

Reinforcement Learning for Automated Negotiation

SCML as an Example

Yasser Mohammad^{1, 2, 3}

¹ NEC Corporation, Global Innovation Unit

²National Institute of Advanced Industrial Science and Technology (AIST), Japan

³Assiut University, Egypt

November 28, 2023



AI 2023

AJCAI 2023 Tutorial Updated on November 28, 2023

Outline

1 Theoretical Foundations

2 The Competition

3 RL for SCML

4 Development Environment

5 References

Why Now?

- ① Industries are moving online.
- ② Automation: Factory floor → The back office.
- ③ Human-Human Negotiation is cumbersome, and inefficient.
- ④ Automated Negotiation opens new possibilities:
 - Too fast for people: Repeated smart contracts.
 - Too large for people: complete supply chains



Negotiation in SCM Business

- Human negotiations lead to an estimated 17-40% *value leakage* in some estimates ¹

¹KPMG report: <https://bit.ly/3kDRy6l>

²Forrester report: <https://bit.ly/3nwXEaY>

³UN/CEFACT Project website: <https://bit.ly/38LOsLX>

⁴Y. Mohammad et al. "Supply Chain Management World: A benchmark environment for situated negotiations". In: *Proceedings of the 22nd International Conference on Principles and Practice of Multi-Agent Systems*. 2019.

Negotiation in SCM Business

- Human negotiations lead to an estimated 17-40% *value leakage* in some estimates ¹
- A recent study suggests that at least 15 companies are working in *contracting support systems* ².

¹KPMG report: <https://bit.ly/3kDRy6l>

²Forrester report: <https://bit.ly/3nwXEaY>

³UN/CEFACT Project website: <https://bit.ly/38LOsLX>

⁴Mohammad et al., "Supply Chain Management World: A benchmark environment for situated negotiations".

Negotiation in SCM Business

- Human negotiations lead to an estimated 17-40% *value leakage* in some estimates ¹
- A recent study suggests that at least 15 companies are working in *contracting support systems* ².
- A recent UNECE UN/CEFACT proposal to standardize negotiation protocols for SCM and other applications ³

¹KPMG report: <https://bit.ly/3kDRy6l>

²Forrester report: <https://bit.ly/3nwXEaY>

³UN/CEFACT Project website: <https://bit.ly/38LOsLX>

⁴Mohammad et al., "Supply Chain Management World: A benchmark environment for situated negotiations".

Negotiation in SCM Business

- Human negotiations lead to an estimated 17-40% *value leakage* in some estimates ¹
- A recent study suggests that at least 15 companies are working in *contracting support systems* ².
- A recent UNECE UN/CEFACT proposal to standardize negotiation protocols for SCM and other applications ³
- More to come⁴.

CONTRACTROOM

pactum

¹KPMG report: <https://bit.ly/3kDRy6l>

²Forrester report: <https://bit.ly/3nwXEaY>

³UN/CEFACT Project website: <https://bit.ly/38LOsLX>

⁴Mohammad et al., "Supply Chain Management World: A benchmark environment for situated negotiations".

The Automated Negotiation Challenge

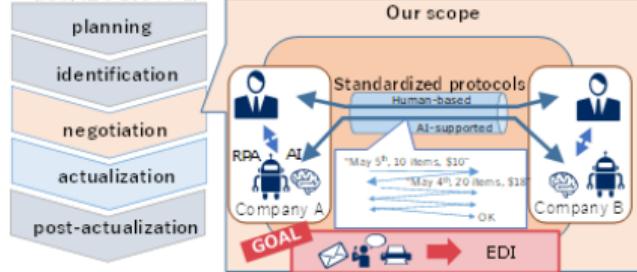
Why is it hard?

- Mechanism Design Problem:
 - Better than haggling?
- Negotiator Design Problem:
 - Generality × Effectiveness

Why is it interesting?

- Easy to state yet hard to solve.
- Multiple levels of abstraction and complexity.
- Several concrete open questions.
- Vibrant yet not saturated research space.

Five fundamental activities of a business transaction (ISO/IEC 15944-1)



attribution: UNECE eNegotiation Project



Automated Negotiating Agents Competition: 2010-

Tutorial Outline

① Theoretical Session (55min) Break (5min)

- Theoretical Foundation
- The Competition

② Development Environment (15min) Break (5min)

③ RL for SCML (35min)

④ Concluding Remarks (5min)

Materials

- ① Tutorial Website: <http://yasserm.com/tutorial-ajcai2023/>
- ② Github Repository: <https://github.com/yasserfarouk/ajcai2023autoneg>
- ③ Handouts: <https://github.com/yasserfarouk/ajcai2023autoneg/raw/main/ajcai2023autoneg.pdf>
- ④ Negmas Documentation: <https://negmas.readthedocs.io>
- ⑤ SCML Documentation: <https://scml.readthedocs.io>
- ⑥ SCML Competition: <https://scml.cs.brown.edu>

Outline

1 Theoretical Foundations

- Basics Game Theory
- Automated Negotiation Basics
- Negotiation With Incomplete Information
- Reinforcement Learning
- RL for Automated Negotiation

2 The Competition

3 RL for SCML

4 Development Environment

5 References

Outline

1 Theoretical Foundations

- Basics Game Theory
- Automated Negotiation Basics
- Negotiation With Incomplete Information
- Reinforcement Learning
- RL for Automated Negotiation

2 The Competition

3 RL for SCML

4 Development Environment

5 References

Motivation

Goal

define and understand the automated negotiation problem in relation to other AI problems.

Summary

- AN in general is a General Game Playing problem in which you are **not** told the game.
- Depending on how much prior information, AN can vary in difficulty.

Types of Games

Copmplete Information Game

- All game parameters are **common knowledge** except player's policies.
- All information sets has a single node.

Imperfect Information Game

- Some partner infomration is unknown.
- Some information sets has multiple nodes.

Incomplete Information Game

- The game is not fully specified (e.g. unknown opponent ufun)
- Can be converted to an Imperfect information game by adding **Nature** as a player.

Outline

1 Theoretical Foundations

- Basics Game Theory
- **Automated Negotiation Basics**
 - Visualizing a Negotiation
 - Negotiation With Incomplete Information
 - Reinforcement Learning
 - RL for Automated Negotiation

2 The Competition

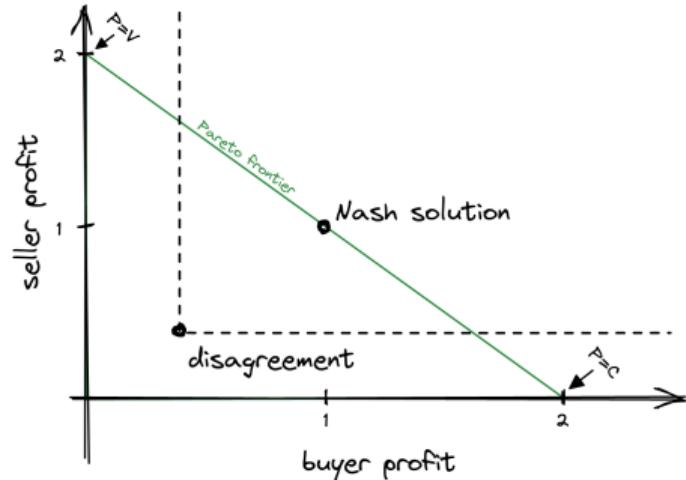
3 RL for SCML

4 Development Environment

5 References

A Simple Trading Problem

- A buyer values a good at V
- A seller can create the good at cost C
- If $V > C$, then there is surplus $V - C$ to be gained (**value creation**)
- Bargaining problem: how much should the buyer pay the seller for the good? (**value division**)
- We might also assume there is an outside option (e.g., eBay), if the negotiation breaks down (i.e., they do not reach an agreement):
 - The buyer (seller) can buy (sell) the good elsewhere for slightly less than V (more than C)



Sketch by Jackson de Campos

No-go Theorem

Desiderata:

- Efficient outcome.
- Individual rationality (IR).
- Incentive compatible (IC).
- Budget balance (BB).

Myerson-Satterthwaite Impossibility Result

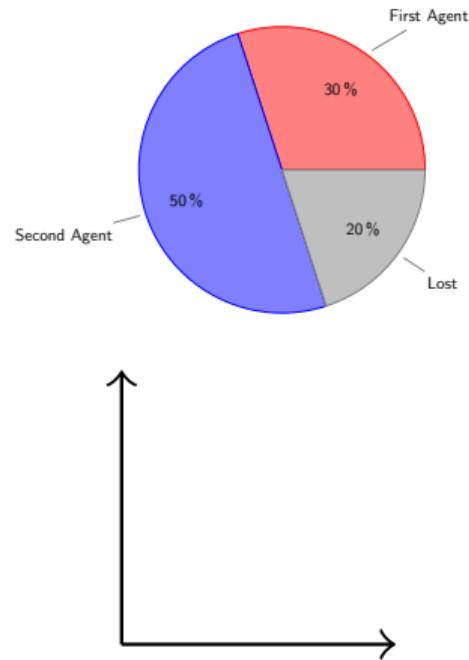
Theorem:⁵ No mechanism can achieve all four of these desiderata.

- A buyer values a good at V .
- A seller can create the good at cost C .
- $V (C)$ is private information, known only to the buyer (seller).
- There is no IR, IC, and BB mechanism that results in agreement, for all $V > C$ values.

⁵Roger B Myerson and Mark A Satterthwaite. "Efficient mechanisms for bilateral trading". In: *Journal of economic theory* 29.2 (1983), pp. 265–281.

Nash Bargaining Game: Description

A single-step full-information bilateral negotiation with $\Omega = [0, 1]^2$ and two utility functions $(\tilde{u}_1, \tilde{u}_2)$ such that:

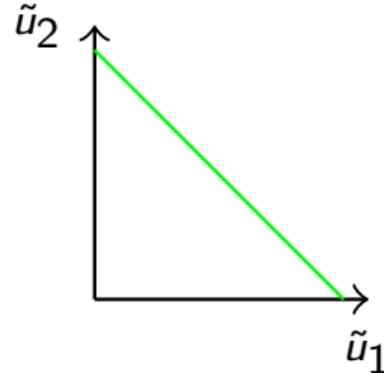
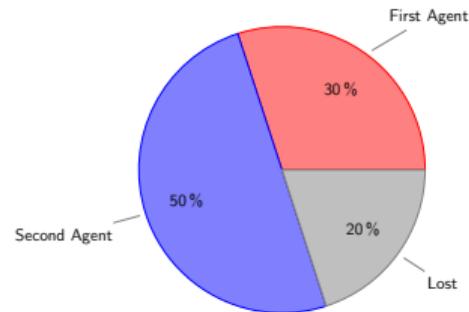


Nash Bargaining Game: Description

A single-step full-information bilateral negotiation with $\Omega = [0, 1]^2$ and two utility functions $(\tilde{u}_1, \tilde{u}_2)$ such that:

- A feasible set of agreements F . A common example is to define F as all the outcomes for which the total utility received by negotiators is less than or equal to one:

$$F = \{(\omega_1, \omega_2) | \tilde{u}_2(\omega_2) + \tilde{u}_1(\omega_1) \leq 1\}.$$

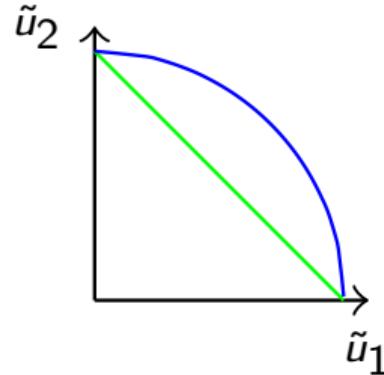
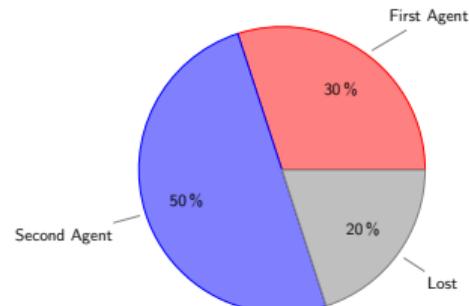


Nash Bargaining Game: Description

A single-step full-information bilateral negotiation with $\Omega = [0, 1]^2$ and two utility functions $(\tilde{u}_1, \tilde{u}_2)$ such that:

- A feasible set of agreements F . A common example is to define F as all the outcomes for which the total utility received by negotiators is less than or equal to one:

$$F = \{(\omega_1, \omega_2) | \tilde{u}_2(\omega_2) + \tilde{u}_1(\omega_1) \leq 1\}.$$

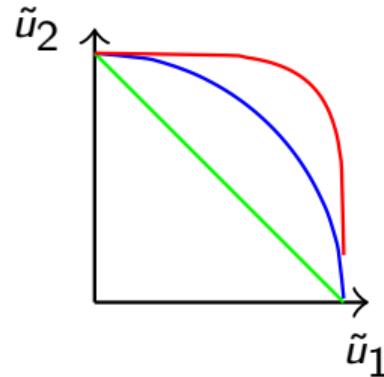
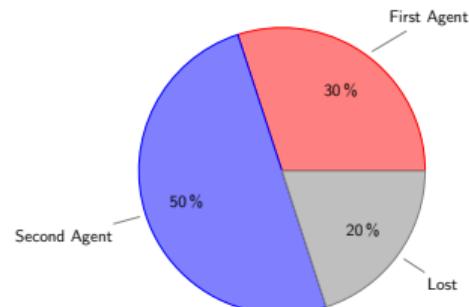


Nash Bargaining Game: Description

A single-step full-information bilateral negotiation with $\Omega = [0, 1]^2$ and two utility functions $(\tilde{u}_1, \tilde{u}_2)$ such that:

- A feasible set of agreements F . A common example is to define F as all the outcomes for which the total utility received by negotiators is less than or equal to one:

$$F = \{(\omega_1, \omega_2) | \tilde{u}_2(\omega_2) + \tilde{u}_1(\omega_1) \leq 1\}.$$



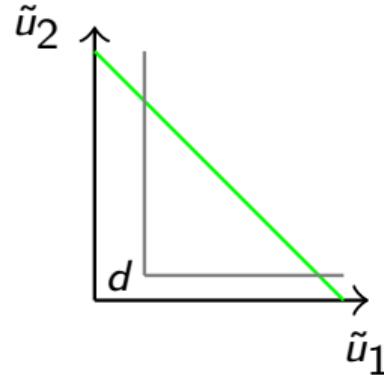
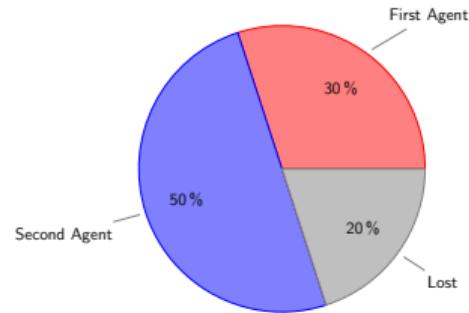
Nash Bargaining Game: Description

A single-step full-information bilateral negotiation with $\Omega = [0, 1]^2$ and two utility functions $(\tilde{u}_1, \tilde{u}_2)$ such that:

- A feasible set of agreements F . A common example is to define F as all the outcomes for which the total utility received by negotiators is less than or equal to one:

$$F = \{(\omega_1, \omega_2) | \tilde{u}_2(\omega_2) + \tilde{u}_1(\omega_1) \leq 1\}.$$

- A disagreement point $d \equiv \tilde{u}_1(\phi) + \tilde{u}_2(\phi) \in \Re^2$ which is the utility value received by the two players in case of disagreement (reserved values).



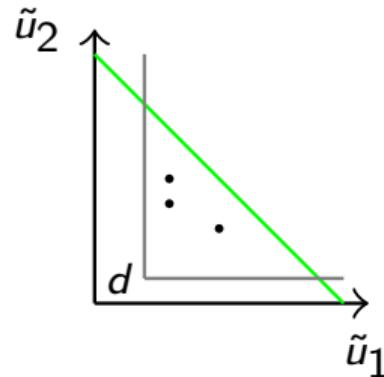
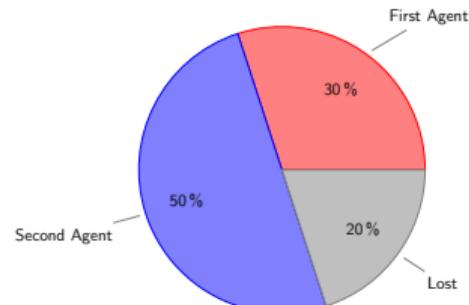
Nash Bargaining Game: Description

A single-step full-information bilateral negotiation with $\Omega = [0, 1]^2$ and two utility functions $(\tilde{u}_1, \tilde{u}_2)$ such that:

- A feasible set of agreements F . A common example is to define F as all the outcomes for which the total utility received by negotiators is less than or equal to one:

$$F = \{(\omega_1, \omega_2) | \tilde{u}_2(\omega_2) + \tilde{u}_1(\omega_1) \leq 1\}.$$

- A disagreement point $d \equiv \tilde{u}_1(\phi) + \tilde{u}_2(\phi) \in \Re^2$ which is the utility value received by the two players in case of disagreement (reserved values).



Other Bargaining Solutions

- Nash Bargaining Solution (1950): The point at which the product of surplus utility (above reservation value) of negotiators is maximized

$$\arg \max_{\omega_1, \omega_2} \prod_{i=1}^2 (u_i(\omega_i) - u_i(\phi))$$

- Kalai-Smorodinsky Bargaining Solution (1975): The Pareto outcome with equal ratios of achieved surplus utility and maximum feasible surplus utility

$$\arg \max_{\omega_1, \omega_2 \in F} (\omega_1 + \omega_2) \text{ s.t. } \left(\frac{u_1(\omega_1) - u_1(\phi)}{u_2(\omega_2) - u_2(\phi)} \right) = \left(\frac{\max_{v \in F} (u_1(v)) - u_1(\phi)}{\max_{v \in F} (u_2(v)) - u_2(\phi)} \right)$$

- Kalai Bargaining Solution (1977): The Pareto outcome maximizing the utility for the unfortunate player. Defining P as the Pareto front

$$\arg \max_{\omega_1, \omega_2 \in P} \min_{i \in \{1,2\}} (u_i(\omega_i) - u_i(\phi))$$

Rubinstein's Bargaining Protocol: Description

The Game

- Two agents sharing a pie.

Rubinstein's Bargaining Protocol: Description

The Game

- Two agents sharing a pie.
- Each agent is under a different time-pressure: $u_i^{t+\Delta}(\omega) < u_i^t(\omega)$. Examples of time-pressure:

Rubinstein's Bargaining Protocol: Description

The Game

- Two agents sharing a pie.
- Each agent is under a different time-pressure: $u_i^{t+\Delta}(\omega) < u_i^t(\omega)$. Examples of time-pressure:

Exponential $u_i^{t+\Delta}(\omega) = \delta_i^\Delta u_i^t(\omega)$.

Linear $u_i^{t+\Delta}(\omega) = u_i^t(\omega) - \Delta c_i$

Rubinstein's Bargaining Protocol: Description

The Game

- Two agents sharing a pie.
- Each agent is under a different time-pressure: $u_i^{t+\Delta}(\omega) < u_i^t(\omega)$. Examples of time-pressure:
 - Exponential $u_i^{t+\Delta}(\omega) = \delta_i^\Delta u_i^t(\omega)$.
 - Linear $u_i^{t+\Delta}(\omega) = u_i^t(\omega) - \Delta c_i$
- Agent's initial utility is the assigned part of the pie: $u_i^0 = \omega_i$.

Rubinstein's Bargaining Protocol: Description

The Game

- Two agents sharing a pie.
- Each agent is under a different time-pressure: $u_i^{t+\Delta}(\omega) < u_i^t(\omega)$. Examples of time-pressure:
 - Exponential $u_i^{t+\Delta}(\omega) = \delta_i^\Delta u_i^t(\omega)$.
 - Linear $u_i^{t+\Delta}(\omega) = u_i^t(\omega) - \Delta c_i$
- Agent's initial utility is the assigned part of the pie: $u_i^0 = \omega_i$.
- Time pressure and utility information are common knowledge.

Rubinstein's Bargaining Protocol: Description

The Game

- Two agents sharing a pie.
- Each agent is under a different time-pressure: $u_i^{t+\Delta}(\omega) < u_i^t(\omega)$. Examples of time-pressure:
 - Exponential $u_i^{t+\Delta}(\omega) = \delta_i^\Delta u_i^t(\omega)$.
 - Linear $u_i^{t+\Delta}(\omega) = u_i^t(\omega) - \Delta c_i$
- Agent's initial utility is the assigned part of the pie: $u_i^0 = \omega_i$.
- Time pressure and utility information are common knowledge.
- No externally imposed time-limit.

Rubinstein's Bargaining Protocol: Description

The Game

- Two agents sharing a pie.
- Each agent is under a different time-pressure: $u_i^{t+\Delta}(\omega) < u_i^t(\omega)$. Examples of time-pressure:
 - Exponential $u_i^{t+\Delta}(\omega) = \delta_i^\Delta u_i^t(\omega)$.
 - Linear $u_i^{t+\Delta}(\omega) = u_i^t(\omega) - \Delta c_i$
- Agent's initial utility is the assigned part of the pie: $u_i^0 = \omega_i$.
- Time pressure and utility information are common knowledge.
- No externally imposed time-limit.
- Zero reservation value: $u_i^\tau(\phi) = 0 \forall \tau$.

Rubinstein's Bargaining Protocol: Description

The Game

- Two agents sharing a pie.
- Each agent is under a different time-pressure: $u_i^{t+\Delta}(\omega) < u_i^t(\omega)$. Examples of time-pressure:
 - Exponential $u_i^{t+\Delta}(\omega) = \delta_i^\Delta u_i^t(\omega)$.
 - Linear $u_i^{t+\Delta}(\omega) = u_i^t(\omega) - \Delta c_i$
- Agent's initial utility is the assigned part of the pie: $u_i^0 = \omega_i$.
- Time pressure and utility information are common knowledge.
- No externally imposed time-limit.
- Zero reservation value: $u_i^\tau(\phi) = 0 \forall \tau$.

Main Result

There is a unique *sub-game perfect equilibrium* that requires a single negotiation step in most cases.

Rubinstein's Bargaining Protocol: Equilibrium

Exponential Discounting

The negotiation ends in **one step** with the first agent proposing and the second agent accepting *for asymmetric cases*:

$$(\omega_1^*, \omega_2^*) = \left(\frac{1 - \delta_2}{1 - \delta_1 \delta_2}, \frac{\delta_2 (1 - \delta_1)}{1 - \delta_1 \delta_2} \right)$$

Rubinstein's Bargaining Protocol: Equilibrium

Exponential Discounting

The negotiation ends in **one step** with the first agent proposing and the second agent accepting *for asymmetric cases*:

$$(\omega_1^*, \omega_2^*) = \left(\frac{1 - \delta_2}{1 - \delta_1 \delta_2}, \frac{\delta_2 (1 - \delta_1)}{1 - \delta_1 \delta_2} \right)$$

Linear Discounting

The negotiation ends in **one step** with the first agent proposing and the second agent accepting:

$$(\omega_1^*, \omega_2^*) = \begin{cases} (c_2, 1 - c_2) & c_1 > c_2 \\ (x, 1 - x) \quad \forall x \in [c_1, 1] & c_1 = c_2 \\ (1, 0) & c_1 < c_2 \end{cases}$$

Negotiation with Complete Information

Hick's Paradox

Why do rational parties negotiate when they have full information?

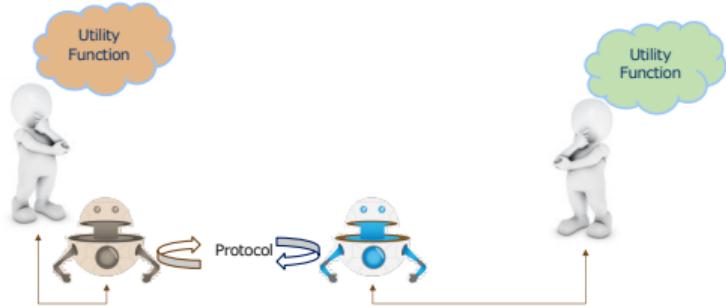
Labor vs. Management

- A union negotiating (in rounds) with management about a wage raise.
- Both parties are perfectly rational and fully informed.
- Threat: the union *can* strike.

Theorem⁶ Subgame perfect equilibria exist in which there is a finite strike followed by agreement.

⁶Raquel Fernandez and Jacob Glazer. *Striking for a bargain between two completely informed agents*. Tech. rep. National Bureau of Economic Research, 1989.

Components of Negotiation



Negotiation Scenario Defines who is negotiating about what

Negotiation Protocol Defines how negotiation is to be conducted

- Alternating Offers Protocol
- Single Text Protocol
- ...

Negotiation Strategy Defines how agents behave during a negotiation

- Time-based strategies: boulware, conceder, ...
- Tit-for-tat variations
- ...

Notation

Negotiation Scenario \mathcal{S}

Agent Indices $\mathcal{A}^{\mathcal{S}}$: The set of n_A agents.

Domain $\mathcal{D}^{\mathcal{S}}$: A tuple $(K, \mathcal{F}^{\mathcal{S}})$ defining the situation:

Outcome Space K : A set of all possible agreements with $\phi \notin K$ representing disagreement. $K^+ \equiv K \cup \{\phi\}$ is called the Extended Outcome Space.

Preferences Tuple $\mathcal{F}^{\mathcal{S}}$: A tuple of n_A members that defines the preferences of each agent engaged in the negotiation.

Information Set Tuple $\mathcal{I}^{\mathcal{S}}$: A tuple of n_A information sets $(\{I_f^{\mathcal{S}}(\mathcal{F}^{\mathcal{S}}) \mid f \in \mathcal{A}^{\mathcal{S}}\})$ where $I_f^{\mathcal{S}}(\mathcal{F}^{\mathcal{S}})$ represents all the information available to agent x about the preferences $\mathcal{F}^{\mathcal{S}}$ of all agents including itself.

Protocol \mathcal{P} : The negotiation protocol.

Negotiation Protocol Execution: $\mathcal{P}(\mathcal{S})$

Protocol

Breaking Rule when the negotiation ends with a timeout.

Action Constraints what actions are valid.

Activation Rule which each agent is activated.

Evaluation Rule the next step:

Continue ▷ Goto breaking rule.

Agreement Succeed with ω .

Disagreement Fail with ϕ .

Agent (Strategy)

Selects an action from the **valid action set**.

Algorithm 1 Negotiation Mechanism Execution: $\mathcal{P}_Y(\lambda)$

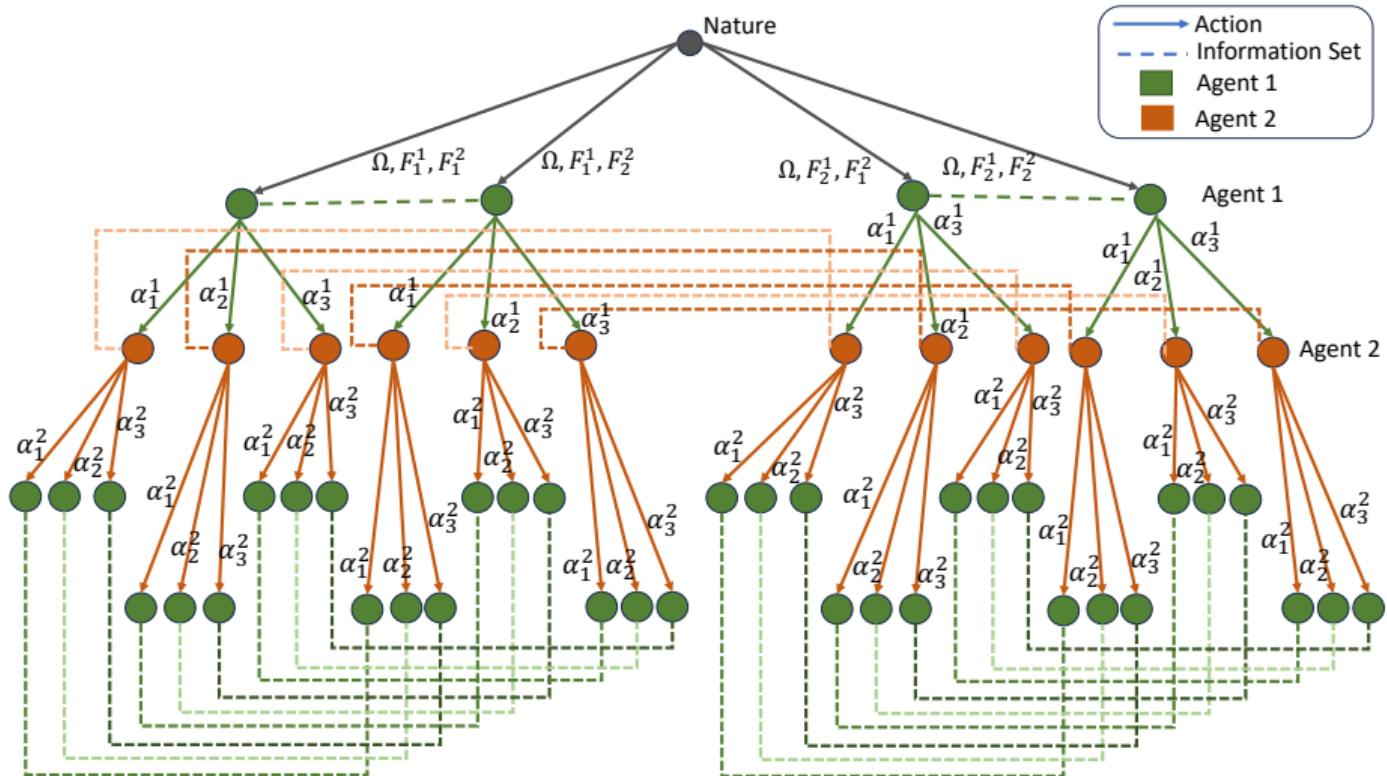
Input: Scenario $\lambda \equiv (\mathcal{N}^\lambda, (\Omega, \mathcal{F}^\lambda), \mathcal{I}^\lambda \equiv \{I_x^\lambda : x \in \mathcal{N}^\lambda\})$,
 Mechanism $\mathcal{P}_Y = (\mathcal{P}, Y) : \mathcal{P} = (\zeta, \chi, \sigma)$

Output: $\omega_* \in \Omega \cup \{\phi\}$

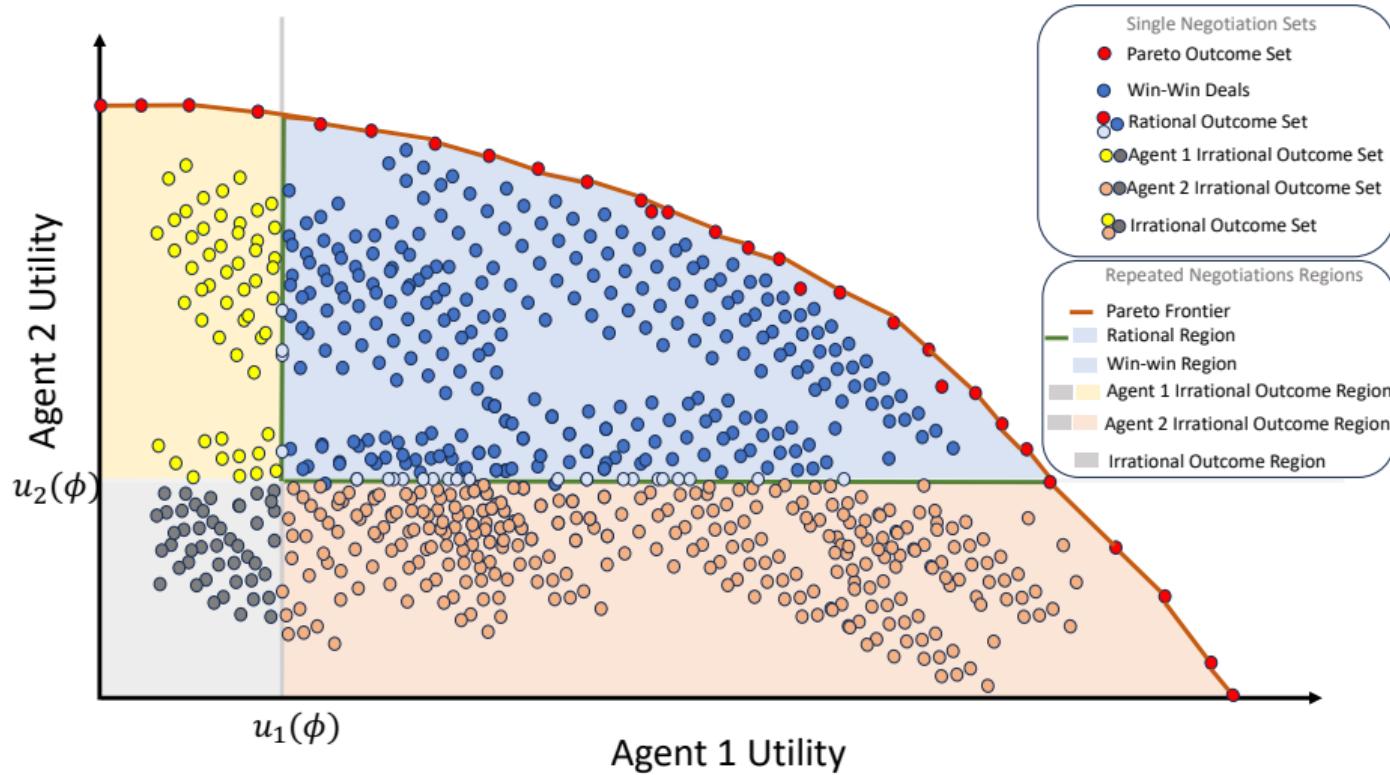
```

1:  $x \approx \pi_x \forall x \in \mathcal{N}^\lambda \leftarrow Y(\lambda)$                                 ▷ Assign strategies
2:  $i, t, t_0, \mathcal{T} \leftarrow 0, 0, \text{clock}(), \langle \rangle$                             ▷ Init step, time and trace
3: while  $\zeta(\mathcal{T}, i, t; \lambda) = 0$  do                                         ▷ check the breaking rule
4:    $\mathcal{N}_i^\lambda \leftarrow \chi(\mathcal{T}, i; \lambda)$                                          ▷ Next active agents
5:   for  $x \in \mathcal{N}_i^\lambda$  do                                              ▷ Activate agents
6:      $\mathbb{A}_x^v \leftarrow \gamma(\mathcal{T}, x; \lambda)$                                          ▷ Valid action set
7:      $\alpha_i^x \leftarrow \pi_x(\mathcal{T}, \mathbb{A}_x^v; I_x^\lambda)$                                ▷ Receive action
8:      $\mathcal{T} \leftarrow \mathcal{T} + \langle \alpha_i^x, x \rangle$                                      ▷ Update the trace
9:      $z \leftarrow \sigma(\mathcal{T}, i; \lambda)$                                          ▷ Evaluation rule
10:    if  $z = \phi$  then return  $\phi$  ▷ End with disagreement
11:    else if  $z \in \Omega$  then return  $\omega_i$                                          ▷ agreement
12:    end if                                         ▷ Otherwise continue to the next step
13:  end for
14:   $i, t \leftarrow i + 1, \text{clock}() - t_0$                                          ▷ Update step and time.
15: end while
16: return  $\phi$                                          ▷ Disagreement on breaking rule returning 1
  
```

Induced Game with Incomplete Information



Visualizing a negotiation



Outline

1 Theoretical Foundations

- Basics Game Theory
- Automated Negotiation Basics
- Negotiation With Incomplete Information
 - Negotiation with Known Acceptance Model
 - Greedy Concession Algorithm
 - Speeding up GCA
 - Extending GCA
- Reinforcement Learning
- RL for Automated Negotiation

2 The Competition

3 RL for SCML

4 Development Environment

5 References

Negotiation with Known Acceptance Model

Scenario

Negotiation with Known Acceptance Model

Scenario

$N \in \mathbb{I}$: The round-limit.

Negotiation with Known Acceptance Model

Scenario

$N \in \mathbb{N}$: The round-limit.

$\Omega \equiv \{\omega\}$: Possible outcomes.

- $\phi \in \Omega$ represents disagreement.

Negotiation with Known Acceptance Model

Scenario

$N \in \mathbb{N}$: The round-limit.

$\Omega \equiv \{\omega\}$: Possible outcomes.

- $\phi \in \Omega$ represents disagreement.

$u : \Omega \rightarrow [0, 1]$ Agent Utility Function.

Negotiation with Known Acceptance Model

Scenario

$N \in \mathbb{I}$: The round-limit.

$\Omega \equiv \{\omega\}$: Possible outcomes.

- $\phi \in \Omega$ represents disagreement.

$u : \Omega \rightarrow [0, 1]$ Agent Utility Function.

$a : \Omega \times [0, N] \rightarrow [0, 1]$ The acceptance model.

Policy (Υ)

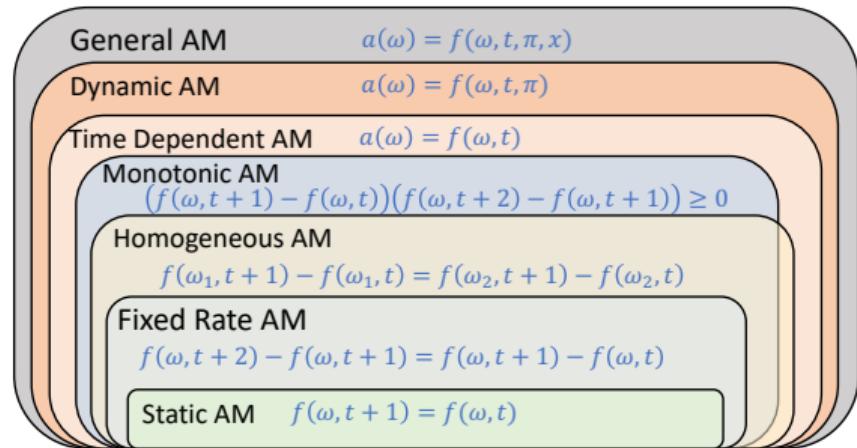
A **deterministic time-dependent** policy $\pi : [1, N] \rightarrow \Omega$ is simply a **sequence** of outcomes

Problem

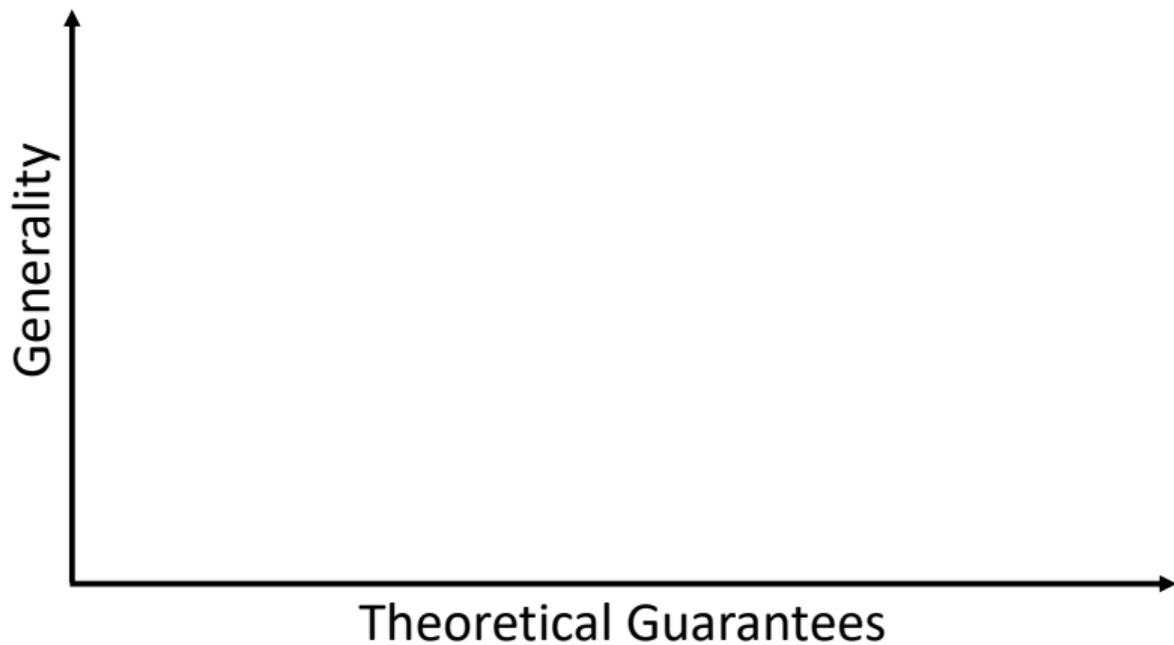
Find the **optimal** deterministic time-dependent policy π^* such that:

$$\pi^* = \arg \max_{\pi} \mathcal{EU}(\pi | \Upsilon)$$

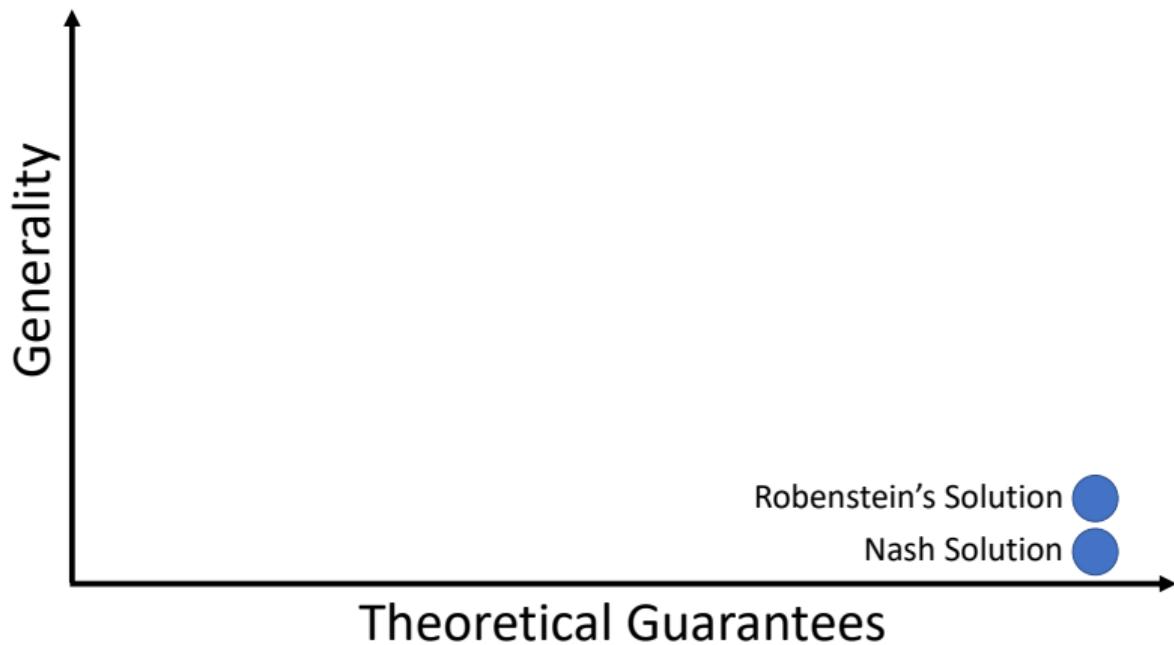
Acceptance Model Progression



Positioning this work



Positioning this work



Positioning this work



Positioning this work



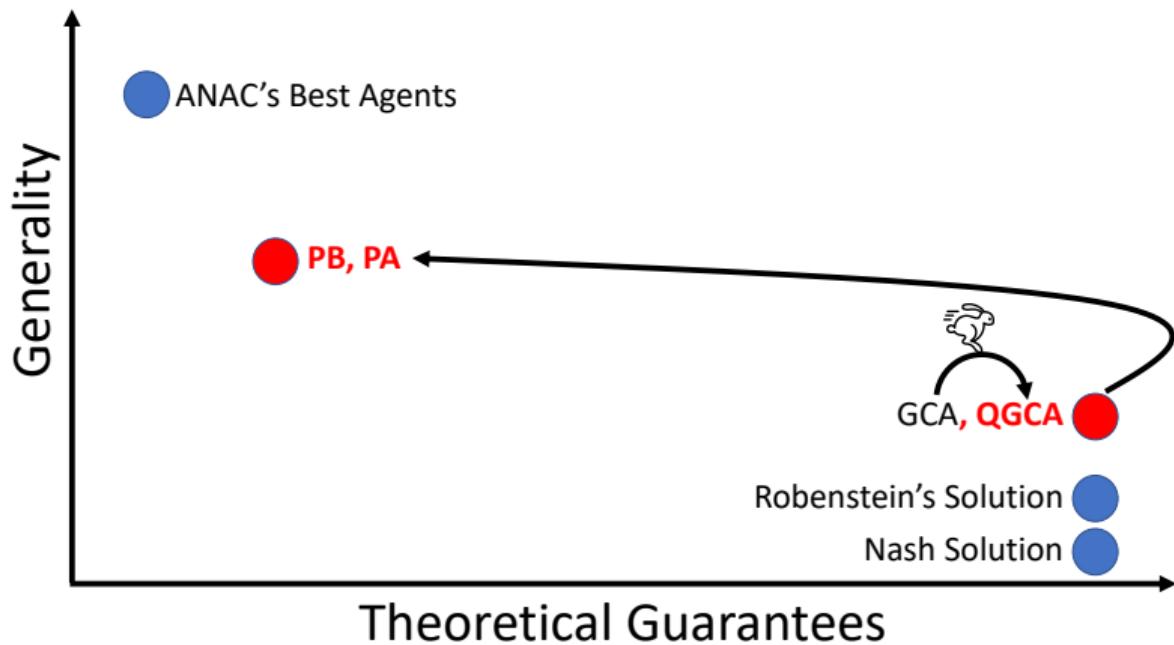
Positioning this work



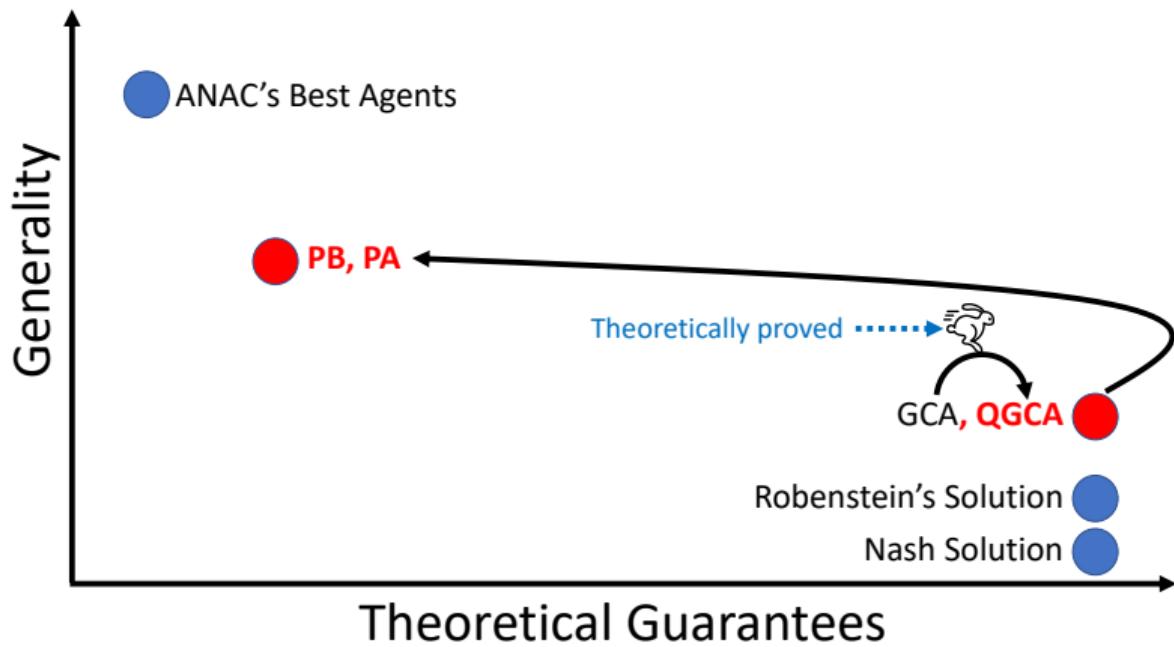
Positioning this work



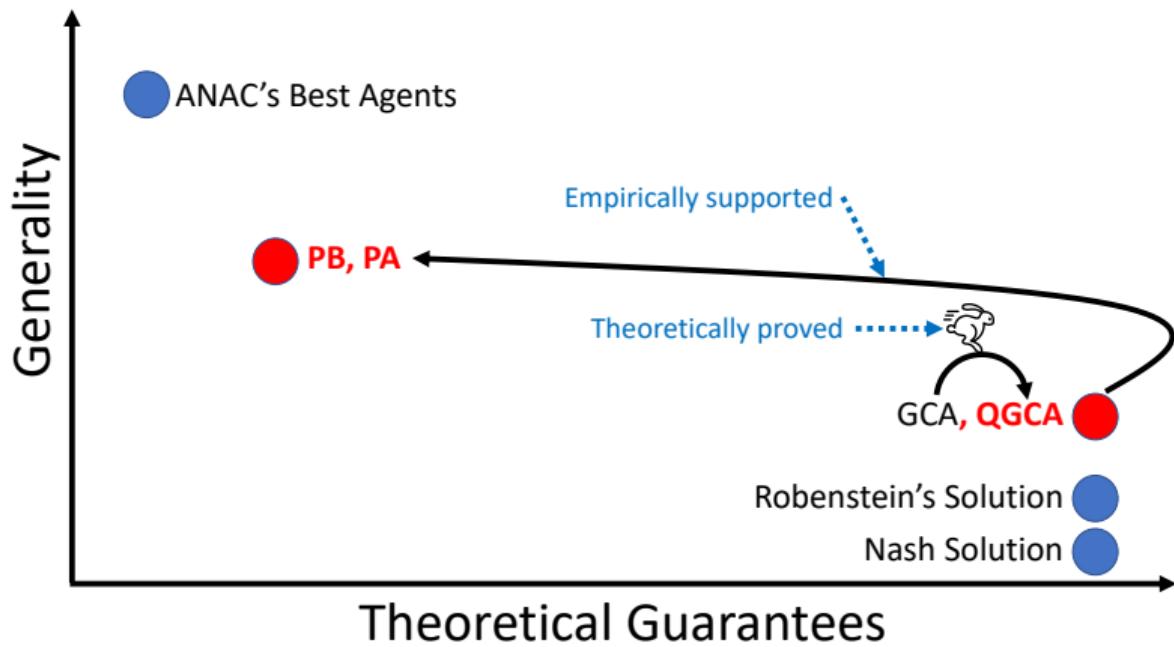
Positioning this work



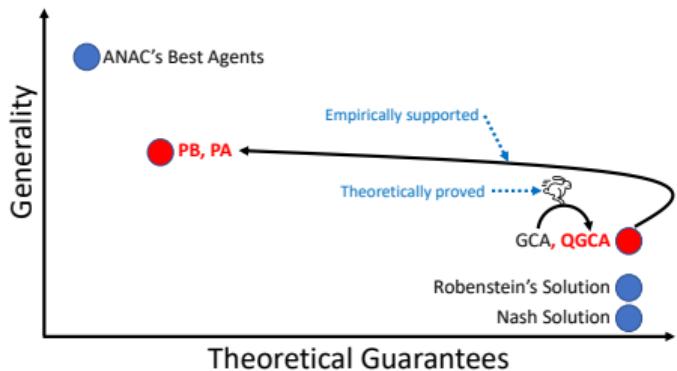
Positioning this work



Positioning this work



Why is this worth doing?



Why are we trying this?

- You cannot argue with .
- Faster calculation of π^* opens new .
- Advance our Fundamental Understanding.
 - When **exactly** is GCA optimal?
 - Why **exactly** is GCA optimal?
- Provide ideas and inspiration for **new strategies** to advance the state of the art.

Greedy Concession Algorithm

Applicability

Designed and proved optimal for negotiations **static** acceptance models, and **no-repetition** of offers^a.

^abaarslag2015gca.

Main Idea

- The optimal policy of length one is known: $\arg \max_{\omega} a(\omega)u(\omega)$
- Optimal policy of length N contains *scrambled* version of the optimal policy of length $N - 1 \rightarrow$ **Optimality of greediness theorem**.
- The optimal policy is a **conceding** policy \rightarrow **the concession lemma**.

Understanding Optimality of GCA

GCA

```
1:  $\pi \leftarrow \langle \rangle$                                 ▷ Empty policy
2: for  $k \leftarrow 1 : N$  do
3:    $\omega^* \leftarrow \arg \max_{\omega \in \Omega - \pi} \mathcal{E}\mathcal{U}(\text{sort}_{\mathcal{E}\mathcal{U}}(\pi \circ \omega))$ 
4:    $\pi \leftarrow \text{sort}_{\mathcal{E}\mathcal{U}}(\pi \circ \omega^*)$            ▷ Operator  $\circ$  inserts an element in a set
```

Understanding Optimality of GCA

GCA

```
1:  $\pi \leftarrow \langle \rangle$                                 ▷ Empty policy
2: for  $k \leftarrow 1 : N$  do
3:    $\omega^* \leftarrow \arg \max_{\omega \in \Omega - \pi} \mathcal{EU}(\text{sort}_{\mathcal{EU}}(\pi \circ \omega))$ 
4:    $\pi \leftarrow \text{sort}_{\mathcal{EU}}(\pi \circ \omega^*)$            ▷ Operator  $\circ$  inserts an element in a set
```

Time Complexity

$$O(NK^2)^a$$

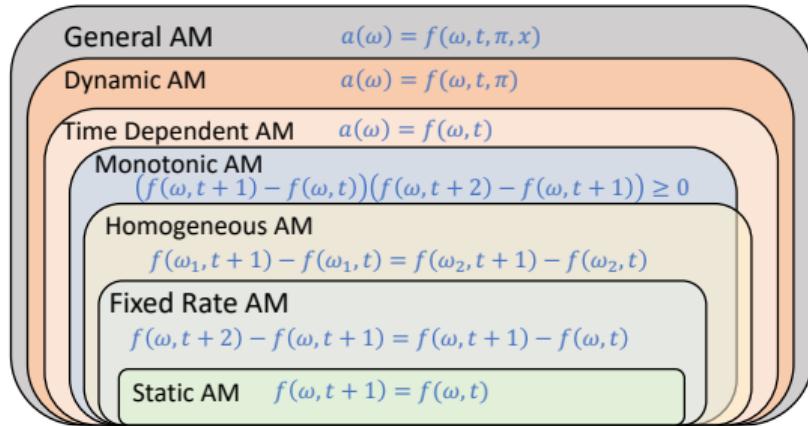
- Linear in negotiation time.
- **Quadratic** in outcome space size.

^atrees do not help here ⓘ

Understanding Optimality of GCA

Can it be extended without modifications?

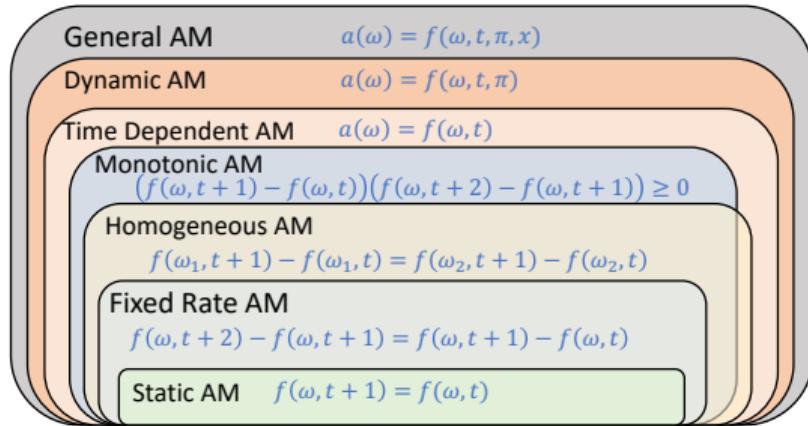
- Allowing repetition →
- Extension **keeping optimality** to more AMs?
 - General TDAM →
 - Monotonic AM →
 - Homogeneous AM →
 - Fixed Rate AM →



Understanding Optimality of GCA

Can it be extended without modifications?

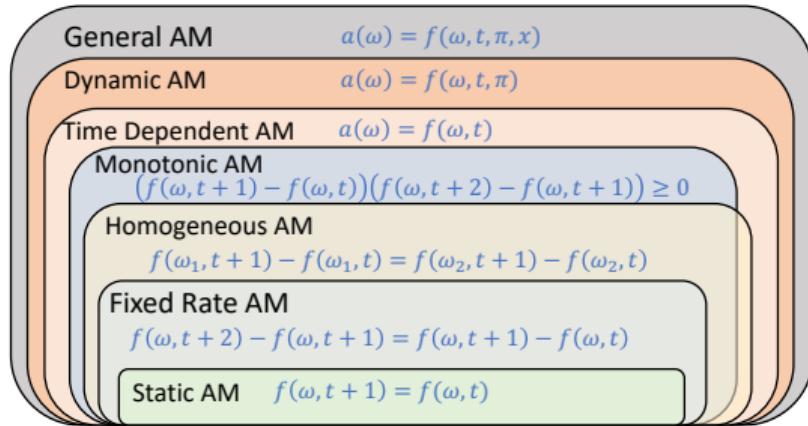
- Allowing repetition → **YES.**
- Extension **keeping optimality** to more AMs?
 - General TDAM →
 - Monotonic AM →
 - Homogeneous AM →
 - Fixed Rate AM →



Understanding Optimality of GCA

Can it be extended without modifications?

- Allowing repetition → **YES.**
- Extension **keeping optimality** to more AMs?
 - General TDAM → **NO.**
 - Monotonic AM → **NO.**
 - Homogeneous AM → **NO.**
 - Fixed Rate AM → **NO.**



Quick GCA

Applicability

Static Acceptance Models.

Quick GCA

Applicability

Static Acceptance Models.

Main Idea

90% of CS is caching ... ^a

- Calculating the \mathcal{EU} of a policy **repeats** many calculations of shorter policies.
- Let's **cache** those.
- Allows us to calculate the effects of various operations on the expected utility in **$O(1)$** time (and space).

^athis is not a quote. Just invented it.

QGCA Details

Cached Quantities

$P : [0, N] \rightarrow [0, 1]$ The aggregate multiplication of probability of rejection used in the evaluation of \mathcal{EU} .

$$P_i = \prod_{j=0}^{i-1} 1 - a(\pi_j) \quad (1)$$

$S : [0, N] \rightarrow [0, 1]$ The cumulative sum of the expected utility value.

$$S_i = \sum_{j=0}^i u(\pi_j) a(\pi_j) P_j = S_{i-1} + u(\pi_i) a(\pi_i) P_i \quad (2)$$

Calculating \mathcal{EU} using ↑

$$\mathcal{EU} = S_N$$

Operations on Policies

Static AM

Operation	Symbol	Effect on $\mathcal{EU}(\pi)$
Replacing π_i with ω	π_i^ω	$P_i(u(\omega)a(\omega, i) - u(\pi_i)a(\pi_i)) + (S_N - S_i) \frac{a(\omega, i) - a(\pi_i, i)}{1 - a(\pi_i, i)}$
Swapping π_i, π_j	$\pi_{i \leftrightarrow j}$	$u_i \left(a_i^j P_j \frac{1 - a_i^j}{1 - a_i^j} - a_i^j P_i \right) + u_j \left(a_j^i P_i - a_j^i P_j \right) + (S_{j-1} - S_{i+1}) \frac{a_i^j - a_j^i}{1 - a_i^j}$
Swapping π_i, π_{i+1}	$\pi_{i \rightarrow 1}$	$u_i \left(a_i^{i+1} P_{i+1} (1 - a_{i+1}^i) - a_i^i P_i \right) + u_{i+1} \left(a_{i+1}^i P_i - a_{i+1}^{i+1} P_{i+1} \right)$

General TDAM

Replacing π_i with ω	π_i^ω	$P_i(u(\omega)a(\omega) - u(\pi_i)a(\pi_i)) + (S_N - S_i) \frac{a(\omega) - a(\pi_i)}{1 - a(\pi_i)}$
Swapping π_i, π_j	$\pi_{i \leftrightarrow j}$	$u_i a_i \left(P_j \frac{1 - a_j}{1 - a_i} - P_i \right) + u_j a_j \left(P_i - P_j \right) + (S_{j-1} - S_{i+1}) \frac{a_i - a_j}{1 - a_i}$
Swapping π_i, π_{i+1}	$\pi_{i \rightarrow 1}$	$P_k a(\pi_i)a(\pi_{i+1}) (u(\pi_{i+1}) - u(\pi_i))$
Inserting outcome ω at location t	$\pi_{\omega @ t}$	$u(\omega)a(\omega)P_t - a(\omega)(S_T - S_{t-1})$

Quick GCA

```

1:  $\pi \leftarrow <>$                                 ▷ Empty linked list
2:  $\mathcal{L}_\omega = 0 \forall \omega \in \Omega$           ▷ Initialize location of all outcomes
3:  $S_{-1}, P_{-1} \leftarrow 0, 1$ 
4: for  $k \leftarrow 1 : N$  do
5:    $d^*, \omega^* \leftarrow -\infty, \phi$ 
6:   for  $\omega \in \Omega$  do
7:      $i \leftarrow \mathcal{L}_\omega$                       ▷ Lookup location of insertion
8:      $d_\omega \leftarrow S_{i-1} + (1 - a(\omega))(\mathcal{E}\mathcal{U}(\pi) - S_{i-1}) + P_i \mathcal{E}\mathcal{U}(\omega) a(\omega)$ 
9:     if  $d_\omega \geq d^*$  then
10:     $d^*, \omega^* \leftarrow d_\omega, \omega$ 
11:     $\pi \leftarrow \pi \circ_{\mathcal{L}_{\omega^*}} \omega^*$            ▷ Insert best outcome in its correct place
12:  Update  $S, P, \mathcal{L}$ 
13:  if no-repetition then
14:     $\Omega \leftarrow \Omega - \{\omega^*\}$ 

```

Time Complexity

$$O(NK)$$

- ➊ Linear in negotiation time.
- ➋ Linear in outcome space size.

Extending GCA to General TDAM

Why cannot we use GCA as it is?

- The concession lemma does not hold.
 - The optimal policy may **NOT** be conceding.
- The greedy-is-optimal theorem does not hold.
 - The optimal policy may **NOT** contain shorter optimal policies (even scrambled).

What to do

- Start by running QGCA → a conceding policy.
 - Use the mean as the SAM.
- Use fast operations to permute this policy for a better one.
 - Policy Bubbling (PB).
- Starting from PB, search for a better policy (i.e. using simulated annealing)
 - Policy Annealing (PA).

Policy Bubbling

```

1:  $\pi \leftarrow \pi^c$                                 ▷ Start from the result of QGCA+
2: for  $r \leftarrow 1 : N$  do
3:    $\pi^- \leftarrow \pi$ 
4:   for  $k \leftarrow 1 : N - 1$  do
5:     if  $\mathcal{EU}(\pi_{k \rightarrow 1}) - \mathcal{EU}(\pi) > 0$  then
6:        $\pi \leftarrow \pi_{k \rightarrow 1}$ 
7:     if  $\pi^- = \pi$  then
8:       return  $\pi^b = \pi$ 
9: return  $\pi^b = \pi$ 

```

Time Complexity

$$O(N^2)$$

Policy Annealing

```
1:  $\pi \leftarrow \pi^b$                                      ▷ Start from the result of PBS
2: for  $r \leftarrow 1 : R$  do
3:   Randomly select a site  $s$  and an outcome  $\omega \in \Omega - \{\pi_s\}$ 
4:    $\delta \leftarrow \mathcal{EU}(\pi_s^\omega) > \mathcal{EU}(\pi)$ 
5:   if  $\delta > 0 \vee \text{rand}() > \exp^{-\delta/\tau(r)}$  then
6:      $\pi \leftarrow \pi_s^\omega$ 
7:   for  $i \leftarrow 1 : N$  except  $s$  do                         ▷ Find the best permutation
8:     if  $\mathcal{EU}(\pi_{i \leftrightarrow s}) > \mathcal{EU}(\pi)$  then
9:        $\pi \leftarrow \pi_{i \leftrightarrow s}$ 
10:  return  $\pi^a = \pi$ 
```

Time Complexity

$$O(NR)$$

Outline

1 Theoretical Foundations

- Basics Game Theory
- Automated Negotiation Basics
- Negotiation With Incomplete Information
- Reinforcement Learning
 - RL In A Nutshell
 - RL Successes?
- RL for Automated Negotiation

2 The Competition

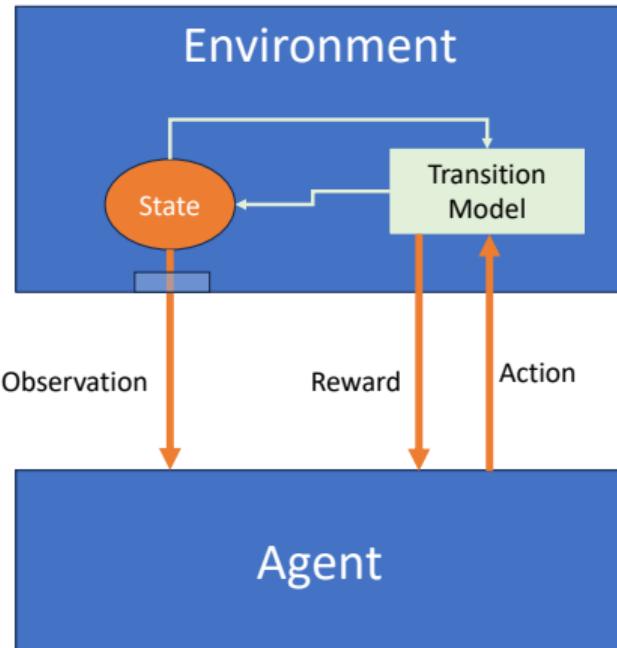
3 RL for SCML

4 Development Environment

5 References

Reinforcement Learning

- One of the most successful ML techniques.
- The agent learns a **policy** that maximizes its **expected reward** by interacting with the environment (or a simulation of it).
- Main Components:
 - **State** Full information about the current state of the environment.
 - **Transition Model** Changes the state based on **current state** and **agent's action**.
 - **Observation** The **state** as *appears* to the agent.
 - **Action** The action generated by the agent's **policy**
 - **Reward** The **feedback** given from the environment to the agent.



RL Successes in Games

- Earlier games solved by RL were: **complete information games**.
 - Chess, Go
- Same method can be used to solve multiple games:
 - Atari
- Some **incomplete information zero-sum games** were also solved.
 - Poker, Stratego
- Next: **incomplete information general-sum**
 - E.g. Automated Negotiation



Outline

1 Theoretical Foundations

- Basics Game Theory
- Automated Negotiation Basics
- Negotiation With Incomplete Information
- Reinforcement Learning
- RL for Automated Negotiation
 - Offer Policy
 - Acceptance Strategy
 - Both

2 The Competition

3 RL for SCML

4 Development Environment

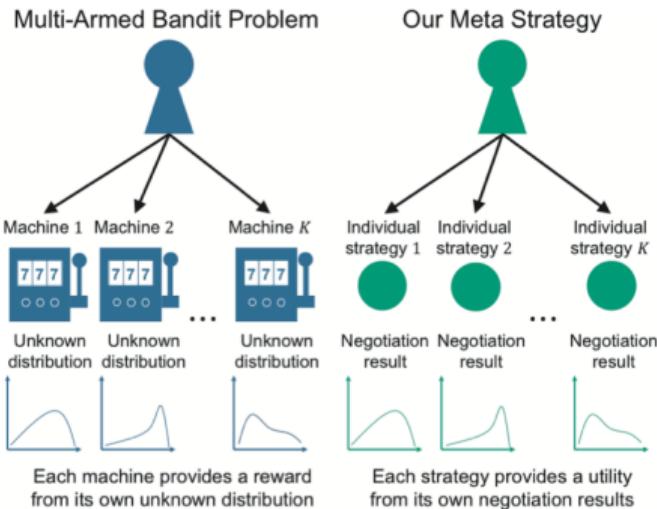
5 References

Multiaremed Bandits for Repeated Negotiations⁷

Treat sub-negotiators as bandits in a standard multi-armed bandits problem.

- Base Strategies: Atlas3, CaduceusDC16, Kawaii, ParsCat, Rubick, YXAgent
- Method:
 - After every negotiation update the corresponding $\hat{\mu}_s$.
 - Use the slot machine (negotiator) that maximizes

$$UCB(s) = \hat{\mu}_s + c \sqrt{\frac{\ln N}{N_s}}$$



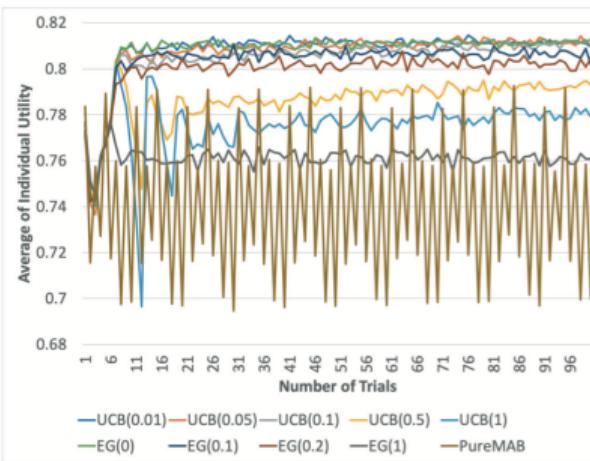
⁷Ryohei Kawata and Katsuhide Fujita. "Meta-Strategy Based on Multi-Armed Bandit Approach for Multi-Time Negotiation". In: *IEICE TRANSACTIONS on Information and Systems* 103.12 (2020), pp. 2540–2548.

Multiaremed Bandits for Repeated Negotiations⁷

Treat sub-negotiators as bandits in a standard multi-armed bandits problem.

- Base Strategies: Atlas3, CaduceusDC16, Kawaii, ParsCat, Rubick, YXAgent
- Method:
 - After every negotiation update the corresponding $\hat{\mu}_s$.
 - Use the slot machine (negotiator) that maximizes

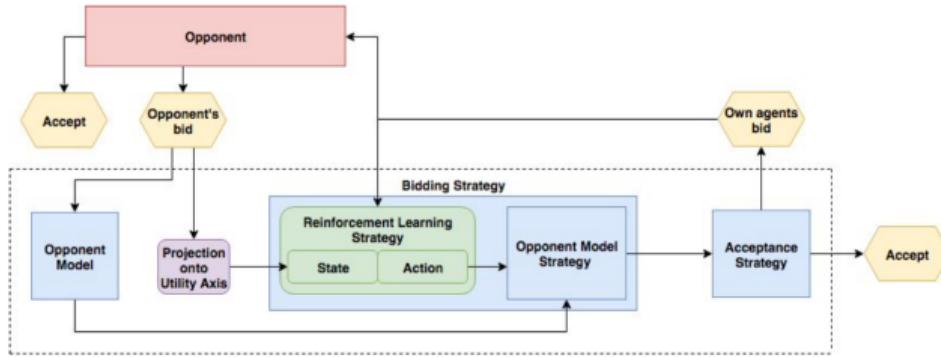
$$UCB(s) = \hat{\mu}_s + c \sqrt{\frac{\ln N}{N_s}}$$



Agent	Individual utility	Social welfare
UCB(0.01)	0.7734	1.4575
<i>Agent33</i>	0.6901	1.4579
<i>AgentNP2018</i>	0.7082	1.4362
<i>Appaloosa</i>	0.7067	1.3706
<i>Ellen</i>	0.6083	1.2223
<i>TimeTraveler</i>	0.7142	1.4573

⁷Kawata and Fujita, "Meta-Strategy Based on Multi-Armed Bandit Approach for Multi-Time Negotiation".

RLBOA: Learning Offering Strategy⁸



Main Points

- Extends the BOA architecture.
- Learns only a bidding strategy:
 - The agent learns how to move *in its own utility axis*.

⁸ Jasper Bakker et al. "RLBOA: A modular reinforcement learning framework for autonomous negotiating agents". In: *Proceedings of the 18th international conference on autonomous agents and multiagent systems*. 2019, pp. 260–268.

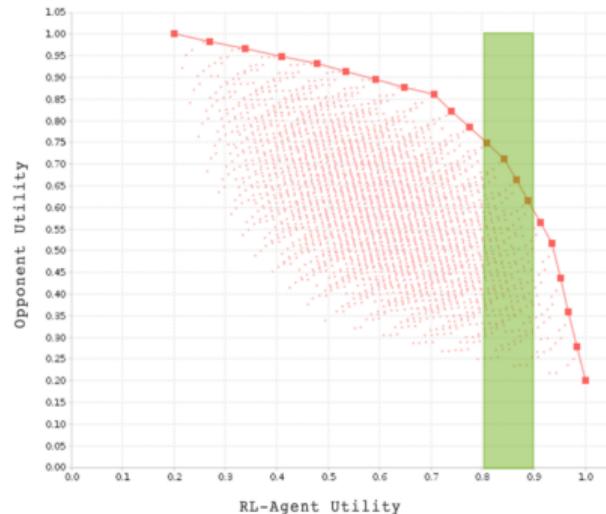
RLBOA: The details

- **State Space:** $\{\hat{u}(\omega_t^s), \hat{u}(\omega_{t-1}^s), \hat{u}(\omega_t^p), \hat{u}(\omega_{t-1}^p), t\}$.
 - $\hat{u}(\omega) = [N \times u(\omega)]^9$
- **Action Space:** $\leftarrow, -, \rightarrow$.
 - First step $\rightarrow i \in [0, N - 1]$
 - Out-of-boundary correction: $-$.
- **Training Method:** Q-learning
- **Acceptance Strategy [Recommended]:** $AC_{next}(\alpha = 1, \beta = 0)$ ¹⁰

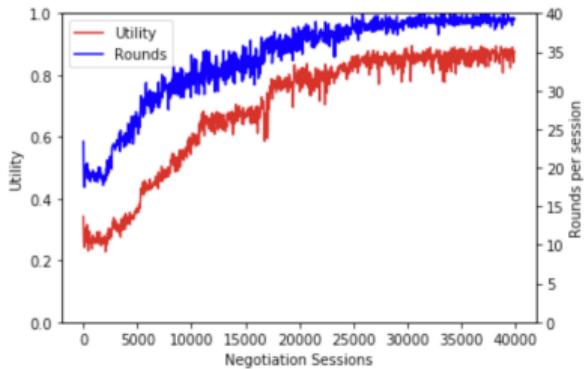
$$a(\omega) = \begin{cases} \text{Accept}, & \text{if } \alpha u(\omega) + \beta \geq u(o(s)) \\ \text{Reject}, & \text{otherwise} \end{cases}$$

¹⁰ Just ignore the special case at $u == 1$.

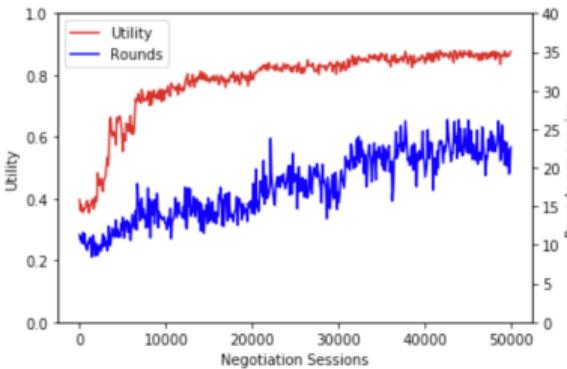
¹⁰ Default acceptance strategy in NegMAS



RLBOA: Evaluation and Results



(a) Scenario generality experiment against the Boulware agent.



(b) Opponent generality experiment in the medium sized domain with low opposition.

- Partners: TFT, Boulware TB
- Projection into one's utility space is surprisingly effective.
- Faster and better agreements!

Domain	Outcome space	Low opp.	High opp.
Small	256	0.2615	0.5178
Medium	3.125	0.3111	0.5444
Large	46.656	0.2595	0.5250

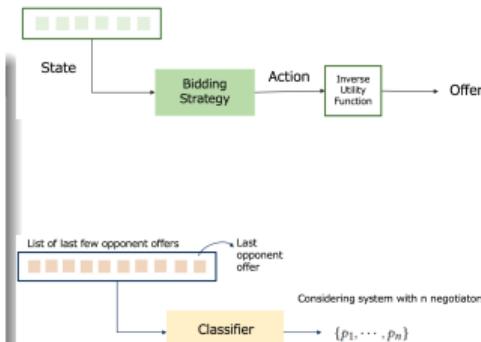
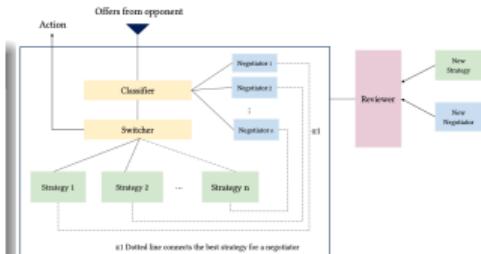
A Framework for Learning Offer Strategies

Main Idea¹¹

- Uses RL for learning **approximate best responses** to some agents.
- Uses Supervised Learning to learn a **realtime switching strategy** between learned best responses.
- Uses a form of Unsupervised Learning for **adapting the system to new partner types**.

Phases

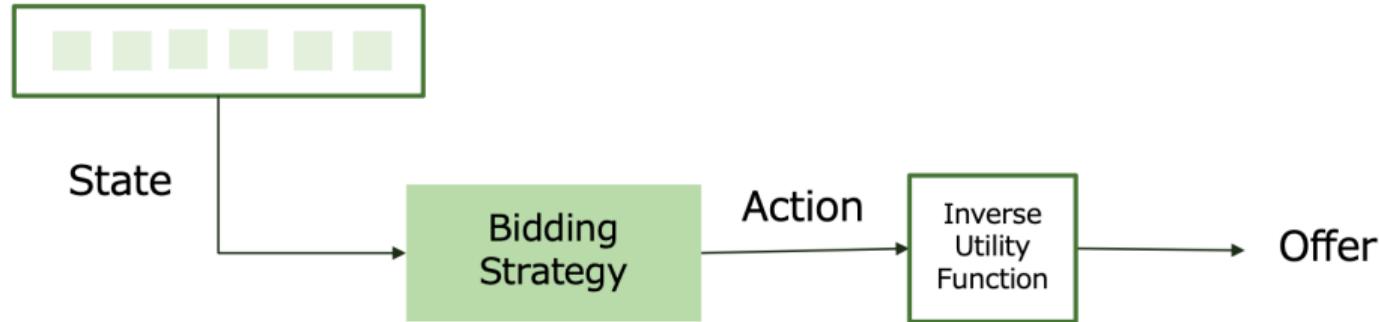
- **Before Negotiation** Learn approximate best responses to **a few** agents.
- **During Negotiation** Switch to the most appropriate **learned app. best response**
- **After Negotiation** Decide whether to add a new **best response**.



¹¹ Ayan Sengupta, Yasser Mohammad, and Shinji Nakada. "An Autonomous Negotiating Agent Framework with Reinforcement Learning Based Strategies and Adaptive Strategy Switching". In: *ICINN 2011: International Conference on Intelligent Networks*. ICINN 2011, 2011, pp. 1–6. doi:10.1109/ICINN.2011.6032520. 42 / 73

Y. Mohammad (NEC CORPORATION)

Before: Learning Approximate Best Response



The RL Component

State Self utility of last N offers plus relative time.

Action Utility of next offer $\in [0, 1]$.

Reward Agreement/disagreement utility.

Trainer Soft Actor Critic (SAC)

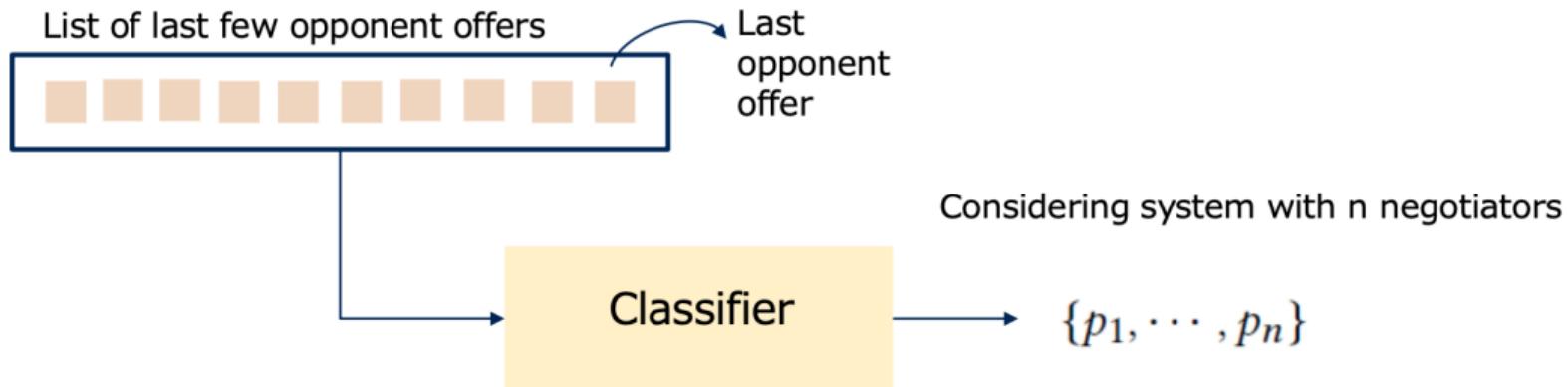
$$s_t = \{t_r, U_s(\omega_s^{t-2}), U_s(\omega_o^{t-2}), U_s(\omega_s^{t-1}), \\ U_s(\omega_o^{t-1}), U_s(\omega_s^t), U_s(\omega_o^t)\}$$

$$a_t = u_s^{t+1} \text{ such that } u_r < u_s \leq 1$$

$$U_s^{-1}(u_s) = \operatorname{argmin}_{\omega} f(\omega), \text{ where}$$

$$f(\omega) = (U_s(\omega) - u_s)^2 \quad \forall \omega \in \Omega$$

During: Learning realtime Partner Classification



The SL Components

Features Opponent last K offers.

Target Opponent Type (discrete set)

After: Reviewing New Pairs

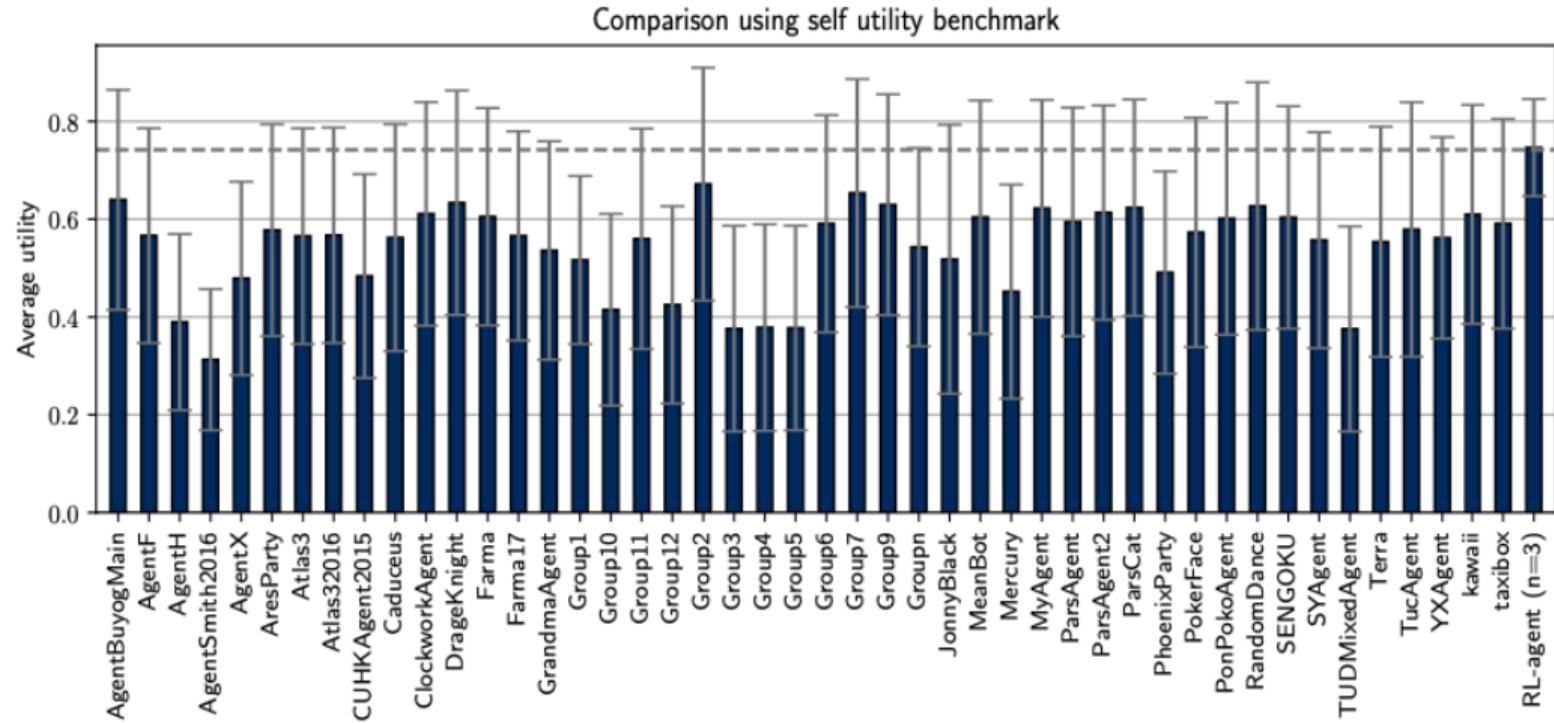
New Partner Type (N_{new}) Encountered

- Train a best response (using SAC) $\rightarrow S_{new}$.
- Evaluate S_{new} against $N_{new} \rightarrow U(S_{new})$
- Evaluate *Current* against $N_{new} \rightarrow U(Current)$
- Add (S_{new}, N_{new}) iff $\beta U(Current) < U(S_{new})$
- Update best responses ↓.

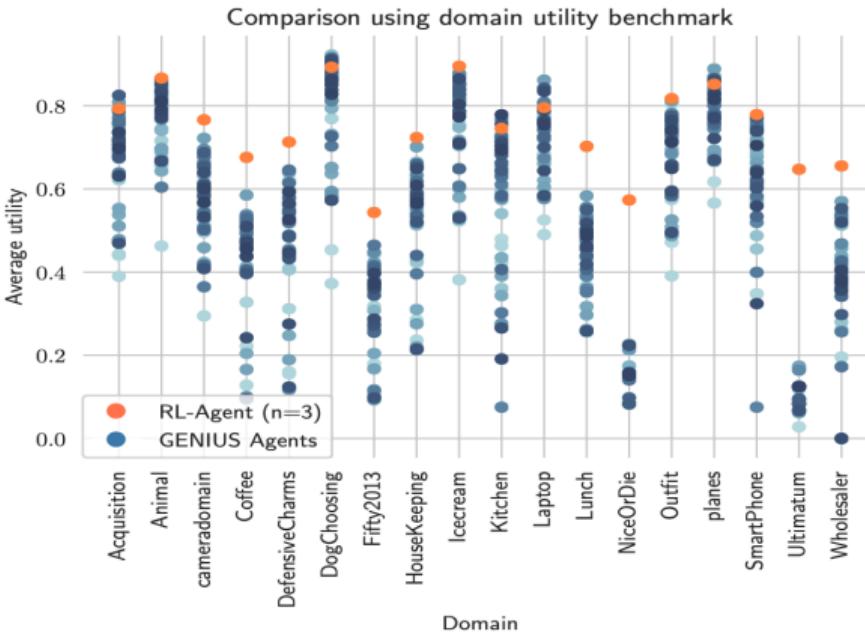
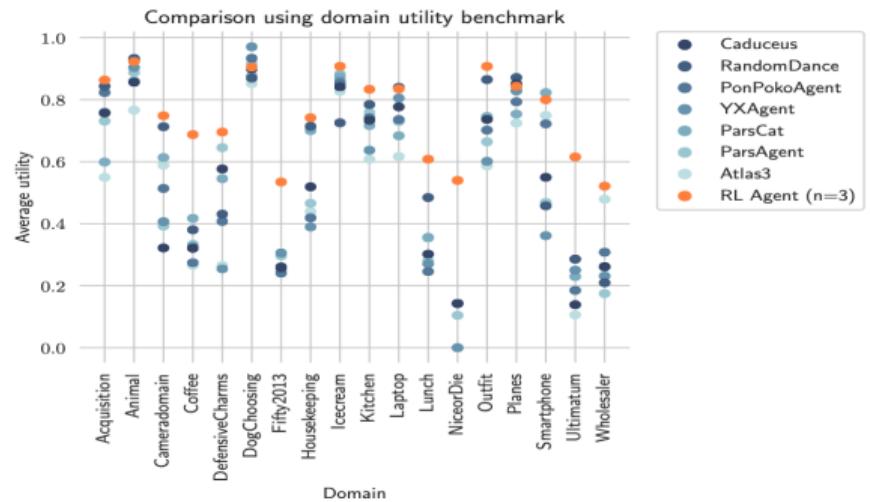
Update Best Responses

- For every learned ABR, negotiator pair (S, N) :
 - Evaluate S_{new} against $N \rightarrow U(S_{new})$
 - Evaluate S against $N \rightarrow U(S)$
 - Replace S with S_{new} iff $\alpha U(S) < U(S_{new})$

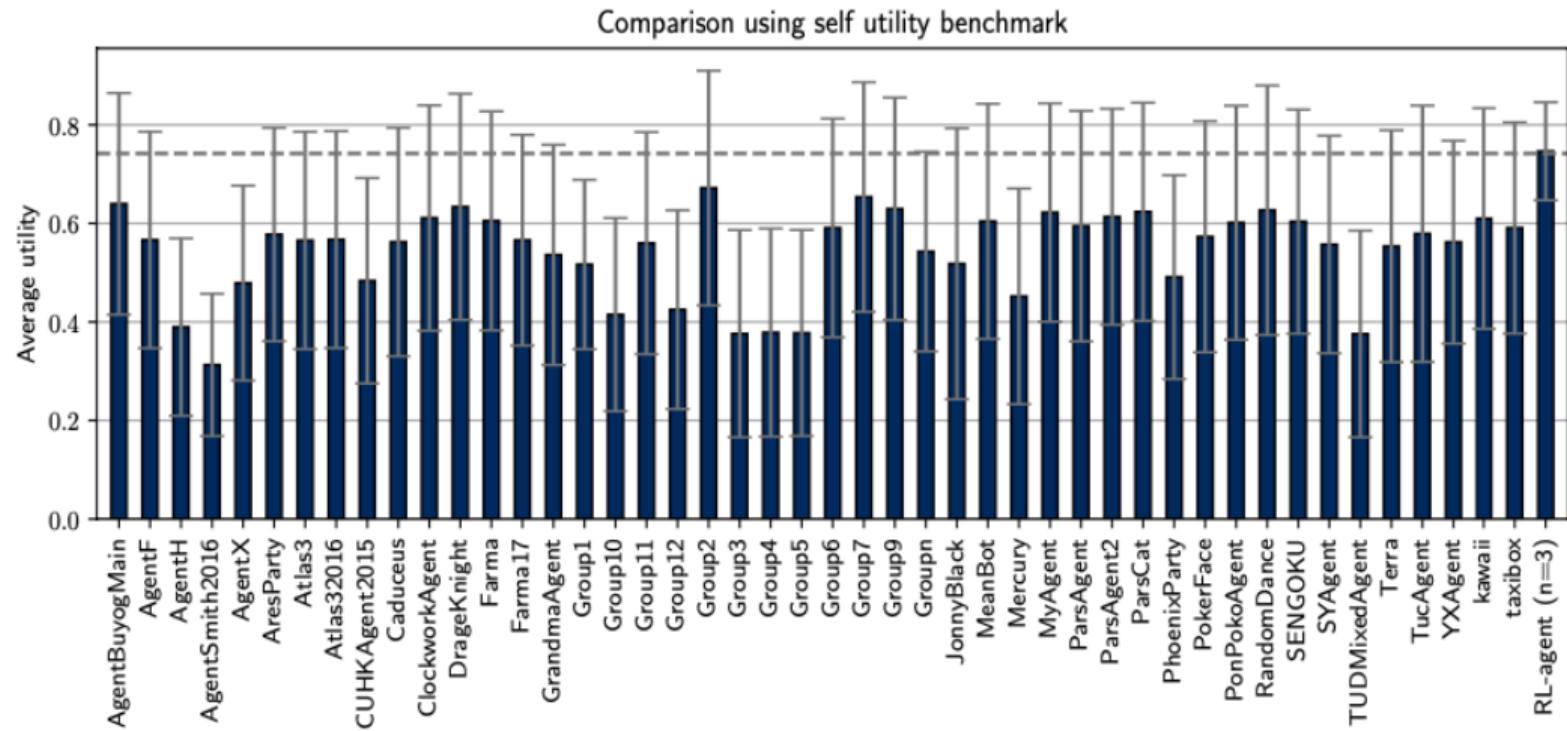
Results: Against Different Opponents



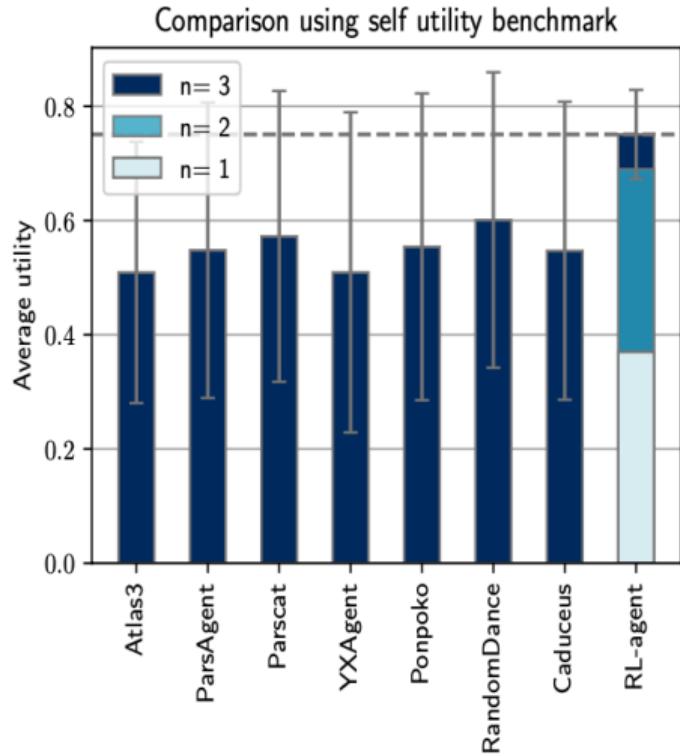
Results: In Different Domains



Results: Compared with SOTA Agents



Results: Improvement with new best responses



DQN for learning Acceptance Strategy¹²

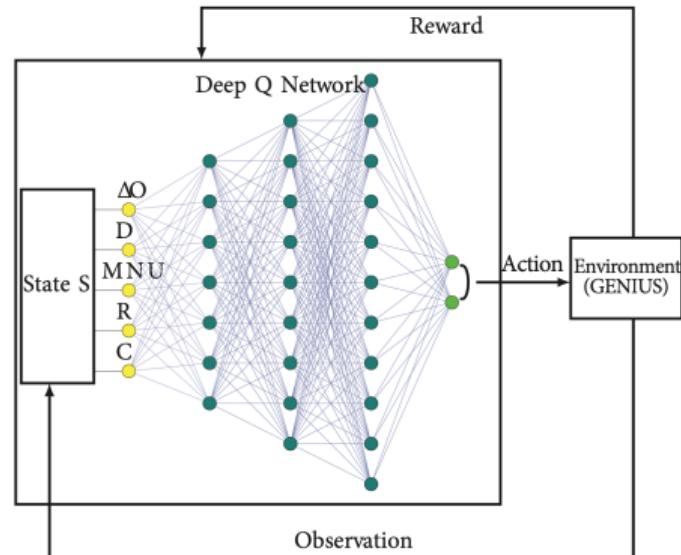
Main Idea

- Learning the acceptance strategy for a fixed offering strategy.

Settings

- State Space** $u(\omega) - u(\phi), 1 - t, u(o(s)), u_t, u(\omega)$
 - u_t is a relatively large target utility (e.g. 0.8).
- Action Space** Accept/Reject
- Reward**

$$r = \begin{cases} -2|u_t - u_f|, & \text{if } u_t > u(\omega_a) \\ +2|u_t - u_f|, & \text{if } u_t < u(\omega_a) \\ 0 & \text{if non-terminal} \end{cases}$$



¹²Yousef Razeghi, Celal Ozan Berk Yavuz, and Reyhan Aydoğan. "Deep reinforcement learning for acceptance strategy in bilateral negotiations". In: *Turkish Journal of Electrical Engineering & Computer Sciences* 28.4 (2020), pp. 1824–1840.

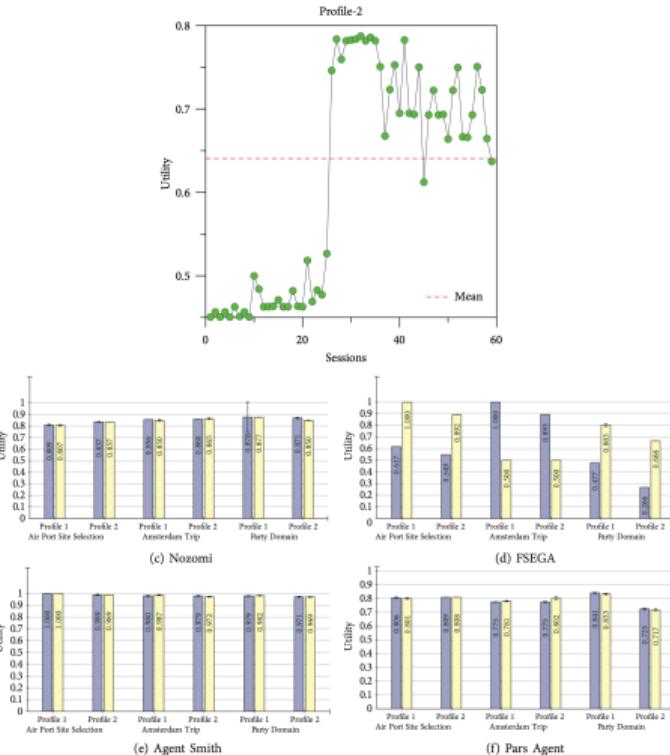
Evaluation

Training

- Domain England-Zimbabwe (576 outcomes)
- Partner Gahboninho
- Offering Strategy AgentK
- Opponent Model AgentLG, Not TFT.

Testing

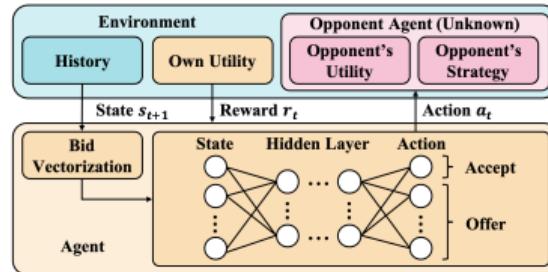
- Domains Party (3072), Amsterdam (3024), Airport (420)
- Partners Agent Smith, Yushu, FSEGA, IAMHaggler, ParsAgent, Nozomi
- Baseline ACnext



Learning Offer and Acceptance Policies together¹³

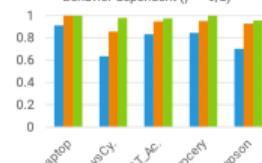
- Fixed domain (i/o using outcomes).
- Discrete Issues: One hot encoding per issue.
- State Space $\omega^s, \omega^o, t, \eta_t$
- Action Space $\Omega \wedge \text{Accept}$
- Reward = $\begin{cases} u(\omega_a), & \text{At the end} \\ 0 & \text{non terminal state} \end{cases}$

VeNAS: Versatile Negotiating Agent Strategy



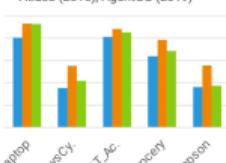
Simple strategies

Time-dependent ($\epsilon = 0.1, 1.0, 5.0$)
Behavior-dependent ($\gamma = 1, 2$)



ANAC champions

AgentK (2010), HardHeaded (2011)
Atlas3 (2015), AgentGG (2019)



VeNAS > DRBOA > Baseline DRBOA > VeNAS > Baseline

¹³Toki Takahashi et al. "VeNAS: Versatile Negotiating Agent Strategy via Deep Reinforcement Learning". In: AAAI 2022. 2022.

Reward-based negotiating agent strategies¹⁴

State set of state $s_t \in \mathcal{S}$: The agent's offer ω_t , opponent's offer ω'_t and accept signal η'_t , and t/T .

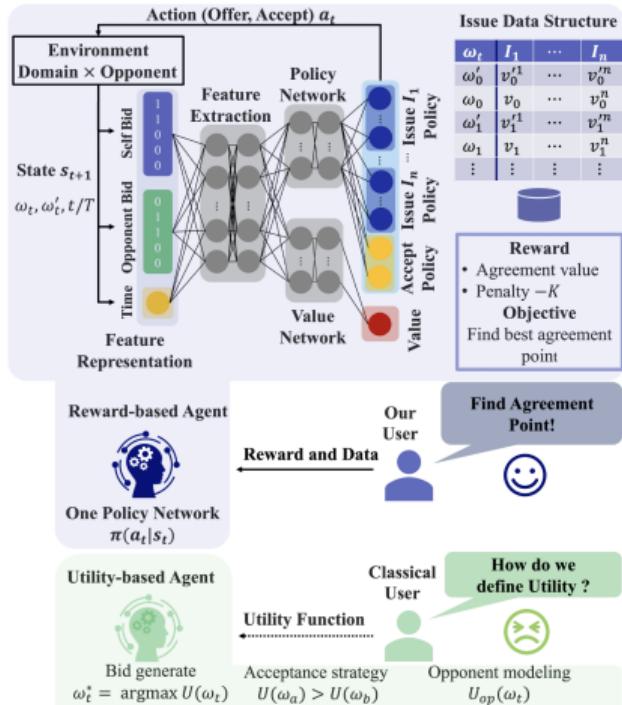
Action set of action $a_t \in \mathcal{A}$: The agent's selected offer ω_t and accept signal η_t . The action space is proportional to the domain size of the utility function.

Reward $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$: When the agent accepts, the agent obtains the final utility value. Penalty $-K$ is given when the negotiation ends without reaching an agreement. Otherwise, the reward is 0.

Policy function $\mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$: The policy function is defined as $\pi_\theta(a_t | s_t) := \Pr(A_t = a_t | S_t = s_t, \theta_t = \theta)$.

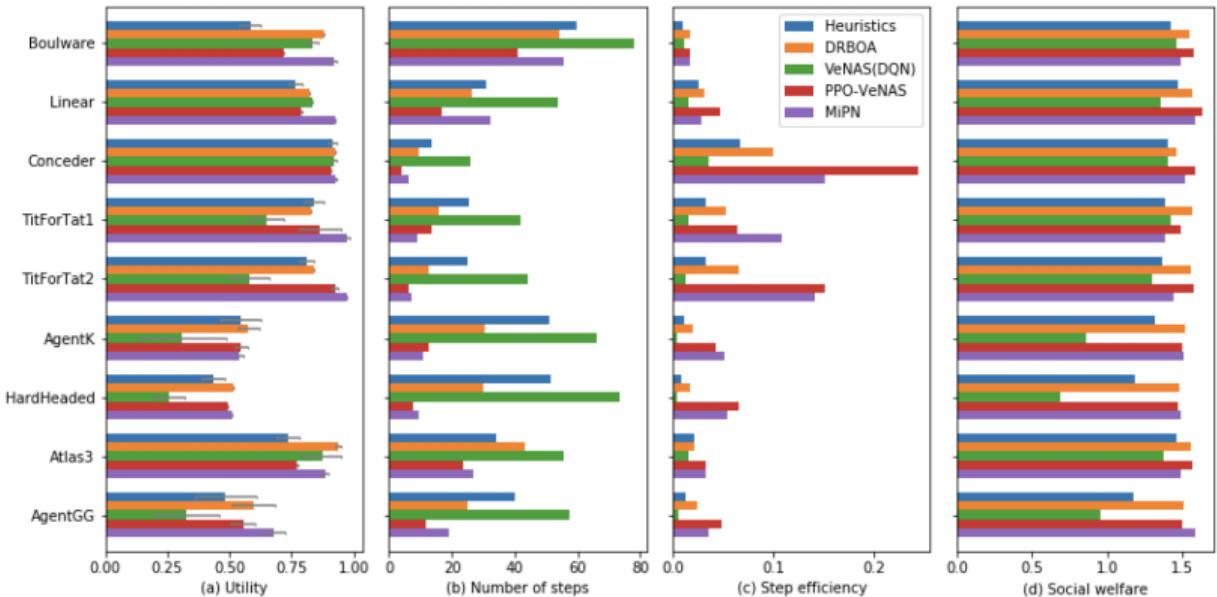
Transition function $\mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$: Transition function is defined as $p(s_{t+1}|s_t, a_t) := \Pr(S_{t+1} = s_{t+1} | S_t = s_t, A_t = a_t)$.

History $\mathcal{D} = (\omega_0, \omega'_0, \dots, \omega_t, \omega'_t)$: The observable data during a negotiation.



¹⁴ Ryota Higa et al. "Reward-based negotiating agent strategies". In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37. 2023, pp. 11569–11577.

Reward-based negotiating agent strategies¹⁵



¹⁵Higa et al., "Reward-based negotiating agent strategies".

Outline

1 Theoretical Foundations

2 The Competition

- ANAC
- SCML

3 RL for SCML

4 Development Environment

5 References

Outline

1 Theoretical Foundations

2 The Competition

- ANAC
- SCML

3 RL for SCML

4 Development Environment

5 References

Automated Negotiating Agents Competition (ANAC)

- An international competition dedicated to automated negotiation.
- Started in conjunction with AAMAS in 2010.
- Running every year with AAMAS or IJCAI (14 iterations so far).
- Next ANAC will be in conjunction with AAMAS 2024 (not official yet).



Outline

1 Theoretical Foundations

2 The Competition

- ANAC
- SCML
 - SCML-OneShot
 - Simulation Steps

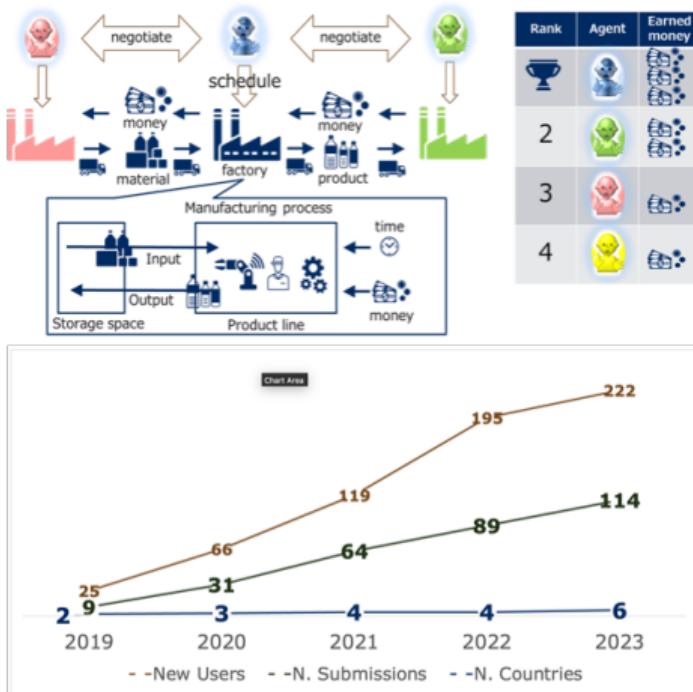
3 RL for SCML

4 Development Environment

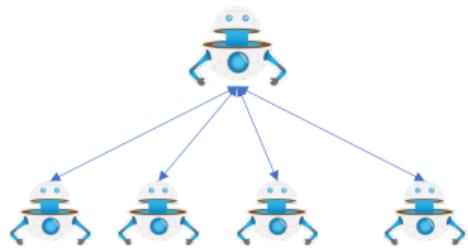
5 References

Supply Chain Management League (SCML)

- Running as part of ANAC since 2019.
- **Scenario** A market in which all trade is done through **concurrent closed bilateral negotiations**:
 - Represents several applications in the real world.
- Agent preferences are endogenous to the simulation.
- **Goal** Maximize your profit.
- **Main Challenges:**
 - Situated Negotiations.
 - Sequential Negotiation (learning between negotiations).
 - Concurrent Negotiation (outside-options).
 - Trust management.



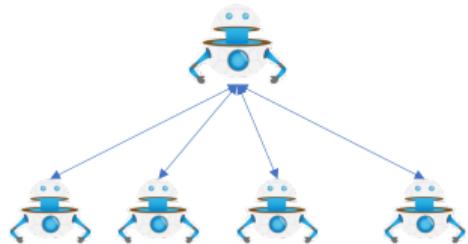
Concurrent Negotiation



Concurrent Negotiation

Generality

- Specific scenario (buyer-seller).
- General domain



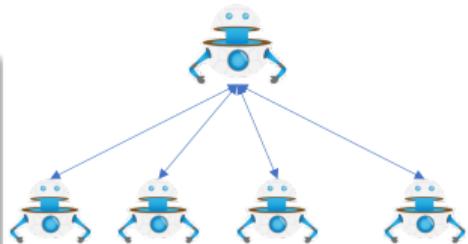
Concurrent Negotiation

Generality

- Specific scenario (buyer-seller).
- General domain

Decommitment

- Symmetric de-commitment.
- Asymmetric de-commitment.
- No de-commitment.



Concurrent Negotiation

Generality

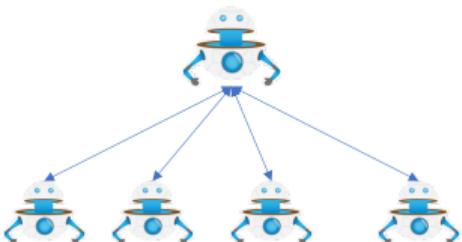
- Specific scenario (buyer-seller).
- General domain

Decommitment

- Symmetric de-commitment.
- Asymmetric de-commitment.
- No de-commitment.

Timing

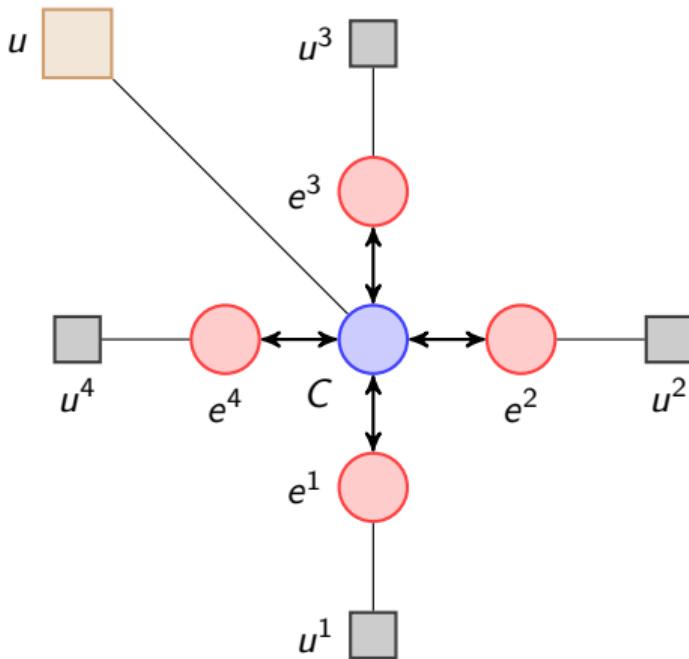
- Synchronous.
- Any-time.



Concurrent Negotiations

These negotiations are dependent, as evidenced by the global ufun u , which depends on a global outcome: i.e., the outcomes of all negotiations.

The problem of how to negotiate in this setting is reminiscent of how to bid in simultaneous auctions, when bidders' utilities are combinatorial.¹⁶

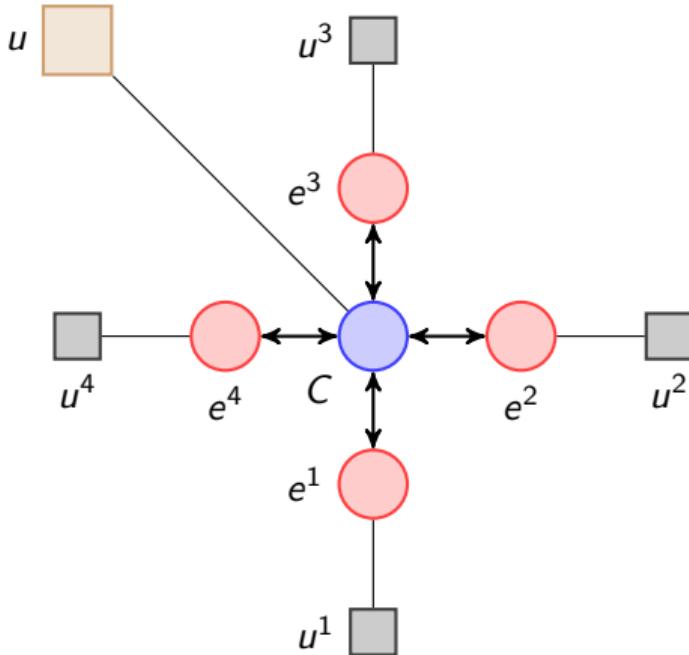


¹⁶ Michael P. Wellman, Eric Sodomka, and Amy Greenwald. "Self-confirming price-prediction strategies for simultaneous one-shot auctions". In: *Games and Economic Behavior* 201 (2017), pp. 339–372.

Concurrent Negotiations

In ANAC SCML, an agent negotiates with multiple agents simultaneously. Moreover, the agent's utility depends on the outcomes of all the negotiations.

An agent represents a factory, whose production depends on the inputs it acquires from all its trading partners, and so, in turn, does its sales.



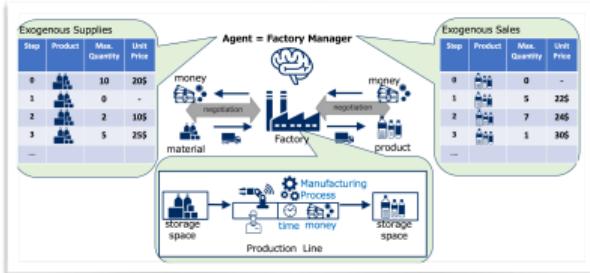
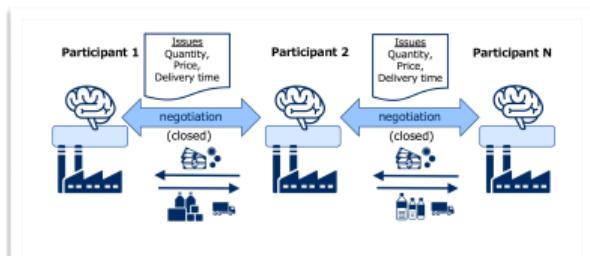
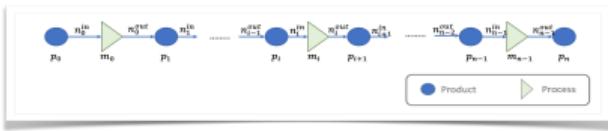
SCML World

Challenge

- Negotiation game with imperfect information
- Concurrent negotiations.
- Repeated negotiations → OneShot.
- Sequential negotiations → Standard.

Information

- **Website** <https://scml.cs.brown.edu/>
- **Code** <https://www.github.com/yasserfarouk/scml>



SCML Competition

Competition Details

- Runs as part of ANAC IJCAI.
- You control one or more factories.
 - **Oneshot track** → one factory (predefined ufun).
 - **Standard track** → one factory (you define your own ufun).
 - **Collusion track** → multiple factories (3).

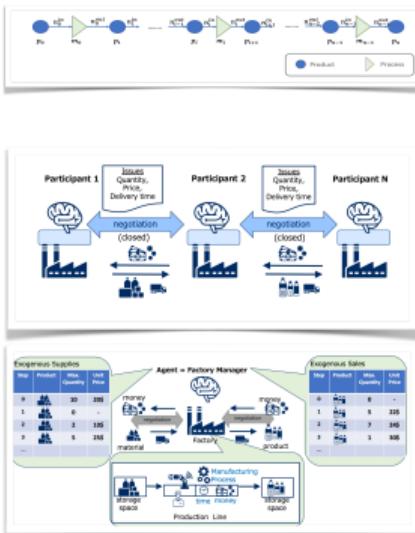
SCML Competition

Competition Details

- Runs as part of ANAC IJCAI.
- You control one or more factories.
 - Oneshot track** → one factory (predefined ufun).
 - Standard track** → one factory (you define your own ufun).
 - Collusion track** → multiple factories (3).

Flavors

- Online competition at <https://scml.cs.brown.edu>
- Official competition as part of ANAC.



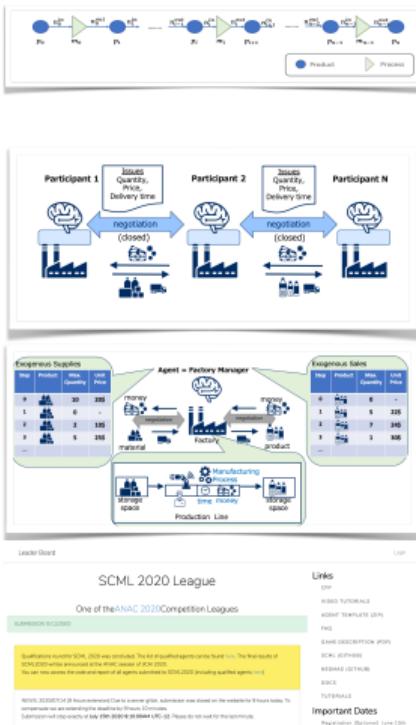
SCML Competition

Competition Details

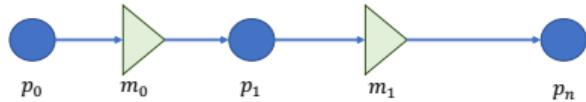
- Runs as part of ANAC IJCAI.
- You control one or more factories.
 - Oneshot track** → one factory (predefined ufun).
 - Standard track** → one factory (you define your own ufun).
 - Collusion track** → multiple factories (3).

Flavors

- Online competition at <https://scml.cs.brown.edu>
- Official competition as part of ANAC.



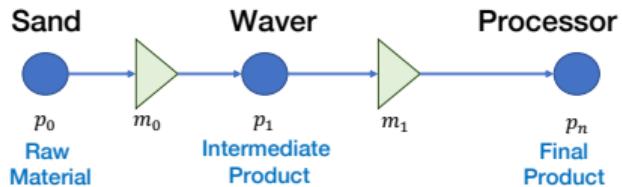
Overview



- A **production-graph** defines what can be produced and how.



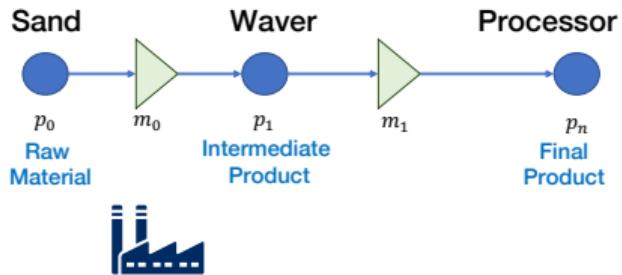
Overview



- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.



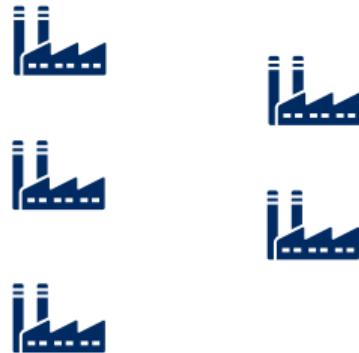
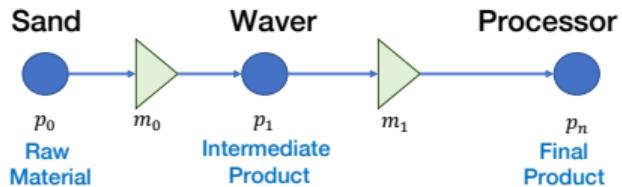
Overview



- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.

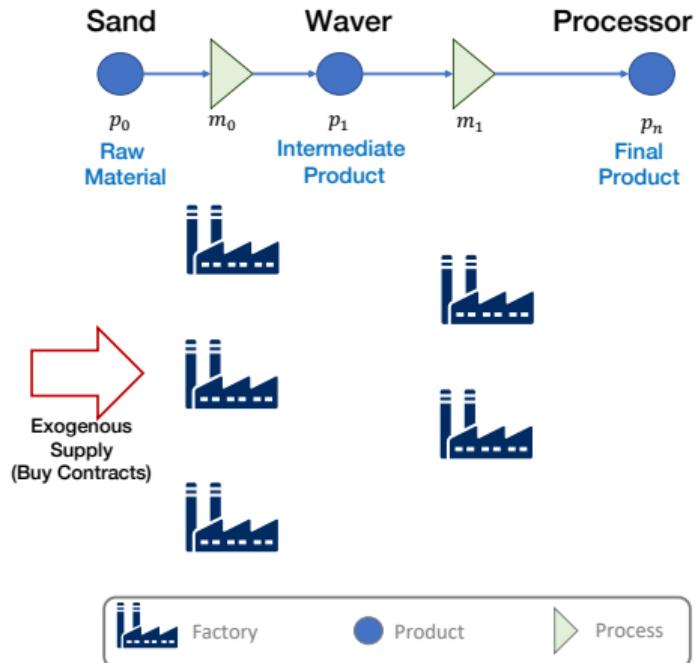


Overview



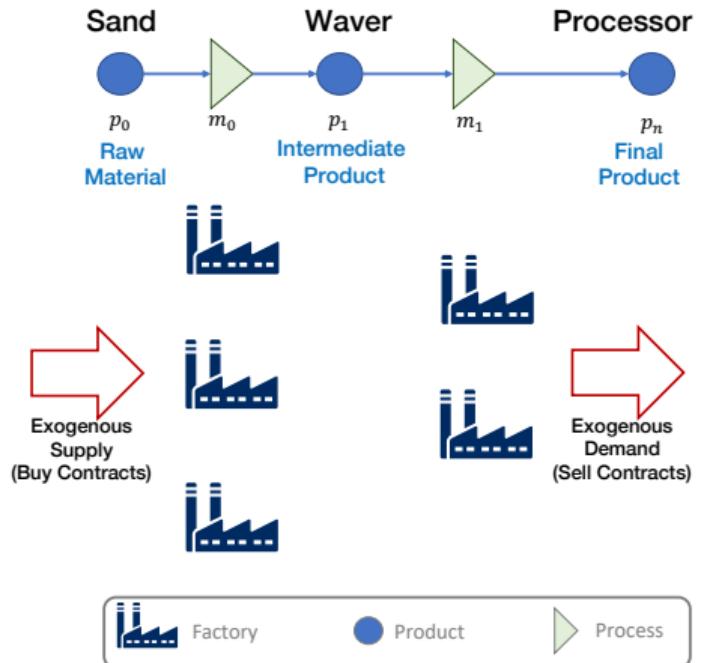
- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.
- We have two layers of factories/agents.

Overview



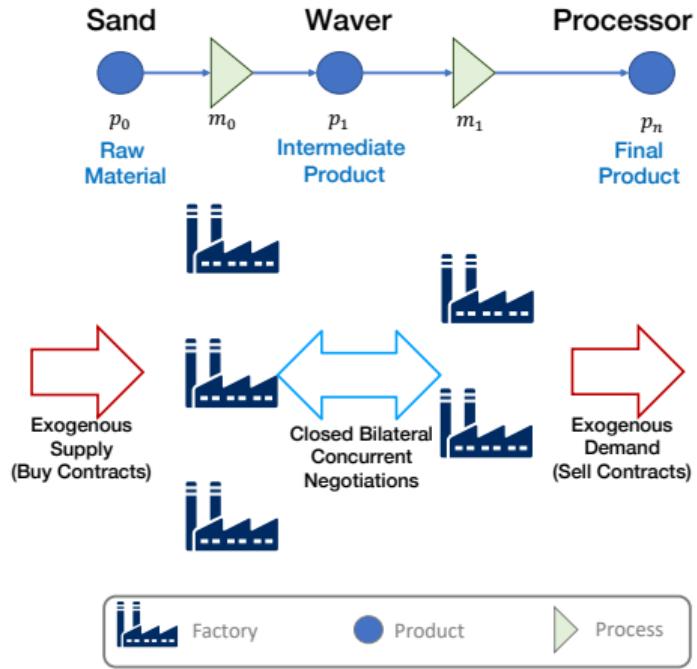
- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.
- We have two layers of factories/agents.
- L_0 factories/agents receive exogenous supplies of **raw material**.

Overview



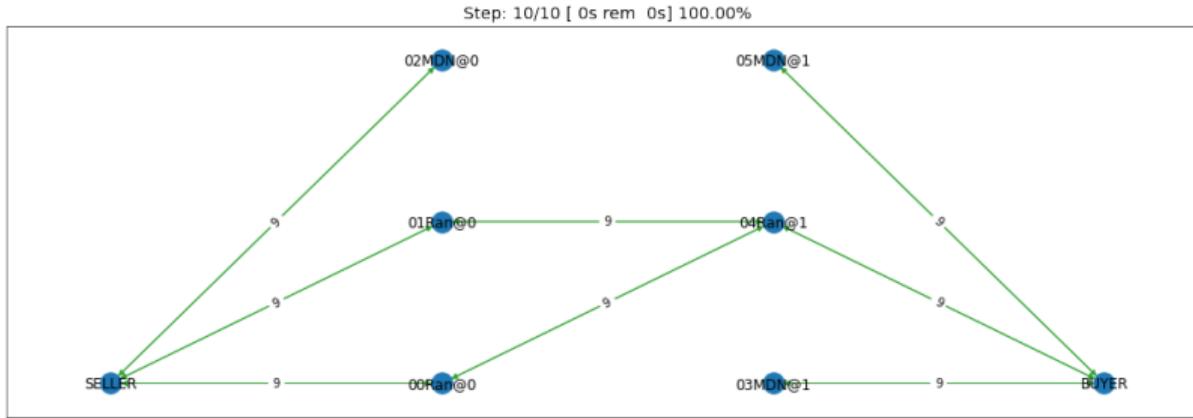
- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.
- We have two layers of factories/agents.
- L_0 factories/agents receive exogenous supplies of **raw material**.
- L_1 factories/agents receive exogenous sales of **final product**.

Overview



- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.
- We have two layers of factories/agents.
- L_0 factories/agents receive exogenous supplies of **raw material**.
- L_1 factories/agents receive exogenous sales of **final product**.
- L_0 negotiate with L_1 agents to exchange **intermediate product**

SCML-OneShot Track



Main Idea

- Agents arranged in two production levels (3 products, 2 processes)
- Every day you get a **fresh set** of exogenous contracts.
- All products perish in one day (no inventory accumulation).

Utility Function

General Form

Utility = Profit = Sales - Supply cost - Production cost - Disposal cost - Delivery Penalty

Sales unit price \times quantity \forall feasible sales.

Supply cost unit price \times quantity \forall supplies.

Production cost unit production cost \times quantity produced.

Disposal cost unit disposal cost \times quantity bought but not produced.

Shortfall penalty unit shortfall penalty \times infeasible sales.

Information about self

Static Information

- Number of production lines.
- Production cost.
- Mean and variance of disposal cost and shortfall penalty.
- Input/output product, consumers/suppliers, n. input/output negotiations.

Dynamic Information

- current input/output negotiation issues.
- current input/output exogenous contracts (quantity, unit price).
- current disposal cost and shortfall penalty.
- Current balance (money in wallet).

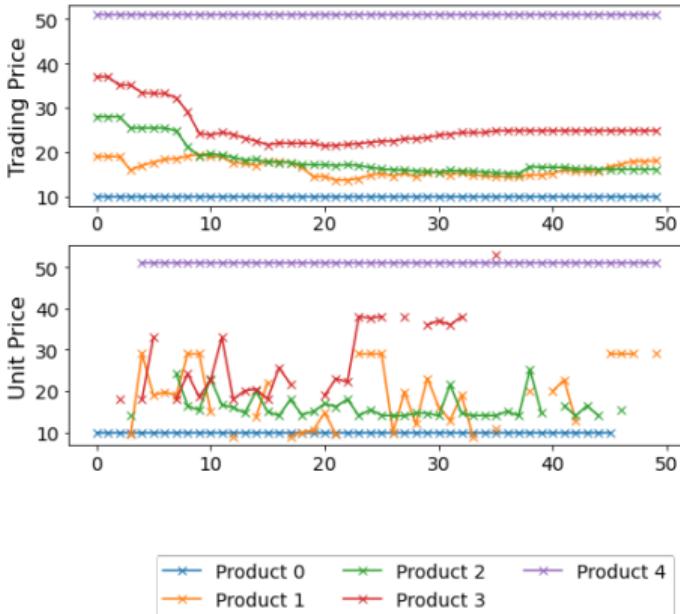
Market Information

Trading prices

- Trading prices represent a weighted running average of different product prices.
- Available to the agent through the AWI in all tracks.

Exogenous Contract Summary

- The total quantity and average prices of all exogenous contracts are now available through the AWI.
 - Exogenous contracts for individual agents are private information.



Product 0 Product 2 Product 4
Product 1 Product 3

Other Agents' Information

Financial Reports

For each agent, a financial report is published every m days (e.g. 5) with the following information:

- Current balance (money in wallet).
- breach probability (fraction of sale contracts not satisfied).
- breach level (average fraction of sales not satisfied).

Simulation Steps

Once



Initialize all agents
• init()

Simulation Steps

Once



Initialize all agents
• init()



Update trading prices

Simulation Steps

Once



Simulation Steps

Once

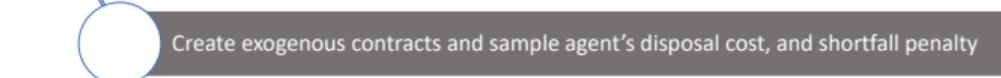


Simulation Steps

Once



Every Day



Simulation Steps

Once



Initialize all agents
• `init()`



Update trading prices



Create exogenous contracts and sample agent's disposal cost, and shortfall penalty



Initialize agents for the day
• `before_step()` [First call every day]



Run All Negotiations

• `propose()` `respond()` | `on_neg*_success()` `on_neg*_failure()`



Calculate profits for all agents by simulating contract execution.

Every Day

Simulation Steps

Once



Simulation Steps

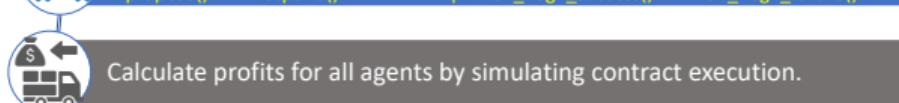
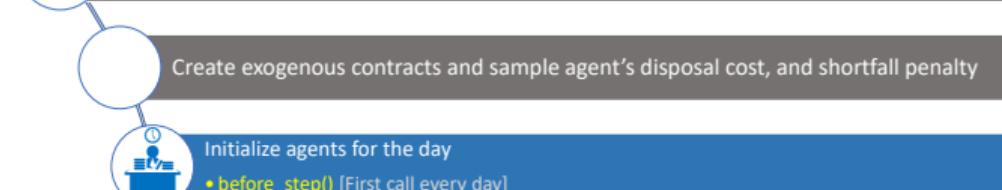
Once



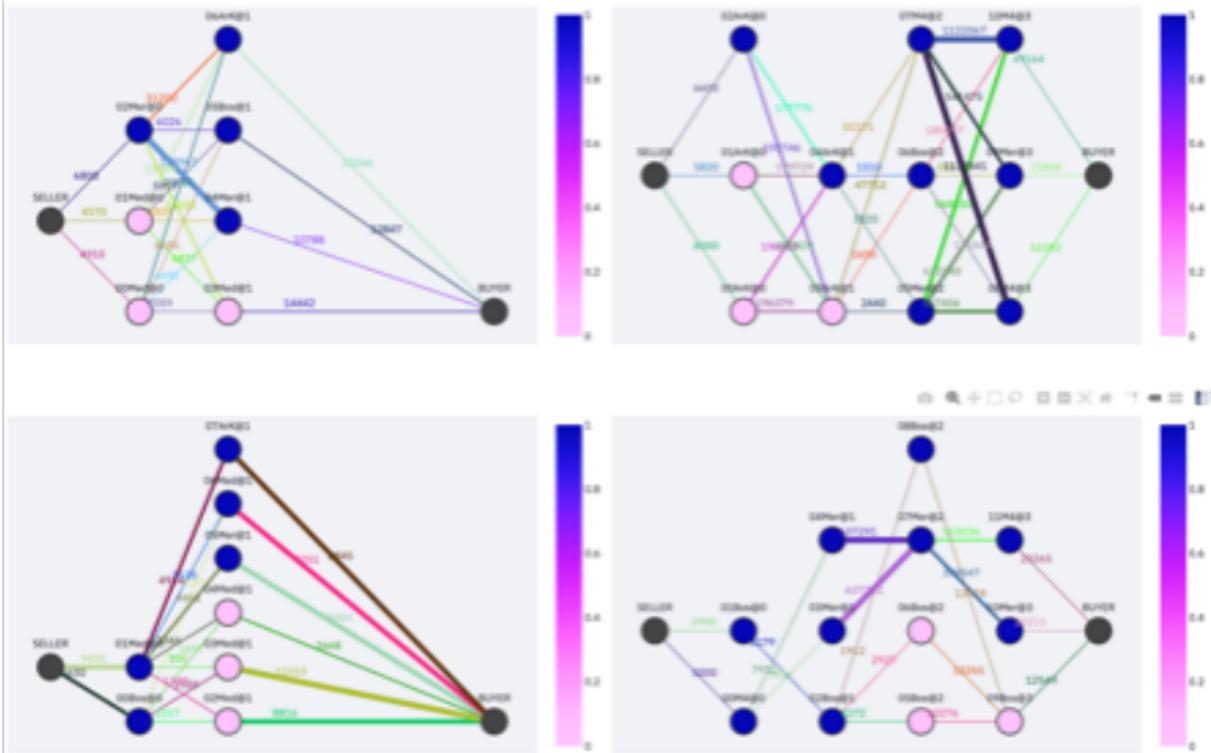
Every Day



Create exogenous contracts and sample agent's disposal cost, and shortfall penalty



SCML Environment



Outline

1 Theoretical Foundations

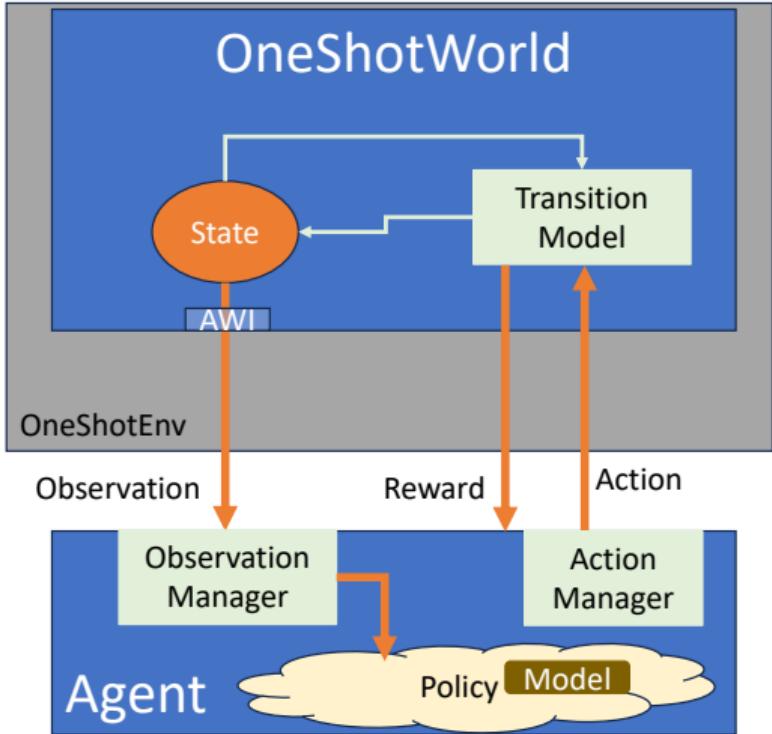
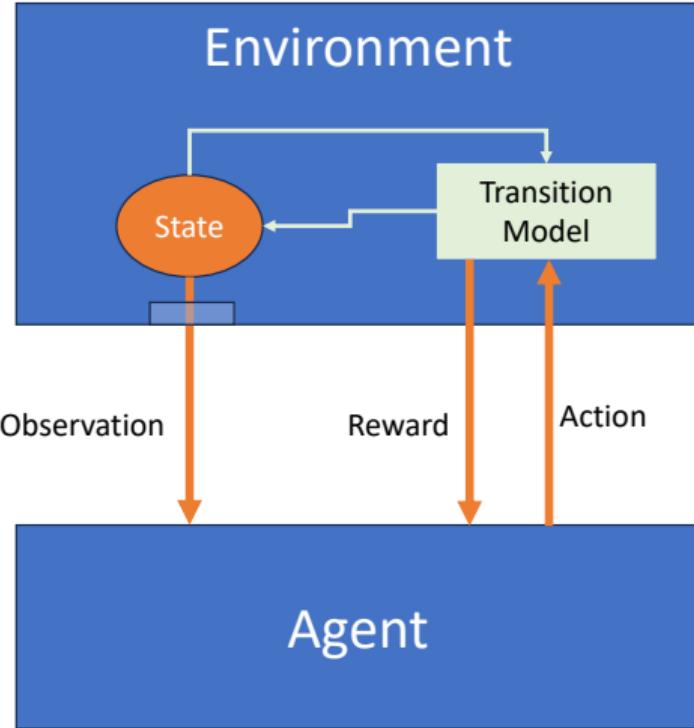
2 The Competition

3 RL for SCML

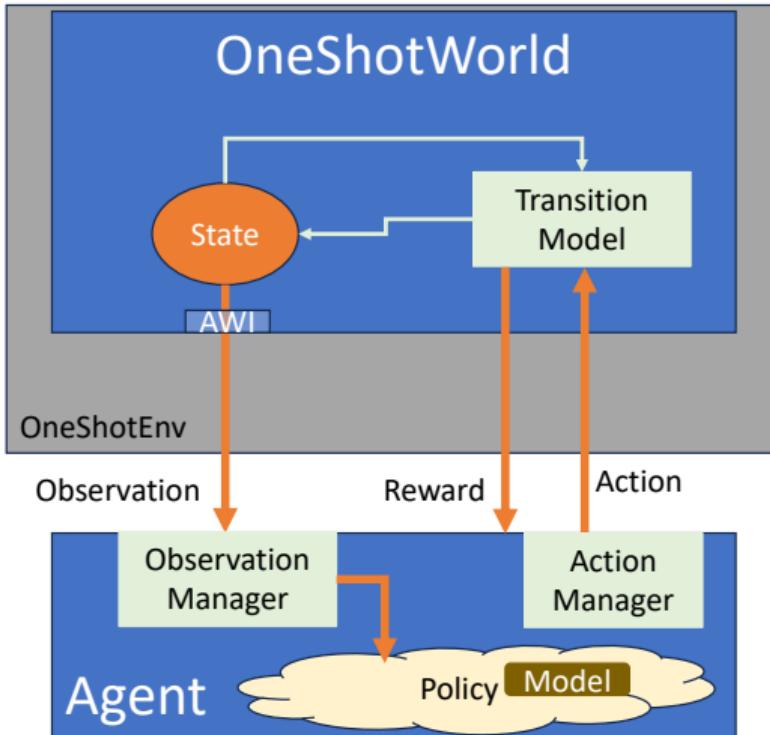
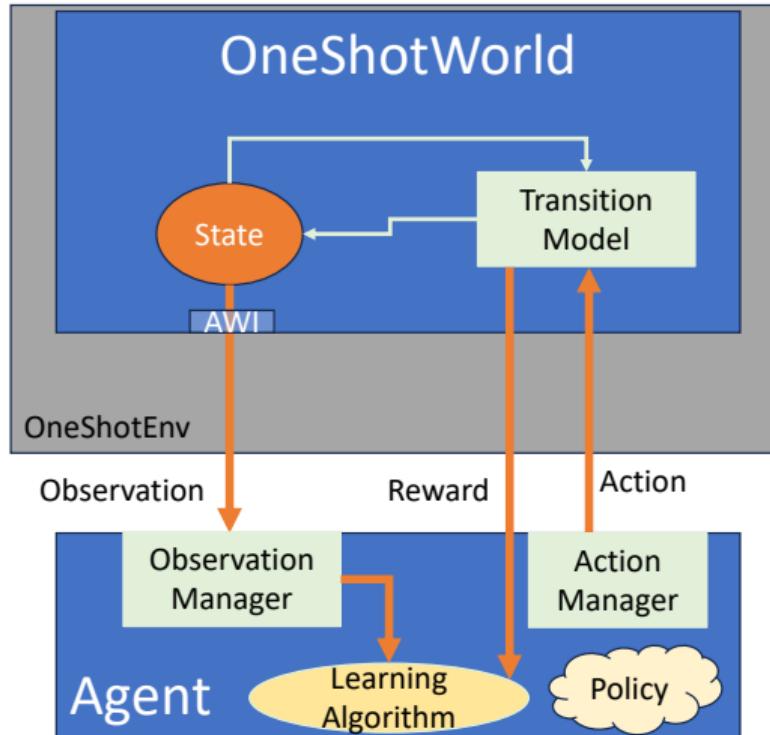
4 Development Environment

5 References

RL for SCML

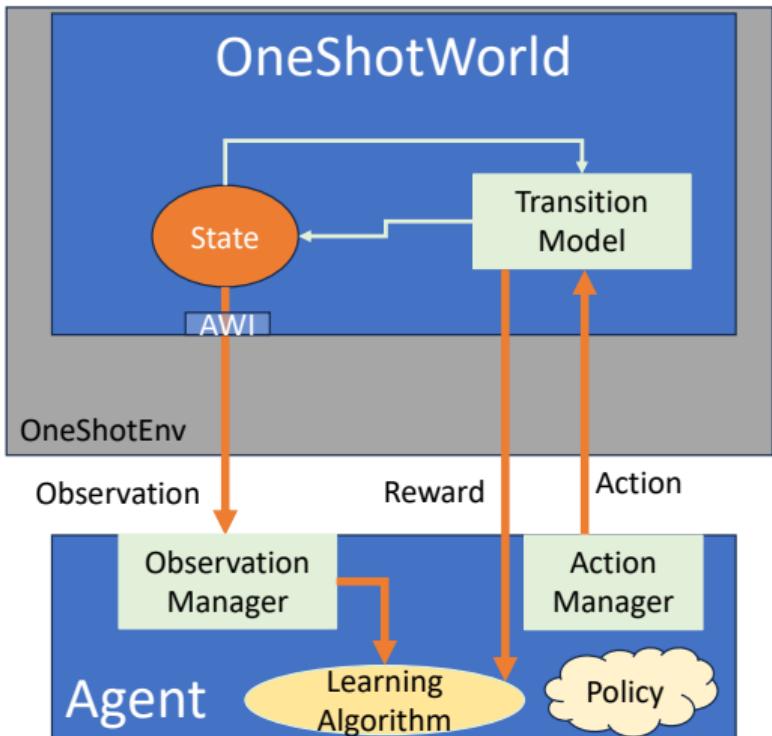


RL for SCML



RL for SCML: Main Challenges

- Huge and hybrid observation space
 - The main challenge is to design a good **ObservationManager**
- Sparse Reward
 - Does **Reward Shaping** help?
 - Can we devise a reward signal other than the profit (e.g. KPIs)?
- Every simulation is different (i.e. competitors, suppliers, consumers).
 - How to handle variable number of negotiations (i.e. elastic action space)?
 - How to handle negotiations ending at different time (in the same day)?
 - How to divide all possible simulations into usable **contexts** suitable for learning?
- How to learn (learning algorithm)?



Outline

1 Theoretical Foundations

2 The Competition

3 RL for SCML

4 Development Environment

- Installation
- Running a Simulation
- Running a Tournament

5 References

Outline

1 Theoretical Foundations

2 The Competition

3 RL for SCML

4 Development Environment

- Installation
- Running a Simulation
- Running a Tournament

5 References

Outline

1 Theoretical Foundations

2 The Competition

3 RL for SCML

4 Development Environment

- Installation
- **Running a Simulation**
- Running a Tournament

5 References

Outline

1 Theoretical Foundations

2 The Competition

3 RL for SCML

4 Development Environment

- Installation
- Running a Simulation
- **Running a Tournament**

5 References

Outline

1 Theoretical Foundations

2 The Competition

3 RL for SCML

4 Development Environment

5 References

References I

- Bakker, Jasper et al. "RLBOA: A modular reinforcement learning framework for autonomous negotiating agents". In: *Proceedings of the 18th international conference on autonomous agents and multiagent systems*. 2019, pp. 260–268.
- Fernandez, Raquel and Jacob Glazer. *Striking for a bargain between two completely informed agents*. Tech. rep. National Bureau of Economic Research, 1989.
- Higa, Ryota et al. "Reward-based negotiating agent strategies". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 10. 2023, pp. 11569–11577.
- Kawata, Ryohei and Katsuhide Fujita. "Meta-Strategy Based on Multi-Armed Bandit Approach for Multi-Time Negotiation". In: *IEICE TRANSACTIONS on Information and Systems* 103.12 (2020), pp. 2540–2548.
- Mohammad, Y. et al. "Supply Chain Management World: A benchmark environment for situated negotiations". In: *Proceedings of the 22nd International Conference on Principles and Practice of Multi-Agent Systems*. 2019.
- Myerson, Roger B and Mark A Satterthwaite. "Efficient mechanisms for bilateral trading". In: *Journal of economic theory* 29.2 (1983), pp. 265–281.

References II

Razeghi, Yousef, Celal Ozan Berk Yavuz, and Reyhan Aydoğan. "Deep reinforcement learning for acceptance strategy in bilateral negotiations". In: *Turkish Journal of Electrical Engineering & Computer Sciences* 28.4 (2020), pp. 1824–1840.

Sengupta, Ayan, Yasser Mohammad, and Shinji Nakadai. "An Autonomous Negotiating Agent Framework with Reinforcement Learning Based Strategies and Adaptive Strategy Switching Mechanism". In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '21. Virtual Event, United Kingdom: International Foundation for Autonomous Agents and Multiagent Systems, 2021, pp. 1163–1172. ISBN: 9781450383073.

Takahashi, Toki et al. "VeNAS: Versatile Negotiating Agent Strategy via Deep Reinforcement Learning". In: *AAAI 2022*. 2022.

Wellman, Michael P., Eric Sodomka, and Amy Greenwald. "Self-confirming price-prediction strategies for simultaneous one-shot auctions". In: *Games and Economic Behavior* 201 (2017), pp. 339–372.

references