Structured Computation and Representation in Deep Reinforcement Learning

Jessica B. Hamrick
DeepMind

AAMAS/ICML/IJCAI Workshop on Prediction and Generative Modeling in RL
Stockholm, Sweden
July 15, 2018
Understanding the World in Terms of Objects and Relations
Understanding the World in Terms of Objects and Relations
Understanding the World in Terms of Objects and Relations
Understanding the World in Terms of Objects and Relations
Structure
Structure

The arrangement of and *relations* between the *parts* or elements of something complex.

(Oxford English Dictionary)
Structure

The arrangement of and *relations* between the *parts* or elements of something complex.

*(Oxford English Dictionary)*

In humans: Evolution \(\rightarrow\) Structure
Structure

The arrangement of and *relations* between the *parts* or elements of something complex.

(Oxford English Dictionary)

In humans: Evolution $\rightarrow$ Structure

In ML: Structure $\rightarrow$ Inductive bias / prior / regularizer
Two Types of Structure
Two Types of Structure

1. **Structured Computation**: the way that individual computations or functions are composed into more complex structures.
Two Types of Structure

1. **Structured Computation**: the way that individual computations or functions are composed into more complex structures.

2. **Structured Representation**: the format of the data that computations are performed over, e.g. sets, graphs, programs, etc.
A Caveat
A Caveat

Structured computation and structured representation go hand in hand!
A Caveat

Structured computation and structured representation go hand in hand!

Often, a structured representation entails a particular structure in the computation, and vice versa.
A Caveat

Structured computation and structured representation go hand in hand!

Often, a structured representation entails a particular structure in the computation, and vice versa.

But it can still be useful to think separately about:
(1) how computations are arranged and composed, and
(2) the specific form of the representations.
A Caveat

Structured computation and structured representation go hand in hand!

Often, a structured representation entails a particular structure in the computation, and vice versa.

But it can still be useful to think separately about:
(1) how computations are arranged and composed, and
(2) the specific form of the representations.
A Caveat

Structured computation and structured representation go hand in hand!

Often, a structured representation entails a particular structure in the computation, and vice versa.

But it can still be useful to think separately about:
(1) how computations are arranged and composed, and
(2) the specific form of the representations.
Two Types of Structure

1. **Structured Computation**: the way that individual computations or functions are composed into more complex structures.
   
   *Hamrick, Ballard, Pascanu, Vinyals, Heess, & Battaglia (2017, ICLR)*

2. **Structured Representation**: the format of the data that computations are performed over.


Claim: flexibility, adaptivity and generalization are about *having the right structure* (which may or may not mean having a model!)
Structure in Deep Networks
Structure in Deep Networks

Fully-Connected Layer
Unstructured computation
Unstructured representation
Structure in Deep Networks

- Fully-Connected Layer
  - Unstructured computation
  - Unstructured representation

- Convolutional Layer
  - Structured computation
  - Unstructured representation

Sharing in space
Structure in Deep Networks

- **Fully-Connected Layer**
  - Unstructured computation
  - Unstructured representation

- **Convolutional Layer**
  - Structured computation
  - Unstructured representation

- **Recurrent Layer**
  - Structured computation
  - Unstructured representation
Structure in Deep Networks

GoogLeNet with Inception modules, Szegedy et al. (2015)
Structure in Model-Free Deep RL

State → Action-Value Network or Policy Network → Action
Structure in Model-Free Deep RL

Mostly unstructured computation

State \rightarrow \text{Action-Value Network or Policy Network} \rightarrow \text{Action}
Structure in Model-Free Deep RL

State → Action-Value Network or Policy Network → Action

Unstructured representation

Mostly unstructured computation
Structure in Model-Based Deep RL

State → Search / Planning / Optimization → Action
Structure in Model-Based Deep RL

State → Model → Search / Planning / Optimization → Action

Structured computation!
Structure in Model-Based Deep RL

- AlphaGo (Silver et al., 2016)
- Imagination-Based Decision Making and Planning (Hamrick et al., 2017; Pascanu et al., 2017)
- Imagination-Augmented Agents (Weber et al., 2017)
- Gradient Based Planning (Henaff et al., 2017)
- Value Prediction Networks (Oh et al., 2017)
- Universal Planning Networks (Srinivas et al., 2018)
- … and more!

Structured computation!
Outline

1. **Structured Computation**: the way that individual computations or functions are composed into more complex structures.
   
   *Hamrick, Ballard, Pascanu, Vinyals, Heess, & Battaglia (2017, ICLR)*

2. **Structured Representation**: the format of the data that computations are performed over.
   

Outline

1. **Structured Computation**: the way that individual computations or functions are composed into more complex structures.
   
   *Hamrick, Ballard, Pascanu, Vinyals, Heess, & Battaglia (2017, ICLR)*

2. **Structured Representation**: the format of the data that computations are performed over.


“Reactive Controller”

Scene \( (x) \) → Controller \( \pi^C \) → World \( (f) \) → Outcome \( (x') \)

Performance loss \( (L_P) \)
"Reactive Controller"

**Controller (MLP):**
Proposes a control that is sent to the world to minimize performance loss.
“Reactive Controller”

**Controller (MLP):** Proposes a control that is sent to the world to minimize performance loss

**World:** The true environment that the agent is acting in

- Scene ($x$)
- Controller ($\pi^C$)
- World ($f$)
- Outcome ($x'$)
- Performance loss ($L_P$)
Reactive controller: simple task
Reactive controller: simple task
Reactive controller: hard task
Reactive controller: hard task
“Iterative” Controller
“Iterative” Controller

*Expert (IN, MLP, etc.):* Model of the world that evaluates proposed controls
“Iterative” Controller

**Expert (IN, MLP, etc.):**
Model of the world that evaluates proposed controls

**Memory (LSTM):**
Encodes the full history of controls and opinions
“Iterative” Controller

Expert (IN, MLP, etc.): Model of the world that evaluates proposed controls

Memory (LSTM): Encodes the full history of controls and opinions
“Iterative” Controller

**Expert (IN, MLP, etc.):** Model of the world that evaluates proposed controls

**Memory (LSTM):** Encodes the full history of controls and opinions
“Iterative” Controller

*Expert (IN, MLP, etc.):* Model of the world that evaluates proposed controls

*Memory (LSTM):* Encodes the full history of controls and opinions
“Iterative” Controller

**Expert (IN, MLP, etc.):** Model of the world that evaluates proposed controls

**Memory (LSTM):** Encodes the full history of controls and opinions

```
Controller \( \pi^C \)

Switch

World \( f \)

Outcome \( x' \)
Performance loss \( L_P \)

Expert \( E \)

Opinion \( e \)

Memory \( \mu \)

Scene \( x \)
History \( h_{n-1} \)

History \( h_n \)

Control \( c_n \)

Scene \( x \)
History \( h_{n-1} \)```
“Iterative” Controller

**Expert (IN, MLP, etc.):** Model of the world that evaluates proposed controls

**Memory (LSTM):** Encodes the full history of controls and opinions
“Iterative” Controller

**Expert (IN, MLP, etc.):**
Model of the world that evaluates proposed controls

**Memory (LSTM):**
Encodes the full history of controls and opinions
“Iterative” Controller

**Expert (IN, MLP, etc.):**
Model of the world that evaluates proposed controls

**Memory (LSTM):**
Encodes the full history of controls and opinions
Iterative controller
Iterative controller
This is using a learned expert, which is trained simultaneously with the agent.
Metacontroller
Metacontroller

Manager (MLP): Meta-level policy that determines whether to send the proposed control to the world, or to an expert, in order to minimize performance loss + resource loss.
Metacontroller

**Manager (MLP):** Meta-level policy that determines whether to send the proposed control to the world, or to an expert, in order to **minimize** performance loss + resource loss.

**Experts (IN, MLP, etc.):** Different models of the world, each with different resource costs.

[Diagram of the Metacontroller system]
**Metacontroller**

**Manager (MLP):** Meta-level policy that determines whether to send the proposed control to the world, or to an expert, in order to **minimize performance loss + resource loss**

**Experts (IN, MLP, etc.):** Different models of the world, each with **different resource costs**

---

Jessica Hamrick (@jhamrick)
**Metacontroller**

**Manager (MLP):** Meta-level policy that determines whether to send the proposed control to the world, or to an expert, in order to **minimize** performance loss + resource loss.

**Experts (IN, MLP, etc.):** Different models of the world, each with **different** resource costs.

---

Jessica Hamrick (@jhamrick)

DeepMind
**Metacontroller**

**Manager (MLP):** Meta-level policy that determines whether to send the proposed control to the world, or to an expert, in order to **minimize** performance loss + resource loss

**Experts (IN, MLP, etc.):** Different models of the world, each with different resource costs
Metacontroller

Manager (MLP): Meta-level policy that determines whether to send the proposed control to the world, or to an expert, in order to minimize performance loss + resource loss.

Experts (IN, MLP, etc.): Different models of the world, each with different resource costs.
Metacontroller

**Manager (MLP):** Meta-level policy that determines whether to send the proposed control to the world, or to an expert, in order to minimize performance loss + resource loss

**Experts (IN, MLP, etc.):** Different models of the world, each with different resource costs
Metacontroller

**Manager (MLP):** Meta-level policy that determines whether to send the proposed control to the world, or to an expert, in order to minimize **performance loss** + **resource loss**

**Experts (IN, MLP, etc.):** Different models of the world, each with **different resource costs**
Metacontroller: simple task
Metacontroller: simple task
Metacontroller: hard task
Metacontroller: hard task
Imagination-Based Planner

(Pascaru, Li, et al., 2017)
Imagination-Based Planner

(Pascanu, Li, et al., 2017)
Takeaways
Takeaways

1. It can be useful to think about the *choice of computation* (which then allows e.g. choosing between multiple models).
Takeaways

1. It can be useful to think about the *choice of computation* (which then allows e.g. choosing between multiple models).

2. Enables a natural tradeoff between model-free and model-based computation by choosing the *amount of computation* being performed.
Takeaways

1. It can be useful to think about the choice of computation (which then allows e.g. choosing between multiple models).

2. Enables a natural tradeoff between model-free and model-based computation by choosing the amount of computation being performed.

3. Building in these choices is a type of structured computation that goes beyond the distinction of simply having a model or not having a model.
Outline

1. **Structured Computation**: the way that individual computations or functions are composed into more complex structures.
   
   *Hamrick, Ballard, Pascanu, Vinyals, Heess, & Battaglia (2017, ICLR)*

2. **Structured Representation**: the format of the data that computations are performed over.
   
Outline

1. **Structured Computation**: the way that individual computations or functions are composed into more complex structures.
   
   Hamrick, Ballard, Pascanu, Vinyals, Heess, & Battaglia (2017, ICLR)

2. **Structured Representation**: the format of the data that computations are performed over.
   
Structure in Deep Networks

- **Fully-Connected Layer**
  - Unstructured computation
  - Unstructured representation

- **Convolutional Layer**
  - Structured computation
  - Unstructured representation

- **Recurrent Layer**
  - Structured computation
  - Unstructured representation
Graph Networks
Graph Networks

Gori et al. (2005), Scarselli et al. (2005), Scarselli et al. (2009), Li et al. (2015), Gilmer et al. (2017)
Graph Networks

Gori et al. (2005), Scarselli et al. (2005), Scarselli et al. (2009), Li et al. (2015), Gilmer et al. (2017)
Graph Networks

Gori et al. (2005), Scarselli et al. (2005), Scarselli et al. (2009), Li et al. (2015), Gilmer et al. (2017)

1. Takes graphs as input, return graphs as output
Graph Networks

Gori et al. (2005), Scarselli et al. (2005), Scarselli et al. (2009), Li et al. (2015), Gilmer et al. (2017)

1. Takes graphs as input, return graphs as output
2. Invariant to the permutation of the nodes and edges
Graph Networks

Gori et al. (2005), Scarselli et al. (2005), Scarselli et al. (2009), Li et al. (2015), Gilmer et al. (2017)

1. Takes graphs as input, return graphs as output
2. Invariant to the permutation of the nodes and edges
3. Invariant to the number of nodes and edges
Graph Networks

Gori et al. (2005), Scarselli et al. (2005), Scarselli et al. (2009), Li et al. (2015), Gilmer et al. (2017)
Graph Networks

Gori et al. (2005), Scarselli et al. (2005), Scarselli et al. (2009), Li et al. (2015), Gilmer et al. (2017)

<table>
<thead>
<tr>
<th>Edges</th>
<th>Nodes</th>
<th>Globals</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$V$</td>
<td>$u$</td>
</tr>
</tbody>
</table>
Graph Networks

Gori et al. (2005), Scarselli et al. (2005), Scarselli et al. (2009), Li et al. (2015), Gilmer et al. (2017)

Edges $E$  Nodes $V$  Globals $u$

Edge update

$e'_{i \rightarrow j} = \phi_e (v_i, v_j, e_{i \rightarrow j}, u)$
Graph Networks

Gori et al. (2005), Scarselli et al. (2005), Scarselli et al. (2009), Li et al. (2015), Gilmer et al. (2017)

\[
E 
\]

\[
V 
\]

\[
\mathbf{u} 
\]

Edge update

\[
e'_{i \rightarrow j} = \phi_e \left( \mathbf{v}_i, \mathbf{v}_j, e_{i \rightarrow j}, \mathbf{u} \right)
\]

Node update

\[
\mathbf{v}'_i = \phi_v \left( \mathbf{v}_i, \sum_j e'_{j \rightarrow i}, \mathbf{u} \right)
\]
Graph Networks

Gori et al. (2005), Scarselli et al. (2005), Scarselli et al. (2009), Li et al. (2015), Gilmer et al. (2017)

<table>
<thead>
<tr>
<th>Edges</th>
<th>Nodes</th>
<th>Globals</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$V$</td>
<td>$u$</td>
</tr>
</tbody>
</table>

Edge update

$$e'_{i \rightarrow j} = \phi_e(v_i, v_j, e_{i \rightarrow j}, u)$$

Node update

$$v'_i = \phi_v(v_i, \sum_j e'_{j \rightarrow i}, u)$$

 Globals update

$$u' = \phi_u(\sum_i v'_i, \sum_{i,j} e'_{i \rightarrow j}, u)$$
The Gluing Task

Goal: glue blocks together to make the tower stable, using the minimum amount of glue.

Joint work with Kelsey Allen (MIT)
The Gluing Task

Goal: glue blocks together to make the tower stable, using the minimum amount of glue.

Joint work with Kelsey Allen (MIT)
The Gluing Task

Goal: glue blocks together to make the tower stable, using the minimum amount of glue.

Joint work with Kelsey Allen (MIT)
The Gluing Task

Goal: glue blocks together to make the tower stable, using the minimum amount of glue.

Joint work with Kelsey Allen (MIT)
Instructions (press 'h' to show/hide)
1. Click on a block (or the floor) to select it.
2. Click on another block (or the floor) to glue them together.
3. Press enter to apply gravity to the tower.
4. You earn 1pt for each block that doesn't fall.
5. Each pair of blocks that is glued costs 1pt.
6. If you use the minimum glue to keep the tower stable, you earn a 10pt bonus.
7. At least one glue is needed for each tower.
Instructions (press 'h' to show/hide)
1. Click on a block (or the floor) to select it.
2. Click on another block (or the floor) to glue them together.
3. Press enter to apply gravity to the tower.
4. You earn 1pt for each block that doesn't fall.
5. Each pair of blocks that is glued costs 1pt.
6. If you use the minimum glue to keep the tower stable, you earn a 10pt bonus.
7. At least one glue is needed for each tower.
Learning a Policy Over the Edges of a Graph
Learning a Policy Over the Edges of a Graph
Learning a Policy Over the Edges of a Graph

![Graph Diagram]

\[ G_0 \rightarrow GN_1 \rightarrow G_1 \rightarrow GN_2 \rightarrow \cdots \rightarrow GN_M \rightarrow G_M \]

\[ G_0 \rightarrow GN_{\text{core}} \times M \rightarrow G_M \]
Learning a Policy Over the Edges of a Graph

\[ G_0 \rightarrow G_{N_1} \rightarrow G_1 \rightarrow G_{N_2} \rightarrow \cdots \rightarrow G_{N_M} \rightarrow G_M \]

\[ G_0 \rightarrow G_{N_{core}} \times M \rightarrow G_M \rightarrow \pi(E) \]
Learning a Policy Over the Edges of a Graph

See also:
- NerveNet (Wang et al., 2017)
- Graph Network Models for Continuous Control (Sanchez-Gonzalez et al., 2018)
- Relational Deep RL (Zambaldi et al., 2018)
Agent Variations
(Trained & tested on towers of size 2-10 blocks)

**Human**: human baseline

**MLP**: multilayer perceptron agent

**GN-FC**: fully connected graph network agent
(nodes=blocks, edges=all-to-all)

**GN**: sparse graph network agent
(nodes=blocks, edges=contacts)
Results

(a)

<table>
<thead>
<tr>
<th>Model</th>
<th>Total Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>900</td>
</tr>
<tr>
<td>MLP</td>
<td>615</td>
</tr>
<tr>
<td>GN-FC</td>
<td>1505</td>
</tr>
<tr>
<td>GN</td>
<td>1689</td>
</tr>
</tbody>
</table>

Optimal
Results

(a) Total Reward

(b) Scaled Reward

No Glue

Optimal
Generalization Results

Scaled reward

-0.2  0.0  0.2  0.4  0.6  0.8  1.0

GN  GN-FC  MLP

7 blocks

GN  GN-FC  MLP

10 blocks

Train

Test

DeepMind

Jessica Hamrick (@jhamrick)
Generalization Results

(Train on 2-6, 8, & 9 blocks, test on 7 & 10)
Generalization Results

(Train on 2-6, 8, & 9 blocks, test on 7 & 10)
Generalization Results

(Train on 2-6, 8, & 9 blocks, test on 7 & 10)
Generalization Results

(Train on 2-6, 8, & 9 blocks, test on 7 & 10)
Force Propagation
Force Propagation

Edge update
Force Propagation

Edge update

Node update
Force Propagation

Edge update

Node update
Force Propagation

Edge update

Node update
Force Propagation

Edge update

Node update
Takeaways
Takeaways

1. Model-free RL with an *appropriate representation* of the data can be highly effective, even when it comes to transfer!
Takeaways

1. Model-free RL with an *appropriate representation* of the data can be highly effective, even when it comes to transfer!

2. Combining the right structured computation with the right structured representation can lead to the *emergence* of behavior that looks model-based.
Outline

1. **Structured Computation**: the way that individual computations or functions are composed into more complex structures.
   
   *Hamrick, Ballard, Pascanu, Vinyals, Heess, & Battaglia (2017, ICLR)*

2. **Structured Representation**: the format of the data that computations are performed over.
   
   
Other Structured Approaches
Other Structured Approaches

Structured computation that integrates hierarchical and model-based RL (e.g. Value Prediction Networks, Oh et al. 2017)
Other Structured Approaches

Structured computation that integrates hierarchical and model-based RL
(e.g. Value Prediction Networks, Oh et al. 2017)

Structured computation that separates inference from planning
(e.g. QMDP-Net, Karkus et al. 2017)
Other Structured Approaches

Structured computation that integrates hierarchical and model-based RL
(e.g. Value Prediction Networks, Oh et al. 2017)

Structured computation that separates inference from planning
(e.g. QMDP-Net, Karkus et al. 2017)

Structured computation which converts unstructured representations to structured representations
(e.g. Relational Neural Expectation Maximization, van Steenkiste et al., 2018)
Other Structured Approaches

Structured computation that integrates hierarchical and model-based RL (e.g. Value Prediction Networks, Oh et al. 2017)

Structured computation that separates inference from planning (e.g. QMDP-Net, Karkus et al. 2017)

Structured computation which converts unstructured representations to structured representations (e.g. Relational Neural Expectation Maximization, van Steenkiste et al., 2018)

Structured representations which are inferred and adapted online (e.g. Neural Relational Inference, Kipf et al. 2018)
Other Structured Approaches

Structured computation that integrates hierarchical and model-based RL
(e.g. Value Prediction Networks, Oh et al. 2017)

Structured computation that separates inference from planning
(e.g. QMDP-Net, Karkus et al. 2017)

Structured computation which converts unstructured representations to structured representations
(e.g. Relational Neural Expectation Maximization, van Steenkiste et al., 2018)

Structured representations which are inferred and adapted online
(e.g. Neural Relational Inference, Kipf et al. 2018)

Structured representations which encode stronger notions of hierarchy
(e.g. Hierarchical Relation Networks, Mrowca et al., 2018)
Flexibility, adaptivity and generalization are about *having the right structure*. 
Flexibility, adaptivity and generalization are about having the right structure.
Flexibility, adaptivity and generalization are about *having the right structure*. 
Thanks!

Peter Battaglia
Kelsey Allen
Victor Bapst
Tina Zhu
Kevin McKee
Josh Tenenbaum
Andy Ballard
Razvan Pascanu
Oriol Vinyals
Nicolas Heess
Alvaro Sanchez-Gonzalez
Yujia Li
Matt Botvinick
Theo Weber
David Reichert
Tobias Pfaff
DeepMind

Hamrick, Ballard, Pascanu, Vinyals, Heess, & Battaglia (2017)


*equal contribution

Extra Slides
Reactive Controller Results

![Graph showing performance loss for different numbers of planets. The x-axis represents the number of planets, and the y-axis represents performance loss. The graph includes lines for one planet, two planets, three planets, four planets, and five planets. Each line is color-coded: one planet is blue, two planets are green, three planets are red, four planets are purple, and five planets are yellow.](image-url)
Reactive Controller Results

![Graph showing performance loss with number of planets]

- **Performance Loss**
  - **Number of simulations**: 51
  - **Performance Loss**: 0.5
Reactive Controller Results

![Diagram showing performance loss for different numbers of planets]
Reactive Controller Results

Performance Loss

- one planet
- two planets
- three planets
- four planets
- five planets

Number of simulations: 51
Reactive Controller Results

Performance Loss

- one planet
- two planets
- three planets
- four planets
- five planets

Number of simulations: 51
Reactive Controller Results

Performance Loss

- one planet
- two planets
- three planets
- four planets
- five planets

Number of simulations: 51
Reactive Controller Results

Number of simulations vs. Performance Loss for different numbers of planets:
- One planet
- Two planets
- Three planets
- Four planets
- Five planets

Performance Loss:
- 0.8
- 0.7
- 0.6
- 0.5
- 0.4
- 0.3
- 0.2
- 0.1
- 0.0

0.5
Iterative Controller Results

Performance Loss vs. Number of simulations

- Blue dots: one planet
- Green dots: two planets
- Red dots: three planets
- Purple dots: four planets
- Orange dots: five planets

Number of simulations:
- 0.5
- 52
Iterative Controller Results

Performance Loss

- one planet
- two planets
- three planets
- four planets
- five planets
Iterative Controller Results

This is using a learned expert, which is trained simultaneously with the agent.

![Graph showing performance loss over number of simulations for different numbers of planets: one planet, two planets, three planets, four planets, five planets. The y-axis represents performance loss ranging from 0.0 to 0.8, and the x-axis represents the number of simulations ranging from 0 to 10.]
Iterative Controller Results

![Graph showing Iterative Controller Results](image-url)
Iterative Controller Results

![Graphs showing Iterative Controller Results](image)

- **True simulation expert**
- **Int. Net. expert**
  - Number of simulations vs. Performance Loss
  - Lines represent different numbers of planets:
    - one planet
    - two planets
    - three planets
    - four planets
    - five planets
Iterative Controller Results

- True simulation expert
  - one planet
  - two planets
  - three planets
  - four planets
  - five planets

- Int. Net. expert

- MLP expert
Metacontroller Results

Effect of difficulty (IN expert)
Metacontroller Results

Effect of difficulty (IN expert)

![Graph showing the effect of difficulty on the number of metacontroller ponder steps and reactive controller loss for different ponder costs.]

Low ponder cost ($\tau = 0.01$)

Medium ponder cost ($\tau = 0.06$)

High ponder cost ($\tau = 0.25$)
Metacontroller Results

Total cost (five planets)

Number of simulations

Performance Loss

Int. Net. expert
Metacontroller Results

Total cost (five planets)

Number of simulations: 56
Metacontroller Results

Total cost (five planets)

Int. Net. expert

Sum of performance and resource loss

Zero resource cost

Number of simulations

Total Cost

0.0
0.2
0.4
0.6
0.8
1.0

0
1
2
3
4
5
6
7
8
9
10

DeepMind
Metacontroller Results

Total cost (five planets)

Int. Net. expert

Number of simulations

Total Cost
Metacontroller Results

Total cost (five planets)

Number of simulations

Different resource costs

Int. Net. expert

Total Cost

0.0
0.2
0.4
0.6
0.8
1.0

0 1 2 3 4 5 6 7 8 9 10
Metacontroller Results

Total cost (five planets)

Int. Net. expert

Different resource costs

Number of simulations

Total Cost

0.0

0.2

0.4

0.6

0.8

1.0
Metacontroller Results

Multiple experts (MLP + IN)
Metacontroller Results
Multiple experts (MLP + IN)

Total number of simulations

Increasing cost of MLP

Increasing cost of Int. Net.
Metacontroller Results

Multiple experts (MLP + IN)

Total number of simulations

Increasing cost of MLP

Increasing cost of Int. Net.

Fraction of sims using MLP expert

Increasing cost of Int. Net.

Increasing cost of MLP

62
Training the Controller and Memory
Training the Manager
Training the Experts

Diagram showing the relationships between various components:
- Scene
- History (n-1)
- Manager
- Controller
- Action
- Control
- Switch
- World
- Expert 1
- Expert 2
- ... Expert K
- Outcome
- Performance loss
- Opinions
- Memory
- History (n)
- Scene
- History (n-1)
Training the Critic

Diagram showing the integration of a Critic with other components such as Manager, Controller, Expert 1, Expert 2, ..., Expert K, Memory, Scene, History, and Opinion. The Critic predicts performance loss and influences the decision-making process.