Efficient Reinforcement Learning with Hierarchies of Machines by Leveraging Internal Transitions

Aijun Bai*
UC Berkeley/Microsoft Research

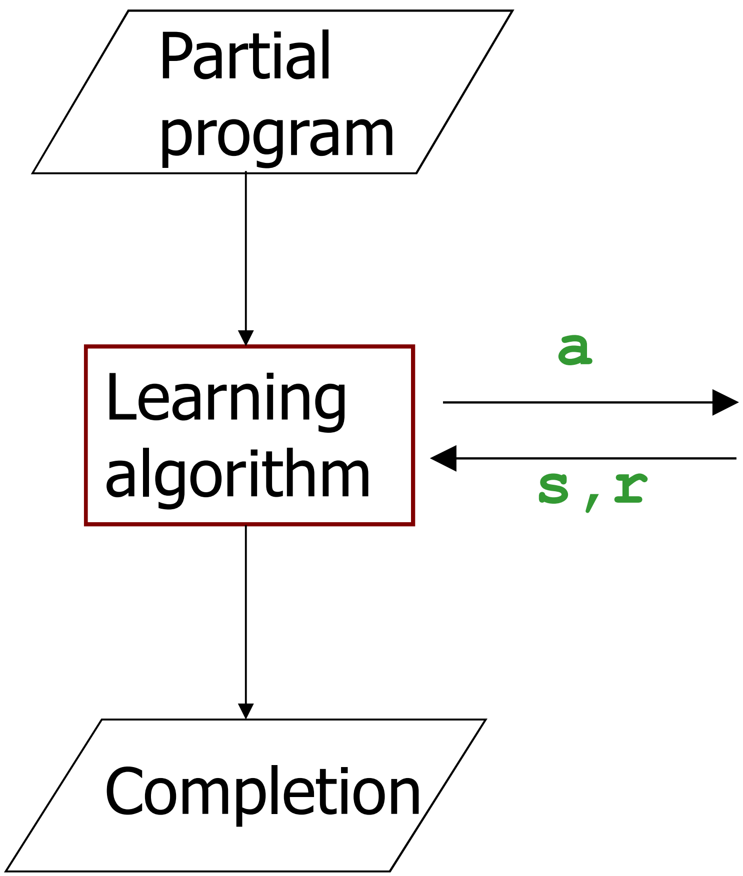
Stuart Russell
UC Berkeley

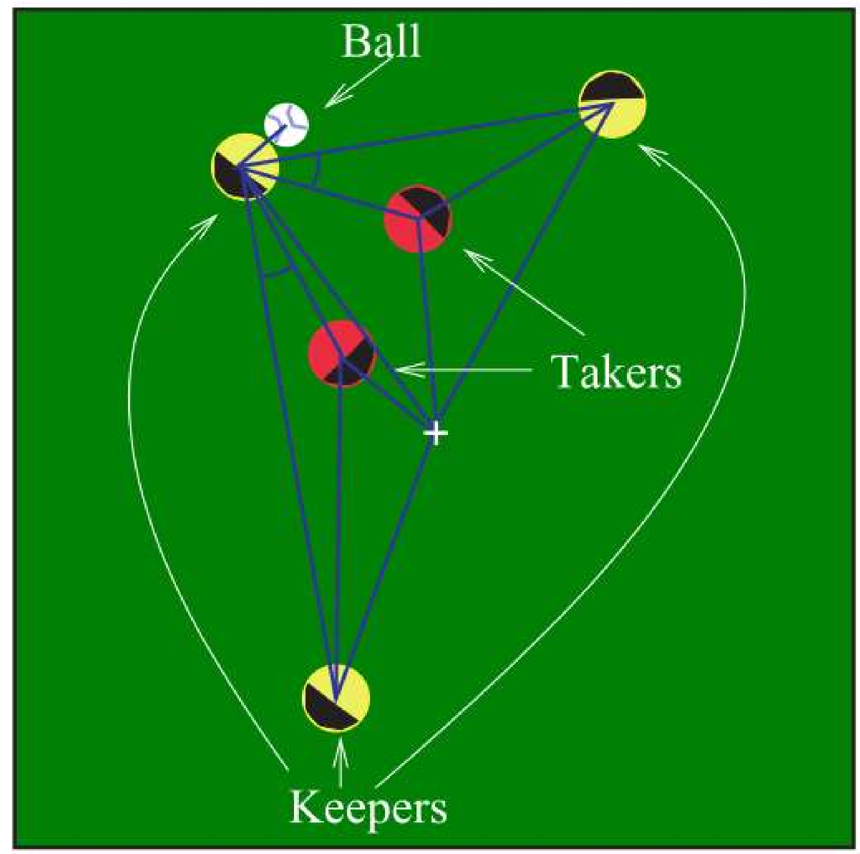
Outline

- Hierarchical RL with partial programs
- Deterministic internal transitions
- Results

Hierarchical RL with partial programs

[Parr & Russell, NIPS 97; Andre & Russell, NIPS 00, AAAI 02; Marthi et al, IJCAI 05]





Hierarchically optimal

for all terminating programs

Partial Program – an Example

repeat forever

Choose({a1,a2,…})

Partial Program – an Example

Navigate(destination)

    while $\neg$At(destination,CurrentState())

        Choose({N,S,E,W})

Concurrent Partial Programs

```
Top()
    for each p in Effectors()
        PlayKeep(p)

PlayKeep(p)
    s ← CurrentState()
    while ¬Terminal(s)
        if BallKickable(s) then Choose({Pass(),Hold()})
        else if FastestToBall(s) then Intercept()
        else Choose(Stay(),Move())

Pass()
    KickTo(Choose(Effectors()\{self}),Choose({slow,fast})
…
```

Technical development

- Decisions based on *internal state*
  - Joint state ω = [s,m] environment state + program state (cf. [Russell & Wefald 1989])
- MDP + partial program = SMDP over choice states in {ω}, learn Q(ω,c) for choices c
- Additive *decomposition* of value functions
  - by *subroutine structure* [Dietterich 00, Andre & Russell 02] Q is a sum of sub-Q functions per subroutine
  - across *concurrent threads* [Russell & Zimdars 03] Q is a sum of sub-Q functions per thread, with decomposed reward signal

Internal Transitions

- Transitions between choice points with no physical action intervening
- Internal transitions take no (real) time and have zero reward
- Internal transitions are deterministic

```
Top()
    for each p in Effectors()
        PlayKeep(p)

PlayKeep(p)
    s ← CurrentState()
    while ¬Terminal(s)
            if BallKickable(s) then Choose({Pass(),Hold()})
            else if FastestToBall(s) then Intercept()
            else Choose(Stay(),Move())

Pass()
    KickTo(Choose(Effectors()\{self}),Choose({slow,fast})
...
```

Idea 1

- Use internal transitions to shortcircuit the computations of Q values recursively if applicable
  - If (s, m, c) -> (s, m’) is an internal transition
  - Then, $Q(s, m, c) = V(s, m') = \max_{c'} Q(s, m', c')$
- Cache internal transitions as **<s, m, c, m’>** tuples
- No need for Q-learning on these

Idea 2

- Identify weakest precondition P(s) for this internal transition to occur (cf EBL, chunking)
- Cache internal transitions as **<P, m, c, m’>** tuples
- Cache size independent of |S|, roughly proportional to size of partial program call graph

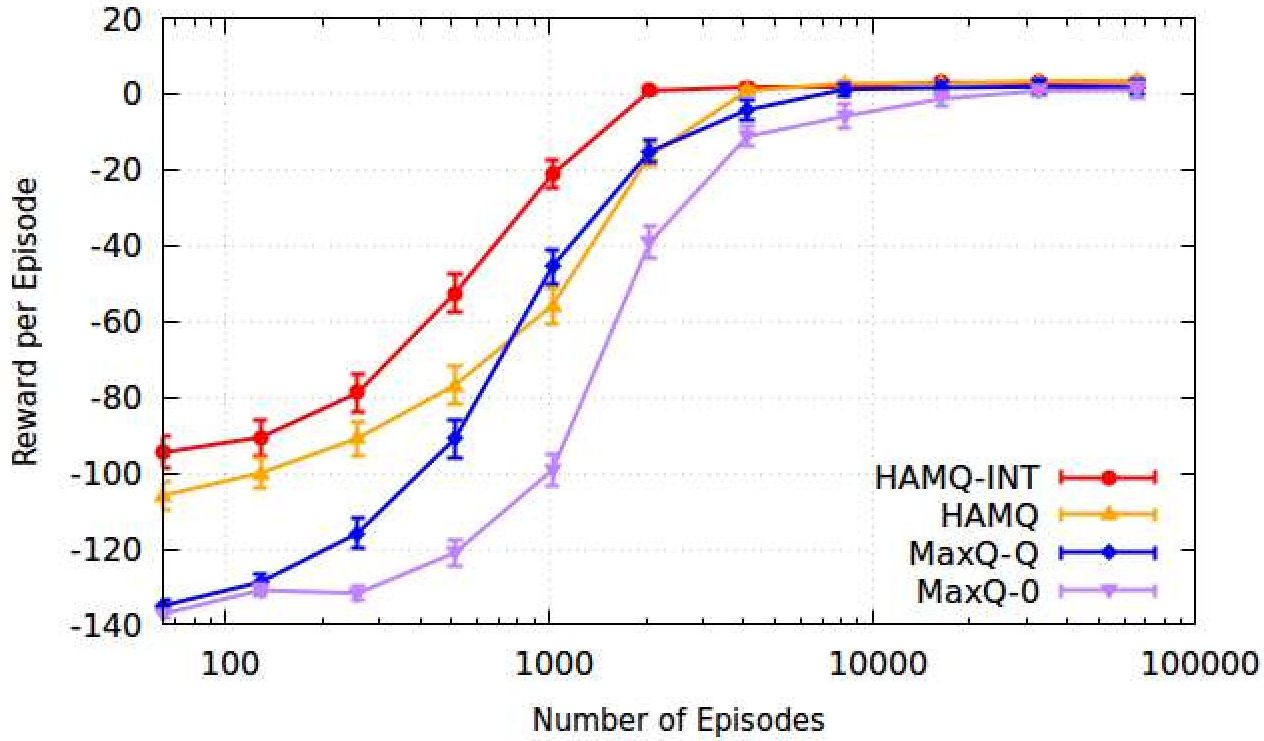
The HAMQ-INT Algorithm

- Track the set of predicates since last choice point
- Save an abstracted rule of internal transition if qualified (τ = 0) in a dictionary ρ
- Use the saved rules to shortcircuit the computation of Q values recursively whenever possible

```
QUpdate(s' : state, z' : stack, r : reward,
  t' : current time, P : evaluated predicates):
if t' = t then
  ρ[P, P(s), z, c] ← z'
else
  QTable(s, z, c) ← (1 − α) QTable(s, z, c)
    +α(r + γ^(t'−t) max_c' Q(s', z', c') )
(t, s, z) ← (t', s', z')

Q(s : state, z : stack, c : choice):
if ∃P s.t. ⟨P, P(s), z, c⟩ ∈ ρ.Keys() then
  q ← −∞
  z' ← ρ[P, P(s), z, c]
  for c' ∈ μ(z.Top()) do
    q ← max(q, Q(s, z', c') )
  return q
else
  return QTable(s, z, c)
```

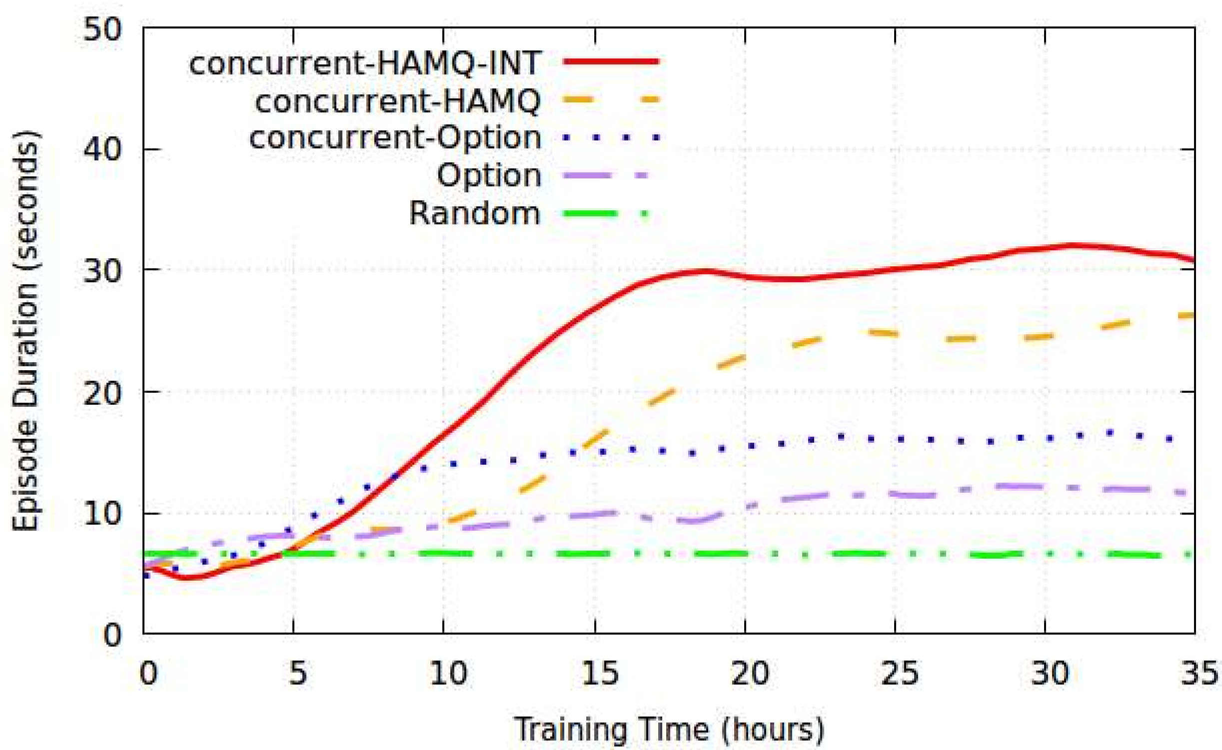
Experimental Result on Taxi



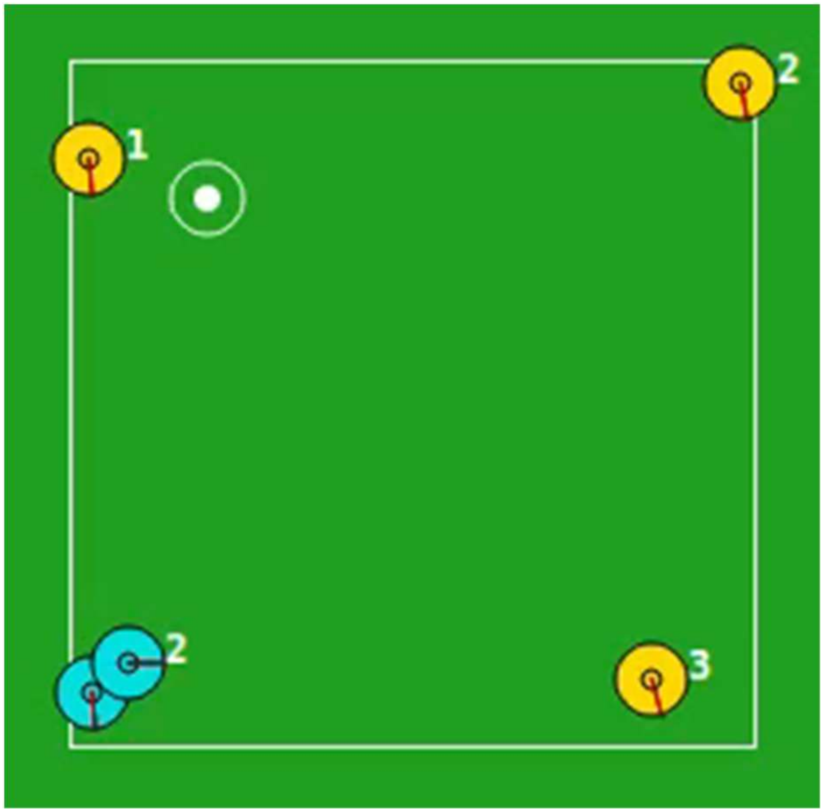
3 vs 2 Keepaway Comparisons

- Option (Stone, 2005):
  - Each keeper learning separately
  - Learn a policy over Hold() and Pass(k, v) if ball kickable; otherwise, follow a fixed policy
    - Intercept() if fastest to the ball; otherwise, GetOpen()
    - GetOpen() is manually programmed for Option
- Concurrent-Option:
  - Concurrent version of Option
    - One global Q function is learnt
- Random: randomized version of Option
- Concurrent-HAMQ
  - Learn its own version of GetOpen() by calling Stay() and Move(d, v)
- Concurrent-HAMQ-INT

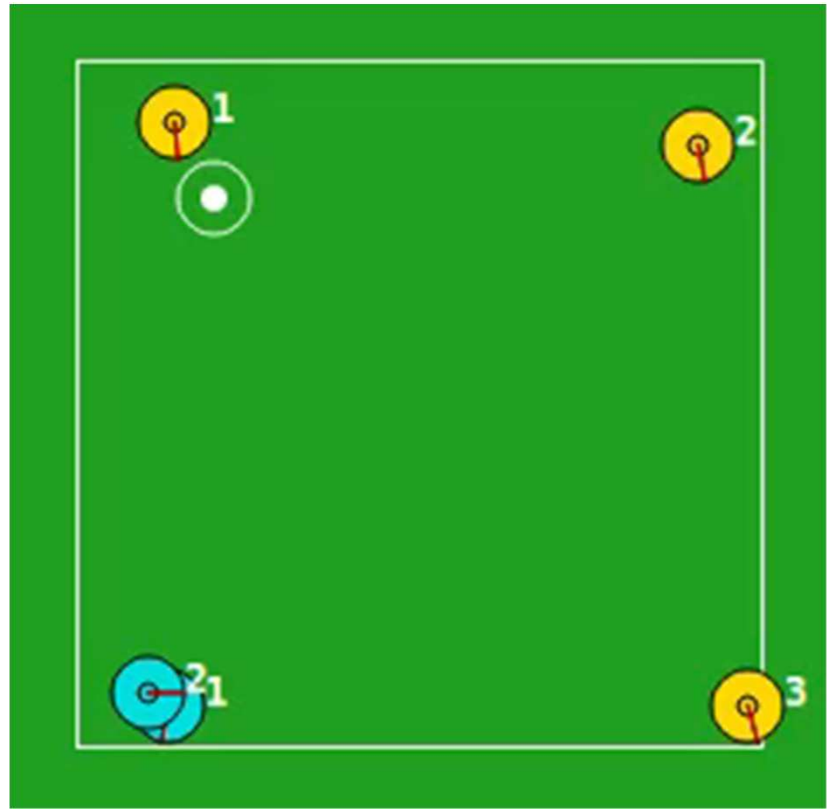
Experimental Result on Keepaway



Before and After



Initial policy



Converged policy

Summary

- HAMQ-INT algorithm
  - Automatically discovers internal transitions
  - Takes advantage of internal transitions for efficient learning
  - Outperforms the state of the art significantly on Taxi and RoboCup Keepaway
- Future work
  - Scale up to full RoboCup task
  - More general integration of model-based and model-free reinforcement learning
  - More flexible forms of partial program (e.g., temporal logic)