Incremental Reinforcement Learning for Autonomous Agents

Kevin R. Dixon

Ph.D. Qualifying Exam

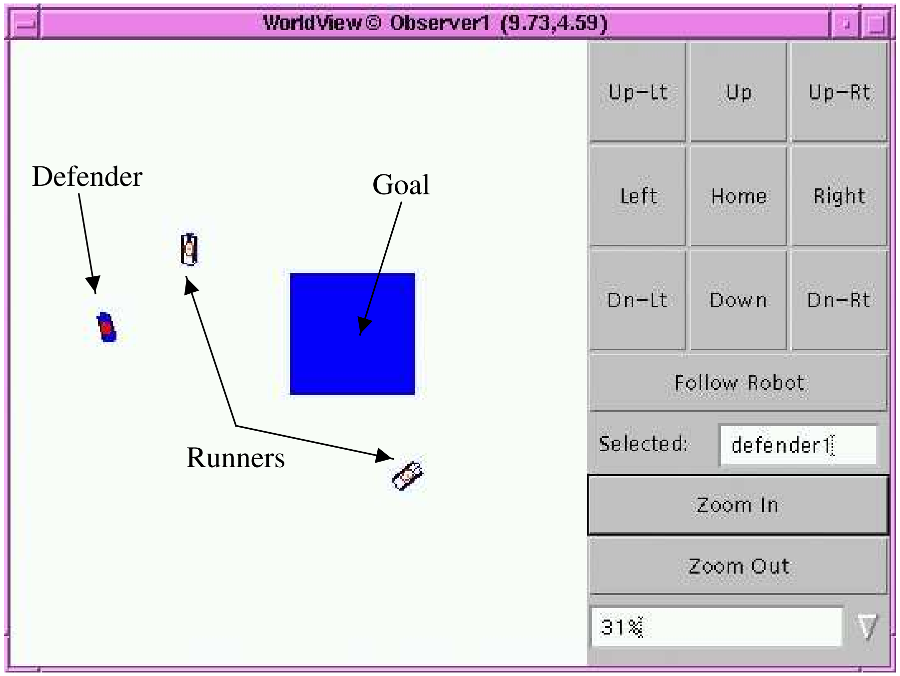
December 13th, 1999

Research Goals

- Use prior knowledge and previously learned information to improve the performance of a learning agent.
- The method should also increase the engineering insight of the learning problem.
- Given a large task, look into “bag of tricks” and use what the agent already knows.

Problem Description

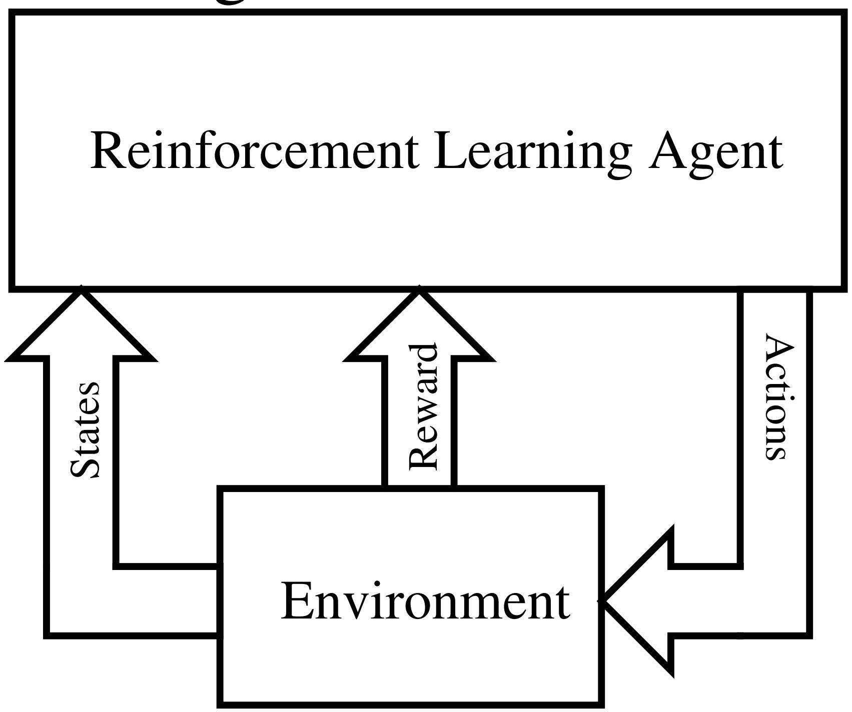
- Play a game where two robots try to go to a goal and another robot tries to stop them.



- Learn "good" strategy for Runners quickly, not necessarily the best possible.

Reinforcement Learning

- Agent learns to correlate states and actions with external reward signal.



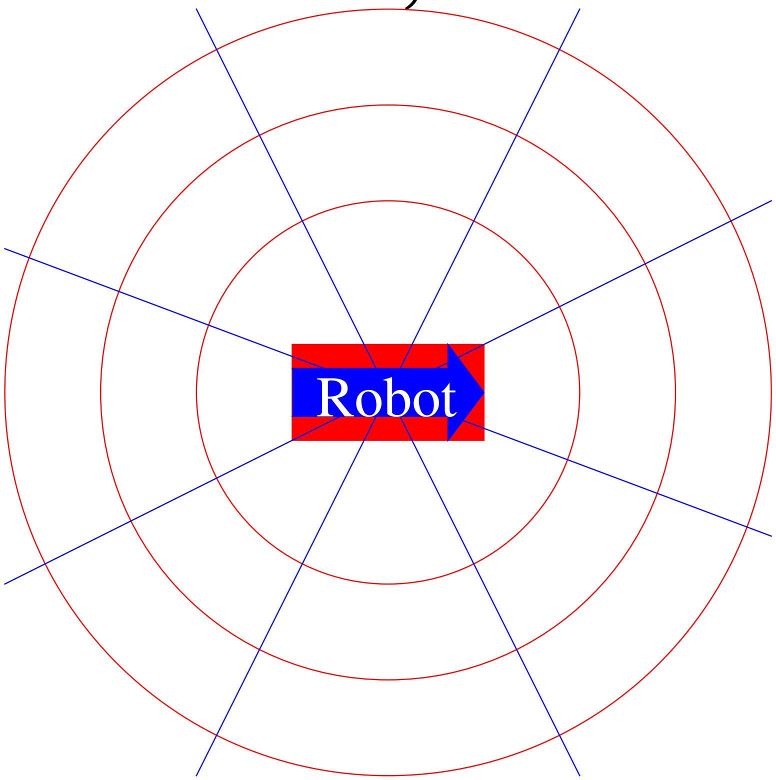
- No model of environment or agent required.
- Learning takes a *long* time.

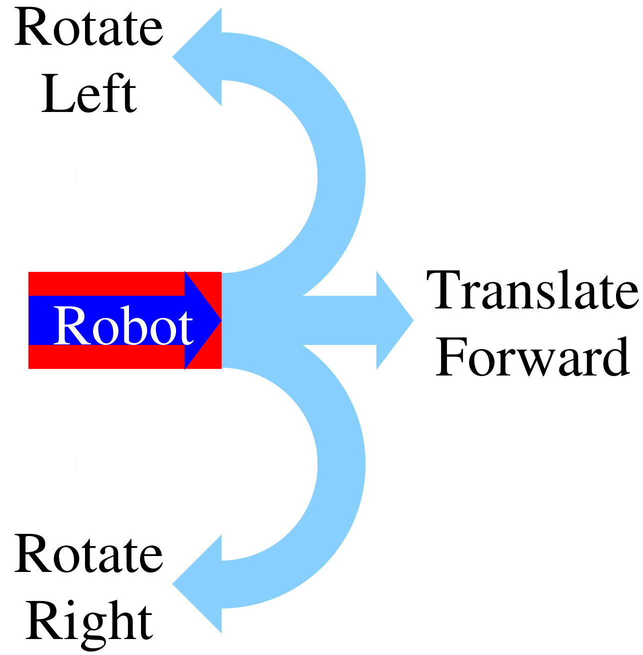
Reinforcement Learning

- Finite set of *states*, S, and *actions*, A, and external reward signal r: $S \times A \times S \rightarrow \Re$.

- Finds policy, π: $S \rightarrow A$, that maximizes expected sum of discounted future rewards through trial-and-error.

States, Actions, and Rewards





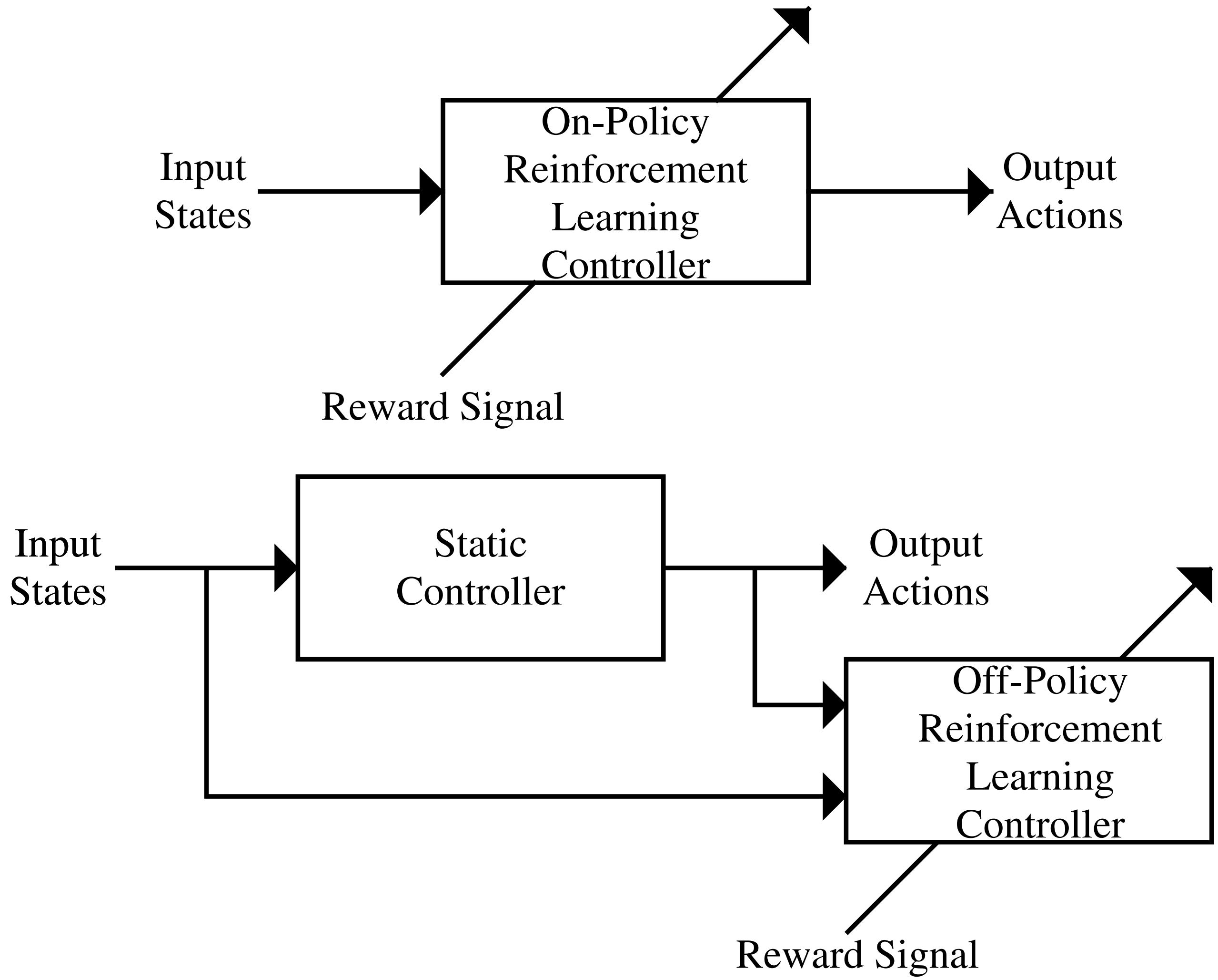
- Two other robots, one goal: $2^{18} = 262{,}144$ states.
- Rewards:

  +1 for reaching goal

  -1 for being “tagged” or moving too far from goal

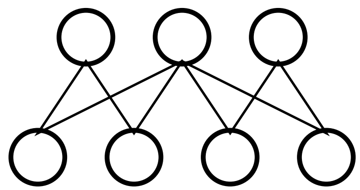
Reinforcement Learning Controllers

- RL Agents have on-policy or off-policy controllers:



Exploration in Reinforcement Learning

- Neural Networks are normally used to represent the controller.



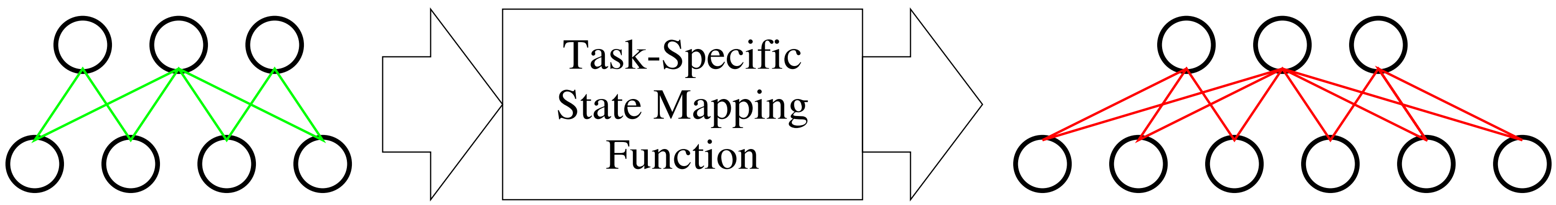
- Data is often too expensive to generate or the state-space is too large to explore sufficiently.
- Thus, exploration may need to be directed.

Methods of Exploration

- Almost Greedy Algorithms
  - Take random actions from time-to-time. [Watkins 89]
- Model-based Exploration
  - Build model and use it to reach unfamiliar states. [Moore and Atkeson 93]
- Augmenting Reward
  - Use heuristic to make reward function "smoother". [Mataric 94]
- Prior Knowledge

Transferring Prior Knowledge

- Transferring weights of Neural Network:

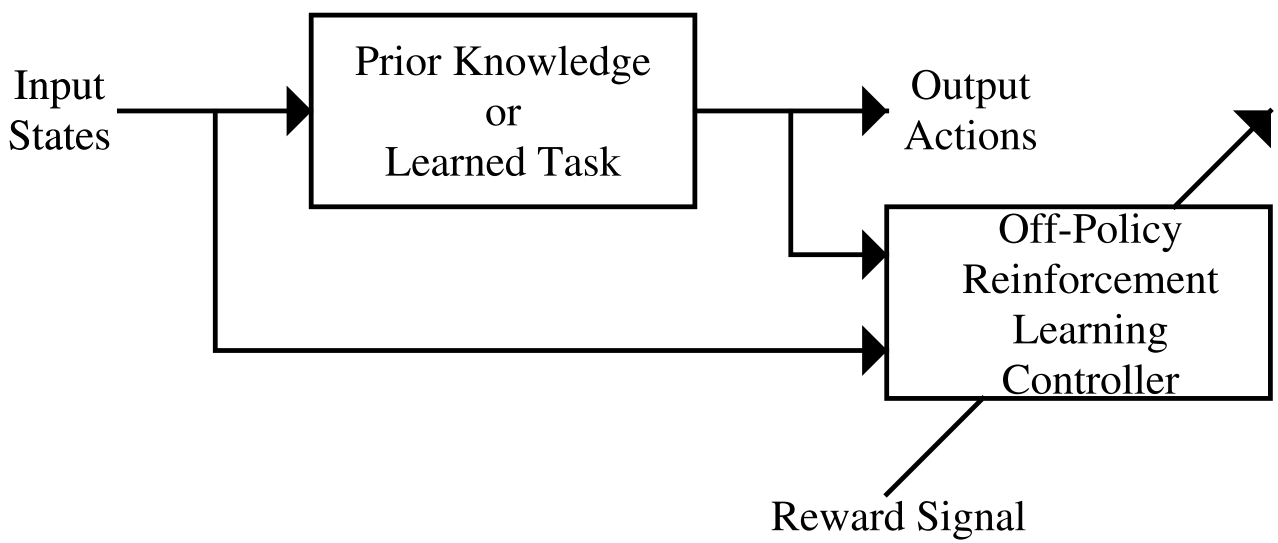


- Susceptible to “Catastrophic Interference”.
  [Sharkey and Sharkey 93]

- Creating State Mapping Function can be extremely difficult and is Task Specific.

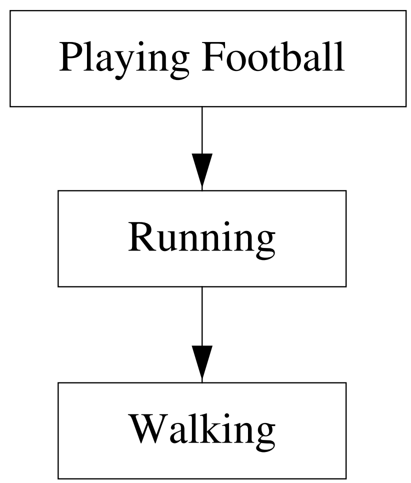
Embedding Prior Knowledge

- Using an Off-Policy RL Controller, we can embed prior knowledge or previously learned sub-tasks in the static controller.



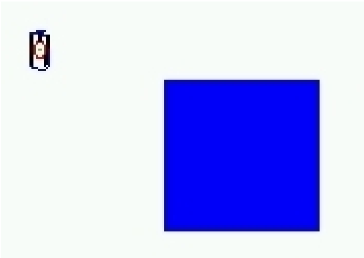
Incremental Reinforcement Learning

- Break a problem up into simpler sub-tasks.
- Solving all sub-tasks may be *better* or *easier* than solving whole problem at once.

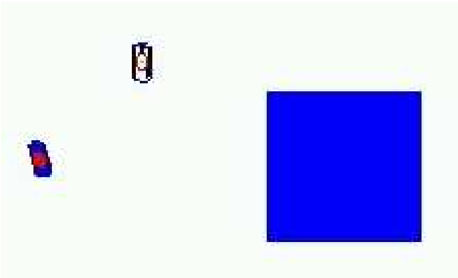


Incremental RL Applied

- First, one Runner versus no Defender (100 games).



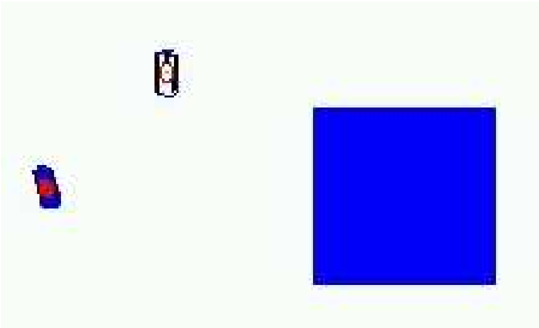
- Next, one Runner versus one Defender (1900 games).



- Finally, two Runners versus one Defender (2000 games).
- Compare Incremental RL versus *tabula rasa* RL.

But…

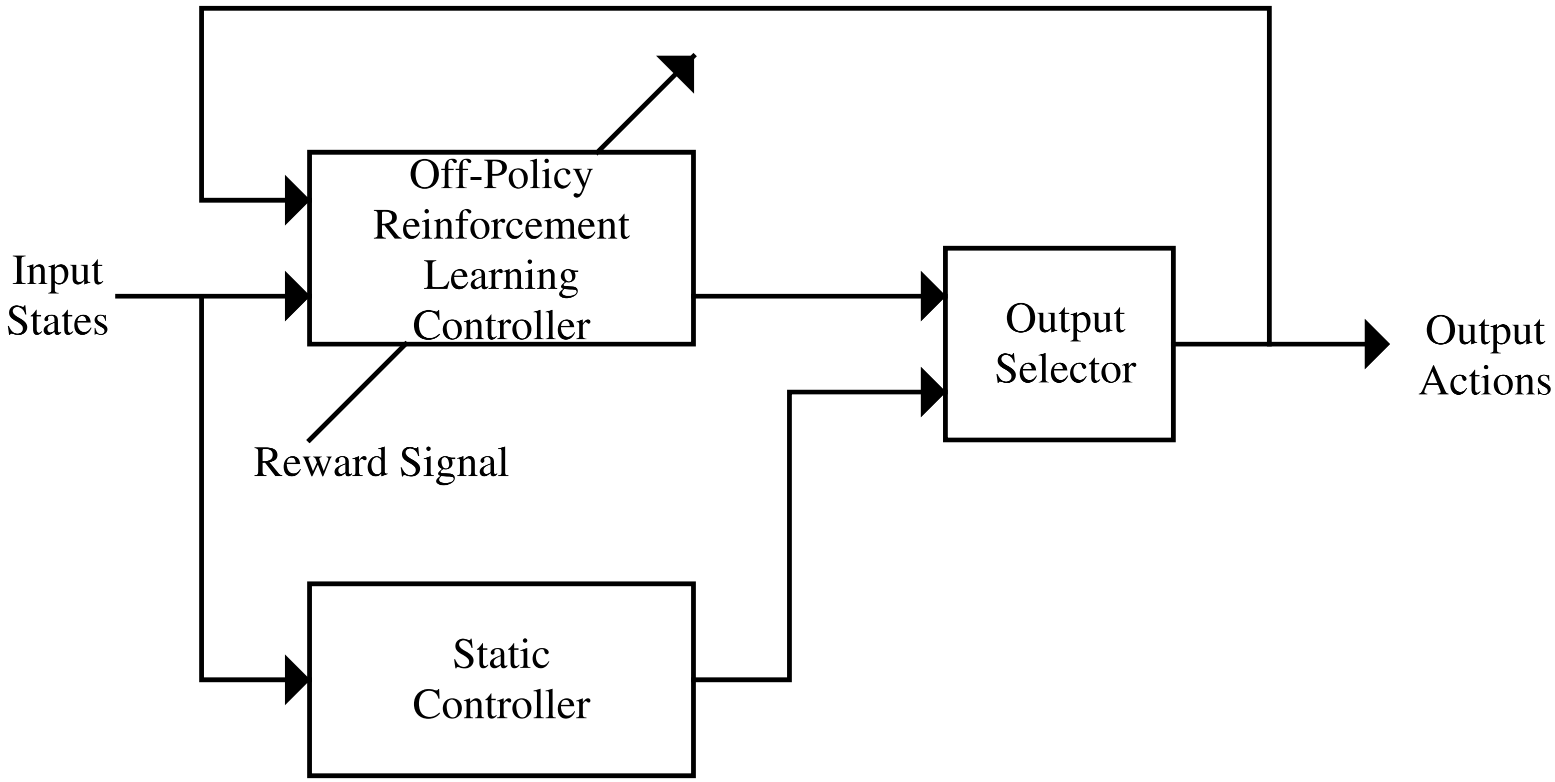
- Using an Off-Policy scheme, this didn't work.



- Runner learned to run away, no matter where it was with respect to the Defender.

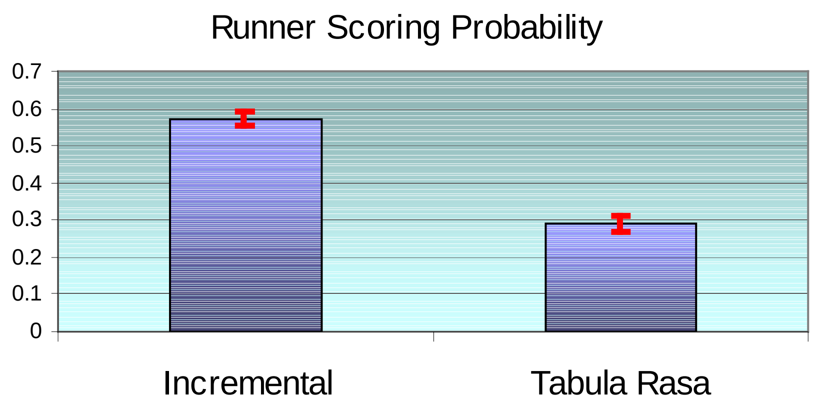
Incremental RL Applied, Take 2

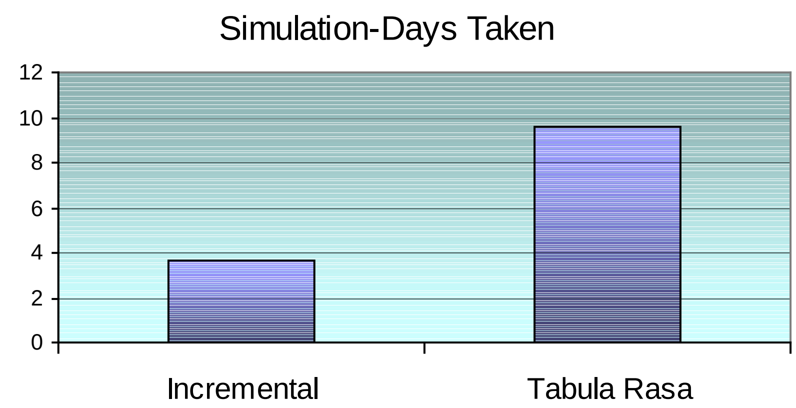
- Modified Off-Policy Controller:



Results

- Incremental method learned task *better* and more quickly than *tabula rasa* teams.





- Better: Play Runners against the Defender.
  - Consider this a Bernoulli trial.
  - Which team has higher probability of scoring.

Conclusions

- Novel scheme for improving performance of RL controllers and embedding prior knowledge.

- Benefits
  - Learning time decreased.
  - Better performance against *tabula rasa* method.
  - Incremental by nature.

- Drawbacks:
  - Problem must be incremental.
  - Increments can be harmful.
  - Difficult to compare performance: apples to water buffaloes.