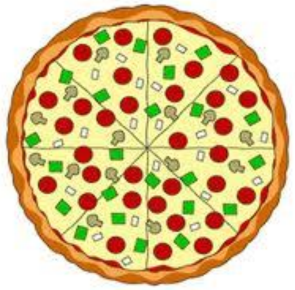
Risk-Variant Policy Switching to Exceed Reward Thresholds

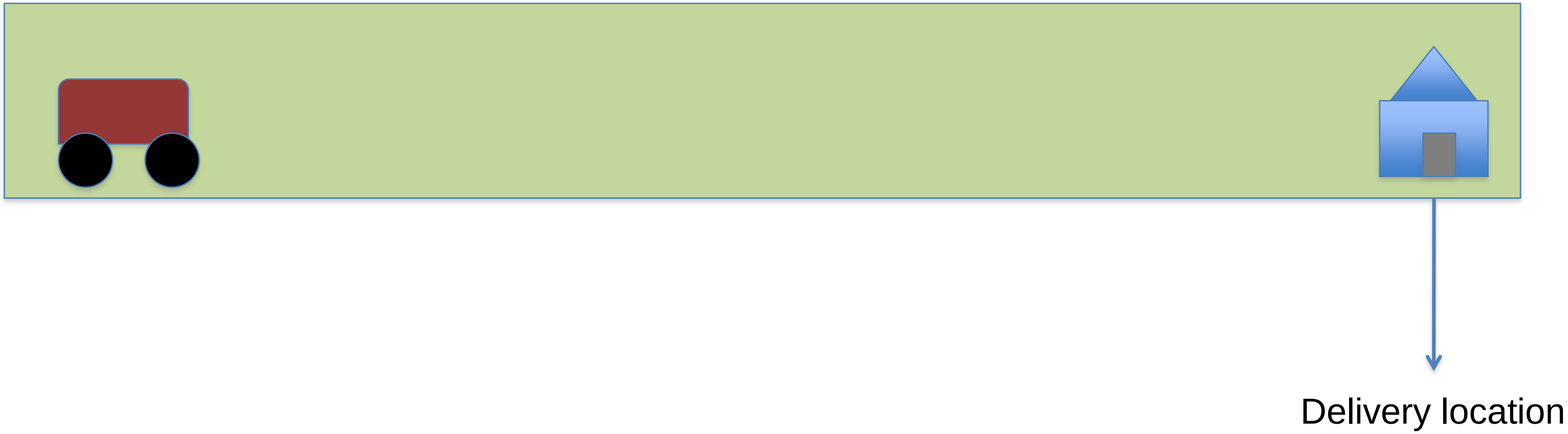
Breelyn Kane and Reid Simmons

June 29, 2012

Pizza Delivery

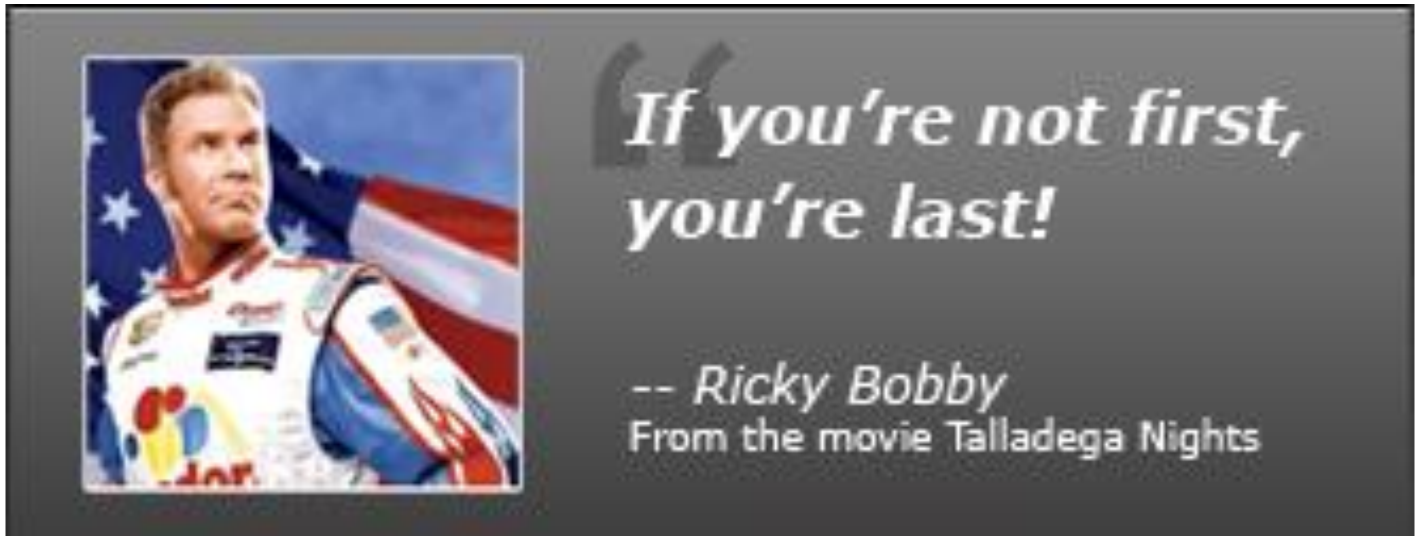


"30 Minutes or It's Free"



Agent's Goal: Exceed a Threshold

- In competitive domains, second is as good as last.
- "The person that said winning isn't everything, never won anything" – Mia Hamm
- Arcade game - not just beating a level, going for the top score.

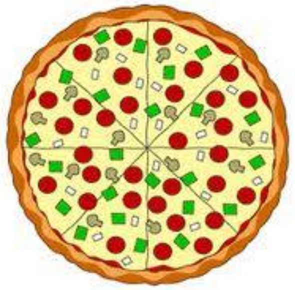


Take Risks to Win

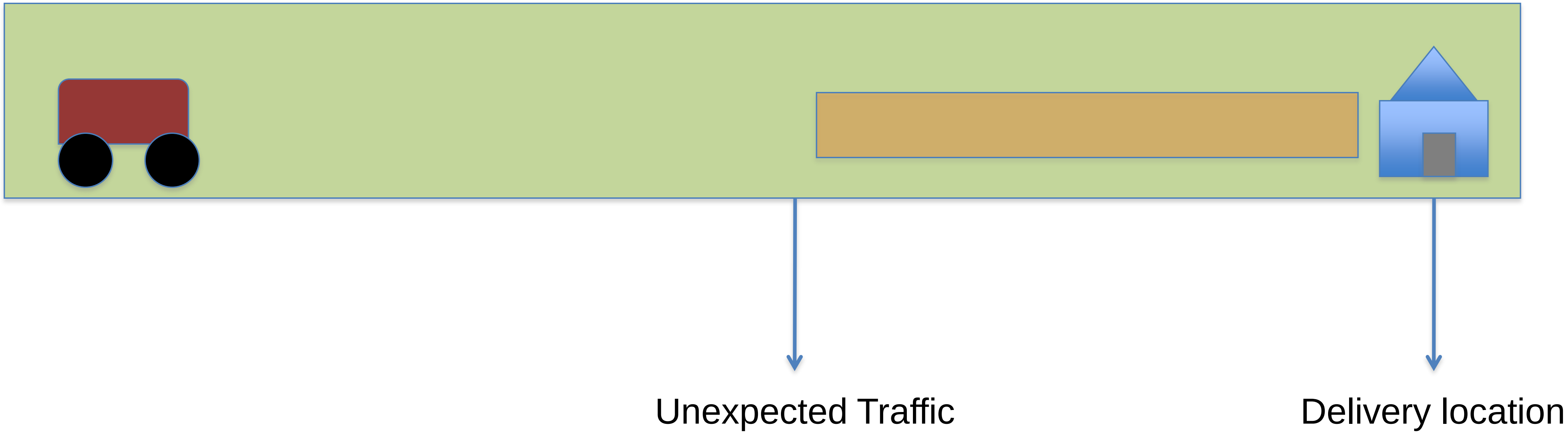
- Change strategy to win.
  - Play more defensively or offensively.

- Hockey: When is the best time to pull the goalie?

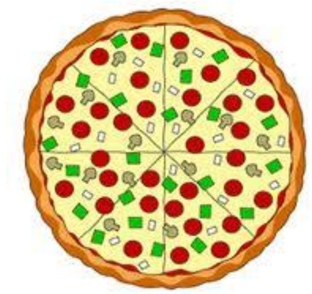
Pizza Delivery



"30 Minutes or It's Free"

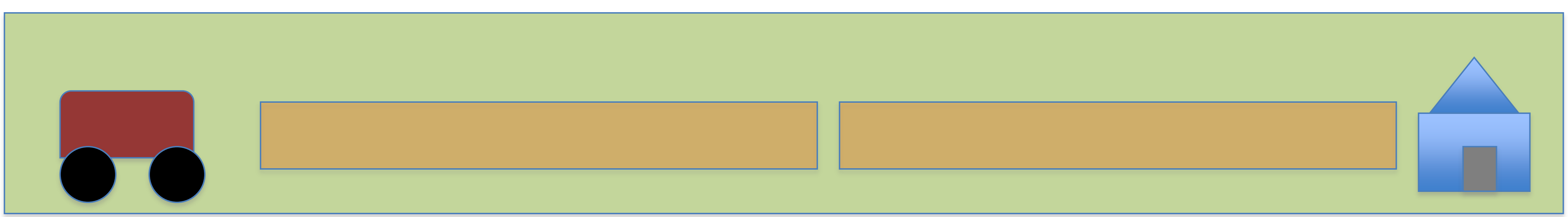


Risky Actions Create Higher Variance



Always risk-neutral

Always risky



Unlucky: cost high

Lucky: cost low

Application

- Specific Domain: thresholded-reward problems.

- Suite of risk-sensitive policies: generated based on exponential functions (assimilate risk).

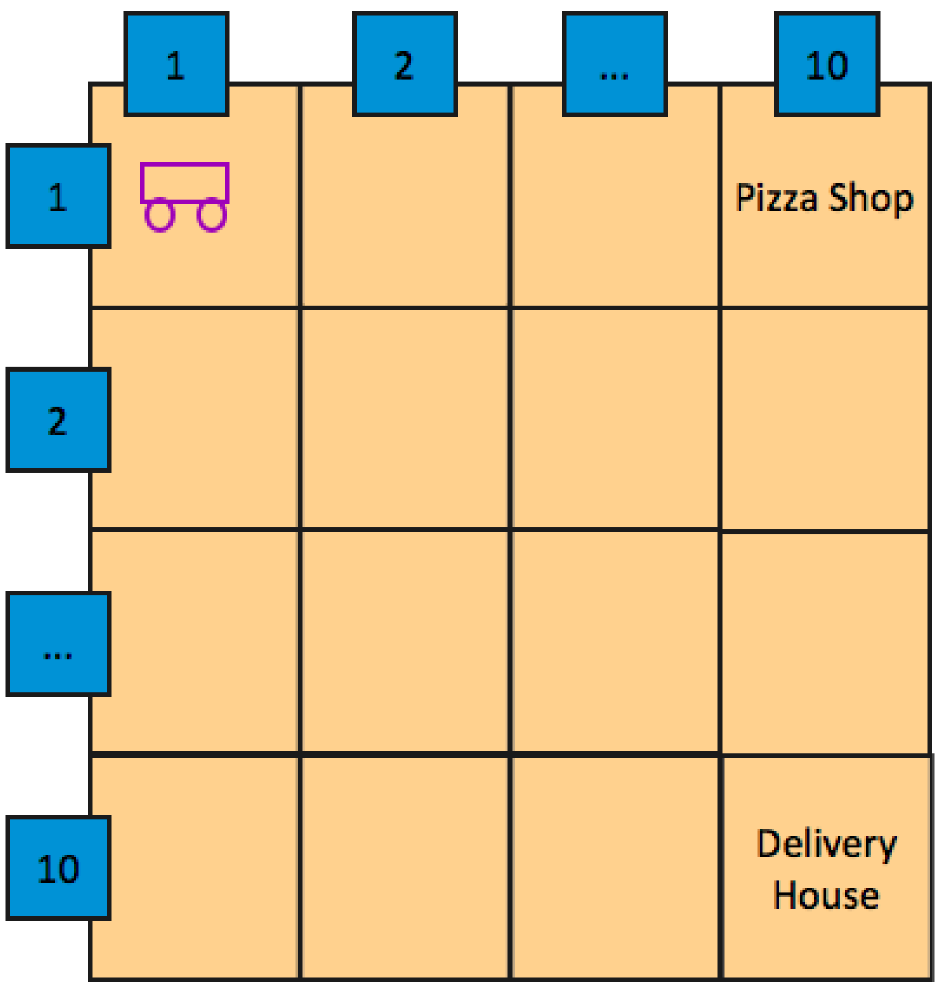
Method

Demonstrate the technique for choosing the next policy to follow based on ***maximizing the probability of exceeding a reward threshold.***

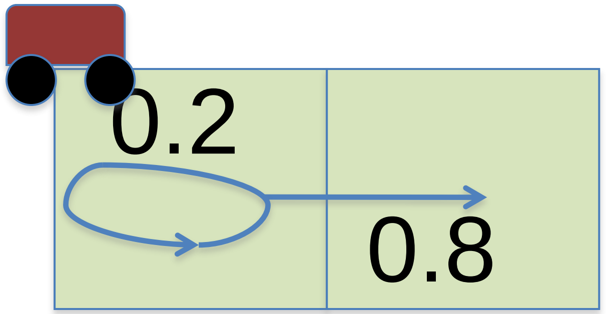
- Given : current state and current running cumulative reward.

Formulate the Problem as an MDP

- Assume world defined as a Markov Decision Process

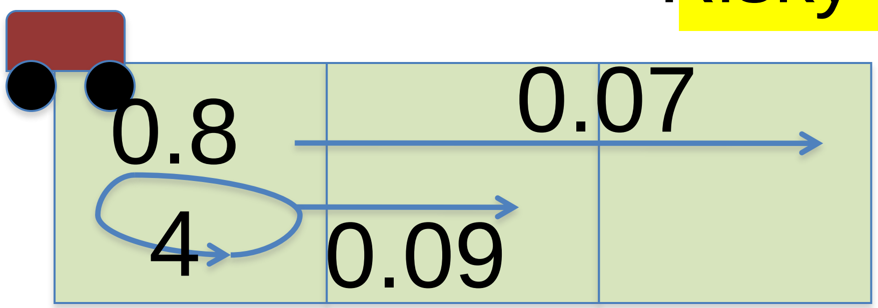


$P(s' \mid s, a)$



$r(s,a)$ -6

Move:E,W,N,S



$r(s,a)$ -2

Risky Move:E,W,N,S

Pickup, Dropoff

Deliver Success: $r(s,a)$ 50

A Planning Problem

- Straightforward Approach
  – Add cumulative reward or time to the state.
    - Significantly increases the state space.
  – Execute the optimal policy.

[McMillen, C, 2007. AAAI Press; MIT Press]

Intractable State Space for Realistic Domains.

A Planning Problem

What about a dynamic policy?

- If the agent is particularly unlucky, adjust how risky actions are at run-time.

Tradeoff some computation and somewhat less optimality for significant savings in planning time.

[Roth, M. 2005. Autonomous Agents and Multiagent Systems.]
[Cassandra,R.1998 PhD Thesis]
[Koenig,S .1995. IJCAI]

Approach

Offline

- **Generate different policies**: policies of varying risk attitude.
- **Estimate the reward distributions.**

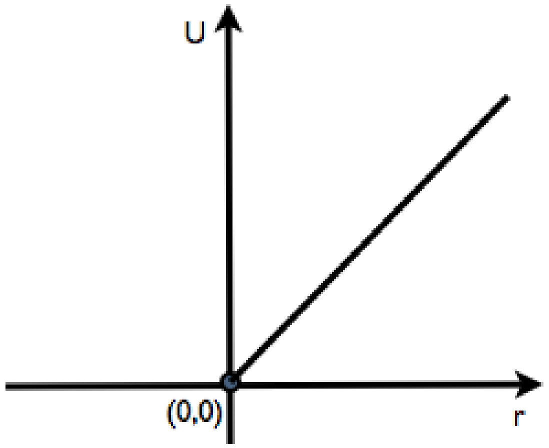
Online

- **Switch between policies**: Calculate the maximum probability of being over a threshold at each time step based on the current cumulative (discounted) reward.

Background on Risk

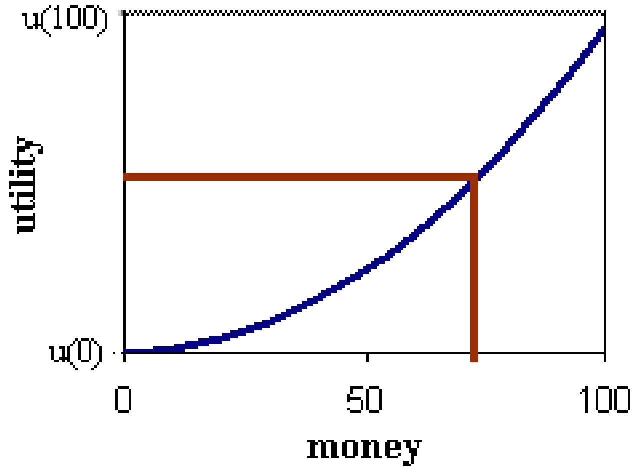
Utility and Risk

- linear utility

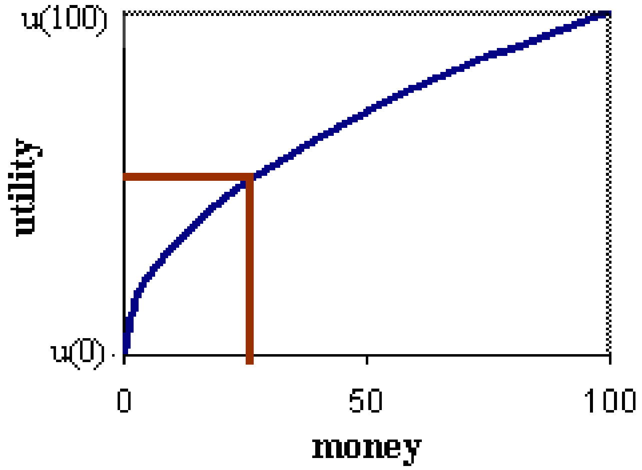


- exponential utility $$U(r) = \pm\delta^r$$
  - most widely used function to represent risk-sensitive utility.

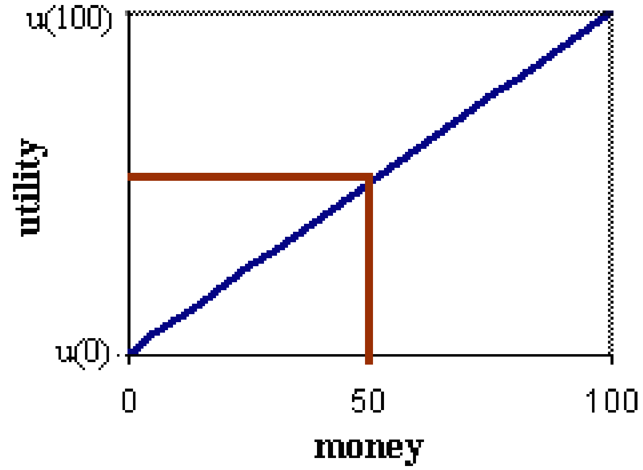
Risk-seeking



Risk-averse



Risk-Neutral



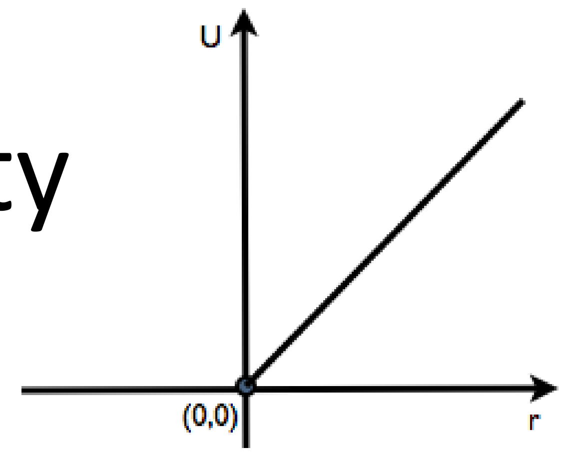
$$U(r) = \delta^r, \delta > 1$$

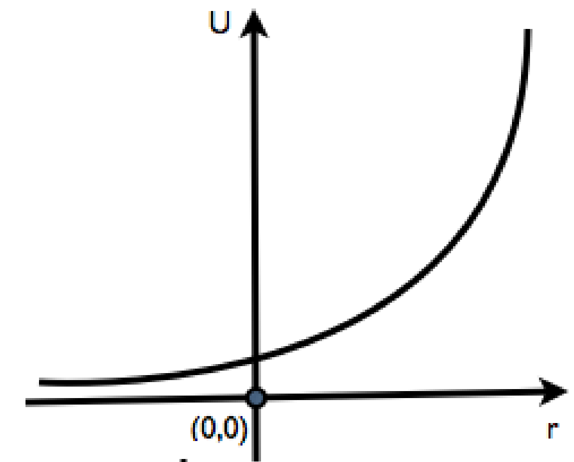
$$U(r) = -\delta^r, 0 < \delta < 1$$

$$U(r) = r$$

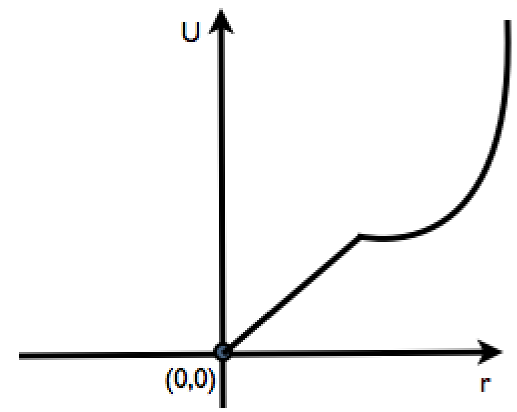
An Agent’s Utility Function

- ‘no-switch’ utility
  - stays constant





- one-switch utility
  - switch from risk-neutral to risk-seeking behavior.



- prefer multi-switch utility

Emulate multi-switch utility by switching between policies at run-time.

[Liu, Y., and Koenig, S. 2008. An exact algorithm for solving mdps under risk-sensitive planning objectives with one-switch utility functions. AAMAS.]

Technical Approach

Approach

Offline

- **Generate different policies**: policies of varying risk attitude.
- **Estimate the reward distributions.**

Online

- **Switch between policies**: Calculate the maximum probability of being over a threshold at each time step based on the current cumulative (discounted) reward.

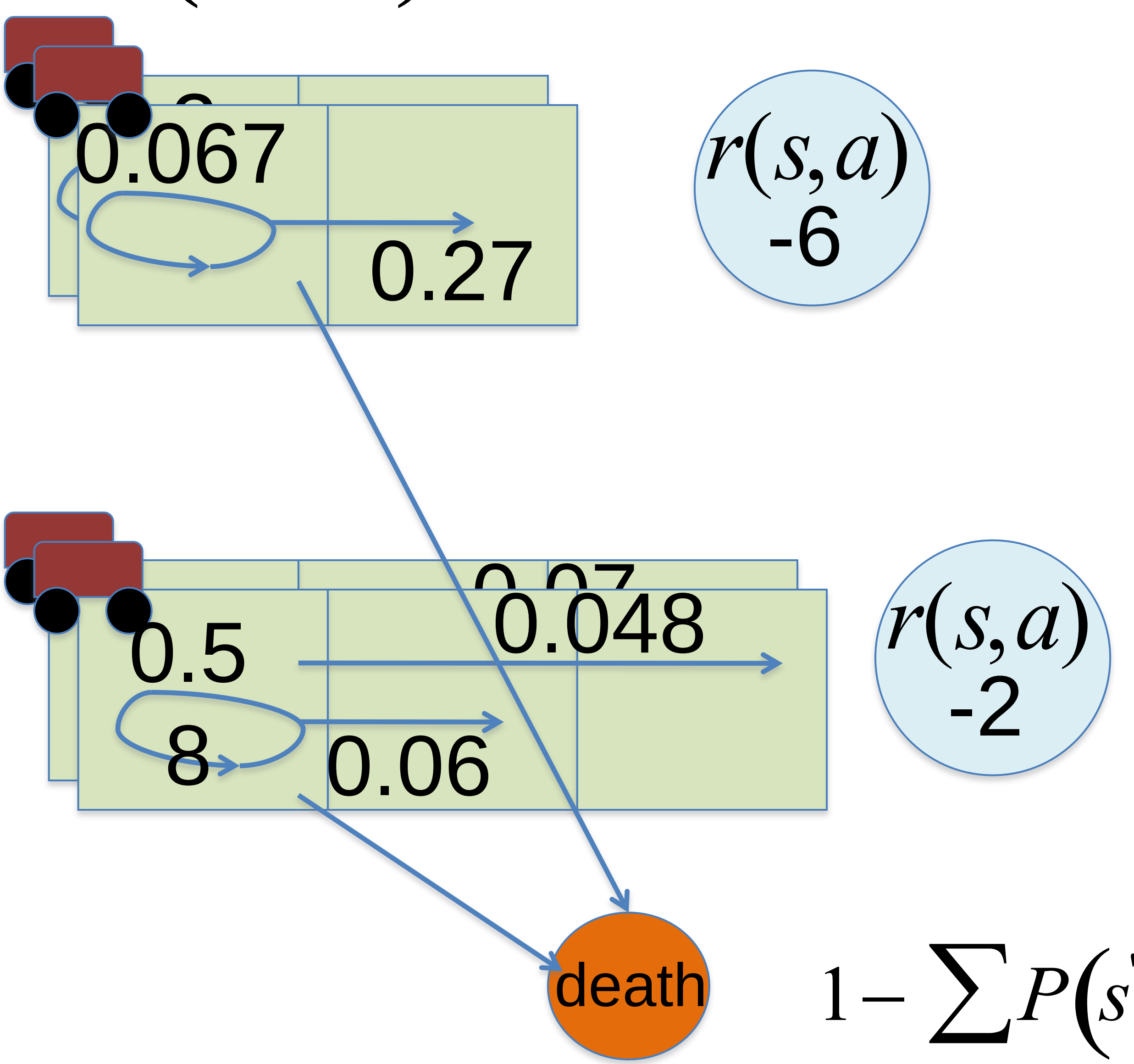
Transform the MDP Probabilities

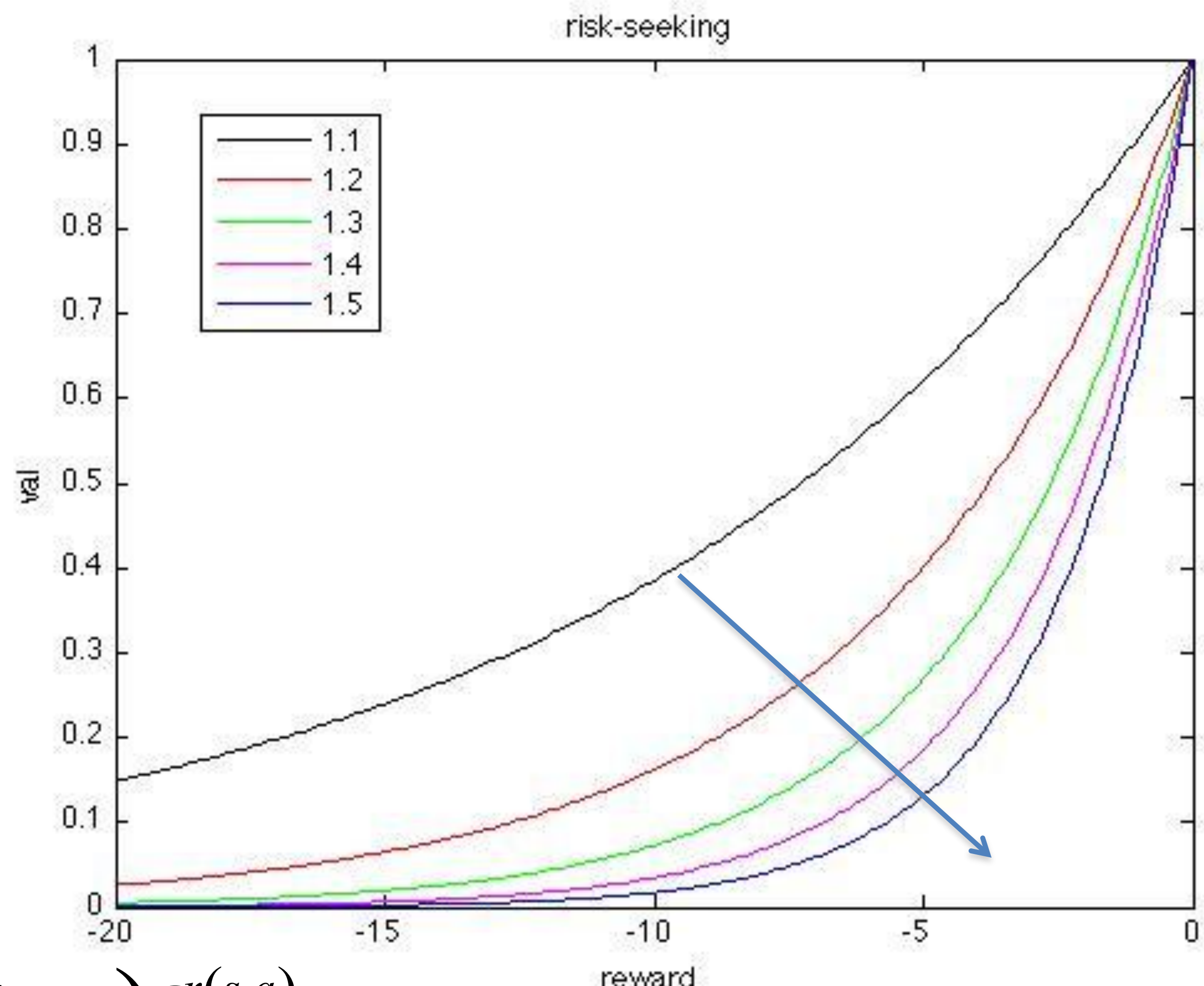
$$P(s' \mid s,a) * \delta^{r(s,a)}$$

$P(s' \mid s,a)$

$\delta = 1.2$

$\delta^{r(s,a)}, \delta > 1$





$$1 - \sum_{s' \in S} P(s' \mid s,a)\,\delta^{r(s,a)}$$

[S. Koenig. Goal-Directed Acting with Incomplete information. PhD thesis1997]

Approach

Offline

- **Generate different policies**: policies of varying risk attitude.
- **Estimate the reward distributions.**

Online

- **Switch between policies**: Calculate the maximum probability of being over a threshold at each time step based on the current cumulative (discounted) reward.

Estimating the Reward Distribution

This work reasons about the **complete** non-parametric reward distribution including the distribution tails.

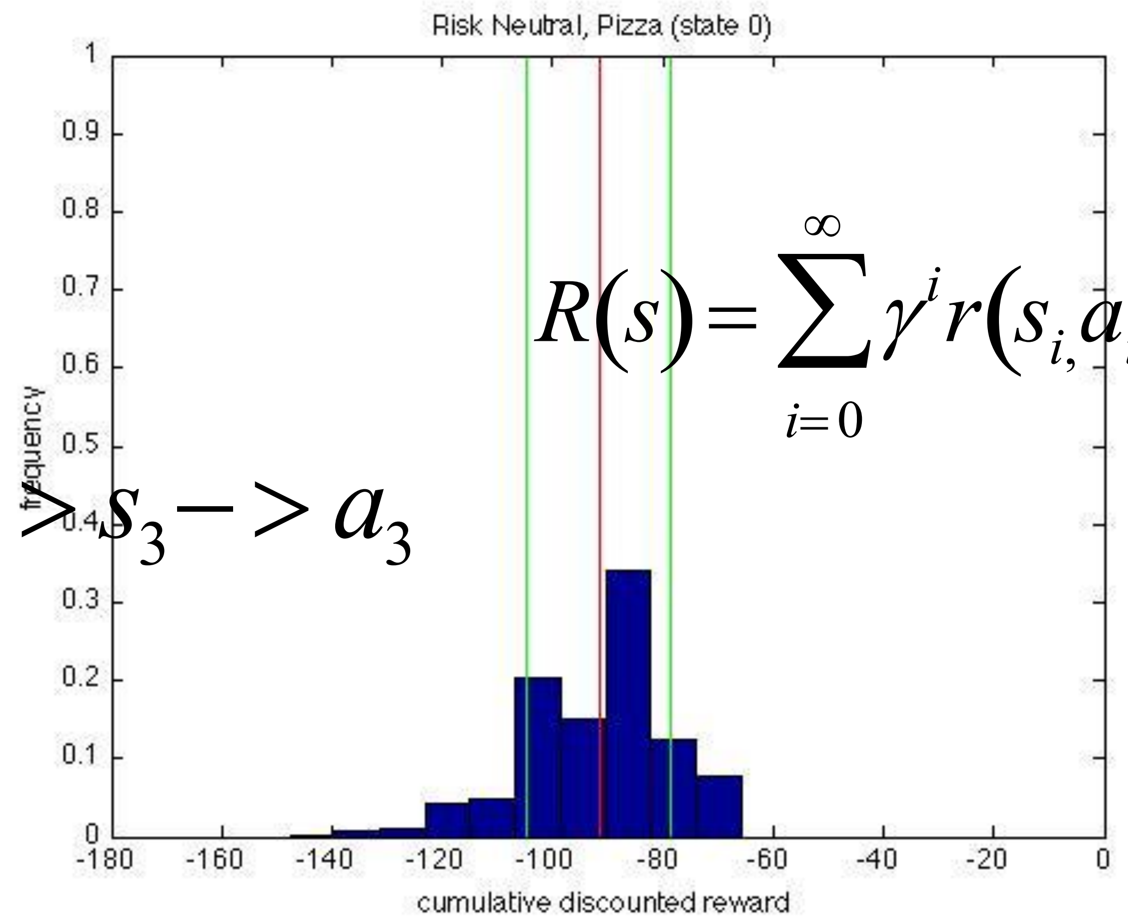
For each policy

For each state

$$s_1 -> a_1 -> s_2 -> a_2$$

$$s_2 -> a_2 -> s_3 -> a_3$$

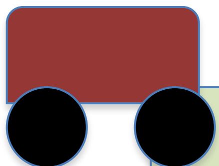
$$s_3 -> a_3 -> s_4 -> a_4 -> s_3 -> a_3$$



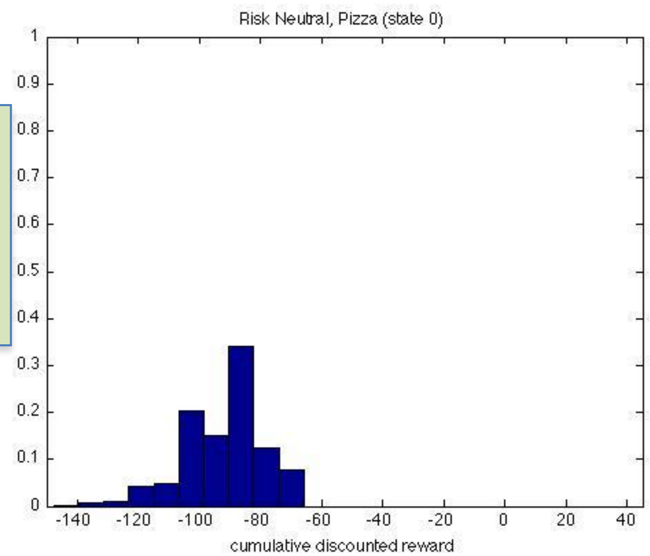
$$R(s) = \sum_{i=0}^{\infty} \gamma^i r(s_i, a_i)$$

Distributions Vary Based on State

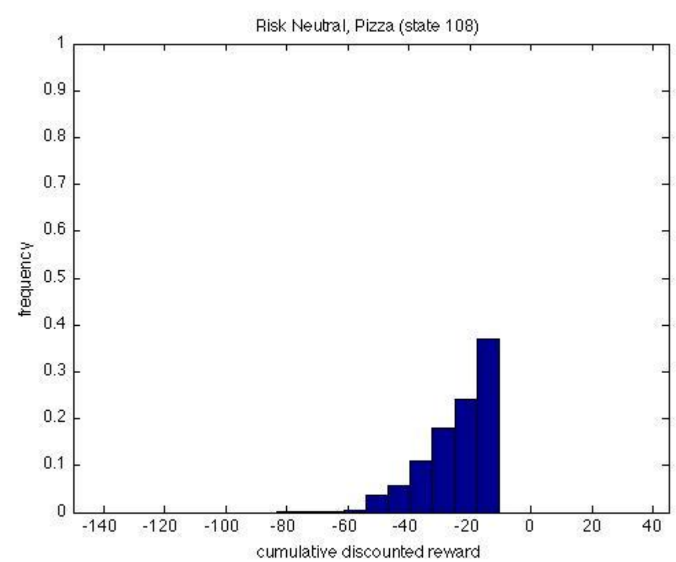
1 4 9 10



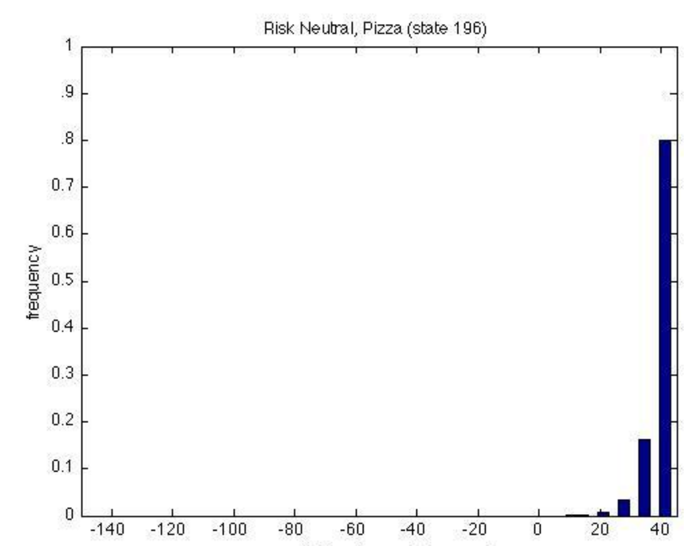
1 1,1



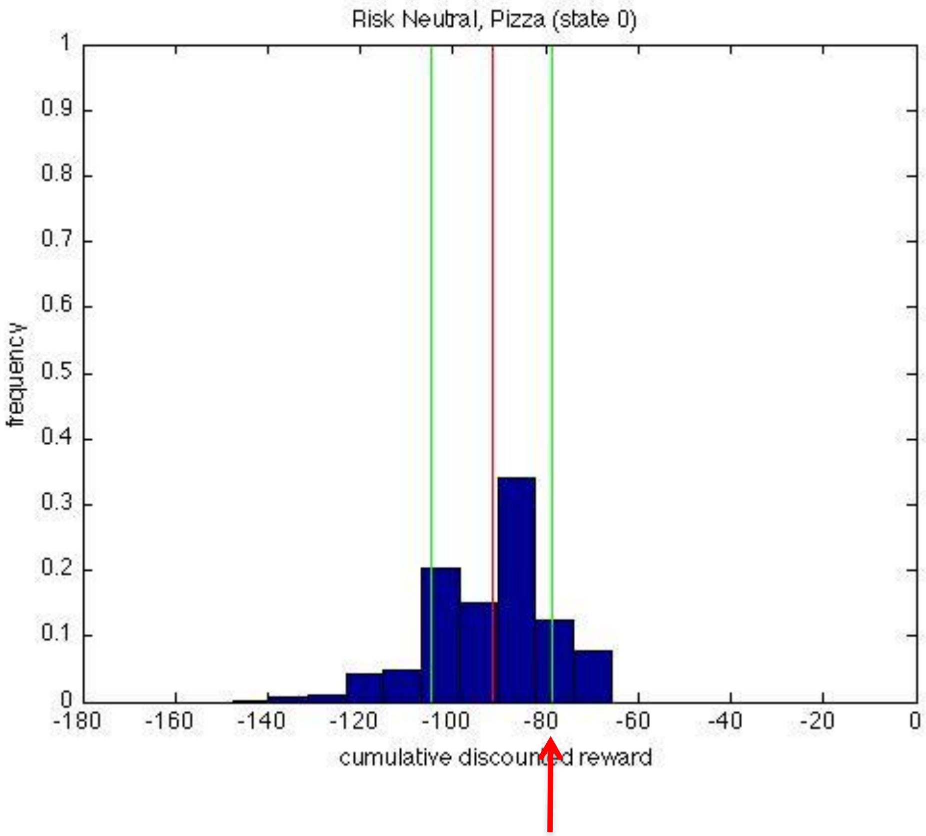
6 6,4

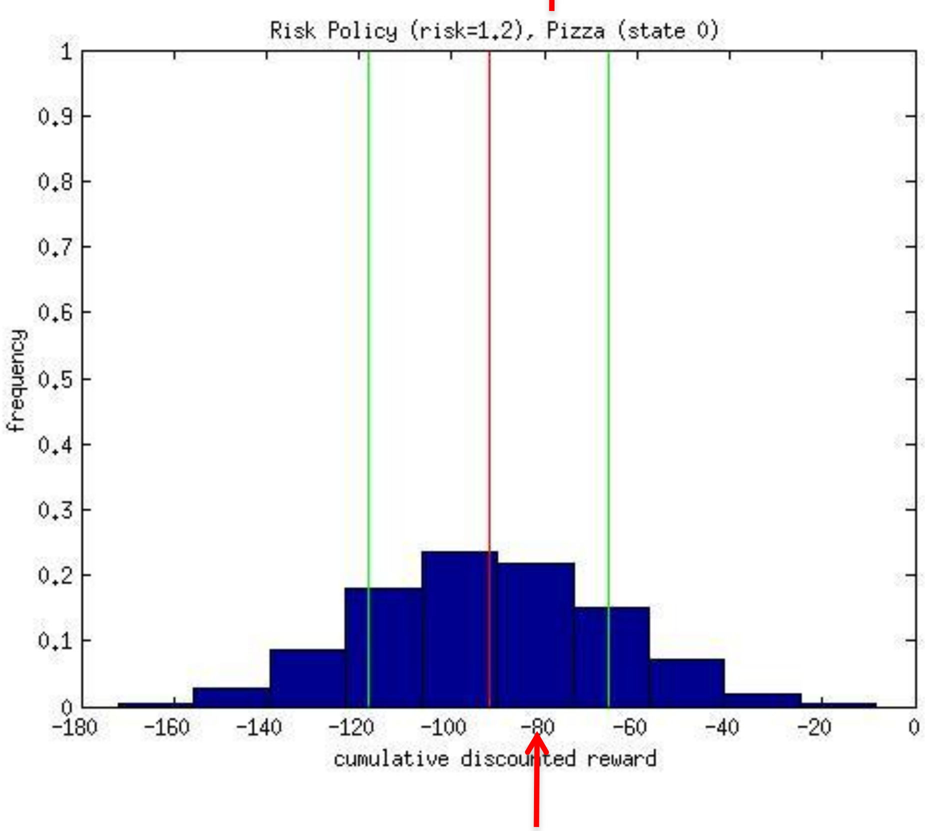


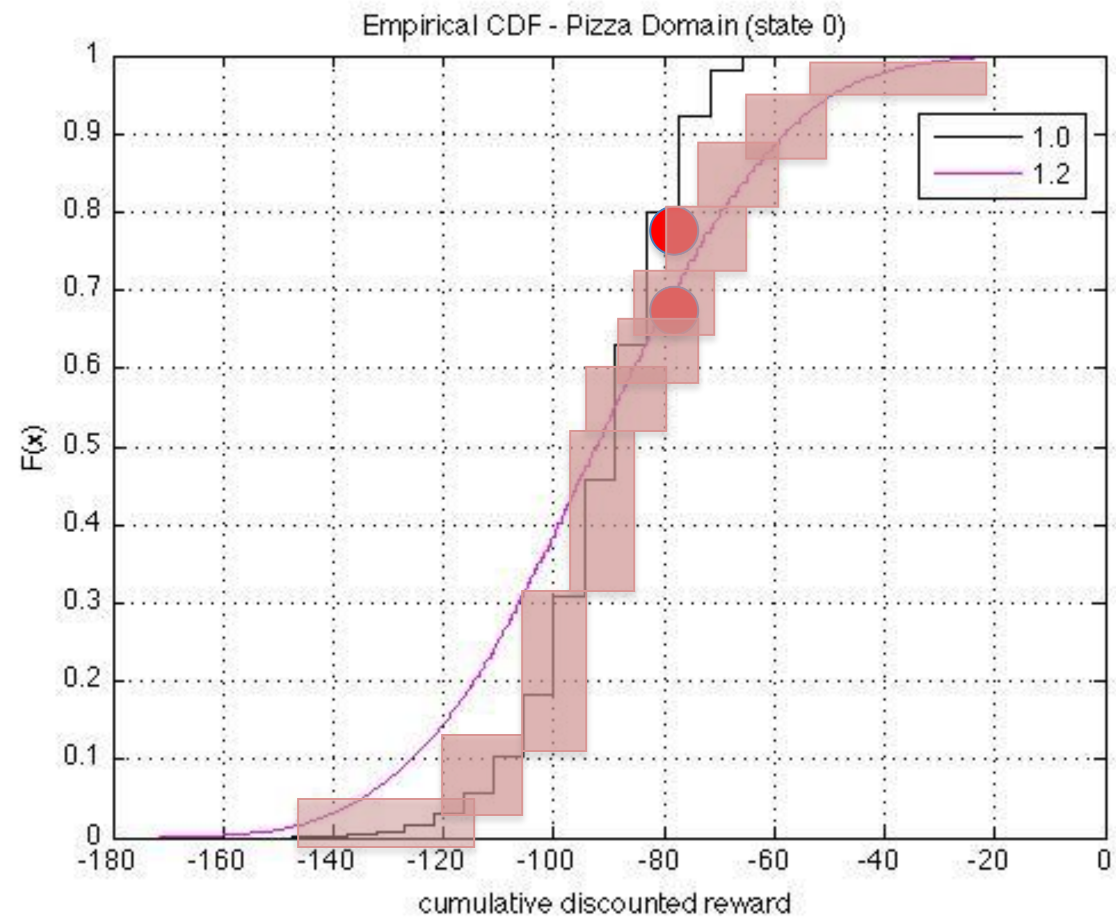
10 10,9



Example Distributions for Different Risk Attitudes







Approach

Offline

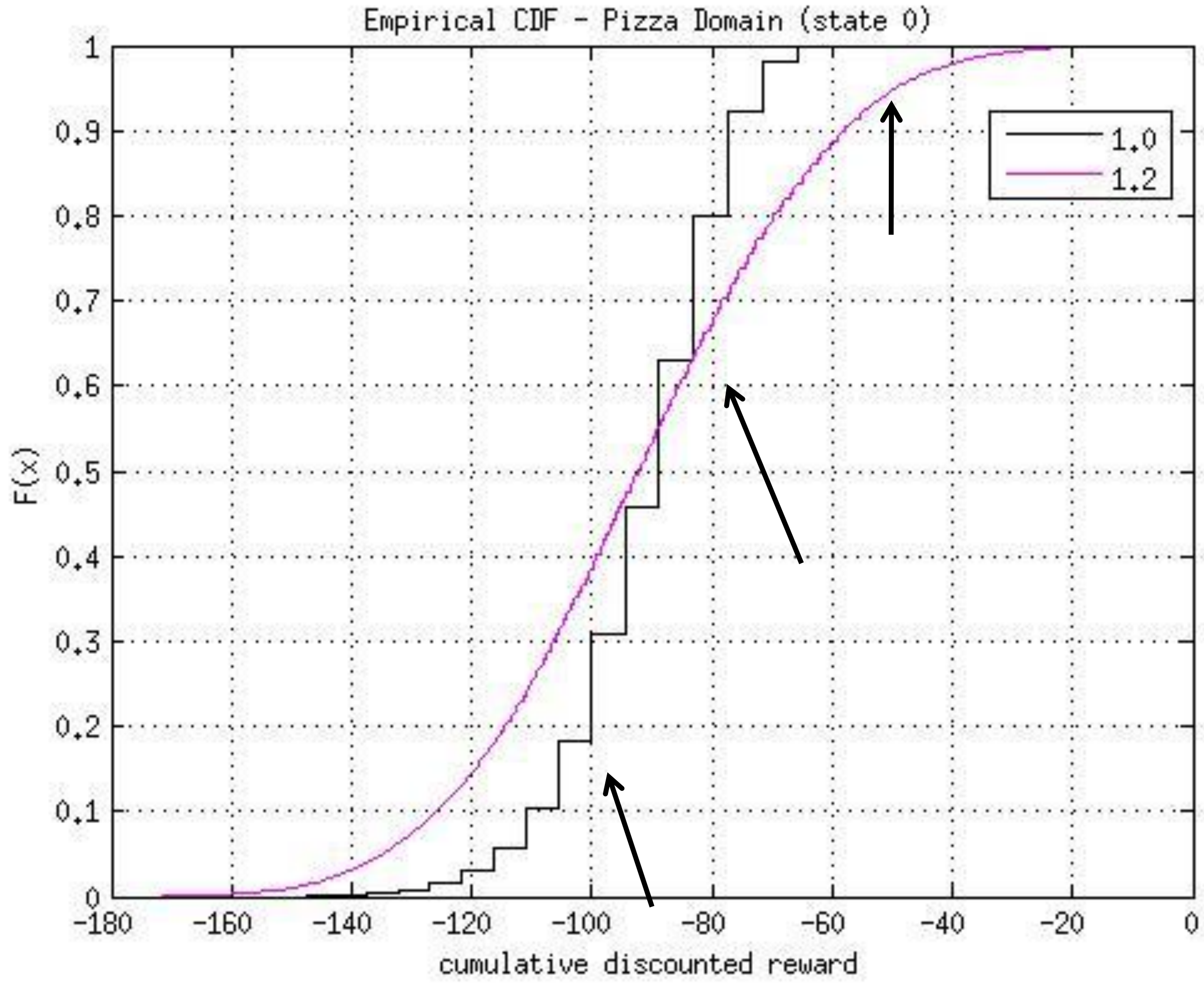
- **Generate different policies**: policies of varying risk attitude.
- **Estimate the reward distributions.**

Online

- **Switch between policies**: Calculate the maximum probability of being over a threshold at each time step based on the current cumulative (discounted) reward.

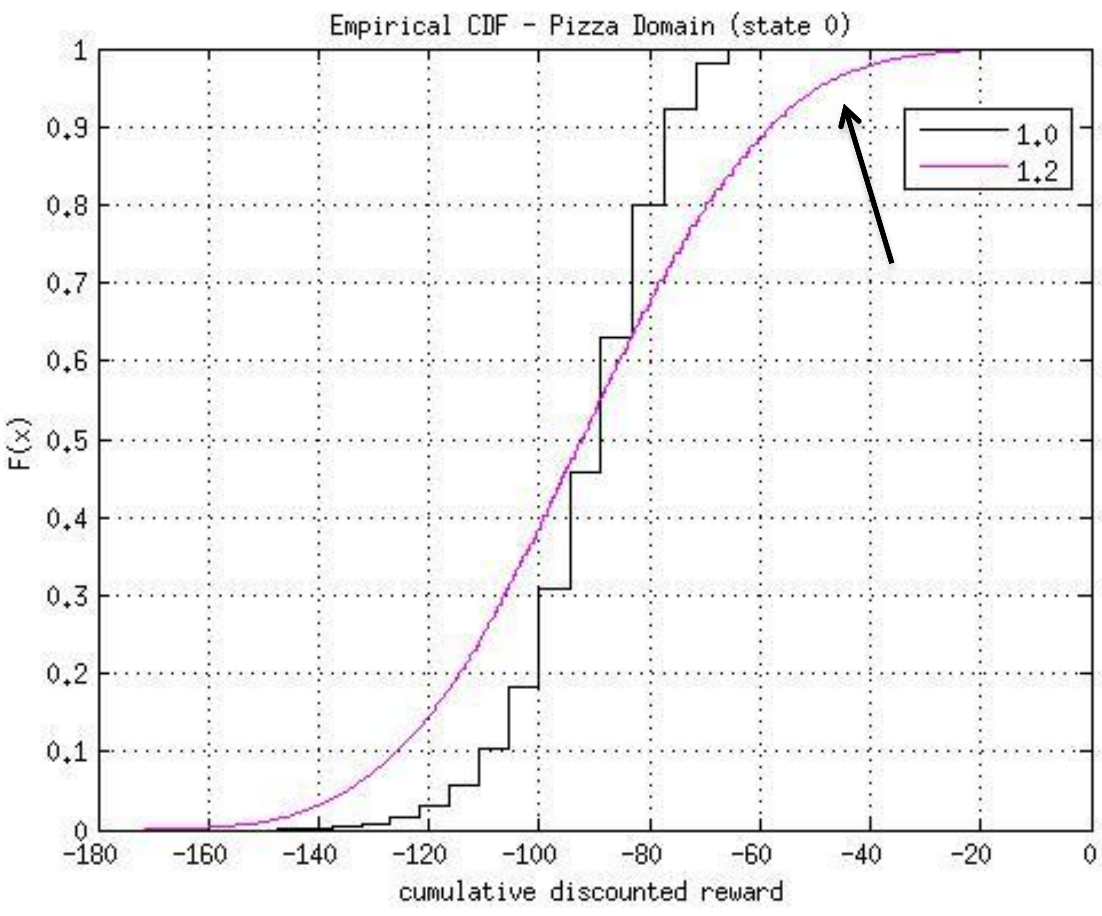
Switching Criteria

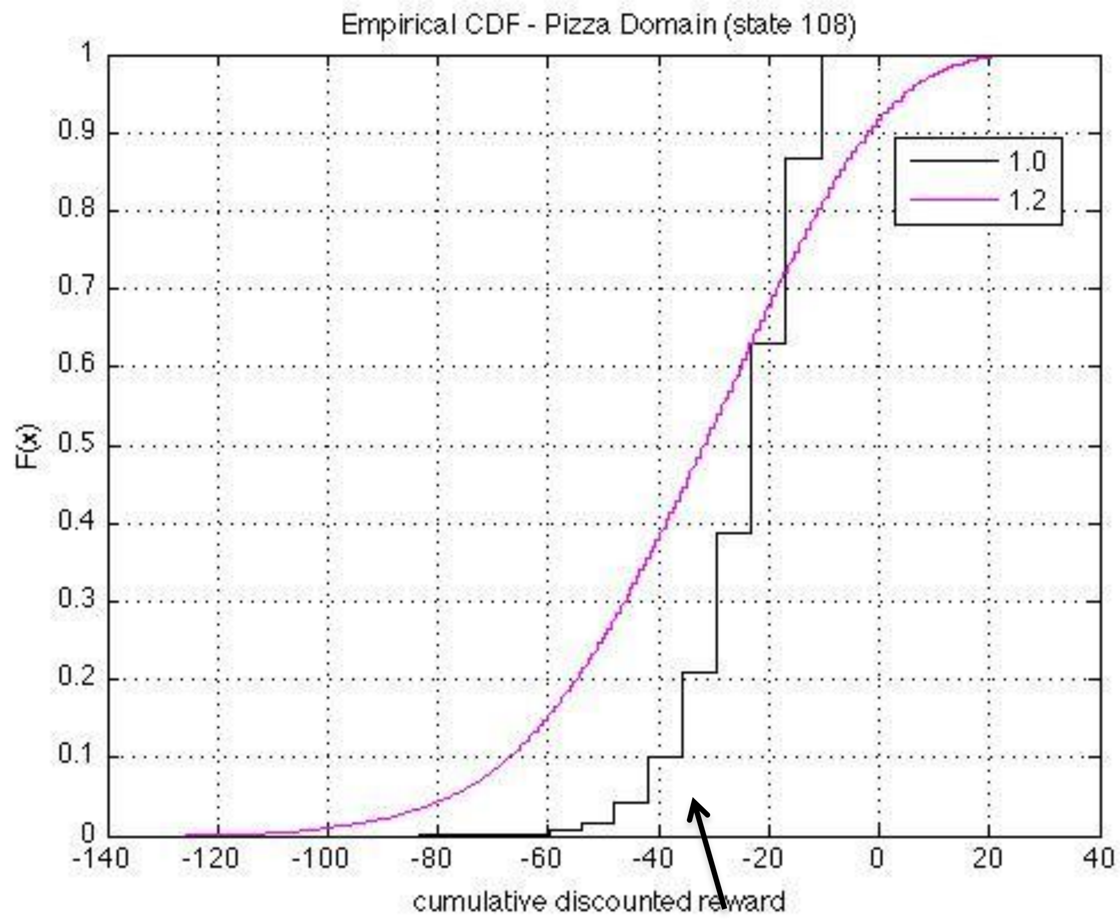
- Use the CDF to know the probability of being greater than the threshold. $1 - F(x) = \int_x^{\infty} f(t)dt = P(V > x)$



Switching Criteria

At each time-step, for that state $s_1 -> a_1 -> s_2 -> a_2$





Results

Switching Shows Improvement, Pizza Domain

- Execute 10,000 runs in ***original*** MDP
  - Same start state every time.
  - Risk-neutral *vs* switching (with risky policy δ=1.2).

Threshold = -100;

Fails to Exceed the Threshold	
Risk Neutral Fails:	3120
Switching Fails:	2166

Fails 9.5% less using switching strategy; Reduces losses by 30.6%

Threshold = -70;

Fails to Exceed the Threshold	
Risk Neutral Fails:	8026
Switching Fails:	5790

Fails 22.4% less using switching strategy; Reduces losses by 27.9%

Augmented State - Pizza Domain

- Add cumulative reward to the state, no discounting.
  - States go from 200 ->30,200

Augmented State	**Risk-Variant Switching**
Offline Time per policy	
Solve policy: 18 hours	Solve policy: < 1min
	Gen rew dist: 5-10 min
	Constr CDF: 1 min
Total: 18 hours	Total: 12 min * 2 policies = 24 min
Execution Time	
.015s	Eval Switch : .02s

Pros: Performs better, closer to optimal.

Cons: Large planning time, and must re-generate the policy per threshold.

Comparing Augmented State to Switching

- Execute 10,000 runs in ***original*** MDP
  - Same start state every time.
  - Augmented Larger State Space *vs* switching (with risky policy δ=1.2).

Threshold = -70;

Fails to Exceed the Threshold	
Risk Neutral Fails:	7946
Switching Fails:	6945
Augmented State Fails:	6256

Augmented state fails 16.9% less than risk neutral;
Reduces losses by 21.2%

Augmented state fails 6.8% less than switching;
Reduces losses by 9.9%

Application to a Mario Domain

[redacted]

Switching Shows Improvement, Mario Domain

- Execute 1,000 runs using ***learned MDP*** for dynamics
  - Same start state every time

Threshold = 100

Fails to Exceed the Threshold	
Risk Neutral Fails:	919
Switching Fails:	738

Fails 18.1% less using switching strategy; Reduces losses by 19.7%

- Execute 1,000 runs using ***Infinite Mario Simulator***
  - 1,000 different worlds
  - Switching based on discounting of macro actions

Threshold = 30

Fails to Exceed the Threshold	
Risk Neutral Fails:	838
Switching Fails:	821

Fails 1.7% less using switching strategy; Reduces losses by 2%

Future Work

- Expand to also switch with conservative policies.
- Implement in real robot domains
  - Multiple service robots
- Extend to more robustly handle situations where the model does not reflect reality.

Conclusion

- Demonstrated a general algorithm that allows an agent to switch between risk-sensitive policies to exceed a threshold.
  - Reason about complete reward distribution
  - Algorithm saves on planning time.
- Showed improved performance over risk-neutral policies.
- Good for domains where want to take risks, resulting in a higher cost of losing, for the increased chance of winning.

Takeaway

- General Algorithm:
  - Accepts any collection of different policies for an agent to employ.
  - Switching strategy that chooses the next ‘best policy’ based on some criteria.

Tradeoff some computation and somewhat less optimality for significant savings in planning time.

Questions?

Acknowledgements

- Advisor: Reid Simmons
- AFOSR grant #FA2386-10-1-4138
- Sven Koenig and Yaxin Liu (risky policy transformations)
- John Laird and Shiwali Mohan (Super Mario domain simulator)