

Integrating Learning into a BDI Agent for Environments with Changing Dynamics

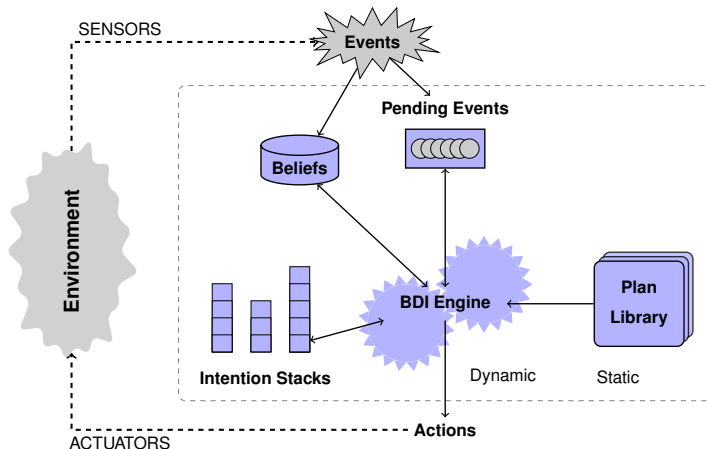
Dhirendra Singh¹ Sebastian Sardina¹ Lin Padgham¹ Geoff James²

¹RMIT University, Melbourne, Australia

²CSIRO Energy Technology, Sydney, Australia

International Joint Conference on Artificial Intelligence
July 2011, Barcelona, Spain

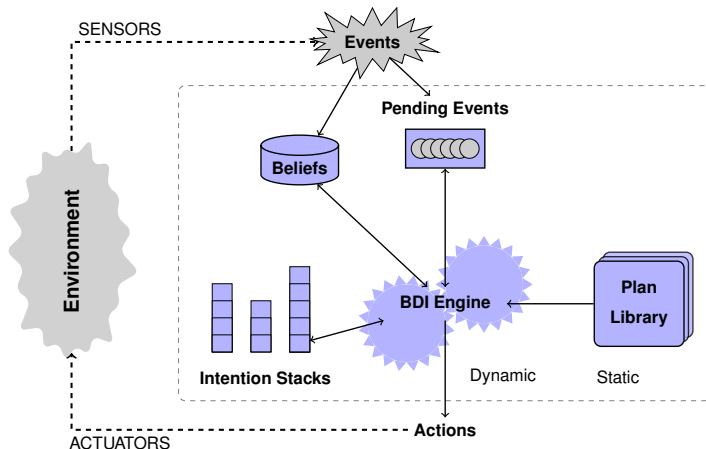
Belief-Desire-Intention (BDI) Agent Architecture



A **plan** is a programmed recipe for achieving a goal in some situation.

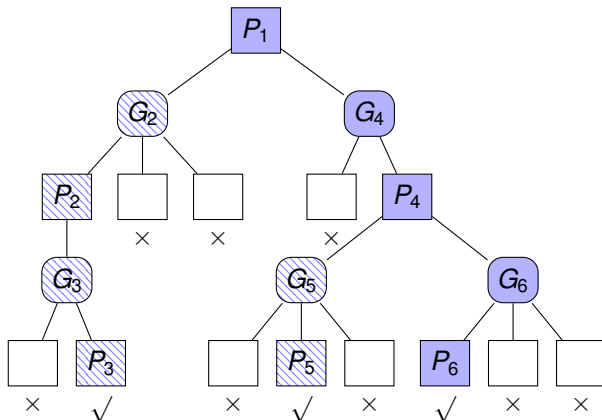
We wish to **improve plan selection** in situations, based on actual experience.

Belief-Desire-Intention (BDI) Agent Architecture



A **plan** is a programmed recipe for achieving a goal in some situation.
We wish to **improve plan selection** in situations, based on actual experience.

Plan Selection in a BDI Goal-Plan Hierarchy



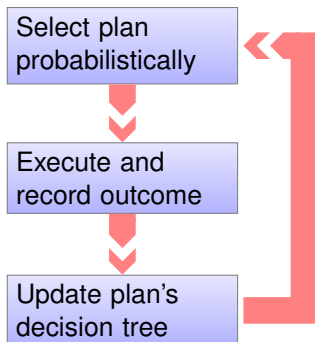
For plan P_1 to succeed, **several correct choices must be made.**

The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.

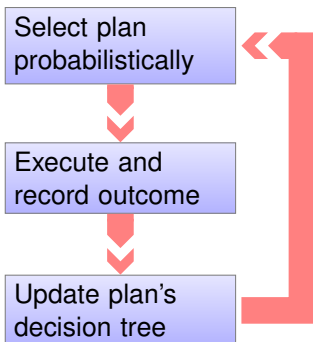
The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



The Learning Framework

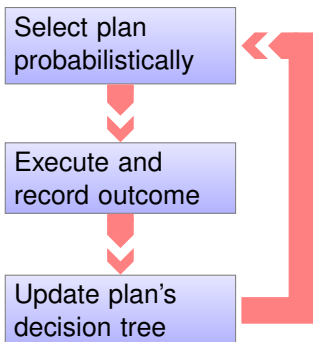
Augment a **decision tree per plan**. State representation includes world features, event parameters, and context variables.



- 1 For every plan whose programmed applicability condition holds, calculate a **selection weight** based on perceived likelihood of success.
- 2 Select a plan probabilistically using selection weights.
- 3 Execute the plan (hierarchy) and record the outcome(s).
- 4 Update the plan's decision tree.
- 5 Repeat.

The Learning Framework

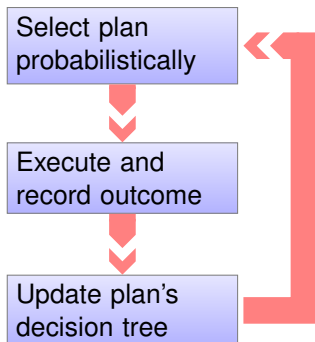
Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



- 1 For every plan whose programmed applicability condition holds, calculate a **selection weight** based on perceived likelihood of success.
- 2 **Select** a plan probabilistically using selection weights.
- 3 **Execute** the plan (hierarchy) and record the outcome(s).
- 4 **Update** the plan's decision tree.
- 5 Repeat.

The Learning Framework

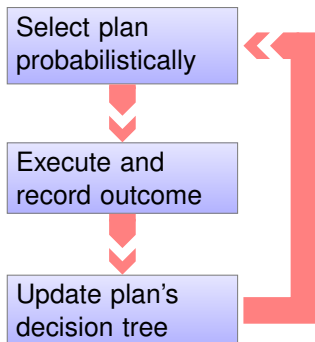
Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



- 1 For every plan whose programmed applicability condition holds, calculate a **selection weight** based on perceived likelihood of success.
- 2 **Select** a plan probabilistically using selection weights.
- 3 **Execute** the plan (hierarchy) and record the outcome(s).
- 4 **Update** the plan's decision tree.
- 5 Repeat.

The Learning Framework

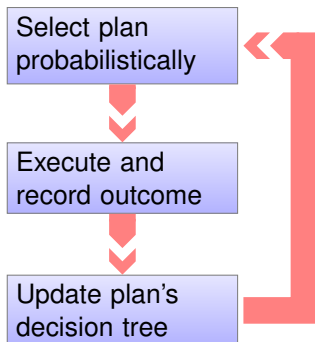
Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



- 1 For every plan whose programmed applicability condition holds, calculate a **selection weight** based on perceived likelihood of success.
- 2 **Select** a plan probabilistically using selection weights.
- 3 **Execute** the plan (hierarchy) and record the outcome(s).
- 4 **Update** the plan's decision tree.
- 5 Repeat.

The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



- 1 For every plan whose programmed applicability condition holds, calculate a **selection weight** based on perceived likelihood of success.
- 2 **Select** a plan probabilistically using selection weights.
- 3 **Execute** the plan (hierarchy) and record the outcome(s).
- 4 **Update** the plan's decision tree.
- 5 Repeat.

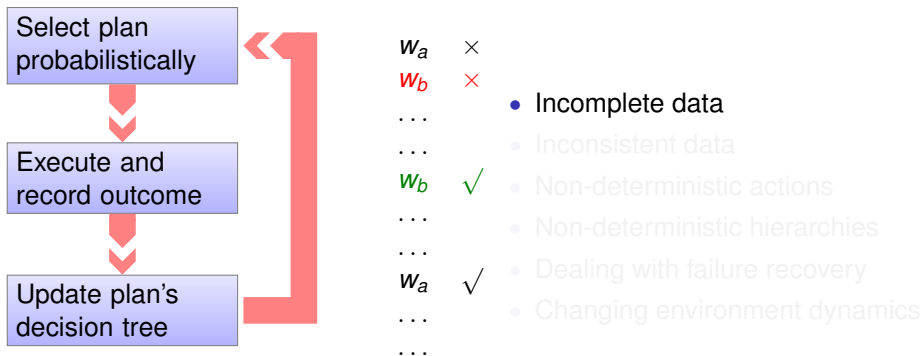
The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



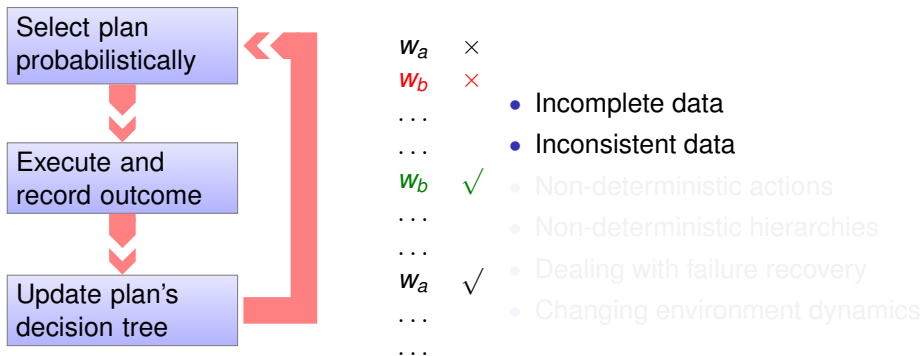
The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



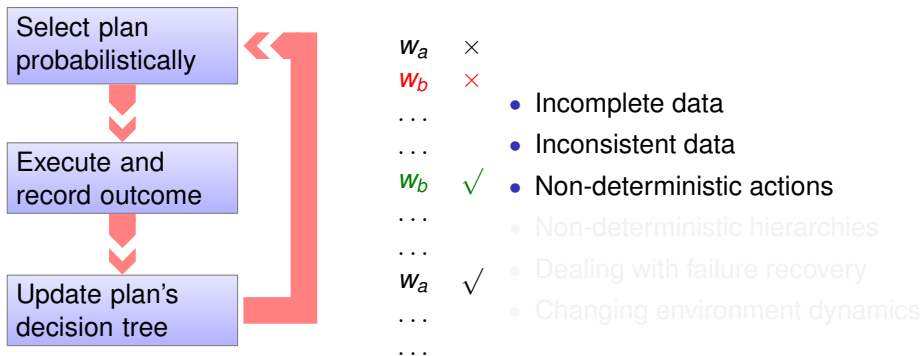
The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



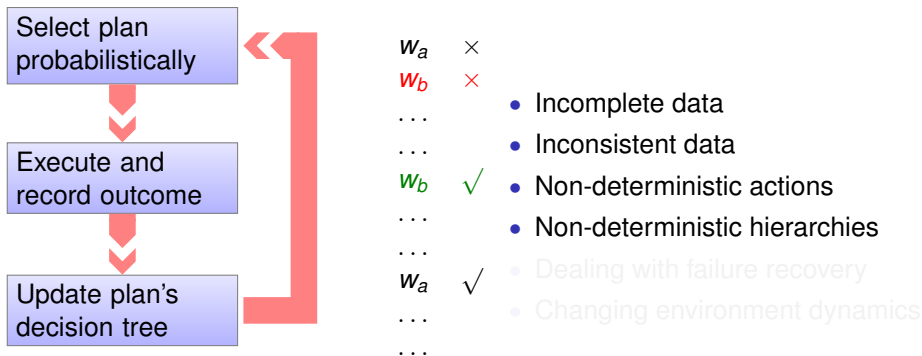
The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



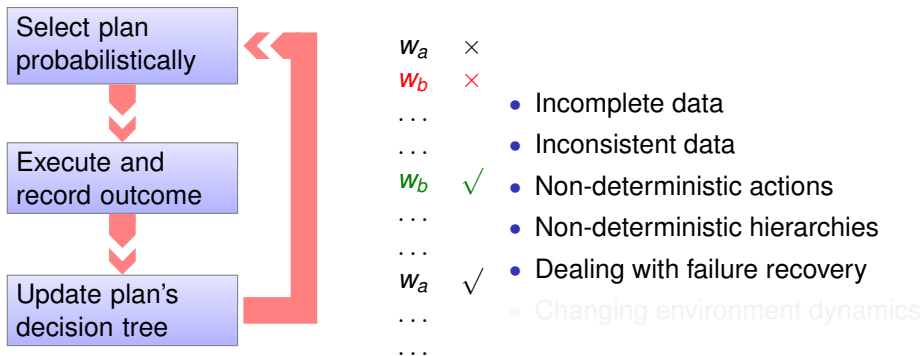
The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



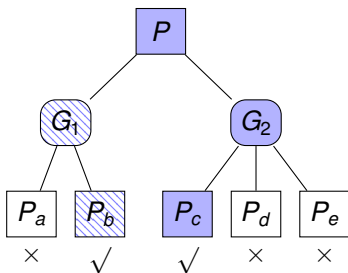
The Learning Framework

Augment **a decision tree per plan**. State representation includes world features, event parameters, and context variables.



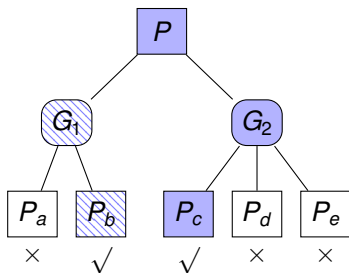
Need a quantitative **measure of confidence** in the current learning.

A Dynamic Confidence Measure



We build confidence from **observed performance** of a plan using:

A Dynamic Confidence Measure

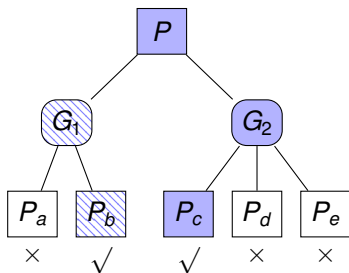


We build confidence from **observed performance** of a plan using:

- how well-informed were the recent decisions, or **stability-based** measure
- how well we know the worlds we are witnessing, or **world-based** measure
- an averaging window n and preference bias α

Plan selection weight, that dictates **exploration**, is then calculated using the predicted **likelihood of success** together with the **confidence** measure.

A Dynamic Confidence Measure

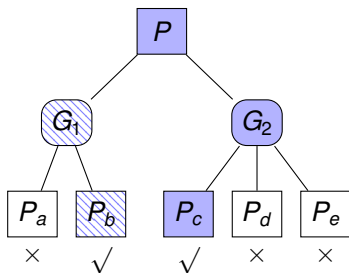


We build confidence from **observed performance** of a plan using:

- how well-informed were the recent decisions, or **stability-based** measure
- how well we know the worlds we are witnessing, or **world-based** measure
- an averaging window n and preference bias α

Plan selection weight, that dictates **exploration**, is then calculated using the predicted **likelihood of success** together with the **confidence** measure.

A Dynamic Confidence Measure

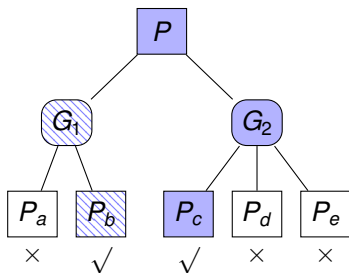


We build confidence from **observed performance** of a plan using:

- how well-informed were the recent decisions, or **stability-based** measure
- how well we know the worlds we are witnessing, or **world-based** measure
- an averaging window n and preference bias α

Plan selection weight, that dictates **exploration**, is then calculated using the predicted **likelihood of success** together with the **confidence** measure.

A Dynamic Confidence Measure

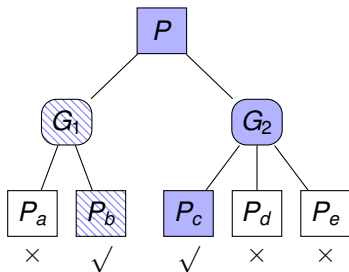


We build confidence from **observed performance** of a plan using:

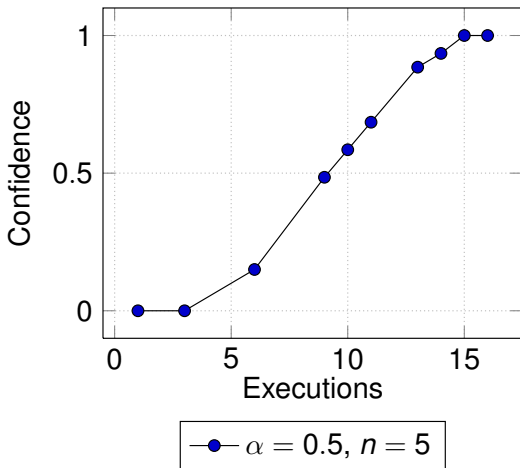
- how well-informed were the recent decisions, or **stability-based** measure
- how well we know the worlds we are witnessing, or **world-based** measure
- an averaging window n and preference bias α

Plan selection weight, that dictates **exploration**, is then calculated using the predicted **likelihood of success** together with the **confidence** measure.

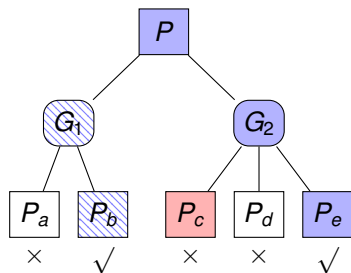
Example: Dynamic Confidence Measure



Solution found at $E=10$ and **full confidence at $E=15$.**



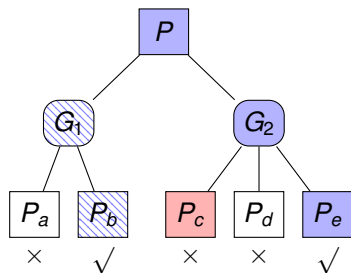
Example: Dynamic Confidence Measure



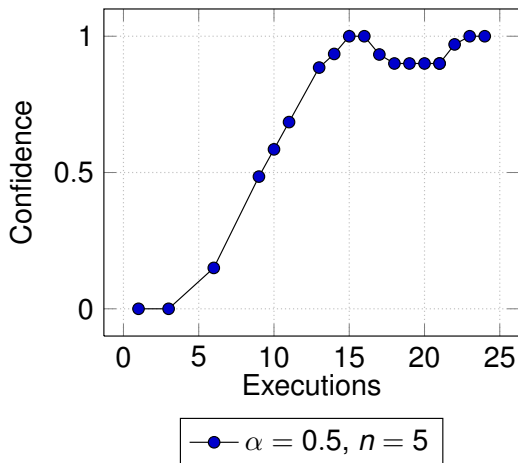
What if the environment changes after we have learnt the solution?

Say after execution 15, plan P_c no longer works for resolving goal G_2 , but plan P_e does.

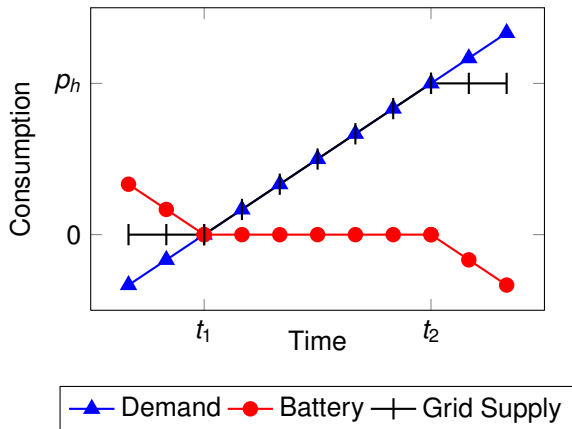
Example: Dynamic Confidence Measure



After $E=15$, P_c starts to fail.
The **confidence drops**,
promoting new exploration and
re-learning.

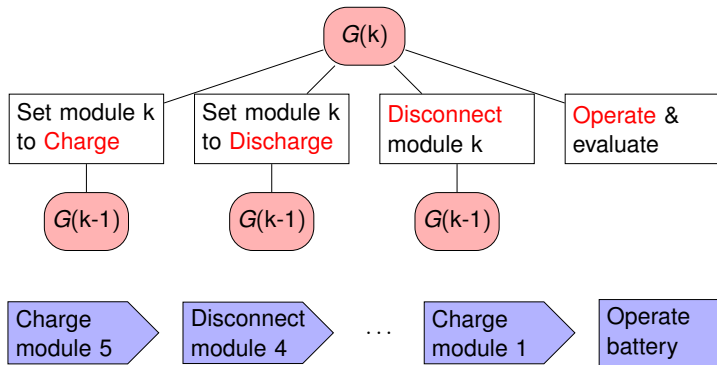


A Battery Storage Application



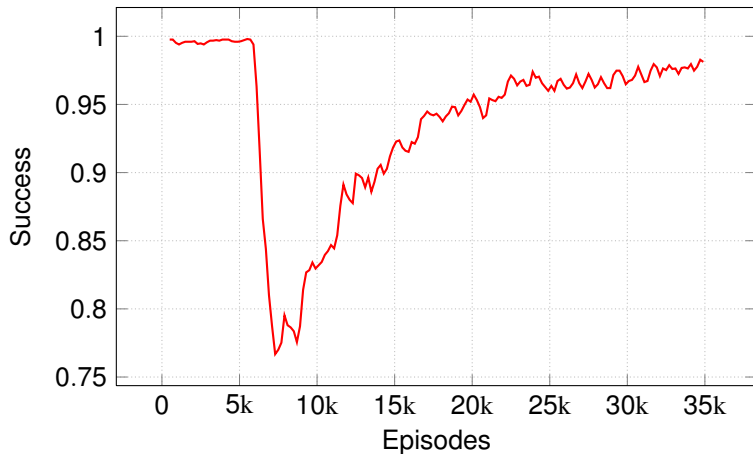
Given net building demand, **calculate an appropriate battery response** in order to maintain grid power consumption within range $[0, p_h]$.

Design: A Battery Storage Application



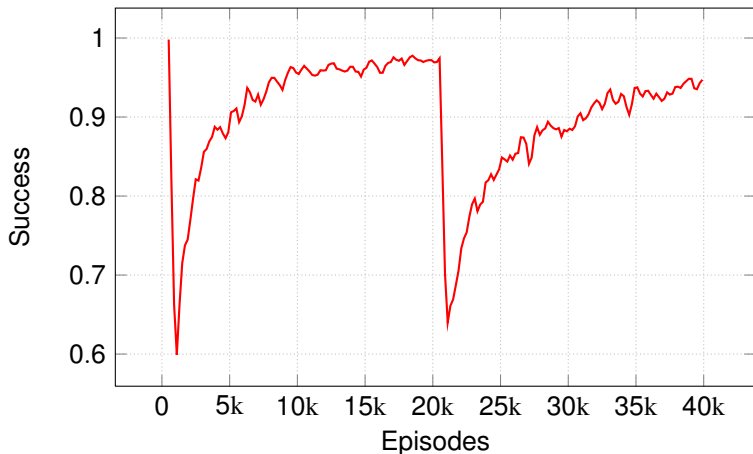
Aim: Learn appropriate plan selection to achieve a desired **battery response rate**, given the current battery state. **State space for a battery with five modules is ≈ 13 million.**

Experiment: Capacity Deterioration



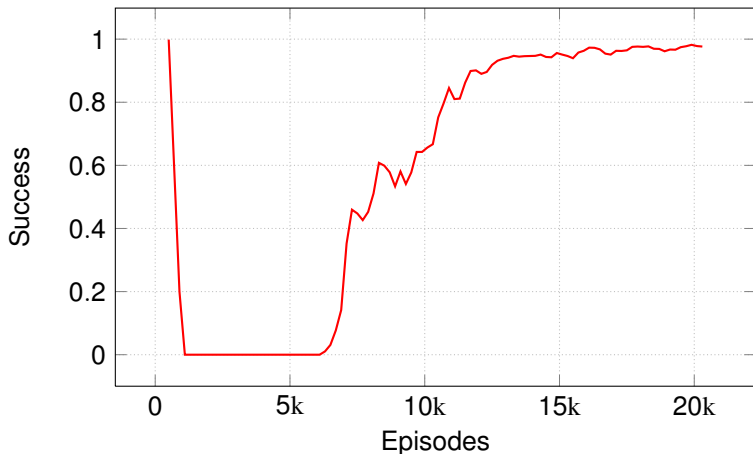
Recovery from **deterioration in module capacities** at 5k episodes.

Experiment: Partial Failure with Restoration



Recovery from **temporary module failures** during $[0, 20k]$, $[20k, 40k]$ episodes.

Experiment: Complete Failure with Restoration



Recovery from **complete system failure** during $[0, 5k]$ episodes.

Limitations and Future Work

- We cannot account for **inter-dependence between subgoals** of a higher-level plan.
- **Maintaining full training data is not practical.** It becomes obsolete when the environment changes. We tried an arbitrary scheme to filter “old” data in the battery agent and reduced the data set by 75% with no loss in performance.
- Onus is on the designer to select appropriate state representation and learning parameters. Some of this could be automatically extracted from the BDI program.

Limitations and Future Work

- We cannot account for **inter-dependence between subgoals** of a higher-level plan.
- **Maintaining full training data is not practical**. It becomes obsolete when the environment changes. **We tried an arbitrary scheme to filter “old” data in the battery agent and reduced the data set by 75% with no loss in performance.**
- Onus is on the designer to select appropriate state representation and learning parameters. Some of this could be automatically extracted from the BDI program.

Limitations and Future Work

- We cannot account for **inter-dependence between subgoals** of a higher-level plan.
- **Maintaining full training data is not practical**. It becomes obsolete when the environment changes. **We tried an arbitrary scheme to filter “old” data in the battery agent and reduced the data set by 75% with no loss in performance.**
- Onus is on the designer to select appropriate state representation and learning parameters. **Some of this could be automatically extracted from the BDI program.**

Summary

- Learning BDI plan selection is desirable when plan applicability cannot be easily programmed or when the **environment changes over time**.
- We present a **learning framework** for additionally determining a plan's applicability conditions using decision trees. Plans are selected **probabilistically** based on their predicted likelihood of success and our perceived **confidence** in the learning.
- We evaluate the framework empirically in a storage domain where a battery controller must continually adapt to **changes that cause previous solutions to become ineffective**.

Summary

- Learning BDI plan selection is desirable when plan applicability cannot be easily programmed or when the **environment changes over time**.
- We present a **learning framework** for additionally determining a plan's applicability conditions using decision trees. Plans are selected **probabilistically** based on their predicted likelihood of success and our perceived **confidence** in the learning.
- We evaluate the framework empirically in a storage domain where a battery controller must continually adapt to **changes that cause previous solutions to become ineffective**.

- Learning BDI plan selection is desirable when plan applicability cannot be easily programmed or when the **environment changes over time**.
- We present a **learning framework** for additionally determining a plan's applicability conditions using decision trees. Plans are selected **probabilistically** based on their predicted likelihood of success and our perceived **confidence** in the learning.
- We evaluate the framework empirically in a storage domain where a battery controller must continually adapt to **changes that cause previous solutions to become ineffective**.

- Learning BDI plan selection is desirable when plan applicability cannot be easily programmed or when the **environment changes over time**.
- We present a **learning framework** for additionally determining a plan's applicability conditions using decision trees. Plans are selected **probabilistically** based on their predicted likelihood of success and our perceived **confidence** in the learning.
- We evaluate the framework empirically in a storage domain where a battery controller must continually adapt to **changes that cause previous solutions to become ineffective**.

References



D. Singh, S. Sardina, L. Padgham.

Extending BDI plan selection to incorporate learning from experience.
Journal of Robotics and Autonomous Systems (RAS), 2010.



D. Singh, S. Sardina, L. Padgham, and S. Airiau.

Learning context conditions for BDI plan selection.
Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS), 2010.



S. Airiau, L. Padgham, S. Sardina, and S. Sen.

Enhancing adaptation in BDI agents using learning techniques.
International Journal of Agent Technologies and Systems (IJATS), 2009.

Poster today in Rooms 133–134 at 16:40–17:20