Reinforcement Learning with Partial Programs

Stuart Russell
Computer Science Division, UC Berkeley

Joint work with Ron Parr, David Andre, Bhaskara Marthi, Andy Zimdars, David Latham, Carlos Guestrin

1

Scaling up

- Human life: one trillion actions
- World: gazillions of state variables

2

Structured behavior

Behavior is usually very structured and deeply nested:

- Moving my tongue
- Shaping this syllable
- Saying this word
- Saying this sentence
- Making a point about nesting
- Explaining structured behavior
- Giving this talk

3

Structured behavior

Behavior is usually very structured and deeply nested:

- Moving my tongue
- Shaping this syllable
- Saying this word
- Saying this sentence
- Making a point about nesting
- Explaining structured behavior
- Giving this talk

Modularity: Choice of tongue motions is independent of almost all state variables, given the choice of word.

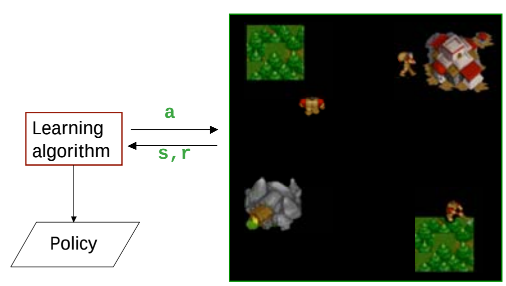
4

Running example

- Peasants can move, pickup and dropoff
- Penalty for collision
- Cost-of-living each step
- Reward for dropping off resources
- Goal : gather 10 gold + 10 wood
- $(3L)^n$++ states s
- 7^n primitive actions a
- (Warren's 4^{th} quadrant)

5

Reinforcement Learning



6

Q-functions

s	a	Q(s,a)
...	...	...
Peas1Loc=(2,3),Peas2Loc=(4,2),Gold=8,Wood=3	(East,Pickup)	7.5
Peas1Loc=(4,4), Peas2Loc=(6,3), Gold=4, Wood=7	(West, North)	-12
Peas1Loc=(5,1), Peas2Loc=(1,2), Gold=8, Wood=3	(South, West)	1.9
...	...	...

Fragment of example Q-function

- Can represent policies using a Q-function
- $Q^{\pi}(s,a)$ = "Expected total reward if I do action a in environment state s and follow policy π thereafter"
- Q-learning provides a model-free solution method

7

Temporal abstraction in RL

- Define temporally extended actions, e.g., "get gold", "get wood", "attack unit x" etc.
- Set up a decision process with choice states and extended choice-free actions
- Resulting decision problem is semi-Markov (Forestier & Varaiya, 1978)
- Temporal abstraction in RL (HAMs (Parr & Russell); Options (Sutton & Precup); MAXQ (Dietterich))

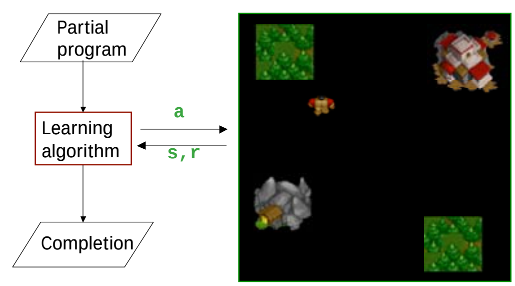
8

Partial programs

- Choices only in some states
  => prior constraints on behavior
  => partially specified programs
- Partial programming language
  = programming language + free choices
- ALisp (Andre & Russell, 2002)
  Concurrent ALisp (Marthi et al., 2005)
- Loose analogy to Bayes Nets
  - Domain experts supply structure (partial programs)
  - Learning fills in numerical details
  - Factored representation of Q-function => faster learning (Dietterich, 2000)

9

RL and partial programs



10

Single-threaded Alisp program

```
(defun top ()
  (loop do
    (choose ‘top-choice
      (gather-wood)
      (gather-gold))))

(defun gather-wood ()
  (with-choice ‘forest-choice
      (dest *forest-list*)
    (nav dest)
    (action ‘get-wood)
    (nav *base-loc*)
    (action ‘dropoff)))

(defun gather-gold ()
  (with-choice ‘mine-choice
      (dest *goldmine-list*)
    (nav dest))
    (action ‘get-gold)
    (nav *base-loc*))
    (action ‘dropoff)))

(defun nav (dest)
  (until (= (pos (get-state))
            dest)
    (with-choice ‘nav-choice
        (move ‘(N S E W NOOP))
      (action move))))
```

- Program state θ includes
  - Program counter
  - Call stack
  - Global variables

11

Q-functions

ω	u	Q(ω,u)
...	...	...
At nav-choice, Pos=(2,3), Dest=(6,5)	North	15
At resource-choice, Gold=7, Wood=3	Gather-wood	-42
...	...	...

Example Q-function

- Represent completions using Q-function
- Joint state $\omega = [s,\theta]$ env state + program state
- MDP + partial program = SMDP over $\{\omega\}$
- $Q^{\pi}(\omega,u)$ = "Expected total reward if I make choice u in ω and follow completion π thereafter"
- Modified Q-learning [AR 02] finds optimal completion

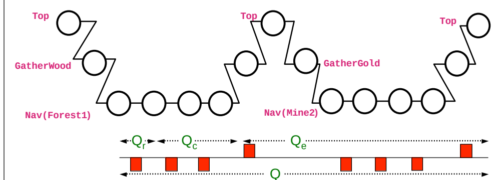
12

Internal state

- Availability of internal state (e.g., goal stack) can greatly simplify value functions and policies
- E.g., while navigating to location (x,y), moving towards (x,y) is a good idea
- Natural local shaping potential (distance from destination) impossible to express in external terms

13

Temporal Q-decomposition



- Temporal decomposition $Q = Q_r + Q_c + Q_e$ where
  - $Q_r(\omega,u)$ = reward while doing u (may be many steps)
  - $Q_c(\omega,u)$ = reward in current subroutine after doing u
  - $Q_e(\omega,u)$ = reward after current subroutine

14

State abstraction

- Temporal decomposition => state abstraction
  E.g., while navigating, Q_c independent of gold reserves
- In general, local Q-components can depend on few variables => fast learning

15

Handling multiple effectors

Multithreaded agent programs

- Threads = tasks
- Each effector assigned to a thread
- Threads can be created/destroyed
- Effectors can be reassigned
- Effectors can be created/destroyed

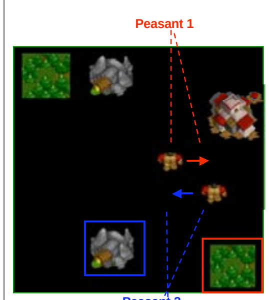
16

An example Single-threaded Alisp program / An example Concurrent ALisp program

```
(defun top ()
  (loop do
    (until (my-effectors)
     (choose 'top-choice
      (gather-gold)
      (gather-wood)))
    (setf peas
      (first (my-effectors))
    (choose 'top-choice
      (spawn gather-wood peas)
      (spawn gather-gold peas))))

(defun gather-gold ()
  (with-choice 'mine-choice
      (dest *goldmine-list*)
    (nav dest)
    (action 'get-gold)
    (nav *base-loc*)
    (action 'dropoff)))

(defun gather-wood ()
  (with-choice 'forest-choice
      (dest *forest-list*)
    (nav dest)
    (action 'get-wood)
    (nav *base-loc*)
    (action 'dropoff)))

(defun nav (dest)
  (until (= (my-pos) dest)
    (with-choice 'nav-choice
        (move '(N S E W NOOP))
      (action move))))
```

17

Concurrent Alisp semantics




```
(defun gather-wood ()
  (setf dest (choose *forests*))
  (nav dest)
  (action 'get-wood)
  (nav *home-base-loc*)
  (action 'put-wood))

(defun nav (Wood1)
  (loop until (at-pos Wood1)
    (setf d (choose '(N S E W R)))
    (do d)
  ))
```

Thread 1 Running / Waiting for joint choice / Making joint action

```
(defun gather-gold ()
  (setf dest (choose *goldmines*))
  (nav dest)
  (action 'get-gold)
  (nav *home-base-loc*)
  (action 'put-gold))

(defun nav (Gold2)
  (loop until (at-pos Gold2)
    (setf d (choose '(N S E W R)))
    (do d)
  ))
```

Thread 2 Paused / Waiting for joint choice / Making joint action

Environment timestep 26 / 27

18

Concurrent Alisp semantics

- Threads execute independently until they hit a choice or action
- Wait until all threads are at a choice or action
  - If all effectors have been assigned an action, do that joint action in environment
  - Otherwise, make joint choice

19

Q-functions

ω	u	Q(ω,u)
...	...	...
Peas1 at NavChoice, Peas2 at DropoffGold, Peas3 at ForestChoice, Pos1=(2,3), Pos3=(7,4), Gold=12, Wood=14	(Peas1:East, Peas3:Forest2)	15.7
...	...	...

Example Q-function

- To complete partial program, at each choice state ω, need to specify choices for all choosing threads
- So Q(ω,u) as before, except u is a joint choice
- Suitable SMDP Q-learning gives optimal completion

20

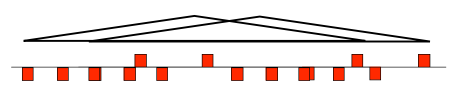
Making joint choices at runtime

- At state ω, want to choose $\mathrm{argmax}_u\, Q(\omega,u)$
  - # joint choices exponential in # choosing threads
- See Parr, R. (2006) "Shameless plug." Proc. NSF Workshop on ADP, Cocoyoc, Mexico.

21

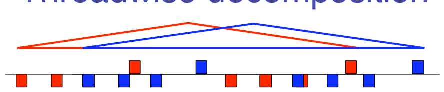
Problems with concurrent activities

- Temporal decomposition of Q-function lost
- No credit assignment among threads
  - Suppose peasant 1 drops off some gold at base, while peasant 2 wanders aimlessly
  - Peasant 2 thinks he's done very well!!
  - Significantly slows learning as number of peasants increases



22

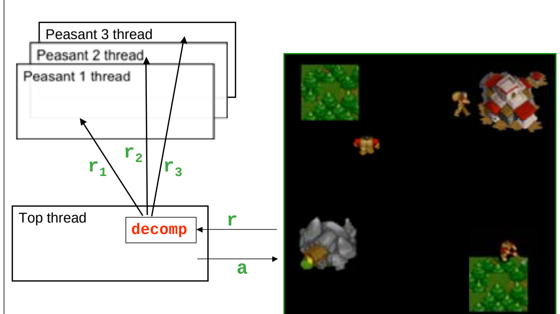
Threadwise decomposition



- Idea : decompose reward among threads (Russell+Zimdars, 2003)
- E.g., rewards for thread j only when peasant j drops off resources or collides with other peasants
- $Q_j^\pi(\omega,u)$ = "Expected total reward received by thread j if we make joint choice u and then do π"
- Threadwise Q-decomposition $Q = Q_1 + \ldots Q_n$
- Recursively distributed SARSA => global optimality

23

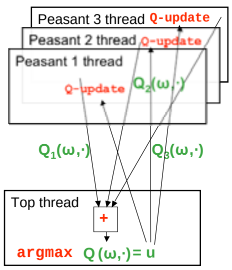
Learning threadwise decomposition



Action state

24

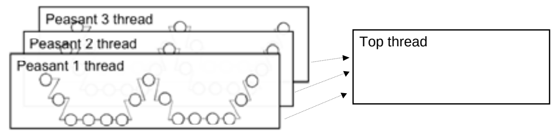
Learning threadwise decomposition



Choice state ω

25

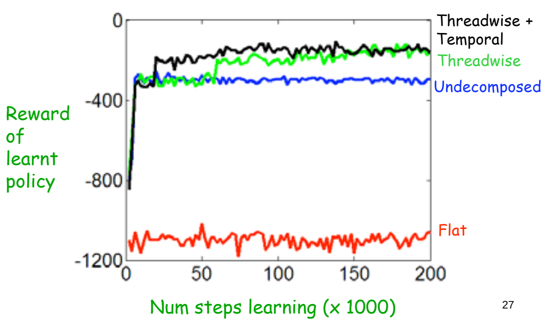
Threadwise and temporal decomposition



- $Q_j = Q_{j,r} + Q_{j,c} + Q_{j,e}$ where
  - $Q_{j,r}$ (ω,u) = Expected reward gained by thread j while doing u
  - $Q_{j,c}$ (ω,u) = Expected reward gained by thread j after u but before leaving current subroutine
  - $Q_{j,e}$ (ω,u) = Expected reward gained by thread j after current subroutine

26

Resource gathering with 15 peasants



27

Main points to take away

- Structure in behavior seems essential for scaling up
- Partial programs
  - Provide natural structural constraints on policies
  - Decompose complex value functions into simple components (based on conditional independence structure of transition model and reward function)
  - Include internal state (e.g., “goals”) that further simplifies value functions, shaping rewards
- Concurrency
  - Simplifies description of multieffector behavior
  - Messes up temporal decomposition and credit assignment (but threadwise reward decomposition restores it)

28

Current directions

- http://www.cs.berkeley.edu/~bhaskara/alisp/
- Model-based learning and lookahead
- Temporal logic as a partial programming language
- Partial observability ([s,θ] is just [θ])
- Complex motor control tasks
- Metalevel RL: choice of computation steps
- Transfer of learned subroutines to new tasks
- Eliminating Q_e by recursive construction [UAI06]
- Learning new hierarchical structure

29

Current directions

- http://www.cs.berkeley.edu/~bhaskara/alisp/
- Model-based learning and lookahead
- Temporal logic as a partial programming language
- Partial observability ([s,θ] is just [θ])
- Complex motor control tasks
- Metalevel RL: choice of computation steps
- Transfer of learned subroutines to new tasks
- Eliminating Q_e by recursive construction [UAI06]
- Learning new hierarchical structure
- Yael’s proposed NIPS workshop

30