Efficient Activation-based Working Memory Forgetting

Nate Derbinsky
University of Michigan

Why Architecturally Forget Working Memory Elements?

Bounding Memory Retrievals

- Procedural (Forgy, 1982)
- Episodic (Derbinsky & Laird, 2009)

Attention-biased Behavior

- Episodic (Nuxoll & Laird, 2007)
- Appraisals (Marinier & Laird, 2004)

Reduced Programmer Burden

- Topographic locality (Laird, Derbinsky, & Voigt 2011)

Challenges

Model Efficacy Ongoing

Reflect intentional focus (Nuxoll, Laird & James 2004; Chong 2003)

Implementation Efficiency This Talk

Scale to large memory stores and dynamic agents

General Agent Development John Laird

Support agent robustness to dynamic knowledge availability

Working Memory Activation in Soar

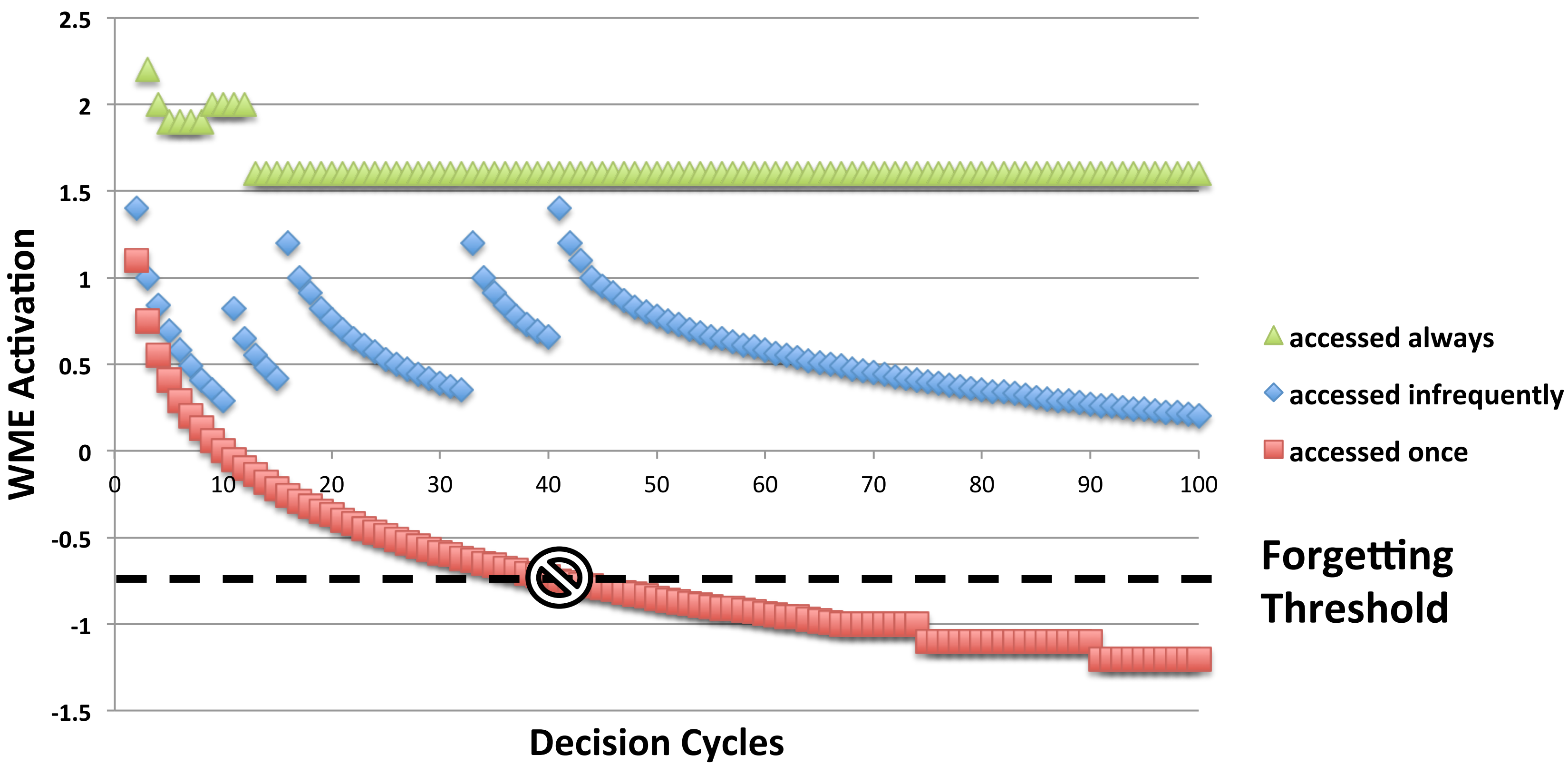
Base-level Activation

- Activating events: create new WME, test WME (Nuxoll, Laird & James 2004)
- Bounded history window (Petrov 2006)

Application

- Parameterized episodic retrieval bias

Activation-based Forgetting Illustrated



Problem. Efficiently detect when element activation falls below threshold (and thus should be removed from working memory)

Characteristics Pertinent to Evaluating an Activation-based Forgetting Mechanism

- Number of elements in working memory (**N**)
  - Large memory: ↑**N**

- Number of WME activation events/cycle (**E**)
  - Dynamic agent: ↑**E**

- WME lifetime (**L**)
  - Frequently accessed elements: ↑**L**
  - High element turnover: ↓**L**

Naïve Approach

<u>Algorithm</u>

- At each decision
  - For each WME
    - If (Activation < Threshold)
      - Forget

<u>Efficiency Evaluation</u>

- Per Decision: O(**N**)
- Per WME: O(**L**)

Efficient Approach: Decay Prediction

<u>Algorithm</u> ~ (Nuxoll, Laird & James 2004)

- On new activation event
  - *Predict** time of future decay
  - Add to *cycle-indexed priority queue**
- Each cycle
  - Remove decayed elements at front of priority queue

<u>Efficiency Evaluation</u>

- Per Decision: O(# decayed WMEs + **E***[Prediction Cost])

Efficient Decay Prediction

1. Cheaply approximate decay on each access
   – Underestimate time of decay by treating each memory access independently: O(1)

2. Exact determination
   – Binary parameter search: $O(\log_2 \mathbf{L})$
   – Not needed if WME is removed by #1 estimate
   – Otherwise, reduced by the degree to which #1 is accurate

Novel Base-level Decay Approximation

Given

constants

- Decay threshold (θ)
- Decay parameter value (*d*)

and a set of memory accesses…

- Time since access (*s*)
- Number of accesses (*n*)

solve for…

- Time till memory decay (*t*)

Algorithm

For each memory access…

$$\ln(n \cdot [t+s]^{-d}) = \theta$$

$$\ln(n) - d \cdot \ln(t+s) = \theta$$

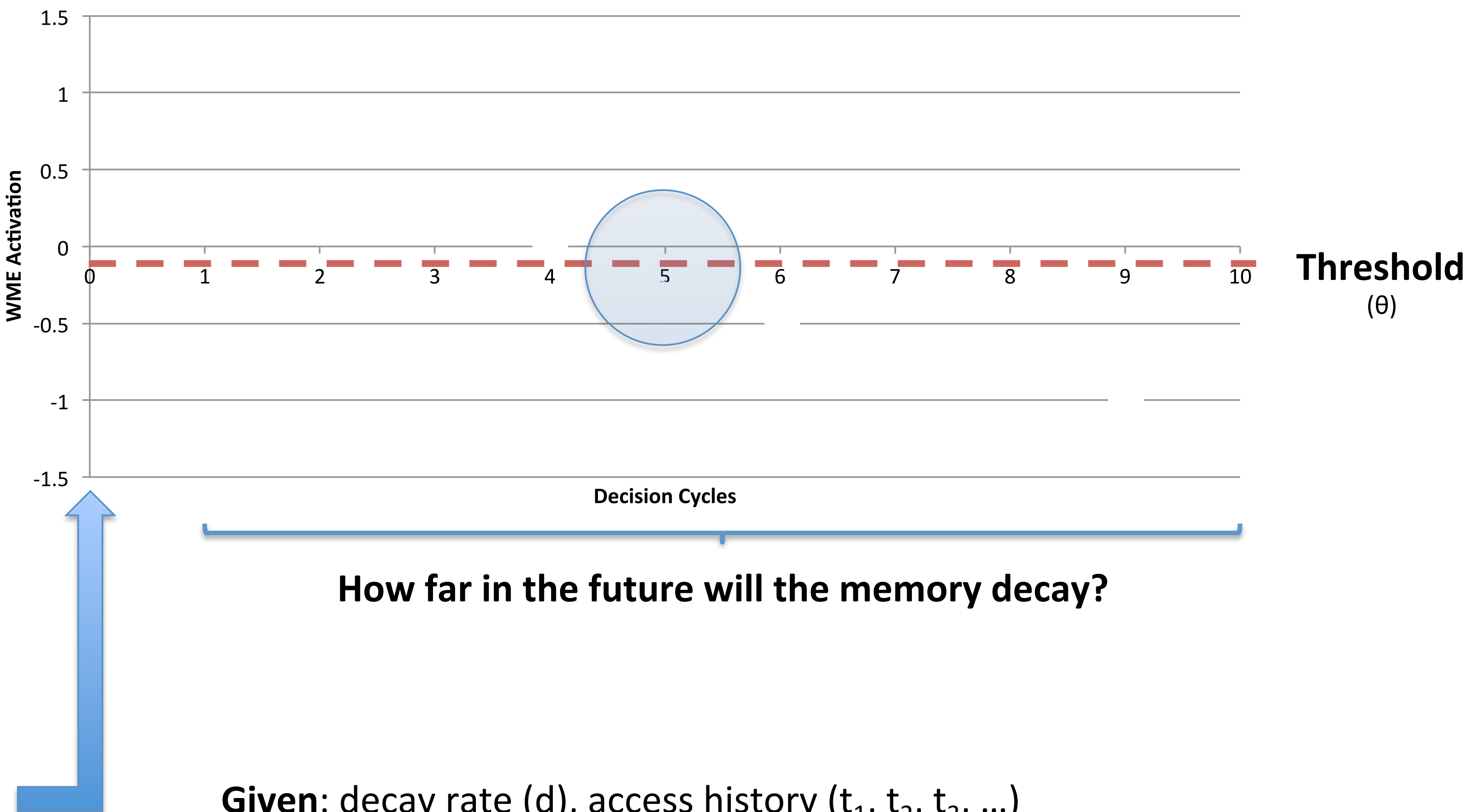
$$\ln(t+s) = \frac{\theta - \ln(n)}{-d}$$

$$t = e^{\frac{\theta - \ln(n)}{-d}} - s$$

Decay estimate = $\sum t$

Activation-based Forgetting

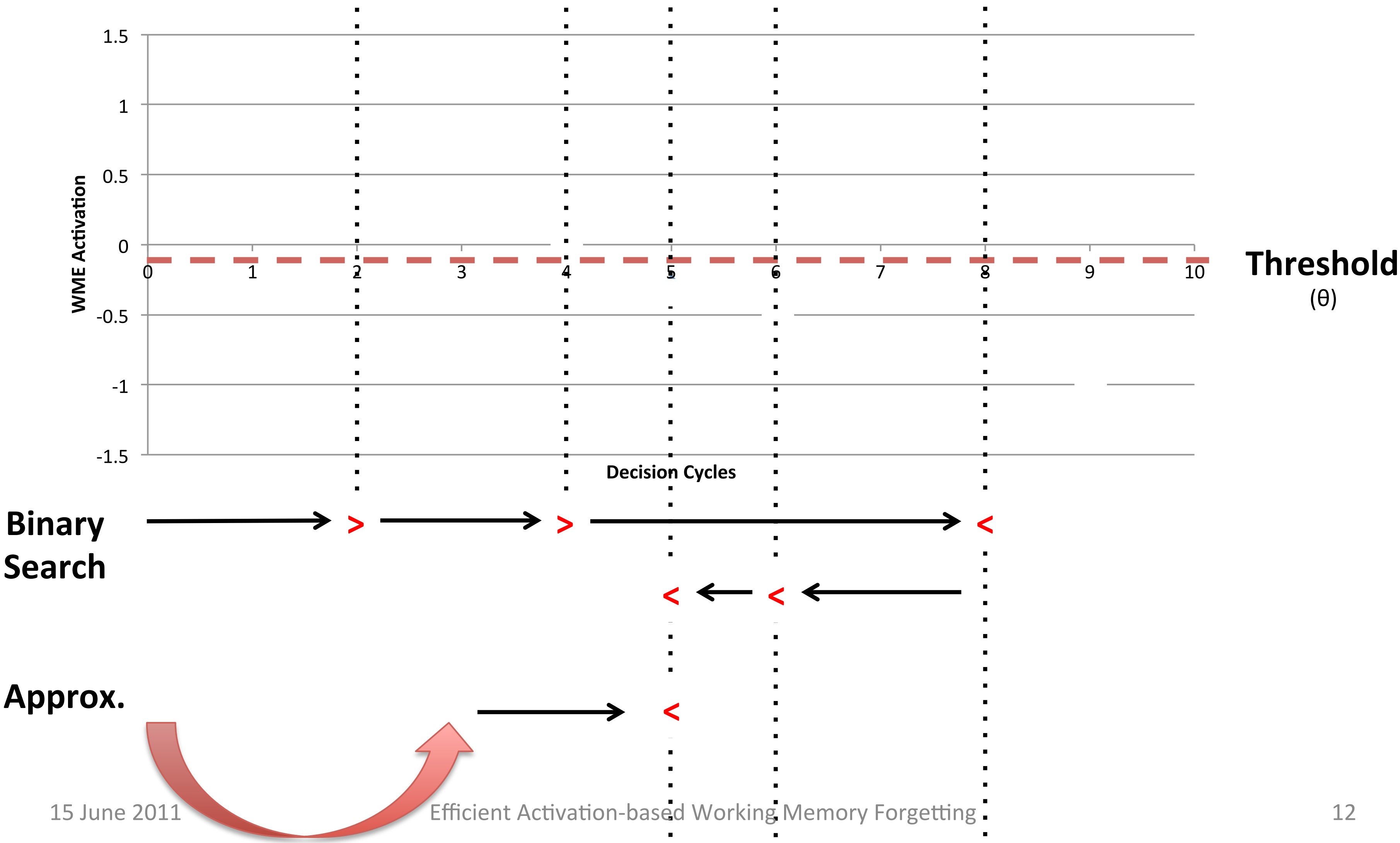
Example



Given: decay rate (d), access history (t_1, t_2, t_3, …)

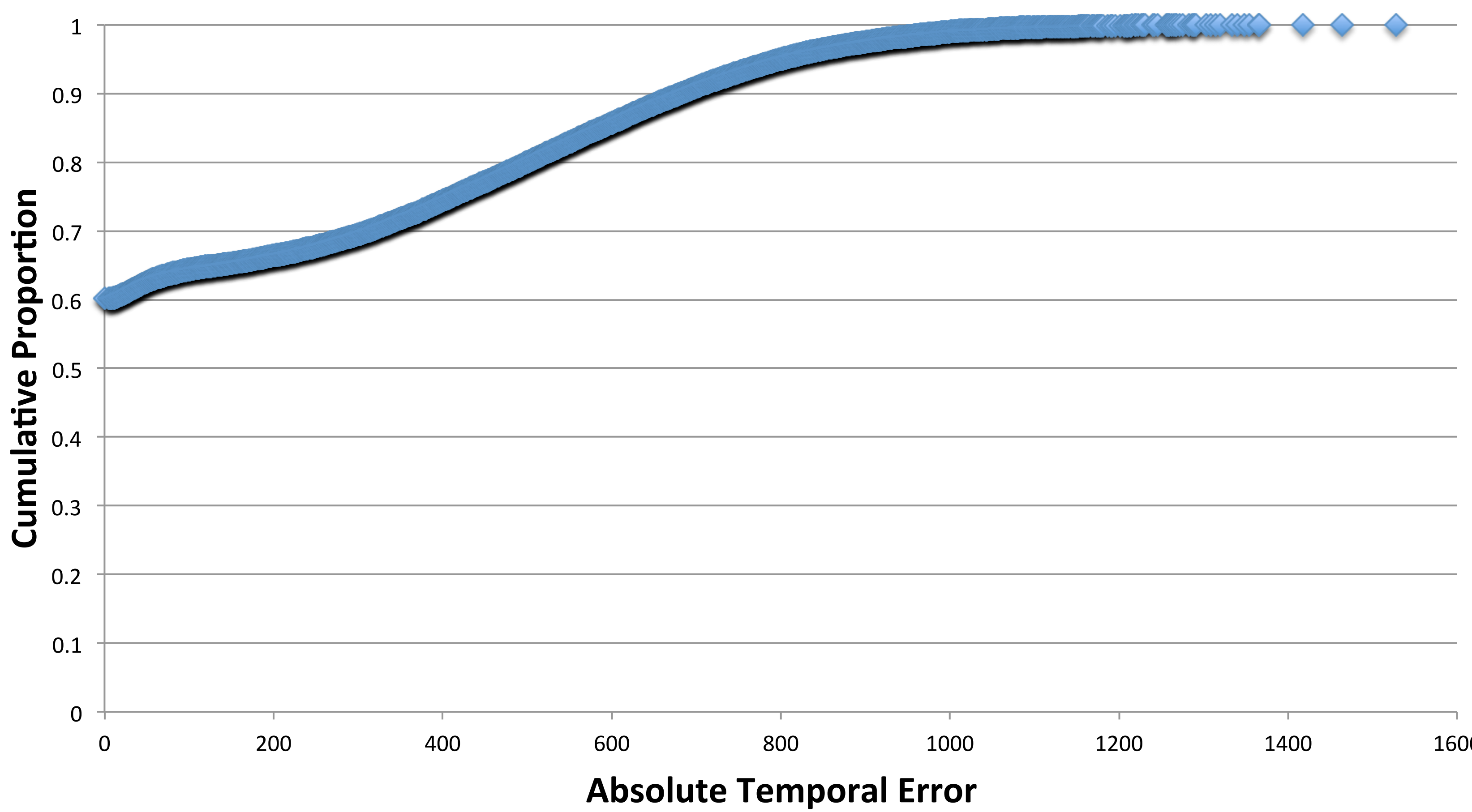
Activation-based Forgetting

Example



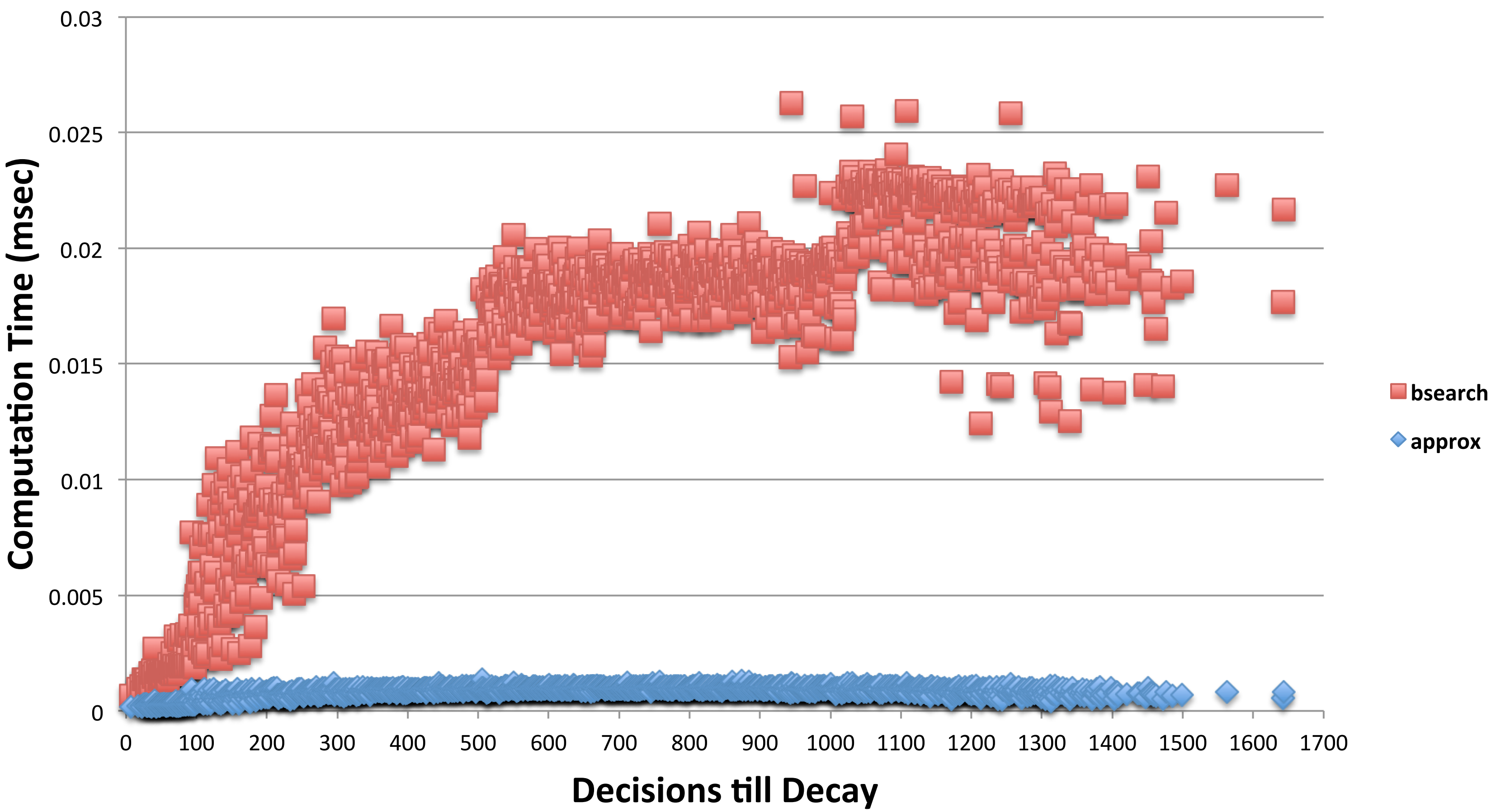
Approximation Quality

50k random histories



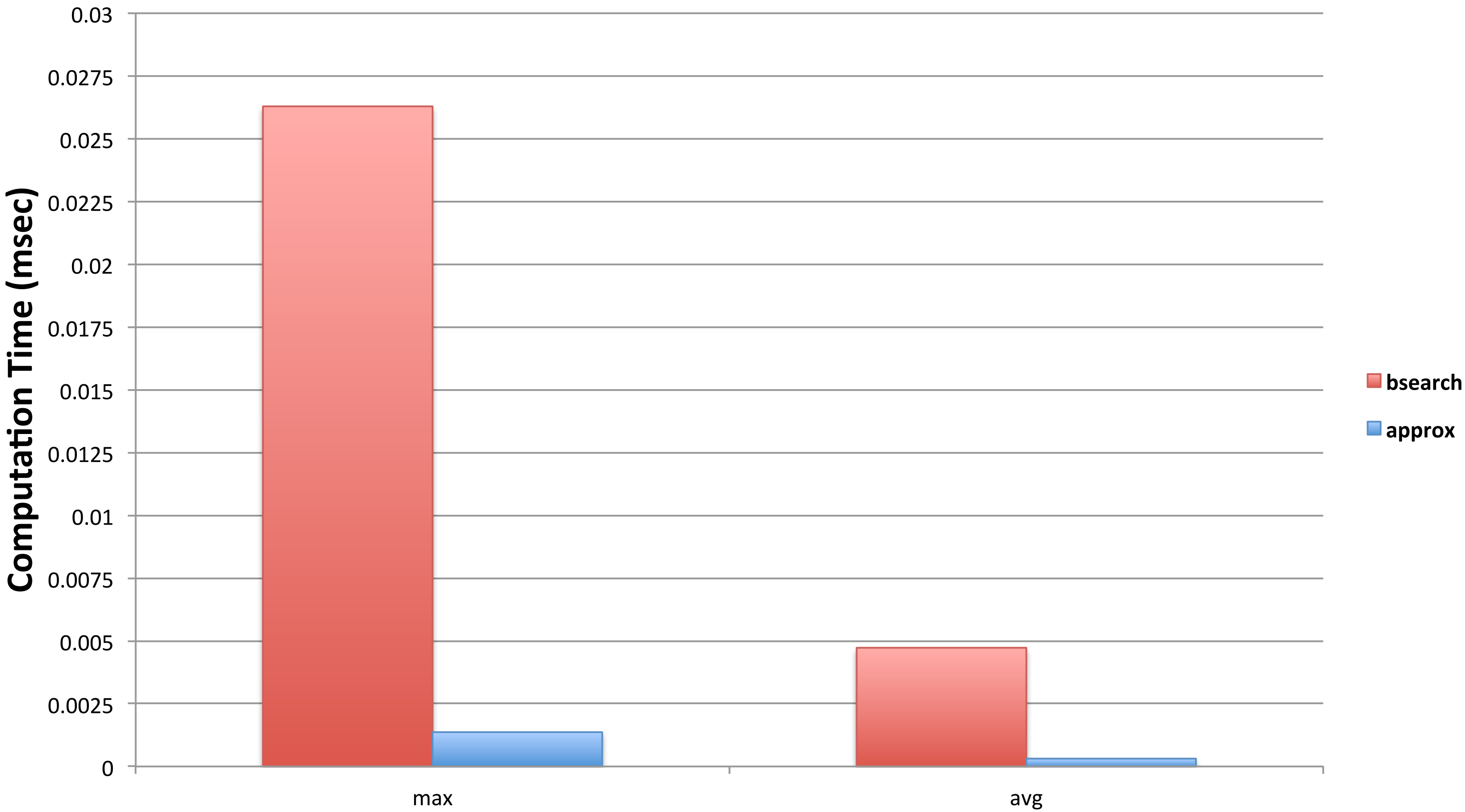
Prediction Computation Comparison

Complexity (50k random histories)



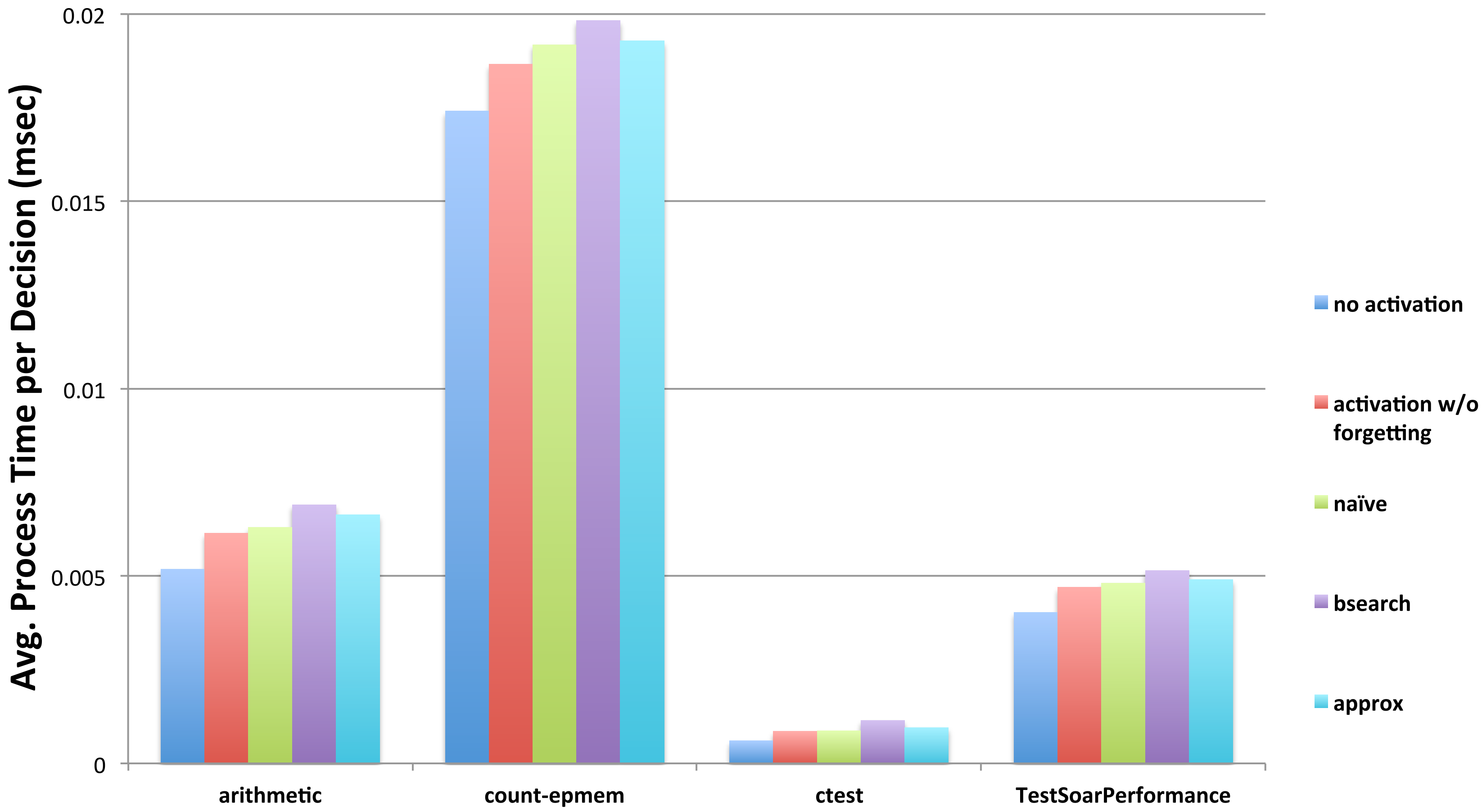
Prediction Computation Comparison

Aggregate Prediction Time (50k random histories)



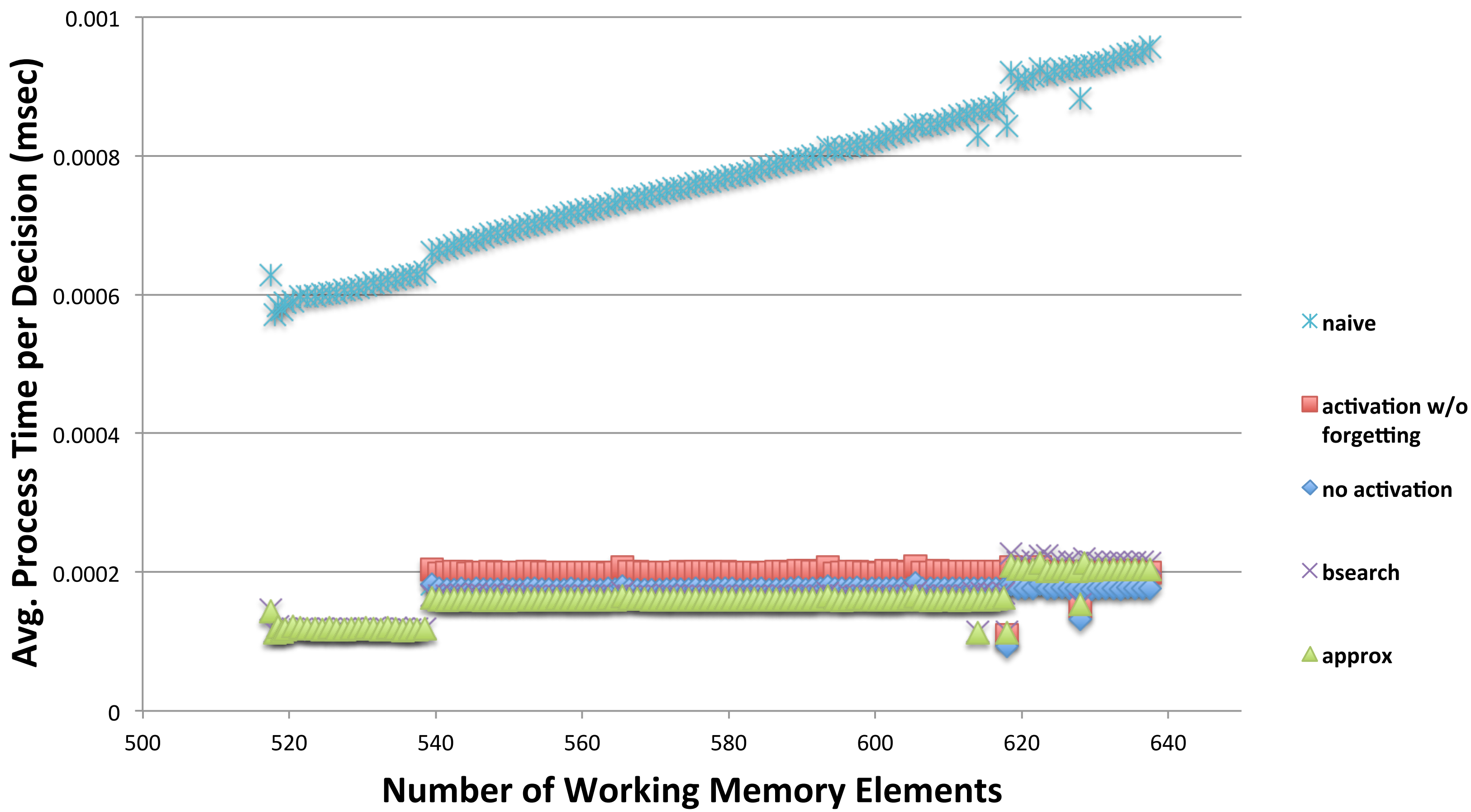
Preliminary Agent Results: Counting

small memory, frequent changes



Preliminary Agent Results: Caching

Monotonically Growing Memory



Evaluation

Nuggets

- Preliminary empirical evidence of efficient activation-based forgetting of WMEs
- Implemented in Soar 9.3.1

Coal

- Limited agent evaluation