solution is unreachable)
  - Add \( n \) to pruned list
Search Vacillation

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

Before:
- $s_0$
- $g=35$
- $f=70$
- $h=35$
- $d=15$

After:
- $s_0$
- $g=35$
- $f=71$
- $h=41$
- $d=20$

Results

Conclusion

Austin Dionne

Heuristic Search Under Deadlines – 20 / 56
Expansion Delay

Maintain a running expansion counter during search.

At state expansion, define expansion delay as:

$$\Delta e = (\text{current exp counter}) - (\text{exp counter at generation})$$
Use mean expansion delay $\overline{\Delta e}$ to calculate $d_{max}$:

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

Introduction
Related Work
DAS
Motivation
Algorithm (1)
Vacillation
Exp Delay
Calc $d_{max}$
Algorithm (2)
Results
Contribution
Conclusion

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 $\hat{d}(n) \leq d_{max}$ (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 $d_{max}$
Start again with a set of nodes with “reachable” solutions:

Estimated expansions remaining: **150**

<table>
<thead>
<tr>
<th>Pruned List:</th>
<th>( f(n) )</th>
<th>( \tilde{d}(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>2.</td>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>3.</td>
<td>25</td>
<td>22</td>
</tr>
<tr>
<td>4.</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td>5.</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>6.</td>
<td>41</td>
<td>34</td>
</tr>
<tr>
<td>7.</td>
<td>48</td>
<td>42</td>
</tr>
<tr>
<td>8.</td>
<td>55</td>
<td>50</td>
</tr>
<tr>
<td>9.</td>
<td>66</td>
<td>56</td>
</tr>
<tr>
<td>10.</td>
<td>70</td>
<td>67</td>
</tr>
</tbody>
</table>

Sum of \( \tilde{d}(n) \) ≤ exp remaining

14+20+22+30+40 = 126 ≤ 150
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

Introduction

Related Work

DAS

- Motivation
- Algorithm (1)
- Vacillation
- Exp Delay
- Calc $d_{max}$
- Algorithm (2)

Results

- Results
- results
- Conclusion

Conclusion

DDT

15-Puzzle

Gridworld

Dynamic Robot

2 Models:
- Unit-Cost
- Inverse Weighted

Uniformly Distributed Random Obstacles (p=0.35)

2 Models:
- Unit-Cost
- Life-Cost

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

Korf 100 Tiles

- DAS
- ARA* (wt=3.0)
- ARA* (wt=6.0)
- RWA* (wt=3.0)
- CS
- RWA* (wt=6.0)

Solution Quality vs Deadline (seconds)
Results: Weighted 15-Puzzle

Solution Quality

Weighted Korf 100 Tiles

DDT

Motivation

Algorithm (1)

Vacillation

Exp. Delay

Calc. $d_{max}$

Algorithm (2)

Results

Conclusions

Conclusion
Results: 4-Way 2000x1200 Unit-Cost Gridworld (p=0.35)
Results: 4-Way 2000x1200 Life-Cost Gridworld (p=0.35)
Results: Dynamic Robot Navigation

Introduction

Related Work

DAS
- Motivation
- Algorithm (1)
- Vacillation
- Exp Delay
- Calc $d_{max}$
- Algorithm (2)
- Results

Results

Conclusion

DDT

Austin Dionne

Heuristic Search Under Deadlines – 32 / 56
Results: Overall

Introduction

Related Work

DAS

- Motivation
- Algorithm (1)
- Vacillation
- Exp Delay
- Calc $d_{\text{max}}$
- Algorithm (2)
- Results

Conclusion

DDT

Austin Dionne
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
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.
In this thesis we have proposed:

- Corrected single-step error model for \( \hat{d}(s) \) and \( \hat{h}(s) \)

- 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!**
<table>
<thead>
<tr>
<th>Section</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td></td>
</tr>
<tr>
<td>Related Work</td>
<td></td>
</tr>
<tr>
<td>DAS</td>
<td></td>
</tr>
<tr>
<td>Conclusion</td>
<td></td>
</tr>
<tr>
<td>Back-up Slides</td>
<td>■ DAS Pseudo-Code</td>
</tr>
<tr>
<td></td>
<td>■ $\hat{d}(s)$</td>
</tr>
<tr>
<td>DDT</td>
<td></td>
</tr>
</tbody>
</table>

**Back-up Slides**
Deadline Aware Search \((\text{starting state}, \text{deadline})\)

1. \(\text{open} \leftarrow \{\text{starting state}\}\)
2. \(\text{pruned} \leftarrow \{\}\)
3. \(\text{incumbent} \leftarrow \text{NULL}\)
4. while \((\text{time}) < (\text{deadline})\) and \(\text{open}\) is non-empty
5. \(d_{\text{max}} \leftarrow \text{calculate\_d\_max}()\)
6. \(s \leftarrow \text{remove state from open with minimal } f(s)\)
7. if \(s\) is a goal and is better than \(\text{incumbent}\)
8. \(\text{incumbent} \leftarrow s\)
9. else if \(\hat{d}(s) < d_{\text{max}}\)
10. for each child \(s'\) of state \(s\)
11. add \(s'\) to \(\text{open}\)
12. else
13. add \(s\) to \(\text{pruned}\)
14. if \(\text{open}\) is empty
15. \(\text{recover\_pruned\_states}(\text{open}, \text{pruned})\)
16. return \(\text{incumbent}\)
Recover Pruned States\( (open, pruned) \)
18. \( \text{exp} \leftarrow \text{estimated expansions remaining} \)
19. while \( \text{exp} > 0 \) and \( pruned \) is non-empty loop
20. \( s \leftarrow \text{remove state from } pruned \text{ with minimal } f(s) \)
21. add \( s \) to \( open \)
23. \( \text{exp} = \text{exp} - \hat{d}(s) \)

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):

\[
\begin{align*}
ed &= d(s_{oc}) - (d(s) - 1) \\
e_h &= h(s_{oc}) - (h(s) - c(s, s_{oc}))
\end{align*}
\]

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

\[
\begin{align*}
\hat{d}(s) &= d(s) \cdot (1 + \bar{e_d}) \\
\hat{h}(s) &= h(s) + \bar{e_h} \cdot \hat{d}(s)
\end{align*}
\]

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

---

Austin Dionne

Heuristic Search Under Deadlines – 22 / 56
Our new proposed model is more correct:

\[ e_d = d(s_{oc}) - (d(s) - 1) \]
\[ e_h = h(s_{oc}) - (h(s) - c(s, s_{oc})) \]

Using average errors \( \overline{e_d} \) and \( \overline{e_h} \):

\[ \hat{d}(s) = \frac{d(s)}{1 - \overline{e_d}} \]
\[ \hat{h}(s) = h(s) + \overline{e_h} \cdot \hat{d}(s) \]

\( s_{oc} \) is selected as the child state of \( s \) with minimal \( f \) excluding the parent of \( s \).
Time Constrained Search

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 $\Delta w$
- If $v < V - \epsilon_{lower}$, decrease $w$ by $\Delta w$
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)
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

The probability of finding a solution under \( s \) before the deadline is a weighted average of the expected value of the new solution and the cost of the default solution, considering the probabilities of finding a solution and the improvement of the solution.
**Algorithm**

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

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

\[ P_{goal} = P(d^* \leq d_{max}) \]  \hspace{1cm} (2)

\[ P_{imp} = P(f^* \leq f_{def}) \]  \hspace{1cm} (3)

\[ P_{imp} \cdot f_{exp} = \int_{f=0}^{f_{default}} P(f^* = f) \cdot f \]  \hspace{1cm} (4)
Off-line Model (Continued)

Measurements on 4-Way 2000x1200 Unit-Cost Gridworld

Currently assume $h^*$ and $d^*$ are independant.
Extends one-step error model to support calculation of heuristic distribution functions.

Assume one-step errors are independent 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:

\[
\mu_{\bar{\epsilon}_d} = \mu_{\epsilon_d} \\
\sigma^2_{\bar{\epsilon}_d} = \sigma^2_{\epsilon_d} \cdot \left(1 - \mu_{\epsilon_d}\right) / d(s)
\]
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 $d^*(s)$:

\[
\text{cdff}_{d^*}(x) = \frac{1}{2} \cdot \left( 1 + \text{ERF} \left( \frac{x - d(s)}{\sqrt{2} \cdot \sigma^2 \cdot (1 - \mu \epsilon)} d(s) \right) \right)
\] (7)

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

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

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

Details can be found in thesis document.
Using CDF for $d^*$ and Gaussian PDF for calculating $P(f^* = f | d^* = d)$ we can calculate $EC(s)$ as follows:

$$P_{imp} = P(f^* \leq f_{default} | d^* = d)$$

$$EC(s | d^* = d) = \left( \int_{f=0}^{f_{default}} P(f^* = f | d^* = d) \cdot f \right) + (1 - P_{imp}) \cdot f_{def}$$

$$EC(s) = \left( \int_{d=0}^{d_{max}} EC(s | d^* = d) \right) + (1 - P_{goal}) \cdot f_{def}$$

---

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

Model of $d^*(s)$ is accurate unless $\bar{\epsilon}_d$
Even in optimistic case DDT does not outperform DAS!
Future Work

- More empirical evaluation of DAS and DDT
- Evaluate other methods of calculating $\hat{d}(s)$ 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