State abstractions

The Value of Abstraction Ho, M., Abel, D., Griffiths, T., and Littman, M.
State + temporal abstractions

(a)

(b)

The Value of Abstraction Ho, M., Abel, D., Griffiths, T., and Littman, M.
Hierarchical Reinforcement Learning

George Konidaris

gdk@cs.brown.edu
Why Hierarchies?
Skill Hierarchies

Hierarchical RL: base hierarchical control on skills.
- Component of behavior.
- Performs continuous, low-level control.
- Can treat as discrete action.

Behavior is modular and compositional.

Skills are like subroutines
def abs(x):
    if(x > 0):
        return x
    else:
        return -x

Development  Specialization  Simplification

[Wilkes, Wheeler and Gill, 1951]
Forms of Abstraction

\[ \langle \bar{S}, \bar{A}, R, T, \gamma \rangle \]

state abstraction

action abstraction

\[ \langle S, A, R, T, \gamma \rangle \]
The Options Framework
Options

**Options Framework**: theoretical basis for skill acquisition, learning and planning using higher-level actions (options).

RL typically solves a *single* problem *monolithically*.

Action abstraction:

- Create and use higher-level macro-actions.
- Problem now contains subproblems.
- Each subproblem is also an RL problem.

[Sutton, Precup, and Singh, 1999]
Hierarchical RL
Hierarchical RL

Skill

Problem
An option is one formal model of a skill.

An option $\omega$ is a policy unit:
- Initiation set $I_\omega : S \rightarrow \{0, 1\}$
- Termination condition $\beta_\omega : S \rightarrow [0, 1]$  
- Option policy $\pi_\omega : S \times A \rightarrow [0, 1]$
Actions as Options

A primitive action $a$ can be represented by an option:

- $I_a(s) = 1, \forall s \in S$
- $\beta_a(s) = 1, \forall s \in S$
- $\pi_a(s, b) = \begin{cases} 1 & a = b \\ 0 & \text{otherwise} \end{cases}$

A primitive action can be executed anywhere, lasts exactly one time step, and always chooses action $a$. 
Questions

Given an MDP:

\[(S, A, R, T, \gamma)\]

... let’s replace A with a set of options O (some of which may be primitive actions).

- How do we characterize the resulting problem?
- How do we plan using options?
- How do we learn using options?
- How do we characterize the resulting policies?
The resulting problem is a Semi-(Markov Decision Process). This consists of:

- $S$: Set of states
- $O$: Set of options
- $P(s', t|o, s)$: Transition model
- $R(s', s, t)$: Reward function
- $\gamma$: Discount factor (per step)

In this case:

- All times are natural numbers.
- “Semi” here means transitions can last $t$ timesteps.
- Transition and reward function involve time taken for option to execute.
Planning?

Easy

\[ Q^{\pi}(s, o) = \mathbb{E}_{t,s'}[R(s', s, t)] + \mathbb{E}_{t,s'}[\gamma^t \pi(s', o')Q^{\pi}(s', o')] \]

where

\[ \mathbb{E}_{t,s'}[R(s', s, t)] = \sum_{t,s'} P(s', t|o, s)R(s', s, t) \]

\[ \mathbb{E}_{t,s'}[\gamma^t \pi(s', o')Q^{\pi}(s', o')] = \sum_{t,s'} P(s', t|o, s)\gamma^t \pi(s', o')Q^{\pi}(s', o') \]

All things flow from Bellman.
Learning and Planning

\[ Q^\pi(s, o) = \mathbb{E}_{t,s'}[R(s', s, t)] + \mathbb{E}_{t,s'}[\gamma^t \pi(s', o')Q^\pi(s', o')] \]

For learning:

• Stochastic samples.
• Use SMDP Bellman equation.

For planning:

• Synchronous Value Iteration via SMDP Bellman eqn
Example

(Sutton, Precup and Singh, AIJ 1999)
Example

(Sutton, Precup and Singh, AIJ 1999)
Example

Primitive and hallway options $O=AUH$

(Sutton, Precup and Singh, AIJ 1999)
Final note: policies.

A policy over an MDP with primitive actions is a Markov policy:

$$\pi : S \times A \rightarrow [0, 1]$$

A policy over an MDP with options could also be Markov:

$$\pi : S \times O \rightarrow [0, 1]$$

... but this could imply a policy in the original MDP that is not, because the probability of taking an action at a state depends on the option currently running.
So

A Markov policy for an SMDP may result in a semi-Markov policy for the underlying MDP.

(Even if the options are Markov options!)

Here, semi-Markov means that the probability of taking a primitive action at each step depends on more than the current state.
What are Options For?

Lots of things!

A few salient points:
- Rewiring.
- Transfer.
- Skill-Specific State Abstractions.
Rewiring

Adding an option changes the connectivity of the MDP. This affects:

- Learning and planning.
- Exploration.
- State-visit distribution.
- Diameter of problem.

(Sutton, Precup and Singh, AIJ 1999)
Transfer

Use experience gained while solving one problem to improve performance in another.

Skill transfer:

- Use options as mechanism for transfer.
- Transfer components of solution.
- Can drastically improve performance
- ... even if it takes a lot of effort to learn them.

General principle: subtasks recur.
Transfer

Tasks drawn from parametrized family.
- Common features present.
- Options defined using only common features.

(a) Learning curves for agents with problem-space options.
(b) Learning curves for agents with agent-space options, with varying numbers of training experiences.

[Konidaris and Barto, IJCAI 2007]
Skill-Specific Abstractions

Options provide opportunities for abstraction

- Split high-dimensional problem into subproblems ...
- ... such that each one supports a solution using an abstraction.

Working hypothesis: *behavior is modular and compositional* and *piecewise low-dimensional*.
Skill Acquisition
Skill Discovery

Where do skills come from?

Research goal: discover options autonomously, through interaction with an environment.

- Typically subgoal options.
- This means that we must determine $\beta_o$.
- Sometimes also $R_o$.

The question then becomes:
- Which states are good subgoals?
Betweenness Centrality

Consider an MDP as a graph.

- States are vertices.
- Edges indicate possible transition between two states.

Further, let us assume a task distribution over start states and goal pairs:

- $P_T(s, e)$

(Simsek and Barto, 2008)
Betweenness Centrality

We can define the *betweenness centrality* of a vertex (state) as:

\[
\sum_{s,e} \frac{\sigma_{se}(v)}{\sigma_{se}} \omega_{se}
\]

This indicates its probability of being on a shortest path from \(s\) to \(e\); if we define:

- *Shortest path as optimal solution.*
- \(\omega_{se} = P_T(s, e)\)

... then we get something sensible for RL.

(Simsek and Barto, 2008)
Betweenness Centrality

(Simsek and Barto, 2008)
Covering Options

More modern:

- Formulate a specific objective
- Find options with formal link to objective

E.g., finding options to aid with exploration:

- “the difficulty of discovering a distant rewarding state in an MDP is bounded by the expected cover time of a random walk over the graph induced by the MDP’s transition dynamics”

- Therefore, find options to minimize cover time.
- This is NP-Hard.
- Bounded-suboptimal approximation algorithm.

[Jinnai et al., 2019]
What About Continuous Domains?
Skill Chaining

Executing one skill should either:

- Solve the problem.
- Let you execute another skill that could solve the problem.

Skills should be *chainable*.

[Konidaris and Barto, NIPS 2009]
Skill Chaining
Skill Chaining: Results
Skill Chaining: Results
Skill Chaining: Results
Deep Skill Chaining

[Bagaria and Konidaris, in submission]
Option-Critic (Bacon et al.)

- Use policy gradient to simultaneously learn high-level policy and option policies / termination conditions
- Assume options can be initiated anywhere
- Options are learned based directly on performance, rather than a heuristic!
Feudal Nets (Vezhnevits et. al)

- Manager module sets goals for the worker and receives environmental reward
- Worker module is rewarded for completing goals set by manager
- Learns to set and accomplish goals that are best for optimizing expected return — no heuristics