

# Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping

Paper by Andrew Ng, Daishi Harada, Stuart Russell

Presentation by Kristin Branson

April 9, 2002



# Reinforcement Learning

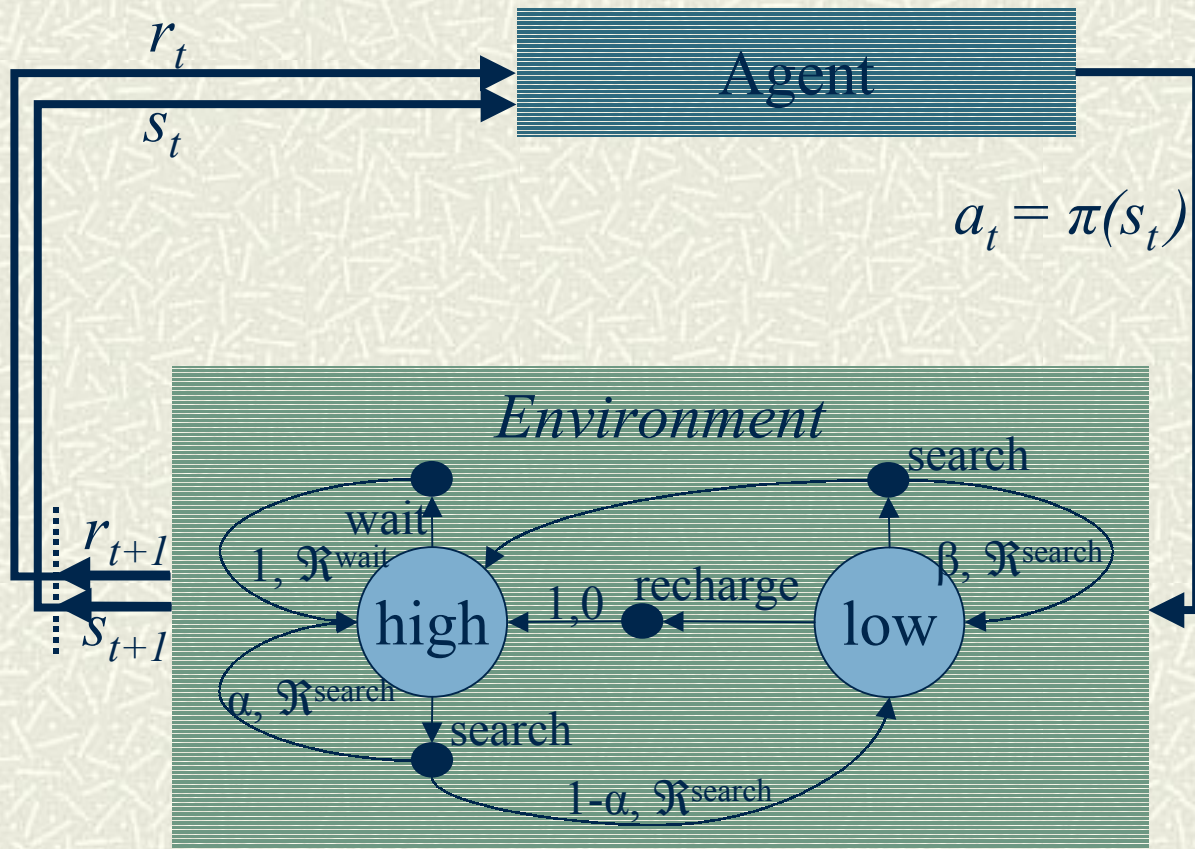
---

- # The reinforcement learning (RL) task:  
Learn the best action to take in any situation to maximize the total rewards earned.
- # The agent learns which actions to take by interacting with its environment.

# Markov Decision Processes (MDP)

- # In an MDP, the environment is represented as a Finite State Automaton with which the agent interacts in discrete episodes.
- # All information relevant to which action to take is in the environment's state (the Markov Property).
- #  $M = (S, A, T = \{P_{sa}(s')\}, \gamma, R)$ :
  - $S$  is the set of **states** of the environment.
  - $A$  is the set of **actions** the agent can take.
  - $P_{sa}(s')$  is the probability  $s'$  is the state after state  $s$  and action  $a$ .
  - $\gamma$  is the discount factor.
  - $R$  is the immediate reward function.

# MDPs: Example 1

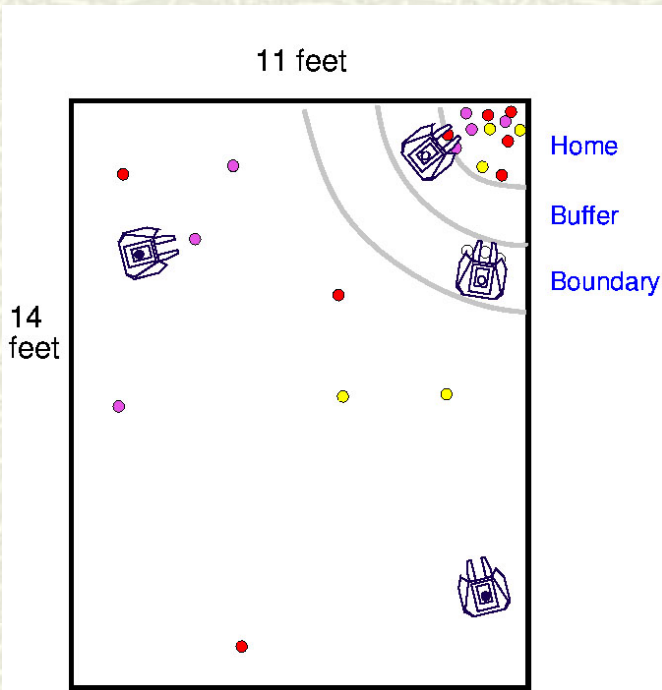


- #  $s_t$  is the state of the environment at time  $t$ .
- #  $r_t$  is the reward received at time  $t$ .
- # A policy,  $\pi$ , is a mapping from states to actions.

[Sutton and Bartow, 2000]

# Example: Foraging Robots

[Mataric, 1994]



- The robots' objective is to collectively find pucks and bring them home.
- Represent the 12D environment by the state variables:
  - have-puck?
  - at-home?
  - near-intruder?



- What should the immediate reward function be?

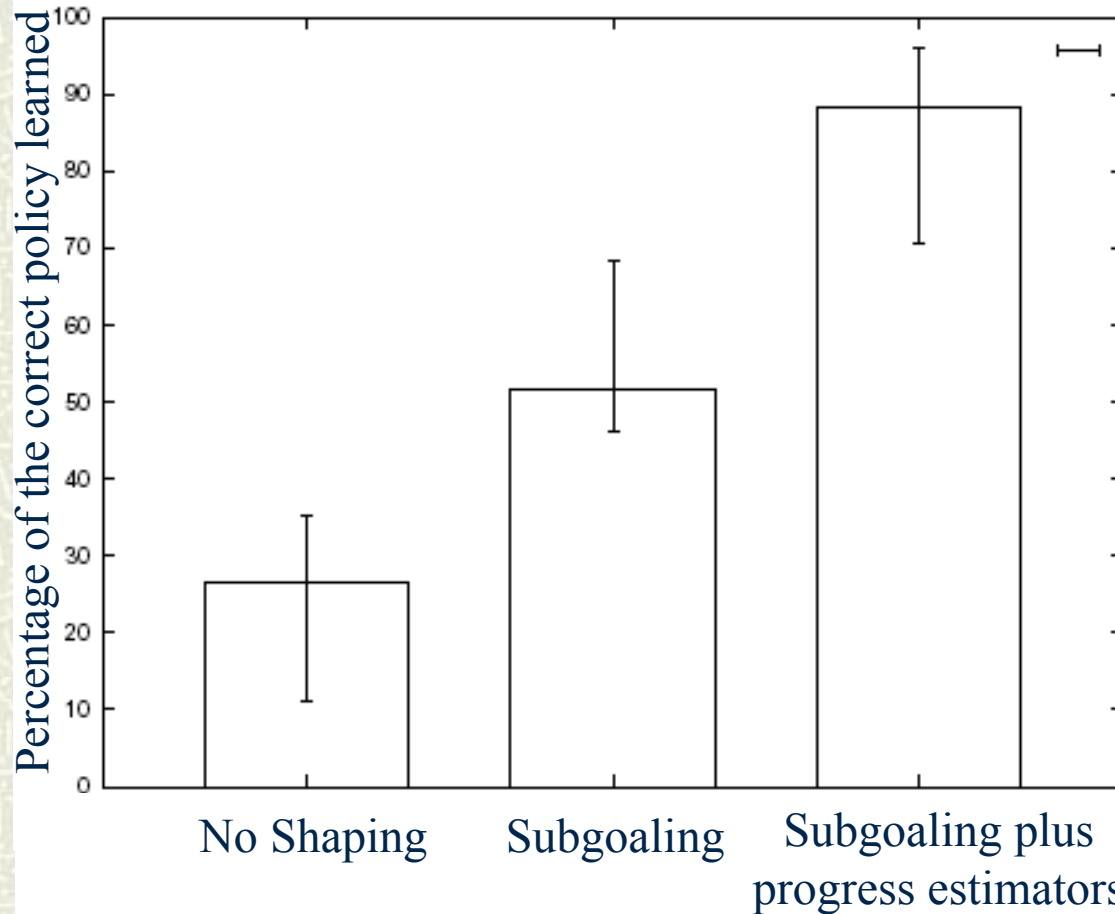
[<http://www-robotics.usc.edu/~agents/>]

# Reward Shaping

---

- # If a reward is given only when a robot drops a puck at home, learning will be extremely difficult.
    - The delay between the action and the reward is large.
  - # Solution: Reward shaping (intermediate rewards).
    - Add rewards/penalties for achieving subgoals/errors:
      - subgoal: grasped-puck
      - subgoal: dropped-puck-at-home
      - error: dropped-puck-away-from-home
    - Add progress estimators:
      - Intruder-avoiding progress function
      - Homing progress function
  - # Adding intermediate rewards will potentially allow RL to handle more complex problems.
-

# Reward Shaping Results



[Mataric, 1994]

# Percentage of policy learned after 15 minutes

# Policy Invariant Reward Transformations

---

- # What reward function transformations do not change the (near-) optimal policy?
  - # The answer is useful for reward shaping.
    - Often, reward shaping does not aim to change the ultimate policy.
    - Instead, it aims to help the agent learn the policy.
    - Policy invariant shaping avoids reward shaping that misleads the agents.
      - Example: Positive reward cycles.
  - # Understanding policy-preserving reward transformations will help us understand how precisely a reward function can be deduced from observing optimal behavior.
-

# Outline

---

- # Introduction to reinforcement learning and reward shaping
  - ➔ Value functions
  - # Potential-based shaping functions
  - # Proof that potential-based shaping functions are policy invariant.
  - # Proof that, given no other knowledge about the domain, potential-based shaping functions are necessary for policy invariance.
  - # Experiments investigating the effects of different potential-based shaping reward functions on RL.
-

# Value Function Definitions

- ✦ The state-value function estimates how good it is to be in a certain state, given the policy:

$$V(s) = E[R | s_0 = s] = E\left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | s_0 = s\right]$$

- ✦ The optimal state-value function is  $V^*(s) = \sup_{\pi} V^{\pi}(s)$

- ✦ The Q-function estimates how good it is to perform an action in a state, given the policy:

$$Q(s, a) = E[R | s_0 = s, a_0 = a] = E\left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | s_0 = s, a_0 = a\right]$$

- ✦ The optimal Q-function is

$$Q^*(s, a) = \sup_{\pi} Q^{\pi}(s, a)$$

# Recursive Relationships of Value Functions

# The Q-functions are recursively related:

$$\begin{aligned} Q^\pi(s, a) &= E_\pi [R_t | s_t = s, a_t = a] \\ &= E_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right] \\ &= E_\pi \left[ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a \right] \\ &= \sum_{s'} P_{sa}(s') \left[ r(s, a, s') + \gamma E_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s', a_{t+1} = a' \right] \right] \\ &= \sum_{s'} P_{sa}(s') \left[ r(s, a, s') + \gamma Q_M^\pi(s', a') \right] \quad \text{Bellman Equation} \end{aligned}$$

# There is one Bellman equation for each state.

# Representation of Shaping Rewards

- Under reward shaping, an increment is added to each immediate reward.
- The MDP  $(S, A, T, \gamma, R)$  is transformed to the MDP  $(S, A, T, \gamma, R')$ , where:

$$r'(s, a, s') = r(s, a, s') + F(s, a, s').$$

and  $F$  is a function  $F : S \times A \times S \mapsto \mathbf{R}$

- A **potential-based** shaping function is the difference of potentials, for all  $s, a, s'$ ,

$$F(s, a, s') = \gamma \Phi(s') - \Phi(s)$$

Discount factor  $\longrightarrow$

# Outline of Sufficiency Proof

- **Sufficiency Theorem:** If  $F$  is a potential-based shaping function, then it is policy invariant.
  - $M$  is the original MDP;  $M'$  is the original MDP plus the shaping function  $F(s, a, s') = \gamma\Phi(s') - \Phi(s)$ .
  - Claim 1: Any policy that optimizes  $Q_{M'}^*(s, a)$  also optimizes  $Q_M^*(s, a)$ .
  - Claim 2: Any policy that optimizes  $Q_M^*(s, a)$  also optimizes  $Q_{M'}^*(s, a)$ .
  - From claims 1 and 2, any optimal policy for  $M$  is also an optimal policy for  $M'$ , and vice-versa. Therefore,  $F$  is policy invariant.

# Sufficiency Theorem

- # If  $F$  is a potential-based shaping function, then it is policy invariant.
- # Claim 1: Any policy that optimizes  $Q_M^*(s, a)$  also optimizes  $Q_M^*(s, a)$ .

- Begin with the Bellman equation for the optimal Q-function for  $M$ .

$$Q_M^*(s, a) = \sum_{s'} P_{sa}(s') \left[ r(s, a, s') + \gamma \max_{a' \in A} Q_M^*(s', a') \right]$$

$$Q_M^*(s, a) - \Phi(s) = \sum_{s'} P_{sa}(s') \left[ r(s, a, s') + \gamma \max_{a' \in A} Q_M^*(s', a') \right] - \Phi(s)$$

$$= \sum_{s'} P_{sa}(s') \left[ r(s, a, s') - \Phi(s) + \gamma \max_{a' \in A} Q_M^*(s', a') \right]$$

$$= \sum_{s'} P_{sa}(s') \left[ r(s, a, s') + \gamma \Phi(s') - \Phi(s) + \gamma \max_{a' \in A} [Q_M^*(s', a') - \Phi(s')] \right]$$

$$= \sum_{s'} P_{sa}(s') \left[ r(s, a, s') + F(s, a, s') + \gamma \max_{a' \in A} [Q_M^*(s', a') - \Phi(s)] \right]$$

$$= \sum_{s'} P_{sa}(s') \left[ r'(s, a, s') + \gamma \max_{a' \in A} [Q_M^*(s', a') - \Phi(s)] \right]$$

# Proof of Sufficiency Theorem, cont.

▣ Above is the Bellman equation for  $Q_{M'}^*(s, a)$ , if we substitute

$$Q_{M'}^* = Q_M^* - \Phi(s) :$$
$$Q_M^*(s, a) - \Phi(s) = \sum_{s'} P_{sa}(s') \left[ r'(s, a, s') + \gamma \max_{a' \in A} \left[ Q_M^*(s', a') - \Phi(s) \right] \right]$$

$$Q_{M'}^*(s, a) = \sum_{s'} P_{sa}(s') \left[ r'(s, a, s') + \gamma \max_{a' \in A} \left[ Q_{M'}^*(s, a) \right] \right]$$

▣ Therefore, any policy that optimizes  $Q_{M'}^*(s, a)$  also optimizes

$Q_{M'}^* = Q_M^* - \Phi(s)$ . Since  $\Phi(s)$  does not depend on the action chosen in state  $s$ , this policy also maximizes  $Q_M^*(s, a)$ . That is, an optimal policy for  $M'$  is also an optimal policy for  $M$ .

▣ Since  $Q_{M'}^* = Q_M^* - \Phi(s)$  and  $Q_M^*(s, a) = \sum_{s'} P_{sa}(s') \left[ r(s, a, s') + \gamma V_M^*(s') \right]$ ,

$$V_{M'}^* = V_M^* - \Phi(s)$$

# Proof of Sufficiency Theorem, cont.

- # Claim 2: Any policy that optimizes  $Q_M^*(s, a)$  also optimizes  $Q_{M'}^*(s, a)$
- # Begin with the Bellman equation for the optimal Q-function for  $M'$ .

$$Q_{M'}^*(s, a) = \sum_{s'} P_{sa}(s') \left[ r'(s, a, s') + \gamma \max_{a' \in A} Q_{M'}^*(s', a') \right]$$

$$\begin{aligned} Q_{M'}^*(s, a) + \Phi(s) &= \sum_{s'} P_{sa}(s') \left[ r'(s, a, s') + \gamma \max_{a' \in A} Q_{M'}^*(s', a') \right] + \Phi(s) \\ &= \sum_{s'} P_{sa}(s') \left[ r'(s, a, s') + \Phi(s) + \gamma \max_{a' \in A} Q_{M'}^*(s', a') \right] \\ &= \sum_{s'} P_{sa}(s') \left[ r'(s, a, s') - \gamma \Phi(s') + \Phi(s) + \gamma \max_{a' \in A} (Q_{M'}^*(s', a') + \Phi(s')) \right] \\ &= \sum_{s'} P_{sa}(s') \left[ r'(s, a, s') - F(s, a, s') + \gamma \max_{a' \in A} (Q_{M'}^*(s', a') + \Phi(s)) \right] \\ &= \sum_{s'} P_{sa}(s') \left[ r(s, a, s') + \gamma \max_{a' \in A} (Q_{M'}^*(s', a') + \Phi(s)) \right] \end{aligned}$$

# Proof of Sufficiency Theorem, cont.

- ⌘ Above is the Bellman equation for  $Q_M^*(s, a)$ , if we substitute  $Q_M^* = Q_{M'}^* + \Phi(s)$  :

$$Q_{M'}^*(s, a) + \Phi(s) = \sum_{s'} P_{sa}(s') \left[ r(s, a, s') + \gamma \max_{a' \in A} (Q_{M'}^*(s', a') + \Phi(s)) \right]$$

$$Q_M^*(s, a) = \sum_{s'} P_{sa}(s') \left[ r(s, a, s') + \gamma \max_{a' \in A} (Q_M^*(s, a')) \right]$$

- ⌘ Therefore, any policy that optimizes  $Q_M^*(s, a)$  also optimizes  $Q_M^* = Q_{M'}^* + \Phi(s)$ . Since  $\Phi(s)$  does not depend on the action chosen in state  $s$ , this policy also maximizes  $Q_{M'}^*(s, a)$ . That is, an optimal policy for  $M$  is also an optimal policy for  $M'$ .

# Robustness

---

- # We have proved that optimal policies for  $M'$  map to optimal policies for  $M$ , and vice versa.
  - # Robustness of mapping: for an arbitrary policy,  $\pi$ , if  $|V_{M'}^\pi(s) - V_{M'}^*(s)| < \varepsilon$ , then  $|V_M^\pi(s) - V_M^*(s)| < \varepsilon$ .
  - # Therefore, near-optimal policies are also preserved.
-

# Necessity Theorem

- # If  $F$  is not potential-based, then there is an MDP  $M$  such that no optimal policy in  $M$  is also optimal in  $M$  with  $F$ .
- # That is, for each  $F$  that is not potential-based, there exists  $M$  s.t. for each policy  $\pi$ , if  $\pi$  is optimal in  $M$  then  $\pi$  is not optimal in  $M$  with  $F$ .
- # Split proof into two cases:
  - Case 1:  $F$  depends on the action, i.e. there exist actions  $a, a'$  such that  $\Delta = F(s, a, s') - F(s, a', s') > 0$ .
  - Case 2:  $F$  does not depend on the action, i.e.  $F(s, a, s') = F(s, s')$ .

# Proof of Necessity Theorem

■ Case 1:  $F$  depends on the action, i.e. there exist actions  $a, a'$  such that  $\Delta = F(s, a, s') - F(s, a', s') > 0$ .

■ Construct  $M$  such that: ■ Given this construction:

■  $P_{sa}(s') = P_{sa'}(s') = 1.0$

■  $r(s, a, s') = 0$

■  $r(s, a', s') = \Delta / 2$

■  $\pi_M^*(s) = a'$

■  $r'(s, a, s') = F(s, a, s')$

■  $r'(s, a', s') = \frac{\Delta}{2} + F(s, a', s')$

$$= \Delta + F(s, a', s') - \frac{\Delta}{2}$$

$$= F(s, a, s') - F(s, a', s') + F(s, a', s') - \frac{\Delta}{2}$$

$$= F(s, a, s') - \frac{\Delta}{2}$$

■ Therefore,  $\pi_{M'}^*(s) = a$ , which is not the same as  $\pi_M^*(s) = a'$

# Proof of Necessity Theorem, cont

- Case 2:  $F$  does not depend on the action, i.e.  $F(s, a, s') = F(s, s')$ .
- We can assume, w/o loss of generality, that  $F(\hat{s}_o, \hat{s}_o) = 0$ , where  $\hat{s}_o$  is the absorbing state, if  $\gamma = 1$ , and some fixed state otherwise.
- Since  $F$  is not potential based, for any potential function  $\Phi$ , there exist states  $s_1, s_2$  such that  $\gamma\Phi(s_2) - \Phi(s_1) \neq F(s_1, s_2)$  where  $s_1, s_2, \hat{s}_o$  are distinct.
- Construct  $M$  as follows:
  - $P_{s_1 a}(\hat{s}_o) = P_{s_1 a'}(s_2) = P_{s_2 a}(\hat{s}_o) = P_{\hat{s}_o a}(\hat{s}_o) = 1.0$
  - $\Delta = F(s_1, s_2) + \gamma F(s_2, \hat{s}_o) - F(s_1, \hat{s}_o)$
  - $r(s_1, a, \hat{s}_o) = \Delta / 2$ , otherwise  $r(\cdot, \cdot, \cdot) = 0$

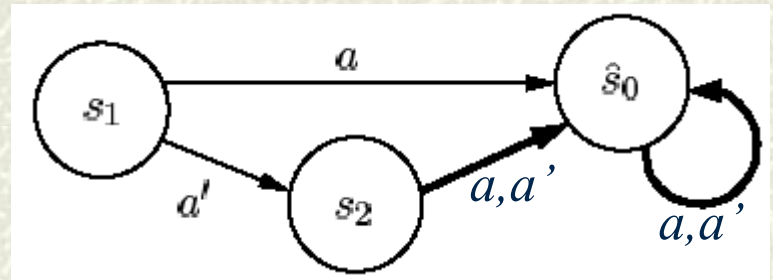


Diagram of  $M$

# Proof of Necessity Theorem, cont

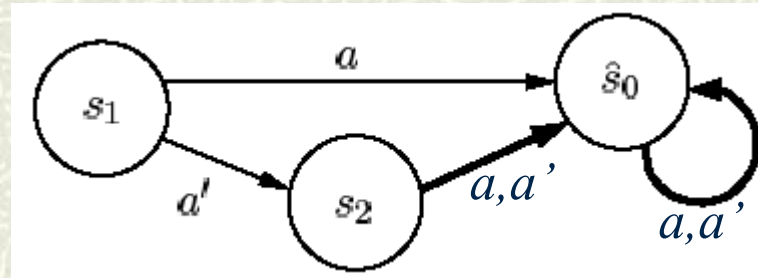
$$\# Q_M^*(s_1, a) = \frac{\Delta}{2} + \gamma V_M^*(\hat{s}_0) = \frac{\Delta}{2}$$

$$\begin{aligned} Q_M^*(s_1, a') &= 0 + \gamma V_M^*(s_2) \\ &= 0 + \gamma * 0 + V_M^*(\hat{s}_0) = 0 \end{aligned}$$

$$\begin{aligned} Q_{M'}^*(s_1, a) &= \frac{\Delta}{2} + F(s_1, \hat{s}_0) \\ &= \Delta + F(s_1, \hat{s}_0) - \frac{\Delta}{2} \\ &= F(s_1, s_2) + \gamma F(s_2, \hat{s}_0) - F(s_1, \hat{s}_0) + F(s_1, \hat{s}_0) - \frac{\Delta}{2} \\ &= F(s_1, s_2) + \gamma F(s_2, \hat{s}_0) - \frac{\Delta}{2}. \end{aligned}$$

$$Q_{M'}^*(s_1, a') = 0 + F(s_1, s_2) + \gamma V_{M'}^*(\hat{s}_0) = F(s_1, s_2) + \gamma F(s_2, \hat{s}_0)$$

# Therefore,  $\pi_M^*(s) = a$ , which is not the same as  $\pi_{M'}^*(s) = a'$



*Diagram of M*

# Choosing a potential function

- # If  $\Phi(s) = V_M^*(s)$  then  $V_{M'}^* = V_M^* - \Phi(s) = V_M^* - V_M^* = 0$
- # This form makes learning easy because the recursion in the Q-equations has been removed:

$$\begin{aligned} Q_M^*(s, a) &= \sum_{s'} P_{sa}(s') [r(s, a, s') + \gamma V_M^*(s')] \\ &= \sum_{s'} P_{sa}(s') [r(s, a, s') + 0] \end{aligned}$$

- # Without knowing the actual value of  $V_M^*(s, a)$  we can use our knowledge about the domain to estimate  $V_M^*(s, a)$ .

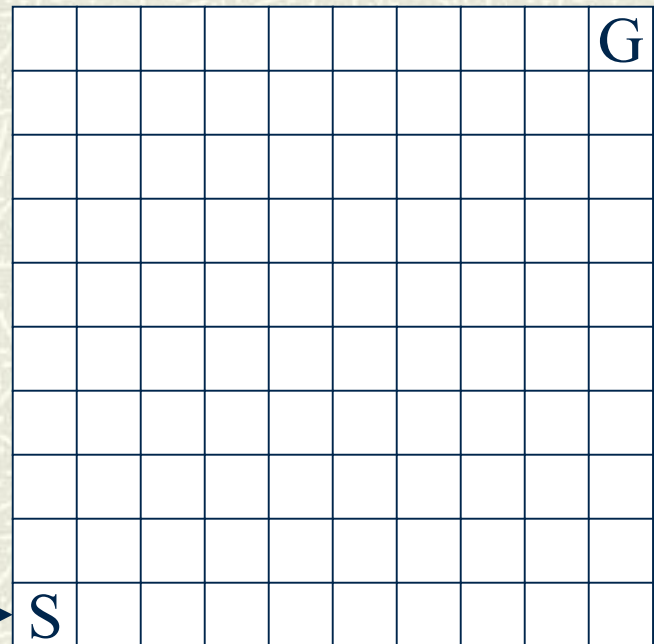
# Experiments: Domain 1

## ■ Grid-world domain:

### ■ Domain 1:

- -1 reinforcement per step
- Actions: four directions
  - Travel 1 step in the intended direction 80%, random direction 20%.

start →



← goal

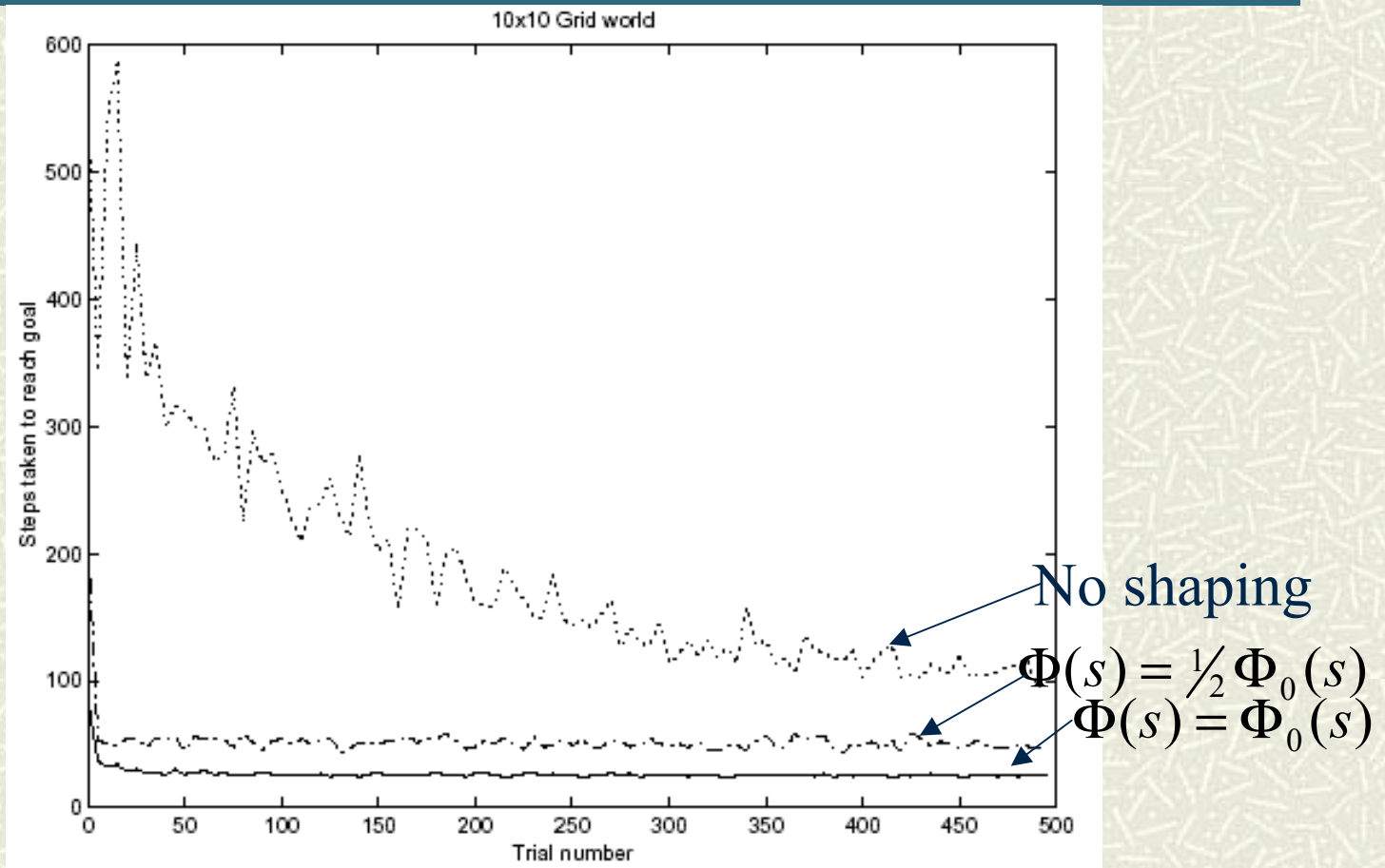
- Estimate the expected number of steps to get to the goal from state  $s$  as:

$$\Phi_0(s) = (-1) \cdot \text{MANHATTAN}(s, G) / 0.8$$

- A shaping function that is far from the estimate of  $V_M^*(s, a)$ :

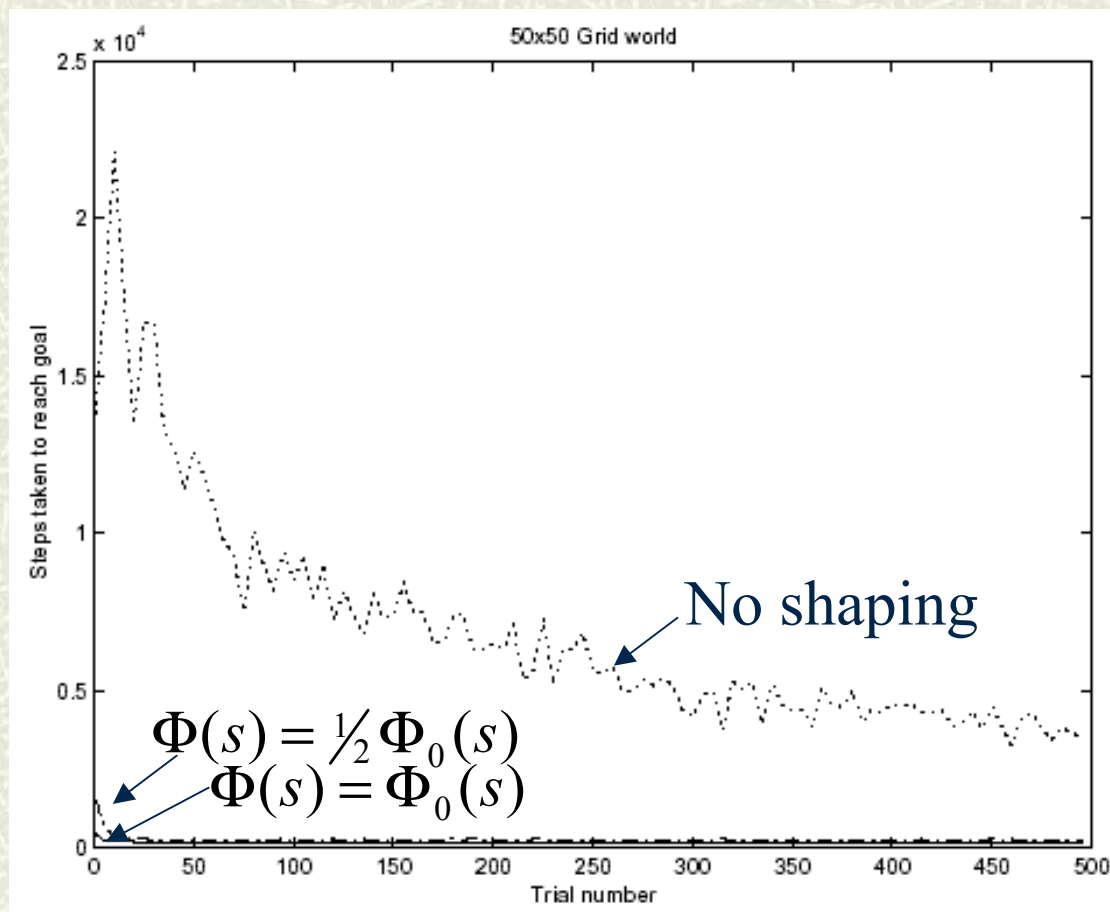
$$\Phi(s) = \frac{1}{2} \Phi_0(s)$$

# Results in Domain 1



Steps taken to reach goal after  $x$  iterations of learning in Domain 1

# Results with 50x50 Grid



Steps taken to reach goal after  $x$  iterations of learning in Domain 1

# Experiments: Domain 2

## # Domain 2:

- Must start at S and travel from 1 to 4 in sequence, then to the goal, G = 5.
- Actions, rewards are the same as in Domain 1.
- Task is completed after the agent has traveled from 1 to 5 in sequence.
- If  $t$  is the number of time steps to reach G from S, estimate the number of steps to go after reaching the  $n$ th subgoal as:

$$\Phi_0(s) = -((5 - n - 0.5) / 5)t$$

$(5 - n)$  subgoals to go

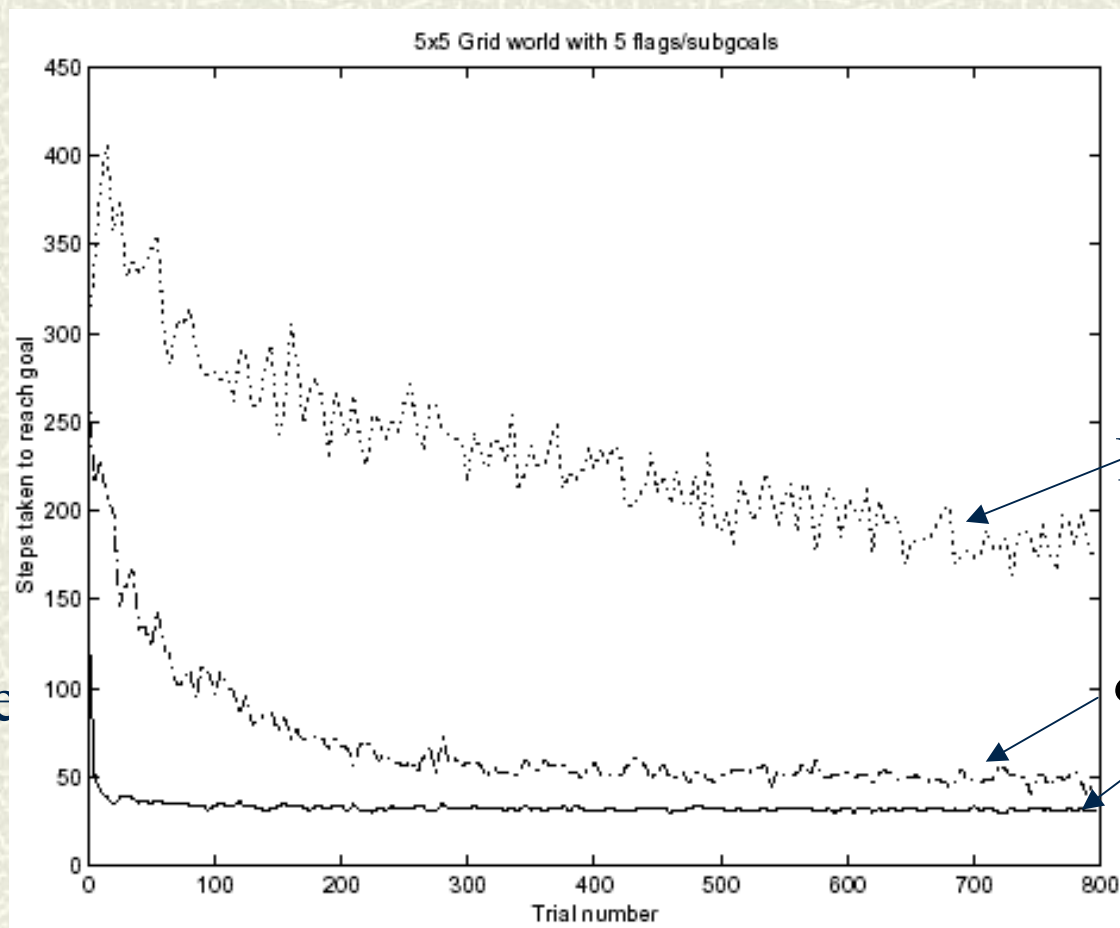
About halfway between goals  $n$  and  $n+1$

5 total goals

|   |   |  |  |   |
|---|---|--|--|---|
|   |   |  |  | G |
|   | 2 |  |  |   |
|   |   |  |  |   |
| 3 |   |  |  | 1 |
| S |   |  |  | 4 |

# Results in Domain 2

$\hat{V}_M(s)$  is a more precise estimate of the remaining time to goal



Steps taken to reach goal after  $x$  iterations of learning in Domain 2

# Conclusions

---

- Potential-based shaping functions are policy invariant, and are the necessary form to ensure policy invariance.
- Policy invariance is desired to prevent misleading shaping.
  - One could also try shaping functions inspired by potential-based shaping functions.
  - Example: The discount factor,  $\gamma$ , is set to be less than 1, not because rewards decrease with time, but to improve convergence. Using an undiscounted shaping function may work.
    - Potential-based function:  $F(s, a, s') = \gamma\Phi(s') - \Phi(s)$
    - Potential-inspired function:  $F(s, a, s') = \Phi(s') - \Phi(s)$

# Conclusions

---

- # Following from the robustness argument (near-optimal policies are preserved by potential-based shaping functions), potential-based shaping functions can be used to choose from a class of policies.
  - Choosing from a class of policies is useful for “parametric” RL, where the policy template is fixed.
    - This is computationally simpler, more compact, and requires less training.
    - This can also be useful to express knowledge about the domain.
- # Potential-based shaping functions also generalize to Semi-Markov Decision Processes.
- # “Advantage learning” and “ $\lambda$ -policy iteration” may be used to optimize the potential function used in reward shaping.