Incremental Learning of Planning Actions in Model-Based Reinforcement Learning

Jun Hao Alvin Ng and Ronald P. A. Petrick
Heriot-Watt University and University of Edinburgh

Presented by Hieu Le

Incremental Learning Model (ILM)

- **Main Goal**: How to learn the policy for an unknown model and deal with a non-stationary environment?

- **Non-stationary**: “True action values change over time”
- Learns only from experience of current episodes
- Does not use **past training data** because of non-stationary environment
- Can use **prior action model** as a starting point to learn new training data.
- Training data: “(s_t, a_t, s_{t+1}) where s_t is the pre-state, a_t is the action, and s_{t+1} is the post-state.”

Outline

- **Definitions of concepts**
- Prior Work
  - Related Work
  - Background: PPDDL, Rules
- Incremental Learning Model (ILM)
  - Algorithm Overview
  - Reliability of Actions
  - Explore of Exploit
  - Learning Rules
- Evaluation
  - Experiments (Tireworld)
  - Conclusion

Definitions of Concepts

- **Model Based Learning**: We train an agent on how to interact with the environment but focused more on learning the state transition dynamics.
  - Sample efficient
- **Planning**: Learning a policy (behavior strategy) to maximize sum of rewards or reach an end goal
- **Incremental Learning Model**: Learns from current episode only. Learns new actions or refine action models. Use learned models to synthesize a policy.
- **Past Training Data**: Data that was from past episodes
- **Prior Action Model**: An action model that was learned from previous training data

Outline

- Definitions of concepts
- **Prior Work**
  - Related Work
  - Background: PPDDL, Rules
- Incremental Learning Model (ILM)
  - Algorithm Overview: Main Contribution
  - Reliability of Actions
  - Explore of Exploit
  - Learning Rules
- Evaluation
  - Experiments (Tireworld)
  - Conclusion

Related Work and Background

- **R-MAX** [Brafman and Tennenholtz, 2002]
  - MBRL algorithm that balances exploration/exploitation by assigning the highest reward to unknown states. A state is known when the number of actions applied on that state surpasses some threshold. Not good for large state spaces.
  - **Incremental Learning Model (ILM) only assume that the arguments for actions are known**
- **Rules Learner** [Pasula et al., 2007]
  - A set of rules that represent actions with probabilistic effects
  - Does not address incremental nature of RL
  - **ILM: extends this work to allow updating rules incrementally and take into account prior action models**
- **Deterministic Action Models** [Rodrigues et al., 2010]
  - Provides a way to learn action models incrementally
  - **ILM: uses this concept to extend the rules learner for incremental learning and updates the rules when there are contradicting experiences**

Related Work and Background (2)

- **Probabilistic Planning Domain Definition Language** (PPDDL): A language to denote how policies can be structured for a specific domain.
  - ILM: uses PPDDL to define action models by their preconditions and effects. An action can applied if the preconditions are met, and once applied, the state changes (s') according the effects (in a probabilistic manner)
- **Rules**
  - Has three parts: name, precondition, and effect
  - "A rule covers a state-action pair (s, a) if it represents a and is applicable in s."
- **Action**
  - A set of rules

Related Work and Background (3)

Name: moveCar(?from ?to)
Precondition: $at(?from) \wedge road(?from\ ?to) \wedge notFlattire()$
Effect: $0.75\ at(?to) \wedge \neg at(?from)$
$0.25\ at(?to) \wedge \neg at(?from) \wedge \neg notFlattire()$
$0 \langle\ noise\ \rangle$

Figure 1: The rule for the true action model representing *moveCar* in the `Tireworld` domain with arguments *?from* and *?to*.

- **Grounded Rule:** By filling in the rule with real value or objects. Such as moveCar(112, 120)

Outline

- Definitions of concepts
- Prior Work
  - Related Work
  - Background: PPDDL, Rules
- **Incremental Learning Model (ILM)**
  - **Algorithm Overview: Main Contribution**
  - Reliability of Actions
  - Explore of Exploit
  - Learning Rules
- Evaluation
  - Experiments (Tireworld)
  - Conclusion

ILM: Algorithm Overview

Algorithm 1: Incremental Learning Model

```
Function
ILM(R_0, RE_0, s_0, g, H, N, ζ, EX_max, tabu):
    h ← H
    R ← R_0
    RE ← RE_0
    T ← ∅
    for t = 0 : N do
        a_t ← explore_or_exploit(s_t, g, h, R, RE, tabu, ζ)
        if a_t = ∅ then break
        s_{t+1}, st ← execute(a_t)
        T.append(s_t, a_t, s_{t+1})
        if st = fail then
            tabu.append(relevant_predicates(s_t, a_t), a_t)
        else
            T.append(synthetic_transition(tabu, s_{t+1}))
        R_prev ← R
        if can_learn(R, EX, EX_max) then
            R ← learn_rules(R_0, T, RE)
        RE, EX ← update(R, RE_0, T, st, R_prev)
        if s' ⊨ g then break
        if (N − t) < H then h ← h − 1
    return R, RE, max(EX, EX_max), tabu
```

- R = set of rules
- RE = reliability
- S_0 = initial state
- g = goal state
- H = planning horizon
- N = max number of iterations
- zeta = threshold for explore/exploit
- EX_{max}= max exposure = zero if not passed in
- tabu = list of failed states, predicates and actions
- T = training set = init to empty

ILM: Algorithm Overview

Algorithm 1: Incremental Learning Model

Function $\texttt{ILM}(R_0, RE_0, s_0, g, H, N, \zeta, EX_{max}, tabu)$:
- $h \leftarrow H$
- $R \leftarrow R_0$
- $RE \leftarrow RE_0$
- $\mathcal{T} \leftarrow \varnothing$
- **for** $t = 0 : N$ **do**
  - $a_t \leftarrow$ explore_or_exploit$(s_t, g, h, R, RE, tabu, \zeta)$
  - **if** $a_t = \varnothing$ **then break**
  - $s_{t+1}, st \leftarrow$ execute(a_t)
  - $\mathcal{T}$.append(s_t, a_t, s_{t+1})
  - **if** $st = fail$ **then**
    - $tabu$.append(relevant_predicates(s_t, a_t), a_t)
  - **else**
    - $\mathcal{T}$.append(synthetic_transition($tabu$, s_{t+1}))
  - $R_{prev} \leftarrow R$
  - **if** $can\_learn(R, EX, EX_{max})$ **then**
    - $R \leftarrow$ learn_rules$(R_0, \mathcal{T}, RE)$
  - $RE, EX \leftarrow$ update$(R, RE_0, \mathcal{T}, st, R_{prev})$
  - **if** $s' \models g$ **then break**
  - **if** $(N - t) < H$ **then** $h \leftarrow h - 1$
- **return** R, RE, max(EX, EX_{max}), $tabu$

- H = planning horizon
- R = set of rules
- RE = reliability
- T = training set
- N = max number of iterations
- S_0 = initial state
- g = goal state
- zeta = threshold for explore/exploit
- EX_{max}= max exposure = zero if not passed in
- tabu = list of failed states, predicates and actions

1. **Explore or exploit to find an action, then executes it**

ILM: Algorithm Overview

Algorithm 1: Incremental Learning Model

Function `ILM`$(R_0, RE_0, s_0, g, H, N, \zeta, EX_{max}, tabu)$**:**
- $h \leftarrow H$
- $R \leftarrow R_0$
- $RE \leftarrow RE_0$
- $\mathcal{T} \leftarrow \varnothing$
- **for** $t = 0 : N$ **do**
  - $a_t \leftarrow$ explore_or_exploit$(s_t, g, h, R, RE, tabu, \zeta)$
  - **if** $a_t = \varnothing$ **then break**
  - $s_{t+1}, st \leftarrow$ execute(a_t)
  - $\boxed{\mathcal{T}.\text{append}(s_t, a_t, s_{t+1})}$
  - **if** $st = fail$ **then**
    - $tabu$.append(relevant_predicates(s_t, a_t), a_t)
  - **else**
    - $\mathcal{T}$.append(synthetic_transition($tabu$, s_{t+1}))
  - $R_{prev} \leftarrow R$
  - **if** *can_learn*(R, EX, EX_{max}) **then**
    - $R \leftarrow$ learn_rules$(R_0, \mathcal{T}, RE)$
  - $RE, EX \leftarrow$ update$(R, RE_0, \mathcal{T}, st, R_{prev})$
  - **if** $s' \models g$ **then break**
  - **if** $(N - t) < H$ **then** $h \leftarrow h - 1$
- **return** R, RE, max(EX, EX_{max}), $tabu$

- H = planning horizon
- R = set of rules
- RE = reliability
- T = training set
- N = max number of iterations
- S_0 = initial state
- g = goal state
- zeta = threshold for explore/exploit
- EX_{max}= max exposure = zero if not passed in
- tabu = list of failed states, predicates and actions

1. Explore or exploit to find an action, then executes it
2. **Saves the SAS as training data.**

ILM: Algorithm Overview

Algorithm 1: Incremental Learning Model

Function `ILM`$(R_0, RE_0, s_0, g, H, N, \zeta, EX_{max}, tabu)$**:**
- $h \leftarrow H$
- $R \leftarrow R_0$
- $RE \leftarrow RE_0$
- $\mathcal{T} \leftarrow \varnothing$
- **for** $t = 0 : N$ **do**
  - $a_t \leftarrow$ explore_or_exploit$(s_t, g, h, R, RE, tabu, \zeta)$
  - **if** $a_t = \varnothing$ **then break**
  - $s_{t+1}, st \leftarrow$ execute(a_t)
  - $\mathcal{T}$.append(s_t, a_t, s_{t+1})
  - **if** $st = fail$ **then**
    - $tabu$.append(relevant_predicates(s_t, a_t), a_t)
  - **else**
    - $\mathcal{T}$.append(synthetic_transition($tabu$, s_{t+1}))
  - $R_{prev} \leftarrow R$
  - **if** $can\_learn(R, EX, EX_{max})$ **then**
    - $R \leftarrow$ learn_rules$(R_0, \mathcal{T}, RE)$
  - $RE, EX \leftarrow$ update$(R, RE_0, \mathcal{T}, st, R_{prev})$
  - **if** $s' \vDash g$ **then break**
  - **if** $(N - t) < H$ **then** $h \leftarrow h - 1$
- **return** R, RE, max(EX, EX_{max}), $tabu$

- H = planning horizon
- R = set of rules
- RE = reliability
- T = training set
- N = max number of iterations
- S_0 = initial state
- g = goal state
- zeta = threshold for explore/exploit
- EX_{max}= max exposure = zero if not passed in
- tabu = list of failed states, predicates and actions

1. Explore or exploit to find an action, then executes it
2. Saves the SAS as training data.
3. **If state fails, then save all information like predicates, state, actions to the tabu list. ELSE it becomes a viable state transition.**

ILM: Algorithm Overview

Algorithm 1: Incremental Learning Model

Function $\texttt{ILM}(R_0, RE_0, s_0, g, H, N, \zeta, EX_{max}, tabu)$ **:**

  $h \leftarrow H$
  $R \leftarrow R_0$
  $RE \leftarrow RE_0$
  $\mathcal{T} \leftarrow \varnothing$
  for $t = 0 : N$ **do**
    $a_t \leftarrow$ explore_or_exploit($s_t, g, h, R, RE, tabu, \zeta$)
    if $a_t = \varnothing$ **then break**
    $s_{t+1}, st \leftarrow$ execute(a_t)
    $\mathcal{T}$.append(s_t, a_t, s_{t+1})
    if $st = fail$ **then**
      $tabu$.append(relevant_predicates(s_t, a_t), a_t)
    else
      $\mathcal{T}$.append(synthetic_transition($tabu$, s_{t+1}))
    $R_{prev} \leftarrow R$
    if $can\_learn(R, EX, EX_{max})$ **then**
      $R \leftarrow$ learn_rules($R_0, \mathcal{T}, RE$)
    $RE, EX \leftarrow$ update($R, RE_0, \mathcal{T}, st, R_{prev}$)
    if $s' \models g$ **then break**
    if $(N - t) < H$ **then** $h \leftarrow h - 1$
  return R, RE, max(EX, EX_{max}), $tabu$

- H = planning horizon
- R = set of rules
- RE = reliability
- T = training set
- N = max number of iterations
- S_0 = initial state
- g = goal state
- zeta = threshold for explore/exploit
- EX_{max}= max exposure = zero if not passed in
- tabu = list of failed states, predicates and actions

1. Explore or exploit to find an action, then executes it
2. Saves the SAS as training data.
3. If state fails, then save all information like predicates, state, actions to the tabu list. ELSE it becomes a viable state transition.
4. **It figures out if it should learn new rules from this iteration. (called delayed learning)**

ILM: Algorithm Overview

Algorithm 1: Incremental Learning Model

Function `ILM`($R_0, RE_0, s_0, g, H, N, \zeta, EX_{max}, tabu$) **:**

- $h \leftarrow H$
- $R \leftarrow R_0$
- $RE \leftarrow RE_0$
- $\mathcal{T} \leftarrow \varnothing$
- **for** $t = 0 : N$ **do**
  - $a_t \leftarrow$ explore_or_exploit($s_t, g, h, R, RE, tabu, \zeta$)
  - **if** $a_t = \varnothing$ **then break**
  - $s_{t+1}, st \leftarrow$ execute(a_t)
  - $\mathcal{T}$.append(s_t, a_t, s_{t+1})
  - **if** $st = \mathit{fail}$ **then**
    - $tabu$.append(relevant_predicates(s_t, a_t), a_t)
  - **else**
    - $\mathcal{T}$.append(synthetic_transition($tabu$, s_{t+1}))
  - $R_{prev} \leftarrow R$
  - **if** *can_learn*(R, EX, EX_{max}) **then**
    - $R \leftarrow$ learn_rules($R_0, \mathcal{T}, RE$)
  - $\boxed{RE, EX \leftarrow \text{update}(R, RE_0, \mathcal{T}, st, R_{prev})}$
  - **if** $s' \models g$ **then break**
  - **if** $(N - t) < H$ **then** $h \leftarrow h - 1$
- **return** R, RE, max(EX, EX_{max}), $tabu$

- H = planning horizon
- R = set of rules
- RE = reliability
- T = training set
- N = max number of iterations
- S_0 = initial state
- g = goal state
- zeta = threshold for explore/exploit
- EX_{max}= max exposure = zero if not passed in
- tabu = list of failed states, predicates and actions

1. Explore or exploit to find an action, then executes it
2. Saves the SAS as training data.
3. If state fails, then save all information like predicates, state, actions to the tabu list. ELSE it becomes a viable state transition.
4. It figures out if it should learn new rules from this iteration. (called delayed learning)
5. **Updates RE and EX**

ILM: Algorithm Overview

Algorithm 1: Incremental Learning Model

Function `ILM`($R_0, RE_0, s_0, g, H, N, \zeta, EX_{max}, tabu$) **:**
  $h \leftarrow H$
  $R \leftarrow R_0$
  $RE \leftarrow RE_0$
  $\mathcal{T} \leftarrow \varnothing$
  for $t = 0 : N$ **do**
    $a_t \leftarrow$ explore_or_exploit($s_t, g, h, R, RE, tabu, \zeta$)
    if $a_t = \varnothing$ **then break**
    $s_{t+1}, st \leftarrow$ execute(a_t)
    $\mathcal{T}$.append(s_t, a_t, s_{t+1})
    if $st = fail$ **then**
      $tabu$.append(relevant_predicates(s_t, a_t), a_t)
    else
      $\mathcal{T}$.append(synthetic_transition($tabu$, s_{t+1}))
    $R_{prev} \leftarrow R$
    if *can_learn*(R, EX, EX_{max}) **then**
      $R \leftarrow$ learn_rules($R_0, \mathcal{T}, RE$)
    $RE, EX \leftarrow$ update($R, RE_0, \mathcal{T}, st, R_{prev}$)
    if $s' \models g$ **then break**
    if $(N - t) < H$ **then** $h \leftarrow h - 1$
  return R, RE, max(EX, EX_{max}), $tabu$

- H = planning horizon
- R = set of rules
- RE = reliability
- T = training set
- N = max number of iterations
- S_0 = initial state
- g = goal state
- zeta = threshold for explore/exploit
- EX_{max}= max exposure = zero if not passed in
- tabu = list of failed states, predicates and actions

1. Explore or exploit to find an action, then executes it
2. Saves the SAS as training data.
3. If state fails, then save all information like predicates, state, actions to the tabu list. ELSE it becomes a viable state transition.
4. It figures out if it should learn new rules from this iteration. (called delayed learning)
5. Updates RE and EX
6. **Stops if reaches goal state or passes various thresholds**

Outline

- Definitions of concepts
- Prior Work
    - Related Work
    - Background: PPDDL, Rules
- **Incremental Learning Model (ILM)**
    - Algorithm Overview
    - **Reliability of Actions: Empirical Estimates of Learning Progress**
    - Explore of Exploit
    - Learning Rules
- Evaluation
    - Experiments (Tireworld)
    - Conclusion

ILM: Reliability

- “The reliability of learned action models are empirical estimates of its learning progress”
- Two purposes: (1) learn new rules, (2) less reliable rules mean we should do exploration

$$\boxed{RE(o) = EX(o)\,(\alpha_s\,SU(o) - \alpha_v\,VO(o)) + \beta^n RE(o_0)}$$

- **O** = action
- **EX** = Exposure
- **SU** = Success rate of action o
- **VO** = Volatility of action o
- **B** = discount factor
- **n** = number of updates
- **RE** = reliability of prior action model (which can be empty)
- **alpha** = scaling factors

ILM: Success Rate

- Measures the success rate of recent actions

$$SU(o) = \beta\, SU(o) + \mathbb{1}(st = success) + 0.5 \times \mathbb{1}(st = partial\ success)$$

- **O** = action
- **B** = discount factor

- +1 if the state is a true success (else +0 for the term)
- +0.5 if it is a partial success (else zero for the term)

ILM: Difference between Two Rules

- Sums all set differences between the two rules for preconditions and effects

$$
\begin{aligned}
d(r_1, r_2) = {} & d^-(r_1^p, r_2^p) + d^-(r_2^p, r_1^p) \\
& + d^-(r_1^e, r_2^e) + d^-(r_2^e, r_1^e)
\end{aligned}
$$

- **p** **=** preconditions
- **e** = effects
- **d^{--}** = set difference. Number of preconditions or effects that only in the first set and not the second set.

d^{--} $(r_1{}^p$, $r_2{}^p)$ = give me all preconditions that only appear in $r_1$that does not exist in r_2

ILM: Normalized Difference for Two Rules

- Difference of two rules over the sum of predicates (both preconditions and effects)

$$\boxed{\tilde{d}(r_1, r_2) \quad = \quad \frac{d(r_1, r_2)}{|r_1| + |r_2|}}$$

ILM: Volatility

- Measures how much the rules are changing after learning
- Low volatility means that the learning is converging to the true action model

$$VO(o) = \beta VO(o) + \tilde{d}(R_{prev}, R)$$

- **O** = action
- **B** = discount factor
- **d_tilda** = normalized difference between two sets of rules

d_tilda is the normalized difference between two SETS of rules (instead of just between two rules as before)

ILM: Exposure

- Pairwise distance between all pre-states weighted by count of successful state transitions. **(variability of pre-states)**
- High exposure means it is a good action that has been applied on many states (with success)

$$EX(o) = \frac{N_s}{|\mathcal{S}|C_2} \sum_{s_i, s_j \in \mathcal{S}} \frac{d^-(s_i, s_j)}{|s_i|} + \frac{d^-(s_j, s_i)}{|s_j|}$$

- **O** = action
- $\mathbf{N_s}$ = count of successful state transitions
- **S** = set of pre-states
- **|S|** = count of unique pre-states
- C_2 = not defined by paper

ILM: Reliability (2)

- We want high exposure for an action
- Success rate should be high
- Volatility should be low (changes between rules)

$$RE(o) = EX(o)\,(\alpha_s\,SU(o) - \alpha_v\,VO(o)) + \beta^n RE(o_0)$$

- **O** = action
- **EX** = Exposure
- **SU** = Success rate of action o
- **VO** = Volatility of action o
- **B** = discount factor
- **n** = number of updates
- **RE** = reliability of prior action model (which can be empty)
- **alpha** = scaling factors

Outline

- Definitions of concepts
- Prior Work
  - Related Work
  - Background: PPDDL, Rules
- **Incremental Learning Model (ILM)**
  - Algorithm Overview
  - Reliability of Actions
  - **Explore of Exploit: When are states known?**
  - Learning Rules
- Evaluation
  - Experiments (Tireworld)
  - Conclusion

ILM: Explore or Exploit

Algorithm 1: Incremental Learning Model

Function $\texttt{ILM}(R_0, RE_0, s_0, g, H, N, \zeta, EX_{max}, tabu)$**:**
- $h \leftarrow H$
- $R \leftarrow R_0$
- $RE \leftarrow RE_0$
- $\mathcal{T} \leftarrow \varnothing$
- **for** $t = 0 : N$ **do**
  - $\boxed{a_t \leftarrow \text{explore\_or\_exploit}(s_t, g, h, R, RE, tabu, \zeta)}$
  - **if** $a_t = \varnothing$ **then break**
  - $s_{t+1}, st \leftarrow \text{execute}(a_t)$
  - $\mathcal{T}.\text{append}(s_t, a_t, s_{t+1})$
  - **if** $st = fail$ **then**
    - $tabu.\text{append}(\text{relevant\_predicates}(s_t, a_t), a_t)$
  - **else**
    - $\mathcal{T}.\text{append}(\text{synthetic\_transition}(tabu, s_{t+1}))$
  - $R_{prev} \leftarrow R$
  - **if** $can\_learn(R, EX, EX_{max})$ **then**
    - $R \leftarrow \text{learn\_rules}(R_0, \mathcal{T}, RE)$
  - $RE, EX \leftarrow \text{update}(R, RE_0, \mathcal{T}, st, R_{prev})$
  - **if** $s' \models g$ **then break**
  - **if** $(N - t) < H$ **then** $h \leftarrow h - 1$
- **return** $R, RE, \max(EX, EX_{max}), tabu$

- H = planning horizon
- R = set of rules
- RE = reliability
- T = training set
- N = max number of iterations
- S_0 = initial state
- g = goal state
- zeta = threshold for explore/exploit
- EX_{max}= max exposure = zero if not passed in
- tabu = list of failed states, predicates and actions

- Keep a list (L) of applicable actions for a state s_t
- A **state is known** if all actions in L have a reliability score that is higher than threshold zeta.
- **Explore** if state is unknown: randomly select one action from L to apply. Dead end if no actions can be applied successfully
- **Exploit** if state is known

Outline

- Definitions of concepts
- Prior Work
  - Related Work
  - Background: PPDDL, Rules
- **Incremental Learning Model (ILM)**
  - Algorithm Overview
  - Reliability of Actions
  - Explore of Exploit
  - **Learning Rules: When to Learn New Rules?**
- Evaluation
  - Experiments (Tireworld)
  - Conclusion

ILM: Learning Rules

Algorithm 1: Incremental Learning Model

Function $\texttt{ILM}(R_0, RE_0, s_0, g, H, N, \zeta, EX_{max}, tabu)$ **:**

- $h \leftarrow H$
- $R \leftarrow R_0$
- $RE \leftarrow RE_0$
- $\mathcal{T} \leftarrow \varnothing$
- **for** $t = 0 : N$ **do**
  - $a_t \leftarrow$ explore_or_exploit$(s_t, g, h, R, RE, tabu, \zeta)$
  - **if** $a_t = \varnothing$ **then break**
  - $s_{t+1}, st \leftarrow$ execute(a_t)
  - $\mathcal{T}$.append(s_t, a_t, s_{t+1})
  - **if** $st = fail$ **then**
    - $tabu$.append(relevant_predicates(s_t, a_t), a_t)
  - **else**
    - $\mathcal{T}$.append(synthetic_transition($tabu$, s_{t+1}))
  - $R_{prev} \leftarrow R$
  - **if** $can\_learn(R, EX, EX_{max})$ **then**
    - $R \leftarrow$ learn_rules$(R_0, \mathcal{T}, RE)$
  - $RE, EX \leftarrow$ update$(R, RE_0, \mathcal{T}, st, R_{prev})$
  - **if** $s' \models g$ **then break**
  - **if** $(N - t) < H$ **then** $h \leftarrow h - 1$
- **return** R, RE, max(EX, EX_{max}), $tabu$

- H = planning horizon
- R = set of rules
- RE = reliability
- T = training set
- N = max number of iterations
- S_0 = initial state
- g = goal state
- zeta = threshold for explore/exploit
- EX_{max}= max exposure = zero if not passed in
- tabu = list of failed states, predicates and actions

- Prevent generalizing rules with low variability (low exposure)
- If R is empty, always learn new rules
- Otherwise: learn if the training set has at least 1 successful training transition, one failed or synthetic transition, and:

$$EX > \alpha_{EX}\, EX_{max} \text{ where } \alpha_{EX} \in [0, 1].$$

ILM: New Rules

- Applies rules learner from [Pasula et al., 2007].
- Goals:
  - Find which parts of the search space is the most desirable
  - Create rules that are the most likely to happen

$$Score(R) = \sum_{(s,a,s') \in \mathcal{T}} log(\hat{P}(s' \mid s, a, r_{(s,a)})) - \alpha_p \sum_{r \in R} PEN(r) - PEN(R, R_0)$$

- **P_hat** = likelihood of r(s,a) being applied to s to reach s'
- **PEN(r):** penalty for r's complexity
- **PEN(R,R_0):** penalty for deviation between old and new sets

ILM: Deviation Penalty

- High RE means higher penalty : if set of rules are reliable, no need to deviate
- High EX: means less penalty: encourage deviation since training data has variability.

$$\boxed{\begin{aligned} PEN(R, R_0) = \frac{RE(o_0)}{EX(o)} \Big[& \alpha_{drop}\, \Delta_{drop}(R, R_0) + \\ & + \alpha_{add}\, \Delta_{add}(R, R_0) \Big] \end{aligned}}$$

- **alphas:** scaling parameters

- **delta_drop:** set diff of removed rules
- **delta_add:** set diff of added rules

Outline

- Definitions of concepts
- Prior Work
  - Related Work
  - Background: PPDDL, Rules
- Incremental Learning Model (ILM)
  - Algorithm Overview
  - Reliability of Actions
  - Explore of Exploit
  - Learning Rules: When to Learn New Rules?
- **Evaluation**
  - Experiments (Tireworld)
  - Conclusion

Evaluation: Tireworld

- Apply algorithm on a planning domain called **Tireworld**.

- Get car from starting point A to destination B
- Every time we move the car, there is a chance it can get a flat tire
- If there is a flat tire and no spare tires are available, it has reached a dead-end (stop experiment)
- All dead-ends are avoidable

Evaluation: Correctness of Model

- Uses **variational distance** between P_hat and the true model P

$$\boxed{VD(P,\hat{P}) = \frac{1}{|\mathcal{T}|}\sum_{T_i \in \mathcal{T}} |P(T_i) - \hat{P}(T_i)|}$$

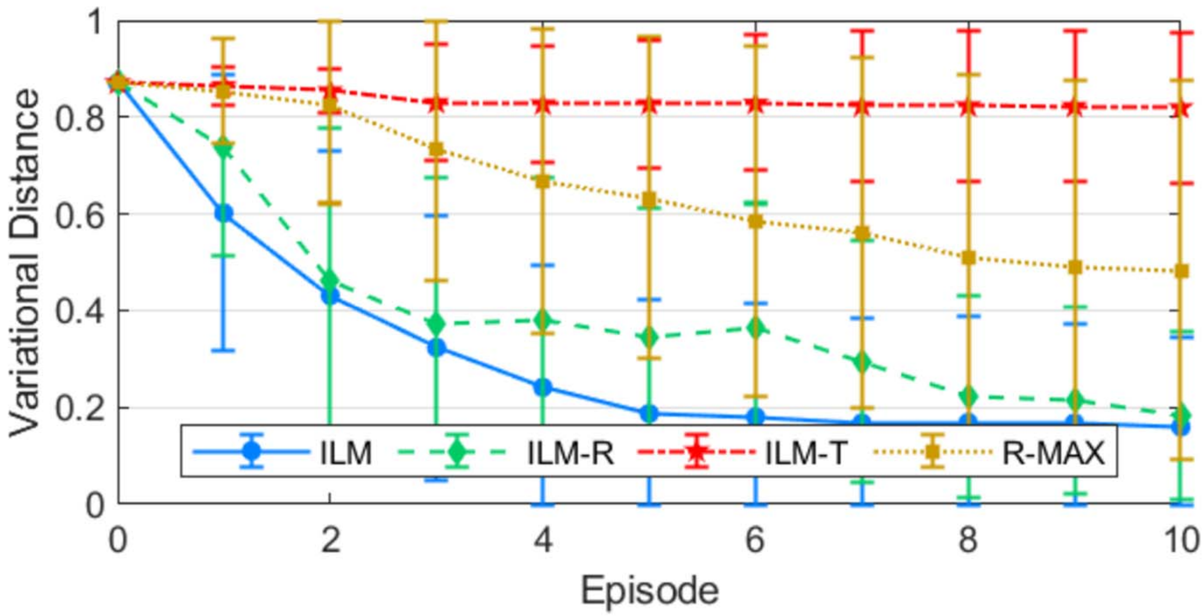
- **T:** generated training samples from true distribution

Evaluation: Comparing ILM

Compares ILM with:

- ILM-R : ILM that does **NOT** use reliability
- ILM-T : ILM that does not keep track of failed states
- R-MAX: from prior work

Evaluation: Results (Tireworld)



- The higher the distance = bad performance
- ILM-T proves that learning failed states is significant contribution to good performance
- Performance between reliability and no reliability is similar

Conclusion

- Learning from failure improved correctness and goal-directedness (helping the agent achieve its goal)
- Reliability concept helped influence how learning and planning are done
- Extended rules learner to use prior action models = more efficient learning
- Limitations:
  - Learning complex domains still require more training data
  - Still cannot use past training data due to non-stationary nature of domain

Some Questions

1. What do high reliability and low reliability mean? How does each case affect learning of new rules?