
Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming

Richard S. Sutton

Presented by Pirooz Chubak

CMPUT 651 Course

October 17th

Outline

- *Dyna* Architecture
 - *Dyna-PI*
 - A Navigation Task
 - Problems of Changing Worlds
 - *Dyna-PI* vs. *Dyna-Q*
 - Changing World Experiments
 - Strengths and Weaknesses of *Dyna*
-

Motivation

- How should a robot decide what to do?
 - ❑ It should plan for each move (**Planning**)
 - ❑ It should plan for all moves and compile its results into a set of rapid *reactions* (**Reactive Systems**)
 - ❑ It should Learn a set of reactions by trial-and-error (**Learning**)
 - ❑ A combination of the above
-

Dyna Architecture

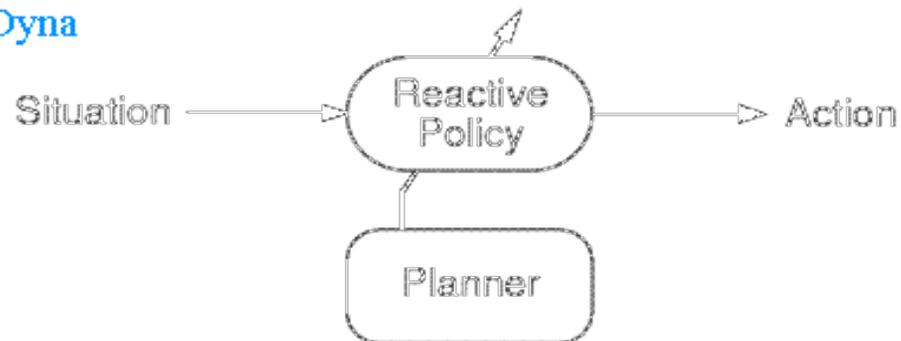
A) Planning



B) Reactive Systems



C) Dyna

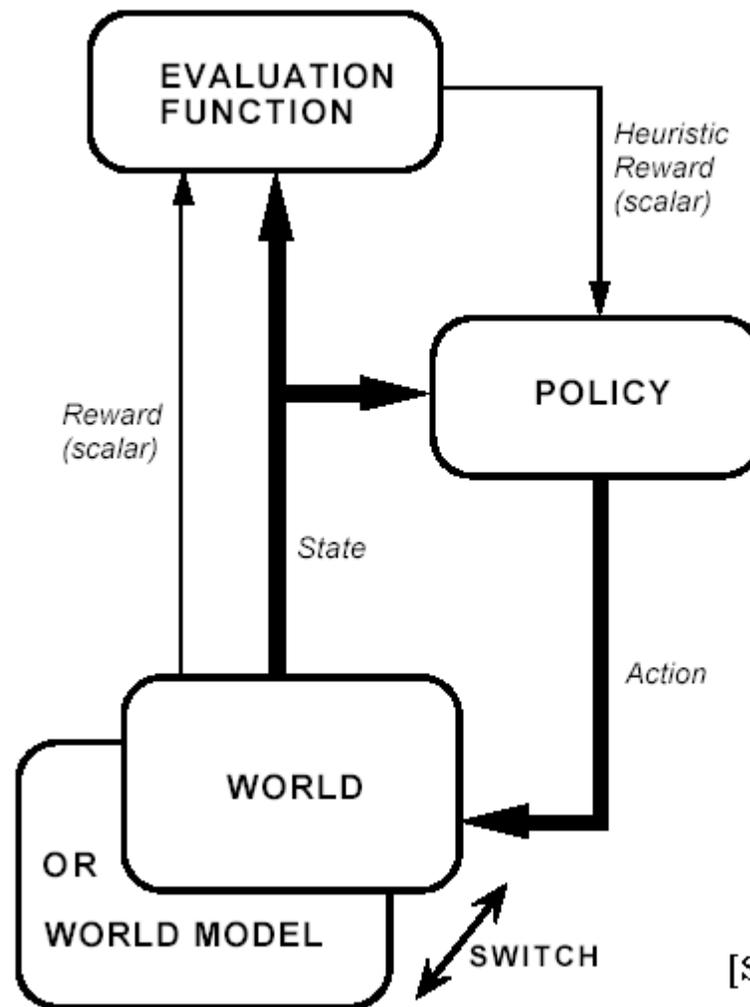


[Sutton, 91b]

Dyna-PI

- Dyna By approximating *policy iteration*
 - Based on dynamic programming policy iteration:
 - Continually **Evaluates** and **Improves** policy
 - Guaranteed to converge to optimal policy on finite MDPs (stochastic environments with a finite set of state and action spaces)
-

Dyna-PI Architecture

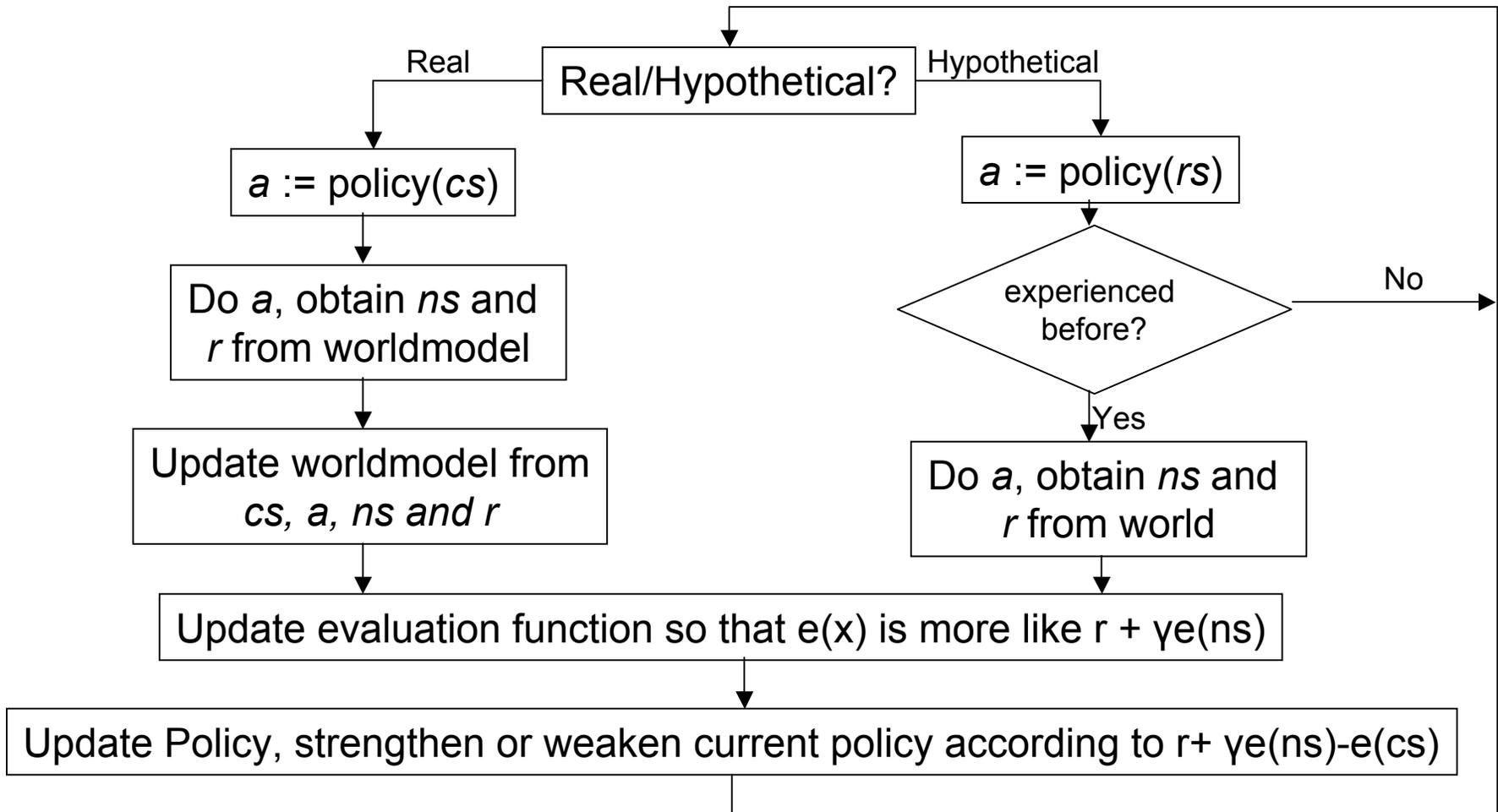


[Sutton, 90]

Dyna-PI

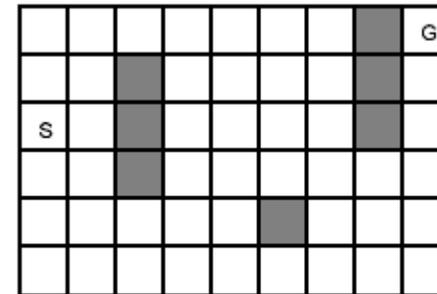
- Fixed Policy (e.g. random policy): a reactive system
 - Policy is continually adjusted by an integrated planning/learning process
 - Planning and learning are integrated at the choice of world/worldmodel for hypothetical/real experiences
-

Dyna-PI Algorithm



A Navigation Task

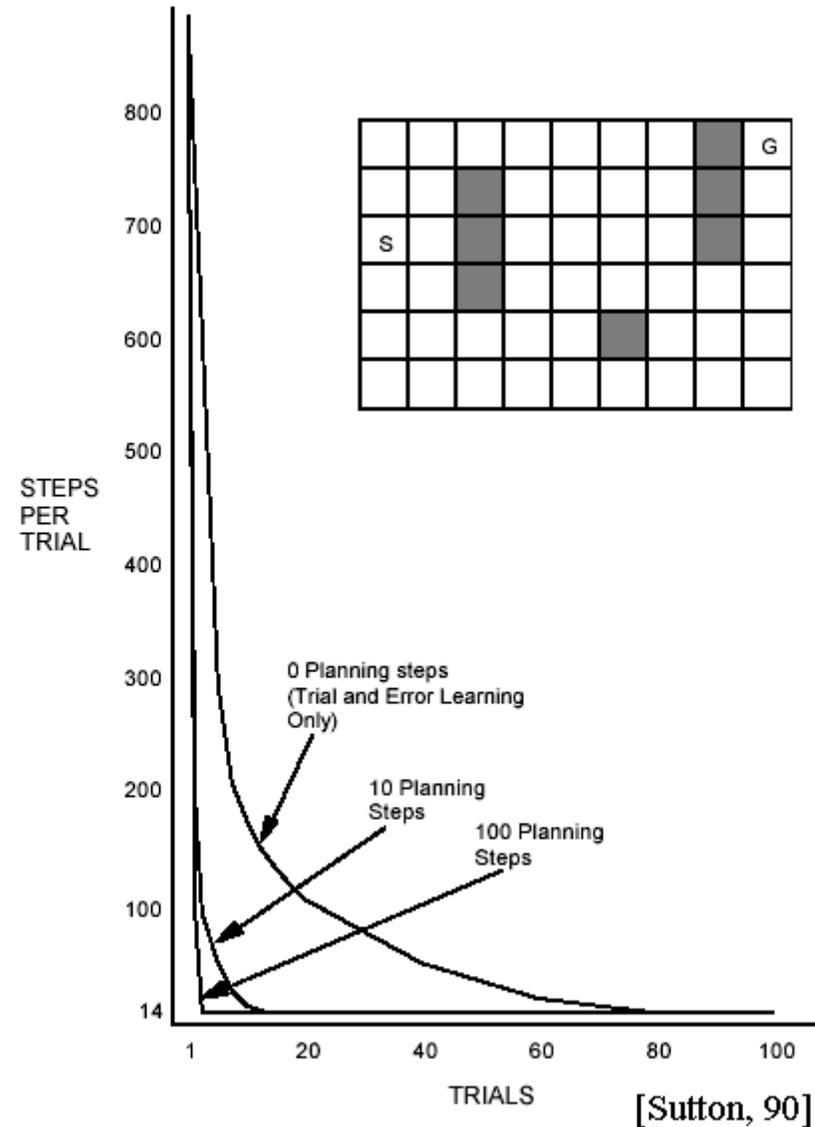
- 6*9 grid
- The system is deterministic but the agent does not know that
- For each experience with the real world, k hypothetical experience were generated with the model (k=0, 10, and 100)
- K=0 includes no planning
- Hypothetical experiences are done only on previously visited states



[Sutton, 90]

A Navigation Task

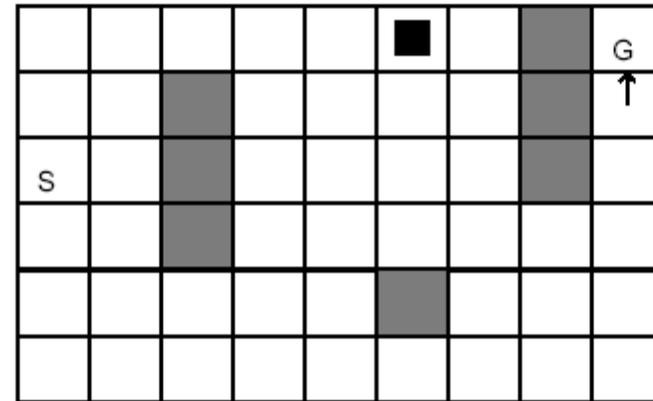
- Even without planning, the agent finds the optimal path in a reasonable number of trials
- With planning the agent finds the optimal solution much faster
- If the cost of real and hypothetical experiences are the same, $k=0$ performs the best!



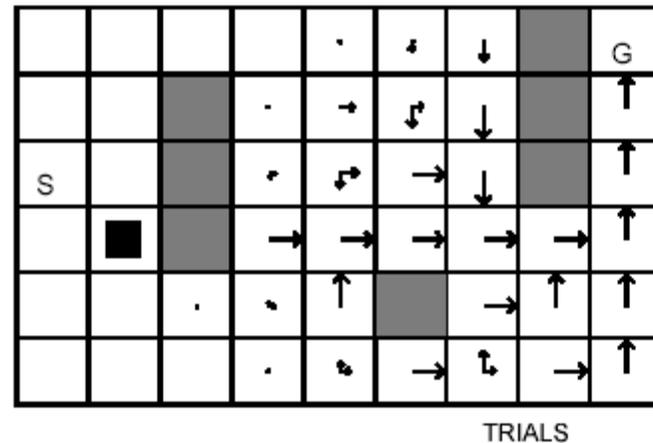
Dyna-PI performance

- Half-way through the second trial
- *First trial*: for $k=0$ and $k=100$ the agent learns only from the last step leading to goal
- *Second trial*: for $k=0$ the agent only learns one step (two steps to the goal) but for $k=100$ the agent learns an extensive policy

WITHOUT PLANNING ($k = 0$)



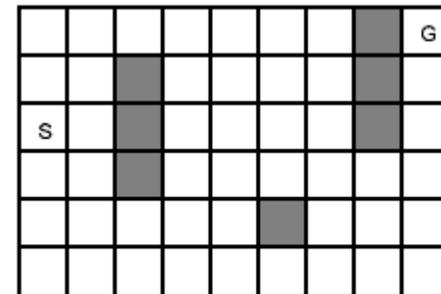
WITH PLANNING ($k = 100$)



[Sutton, 90]

Dyna-PI Implementation

- Policy is a $n(\text{state}) \times n(\text{action})$ table that has an entry w_{xa} for every entry of state x and action a
- Evaluation function $e(x)$ is updated by: $e(x) := e(x) + \beta(r + \gamma e(y) - e(x))$
- actions are selected according to a Boltzman Distribution $P(a|x) = \frac{e^{w_{xa}}}{\sum_j e^{w_{xj}}}$
- Policy is updated according to: $w_{xa} := w_{xa} + \alpha(r + \gamma e(y) - e(x))$
- Initially values and policy table entries have all values of zero
- $B = 0.1$
- $\gamma = 0.9$
- $\alpha = 1000$ ($k=0$) & $\alpha = 10$ ($k=10$ or 100)



[Sutton, 90]

Problem!

- *Dyna-PI* performed well on finding an optimal path, but may find two problems with changing worlds
 - *Blocking problem*: if a barrier is added that blocks the optimal path
 - *Dyna-PI* uses the previously learned values hundreds of times
 - *Shortcut problem*: if a barrier is removed that permits a shorter path from start to goal
 - *Dyna-PI* never explores to find the new optimal path
 - *Dyna-Q+* solves the problem of changing worlds by adding more exploration
-

Dyna-Q

- Simpler structure than Dyna-PI
 - Combination of Dyna Architecture and Watkins Q-learning
 - Uses only one memory structure for evaluation function and policy, called Q
 - Uses only one rule to update evaluation function and policy
-

Dyna-PI vs. Dyna-Q

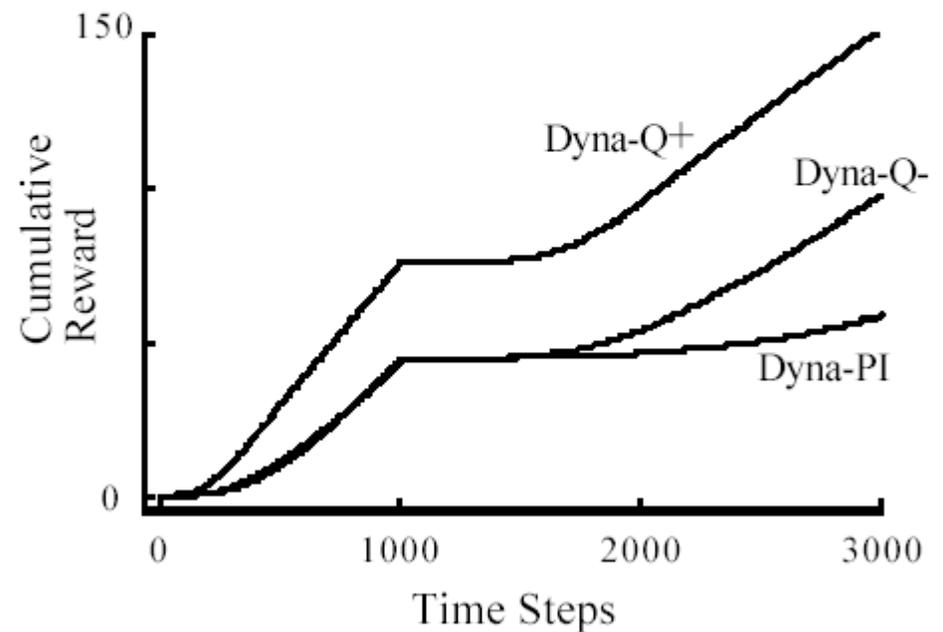
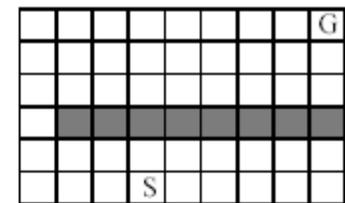
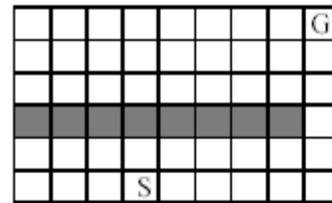
	Dyna-PI	Dyna-Q
Policy update rule	$w_{xa} := w_{xa} + \alpha(r + \gamma e(y) - e(x))$	$Q_{xa} := Q_{xa} + \beta(r + \gamma e(y) - Q_{xa})$
Evaluation function update rule	$e(x) := e(x) + \beta(r + \gamma e(y) - e(x))$	$Q_{xa} := Q_{xa} + \beta(r + \gamma e(y) - Q_{xa})$
Action selection according to	$P(a x) = e^{w_{xa}} / \sum_j e^{w_{xj}}$	$P(a x) = e^{\alpha Q_{xa}} / \sum_j e^{\alpha Q_{xj}}$

Dyna-Q+

- Dyna-Q with exploration bonus
 - Uses a new memory structure n_{xa} that is the number of time steps elapsed since a was tried in x in a real experience.
 - For real experiences the policy is to select the action a that maximizes $Q_{xa} + \epsilon \sqrt{n_{xa}}$
-

Blocking problem

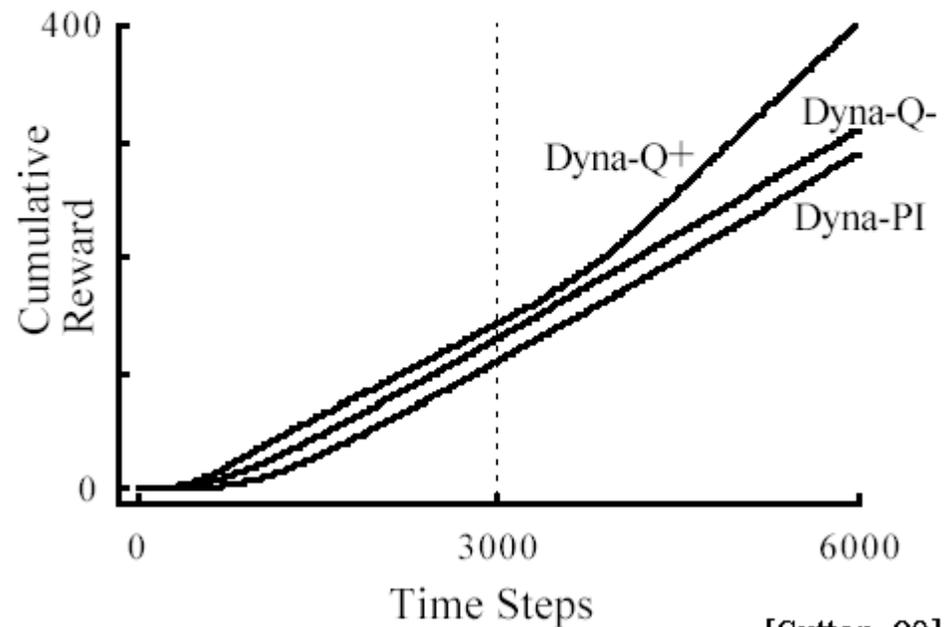
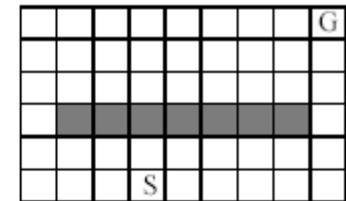
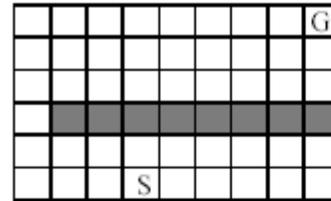
- $K=10$
- Average results over 50 runs
- After 1000 time steps short path is blocked and a longer path is opened



[Sutton, 90]

Shortcut problem

- $K=10$
- Average results over 50 runs
- After 3000 time steps a barrier is removed and a short path is opened



[Sutton, 90]

Pros and cons of Dyna

■ Pros

- ❑ Can be applied to stochastic environments
- ❑ Planning can be done on incomplete, changing and probably incorrect world models constructed through learning

■ Cons

- ❑ Dyna needs to store all the experienced state-action pairs for planning
 - ❑ Results shown here are very limited, state and action spaces are small and finite
 - ❑ It is not discussed in this paper why *Dyna-Q+* performs better than *Dyna-Q* on non-changing environments
-

Applicability to project

- Our project, with alborz is a Real-time multi-agent path finding for RTS games
 - We would like to enhance the memory consumption of the current state of the art multi-agent path finding method
 - Current method reserves expected positions in time dimension for every agent, so it consumes a lot of memory
 - Dyna is not applicable to our project, we have to implement a reactive system
-

References

- [Sutton, 90] Sutton, R.S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216-224, Morgan Kaufmann. Also appeared as "Artificial intelligence by dynamic programming," in *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, pp. 89-95.
 - [Sutton, 91b] Sutton, R.S. (1991). Dyna, an integrated architecture for learning, planning and reacting. *Working Notes of the 1991 AAAI Spring Symposium on Integrated Intelligent Architectures and SIGART Bulletin 2*, pp. 160-163.
 - [Sutton, 91a] Sutton, R.S. (1991). Planning by incremental dynamic programming. *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 353-357, Morgan Kaufmann
 - [Sutton and Barto, 98] Sutton, R.S., Barto, A.G. (1998). Reinforcement Learning: An Introduction. MIT Press
-