

Efficient MDP Solutions and Extensions

Reinforcement Learning and
Planning under Uncertainty

ANU COMP6460/4640, Sem 2, 2008

Scott Sanner, Patrik Haslum

NICTA / RSISE

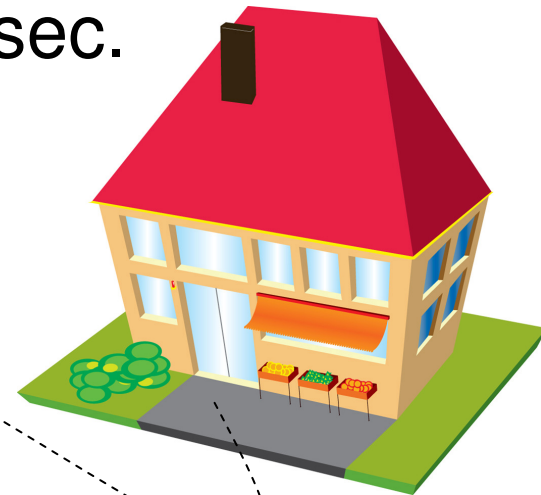
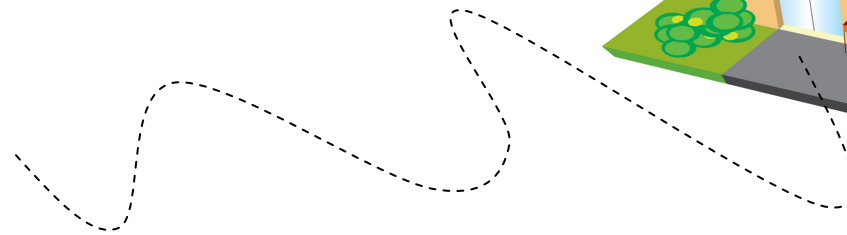
First.Last@nicta.com.au

Reward Shaping

**(Ng, Harada, Russell,
ICML-1999)**

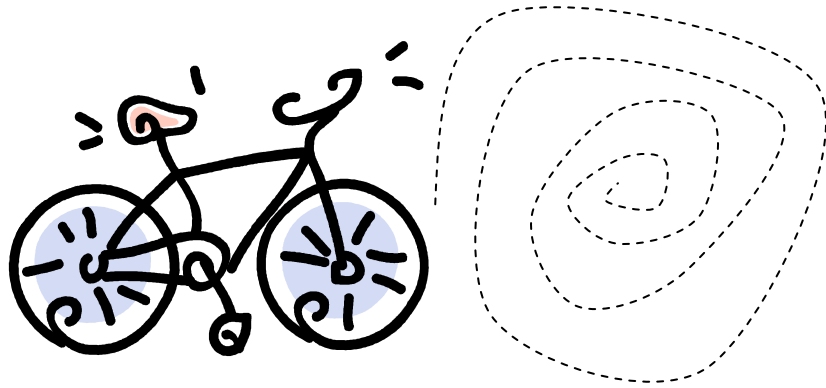
Ride a Bike to a Goal

- To reach goal (only non-zero reward)...
have to first learn to balance on a bike
 - Idea: provide “hints” by shaping reward
 - +1 for balancing a bike for 5 sec.



Problem with Reward Shaping

- Get an indefinitely large reward just for maintaining balance
 - Reward shaping dwarfs goal reward
 - A good policy may bike in circles!

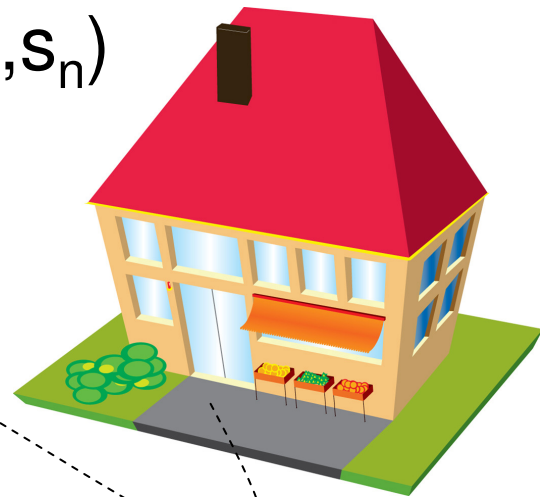


Question

- **What kind of reward shaping ensures same optimal policies in original MDP?**
 - Shaped $R'(s,a,s') = R(s,a,s') + R_s(s,a,s')$
 - Let π^* be optimal for R'
 - What conditions on R_s guarantee π^* is also optimal for R ?
- Hint: make shaping in cycles sum to zero

Solution

- Given: $R'(s,a,s') = R(s,a,s') + R_s(s,a,s')$
- Define: state-based shaping function $\phi(s)$
- Require: $R_s(s,a,s') = \phi(s') - \phi(s)$
 - $R_s(s_1,a_1,s_2) + \dots + R_s(s_{n-1},a_{n-1},s_n) + R_s(s_n,a_n,s_1) = 0$



Reward Shaping Theory

- Condition: $R_s(s,a,s') = \phi(s') - \phi(s)$
- Guarantees:
 - **Sufficiency:** If condition holds, any policy optimal for R' will also be optimal for R (and vice versa)
 - prove by showing Bellman fixed-point optimality equations for R are also satisfied for R' (same $V^* \Rightarrow$ same π^*)
 - **Necessity:** If condition does not hold, there exists some MDP for which any π^* for R' will not be optimal for R
 - proof is more tedious
- So this is the *most general condition* for optimal policy invariance under reward shaping

Policy Constraints and Partial Programs

(Parr, Russell, NIPS-1998)

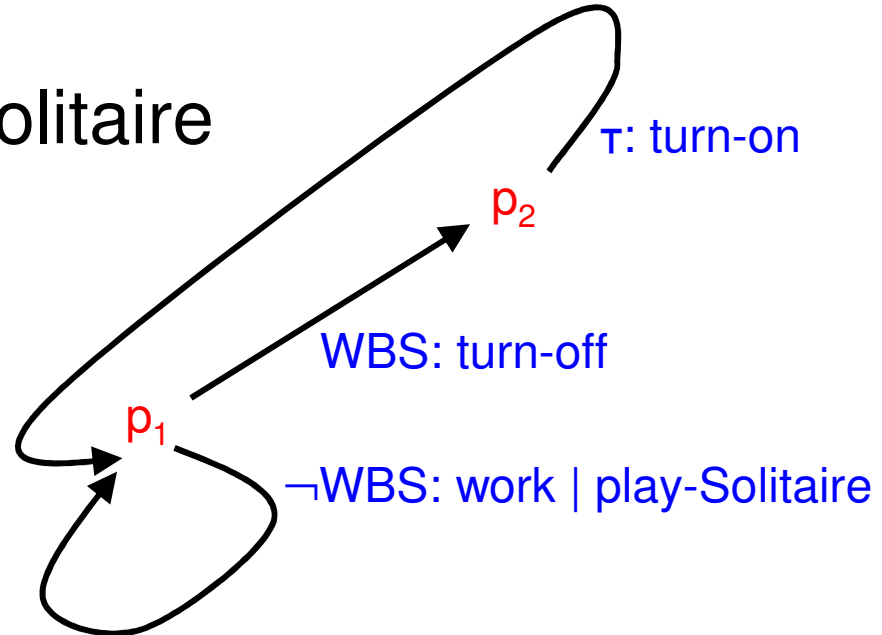
Policy Constraints

- For an MDP, may have general idea of good policy structure
 - e.g., Task A should be finished before task B
- Or a policy must simply obey constraints
 - e.g., Elevator must never reverse if occupant going in current direction
- But: policy only constrains what to do...
 - e.g., Do A or B (but not C)
 - Can optimally fill in decisions of partial policy?

Policy Constraints = Programs = FSAs

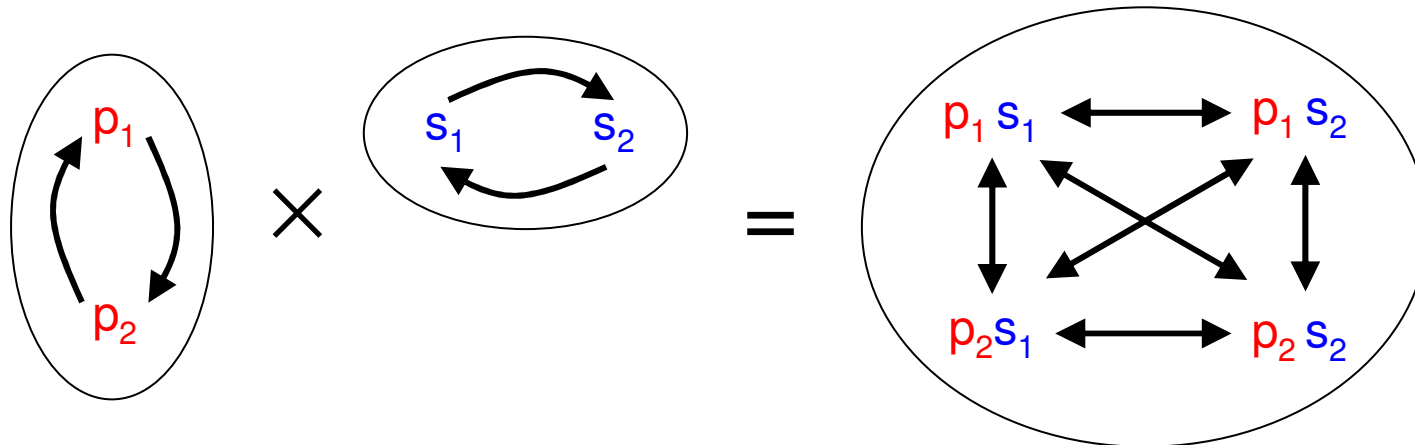
- Partial policy as partial program:
 - **while (true)**
 - if** (Windows-blue-screen [WBS]) **then**
 - do** turn-off, turn-on
 - else**
 - do** work | play-Solitaire

- As a finite state automata (FSA):



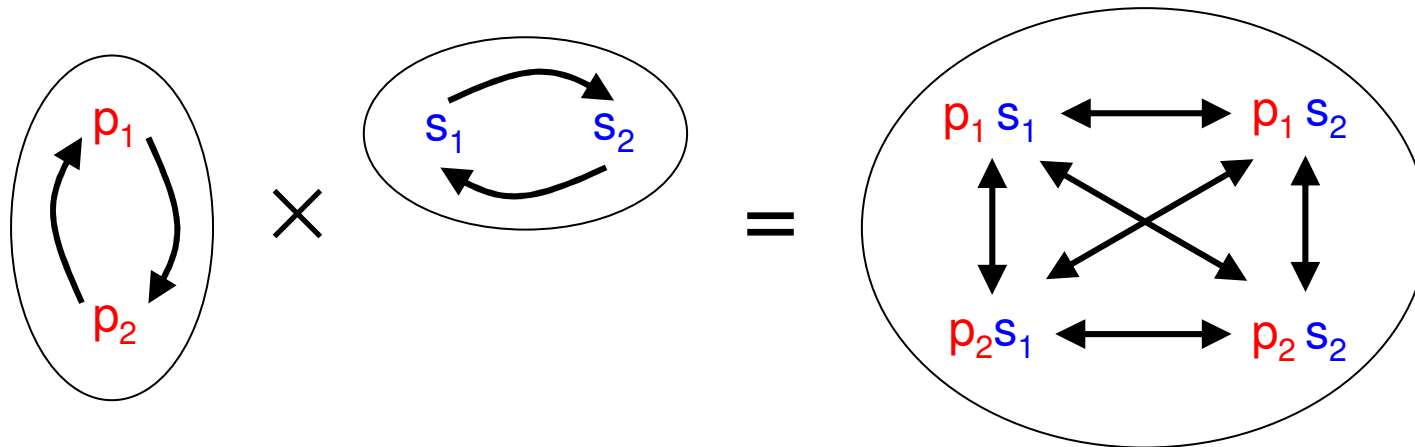
Cross-product MDP

- Have program state and MDP state
 - Take cross-product



- Only actions consistent with policy constraints are legal from any joint state

Solving Cross-product MDP



- Solve cross-product MDP as usual MDP!
 - Just don't consider restricted actions during max
 - Works for model-based or model-free (RL)
- Gives optimal solution w.r.t. policy constraints!
 - Caveat: cross-product MDP may be large

Macro Actions

**(Hauskrecht, Meuleau, Pack
Kaelbling, Dean, Boutilier,
UAI-1998)**

Macro Actions

- Often want to define macro actions
 - Grid world example: **move-to-room4**

I	I	I	I	I	I	I
I	1	I	I	2	I	I
I	I	I	I	I	I	I
I	I	I			E	
I	3	I		4		
I	I	I				
I	I	I	E			

Macro action can start anywhere in states I

Terminates on reaching states E

- Macros take multiple time steps
 - advantage: fewer iterations of value iteration?

Macro Actions

- Macro action definition
 - Can be executed in a set of initial states I
 - Must exit when it reaches a set of exit states $E \subseteq I$
 - I is transitive closure of states reachable from I without touching E
- Problem:
 - macro-actions take ≥ 1 time steps
 - Then how to handle discounting properly?
 - can't treat as 1-step primitive action
- Solution:
 - Could formalize as semi-MDP...

Semi-MDPs

- Semi-MDP
 - Action can take arbitrary duration τ
 - Need to define $P(\tau|s,a)$ in addition to $P(s'|s,a,\tau)$

- Value iteration for Semi-MDPs

$$V^{k+1}(s) = \max_a Q^{k+1}(s, a)$$

$$Q^{k+1}(s, a) = R(s, a) + \sum_{\tau=0}^{\infty} P(\tau|s, a) \gamma^{\tau} \sum_{s'} P(s'|s, a, \tau) V^k(s')$$

- Alternate formalization
 - Macro-action semi-MDPs can be converted to standard MDPs
 - Trick is to fold discounting into transition and reward for macros...

Solving MDPs with Macro Actions

- Macro m is fixed policy π_m defined over I and exiting in $E \subseteq I$
- Solve following linear systems for T_m, R_m

$$\forall s \in I, s' \in E, T_m(s, s') = \\ T(s, \pi_m(s), s') + \gamma \sum_{s''} T(s, \pi_m(s), s'') T_m(s'', s')$$

$$\forall s \in I, R_m(s) = \\ R(s, \pi_m(s)) + \gamma \sum_{s'} T(s, \pi_m(s), s') R_m(s')$$

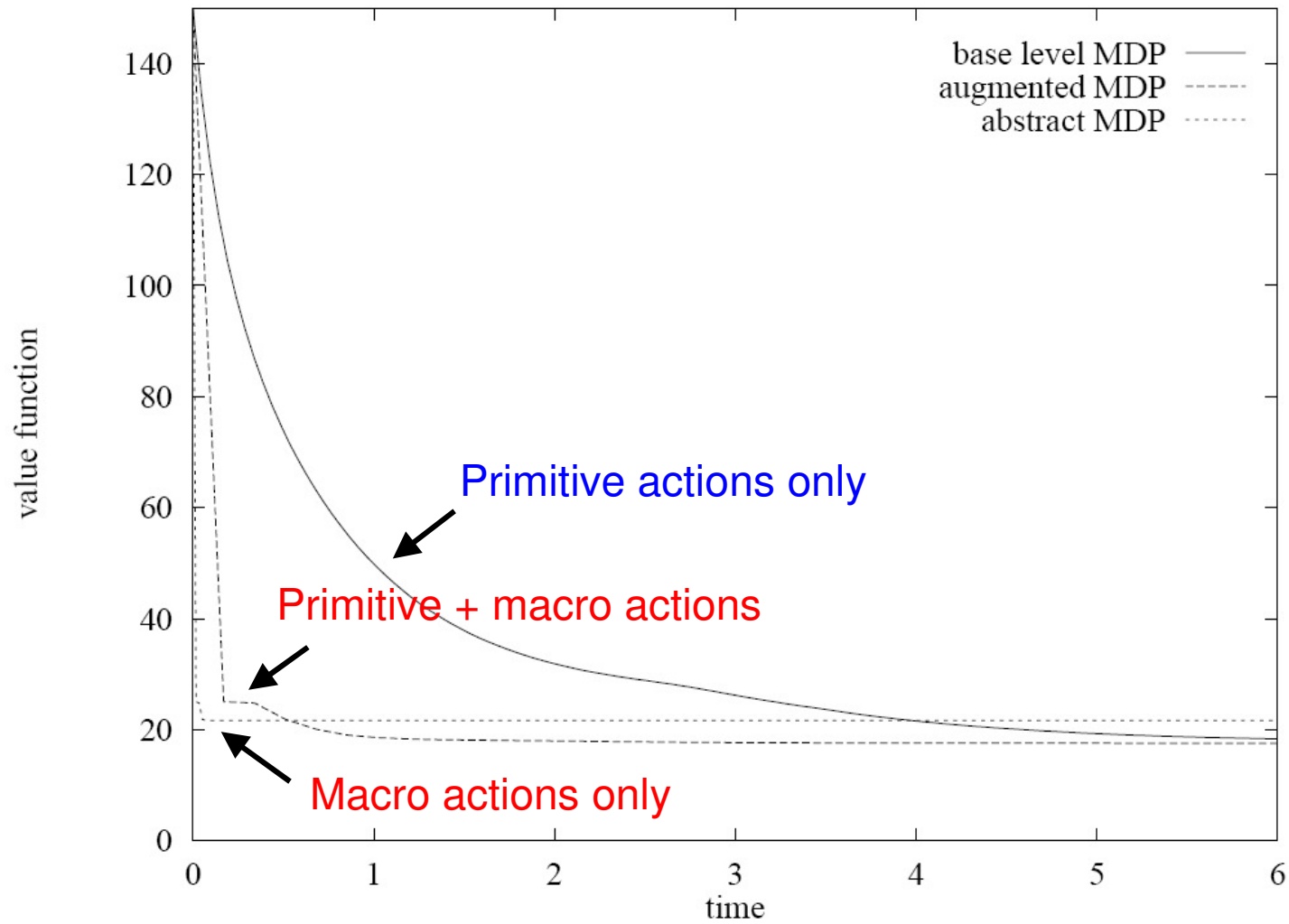
Solving MDPs with Macro Actions

- When solving MDP with macros...
 - Just substitute T_m, R_m whenever $a=m \in M$ (macro set)
 - For non-macro actions $a \notin M$ use T, R as usual

$$Q^{k+1}(s, a) = \begin{cases} a \notin M : & R(s, a) + \gamma \sum_{s'} T(s, a, s') V^k(s') \\ a = m \in M : & R_m(s) + \gamma \sum_{s'} T_m(s, s') V^k(s') \end{cases}$$

- Can solve with macros only, discard primitives
 - Cannot guarantee same optimal policy as primitives
 - But with good macros, often no loss & much faster...

Performance with & without Macros



Inverse Reinforcement Learning

(Ng & Russell, ICML-2000)

A Novel Problem – Not Vanilla RL

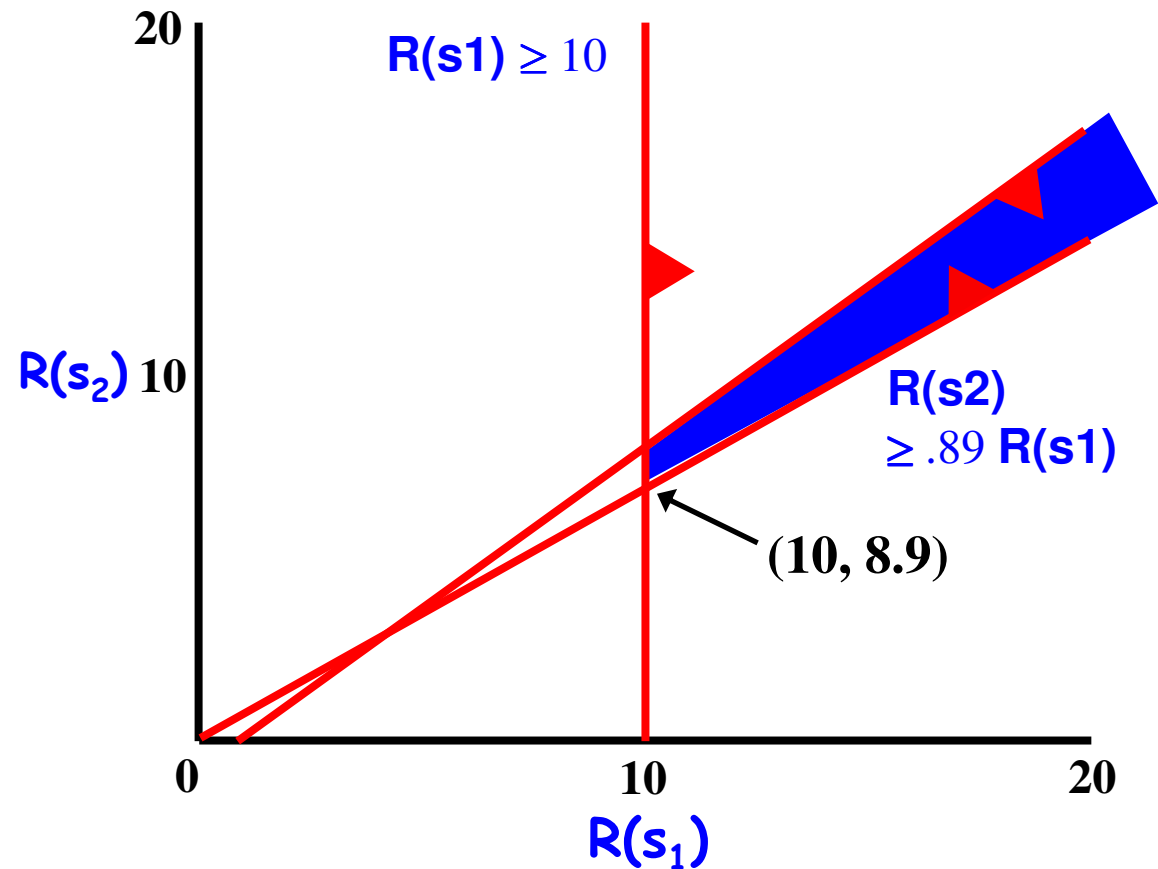
- Assume we observe (state, action) samples from policy π
- Want to derive R that resulted in π
 - assuming actor is rational
 - thus acting so as to maximize reward...
- Call this “Inverse RL”
 - Derive R from π , *not* π from R !

Idea: Use constraints

- Assume $S = \{s_1, s_2, s_3\}$, $A = \{a_1, a_2, a_3\}$
- Observe (s_1, a_2)
 - Then can infer (due to agent rationality)
 - $Q(s_1, a_2) \geq Q(s_1, a_1)$
 - $Q(s_1, a_2) \geq Q(s_1, a_3)$
- From Bellman equations
 - $Q(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a) \max_b Q(s', b)$
 - So expanding constraints $Q(s_1, a_2) \geq Q(s_1, a_1)$ with Q-definitions provides constraints on R!

Problem Solved?

- We have constraints on R , but many solutions (blue area)
- Which R is best?



Which Reward to Choose?

- Linear programming formulation in original paper specifies a linear objective
 - Separates best action Q-value from suboptimal
 - This version not often used in practice
- Instead, people use apriori preferences or Bayesian priors on rewards
 - Incorporate prior knowledge into problem
 - See Abbeel, Ng, (ICML-2004)
Ramachandran, Amir (IJCAI-2007)

Summary

- Reward shaping
 - Speedup learning process by providing hints in reward
 - Preserve policy optimality under shaping conditions
- Policy constraints
 - Useful if know a partial policy (program)
 - Write a partial program, have RL optimally fill-in rest!
- Macro actions
 - Can speedup solutions if know good subtasks for a problem
 - Can convert semi-MDP to MDP to solve MDP with macros
- Inverse reinforcement learning
 - A completely different problem than standard RL
 - Given samples of π , infer reward
 - Formulate as a constrained optimization problem