Reinfocement Learning for Automated Negotiation

Yasser Mohammad
Data Science Labs
NEC

May 19, 2025

Outline

Timeline

- 8:30 - 10:00: Theoretical Foundations
- 10:00 - 10:45: Coffee Break
- 10:45 - 11:45: Recent Advances
- 10:45 - 12:30: Developing a negotiator using RL

More Information

- Tutorial website: `http://www.yasserm.com/tutorials/aams2025.html`
- NegMAS Documentation: `https://negmas.readthedocs.io/en/latest/`

Table of Contents

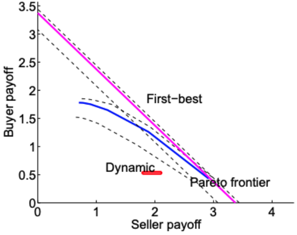
Automated Negotiation

Motivation

- Negotiation is ubiquitous in societal and business interactions.
- Increased utilization of AI agents in businesses → higher need and opportunity for automating negotiation.
- People are not very good at negotiation. Market studies routinely reveal somewhere between 9% to 20% value loss due to negotiation inefficiencies..

Automated negotiation can lead to

- **Better** agreements → Less money on the table.
- **Faster** agreements → More dynamic markets.
- **New** Opportunities → New applications.



9-20% loss based on 27, 000 negotiations[1]

Bradley J Larsen. "The efficiency of real-world bargaining: Evidence from wholesale used-auto auctions". In: *The Review of Economic Studies* 88.2 (2021), pp. 851–882

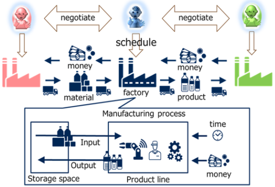
RL Motivation: For Automated Negotiation Researchers

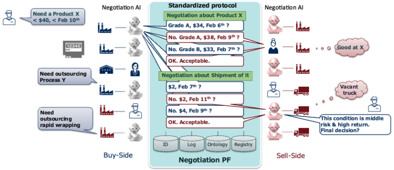
The need

- No general-purpose AN solution.
- No known equilibrium for bargaining with incomplete information [2].

The opportunity

- RL has been successfully applied to various domains, including games, robotics, and finance.
- AN can be cast as a multi-agent RL problem (or a single-agent RL problem).





[2] Except in a vanishingly small set of scenarios.

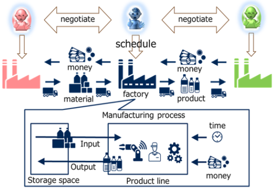
RL Motivation: For Automated Negotiation Researchers

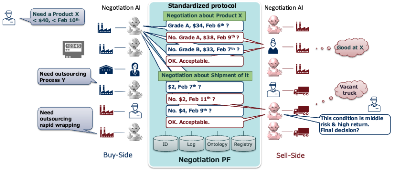
The need

- No general-purpose AN solution.
- No known equilibrium for bargaining with incomplete information [2].

The opportunity

- RL has been successfully applied to various domains, including games, robotics, and finance.
- AN can be cast as a multi-agent RL problem (or a single-agent RL problem).





[2] Except in a vanishingly small set of scenarios.

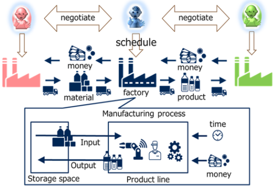
RL Motivation: For Automated Negotiation Researchers

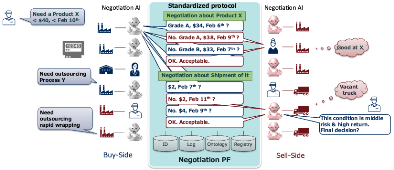
The need

- No general-purpose AN solution.
- No known equilibrium for bargaining with incomplete information [2].

The opportunity

- RL has been successfully applied to various domains, including games, robotics, and finance.
- AN can be cast as a multi-agent RL problem (or a single-agent RL problem).





[2] Except in a vanishingly small set of scenarios.

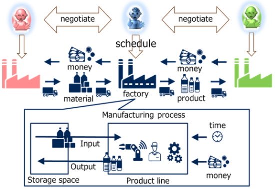
RL Motivation: For Automated Negotiation Researchers

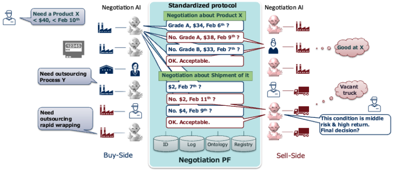
The need

- No general-purpose AN solution.
- No known equilibrium for bargaining with incomplete information [2].

The opportunity

- RL has been successfully applied to various domains, including games, robotics, and finance.
- AN can be cast as a multi-agent RL problem (or a single-agent RL problem).

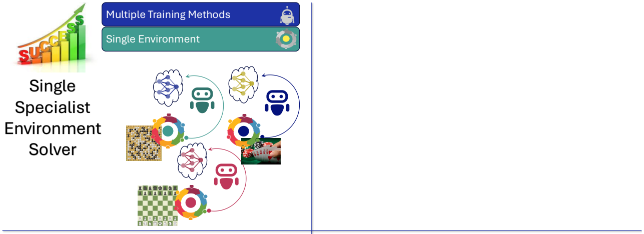




[2] Except in a vanishingly small set of scenarios.

AN Motivation: For Reinforcement Learning Researchers

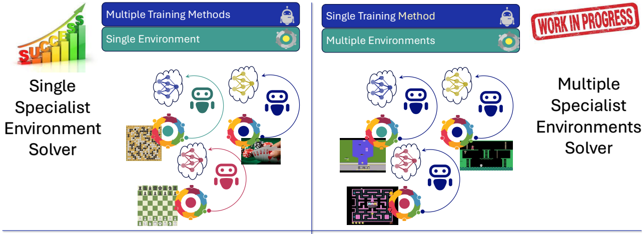
- Most successful RL applications focused on a single environment/game.
- Some work have been done in generalizing over environments.
- The **holly-grail** general game playing.
- A missing middle ground is a single environment that generate several **related** games.
- This regimes is very common in real-world business applications.



- Automated negotiation provides an environment that fits this missing quadrant.
- We already have simple baseline solutions

AN Motivation: For Reinforcement Learning Researchers

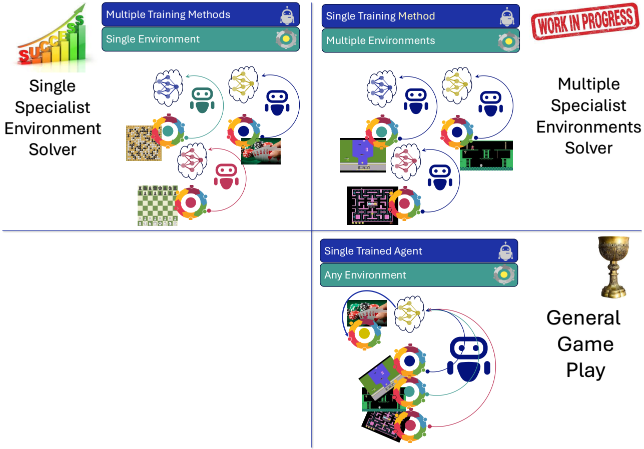
- Most successful RL applications focused on a single environment/game.
- Some work have been done in generalizing over environments.
- The **holly-grail** general game playing.
- A missing middle ground is a single environment that generate several **related** games.
- This regimes is very common in real-world business applications.



- Automated negotiation provides an environment that fits this missing quadrant.
- We already have simple baseline solutions

AN Motivation: For Reinforcement Learning Researchers

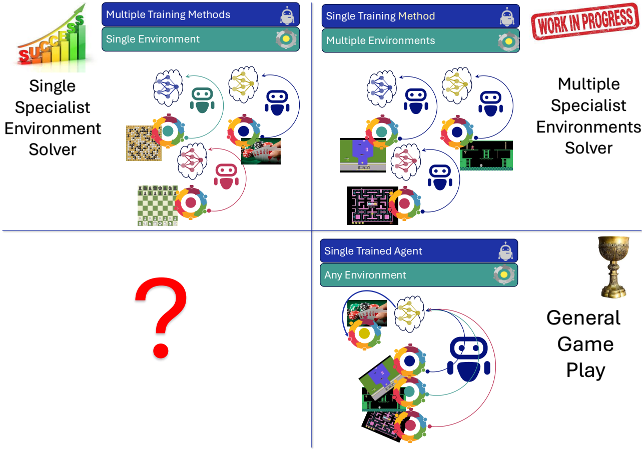
- Most successful RL applications focused on a single environment/game.
- Some work have been done in generalizing over environments.
- The **holly-grail** general game playing.
- A missing middle ground is a single environment that generate several **related** games.
- This regimes is very common in real-world business applications.



- Automated negotiation provides an environment that fits this missing quadrant.
- We already have simple baseline solutions

AN Motivation: For Reinforcement Learning Researchers

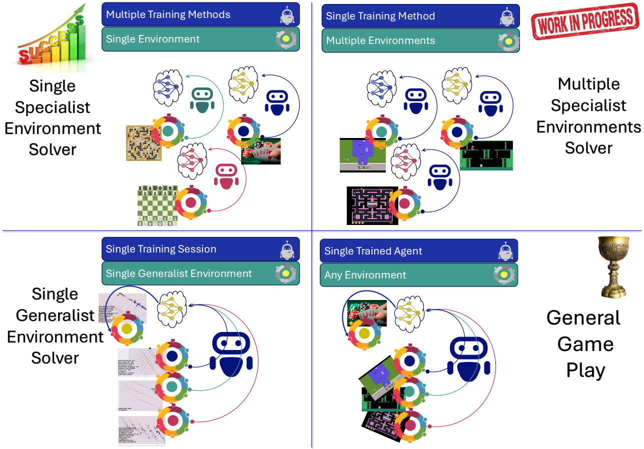
- Most successful RL applications focused on a single environment/game.
- Some work have been done in generalizing over environments.
- The **holly-grail** general game playing.
- A missing middle ground is a single environment that generate several **related** games.
- This regimes is very common in real-world business applications.



- Automated negotiation provides an environment that fits this missing quadrant.
- We already have simple baseline solutions

AN Motivation: For Reinforcement Learning Researchers

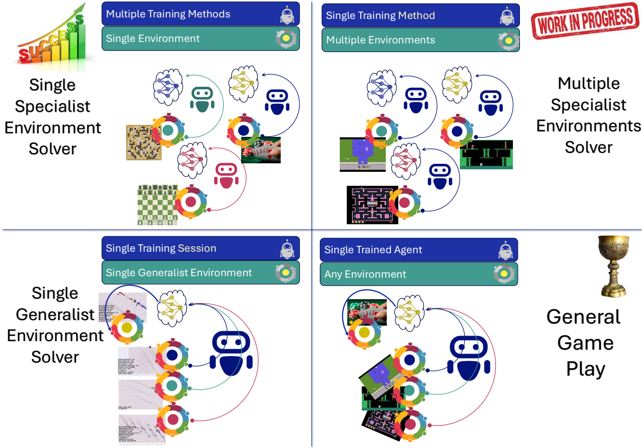
- Most successful RL applications focused on a single environment/game.
- Some work have been done in generalizing over environments.
- The **holly-grail** general game playing.
- A missing middle ground is a single environment that generate several **related** games.
- This regimes is very common in real-world business applications.



- Automated negotiation provides an environment that fits this missing quadrant.
- We already have simple baseline solutions

AN Motivation: For Reinforcement Learning Researchers

- Most successful RL applications focused on a single environment/game.
- Some work have been done in generalizing over environments.
- The **holly-grail** general game playing.
- A missing middle ground is a single environment that generate several **related** games.
- This regimes is very common in real-world business applications.

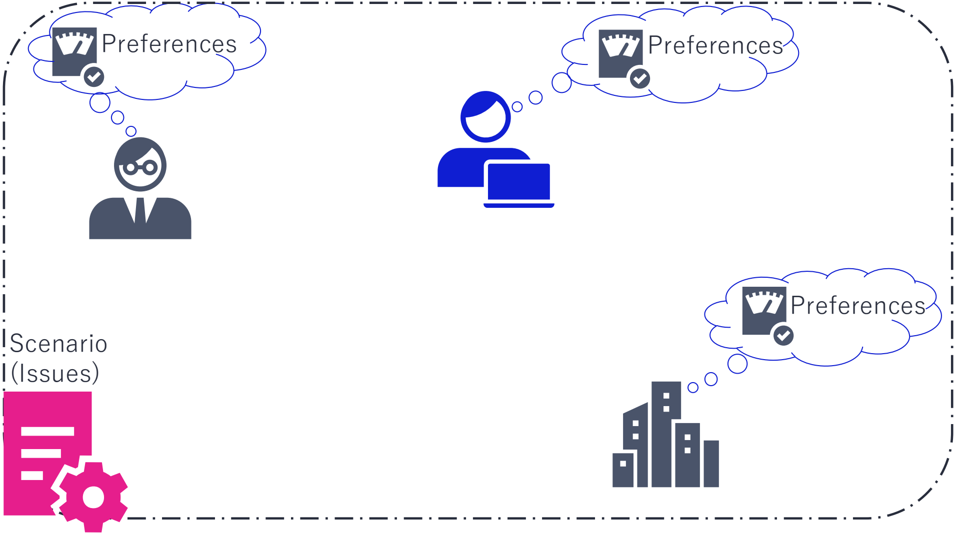


- Automated negotiation provides an environment that fits this missing quadrant.
- We already have simple baseline solutions
  - both using and not using RL.

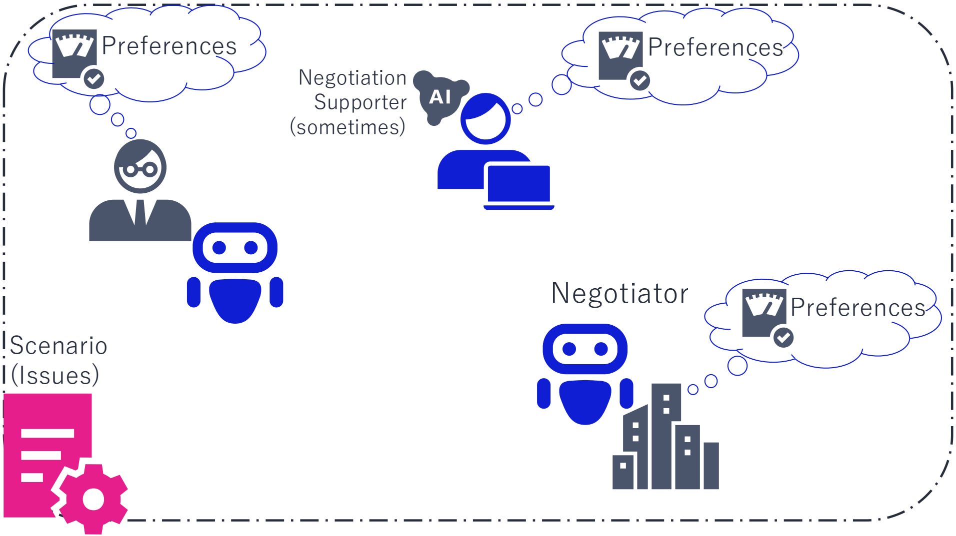
What is negotiation?

What is negotiation?

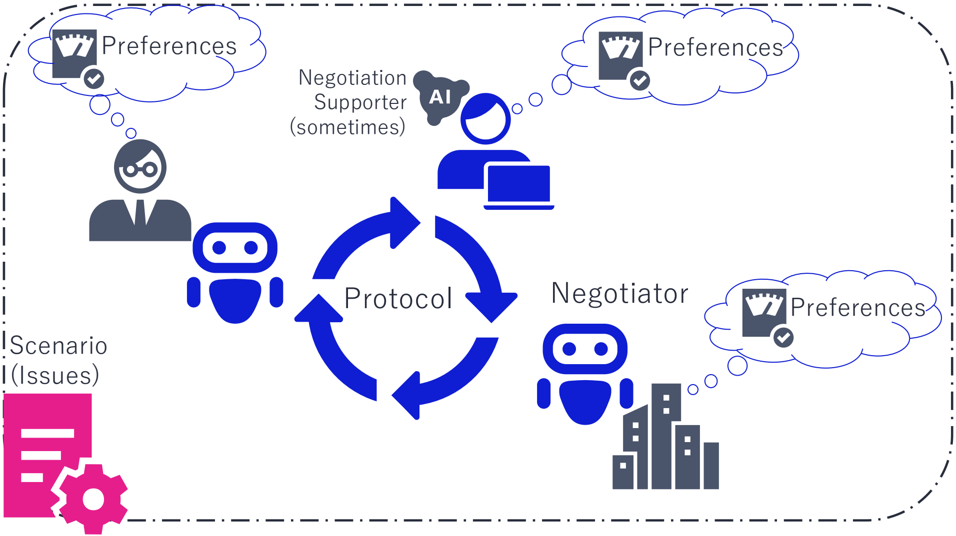
What is negotiation?



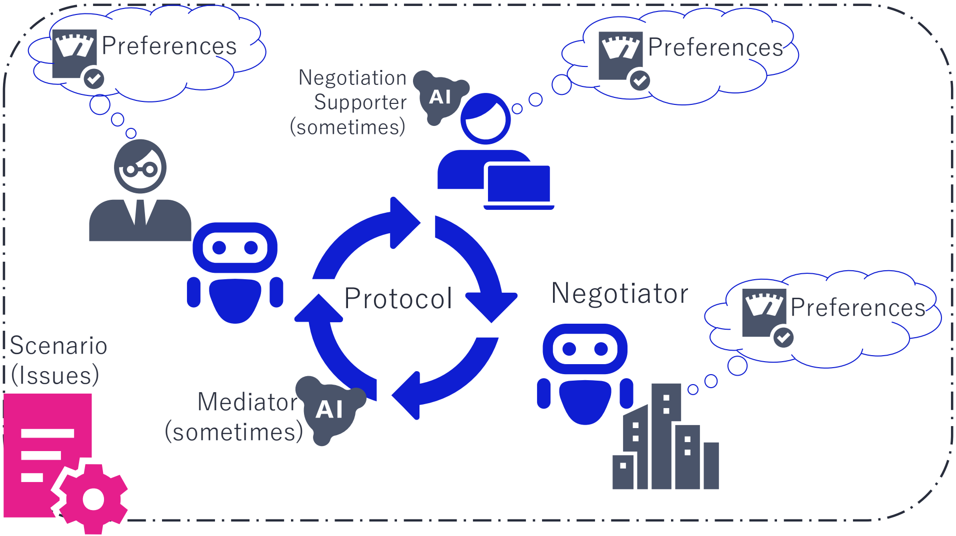
What is negotiation?



What is negotiation?



What is negotiation?



When do we need to negotiate?

- More than one actor (multiagent system).
- Actors have different **interests** represented by different **preferences**.
- There is a boundary: Actors cannot share information freely.
- Each actor thinks it may benefit from an agreement with the others.
- Actors can agree on a protocol and agenda for the negotiation.

When do we need to negotiate?

- More than one actor (multiagent system).
- Actors have different **interests** represented by different **preferences**.
- There is a boundary: Actors cannot share information freely.
- Each actor thinks it may benefit from an agreement with the others.
- Actors can agree on a protocol and agenda for the negotiation.

When do we need to negotiate?

- More than one actor (multiagent system).
- Actors have different **interests** represented by different **preferences**.
- There is a boundary: Actors cannot share information freely.
- Each actor thinks it may benefit from an agreement with the others.
- Actors can agree on a protocol and agenda for the negotiation.

When do we need to negotiate?

- More than one actor (multiagent system).
- Actors have different **interests** represented by different **preferences**.
- There is a boundary: Actors cannot share information freely.
- Each actor thinks it may benefit from an agreement with the others.
- Actors can agree on a protocol and agenda for the negotiation.

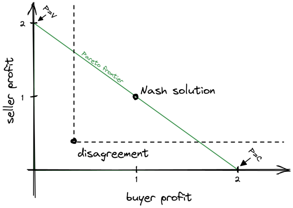
When do we need to negotiate?

- More than one actor (multiagent system).
- Actors have different **interests** represented by different **preferences**.
- There is a boundary: Actors cannot share information freely.
- Each actor thinks it may benefit from an agreement with the others.
- Actors can agree on a protocol and agenda for the negotiation.

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)



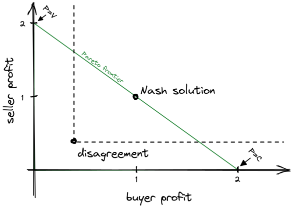
Sketch by Jackson de Campos

Slide by Amy Greenwald

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)

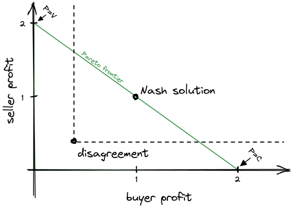


Sketch by Jackson de Campos

Slide by Amy Greenwald

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)



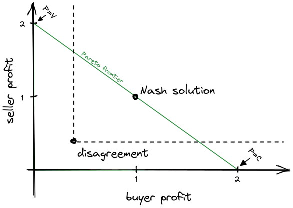
Sketch by Jackson de Campos

Slide by Amy Greenwald

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)

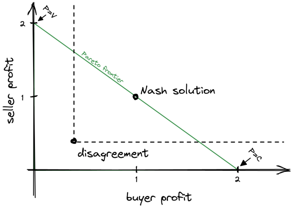


Sketch by Jackson de Campos

Slide by Amy Greenwald

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)



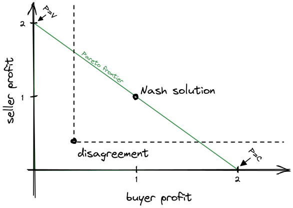
Sketch by Jackson de Campos

Slide by Amy Greenwald

- 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)

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)



Sketch by Jackson de Campos

Slide by Amy Greenwald

- 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)

Abstract Bargaining Problem

The two-agent bargaining problem can be defined abstractly by

- A set $F \subset \Omega$ of feasible outcomes
- Two agents with utility functions $u_1, u_2 : \Omega \to \mathbb{R}$
- A disagreement point $\phi \in \Omega$, also called the status quo. The value $u_i(\phi)$ is called agent i's reservation value.

Individual rationality assumption: No agent will ever agree to a utility below their reservation value.

An efficient outcome is one on the Pareto frontier, where neither agent can be made strictly better off without making the other worse off

Challenge: We seek a cooperative outcome (i.e., an efficient one) in a non-cooperative game

Abstract Bargaining Problem

The two-agent bargaining problem can be defined abstractly by

- A set $F \subset \Omega$ of feasible outcomes
- Two agents with utility functions $u_1, u_2 : \Omega \to \mathbb{R}$
- A disagreement point $\phi \in \Omega$, also called the status quo. The value $u_i(\phi)$ is called agent i's reservation value.

Individual rationality assumption: No agent will ever agree to a utility below their reservation value.

An efficient outcome is one on the Pareto frontier, where neither agent can be made strictly better off without making the other worse off

Challenge: We seek a cooperative outcome (i.e., an efficient one) in a non-cooperative game

Abstract Bargaining Problem

The two-agent bargaining problem can be defined abstractly by

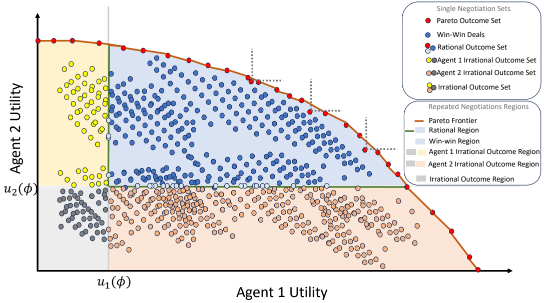
- A set $F \subset \Omega$ of feasible outcomes
- Two agents with utility functions $u_1, u_2 : \Omega \to \mathbb{R}$
- A disagreement point $\phi \in \Omega$, also called the status quo. The value $u_i(\phi)$ is called agent i's reservation value.

Individual rationality assumption: No agent will ever agree to a utility below their reservation value.

An efficient outcome is one on the Pareto frontier, where neither agent can be made strictly better off without making the other worse off

Challenge: We seek a cooperative outcome (i.e., an efficient one) in a non-cooperative game

Important Concepts



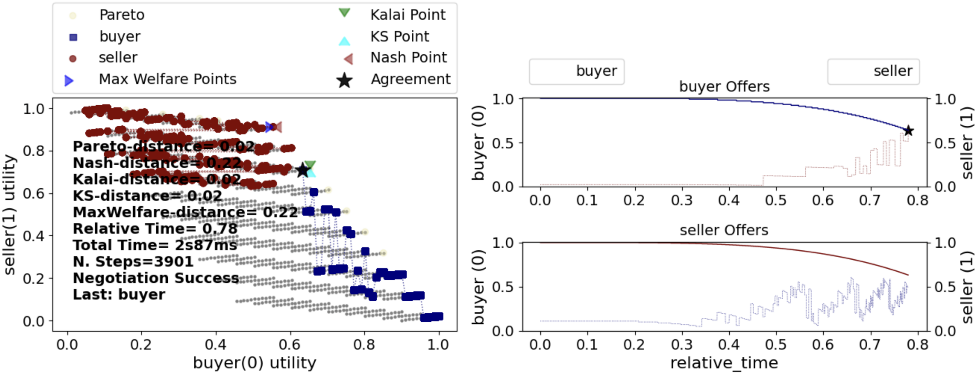
Pareto Frontier Outcomes that cannot be improved for one actor without making another worse off.

Welfare Total utility received by all actors.

Surplus utility Utility above disagreement utility.

Nash Equilibrium Strategies that are best responses to each other.

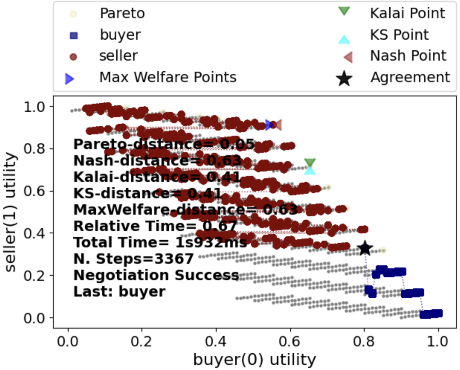
Visualizing a Negotiation

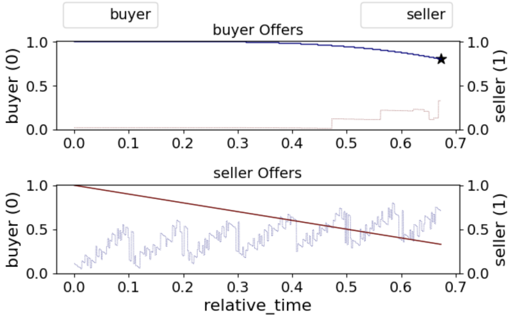


A buyer and a seller negotiating price, quantity and delivery date.

- Is this a zero-sum game?
- What are the reservation values?
- Is this a **good** result?

Visualizing a Negotiation





A buyer and a seller negotiating price, quantity and delivery date.

- Is this a zero-sum game?
- What are the reservation values?
- Is this a **good** result?

von Neumann-Morgenstern Utility Theorem

It is natural to express agent's preferences as comparisons: e.g., "I prefer apples to bananas."
We can also compare lotteries (i.e., randomized outcomes): e.g., when I am very hungry, "I prefer a banana with probability 90% to an apple with probability 50%."

Theorem[3] Given an agent with preferences over randomized outcomes that satisfy various axioms,
there exists a unique utility function $u : \Omega \to \mathbb{R}$ s.t. $\sigma \succ \tau$ iff $\mathbb{E}[u(\sigma)] > \mathbb{E}[u(\tau)]$), up to scaling.
The axioms are completeness, transitivity, continuity, and IIA.

Why is this relevant?

- Justifies focusing on bargaining assuming utility functions (hereafter, ufuns).
- Justifies modelling the preferences of negotiation partners (hereafter, opponents) via ufuns.

J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1947

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).

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).

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 Point (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 Point (1975): The Pareto outcome with equal ratios of achieved surplus utility and maximum feasible surplus utility

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

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

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

Other Bargaining Solutions

- Nash Point (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 Point (1975): The Pareto outcome with equal ratios of achieved surplus utility and maximum feasible surplus utility

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

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

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

Other Bargaining Solutions

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

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

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

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

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

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

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: 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: 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: 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: 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: 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: 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: 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*:

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

Linear Discounting

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

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

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*:

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

Linear Discounting

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

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

Myerson-Satterthwaite Impossibility Result

Desirata

- Ex-post Efficient outcome.
- Individual rationality (IR).
- (Nash Equilibrium) Incentive compatible (IC).
- Budget balance (BB).

Theorem

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.*

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

Summary of Classical Results

Solved

- Singe-shot bilateral negotiation with complete information (Nash/Kalai/Kalai-Smorodinsky Solutions).
- Infinite horizon bilateral negotiation with time-pressure and complete information (Rubinstein's SPE).
- Sometimes we still need negotiations even with complete information (Hick's Paradox).
- Incomplete information: Myerson-Satterthwaite Impossibility Result.

Open

- What is the PBE of time-limited bargaining with unknown partner utility function?
- What is the PBE of time-limited bargaining with partially-known partner utility function?

Issues and Outcomes

Cartesian Outcome Space

The Cartesian product of a set of issues:
$\Omega = I_0 \times I_1 \times \cdots \times I_{N-1}$.

Issue Types

Catgorical Set of values: $\{v_i | v_i \in I\}$

Ordinal with defined order

Cardinal with defined difference

Numeric with defined numeric value (integer/real)

Issues and Outcomes

Cartesian Outcome Space

The Cartesian product of a set of issues:
$\Omega = I_0 \times I_1 \times \cdots \times I_{N-1}$.

Issue Types

Catgorical Set of values: $\{v_i | v_i \in I\}$

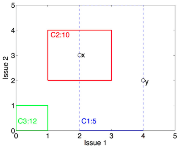
Ordinal with defined order

Cardinal with defined difference

Numeric with defined numeric value (integer/real)

Preferences and Utility Functions

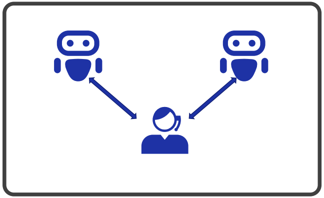
- Partial Ordering $\omega_i \succeq \omega_j \forall \omega_i, \omega_j \in \Omega$
- Full Ordering $\omega_i \succ \omega_j \forall \omega_i, \omega_j \in \Omega$
- Cardinal $\delta_{ij} = \omega_i - \omega_j \in \Re \forall \omega_i, \omega_j \in \Omega$
- Utility Function $u(\omega) \in \Re \ \forall \omega \in \Omega$
- Normalized Utility Function $u(\omega) \in [0,1] \forall \omega \in \Omega$
- Linear UFuns $u(\omega) = \sum_{i=0}^{|\omega|} \alpha_i \times \omega_i$
- Linear Additive UFuns $u(\omega) = \sum_{i=0}^{|\omega|} \omega_i \times f_i(\omega_i)$
- Generalized Additive UFuns $u(\omega) = \sum_{i=0}^{K} \omega_k \times f_k(\omega_j \forall j \in G_k)$
- Hyper Rectangle UFuns $u(\omega) = \sum_{k=0}^{K} c_k \times \delta[\omega \in C_k]$
- Genrealized Hyper Rectangle UFuns $u(\omega) = \sum_{k=0}^{K} f_k(\omega) \times \delta[\omega \in C_k]$



Mediated Protocols

Main Features

- Has A central *mediator*.
- Agents negotiate by exchanging *messages* with the *mediator*.
- Proposals can come from the mediator or the negotiators.



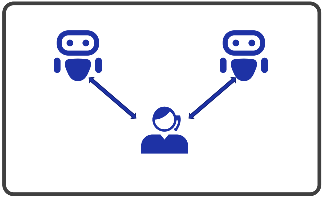
Examples

Single Text Protocol The mediator proposes a single hypothetical agreements, gets feedback about it and modifies it based on this feedback.

Mediated Protocols

Main Features

- Has A central *mediator*.
- Agents negotiate by exchanging *messages* with the *mediator*.
- Proposals can come from the mediator or the negotiators.



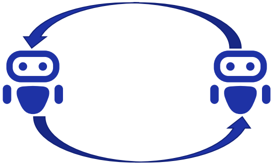
Examples

Single Text Protocol The mediator proposes a single hypothetical agreements, gets feedback about it and modifies it based on this feedback.

Unmediated Protocols

Main Features

- No central coordinator.
- Agents negotiate by exchanging *messages*.
- All proposals come from negotiators.



Examples

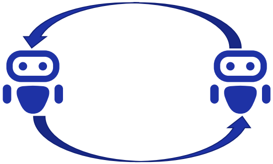
Nash Bargaining Game	Single iteration, single issue, bilateral protocol with complete information.
Rubinstein Bargaining Protocol	Infinite horizon, single issue, bilateral protocol with complete information[4].
Alternating Offers Protocol	Finite horizon, multi-issue, multilateral protocol with partial information[5].

Ariel Rubinstein. "Perfect equilibrium in a bargaining model". In: *Econometrica: Journal of the Econometric Society* (1982), pp. 97–109
Reyhan Aydoğan et al. "Alternating offers protocols for multilateral negotiation". In: *Modern Approaches to Agent-based Complex Automated Negotiation*. Springer, 2017, pp. 153–167

Unmediated Protocols

Main Features

- No central coordinator.
- Agents negotiate by exchanging *messages*.
- All proposals come from negotiators.



Examples

Nash Bargaining Game Single iteration, single issue, bilateral protocol with complete information.

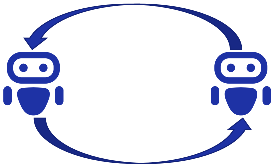
Rubinstein Bargaining Protocol Infinite horizon, single issue, bilateral protocol with complete information[4].

Alternating Offers Protocol Finite horizon, multi-issue, multilateral protocol with partial information[5].

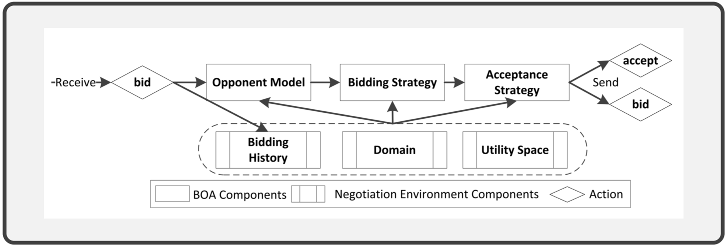
Ariel Rubinstein. "Perfect equilibrium in a bargaining model". In: *Econometrica: Journal of the Econometric Society* (1982), pp. 97–109
Reyhan Aydoğan et al. "Alternating offers protocols for multilateral negotiation". In: *Modern Approaches to Agent-based Complex Automated Negotiation*. Springer, 2017, pp. 153–167

Stacked Alternating Offers Protocol


```python
n_agreed, current = 0, randint(0, n_agents)
offer = agents[current].offer()
while not timeout()
        current = (current + 1) % n_agents
        response = agents[current].respond(offer)
        if response == 'accept':
                n_agreed += 1
        if n_agreed == n_agents:
                return offer
        elif response == 'end_negotiation':
                return 'failed'
        elif response == 'reject':
                offer = agents[current].offer()
return "timedout"
```



Negotiator Components



BOA Architecture

Opponent Model Learns about the partner/opponent.

Offer Policy Generates new bids, Also called Offer Policy

Acceptance Policy Decides when to accept, Also called Acceptance Policy.

Tim Baarslag et al. "Decoupling Negotiating Agents to Explore the Space of Negotiation Strategies". In: *Novel Insights in Agent-based Complex Automated Negotiation*. Ed. by Ivan Marsa-Maestre et al. Tokyo: Springer Japan, 2014, pp. 61–83. ISBN: 978-4-431-54758-7. DOI: 10.1007/978-4-431-54758-7_4. URL: https://doi.org/10.1007/978-4-431-54758-7_4

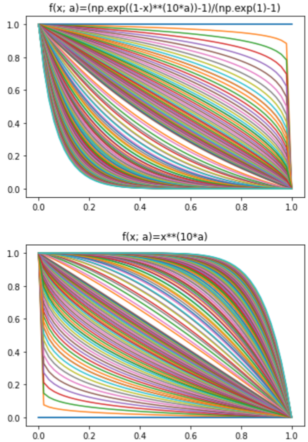
Time-based Offer Policy

Time-based strategies

- The negotiator's offers and decisions (acceptance, ending) depend **only** on the relative negotiation time.
- Monotonically decreasing utility (usually).
- Usually requires an inverse utility function.

Common TB Strategies

- Boulware: Slow then fast concession (i.e. $a > 1$)
- Linear: Linear concession (i.e. $a = 1$)
- Conceder Fast then slow concession (i.e. $a < 1$)



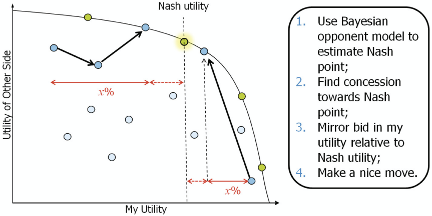
Behavioral Offer Policies

Behavior Based Strategies

- Responds to the opponent offers.
- Usualy Tit-for-Tat.
- Usually requires an opponent model.

(Nice) Tit-for-Tat (bilateral) [6]

Concede as much as the opponent toward the estimated nash-point and do not retaliate.



Tim Baarslag, Koen Hindriks, and Catholijn Jonker. "A tit for tat negotiation strategy for real-time bilateral negotiations". In: *Complex Automated Negotiations: Theories, Models, and Software Competitions*. Springer, 2013, pp. 229–233

Acceptance Policy

Accept if $\alpha u(\omega) + \beta$ is greater than:

- **Threshold** a utility threshold (τ).
  - **Constant** May be a fraction of maximum utility ($AC_{const}(\gamma)$).
  - **Time-based** Monotonically non-increasing with time ($AC_{monotonic}(t)$).
- **Last** my last offer (AC_{last}).
- **Next** what I am about to offer (AC_{next}).
- **Best** the best offer I received in a given window ($AC_{best}(K)$).
- **Average** the average utility I received in a given window ($AC_{avg}(K)$).
- **Time** the reserved value and $T \in [0, 1]$ fraction of the negotiation have passed ($AC_{time}(T)$)
- **Expected** the best offer I expect to receive (e.g. Gaussian Processes, needs opponent offer and acceptance policies).

Acceptance Policy

Accept if $\alpha u(\omega) + \beta$ is greater than:

- **Threshold** a utility threshold (τ).
  - **Constant** May be a fraction of maximum utility ($AC_{const}(\gamma)$).
  - **Time-based** Monotonically non-increasing with time ($AC_{monotonic}(t)$).
- **Last** my last offer (AC_{last}).
- **Next** what I am about to offer (AC_{next}).
- **Best** the best offer I received in a given window ($AC_{best}(K)$).
- **Average** the average utility I received in a given window ($AC_{avg}(K)$).
- **Time** the reserved value and $T \in [0, 1]$ fraction of the negotiation have passed ($AC_{time}(T)$)
- **Expected** the best offer I expect to receive (e.g. Gaussian Processes, needs opponent offer and acceptance policies).

Acceptance Policy

Accept if $\alpha u(\omega) + \beta$ is greater than:

- **Threshold** a utility threshold (τ).
  - **Constant** May be a fraction of maximum utility ($AC_{const}(\gamma)$).
  - **Time-based** Monotonically non-increasing with time ($AC_{monotonic}(t)$).
- **Last** my last offer (AC_{last}).
- **Next** what I am about to offer (AC_{next}).
- **Best** the best offer I received in a given window ($AC_{best}(K)$).
- **Average** the average utility I received in a given window ($AC_{avg}(K)$).
- **Time** the reserved value and $T \in [0, 1]$ fraction of the negotiation have passed ($AC_{time}(T)$)
- **Expected** the best offer I expect to receive (e.g. Gaussian Processes, needs opponent offer and acceptance policies).

Acceptance Policy

Accept if $\alpha u(\omega) + \beta$ is greater than:

- **Threshold** a utility threshold (τ).
  - **Constant** May be a fraction of maximum utility ($AC_{const}(\gamma)$).
  - **Time-based** Monotonically non-increasing with time ($AC_{monotonic}(t)$).
- **Last** my last offer (AC_{last}).
- **Next** what I am about to offer (AC_{next}).
- **Best** the best offer I received in a given window ($AC_{best}(K)$).
- **Average** the average utility I received in a given window ($AC_{avg}(K)$).
- **Time** the reserved value and $T \in [0, 1]$ fraction of the negotiation have passed ($AC_{time}(T)$)
- **Expected** the best offer I expect to receive (e.g. Gaussian Processes, needs opponent offer and acceptance policies).

Acceptance Policy

Accept if $\alpha u(\omega) + \beta$ is greater than:

- **Threshold** a utility threshold (τ).
  - **Constant** May be a fraction of maximum utility ($AC_{const}(\gamma)$).
  - **Time-based** Monotonically non-increasing with time ($AC_{monotonic}(t)$).
- **Last** my last offer (AC_{last}).
- **Next** what I am about to offer (AC_{next}).
- **Best** the best offer I received in a given window ($AC_{best}(K)$).
- **Average** the average utility I received in a given window ($AC_{avg}(K)$).
- **Time** the reserved value and $T \in [0, 1]$ fraction of the negotiation have passed ($AC_{time}(T)$)
- **Expected** the best offer I expect to receive (e.g. Gaussian Processes, needs opponent offer and acceptance policies).

Acceptance Policy

Accept if $\alpha u(\omega) + \beta$ is greater than:

- **Threshold** a utility threshold (τ).
  - **Constant** May be a fraction of maximum utility ($AC_{const}(\gamma)$).
  - **Time-based** Monotonically non-increasing with time ($AC_{monotonic}(t)$).
- **Last** my last offer (AC_{last}).
- **Next** what I am about to offer (AC_{next}).
- **Best** the best offer I received in a given window ($AC_{best}(K)$).
- **Average** the average utility I received in a given window ($AC_{avg}(K)$).
- **Time** the reserved value and $T \in [0, 1]$ fraction of the negotiation have passed ($AC_{time}(T)$)
- **Expected** the best offer I expect to receive (e.g. Gaussian Processes, needs opponent offer and acceptance policies).

Acceptance Policy

Accept if $\alpha u(\omega) + \beta$ is greater than:

- **Threshold** a utility threshold (τ).
  - **Constant** May be a fraction of maximum utility ($AC_{const}(\gamma)$).
  - **Time-based** Monotonically non-increasing with time ($AC_{monotonic}(t)$).
- **Last** my last offer (AC_{last}).
- **Next** what I am about to offer (AC_{next}).
- **Best** the best offer I received in a given window ($AC_{best}(K)$).
- **Average** the average utility I received in a given window ($AC_{avg}(K)$).
- **Time** the reserved value and $T \in [0, 1]$ fraction of the negotiation have passed ($AC_{time}(T)$)
- **Expected** the best offer I expect to receive (e.g. Gaussian Processes, needs opponent offer and acceptance policies).

Acceptance Policy

Accept if $\alpha u(\omega) + \beta$ is greater than:

- **Threshold** a utility threshold (τ).
  - **Constant** May be a fraction of maximum utility ($AC_{const}(\gamma)$).
  - **Time-based** Monotonically non-increasing with time ($AC_{monotonic}(t)$).
- **Last** my last offer (AC_{last}).
- **Next** what I am about to offer (AC_{next}).
- **Best** the best offer I received in a given window ($AC_{best}(K)$).
- **Average** the average utility I received in a given window ($AC_{avg}(K)$).
- **Time** the reserved value and $T \in [0, 1]$ fraction of the negotiation have passed ($AC_{time}(T)$)
- **Expected** the best offer I expect to receive (e.g. Gaussian Processes, needs opponent offer and acceptance policies).

Acceptance Policy

Accept if $\alpha u(\omega) + \beta$ is greater than:

- **Threshold** a utility threshold (τ).
  - **Constant** May be a fraction of maximum utility ($AC_{const}(\gamma)$).
  - **Time-based** Monotonically non-increasing with time ($AC_{monotonic}(t)$).
- **Last** my last offer (AC_{last}).
- **Next** what I am about to offer (AC_{next}).
- **Best** the best offer I received in a given window ($AC_{best}(K)$).
- **Average** the average utility I received in a given window ($AC_{avg}(K)$).
- **Time** the reserved value and $T \in [0, 1]$ fraction of the negotiation have passed ($AC_{time}(T)$)
- **Expected** the best offer I expect to receive (e.g. Gaussian Processes, needs opponent offer and acceptance policies).

Acceptance Policy

Accept if $\alpha u(\omega) + \beta$ is greater than:

- **Threshold** a utility threshold (τ).
  - **Constant** May be a fraction of maximum utility ($AC_{const}(\gamma)$).
  - **Time-based** Monotonically non-increasing with time ($AC_{monotonic}(t)$).
- **Last** my last offer (AC_{last}).
- **Next** what I am about to offer (AC_{next}).
- **Best** the best offer I received in a given window ($AC_{best}(K)$).
- **Average** the average utility I received in a given window ($AC_{avg}(K)$).
- **Time** the reserved value and $T \in [0, 1]$ fraction of the negotiation have passed ($AC_{time}(T)$)
- **Expected** the best offer I expect to receive (e.g. Gaussian Processes, needs opponent offer and acceptance policies).

Combining Acceptance Policies

Combined Acceptance Strategy[7]

- Combines multiple simple acceptance policies.
- $AC_{combi}(\tau, \gamma) = AC_{next} \vee (AC_{time}(\tau) \wedge AC_{const}(\gamma))$
- $AC^{best}_{combi}(\tau, W) = AC_{next} \vee (AC_{time}(\tau) \wedge AC_{best}(W))$
- $AC^{avg}_{combi}(\tau, W) = AC_{next} \vee (AC_{time}(\tau) \wedge AC_{avg}(W))$
- $AC^{best}_{combi}(\tau) = AC_{next} \vee (AC_{time}(\tau) \wedge AC_{best}(T))$

NegMAS

```
ACCombi = ACNext() or (ACTime(tau) and ACConst(gamma))
ACBest = ACNext() or (ACTime(tau) and ACBest(w))
ACAvg = ACNext() or (ACTime(tau) and ACLastKReceived(K, op=math.mean))
ACBestAll = ACNext() or (ACTime(tau) and ACBest())
```


[7] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. "Effective acceptance conditions in real-time automated negotiation". In: *Decision Support Systems* 60 (2014), pp. 68–77

Opponent Modeling

Opponent Components

- Opponent preferences $u^o(\omega) \forall \omega \in \Omega$
- Offer Policy $\pi^o(s)$
- Acceptance Policy $a(\omega, s)$

What is being modeled?

- Any of the 3 components.
- Opponent Type.

When is it modeled?

- Before the negotiation: Static Model.
- During the negotiation: Dynamic Model.

Data

- This opponent vs. this opponent group vs. all opponents.
- Only agreements vs. All exchanged offers.

Opponent Modeling

Opponent Components

- Opponent preferences $u^o(\omega)\forall\omega \in \Omega$
- Offer Policy $\pi^o(s)$
- Acceptance Policy $a(\omega, s)$

When is it modeled?

- Before the negotiation: Static Model.
- During the negotiation: Dynamic Model.

Data

- This opponent vs. this opponent group vs. all opponents.
- Only agreements vs. All exchanged offers.

What is being modeled?

- Any of the 3 components.
- Opponent Type.

Opponent Modeling

Opponent Components

- Opponent preferences $u^o(\omega) \forall \omega \in \Omega$
- Offer Policy $\pi^o(s)$
- Acceptance Policy $a(\omega, s)$

What is being modeled?

- Any of the 3 components.
- Opponent Type.

When is it modeled?

- Before the negotiation: Static Model.
- During the negotiation: Dynamic Model.

Data

- This opponent vs. this opponent group vs. all opponents.
- Only agreements vs. All exchanged offers.

Some Automated Negotiation Platforms

Genius [8]

a Java-based negotiation platform to develop general negotiating agents and create negotiation scenarios.

GENIUS
General Environment for Negotiation with Intelligent multi-purpose Usage Simulation.

GeniusWeb

A distributed platform for automated negotiation on the internet

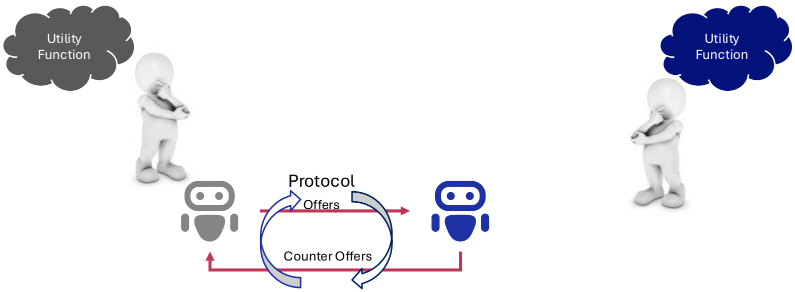
NegMAS [9]

a Python-based negotiation platform for developing autonomous negotiation agents embedded in simulation environments.

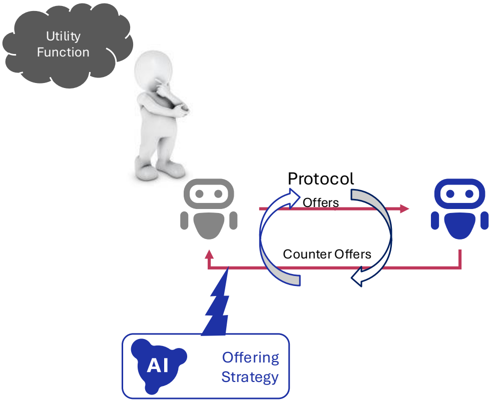
Raz Lin et al. "Genius: An Integrated Environment for Supporting the Design of Generic Automated Negotiators". In: *Computational Intelligence* 30.1 (2014), pp. 48–70. ISSN: 1467-8640. DOI: 10.1111/j.1467-8640.2012.00463.x. URL: http://dx.doi.org/10.1111/j.1467-8640.2012.00463.x

Yasser Mohammad, Amy Greenwald, and Shinji Nakadai. "NegMAS: A platform for situated negotiations". In: *Twelfth International Workshop on Agent-based Complex Automated Negotiations (ACAN2019) in conjunction with IJCAI*. Macau, China, 2019. URL: https://github.com/yasserfarouk/negmas

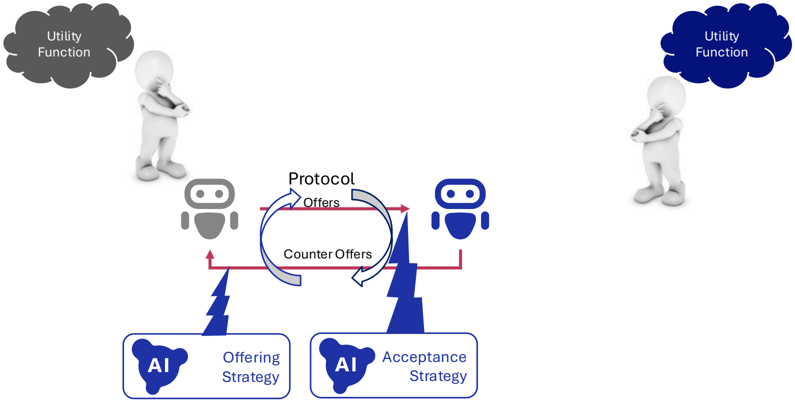
Role of ML in Automated Negotiation



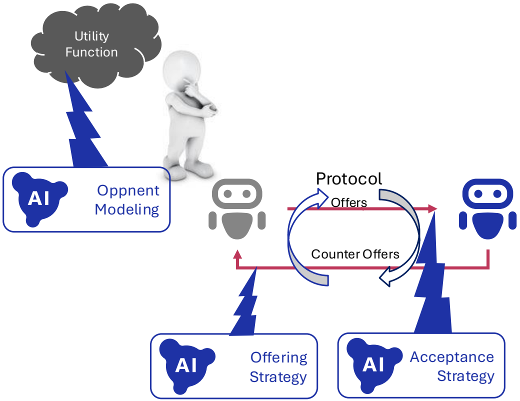
Role of ML in Automated Negotiation

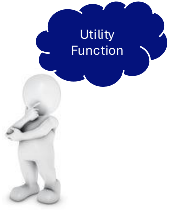


Role of ML in Automated Negotiation

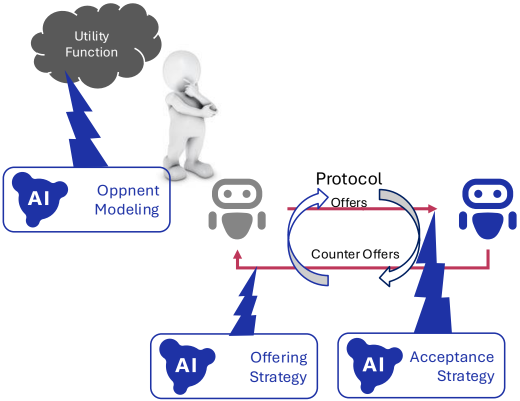


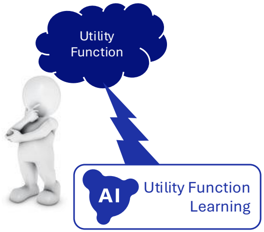
Role of ML in Automated Negotiation



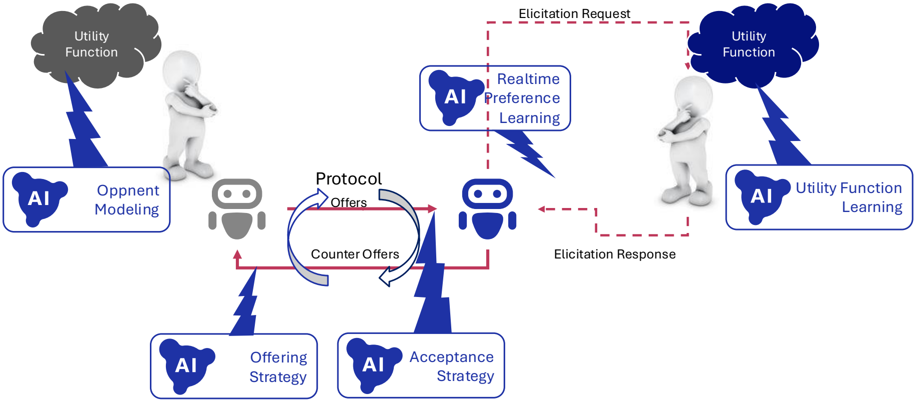


Role of ML in Automated Negotiation

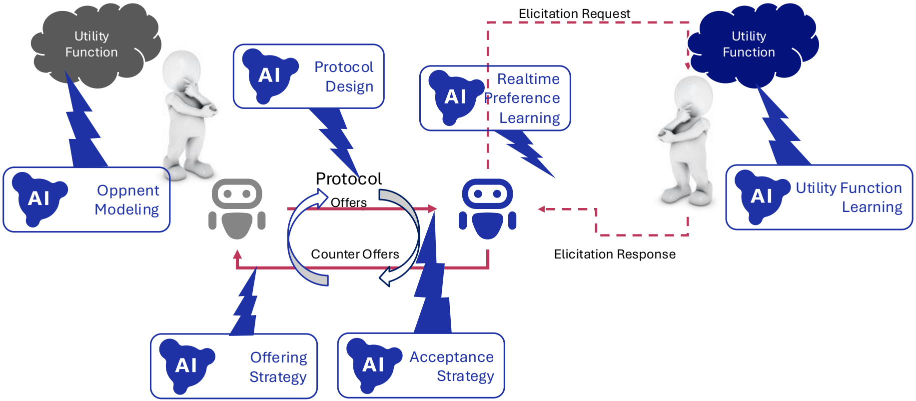




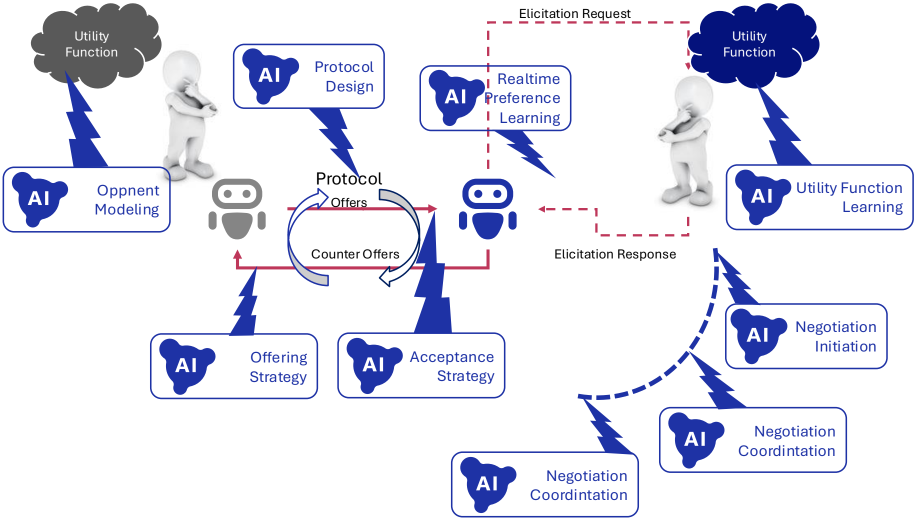
Role of ML in Automated Negotiation



Role of ML in Automated Negotiation



Role of ML in Automated Negotiation



ANAC Competition

Automated Negotiation League

Explore the strategies and difficulties in creating efficient agents whose primary purpose is to negotiate with other agent's strategies.

Details | CFP | Live | Results 2024

Supply Chain Management League

Design and build an autonomous agent that negotiates on behalf of a factory manager situated in a supply chain management simulation.

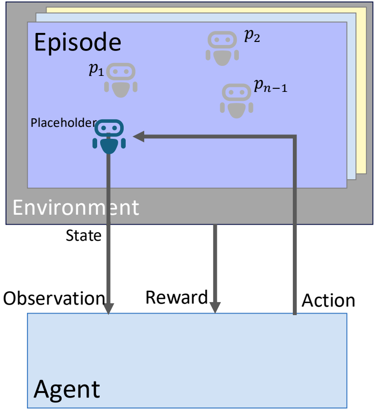
Details | CFP | Live | Results 2024

- ANAC is the **Automated Negotiating Agents Competition** running since 2010.
- This year it will be with IJCAI. Submission is open until June 1st.
- ANAC's 2024 demo is scheduled for Thursday (22nd) 10AM Demo Session.

Reinforcement Learning

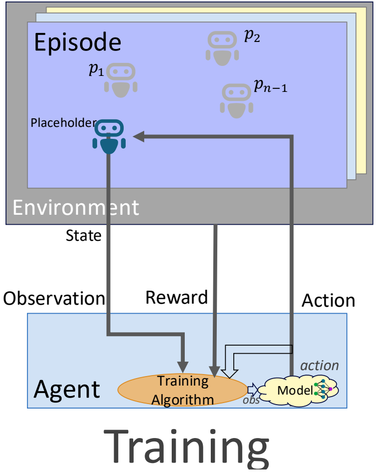
What is Reinforcement Learning?

- The agent receives state for an environment and sends back an action. It then receives a reward.
- Training consists of learning a **strategy** for maximizing expected accomulated rewards.
- DLR uses deep learning to train neural models that represent the strategy and/or the value of stats or stat/action pairs (Q-functions).
- MARL: Multiple agents are trained together:
  - Central Training Distributed Execution.



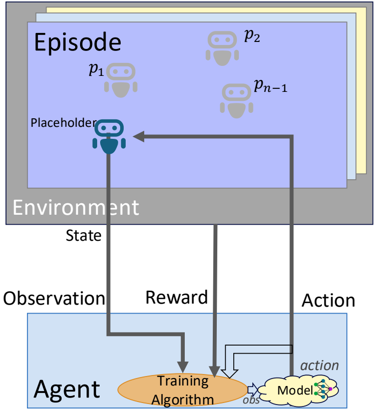
What is Reinforcement Learning?

- The agent receives state for an environment and sends back an action. It then receives a reward.
- Training consists of learning a **strategy** for maximizing expected accomulated rewards.
- DLR uses deep learning to train neural models that represent the strategy and/or the value of stats or stat/action pairs (Q-functions).
- MARL: Multiple agents are trained together:
  - Central Training Distributed Execution.



What is Reinforcement Learning?

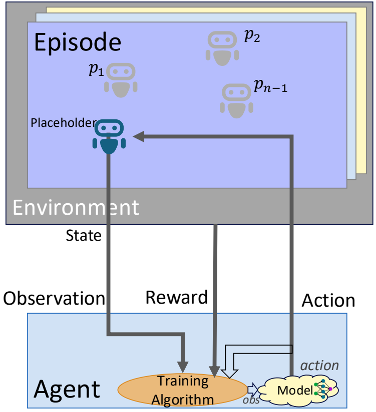
- The agent receives state for an environment and sends back an action. It then receives a reward.
- Training consists of learning a **strategy** for maximizing expected accomulated rewards.
- DLR uses deep learning to train neural models that represent the strategy and/or the value of stats or stat/action pairs (Q-functions).
- MARL: Multiple agents are trained together:
  - Central Training Distributed Execution.



Training

What is Reinforcement Learning?

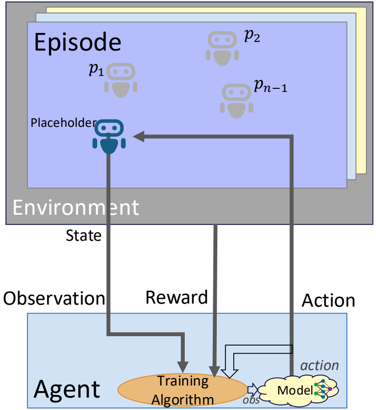
- The agent receives state for an environment and sends back an action. It then receives a reward.
- Training consists of learning a **strategy** for maximizing expected accomulated rewards.
- DLR uses deep learning to train neural models that represent the strategy and/or the value of stats or stat/action pairs (Q-functions).
- MARL: Multiple agents are trained together:
  - Central Training Distributed Execution.



Training

What is Reinforcement Learning?

- The agent receives state for an environment and sends back an action. It then receives a reward.
- Training consists of learning a **strategy** for maximizing expected accomulated rewards.
- DLR uses deep learning to train neural models that represent the strategy and/or the value of stats or stat/action pairs (Q-functions).
- MARL: Multiple agents are trained together:
  - Central Training Distributed Execution.



Training

A little more formal

Fully Observable Environment

A 4-tuple (S, A, P, R)

S State space.

A Action space.

$P(s, s')$ State transition function.

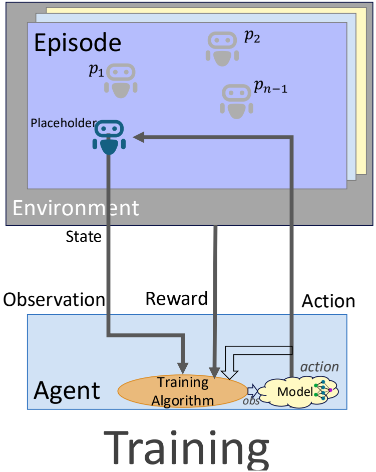
$R(s, a, s')$ Reward function.

The agent receives a state $s_t \in S$ and sends an action $a_t \in A$. It then receives a reward $r_t = R(s_t, a_t, s_{t+1})$ and the next state $s_{t+1} \sim P(s_t, a_t)$.

Objective

$$\arg\max_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right]$$

$$\arg\max_\pi \left[\sum_{t=0}^{H} R(s_t, a_t, s_{t+1})\right]$$



A little more formal

Partially Observable Environment

A 6-tuple $(S, A, P, R, \Omega, \mathcal{O})$

Ω Observation space.

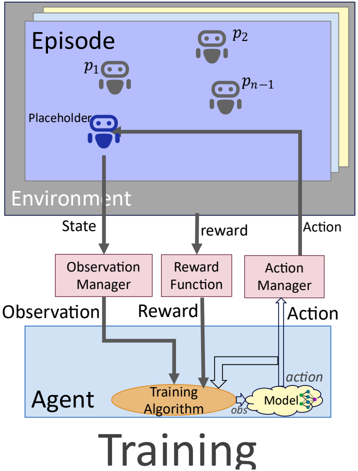
$\mathcal{O}(\omega|s)$ Observation model.

The agent receives an observation ω_t (sampled from the observation model) ...

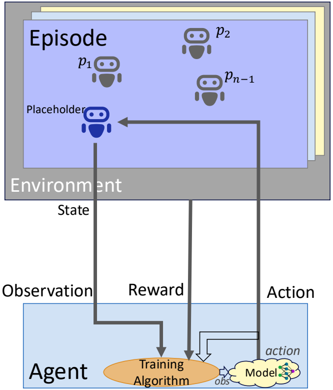
Objective

$\arg\max_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right]$

$\arg\max_\pi \left[\sum_{t=0}^{H} R(s_t, a_t, s_{t+1})\right]$

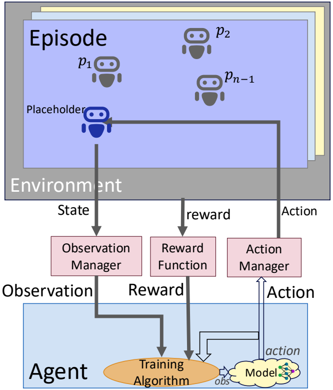


(Practical) Stages of RL



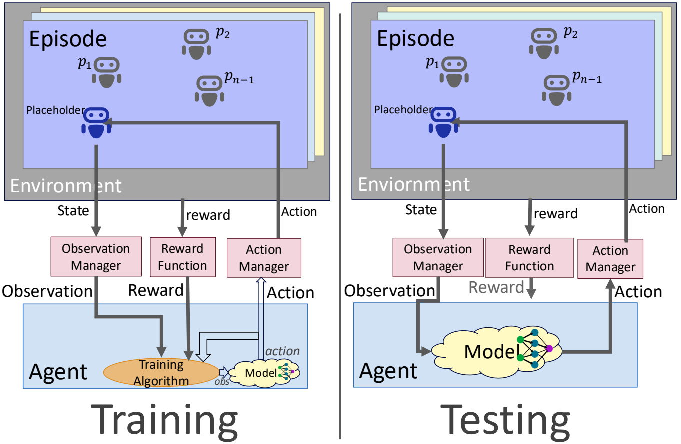
Training

(Practical) Stages of RL



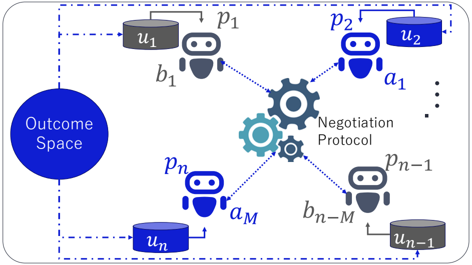
Training

(Practical) Stages of RL



Automated Negotiation as an RL Problem

The Negotiation Environment

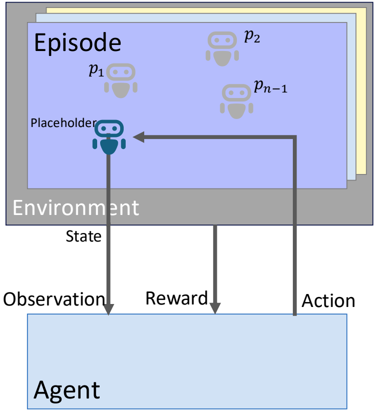


- A negotiation session is defined by the outcome-space, protocol, prefernces, and agent strategies.
- We can separate the partners ($p_{1:N}$) into:
  - Learners $a_{1:M}$ trained to improve their performance.
  - Background Agents $b_{1:N-M}$ not allowed to learn.

Automated Negotiation as an RL Problem

Mapping

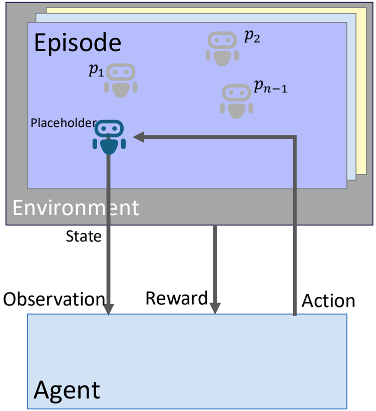
- Environment $\leftrightarrow$ Protocol + Partner strategies
- Agent $\leftrightarrow$ Negotiator
- State $\leftrightarrow$ Complete negotiation history (trace)
- Reward $\leftrightarrow$ Utility (if not shaped)
- Action $\leftrightarrow$ Action: Offer/Acceptance/Leaving (for AOP)
- Transition Function $\leftrightarrow$ Protocol step
- Episode $\leftrightarrow$ Negotiation Session
- Observation Function $\leftrightarrow$ ObservationManager



Automated Negotiation as an RL Problem

Mapping

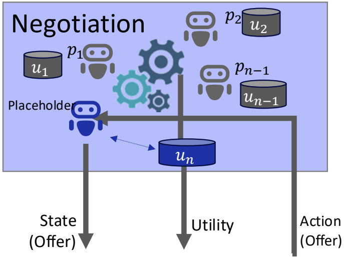
- Environment $\leftrightarrow$ Protocol + Partner strategies
- Agent $\leftrightarrow$ Negotiator
- State $\leftrightarrow$ Complete negotiation history (trace)
- Reward $\leftrightarrow$ Utility (if not shaped)
- Action $\leftrightarrow$ Action: Offer/Acceptance/Leaving (for AOP)
- Transition Function $\leftrightarrow$ Protocol step
- Episode $\leftrightarrow$ Negotiation Session
- Observation Function $\leftrightarrow$ ObservationManager



Automated Negotiation as an RL Problem

Mapping

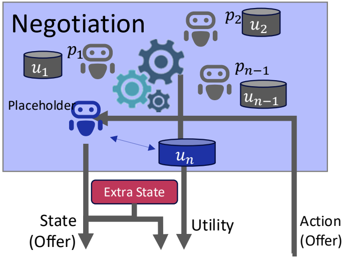
- Environment $\leftrightarrow$ Protocol + Partner strategies
- Agent $\leftrightarrow$ Negotiator
- State $\leftrightarrow$ Complete negotiation history (trace)
- Reward $\leftrightarrow$ Utility (if not shaped)
- Action $\leftrightarrow$ Action: Offer/Acceptance/Leaving (for AOP)
- Transition Function $\leftrightarrow$ Protocol step
- Episode $\leftrightarrow$ Negotiation Session
- Observation Function $\leftrightarrow$ ObservationManager



Automated Negotiation as an RL Problem

Mapping

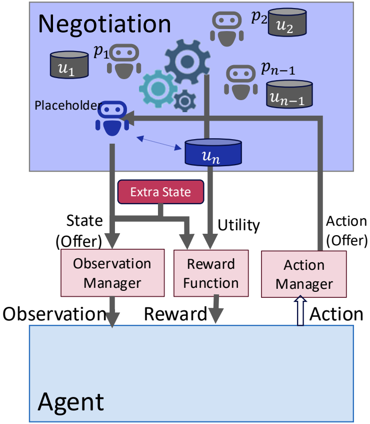
- Environment $\leftrightarrow$ Protocol + Partner strategies
- Agent $\leftrightarrow$ Negotiator
- State $\leftrightarrow$ Complete negotiation history (trace)
- Reward $\leftrightarrow$ Utility (if not shaped)
- Action $\leftrightarrow$ Action: Offer/Acceptance/Leaving (for AOP)
- Transition Function $\leftrightarrow$ Protocol step
- Episode $\leftrightarrow$ Negotiation Session
- Observation Function $\leftrightarrow$ ObservationManager



Automated Negotiation as an RL Problem

Mapping

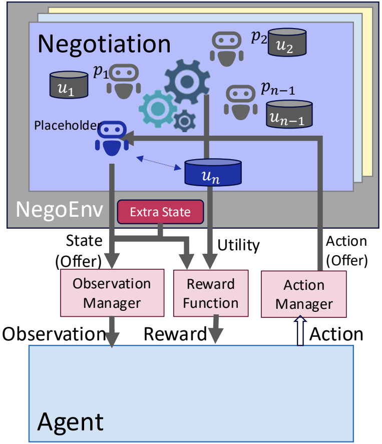
- Environment ↔ Protocol + Partner strategies
- Agent ↔ Negotiator
- State ↔ Complete negotiation history (trace)
- Reward ↔ Utility (if not shaped)
- Action ↔ Action: Offer/Acceptance/Leaving (for AOP)
- Transition Function ↔ Protocol step
- Episode ↔ Negotiation Session
- Observation Function ↔ ObservationManager



Automated Negotiation as an RL Problem

Mapping

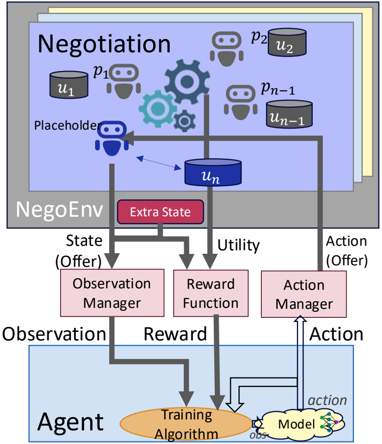
- Environment ↔ Protocol + Partner strategies
- Agent ↔ Negotiator
- State ↔ Complete negotiation history (trace)
- Reward ↔ Utility (if not shaped)
- Action ↔ Action: Offer/Acceptance/Leaving (for AOP)
- Transition Function ↔ Protocol step
- Episode ↔ Negotiation Session
- Observation Function ↔ ObservationManager



Automated Negotiation as an RL Problem

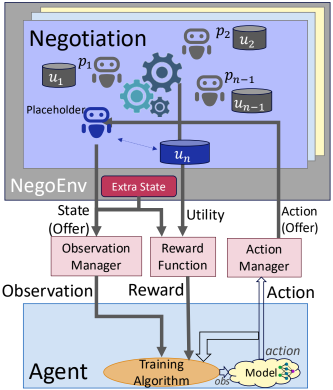
Mapping

- Environment $\leftrightarrow$ Protocol + Partner strategies
- Agent $\leftrightarrow$ Negotiator
- State $\leftrightarrow$ Complete negotiation history (trace)
- Reward $\leftrightarrow$ Utility (if not shaped)
- Action $\leftrightarrow$ Action: Offer/Acceptance/Leaving (for AOP)
- Transition Function $\leftrightarrow$ Protocol step
- Episode $\leftrightarrow$ Negotiation Session
- Observation Function $\leftrightarrow$ ObservationManager



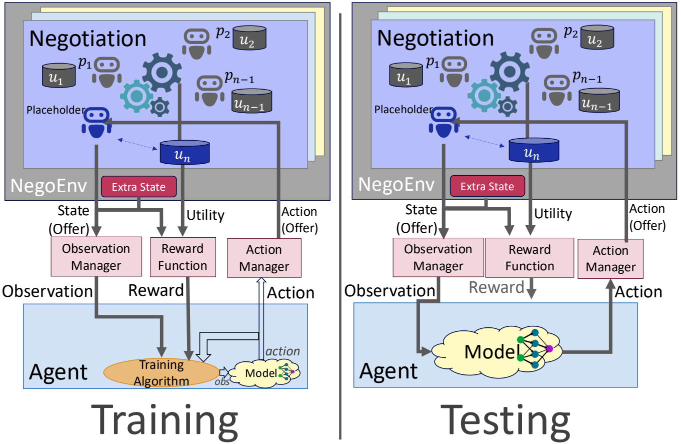
Training

Automated Negotiation as an RL Problem

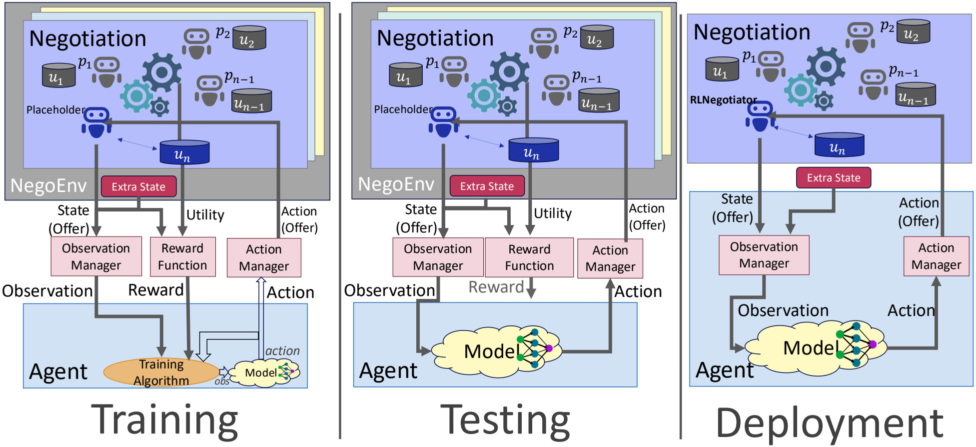


Training

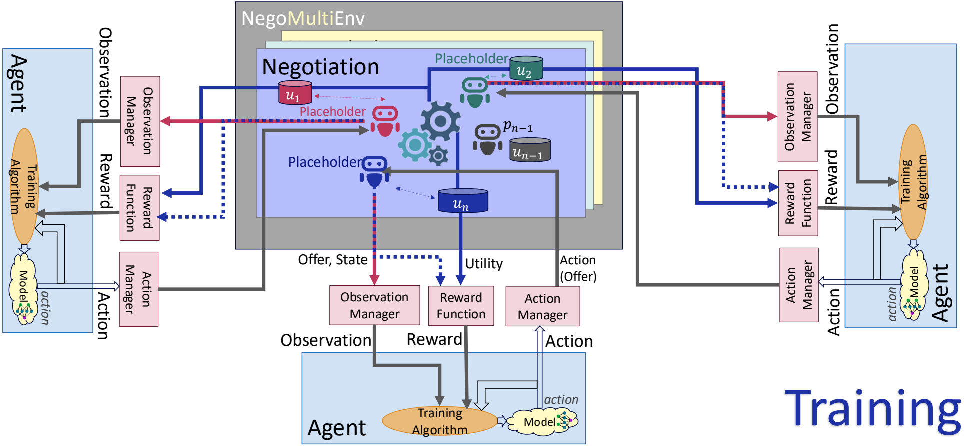
Automated Negotiation as an RL Problem



Automated Negotiation as an RL Problem

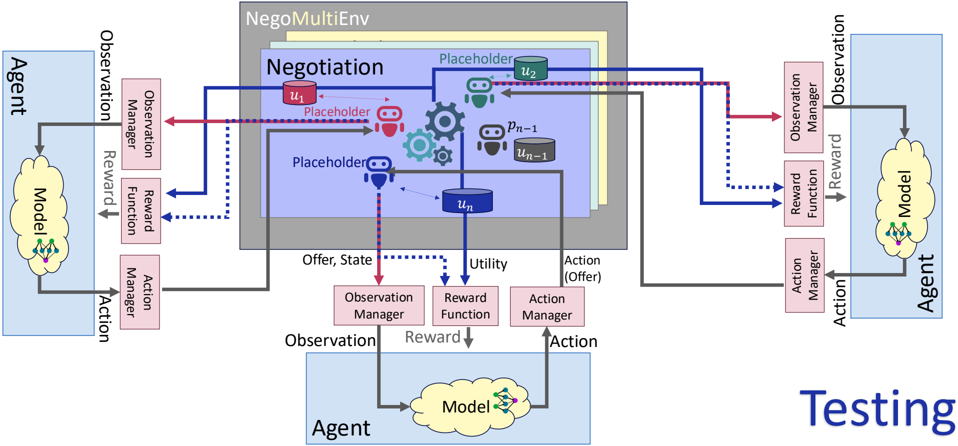


Automated Negotiation as a MARL Problem



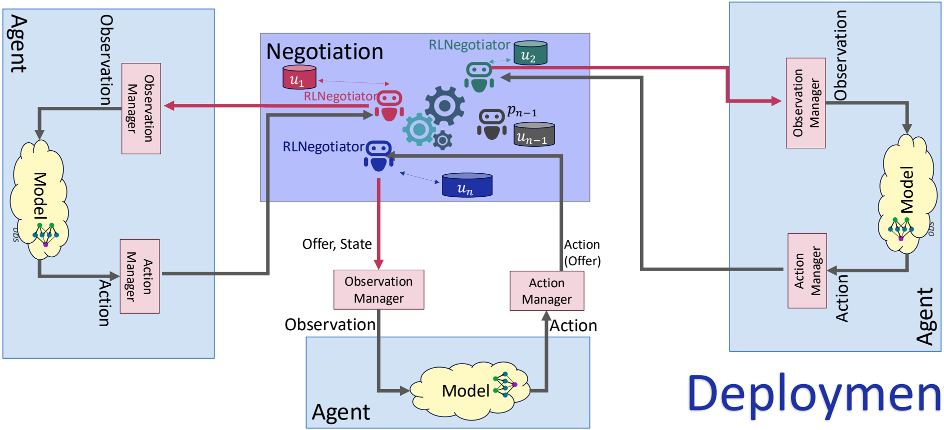
Training

Automated Negotiation as a MARL Problem



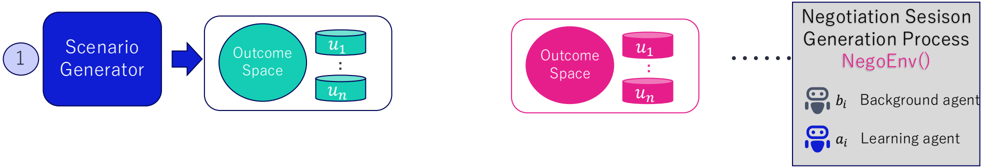
Testing

Automated Negotiation as a MARL Problem

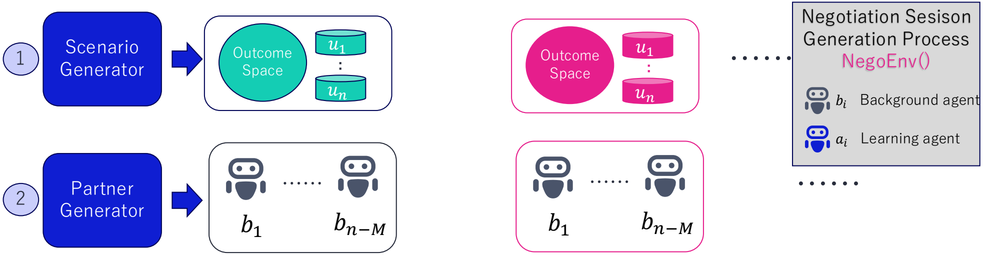


Deployment

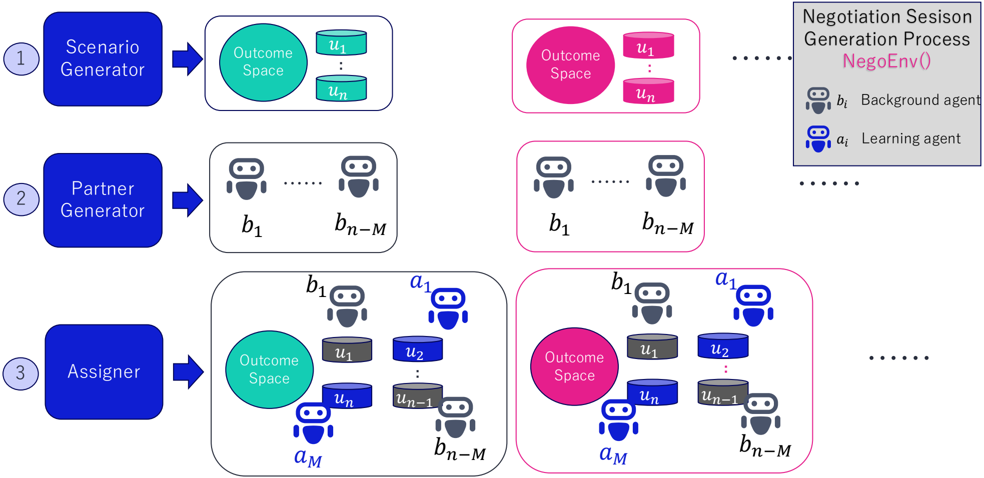
Generating Negotiation Scenarios for Training/Testing



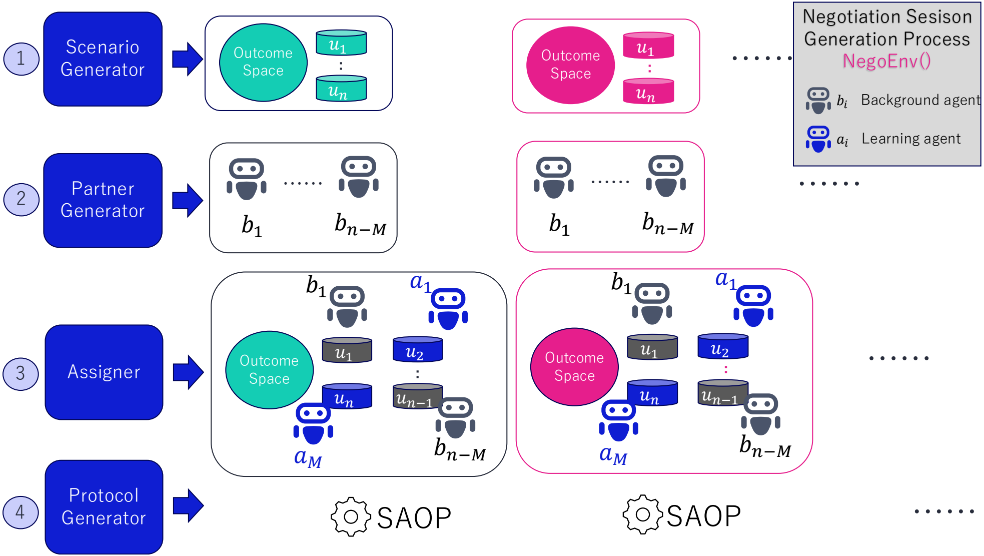
Generating Negotiation Scenarios for Training/Testing



Generating Negotiation Scenarios for Training/Testing



Generating Negotiation Scenarios for Training/Testing



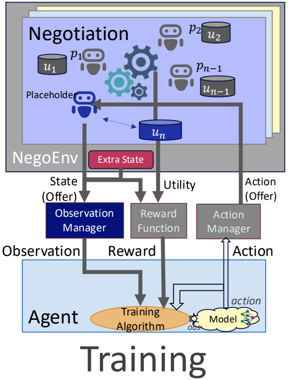
Observation Manager/Encoder

Responsibilities

- Decides what is being observed.
- Maps the observation to the model's **Space**.

Examples

Received	Last partner offer.
ReceivedU	Utility of last partner offer.
LastU	Utility of last two offers.
Window(K)	Last K partner offers.
WindowU(K)	Utilities of last K partner offers.



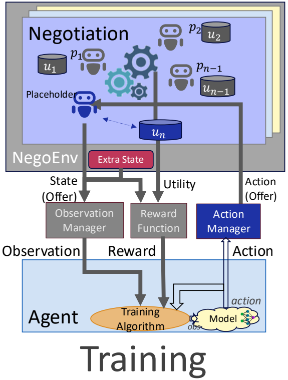
Action Manager/Decoder

Responsibilities

- Defines what the agent can control (e.g. acceptance decision, offer, etc).
- Maps **from** the model's **Space**.

Examples

Offer	Next offer.
Strategy	Next Strategy (switcher).
Utility	Utility of the next offer (an inverse ufun is needed).
Utility Range	A range of utilities to sample from.



Reward Function

Responsibilities

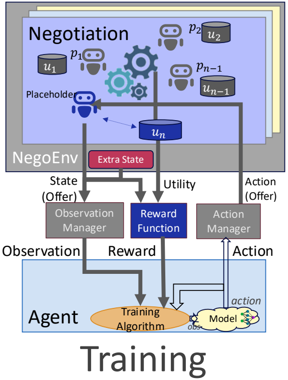
- Defines how the agent is being rewarded.

Examples

Utility Simplest option.

Time Penalize longer (shorter) negotiations.

Partner Utility Needs an opponent model.



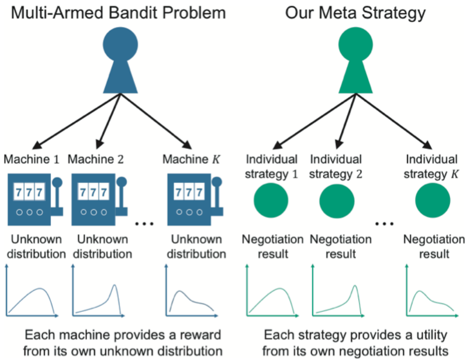
Example RL Negotiators

Multiaremed Bandits for Repeated Negotiations

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

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



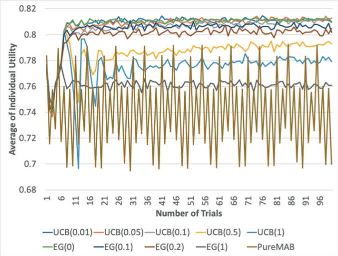
Ryohei Kawata and Katsuhide Fujita. "Meta-Strategy Based on Multi-Armed Bandit Approach for Multi-Time Negotiation". In: *IEICE TRANSACTIONS on Information and Systems* 103.12 (2020), pp. 2540–2548

Multiaremed Bandits for Repeated Negotiations

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

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



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

Ryohei 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

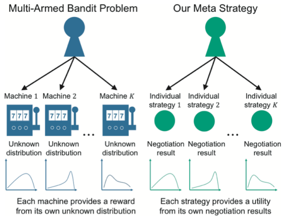
Multiarmed Bandits: Mapping

Components

ObservationManager N/A

RewardFunction utility

ActionManager strategy index



Supporting components

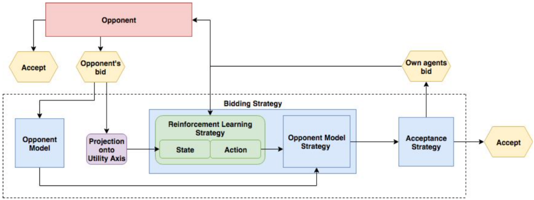
- Base strategies Atlas3, CaduceusDC16, Kawaii, ParsCat, Rubick, YXAgent

- Applies to **repeated** negotiations.

Training Method

- After every negotiation update the corresponding $\hat{\mu}_s$.
- Use the slot machine (negotiator) that maximizes $UCB(s) = \hat{\mu}_s + c\sqrt{\frac{lnN}{N_s}}$.

RLBOA: Learning Offering Strategy



Main Points

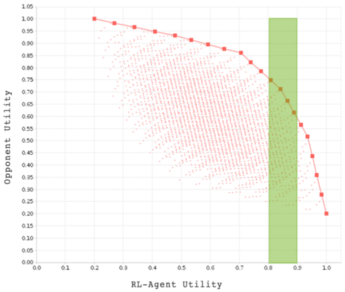
- Extends the BOA architecture.
- Learns only a bidding strategy:
  - The agent learns how to move *in its own utility axis*.

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: $\{\hat{u}(\omega^s_t), \hat{u}(\omega^s_{t-1}), \hat{u}(\omega^p_t), \hat{u}(\omega^p_{t-1}), t\}$.
  - $\hat{u}(\omega) = \lfloor N \times u(\omega) \rfloor$[10]
- Action Space: $\leftarrow, -, \rightarrow$.
  - First step $\rightarrow i \in [0, N-1]$
  - Out-of-boundary correction: $-$.
- Training Method: Q-learning
- Acceptance Strategy [Recommended]: $AC_{next}(\alpha = 1, \beta = 0)$[11]

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



RLBOA: Mapping

Components

ObservationManager Utility range (discrete).

RewardFunction utility

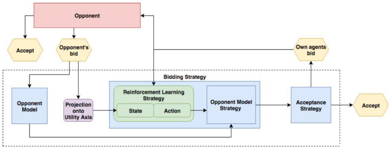
ActionManager Up to one utility band (3 valued).

Supporting components

- Utility Inverter Opponent model.
- Utility Inverter Samples an outcome in a range of utilities that maximizes the partner's utility.

Training Method

- Q-learning.



- Applies within a single negotiation.

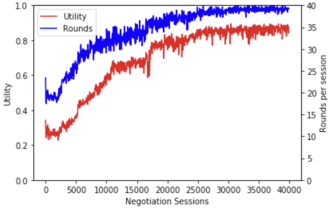
RLBOA: Observation/Action Manager Code

```python
@define
class RLBoaEncoder(DictEncoder):
    """The observation encoder of RLBOA according to the paper."""

    children: tuple[ObservationEncoder, ...] = field(init=False, factory=tuple)
    names: tuple[str, ...] = field(init=False, factory=tuple)

    def __attrs_post_init__(self):
        self.names = ("time", "utility")
        self.children = (
            DTimeEncoder(n_levels=5),
            DWindowedUtilityEncoder(
                n_offers=4,
                n_levels=10,
            ),
        )

@define
class RLBoaDecoder(DRelativeUtilityDecoder1D):
    """The action decoder for RLBOA"""
```

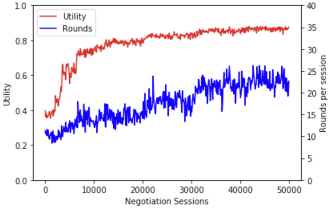
RLBOA: Negotiator Code

```python
class RLBoa(SAORLNegotiator):
    """RLBOA implementation"""

    @classmethod
    def default_trainer_type(cls) -> type["BaseAlgorithm"]:
        from stable_baselines3.ppo import PPO

        return PPO

    @classmethod
    def default_obs_encoder_type(cls) -> type[RLBoaEncoder]:
        return RLBoaEncoder

    @classmethod
    def default_action_decoder_type(cls) -> type[RLBoaDecoder]:
        return RLBoaDecoder
```

RLBOA: Evaluation and Results



(a) Scenario generality experiment against the Boulware agent.



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

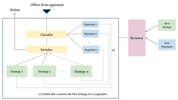
- Partners: TFT, Boulware TB
- Projection into one's utility space is surprisingly effective.
- Faster and better agreements!

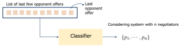
Domain	Outcome space	Low opp.	High opp.
Small	256	0.2615	0.5178
Medium	3.125	0.3111	0.5444
Large	46.656	0.2595	0.5250

A Framework for Learning Offer Strategies

Main Idea[12]

- 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.



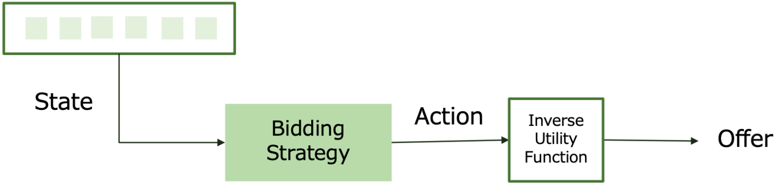


Phases

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

Ayan Sengupta, 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

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)

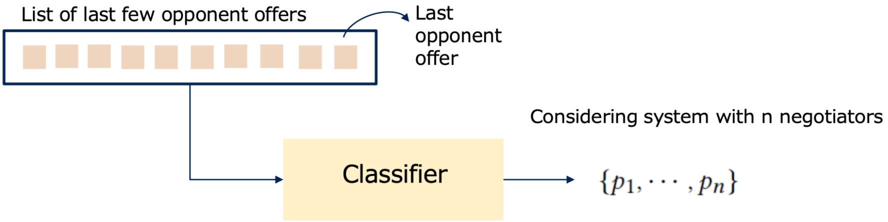
$$s_t = \{t_r, U_s(\omega_s^{t-2}), U_s(\omega_o^{t-2}), U_s(\omega_s^{t-1}), U_s(\omega_o^{t-1}), U_s(\omega_s^{t}), U_s(\omega_o^{t})\}$$

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

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

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

During: Learning realtime Partner Classification



The SL Components

Features Opponent last K offers.

Target Opponent Type (discrete set)

After: Reviewing New Pairs

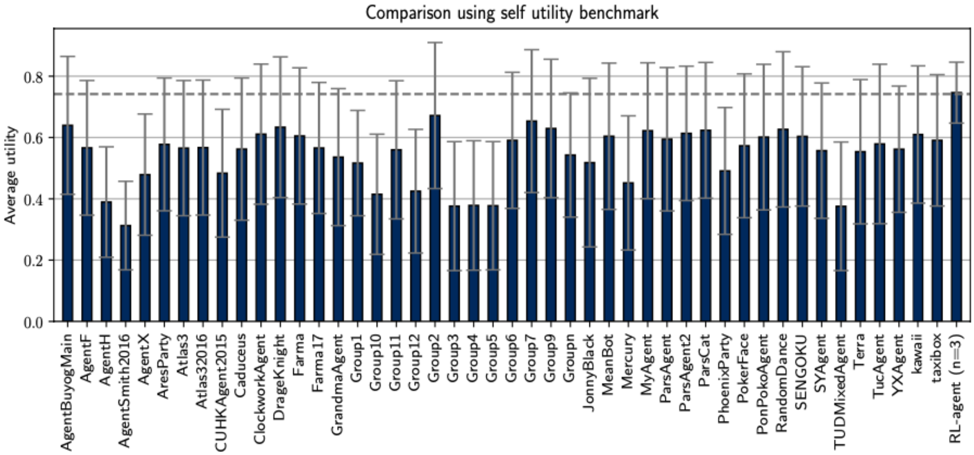
New Partner Type (N_{new}) Ecncountered

- 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 $\downarrow$.

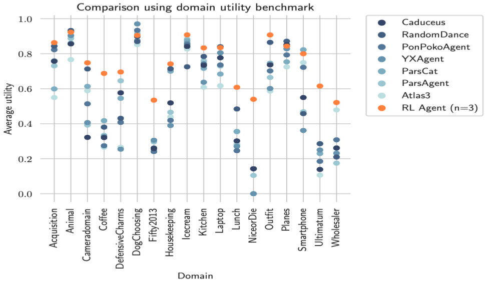
Update Best Responses

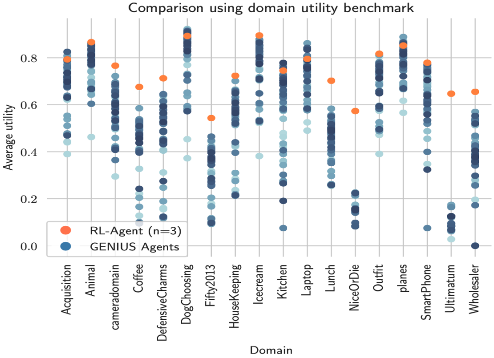
- For every learned ABR, negotiator pair (S, N):
  - Evaluate S_{new} against $N \rightarrow U(S_{new})$
  - Evaluate S against $N \rightarrow U(S)$
  - Replace S with S_{new} iff $\alpha U(S) < U(S_{new})$

Results: Against Different Opponents

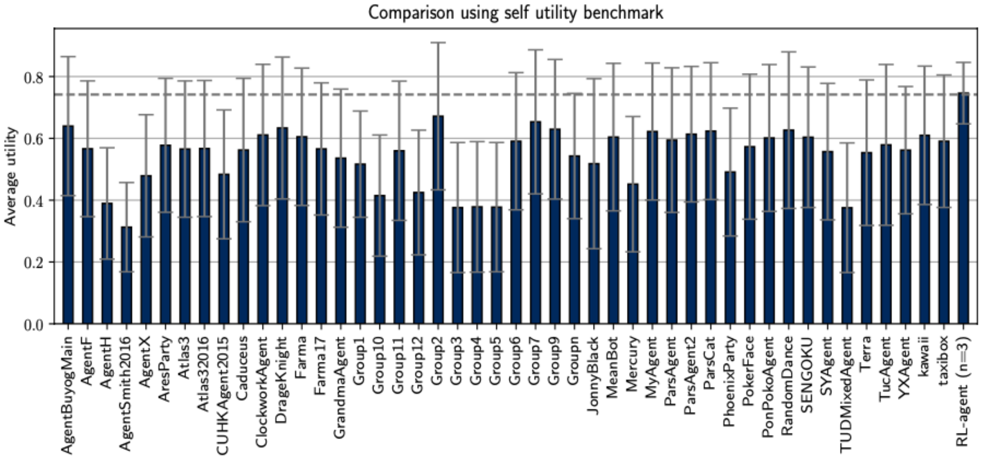


Results: In Different Domains



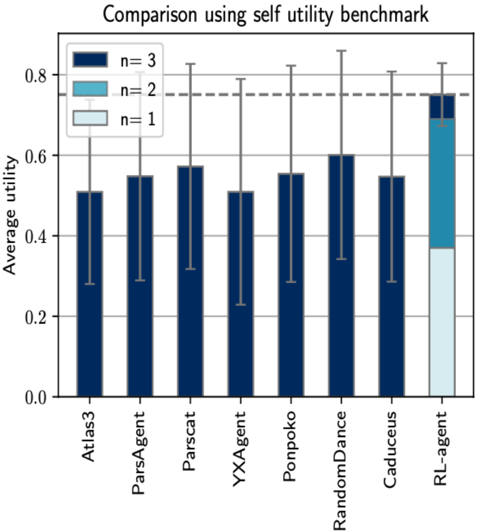


Results: Compared with SOTA Agents



Results: Improvement with new best responses

- Training against Boulware approaches but cannot exceed SOTA strategies.
- Adding Atlas3, outperforms SOTA strategies.
- Adding AgentK, further improves performance.
- Interestingly, the performance saturates at this point.



DQN for learning Acceptance Strategy

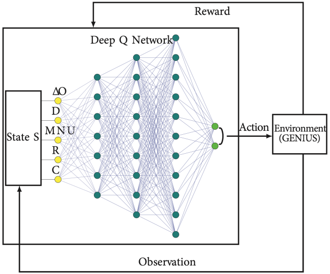
Main Idea

- 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

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



Yousef 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

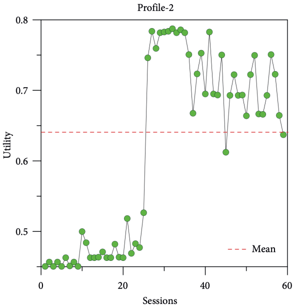
Evaluation

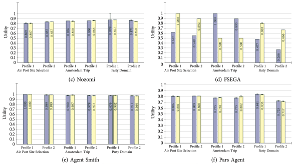
Training

- 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



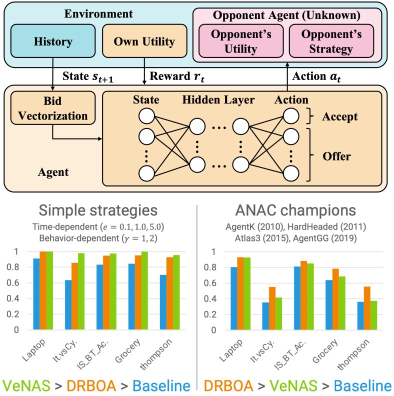


Learning Offer and Acceptance Policies

- Fixed domain (i/o using outcomes).
- Discrete Issues: One hot encoding per issue.

- State Space $\omega^s, \omega^o, t, \eta_t$
- Action Space $\Omega \wedge$ Accept
- Reward $= \begin{cases} u(\omega_a), & \text{At the end} \\ 0 & \text{non terminal state} \end{cases}$

VeNAS: Versatile Negotiating Agent Strategy

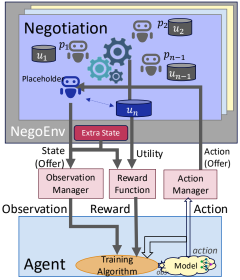


Toki Takahashi et al. "VeNAS: Versatile Negotiating Agent Strategy via Deep Reinforcement Learning". In: *AAAI 2022*. 2022

Wrap up

Wrap Up

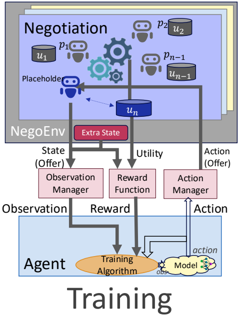
- Automated negotiation (AN) is a well-defined long-standing challenging problem in multi-agent coordination.
- AN is becoming more relevant to real world business applications due to the faster pace of automation.
- Reinforcement learning (and MARL) provide a potential effective method for strategy learning in AN.
- Automated negotiation is a challenging (and directly applicable) problem for RL.
- negmas-rl simplifies the process of developing negotiators using RL.



Training

Wrap Up

- Automated negotiation (AN) is a well-defined long-standing challenging problem in multi-agent coordination.
- AN is becoming more relevant to real world business applications due to the faster pace of automation.
- Reinforcement learning (and MARL) provide a potential effective method for strategy learning in AN.
- Automated negotiation is a challenging (and directly applicable) problem for RL.
- negmas-rl simplifies the process of developing negotiators using RL.



Thank you

y.mohammad@nec.com

References I

Aydoğan, Reyhan et al. "Alternating offers protocols for multilateral negotiation". In: *Modern Approaches to Agent-based Complex Automated Negotiation*. Springer, 2017, pp. 153–167.

Baarslag, Tim, Koen Hindriks, and Catholijn Jonker. "A tit for tat negotiation strategy for real-time bilateral negotiations". In: *Complex Automated Negotiations: Theories, Models, and Software Competitions*. Springer, 2013, pp. 229–233.

— ."Effective acceptance conditions in real-time automated negotiation". In: *Decision Support Systems* 60 (2014), pp. 68–77.

Baarslag, Tim et al. "Decoupling Negotiating Agents to Explore the Space of Negotiation Strategies". In: *Novel Insights in Agent-based Complex Automated Negotiation*. Ed. by Ivan Marsa-Maestre et al. Tokyo: Springer Japan, 2014, pp. 61–83. ISBN: 978-4-431-54758-7. DOI: 10.1007/978-4-431-54758-7_4. URL: https://doi.org/10.1007/978-4-431-54758-7_4.

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.

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.

Larsen, Bradley J. "The efficiency of real-world bargaining: Evidence from wholesale used-auto auctions". In: *The Review of Economic Studies* 88.2 (2021), pp. 851–882.

Lin, Raz et al. "Genius: An Integrated Environment for Supporting the Design of Generic Automated Negotiators". In: *Computational Intelligence* 30.1 (2014), pp. 48–70. ISSN: 1467-8640. DOI: 10.1111/j.1467-8640.2012.00463.x. URL: http://dx.doi.org/10.1111/j.1467-8640.2012.00463.x.

Mohammad, Yasser, Amy Greenwald, and Shinji Nakadai. "NegMAS: A platform for situated negotiations". In: *Twelfth International Workshop on Agent-based Complex Automated Negotiations (ACAN2019) in conjunction with IJCAI*. Macau, China, 2019. URL: https://github.com/yasserfarouk/negmas.

References II

- Myerson, Roger B and Mark A Satterthwaite. "Efficient mechanisms for bilateral trading". In: *Journal of economic theory* 29.2 (1983), pp. 265–281.
- Neumann, J. von and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1947.
- 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.
- Rubinstein, Ariel. "Perfect equilibrium in a bargaining model". In: *Econometrica: Journal of the Econometric Society* (1982), pp. 97–109.
- 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.

\Orchestrating a brighter world