REWARD MACHINES FOR COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING

Cyrus Neary, Zhe Xu, Bo Wu, and Ufuk Topcu
The University of Texas at Austin
Task decomposition in multi-agent reinforcement learning

Cooperative multi-agent RL: A team of agents learn interact in a shared environment to achieve a common objective.
Task decomposition in multi-agent reinforcement learning

**Cooperative multi-agent RL:** A team of agents learn interact in a shared environment to achieve a common objective.
Task decomposition in multi-agent reinforcement learning

**Cooperative multi-agent RL:** A team of agents learn to interact in a shared environment to achieve a common objective.

**Observation:** Often, agents only interact in several crucial moments of the task.
Task decomposition in multi-agent reinforcement learning

Cooperative multi-agent RL: A team of agents learn interact in a shared environment to achieve a common objective.

Objectives:
1) Present a framework for specifying structured representations of team tasks.

Observation: Often, agents only interact in several crucial moments of the task.
Task decomposition in multi-agent reinforcement learning

Cooperative multi-agent RL: A team of agents learn interact in a shared environment to achieve a common objective.

**Objectives:**

1) Present a framework for specifying structured representations of team tasks.
2) Use this specification to decompose problem into necessary individual behaviors.

**Observation:** Often, agents only interact in several crucial moments of the task.
Task decomposition in multi-agent reinforcement learning

Cooperative multi-agent RL: A team of agents learn interact in a shared environment to achieve a common objective.

<table>
<thead>
<tr>
<th>Objectives:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Present a framework for specifying structured representations of team tasks.</td>
</tr>
<tr>
<td>2) Use this specification to decompose problem into necessary individual behaviors.</td>
</tr>
<tr>
<td>3) Present a reinforcement learning algorithm that uses decomposition to simplify multi-agent learning.</td>
</tr>
</tbody>
</table>

Observation: Often, agents only interact in several crucial moments of the task.
An Illustrative Running Example

Team objective:
Have $A_1$ safely reach the goal location $Goal$. 
An Illustrative Running Example

Team objective:
Have $A_1$ safely reach the goal location $Goal$.

Colored walls block the agents’ paths.
An Illustrative Running Example

Team objective: Have $A_1$ safely reach the goal location $Goal$. 

Colored walls block the agents’ paths. 

Buttons remove the wall of the corresponding color.
An Illustrative Running Example

Team objective:
Have $A_1$ safely reach the goal location $Goal$.  

Colored walls block the agents’ paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.
An Illustrative Running Example

Team objective:
Have $A_1$ safely reach the goal location $Goal$.

Colored walls block the agents’ paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.

Environment states $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \mathcal{S}_3$

Environment actions $\mathcal{A} = A_1 \times A_2 \times A_3$

Set of team environment states
Local states of agent 3
Set of team action
Actions available to agent 3
An Illustrative Running Example

Team objective:
Have $A_1$ safely reach the goal location $Goal$.

Colored walls block the agents’ paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.
An Illustrative Running Example

**Team objective:**
Have $A_1$ safely reach the goal location $Goal$.

Colored walls block the agents’ paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.
An Illustrative Running Example

Team objective:
Have $A_1$ safely reach the goal location $Goal$.

Colored walls block the agents’ paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.
An Illustrative Running Example

Team objective:
Have $A_1$ safely reach the goal location $Goal$.

Colored walls block the agents’ paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.
Representing Temporally Extended Team Tasks – Reward Machines
Representing Temporally Extended Team Tasks – Reward Machines

\[
U = \{u_1, u_1, u_2, \ldots, u_7\} \quad \text{– Set of states}
\]
Representing Temporally Extended Team Tasks – Reward Machines

\[ U = \{u_1, u_1, u_2, ..., u_7\} \quad \text{— Set of states} \]
\[ \Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{-R_B}, A_3^{R_B}, A_3^{-R_B}, \text{Goal}\} \quad \text{— Set of events} \]
Representing Temporally Extended Team Tasks – Reward Machines

$U = \{u_1, u_1, u_2, ..., u_7\}$ — Set of states

$\Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, Goal\}$ — Set of events

$\delta : U \times \Sigma \rightarrow U$ — Transition function
Representing Temporally Extended Team Tasks – Reward Machines

\[ U = \{u_1, u_1, u_2, \ldots, u_7\} \] — Set of states
\[ \Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, Goal\} \] — Set of events
\[ \delta : U \times \Sigma \rightarrow U \] — Transition function
\[ \sigma : U \times U \rightarrow \mathbb{R} \] — Output function
\[ F \] — Set of final states
Representing Temporally Extended Team Tasks – Reward Machines

\[ \Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, \text{Goal}\} \]
Representing Temporally Extended Team Tasks – Reward Machines

\[
\Sigma = \{ Y_B, G_B, R_B, A_2^{RB}, A_2^{\neg RB}, A_3^{RB}, A_3^{\neg RB}, \text{Goal} \}
\]
Representing Temporally Extended Team Tasks – Reward Machines

$$\Sigma = \{ Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, \text{Goal} \}$$
Representing Temporally Extended Team Tasks – Reward Machines

\[ \Sigma = \{ Y_B, G_B, R_B, A_2^{ RB }, A_2^{- RB }, A_3^{ RB }, A_3^{- RB }, \text{Goal} \} \]
Representing Temporally Extended Team Tasks – Reward Machines

\[ \Sigma = \{Y_B, G_B, R_B, A_2^{RB}, A_2^{RB}, A_3^{RB}, A_3^{RB}, A_3^{RB}, \text{Goal} \} \]
Connecting Environment States to a Reward Machine

The labeling function: Relate the environment state to collections of high-level events.

High-Level Task
(Model: Reward Machine)

- Reward machine states $U$
- High-level Events $\Sigma$
Connecting Environment States to a Reward Machine

The labeling function: Relate the environment state to collections of high-level events.

- Environment states $S = S_1 \times S_2 \times S_3$
- Environment actions $A = A_1 \times A_2 \times A_3$

High-Level Task (Model: Reward Machine)

- Reward machine states $U$
- High-level Events $\Sigma$

Environment Dynamics Model: Transition distribution $p(\cdot \mid s, a)$
Connecting Environment States to a Reward Machine

The labeling function: Relate the environment state to collections of high-level events.

- Reward machine states $U$
- High-level Events $\Sigma$

- Environment states $S = S_1 \times S_2 \times S_3$
- Environment actions $A = A_1 \times A_2 \times A_3$
Using Reward Machines for Reinforcement Learning

Perform q-learning to learn a separate policy for each state of the reward machine.
Using Reward Machines for Reinforcement Learning

Perform q-learning to learn a separate policy for each state of the reward machine.
Using Reward Machines for Reinforcement Learning

Perform q-learning to learn a separate policy for each state of the reward machine.
Using Reward Machines for Reinforcement Learning

Perform q-learning to learn a separate policy for each state of the reward machine.
Using Reward Machines for Reinforcement Learning

Perform q-learning to learn a separate policy for each state of the reward machine.
Using Reward Machines for Reinforcement Learning

Perform q-learning to learn a separate policy for each state of the reward machine.
Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_1 \subseteq \Sigma$?

Example: Event subset for $A_1$ is $\Sigma_1 = \{Y_B, R_B, Goal\}$
Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_1 \subseteq \Sigma$?

- Transitions triggered by missing events are not observed.

Example: Event subset for $A_1$ is $\Sigma_1 = \{Y_B, R_B, Goal\}$

- Transitions triggered by missing events are not observed.
Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_1 \subseteq \Sigma$?

Example: Event subset for $A_1$ is $\Sigma_1 = \{Y_B, R_B, Goal\}$

- Transitions triggered by missing events are not observed.
- Merge states that cannot be differentiated by events from $\Sigma_1$. 
Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_1 \subseteq \Sigma$?

Projected Reward Machine $R_1$

- Transitions triggered by missing events are not observed.
- Merge states that cannot be differentiated by events from $\Sigma_1$. 

![Projected Reward Machine](image)
Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_1 \subseteq \Sigma$?

Note: reward machine projections may be computed automatically.
Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_i \subseteq \Sigma$?

Projected reward machines encode the sub-tasks of individual agents who only observe events in $\Sigma_i$.

Note: reward machine projections may be computed automatically.
Problem Equivalence

When is the task described by the team reward machine equivalent to the composition of its projections?

Team Reward Machine $R$

Projected Reward Machine $R_1$

Projected Reward Machine $R_2$

... Projected Reward Machine $R_N$
Problem Equivalence

When is the task described by the team reward machine equivalent to the composition of its projections?

Bisimulation: Behavioral equivalence of reward machines

Parallel composition: Concurrent combination of reward machines
Local Labeling Functions

Connecting environment dynamics with projected reward machines?

Team Reward Machine $R$

Labeling function $L$

Team environment states

Environment Dynamics

$\Rightarrow$ Projected Reward Machine $R_1$ \quad $\Rightarrow$ Projected Reward Machine $R_2$ \quad $\Rightarrow$ ... \quad $\Rightarrow$ Projected Reward Machine $R_N$
Local Labeling Functions

Connecting environment dynamics with projected reward machines?

Environment Dynamics → Team Reward Machine $R$ → Labeling function $L$ → Local environment states

Project $\rightarrow$ Local labeling function $L_1$ → Local environment states

Project $\rightarrow$ Local labeling function $L_2$ → Local environment states

Project $\rightarrow$ Local labeling function $L_N$ → Local environment states

Local environment states → Team environment states
Local Labeling Functions

Connecting environment dynamics with projected reward machines?

- Team Reward Machine $R$
- Projected Reward Machine $R_1$
- Projected Reward Machine $R_2$
- ... ...
- Projected Reward Machine $R_N$

- Labeling function $L$
- Local labeling function $L_1$
- Local labeling function $L_2$
- ... ...
- Local labeling function $L_N$

- Synchronization on shared events

Environment Dynamics

Team environment states

Local environment states
Problem Equivalence

Task complete $\iff$ Subtask 1 complete $\land$ Subtask 2 complete $\land$ ... $\land$ Subtask $N$ complete

Team Reward Machine $R$ $\Rightarrow$ Projected Reward Machine $R_1$ $\Rightarrow$ Projected Reward Machine $R_2$ $\Rightarrow$ ... $\Rightarrow$ Projected Reward Machine $R_N$

Labeling function $L$ $\Rightarrow$ Local labeling function $L_1$ $\Rightarrow$ Local labeling function $L_2$ $\Rightarrow$ ... $\Rightarrow$ Local labeling function $L_N$

Local environment states

Team environment states

Environment Dynamics
Problem Equivalence

Task complete $\iff$ Subtask 1 complete $\land$ Subtask 2 complete $\land$ ... $\land$ Subtask $N$ complete

Team Reward Machine $R$ $\equiv$ Projected Reward Machine $R_1$ $\equiv$ Projected Reward Machine $R_2$ $\equiv$ ... $\equiv$ Projected Reward Machine $R_N$

Labeling function $L$ $\mapsto$ function $L_1$ $\mapsto$ function $L_2$ $\mapsto$ ... $\mapsto$ function $L_N$

Observation: Agent $i$ may use $R_i$ to learn its subtask, without observing the states of its teammates.
Decentralized Q-Learning with Projected Reward Machines (DQPRM)

Agents learn to accomplish their subtasks in the absence of teammates.
Decentralized Q-Learning with Projected Reward Machines (DQPRM)

Agents learn to accomplish their subtasks in the absence of teammates.
Decentralized Q-Learning with Projected Reward Machines (DQPRM)

Agents learn to accomplish their subtasks in the absence of teammates.
Decentralized Q-Learning with Projected Reward Machines (DQPRM)

While ensuring the resulting composite behavior accomplishes the team task.
Decentralized Q-Learning with Projected Reward Machines (DQPRM)

While ensuring the resulting composite behavior **accomplishes the team task.**
Decentralized Q-Learning with Projected Reward Machines (DQPRM)

Agents learn to accomplish their subtasks in the absence of teammates.
Decentralized Q-Learning with Projected Reward Machines (DQPRM)

While ensuring the resulting composite behavior **accomplishes the team task.**
Decentralized Q-Learning with Projected Reward Machines (DQPRM)

While ensuring the resulting composite behavior **accomplishes the team task.**

**Team reward machine** $R$

**Comparisons to baselines (lower is better)**

![Diagram showing A_1, A_2, A_3, G, and a team reward machine R with comparisons to baselines.](image)
DQPRM Scaling with the Number of Agents

Rendezvous Experiment

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₂</th>
<th>A₄</th>
<th>G₅</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A₃</td>
<td></td>
<td>G₃</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>A₆</td>
<td></td>
<td>A₈</td>
<td></td>
</tr>
<tr>
<td>G₈</td>
<td></td>
<td>A₁₀</td>
<td></td>
</tr>
<tr>
<td>A₇</td>
<td></td>
<td>G₂</td>
<td></td>
</tr>
<tr>
<td>G₁₀</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A₅</td>
<td>A₉</td>
<td>G₁</td>
<td>G₄</td>
</tr>
</tbody>
</table>

**Team Task:** Each agent must *simultaneously* occupy the rendezvous location $R$, before proceeding to their respective goals $G_i$. 
DQPRM Scaling with the Number of Agents

Rendezvous Experiment

<table>
<thead>
<tr>
<th></th>
<th>A₁</th>
<th>A₂</th>
<th>A₄</th>
<th>G₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>A₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A₆</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G₈</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A₇</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G₁₀</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A₅</td>
<td></td>
<td>A₉</td>
<td>G₁</td>
<td>G₄</td>
</tr>
<tr>
<td>G₉</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A₁₀</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Team Task:** Each agent must *simultaneously* occupy the rendezvous location \( R \), before proceeding to their respective goals \( G_i \).

- Two agent rendezvous.
- Ten agent rendezvous.
Rendezvous Experiment

**Team Task:** Each agent must *simultaneously* occupy the rendezvous location \( R \), before proceeding to their respective goals \( G_i \).

- Two agent rendezvous.
- Ten agent rendezvous.
DQPRM Scaling with the Number of Agents

Rendezvous Experiment

Team Task: Each agent must *simultaneously* occupy the rendezvous location $R$, before proceeding to their respective goals $G_i$.

- Two agent rendezvous.
- Ten agent rendezvous.
DQPRM Scaling with the Number of Agents

Rendezvous Experiment

Team Task: Each agent must *simultaneously* occupy the rendezvous location $R$, before proceeding to their respective goals $G_i$.

- Two agent rendezvous.
- Ten agent rendezvous.
DQPRM scales well with the number of agents.

Each agent learns in the absence of its teammates.
Each agent learns in the absence of its teammates.

Composite behavior accomplishes the team task.

DQPRM scales well with the number of agents.
DQPRM scales well with the number of agents.

Two agent rendezvous

Ten agent rendezvous
Reward machines for MARL task decomposition

Reward machines to **specify** and **decompose** cooperative RL tasks.

Cyrus Neary
cneary@utexas.edu
Reward machines for MARL task decomposition

Reward machines to **specify** and **decompose** cooperative RL tasks.

Conditions guaranteeing equivalence between original and decomposed tasks.

Cyrus Neary
cneary@utexas.edu
Reward machines for MARL task decomposition

Reward machines to **specify** and **decompose** cooperative RL tasks.

Conditions guaranteeing equivalence between original and decomposed tasks.

Decentralized Q-learning algorithm that trains each agent separately.

Cyrus Neary
cneary@utexas.edu
Reward machines for MARL task decomposition

Reward machines to **specify** and **decompose** cooperative RL tasks.

Conditions guaranteeing equivalence between original and decomposed tasks.

Decentralized Q-learning algorithm that trains each agent separately.

Cyrus Neary
cneary@utexas.edu