Hierarchical Reinforcement Learning and Soar

Shelley Nason

Soar Workshop 2006

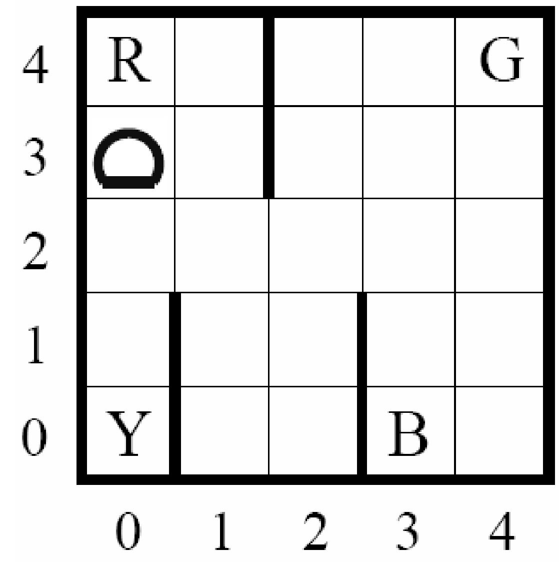
Soar-RL: Soar with an architectural RL mechanism

- Reinforcement learning:
  Learning how to act so as to maximize the expected cumulative value of a (numeric) reward signal
- In Soar terminology, RL learns operator comparison knowledge
- Uses:
  - Learning without model of operator effects
  - Learning in nondeterministic domains
  - Learning in non-goal-based tasks
- Limitation of old work:
  RL incompatible with Soar impasses

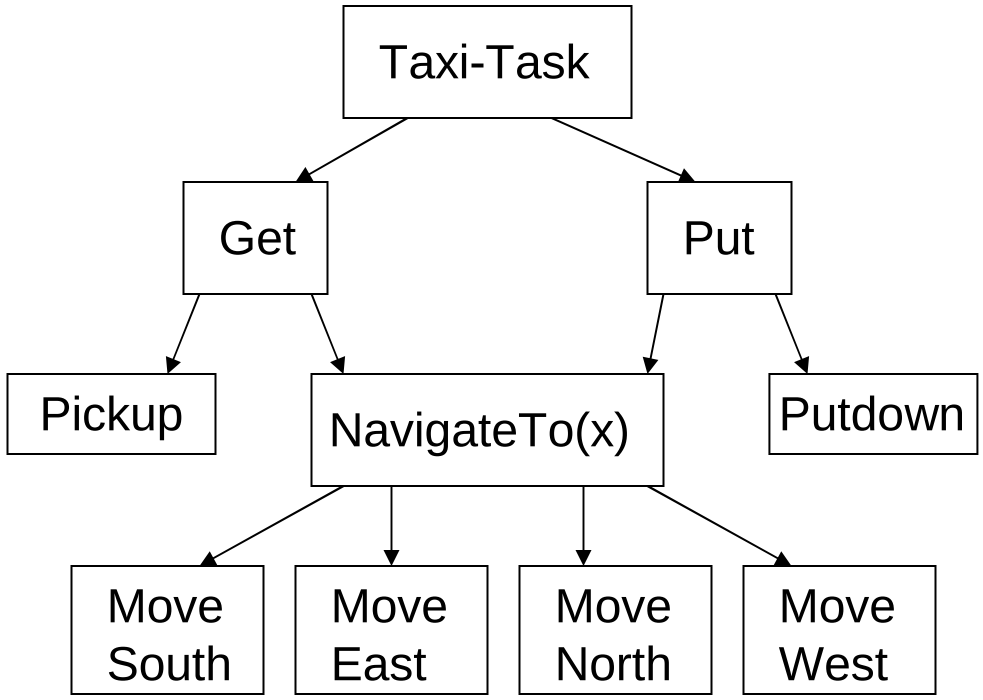
RL and Soar impasses

- Learning impasses (i.e., tie impasse)
  - A place for RL? Interesting question.
  - Not what this talk is about
- Michigan-style goal-stack (op no-change)
  - Hierarchical task decomposition
    → Hierarchical RL
  - Hierarchical RL: RL with temporally-extended actions,
    - Actions of variable timespan
    - Actions whose expected reward, timespan, and next state depend on subtask policy
  - Impassed operator → temporally-extended action

Example Hierarchical Task: Taxi Domain

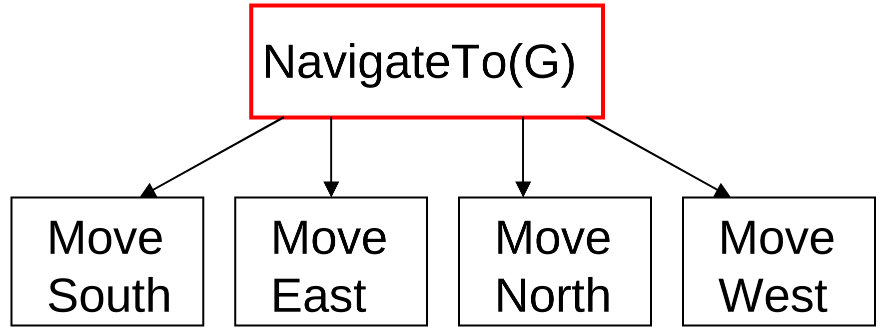


- A higher-level task: Navigate to **G**
- A lower-level task (primitive action): Move East



Hierarchical RL:

Higher-level operator



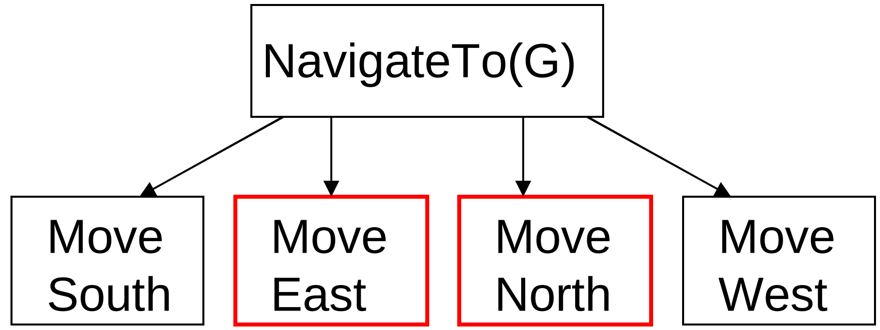
- Higher-level operator (i.e., NavigateTo(G))
  - Treated as temporally extended action.
  - Value includes:
    - Rewards received while operator selected
    - Value of next state, discounted by length of time operator was selected

$r_1 + \lambda * r_2 + \lambda^2 * r_3 + \lambda^3 * r_4 + \lambda^4 * \text{Pred(Putdown)}$

S1 NavTo(G) NavTo(G) NavTo(G) NavTo(G) Putdown

S2 MoveEast MoveNorth

Hierarchical RL:
Lower-level operator



- Lower-level operators (i.e., MoveEast)
  - Learning in subtask - try to divorce subtask value function from context
    - Subtask value function does not predict beyond end of subtask
    - Task-specific reward for subtask end state

S1 NavTo(G) NavTo(G) NavTo(G) NavTo(G) Putdown

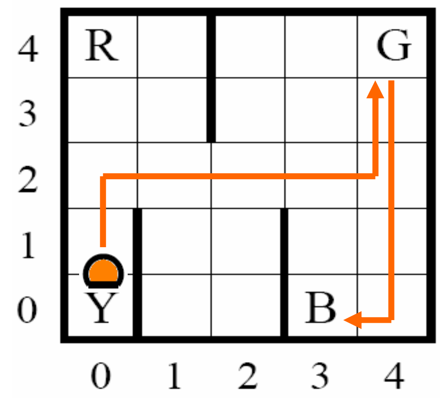
S2 MoveEast MoveNorth

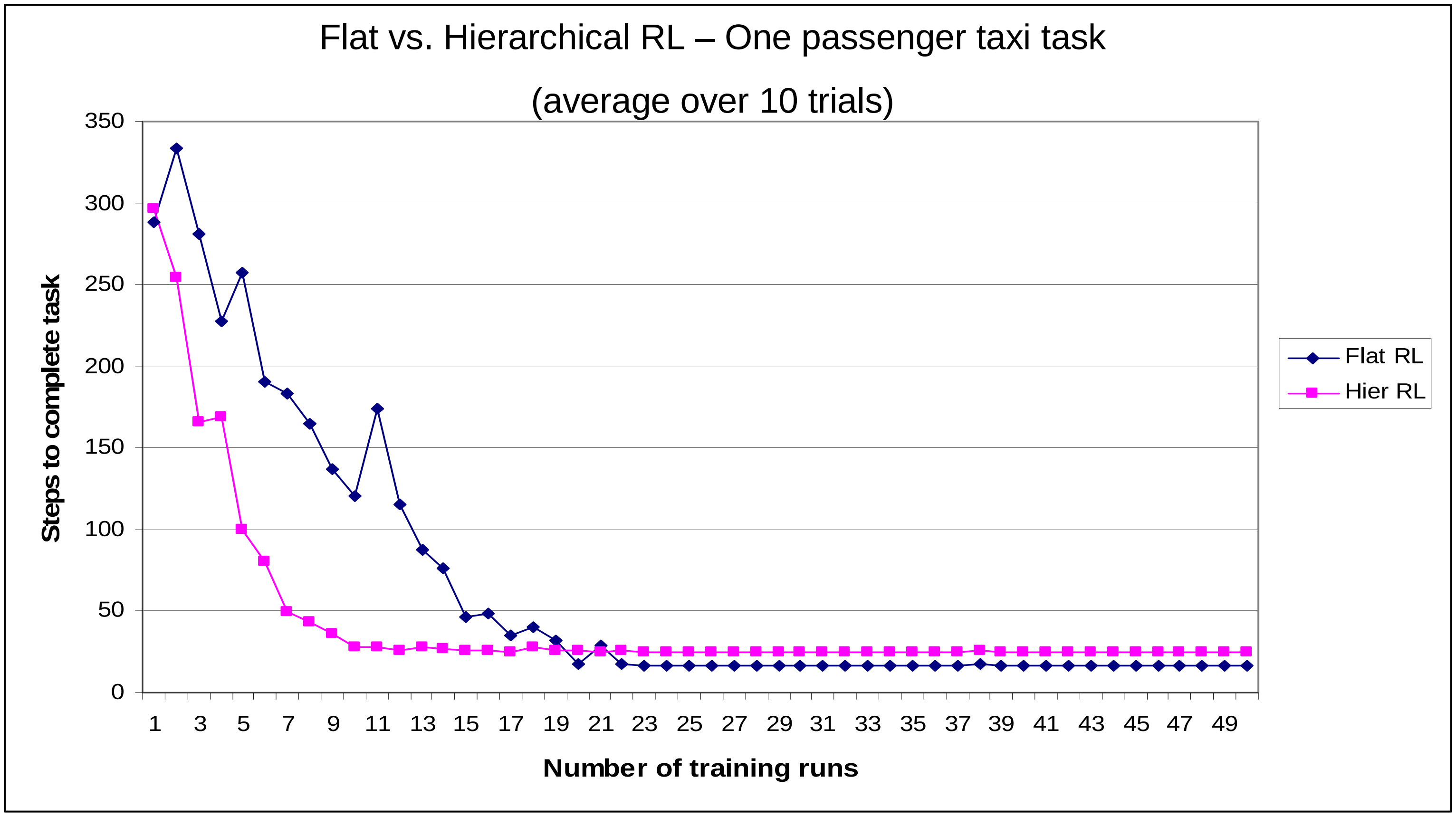
r_2

$r_1 + \lambda$*Pred(MoveNorth)

What is gained?

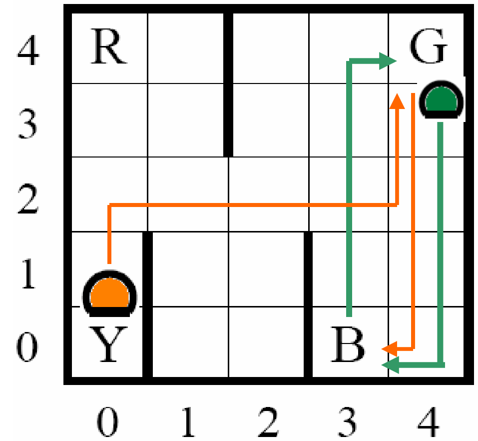
- Faster learning in single task
  – More immediate feedback since rewards received by end of subtask





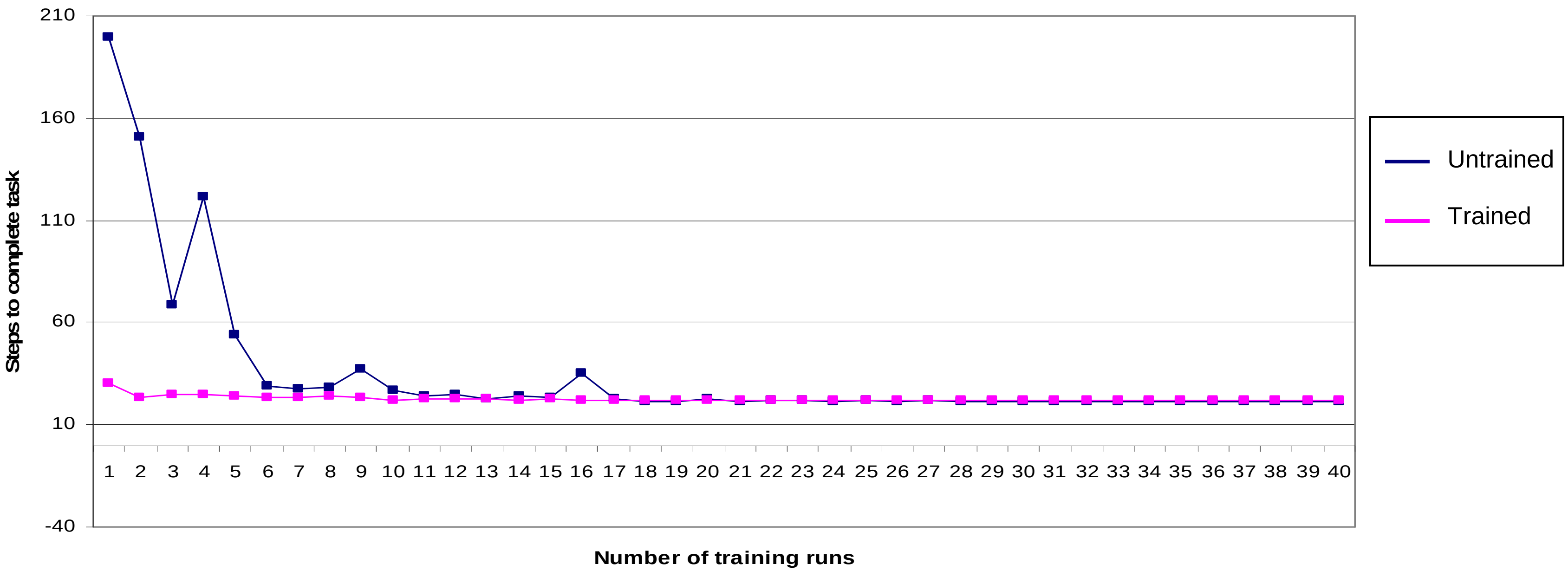
What is gained?

- Transfer of subtask policy between higher-level tasks
  – Made possible by generalizing away higher-level context in subtask value functions



Untrained vs. Trained RL – Transfer one-passenger taxi task

(average over 10 trials)

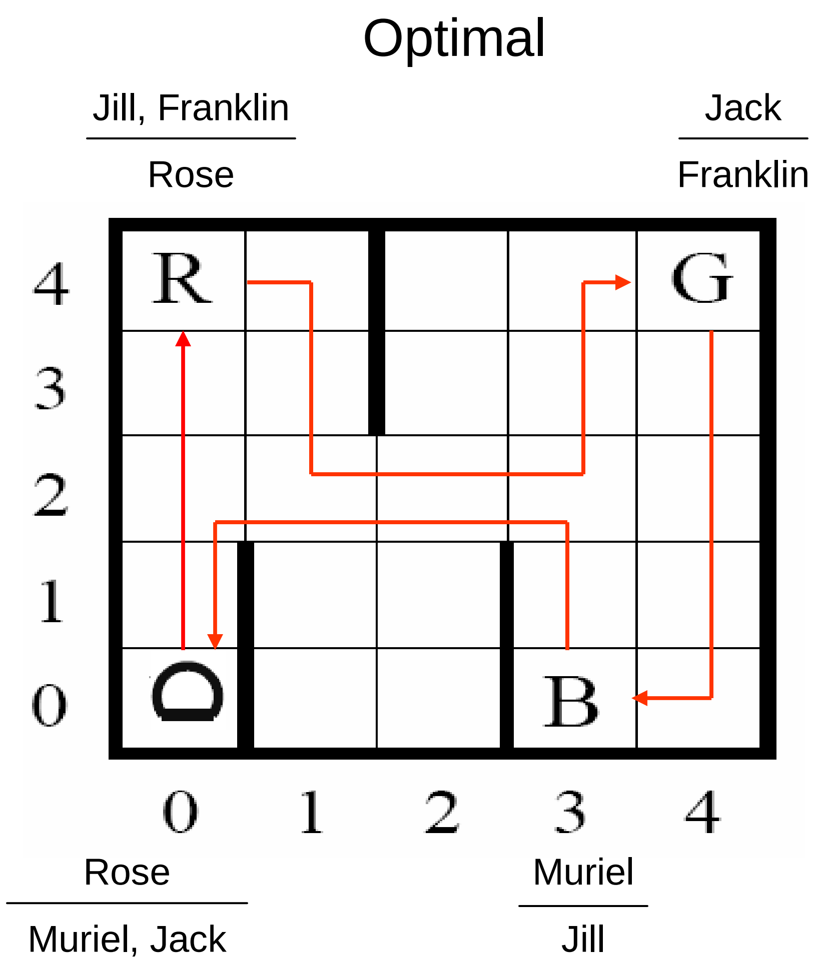


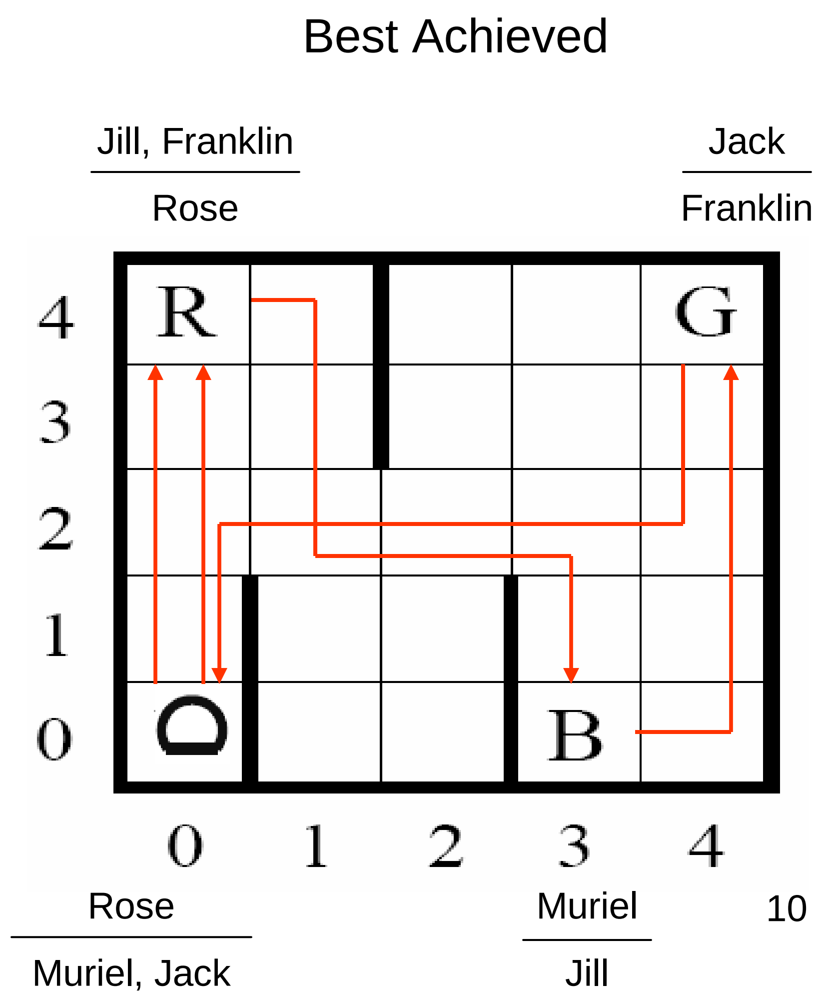
Learning to choose among high-level operators

- The task:
  - 5 passengers with various locations, destinations
  - Max of 2 taxi occupants
  - Learn the shortest path
- Decision-making:
  - Using learned policies for navigate
  - Select among Get($Pass_i$) & Put($Pass_i$) tasks
- Unhappy result – Convergence to non-optimal policies

5-passenger taxi task results

- Optimal policy – 61 steps
- In 5 trials converged to policies of length 75, 87, 69, 86, and 78





5-passenger taxi task –
What went wrong? Insufficient exploration

- The Good – Estimated values for the converged-upon policy accurate
- The Bad – Exploration:
  Experimental exploration strategy in which exploration probability decreases with visits to state
  - The Good – Automatically handles exploration → exploitation shift
    - Useful when different subtasks trained to varying degrees
    - Useful when different levels of hierarchy trained to varying degrees
  - The Bad – Lacks sufficient subtlety
    - Exploration in earlier states must continue longer than in later states
    - Exploration at higher levels of hierarchy must continue longer than at lower levels
- Exploration must improve before attempts to learn simultaneously at multiple levels of the hierarchy – which is possible in theory

Nuggets & Coal

- Nuggets
  - Using Soar's operator no-change impasses for hierarchical task decomposition provides a nice structure for implementing established hierarchical RL ideas
- Coal
  - Relies on impasses for task decomposition
  - Exploration strategies insufficient
  - Soar-RL rewards difficult to use – particularly making rewards appear right before a subgoal retracts