Specifying Adaptive Programs a challenge for RL

Specifying Adaptive Programs a challenge for RL

with

Chris Lin &

Mausam GoogalLastNamePolicySucks

Motivation:

Optimal Control of Crowdsourced Workflows

Crowdsourcing

- Obtaining ideas / content by soliciting contributions from a large group of people
- Combine the efforts of volunteers/part-time workers (each contributing a small portion) which adds to a relatively large or significant result

Widespread Success of Crowdsourcing

- Obtaining ideas / content by soliciting contributions from a large group of people
- Combine the efforts of volunteers/part-time workers (each contributing a small portion) which adds to a relatively large or significant result

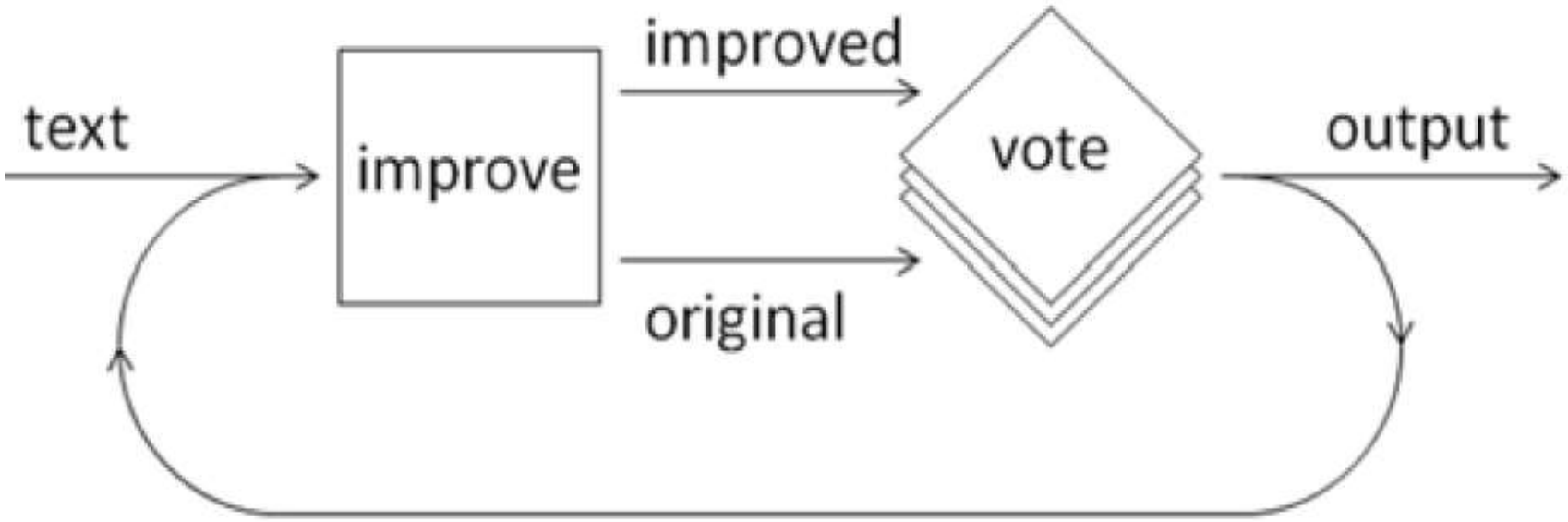
Large Tasks from Micro-Contributions

- Challenges
  - Small work units
  - Reliability & skill of individual workers vary

- Therefore
  - Use workflow to aggregate results & ensure quality
  - Manage workers with (unreliable) workers

  - *Eg*, Turkit, Automan: workflow programming langs

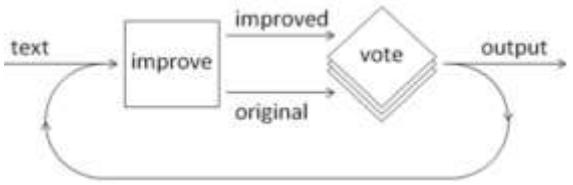
Ex: Iterative Improvement



[Little et al, 2010]

Iterative Improvement

[Little et al, 2010]



First version

A parial view of a pocket calculator together with some coins and a pen.

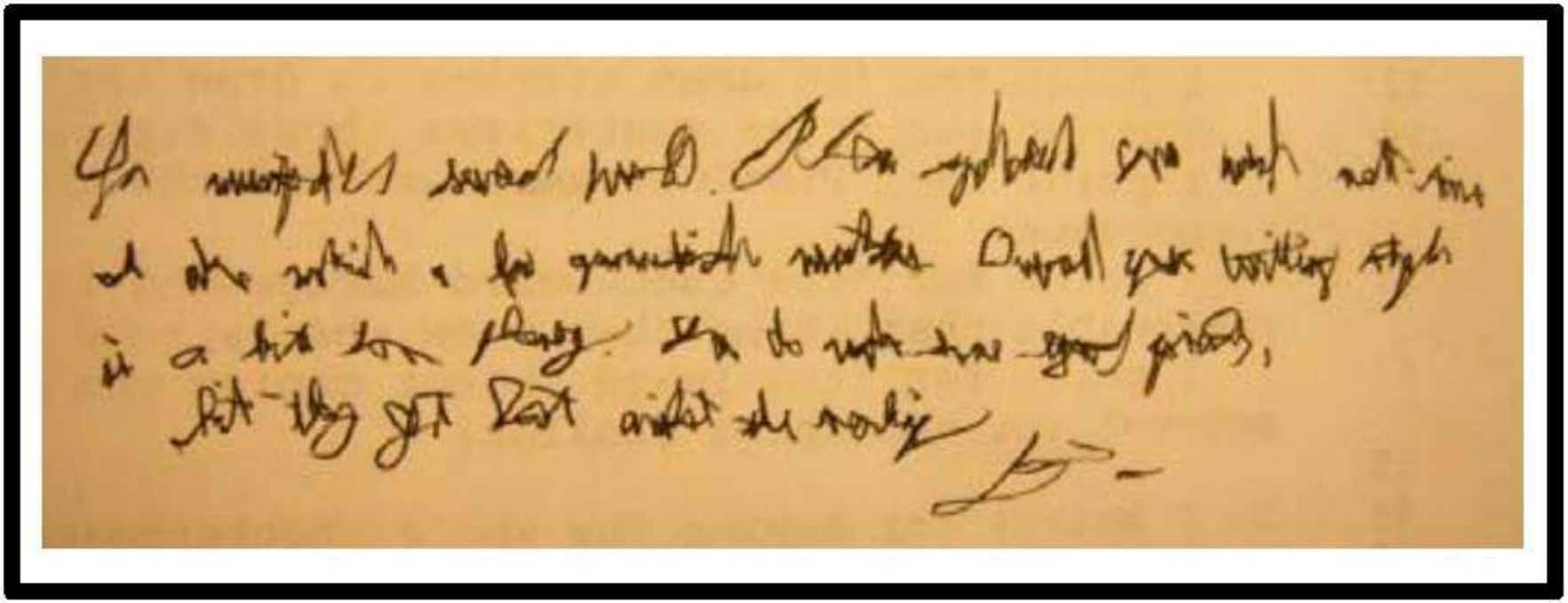
Version after 8 iterations

A CASIO multi-function, solar powered scientific calculator.

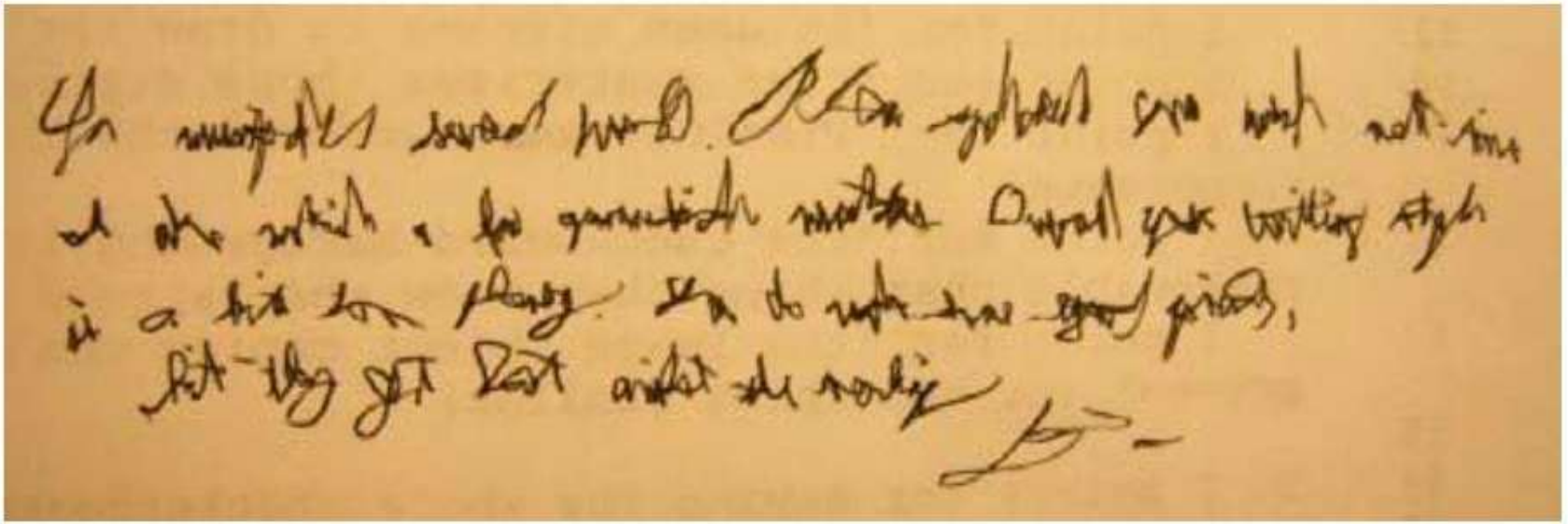
A blue ball point pen with a blue rubber grip and the tip extended.

Six British coins; two of £1 value, three of 20p value and one of 1p value.

Seems to be a theme illustration for a brochure or document cover treating finance - probably personal finance.



[Little et al, 2010]

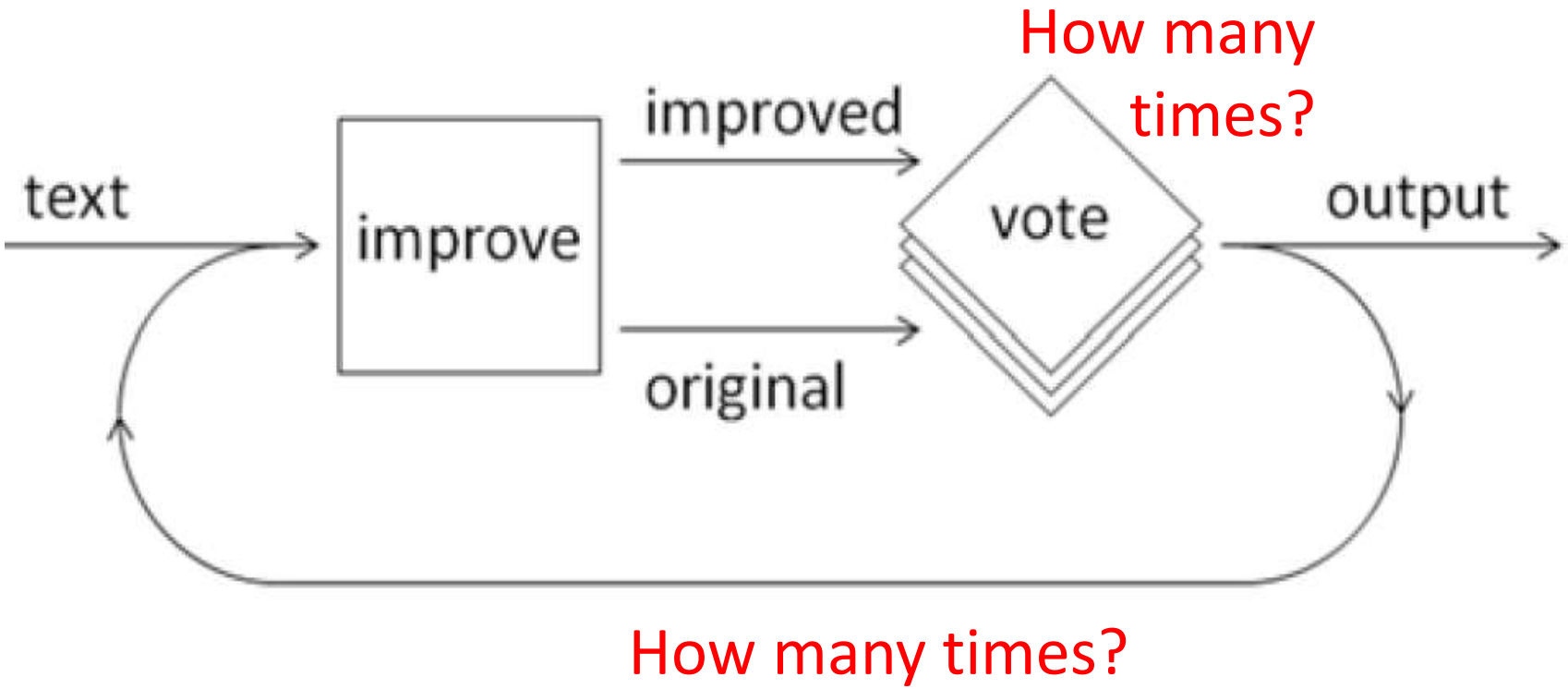


"You (misspelled) (several) (words). Please spellcheck your work next time. I also notice a few grammatical mistakes. Overall your writing style is a bit too phoney. You do make some good (points), but they got lost amidst the (writing). (signature)"

According to our ground truth, the highlighted words should be "flowery", "get", "verbiage" and "B-" respectively.

[Little et al, 2010]

Workflow Control Problem

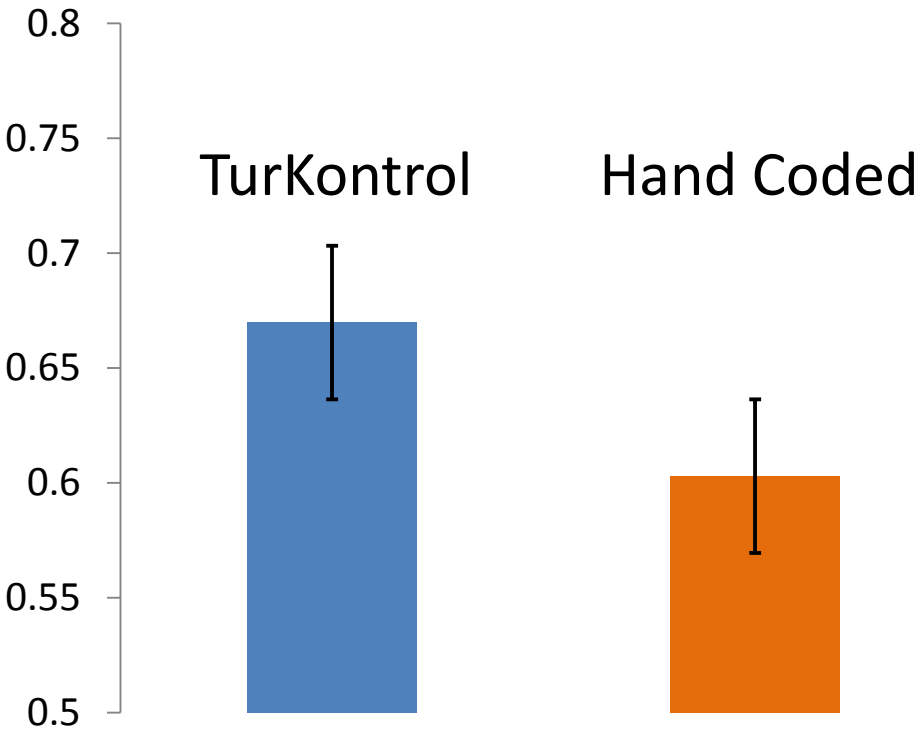


Decision-Theoretic Control

- Workflow = POMDP
  - Partially-Observable Markov Decision Process
- World State: Quality of artifact(s), skill of workers
- Belief State: Probability distribution over world states
- Actions: Submission & observation of HITs
  - Eg, improve — prob distribution on new artifact
  - Eg, submit ballot job — Bayesian update on quality
    EM update on difficulty, worker diligence
- Objective: Maximize $\mathbf{E}[R(w) - \sum c]$

[Dai et al, AAAI-2010]

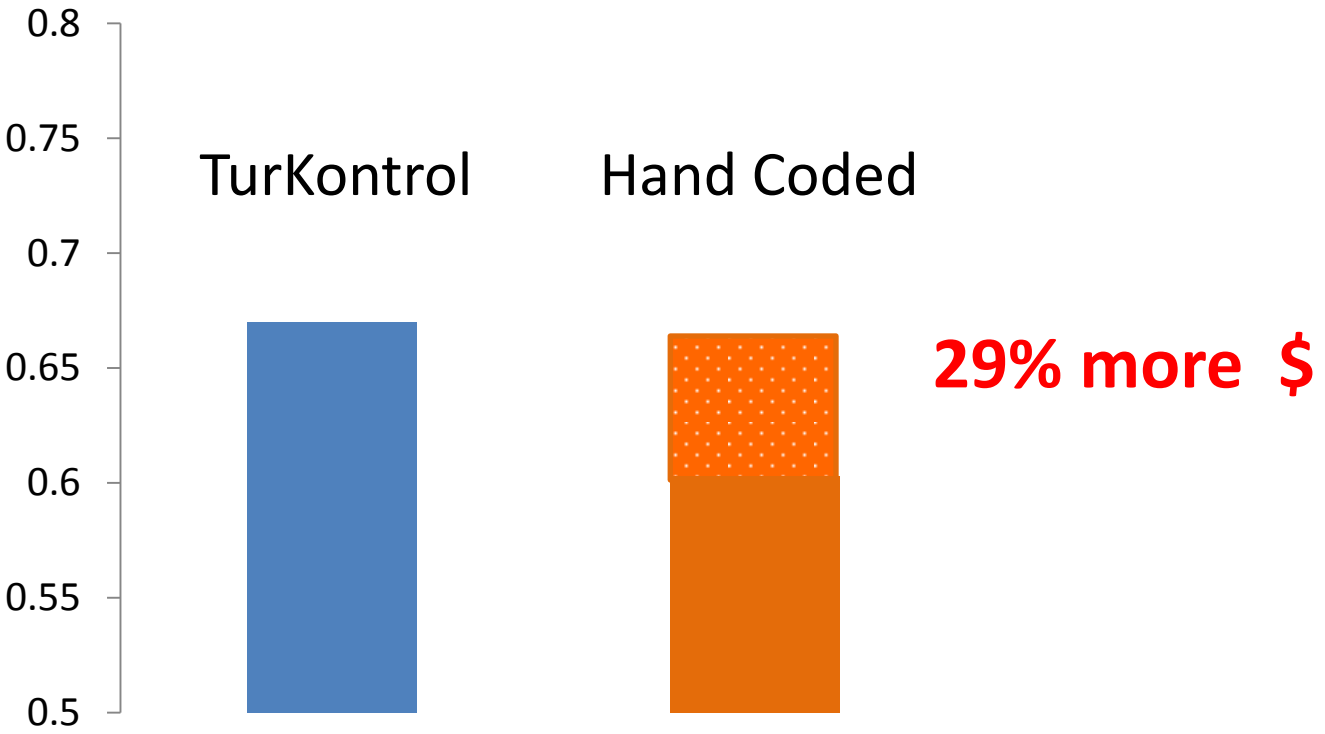
Quality (equal cost)



40 images, same average cost

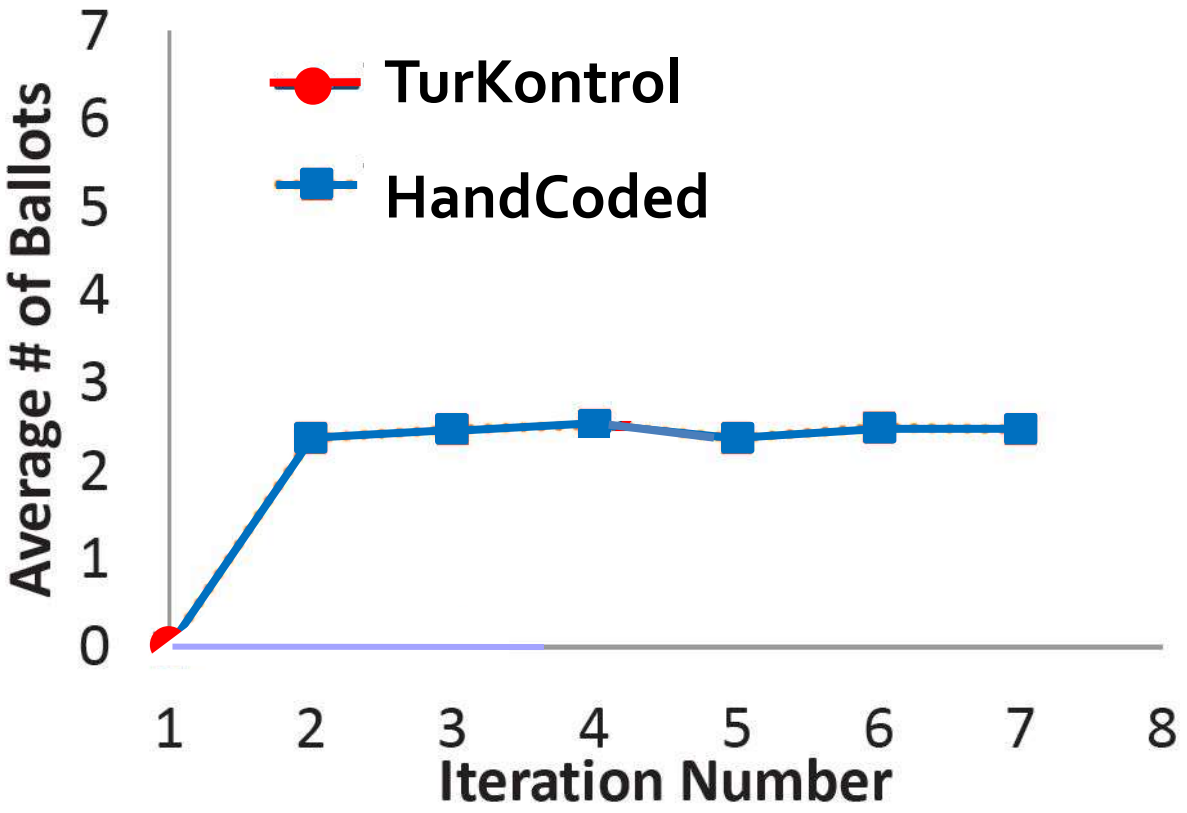
[Dai et al, AAAI-2011]

Cost (equal quality)



[Dai et al, AAAI-2011]

Observation: Ballot Use



[Dai et al, AAAI-2011]

World Domination?

Cool, huh?
Why don't you use it too?

?!?!?
My students don't really get this POMDP, RL stuff

Specifying a POMDP

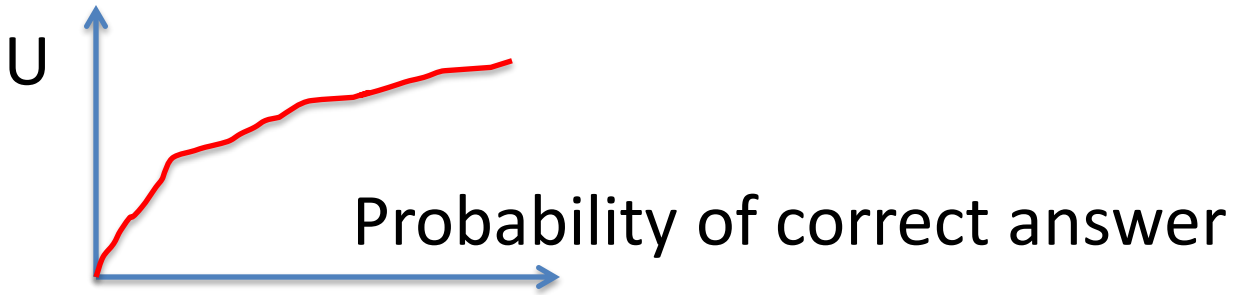
- Actions
  - ☺ Worker tasks – user has to specify anyway
- Transition & Observation Probabilities
  - ☺ Learned from experience (*vs* PDDL)
- World State
  - ☹ POMDP specific, unintuitive …→ … templates
- Utility Function
  - ☹ Implicit, F(world state) … → utility elicitation
- Control Guidance
  - ☹ Options? HAMs? MAX-Q? Basis functions? Constraints?

Related Work

- PPDDL, RDDL
- Alisp [Andre *et al*, 2002]
- A^2BL [Simpkins *et al*, 2008]
- Adaptive Programs [Pinto *et al*, 2010]

Utility / Control Advice

- Users specify utility *procedurally!*
  - Utility is implicit in their control program
  - Eg, "majority vote of three people"
  - *Vs* "83% probability threshold"
  - Let alone utility curve

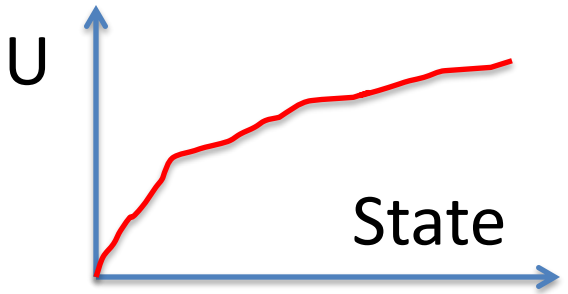


- Procedural behavior also good for roll-out policy

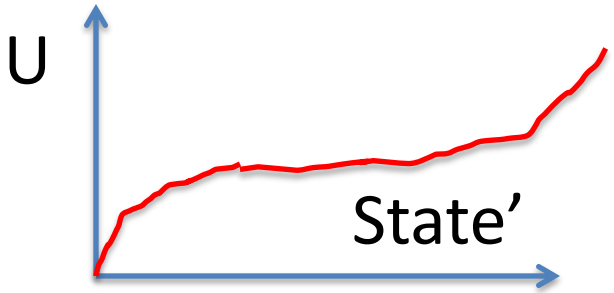
How Compose Utility?

- Want to encapsulate sub-behaviors

(define-behavior X …
    (do A …
    (choose-between B or C…
    (do iterative-improvement …)



Transformed utility fn
For iterative improvement



POAPS Primitives

A *primitive* is a ten-tuple $\langle \mathcal{D}, \mathcal{R}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{I}, \mathcal{C}, \mathcal{D}_U, \mathcal{R}_U, \mathcal{F} \rangle$, where:

- $\mathcal{D} = \mathcal{D}^1 \times \ldots \times \mathcal{D}^n$ is a set of *domain states.*
- $\mathcal{R}$ is a set of *range states.*
- Ω is a set of *observations.*
- $\mathcal{T} : \mathcal{D} \times \mathcal{R} \to [0, 1]$ is a *transition function.*
- $\mathcal{O} : \mathcal{R} \times \Omega \to [0, 1]$ is an *observation function.*
- $\mathcal{I}$ is an n-dimensional indicator vector indicating which of the $\mathcal{D}^i$ are observable.
- $\mathcal{C} : \mathcal{D} \to \mathbb{R}^+$ is a *cost function.*
- $\mathcal{D}_U = \mathcal{D}_U^1 \times \ldots \times \mathcal{D}_U^n$ is a set of *user domain states.*
- $\mathcal{R}_U$ is a set of *user range states.*
- $\mathcal{F} : \mathcal{D}_U \to \mathcal{R}_U$ is a *user function.*

Model of the function

Some Function

Primitive: *c-imp*

- $\mathcal{F}(\alpha \in \mathcal{D}_U) = \text{calltoAPI}(\alpha)$
- $\mathcal{D}_U, \mathcal{R}_U$ is the set of all artifacts α.

- $\mathcal{D} = \mathcal{R} = [0, 1]$
- $\mathcal{T}(q \in \mathcal{D}, q' \in \mathcal{R}) = P(q'|q)$
- $\mathcal{C} = \$0.05$
- No observations/observation function
- $\mathcal{I} = (0)$

POAPS LANGUAGE

```
(define (improve text)
    (choose
        (improve (c-imp text))
        text)))
```

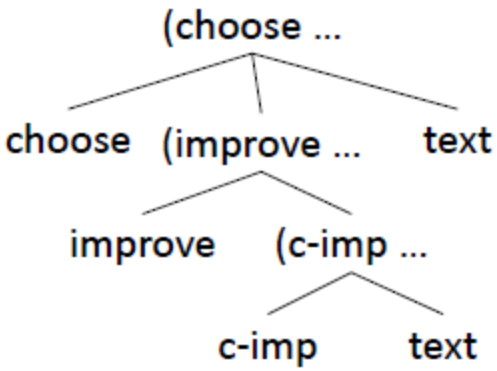
Call by *Poaps Value* Semantics

Compile into HAM

Step 1: Define a set of states *S(p)*

```
(define (improve text)
    (choose
        (improve (c-imp text))
        text)))
```

A state variable for every argument

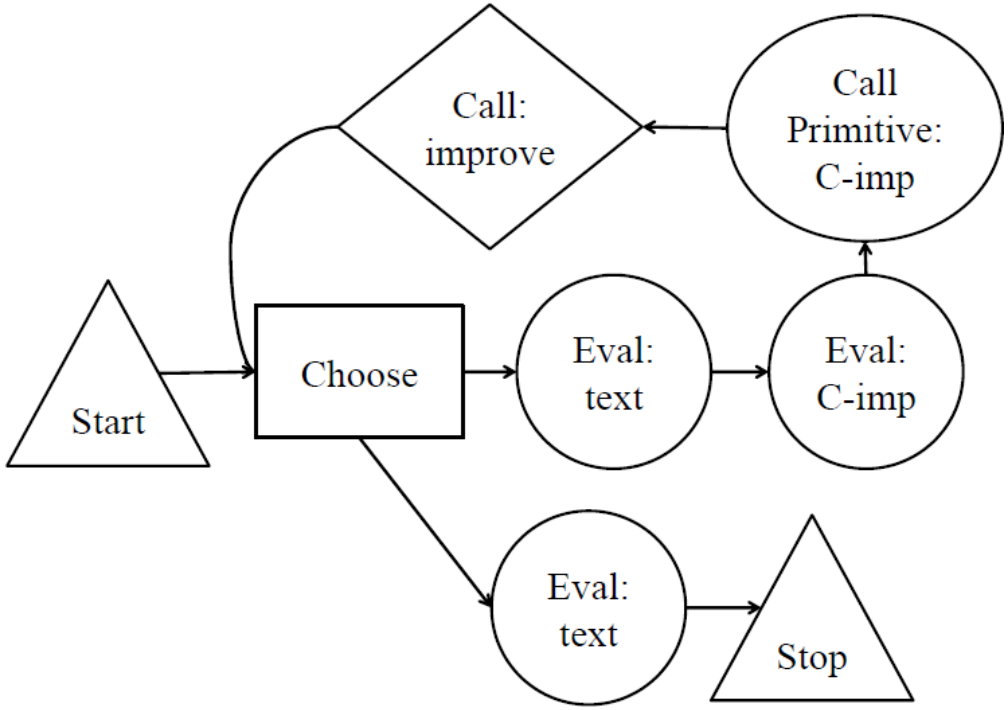


A state variable for every sub-expression

Not Shown: State variables for called functions (Recursive Definition)

Step 2: Construct a HAM

```
(define (improve text)
    (choose
        (improve (c-imp text))
        text)))
```



Step 3: Merge

- Final State Space: *S(p)* + States of HAM
- Actions – Given by HAM
- Transitions – Ensure “Call by *Poaps value* semantics”
- Observations – given by primitives
- Costs – given by primitives

Conclusion

- How bring RL to the masses?
- Compose dynamical systems?
- Specify & compose utility functions?
- RL algorithms?