

Automated Negotiation: Challenges and Tools

AAAI 2022 Tutorial

Yasser Mohammad ^{1, 2, 3} Amy Greenwald ⁴

¹ NEC Corporation, Global Innovation Unit

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

³Assiut University, Egypt

⁴Brown University, USA

February 23, 2022



BROWN
UNIVERSITY



NEC-AIST
AI Cooperative
Research Laboratory



Outline

- ① Why [Business]?
- ② Why [Academia]?
- ③ What?

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



The Automated Negotiation Challenge

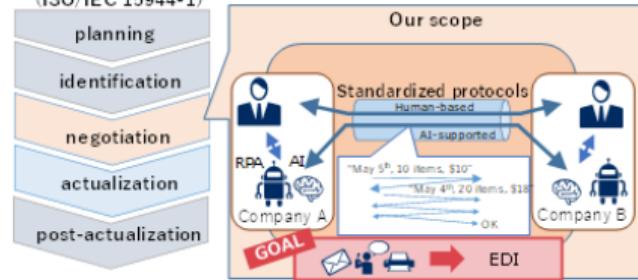
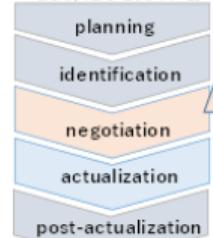
Why is it hard?

- 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-

Outline

- ① Introduction and Classic Results (45min) Break (5min)
- ② Protocols, Strategies and Platforms (50min)
 - ① Hands On Experience
Break (10min)
- ③ Learning in Negotiation (40min)
Break (10min)
- ④ Supply Chain Management Competition (20min)
 - ① Hands On Experience
Break (5min)
- ⑤ Challenges and Open Problems (35min)
- ⑥ Concluding Remarks (5min)

Materials

- ① Tutorial Website:

http://yasserm.com/aaai2022tutorial-automated_negotiation_challenges_and_tools/

- ② Github Repository: <https://github.com/yasserfarouk/Aaai2022AutomatedNegotiation>

- ③ Handouts:

<https://github.com/yasserfarouk/Aaai2022AutomatedNegotiation/raw/main/handouts.pdf>

- ④ Negmas Documentation: <http://www.yasserm.com/negmas>

- ⑤ SCML Documentation: <http://www.yasserm.com/scml/scmldocs>

- ⑥ SCML Competition: <https://scml.cs.brown.edu>

Automated Negotiation: Challenges and Tools

Introduction and Classic Results

Yasser Mohammad^{1, 2, 3} Amy Greenwald⁴

¹ NEC Corporation, Global Innovation Unit

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

³Assiut University, Egypt

⁴Brown University, USA

February 23, 2022



NEC-AIST
AI Cooperative
Research Laboratory



Outline

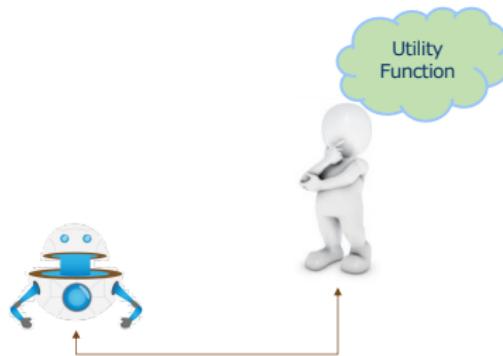
1 Negotiation

2 Bargaining 101

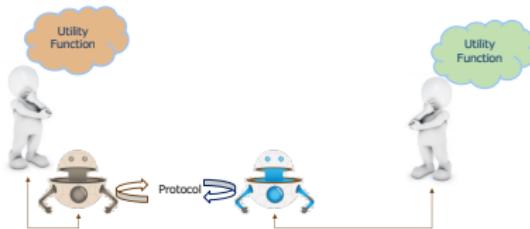
Outline

① Negotiation

② Bargaining 101



Components of Negotiation



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
- ...

Dimensions of Automated Negotiation

Negotiator Type

- ① Agent-agent
- ② Agent-human

Outcome Space Type

- ① Single Issue
- ② Multiple Issues

Number of Negotiators

- ① Bilateral negotiation
- ② Multilateral negotiation

Protocol Type

- ① Mediated
- ② Unmediated

Outline

1 Negotiation

2 Bargaining 101

- Abstract Problem Definition
- Bargaining Games and Solutions

Outline

1 Negotiation

2 Bargaining 101

- Abstract Problem Definition
- Bargaining Games and Solutions

A Joint Venture

- A factory F can manufacture a widget at cost C
- A distributor D can sell a widget for revenue R
- $R > C$, so there is surplus $R - C$ to be gained (value creation)
- Bargaining problem: how much should D pay F for the widget?
(value division)

Abstract Bargaining Problem

The two-person bargaining problem can be defined abstractly by

- A set $F \subset \mathbb{R}^2$ of feasible utilities (u_1, u_2)
- A disagreement point $\phi \doteq (d_1, d_2) \in \mathbb{R}^2$, also called the status quo.
The value d_i is called agent i 's reservation value.

Individual rationality assumption: No agent will ever agree to a utility below their disagreement point

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

von Neumann-Morgenstern Utility Theorem (1944)

It is natural to express agent's preferences as comparisons: e.g., "I like apples better than bananas."

Given an agent with partial preferences with at least one strict inequality that satisfy various axioms (completeness, transitivity, continuity, and IIA),

$$\exists u : \Omega \rightarrow \mathbb{R} \text{ such that } \omega \succ \psi \text{ iff } \mathbb{E}[u(\omega)] < \mathbb{E}[u(\psi)]$$

and u is unique up to scaling.

Why is this relevant?

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

Outline

1 Negotiation

2 Bargaining 101

- Abstract Problem Definition
- Bargaining Games and Solutions

Nash's Protocol (1950)

Arguably the simplest possible protocol.

- Both agents announce their demands simultaneously.
- If their demands are feasible, an agreement is reached.
- If not, the outcome is the disagreement point.

In the joint venture game (because there are now rules), we might define the agents' demands to be feasible iff the price announced by the distributor is at least the price announced by the manufacturer.

There are many possible equilibria in bargaining problems.

In the joint venture game, any price at which a trade transpires is an equilibrium.

Nash sought a theory of bargaining which would characterize a unique outcome of a negotiation.

Nash's Demand Game (1950)

The Nash demand game:

- Two agents are offered the chance to split a dollar.
- They both announce a number in $[0, 1]$ simultaneously.
- If the sum of the two numbers does not exceed 1, they each win what they demanded.
- Otherwise, they win nothing.

There are again many possible equilibria in the Nash demand game.

Define a player i 's surplus as the difference between the value of an agreement and its reservation value.

Nash's bargaining solution is the point at which the product of each agent's surplus is maximized:

$$\arg \max_{\omega} (u_1(\omega) - u_1(\phi))(u_2(\omega) - u_2(\phi))$$

This is the unique point that satisfies several natural axioms: efficiency, symmetry, invariance, and IIA.

Automated Negotiation: Challenges and Tools

Protocols and Strategies

Yasser Mohammad^{1, 2, 3} Amy Greenwald⁴

¹ NEC Corporation, Global Innovation Unit

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

³Assiut University, Egypt

⁴Brown University, USA

February 23, 2022



Outline

- 1 Platforms Used in this Tutorial
- 2 Outcome Space and Preferences
- 3 Negotiation Protocols
- 4 Anatomy of a SAOP Negotiation Agent
- 5 Basic Offer Policies
- 6 Basic Acceptance Policies
- 7 Basic Opponent Models

Outline

- 1 Platforms Used in this Tutorial
- 2 Outcome Space and Preferences
- 3 Negotiation Protocols
- 4 Anatomy of a SAOP Negotiation Agent
- 5 Basic Offer Policies
- 6 Basic Acceptance Policies
- 7 Basic Opponent Models
- 8 References

Some Automated Negotiation Platforms

Genius¹

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

NegMAS²

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



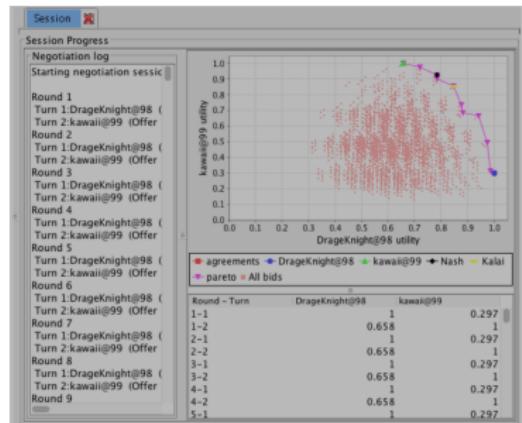
GENIUS

Genius³

- Originally Developed by University of Delft and University of Bar Ilan. Currently maintained by a consortium of researchers.
- Since 2008 and used in all ANAC competitions
- The defacto-standard.
- Java-based ⁴.
- Has a GUI for running negotiations and tournaments.
- Includes SOTA agents from ANAC competitions since 2010.

GENIUS

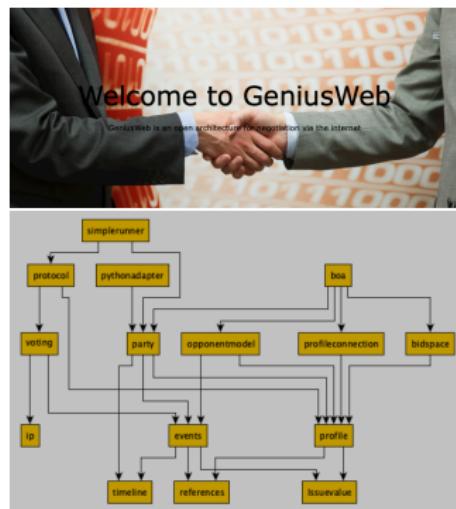
>>> General Environment for Negotiation with Intelligent multi-purpose Usage Simulation.



GeniusWeb

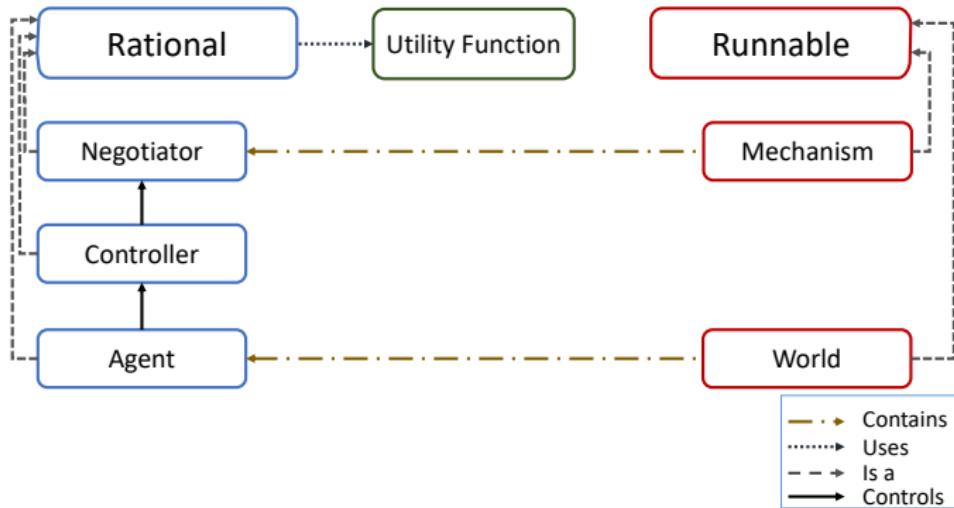
GeniusWeb 5

- An open architecture for negotiation via the internet.
 - Implemented in Java with Python support for developing agents.
 - Used since ANAC 2019.
 - Negotiators can be distributed over multiple machines.



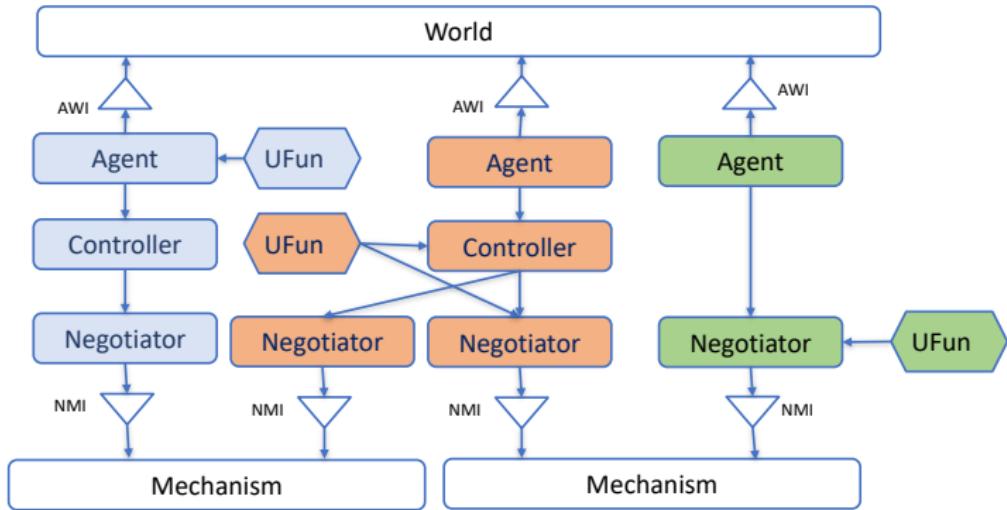
⁵GeniusWeb Team. *GeniusWeb Website*. 2021. URL: <https://tracinsy.ewi.tudelft.nl/pubtrac/GeniusWeb>.

NegMAS⁶ in one slides



⁶<https://www.github.com/yasserfarouk/negmas>

NegMAS⁷ in almost one slide



⁷<https://www.github.com/yasserfarouk/negmas>

Outline

1 Platforms Used in this Tutorial

2 Outcome Space and Preferences

- Outcome Spaces

3 Negotiation Protocols

4 Anatomy of a SAOP Negotiation Agent

5 Basic Offer Policies

6 Basic Acceptance Policies

7 Basic Opponent Models

8 References

Outline

1 Platforms Used in this Tutorial

2 Outcome Space and Preferences

- Outcome Spaces

3 Negotiation Protocols

4 Anatomy of a SAOP Negotiation Agent

5 Basic Offer Policies

6 Basic Acceptance Policies

7 Basic Opponent Models

8 References

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

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

Ordinal with defined order

Cardinal with defined difference

Numeric with defined numeric value
(integer/real)

NegMAS

```
make_os([
    make_issue(["to be", "not to be"], "the question"),
    make_issue(10),
    make_issue((0.0, 1.0)),
    make_issue([('happy', "kitten"), ("sad", "dog")])
])
```

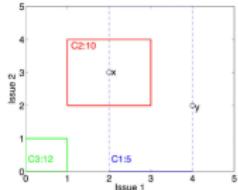
Preferences and Utility Functions

- 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

$$u(\omega) = \sum_{k=0}^K f_k(\omega) \times \delta[\omega \in C_k]$$



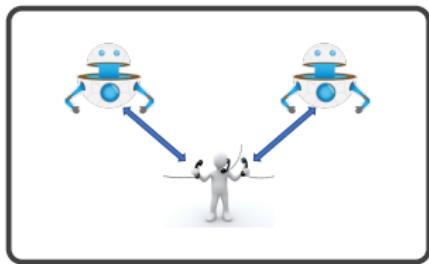
Outline

- 1 Platforms Used in this Tutorial
- 2 Outcome Space and Preferences
- 3 Negotiation Protocols
- 4 Anatomy of a SAOP Negotiation Agent
- 5 Basic Offer Policies
- 6 Basic Acceptance Policies
- 7 Basic Opponent Models
- 8 References

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.



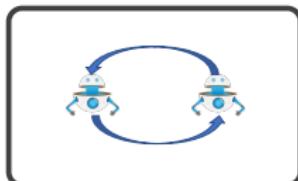
Examples

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^a.

Alternating Offers Protocol **Finite** horizon, **multi-issue**, **multilateral** protocol with partial information^b.

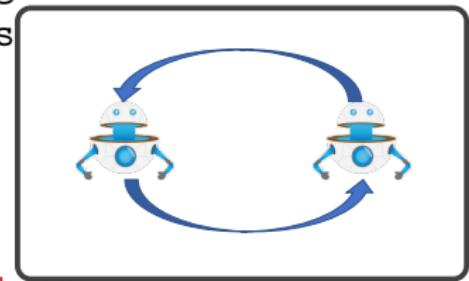
^aAriel Rubinstein, "Perfect equilibrium in a bargaining model". In: *Econometrica: Journal of the Econometric Society*

Stacked Alternating Offers Protocol

```

n_agreed, current = 0, randint(0, n_agents)
offer = agents[current].offer()
while not timeout():
    current = (current + 1) % n_agents
    response = agents[current].res
    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"

```

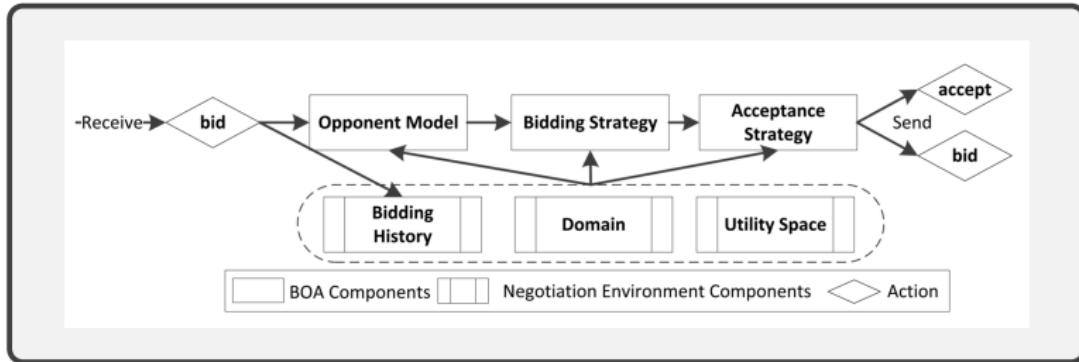




Outline

- 1 Platforms Used in this Tutorial
- 2 Outcome Space and Preferences
- 3 Negotiation Protocols
- 4 Anatomy of a SAOP Negotiation Agent
- 5 Basic Offer Policies
- 6 Basic Acceptance Policies
- 7 Basic Opponent Models
- 8 References

Negotiator Components⁸



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**.

⁸ 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.



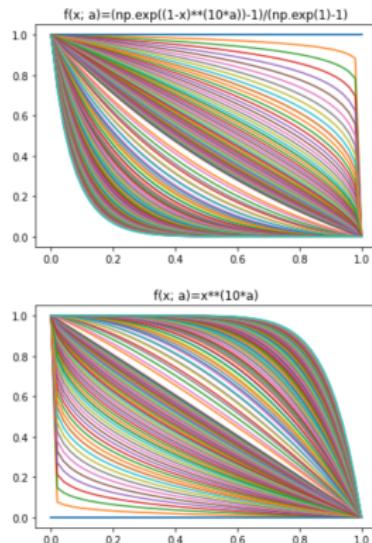
Outline

- 1 Platforms Used in this Tutorial
- 2 Outcome Space and Preferences
- 3 Negotiation Protocols
- 4 Anatomy of a SAOP Negotiation Agent
- 5 Basic Offer Policies
- 6 Basic Acceptance Policies
- 7 Basic Opponent Models
- 8 References

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.



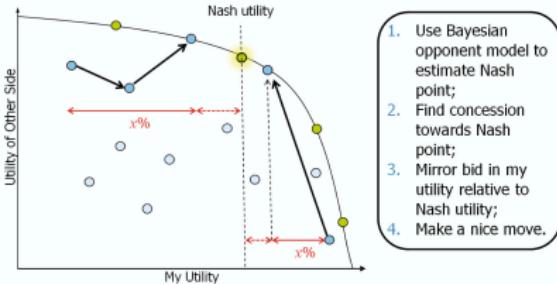
Behavioral Offer Policies

Behavior Based Strategies

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

(Nice) Tit-for-Tat (bilateral)⁹

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



⁹ 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.



Outline

- 1 Platforms Used in this Tutorial
- 2 Outcome Space and Preferences
- 3 Negotiation Protocols
- 4 Anatomy of a SAOP Negotiation Agent
- 5 Basic Offer Policies
- 6 Basic Acceptance Policies
- 7 Basic Opponent Models
- 8 References

Acceptance Policy

Examples

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,

Combining Acceptance Policies

Combined Acceptance Strategy¹⁰

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

NegMAS

```

ACCombi = ACNext() or (ACtauime(tau) and ACCConst(gamma))
ACBest = ACNext() or (ACtauime(tau) and ACLastKReceived(K))
ACAvg = ACNext() or (ACtauime(tau) and ACLastKReceived(K, op=)
ACBestAll = ACNext() or (ACtauime(tau) and ACLastKReceived())

```

¹⁰Tim Baarslag, Koen Hindriks, and Catholijn Jonker. "Effective acceptance conditions in real-time automated negotiation" 20 / 24

Outline

1 Platforms Used in this Tutorial

2 Outcome Space and Preferences

3 Negotiation Protocols

4 Anatomy of a SAOP Negotiation Agent

5 Basic Offer Policies

6 Basic Acceptance Policies

7 Basic Opponent Models

8 References

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.

Outline

- 1 Platforms Used in this Tutorial
- 2 Outcome Space and Preferences
- 3 Negotiation Protocols
- 4 Anatomy of a SAOP Negotiation Agent
- 5 Basic Offer Policies
- 6 Basic Acceptance Policies
- 7 Basic Opponent Models
- 8 References

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.

References II

- 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>.
- Rubinstein, Ariel. "Perfect equilibrium in a bargaining model". In: *Econometrica: Journal of the Econometric Society* (1982), pp. 97–109.
- Team, Genius. *Genius Website*. 2021. URL: <http://ii.tudelft.nl/genius/>.

References III

Team, GeniusWeb. *GeniusWeb Website*. 2021. URL:
<https://tracinsy.ewi.tudelft.nl/pubtrac/GeniusWeb>.

Automated Negotiation: Challenges and Tools

Learning in Negotiation

Yasser Mohammad^{1, 2, 3} Amy Greenwald⁴

¹ NEC Corporation, Global Innovation Unit

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

³Assiut University, Egypt

⁴Brown University, USA

February 23, 2022



BROWN
UNIVERSITY



NEC-AIST
AI Cooperative
Research Laboratory



Outline

- 1 Learning in Automated Negotiation
- 2 Learning how to negotiate
- 3 Learning Preferences
- 4 References

Outline

① Learning in Automated Negotiation

② Learning how to negotiate

③ Learning Preferences

④ References

Learning in Automated Negotiation

What?

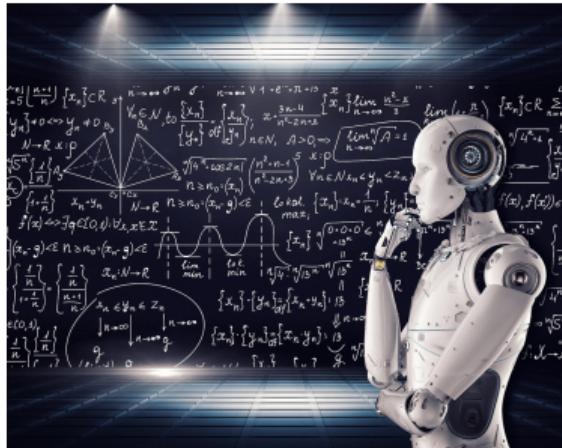
- ① Acceptance Policy
- ② Offering Policy
- ③ Opponent/Partner Model

When?

- ① Within Negotiation
- ② Between Negotiations

How?

- ① Supervised Learning
- ② Reinforcement Learning
- ③ Unsupervised Learning



"Artificial Intelligence & AI & Machine Learning" by mikemacmarketing

Outline

1 Learning in Automated Negotiation

2 Learning how to negotiate

- Offer Policy
- Acceptance Strategy
- Both

3 Learning Preferences

4 References

Outline

1 Learning in Automated Negotiation

2 Learning how to negotiate

• Offer Policy

- Multiarmed Bandits
- RLBOA
- Adaptive Automated Negotiating Agent Framework (A^3F)

• Acceptance Strategy

• Both

3 Learning Preferences

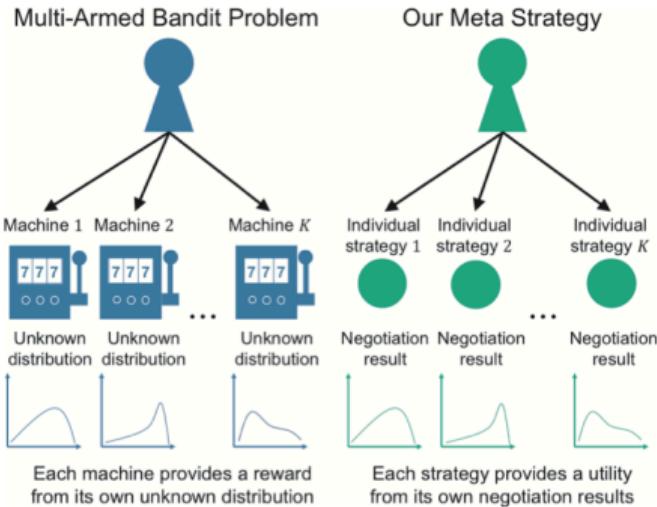
4 References

Multiaremed Bandits for Repeated Negotiations¹

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

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



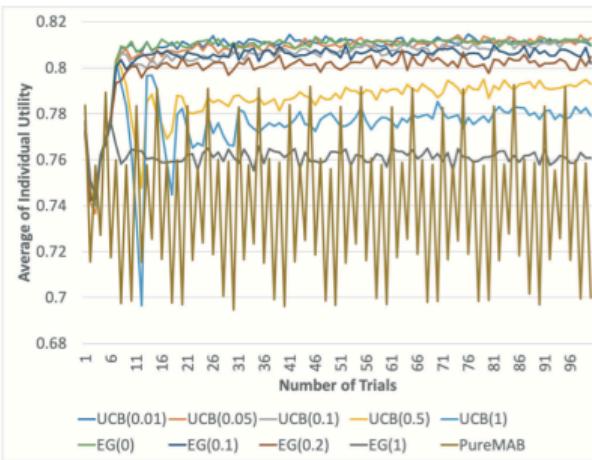
¹Ryohei Kawata and Katsuhide Fujita. "Meta-Strategy Based on Multi-Armed Bandit Approach for Multi-Time Negotiation". In: *IEICE TRANSACTIONS on Information and Systems* 103.12 (2020), pp. 2540–2548.

Multiarmed Bandits for Repeated Negotiations¹

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

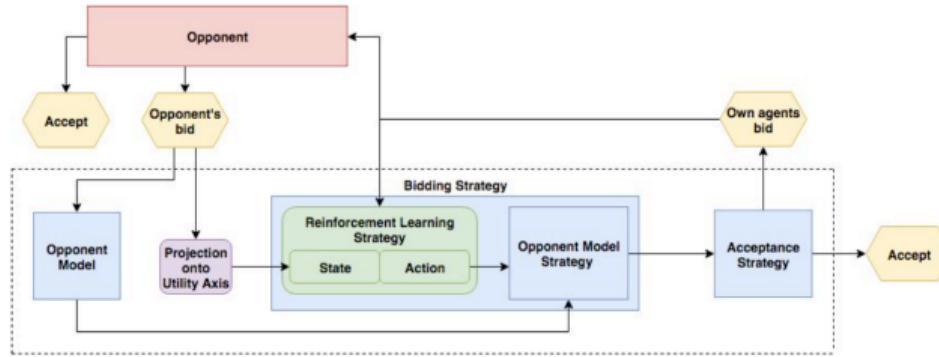
$$UCB(s) = \hat{\mu}_s + c \sqrt{\frac{\ln N}{N_s}}$$



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

¹Kawata and Fujita, "Meta-Strategy Based on Multi-Armed Bandit Approach for Multi-Time Negotiation".

RLBOA: Learning Offering Strategy²



Main Points

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

²Jasper 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.

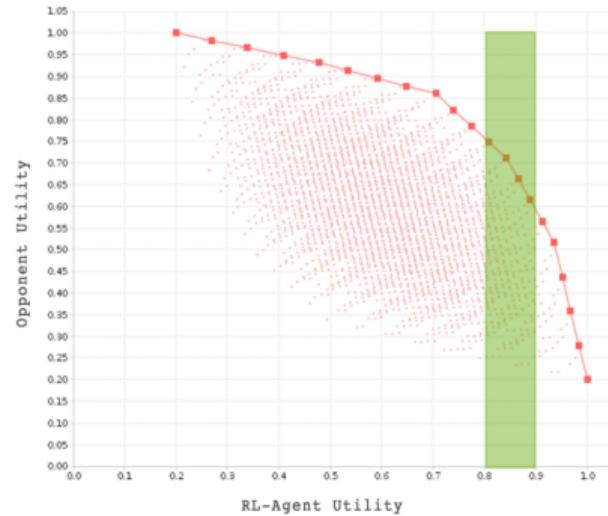
RLBOA: The details

- **State Space:** $\{\hat{u}(\omega_t^s), \hat{u}(\omega_{t-1}^s), \hat{u}(\omega_t^p), \hat{u}(\omega_{t-1}^p), t\}$.
- $\hat{u}(\omega) = [N \times u(\omega)]^3$
- **Action Space:** $\leftarrow, -, \rightarrow$.
- First step $\rightarrow i \in [0, N - 1]$
- Out-of-boundary correction: $-$.
- **Training Method:** Q-learning
- **Acceptance Strategy [Recommended]:**
 $AC_{next}(\alpha = 1, \beta = 0)$ ⁴

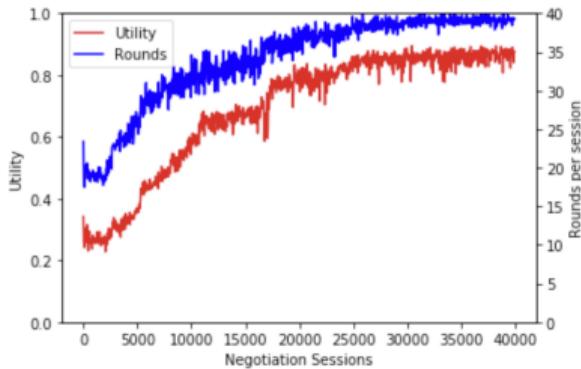
$$a(\omega) = \begin{cases} \text{Accept}, & \text{if } \alpha u(\omega) + \beta \geq u(o(s)) \\ \text{Reject}, & \text{otherwise} \end{cases}$$

⁴Just ignore the special case at $u == 1$.

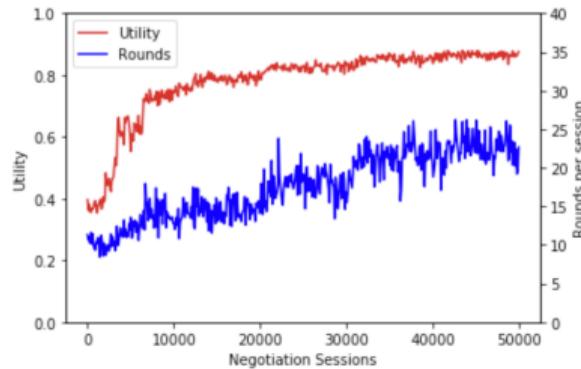
⁴Default acceptance strategy in NegMAS



RLBOA: Evaluation and Results



(a) Scenario generality experiment against the Boulware agent.



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

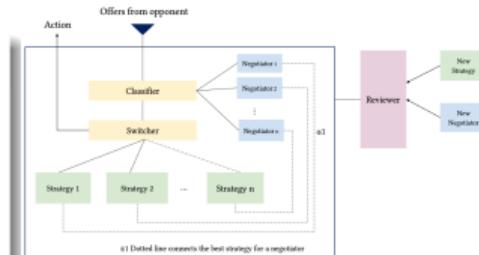
- Partners: TFT, Boulware TB
- Projection into one's utility space is surprisingly effective.
- Faster and better agreements!

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

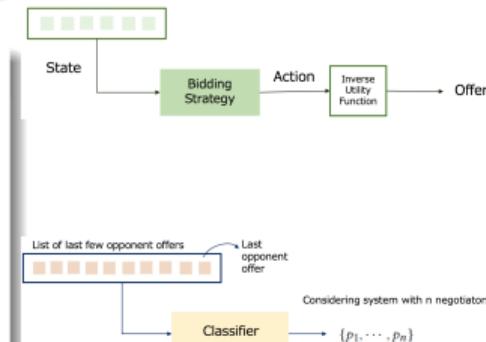
Main Idea⁵

- 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**.



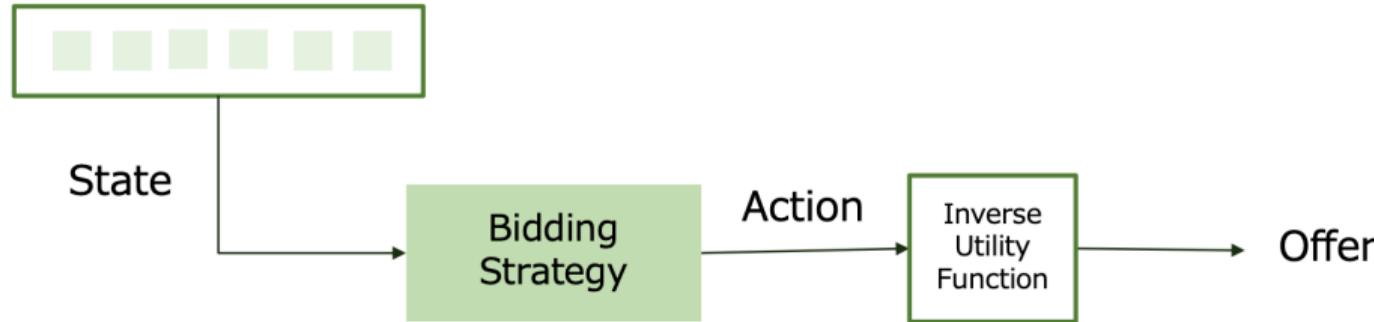
Phases

- **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**.



⁵Ayan Sengupta, Yasser Mohammad, and Shinji Nakada. "An Autonomous Negotiating Agent Framework with Reinforcement Learning Based Strategies and Adaptive Strategy Switching".

Before: Learning Approximate Best Response



State

Bidding
Strategy

Action

Inverse
Utility
Function

Offer

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)

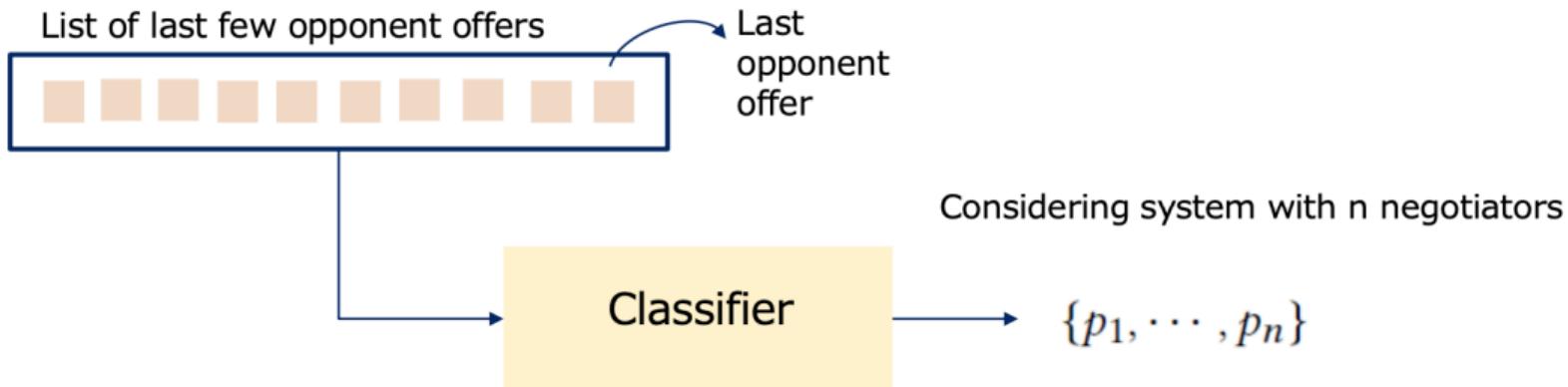
$$s_t = \{t_r, U_s(\omega_s^{t-2}), U_s(\omega_o^{t-2}), U_s(\omega_s^{t-1}), U_s(\omega_o^{t-1}), U_s(\omega_s^t), U_s(\omega_o^t)\}$$

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

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

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

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

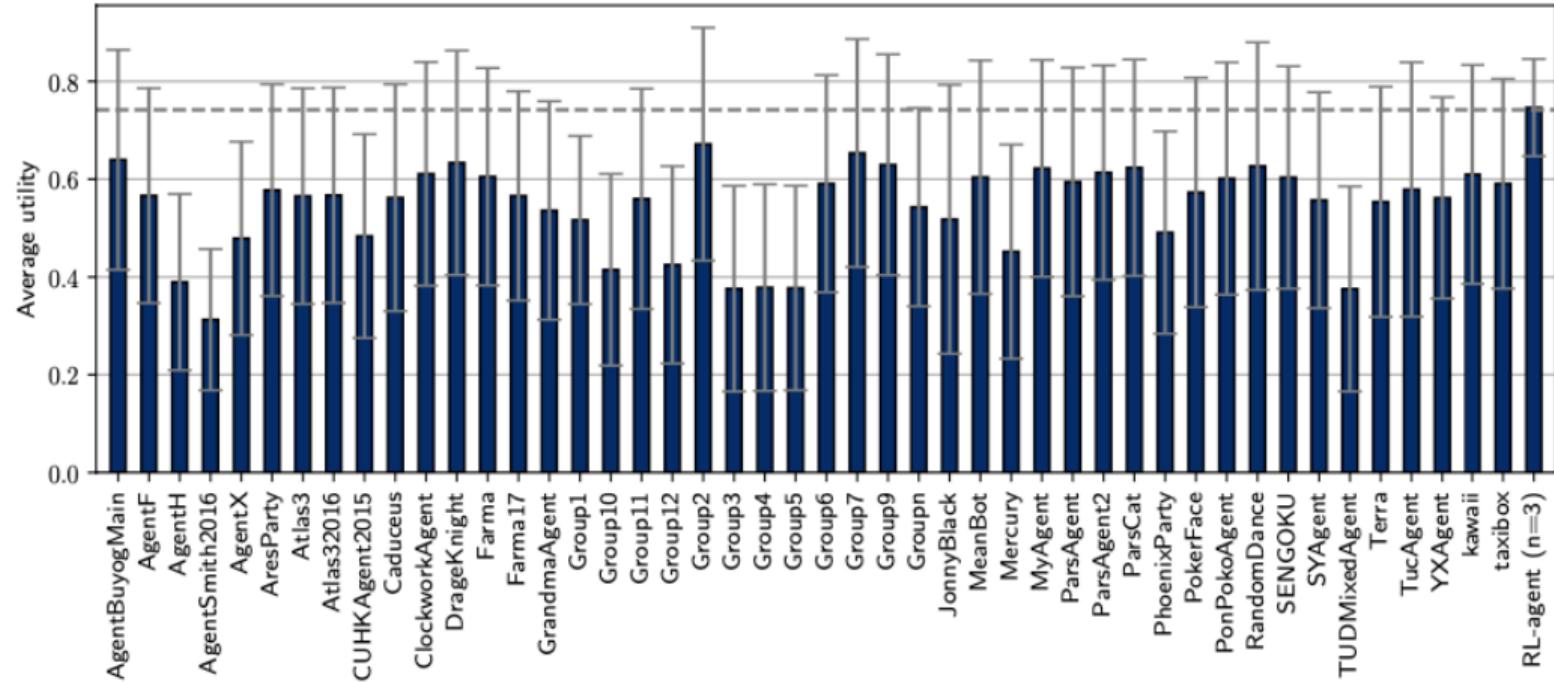
- Train a best response (using SAC) $\rightarrow S_{new}$.
- Evaluate S_{new} against $N_{new} \rightarrow U(S_{new})$
- Evaluate *Current* against $N_{new} \rightarrow U(Current)$
- Add (S_{new}, N_{new}) iff $\beta U(Current) < U(S_{new})$
- Update best responses ↓.

Update Best Responses

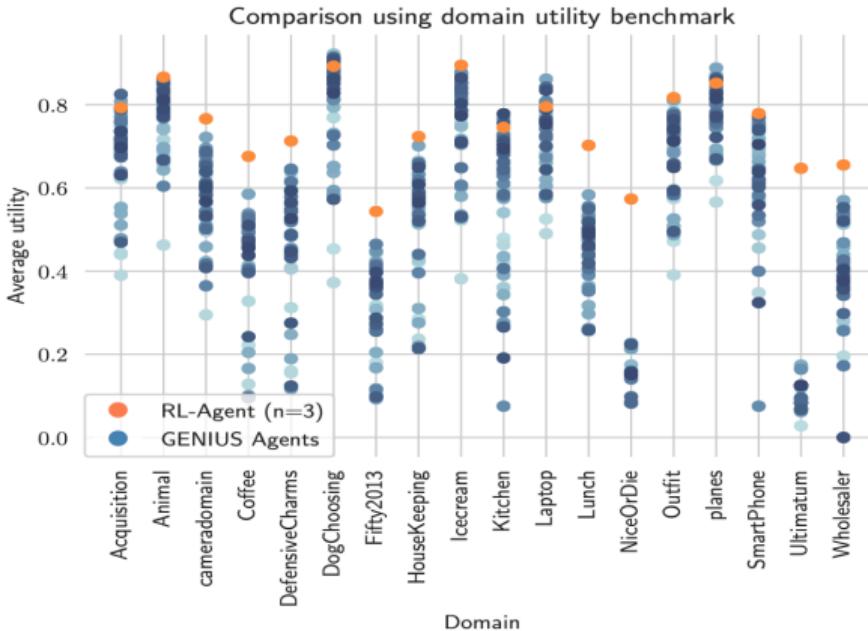
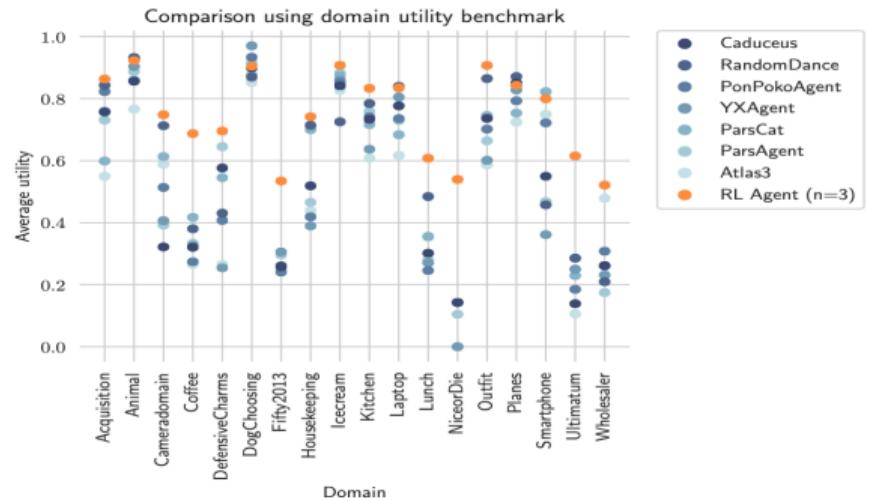
- For every learned ABR, negotiator pair (S, N) :
 - Evaluate S_{new} against $N \rightarrow U(S_{new})$
 - Evaluate S against $N \rightarrow U(S)$
 - Replace S with S_{new} iff $\alpha U(S) < U(S_{new})$

Results: Against Different Opponents

Comparison using self utility benchmark

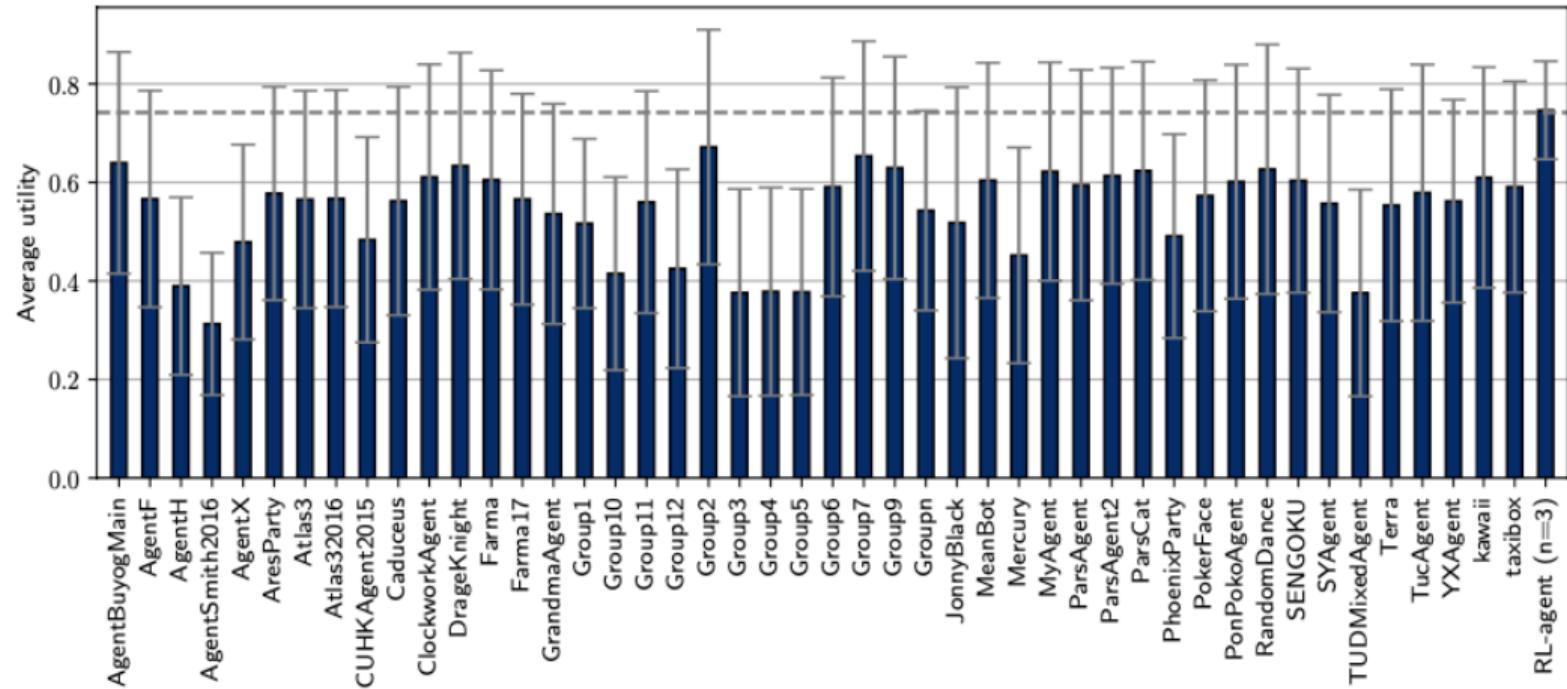


Results: In Different Domains

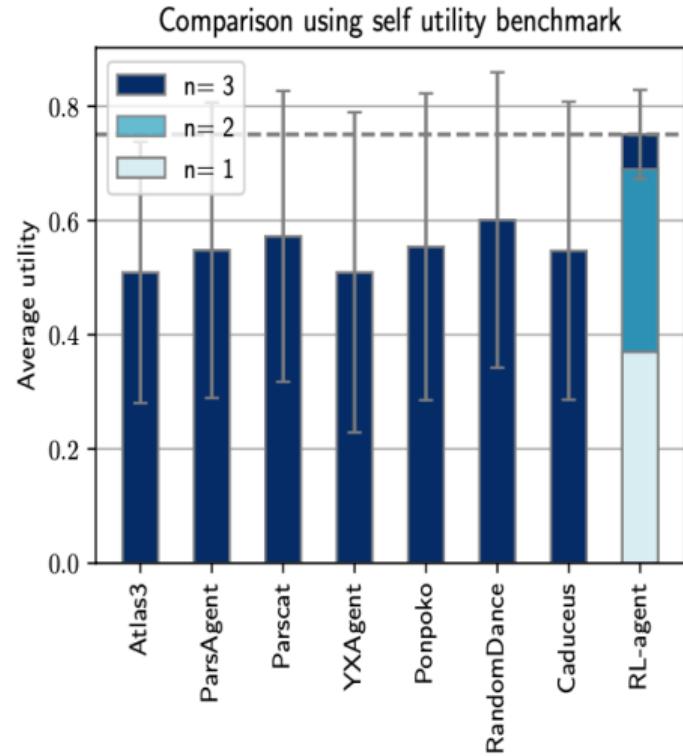


Results: Compared with SOTA Agents

Comparison using self utility benchmark



Results: Improvement with new best responses



Outline

1 Learning in Automated Negotiation

2 Learning how to negotiate

- Offer Policy
- **Acceptance Strategy**
- Both

3 Learning Preferences

4 References

DQN for learning Acceptance Strategy⁶

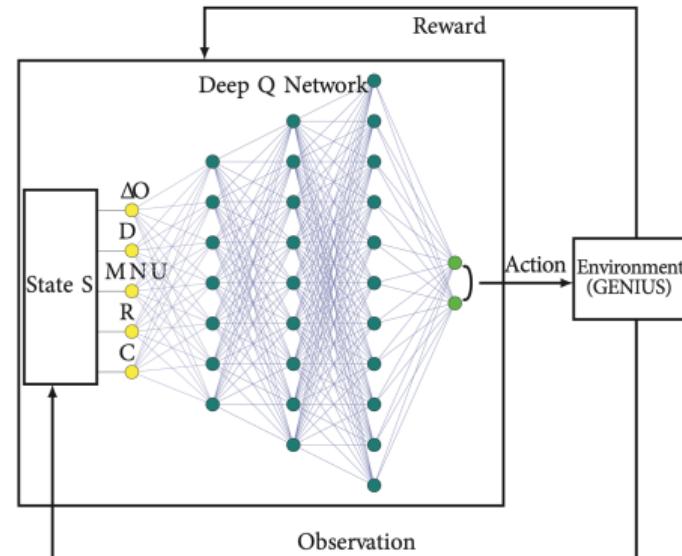
Main Idea

- 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**

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



⁶Yousef 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

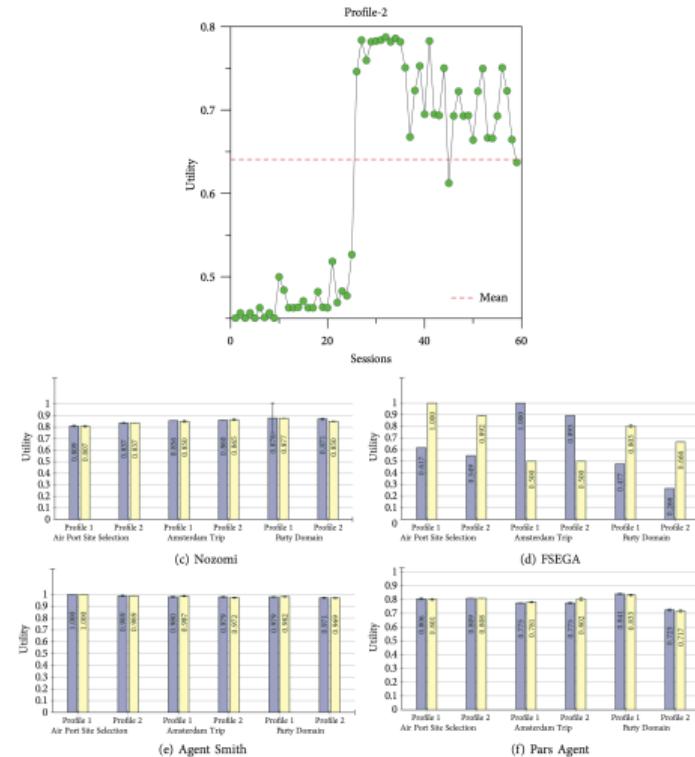
Evaluation

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



Outline

1 Learning in Automated Negotiation

2 Learning how to negotiate

- Offer Policy
- Acceptance Strategy
- Both

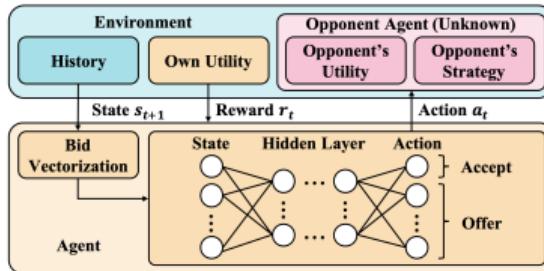
3 Learning Preferences

4 References

Learning Offer and Acceptance Policies together⁷

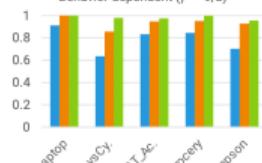
- Fixed domain (i/o using outcomes).
- Discrete Issues: One hot encoding per issue.
- State Space $\omega^s, \omega^o, t, \eta_t$
- Action Space $\Omega \wedge \text{Accept}$
- Reward = $\begin{cases} u(\omega_a), & \text{At the end} \\ 0 & \text{non terminal state} \end{cases}$
- Feb 25, 9:00-10:45am (PST) Blue 5
- Feb 26, 4:45-06:30am (PST) Blue 5

VeNAS: Versatile Negotiating Agent Strategy



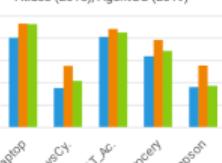
Simple strategies

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

⁷ Toki Takahashi et al. "VeNAS: Versatile Negotiating Agent Strategy via Deep Reinforcement Learning". In: AAAI 2022. 2022.

Outline

- 1 Learning in Automated Negotiation
- 2 Learning how to negotiate
- 3 Learning Preferences
 - Partner Preferences: Opponent Modeling
 - Own Preferences: Elicitation
- 4 References

Outline

- 1 Learning in Automated Negotiation
- 2 Learning how to negotiate
- 3 Learning Preferences
 - Partner Preferences: Opponent Modeling
 - Frequentist
 - Bayesian
 - Own Preferences: Elicitation
- 4 References

HardHeaded Opponent Modeling Strategy⁸

Context

- Winner of the ANAC 2011 competition.

$$u(\omega) = \sum_{i=1}^n \alpha_i F[i, \omega_i]$$

- Assumes a Discrete Outcome Space, Linear Additive Utility Function and a bilateral negotiation.
- Learns while negotiating.

Main Idea

- The opponent is likely to change values for issues that are less important.

⁸Thijs van Krimpen, Daphne Looije, and Siamak Hajizadeh. "HardHeaded". In: *Complex Automated Negotiations: Theories, Models and Software Competitions*. Ed. by Ito Takayuki et al. Springer, 2013, pp. 223–227.

HardHeaded Opponent Model: Pseudo-code

```
# M issues and N values per issue
F, alpha = np.zeros((M, N)), np.zeros(M)
epsilon, last_offer = 0.02, None
def after_receiving(self, state, offer): # update model
    if not self.last_offer:
        self.last_offer = offer
        return
    for i in range(M):
        if offer[i] != self.last_offer[i]:
            continue
        self.F[i, offer[i]] += 1
        self.alpha[i] += self.epsilon
    self.alpha /= self.alpha.sum()

def eval(self, offer): # estimate partner's utility value
    return self.alpha * self.F[:, offer]
```

Opponent Model: Bayesian



Bayesian Learning

Hypothesis A hypothesis about the opponent's behavior.

Opponent Model: Bayesian



Bayesian Learning

Hypothesis A hypothesis about the opponent's behavior.

Evidence Behavior of the agent (e.g. its counteroffers/rejections).

Opponent Model: Bayesian

Bayesian Learning

Hypothesis A hypothesis about the opponent's behavior.

Evidence Behavior of the agent (e.g. its counteroffers/rejections).

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

Opponent Model: Bayesian



Bayesian Learning

Hypothesis A hypothesis about the opponent's behavior.

Evidence Behavior of the agent (e.g. its counteroffers/rejections).

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

Example

Hypothesis space: Utility function as a weighted sum of basis functions

$$u(\omega) = \sum_{i=1}^n \alpha_i f_i(\omega_i; \sigma_i)$$

Opponent Model: Bayesian



Bayesian Learning

Hypothesis A hypothesis about the opponent's behavior.

Evidence Behavior of the agent (e.g. its counteroffers/rejections).

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

Example

Hypothesis space: Utility function as a weighted sum of basis functions

$$u(\omega) = \sum_{i=1}^n \alpha_i f_i(\omega_i; \sigma_i)$$

Evidence: Rejection and offers (assuming a strategy).

Bayesian Opponent Model Learner⁹

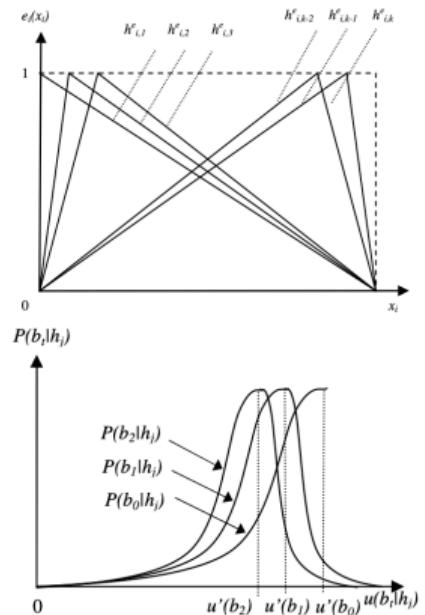
Assumptions

- Opponent has a Linear Additive UFun ($u(\omega) = \sum_{i=1}^{|\omega|} \alpha_i f_i(\omega_i, \sigma_i)$)
- Value functions (f_i) are triangle like (or linear).

Settings

- Hypothesis Space: values of α_i and σ_i
- Evidence: $P(\omega|\alpha_i, \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{(u(\omega|\alpha_i, \sigma_i) - \hat{u}(\omega))^2}{\sigma^2}$ with $\hat{u}(\omega) = 1 - \frac{t}{20}$.
- Estimated opponent utility value:

$$u^o(\omega) = \sum_{j=1}^{|H|} P(\alpha_j, \sigma_j | \Omega^o) u(\omega | \alpha_j, \sigma_j)$$



⁹Koen Hindriks and Dmytro Tykhonov. "Opponent modelling in automated multi-issue negotiation using bayesian learning". In: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*. 2008, pp. 331–338.

Outline

- 1 Learning in Automated Negotiation
- 2 Learning how to negotiate
- 3 Learning Preferences
 - Partner Preferences: Opponent Modeling
 - Own Preferences: Elicitation
 - Procedure and Strategies
 - Value of Information Algorithm
- 4 References

Preference Elicitation

The challenge

How to reduce Uncertainty in user preferences:

- before negotiation (offline preference elicitation).
- while negotiating (online preference elicitation).

Preference Elicitation

The challenge

How to reduce Uncertainty in user preferences:

- before negotiation (offline preference elicitation).
- while negotiating (online preference elicitation).

Types of questions

Utility Value what is $\tilde{u}(\omega)$?

Utility Constraint Is $\tilde{u}(\omega) \geq x$? Usually implemented as a standard gamble.

Utility Comparison Is $\omega_1 \succ \omega_2$?

Elicitation Procedures

- ① Long history in the decision support and economics research community.
- ② Take away message: .

Elicitation Procedures

- ① Long history in the decision support and economics research community.
- ② Take away message: **Do not ask about the utility directly..**

Elicitation Procedures

- ① Long history in the decision support and economics research community.
- ② Take away message: **Do not ask about the utility directly..**
- ③ Practical elicitation uses a **series** of comparisons between outcomes to assess utilities.

Elicitation Procedures

- ① Long history in the decision support and economics research community.
- ② Take away message: **Do not ask about the utility directly..**
- ③ Practical elicitation uses a **series** of comparisons between outcomes to assess utilities.

A Gamble

(ω^*, ω_*, p) : Getting ω^* with probability p otherwise ω_*

Elicitation Procedures

- ① Long history in the decision support and economics research community.
- ② Take away message: **Do not ask about the utility directly..**
- ③ Practical elicitation uses a **series** of comparisons between outcomes to assess utilities.

A Gamble

(ω^*, ω_*, p) : Getting ω^* with probability p otherwise ω_*

Example query

Do you prefer to get ω for certain over (ω^*, ω_*, p) ?

Elicitation Procedures/Strategies



Probability Equivalence

find p so that $\omega = (\omega^*, \omega_*, p)$

Certainty Equivalence

find ω so that $\omega = (\omega^*, \omega_*, p)$

- Both require *normalized* utilities.
- Both require knowledge of $\omega^* \succ \omega \succ \omega_*$.
- Lead to different biases.

Comparison-only Procedures

① Titration-down: $p_k = 1 - s \times k$

② Titration-up: $p_k = s \times k$

③ Ping-pong: $p_k = \begin{cases} s \times \lfloor k/2 \rfloor & k \text{ is odd} \\ 1 - s \times k/2 & k \text{ is even} \end{cases}$

Importance of Elicitation

Negotiation with Elicitation

$$m, \Omega, R, \tilde{U}_i \forall 1 \leq i \leq m, \hat{U}_i^0 \forall 1 \leq i \leq m$$

m Number of agents/actors

$\Omega = \{\omega_j\}$ Possible outcomes (assumed countable)

n Number of outcomes $|\Omega|$

$R(i) \equiv r_i$ Reserved value for agent *i*

$\tilde{U}_i : \Omega \rightarrow [0, 1]$ Utility of outcomes to **actor** *i*

$\hat{U}_i^0 : \Omega \rightarrow P$ Probability distribution of utility values for **agent** *i*

$$\hat{U}_{ij}^0 \equiv \hat{U}_i^0(\omega_j)$$

$P : \{[0, 1] \rightarrow [0, 1]\}$ A probability distribution on the closed interval $[0, 1]$

What is Elicitation Doing?

Reduces uncertainty in \hat{U}

State of the Art

- Lots of work on preferences/utility elicitation in decision making domain.
- Some work on incremental utility elicitation.
- Few works on incremental utility elicitation during negotiations

State of the Art

- Lots of work on preferences/utility elicitation in decision making domain.
- Some work on incremental utility elicitation.
- Few works on incremental utility elicitation during negotiations

Why Is Negotiation Different

- ① The acceptance model changes over time → environment dynamics are not static.
- ② Exploration is extremely costly.
- ③ Usually negotiations are not repeated much.
- ④ Cannot train on a simulator (in most cases).

Value of Information Algorithm

- Based on¹⁰ in decision-support context.
- Adapted to the negotiation context.

Main Idea

- Assume an accurate opponent model (acceptance probability)
- Given a set of queries $Q \rightarrow$ find the one with the maximum difference between the expected expected utility before and after asking it¹¹¹².

¹⁰ Urszula Chajewska, Daphne Koller, and Ronald Parr. "Making rational decisions using adaptive utility elicitation". In: AAAI/IAAI. 2000, pp. 363–369.

¹² Tim Baarslag and Michael Kaisers. "The value of information in automated negotiation: A decision model for eliciting user preferences". In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. International Foundation for Autonomous Agents and Multiagent Systems. 2017, pp. 391–400.

¹² Yasser Mohammad and Shinji Nakadai. "FastVOI: Efficient Utility Elicitation During Negotiations". In: International Conference on Principles and Practice of Multi-Agent Systems (PRIMA). Springer. 2018, pp. 560–567.

VOI Based Elicitation

Policy

$\pi^t = (\omega^t, \omega^{t+1}, \dots, \omega^N)$ where $\omega^x \in \Omega$ $K(\omega|\pi) \equiv$ index of ω in π , $\pi(k) = \omega$ where $K(\omega|\pi) = k$

Optimal Policy

$$\pi^{t*} = \arg \max_{\pi} EEU^t \left(\pi, \left\{ \hat{U}_{\omega}^t \right\} \right)$$

VOI Based Elicitation

Policy

$\pi^t = (\omega^t, \omega^{t+1}, \dots, \omega^N)$ where $\omega^x \in \Omega$ $K(\omega|\pi) \equiv$ index of ω in π , $\pi(k) = \omega$ where $K(\omega|\pi) = k$

Probability of Agreement

$$Pa^t(\omega|\pi) = \begin{cases} \Lambda^t(\omega) \prod_{k=1}^{K_\pi(\omega)-1} (1 - \Lambda^t(\pi(k))) & \omega \in \pi \\ 0 & \text{otherwise} \end{cases}$$

Optimal Policy

$$\pi^{t*} = \arg \max_{\pi} EEU^t \left(\pi, \left\{ \hat{U}_{\omega}^t \right\} \right)$$

VOI Based Elicitation

Policy

$\pi^t = (\omega^t, \omega^{t+1}, \dots, \omega^N)$ where $\omega^x \in \Omega$ $K(\omega|\pi) \equiv$ index of ω in π , $\pi(k) = \omega$ where $K(\omega|\pi) = k$

Probability of Agreement

$$Pa^t(\omega|\pi) = \begin{cases} \Lambda^t(\omega) \prod_{k=1}^{K_\pi(\omega)-1} (1 - \Lambda^t(\pi(k))) & \omega \in \pi \\ 0 & \text{otherwise} \end{cases}$$

Expected Expected Utility¹³

$$EEU^t\left(\pi, \left\{ \hat{U}_\omega^t \right\}\right) = \sum_{\omega \in \Omega} Pa(\omega|\pi) \mathbb{E}\left(\hat{U}_\omega^t\right)$$

Optimal Policy

$$\pi^{t*} = \arg \max_{\pi} EEU^t\left(\pi, \left\{ \hat{U}_\omega^t \right\}\right)$$

VOI Based Elicitation II

Questions

$$\begin{aligned} Q &\equiv \{q_I\} \\ q_I &\equiv \{(Ans_s^I, p_s)\} \end{aligned}$$

Answers

$$\begin{aligned} Ans_s^I &\equiv \{\hat{U}_\omega^{t+1}\} \\ \sum_s p_s &= 1 \end{aligned}$$

VOI Based Elicitation II

Questions

$$\begin{aligned} Q &\equiv \{q_I\} \\ q_I &\equiv \{(Ans_s^I, p_s)\} \end{aligned}$$

Answers

$$\begin{aligned} Ans_s^I &\equiv \{\hat{U}_\omega^{t+1}\} \\ \sum_s p_s &= 1 \end{aligned}$$

Expected value of information

$$EVOI(q', \{\hat{U}_\omega^t\}) = \mathbb{E}_s (\max_{\pi} EEU(\pi, Ans_s')) - \max_{\pi} EEU(\pi, \{\hat{U}_\omega^t\})$$

VOI Based Elicitation II

Questions

$$\begin{aligned} Q &\equiv \{q_I\} \\ q_I &\equiv \{(Ans_s^I, p_s)\} \end{aligned}$$

Answers

$$\begin{aligned} Ans_s^I &\equiv \{\hat{U}_\omega^{t+1}\} \\ \sum_s p_s &= 1 \end{aligned}$$

Expected value of information

$$EVOI(q', \{\hat{U}_\omega^t\}) = \mathbb{E}_s (\max_{\pi} EEU(\pi, Ans_s')) - \max_{\pi} EEU(\pi, \{\hat{U}_\omega^t\})$$

Elicitation

Ask q^* where

$$q^* = \arg \max_q (EVOI(q', \{\hat{U}_\omega^t\}) - c_q)$$

c_q Cost of asking question q

VOI main Issues

Accurate Agreement Model Assumption

- Everything depends on the probability of agreement (Pa)
- Pa depends on the **product** of probabilities in the acceptance model (Λ^t)

$$Pa^t(\omega|\pi) = \begin{cases} \Lambda^t(\omega) \prod_{k=1}^{K_\pi(\omega)-1} (1 - \Lambda^t(\pi(k))) & \omega \in \pi \\ 0 & \text{otherwise} \end{cases}$$

VOI main Issues

Accurate Agreement Model Assumption

- Everything depends on the probability of agreement (Pa)
- Pa depends on the **product** of probabilities in the acceptance model (Λ^t)

$$Pa^t(\omega|\pi) = \begin{cases} \Lambda^t(\omega) \prod_{k=1}^{K_\pi(\omega)-1} (1 - \Lambda^t(\pi(k))) & \omega \in \pi \\ 0 & \text{otherwise} \end{cases}$$

Speed: Complexity = $O(nN|Q||Ans|)$

- Too many *argmax* and \mathbb{E} operations.
- Every policy extends to the end of the negotiation.

$$q^* = \arg \max_q \left(EVOI \left(q', \left\{ \hat{U}_\omega^t \right\} \right) - c_q \right)$$

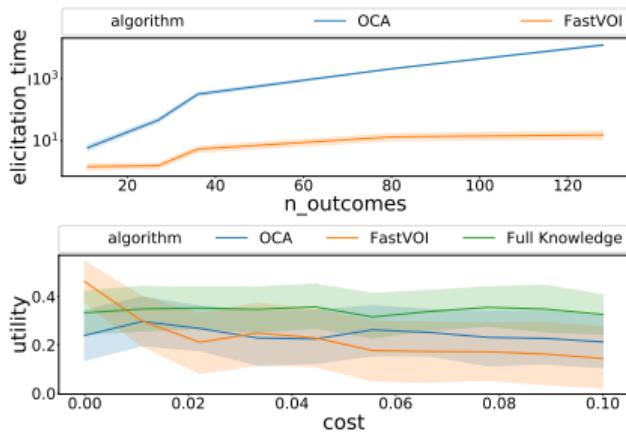
$$EVOI \left(q', \left\{ \hat{U}_\omega^t \right\} \right) = \mathbb{E}_s \left(\max_\pi EEU \left(\pi, Ans_s' \right) \right) - \max_\pi EEU \left(\pi, \left\{ \hat{U}_\omega^t \right\} \right)$$

$$\pi^{t*} = \arg \max_\pi EEU^t \left(\pi, \left\{ \hat{U}_\omega^t \right\} \right)$$

Extending VOI

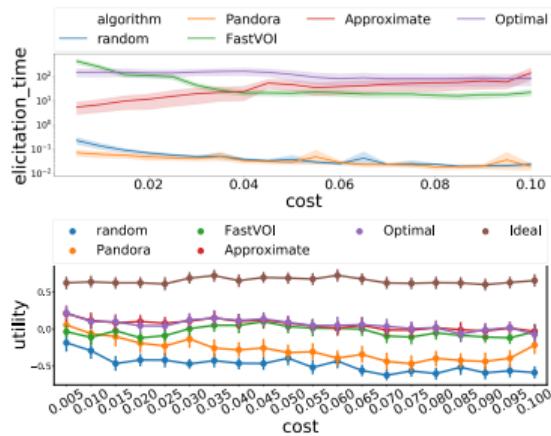
FastVOI¹⁴

- A faster approximate version of VOI



OptimalVOI¹⁵

- Extends Applicability to Infinite N. Questions.



¹⁵Yasser Mohammad and Shinji Nakadai. "Fastvoi: Efficient utility elicitation during negotiations". In: *International Conference on Principles and Practice of Multi-Agent Systems*. Springer. 2018, pp. 560–567.

¹⁵Yasser Mohammad and Shinji Nakadai. "Optimal value of information based elicitation during negotiation". In: *Proceedings of the 18th international conference on autonomous agents and multiagent systems*. 2019, pp. 242–250.

Outline



- 1 Learning in Automated Negotiation
- 2 Learning how to negotiate
- 3 Learning Preferences
- 4 References

References I

- Baarslag, Tim and Michael Kaisers. "The value of information in automated negotiation: A decision model for eliciting user preferences". In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2017, pp. 391–400.
- 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.
- Boutilier, Craig. "On the foundations of expected expected utility". In: *IJCAI*. Vol. 3. Citeseer. 2003, pp. 285–290.
- Chajewska, Urszula, Daphne Koller, and Ronald Parr. "Making rational decisions using adaptive utility elicitation". In: *AAAI/IAAI*. 2000, pp. 363–369.
- Hindriks, Koen and Dmytro Tykhonov. "Opponent modelling in automated multi-issue negotiation using bayesian learning". In: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*. 2008, pp. 331–338.

References II

- 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.
- Krimpen, Thijs van, Daphne Looije, and Siamak Hajizadeh. "HardHeaded". In: *Complex Automated Negotiations: Theories, Models and Software Competitions*. Ed. by Ito Takayuki et al. Springer, 2013, pp. 223–227.
- Mohammad, Yasser and Shinji Nakadai. "FastVOI: Efficient Utility Elicitation During Negotiations". In: *International Conference on Principles and Practice of Multi-Agent Systems (PRIMA)*. Springer. 2018, pp. 560–567.
- . "Fastvoi: Efficient utility elicitation during negotiations". In: *International Conference on Principles and Practice of Multi-Agent Systems*. Springer. 2018, pp. 560–567.
 - . "Optimal value of information based elicitation during negotiation". In: *Proceedings of the 18th international conference on autonomous agents and multiagent systems*. 2019, pp. 242–250.
- 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.

References III

- 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.

Automated Negotiation: Challenges and Tools

Automagted Negotiation is SCM

Yasser Mohammad^{1, 2, 3} Amy Greenwald⁴

¹ NEC Corporation, Global Innovation Unit

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

³Assiut University, Egypt

⁴Brown University, USA

February 23, 2022



BROWN
UNIVERSITY



NEC-AIST
AI Cooperative
Research Laboratory



Outline

- 1 Automated Negotiation in SCML
- 2 The SCML Game
- 3 References

Outline

1 Automated Negotiation in SCML

2 The SCML Game

3 References

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/3kDRy6I>

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/3kDRy6I>

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.

The logo for Contract Room, featuring the word "CONTRACT" in blue and "ROOM" in green.The logo for Pactum, featuring the word "pactum" in blue with a stylized orange swoosh graphic.

¹KPMG report: <https://bit.ly/3kDRy6I>

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".

Outline

1 Automated Negotiation in SCML

2 The SCML Game

- SCML-OneShot
- Simulation Steps

3 References

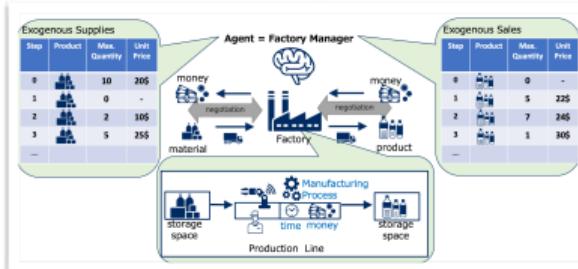
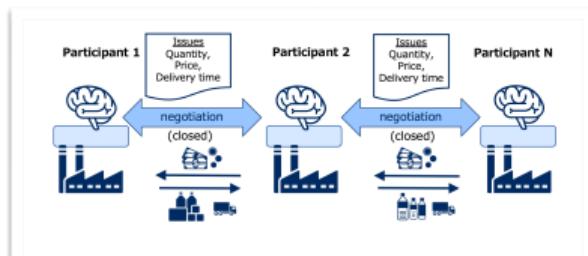
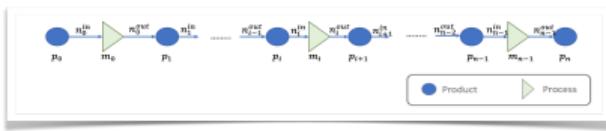
SCML World

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).

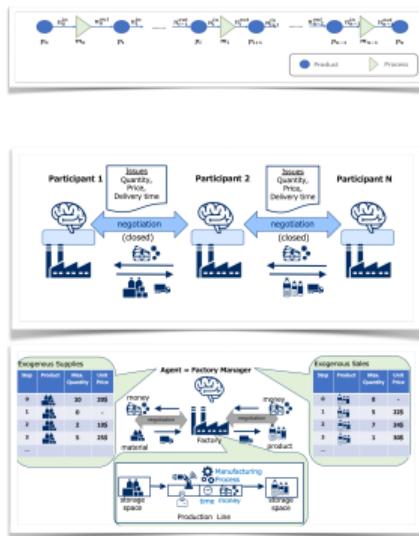
SCML Competition

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.



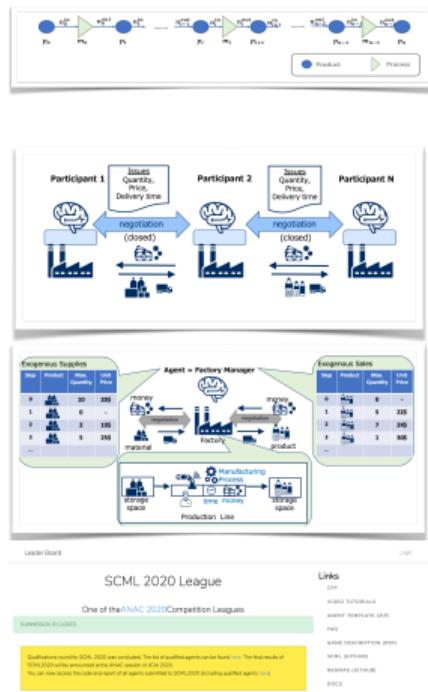
SCML Competition

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.



Outline

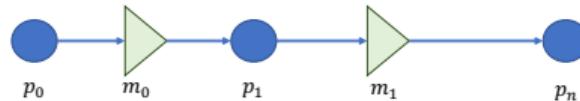
1 Automated Negotiation in SCML

2 The SCML Game

- SCML-OneShot
 - Available Information
 - Simulation Steps

3 References

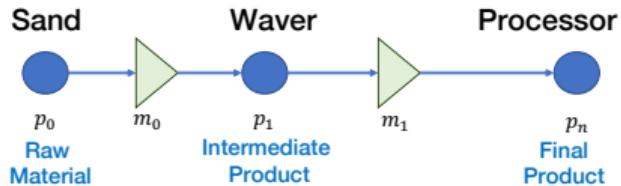
Overview



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



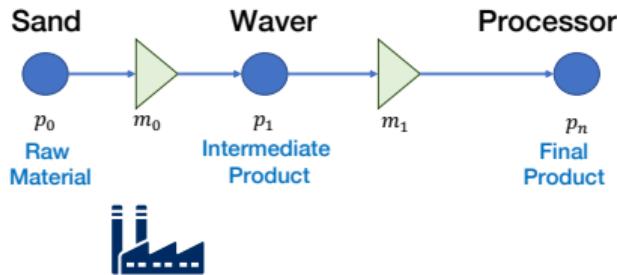
Overview



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

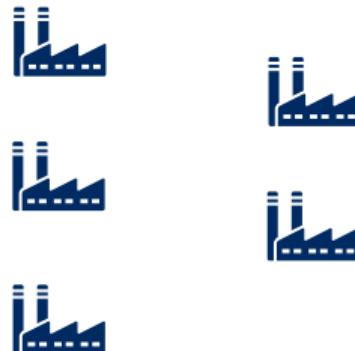
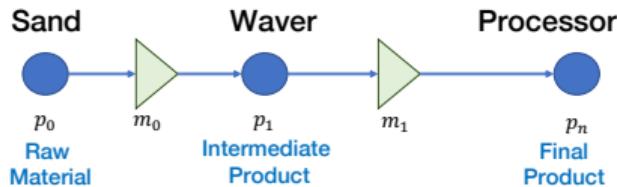


Overview



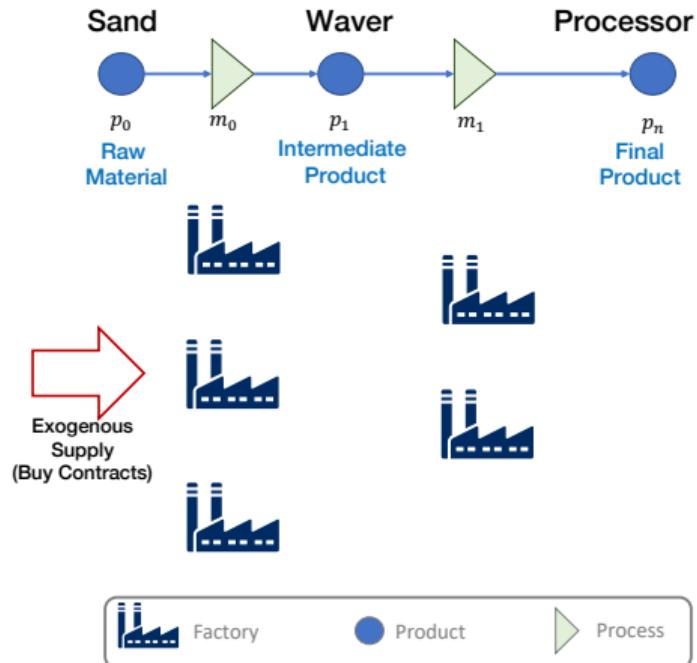
- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.

Overview



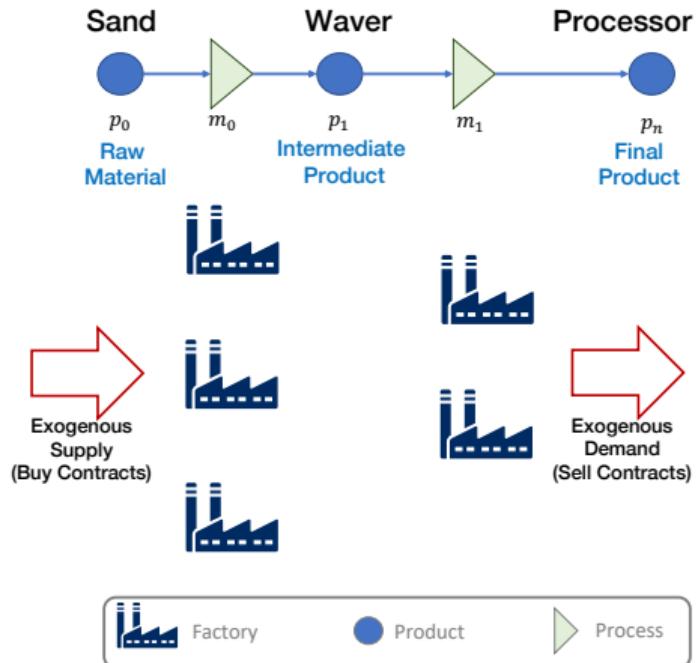
- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.
- We have two layers of factories/agents.

Overview



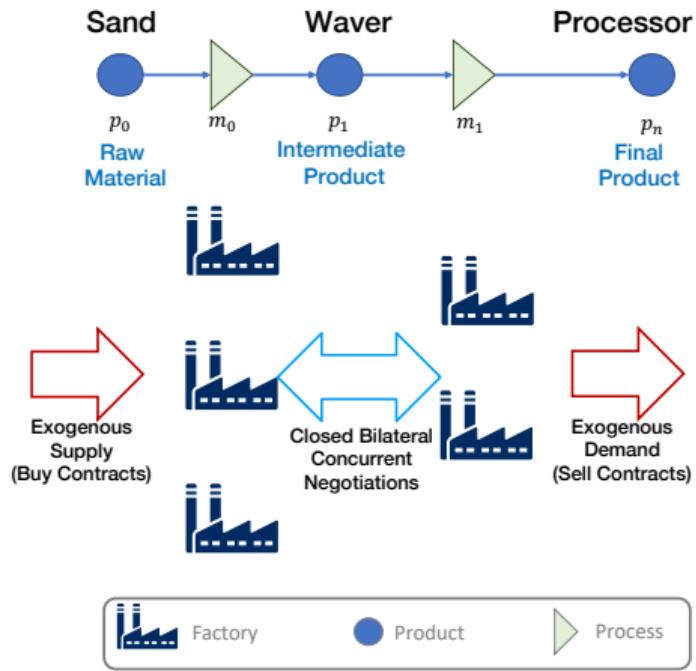
- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.
- We have two layers of factories/agents.
- L_0 factories/agents receive exogenous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

Overview



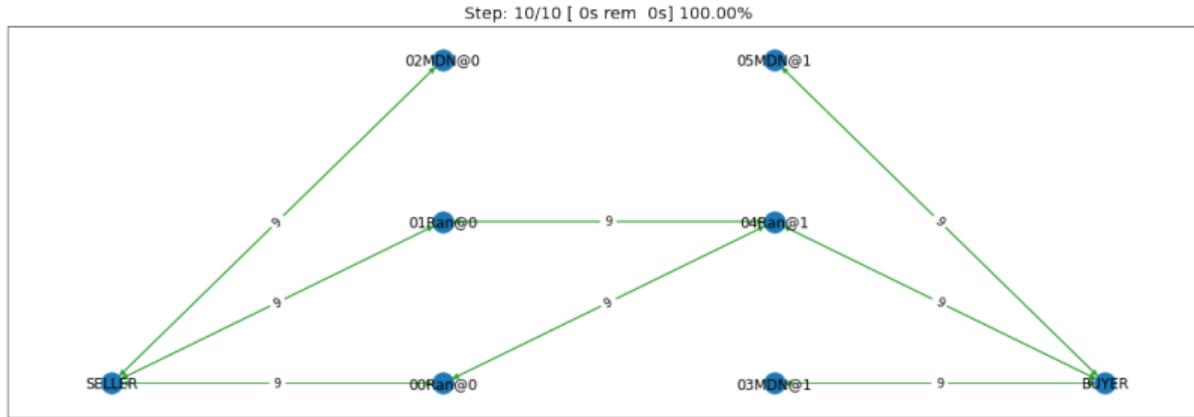
- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.
- We have two layers of factories/agents.
- L_0 factories/agents receive exogenous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

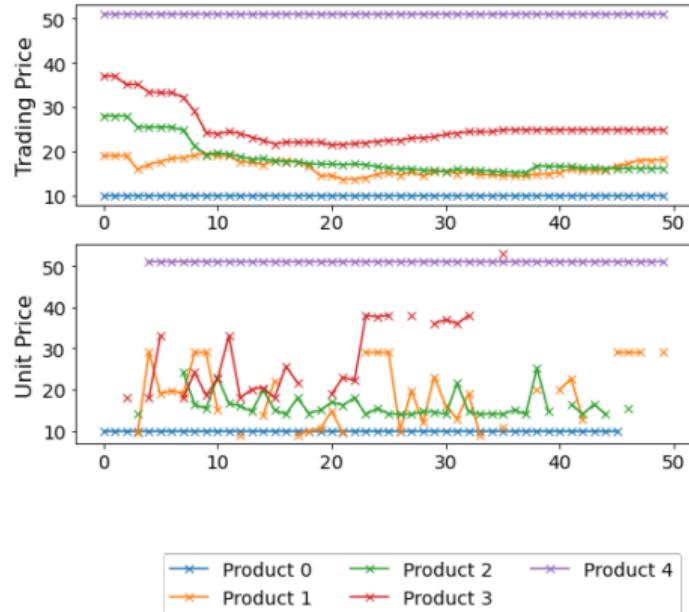
Market Information

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.



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).

Outline

1 Automated Negotiation in SCML

2 The SCML Game

- SCML-OneShot
- Simulation Steps

3 References

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



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

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.

Finalize agents for the day

• `step()` [Last call every day]

Publish Financial Reports



Outline



- 1 Automated Negotiation in SCML
- 2 The SCML Game
- 3 References

References I

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.

Automated Negotiation: Challenges and Tools

Future Challenges

Yasser Mohammad^{1, 2, 3} Amy Greenwald⁴

¹ NEC Corporation, Global Innovation Unit

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

³Assiut University, Egypt

⁴Brown University, USA

February 23, 2022



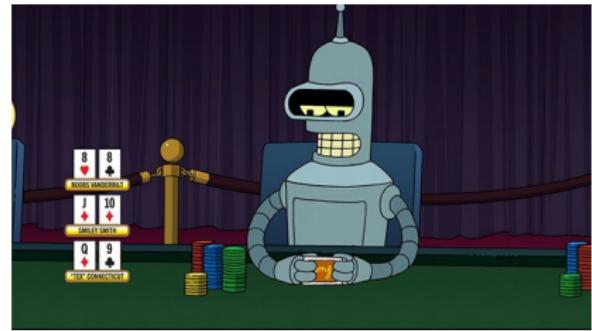
Outline

- 1 AI in Games
- 2 Finding a Best Response to a Fixed Opponent
- 3 Finding a Best Response to Multiple Fixed Opponents
- 4 Conclusion
- 5 References

Outline

- 1 AI in Games
- 2 Finding a Best Response to a Fixed Opponent
- 3 Finding a Best Response to Multiple Fixed Opponents
- 4 Conclusion
- 5 References

AI has learned to play games (well!)



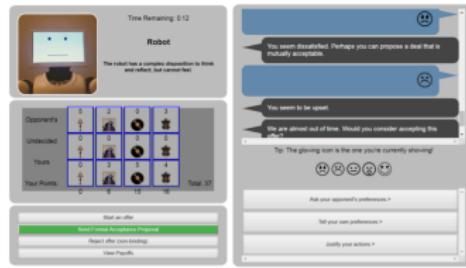
A brief history of ANAC



2022	??, HAN, Werewolf, SCML	2016	En
2021	ANL (Repeated), HAN, Werewolf, SCML	2015	The
2020	ANL (Elicitation), HAN, Werewolf, SCML , HUMAINE	2014	Le
2019	Uncertainty, Diplomacy, HAN, Werewolf, SCML	2012	Re
2018	Repeated Negotiations, Diplomacy, HAN	2011	Lir
2017	Repeated Negotiations, Diplomacy, HAN	2010	Do

ANAC 2021 Leagues

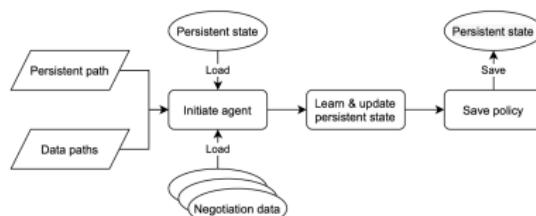
HUMAINE: Negotiating with a Human



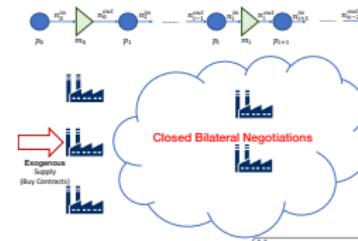
Werewolf: Negotiating with M



ANL: Learning in Repeated Negotiations (Preference Elicitation)



SCML: Supply Chain Management



Game Characteristics

Solved Games

- Sequential decision making (e.g., turn-taking)
- Imperfect (and perfect) information
- Two-player & multi-player
- **Zero-sum**

Agent Negotiation

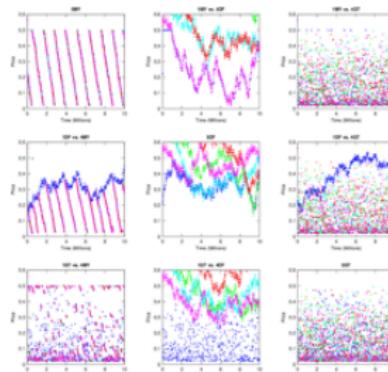
- Sequential decision making (e.g., turn-taking)
- Imperfect (and perfect) information
- Two-player & multi-player
- **General-sum**

Empirical Game-Theoretic Analysis (EGTA)

EGTA is a MAS tool for analyzing multiagent interactions.¹

Key Idea: Derive so-called **empirical games** from data. Solve these games.

Empirical games are often higher-level games, played with higher-level strategies (i.e., heuristics).²³



	4MY	4DF	4GT
1MY	.0337, .0337	.0690, .0225	.0185, .0109
1DF	.0136, .0335	.0387, .0387	.0134, .0159
1GT	.0119, .0169	.0536, .0226	.0129, .0129

Double Oracle Algorithm⁴(PSRO⁵)

Theorem The double oracle algorithm converges to a Nash equilibrium in zero-sum games.

Algorithm 1 Double Oracle Algorithm

Input: A game with strategy sets Π_i , for $i \in \{1, 2\}$

Input: An initial strategy set $\Pi_i^0 \in \Pi_i$, for $i \in \{1, 2\}$

Output: Nash equilibrium

- 1: **repeat** $t \in \{0, 1, 2, \dots\}$
- 2: Find π^* , a Nash equilibrium in the game Π_i^t , for $i \in \{1, 2\}$
- 3: **for** $i \in \{1, 2\}$ **do**
- 4: Find a best response $\beta_i \in \Pi_i$ to π_{-i}^*
- 5: Expand strategy set: $\Pi_i^{t+1} \leftarrow \Pi_i^t \cup \beta_i$
- 6: **until** $\Pi_i^{t+1} = \Pi_i^t$, for $i \in \{1, 2\}$
- 7: **Return:** π^*

Outline

- 1 AI in Games
- 2 Finding a Best Response to a Fixed Opponent
- 3 Finding a Best Response to Multiple Fixed Opponents
- 4 Conclusion
- 5 References

Finding a Best Response to a Fixed Opponent

Learning a best-response policy, offline

- Fixed acceptance policy: use RL to learn an offer policy
- Fixed offer policy: use deep learning to learn an acceptance policy

Some issues with this approach

- Why decouple these decisions? Doing so is unlikely to be optimal.
Perhaps for tractability?
- Possible way out: Formulate as an MDP and solve
- An even bigger problem: when is a negotiating partner so benevolent that they give you access to a simulator of their strategy that you can run for thousands of iterations to learn a best response?
- Possible way out: EGTA

Negotiation as an MDP

Assume a **fixed** opponent: one's whose strategy does depend *not* on our actions.

E.g., an **aspiration** agent.

When the opponent is known, our agent's decision making problem is an **MDP**.

- States: negotiation round plus two designated states, agreement and disagreement
- Actions: offers and accept/reject
- Transitions: capture opponent's behavior (e.g., if they accept, transition to the agreement state)
- Rewards: zero everywhere except at agreement, where they are given by the agent's utility function

Solving MDPs:

- Planning (e.g., VI), if the opponent model is in closed form
- RL, if we have only a generative model of the opponent model

Opponent Model

An opponent model comprises **an acceptance model** and **an offer policy**.

(Opponent ufuncs are not modelled, because our decisions are conditionally independent of them, given an acceptance model and an offer policy.)

TDAM: time-dependent acceptance model

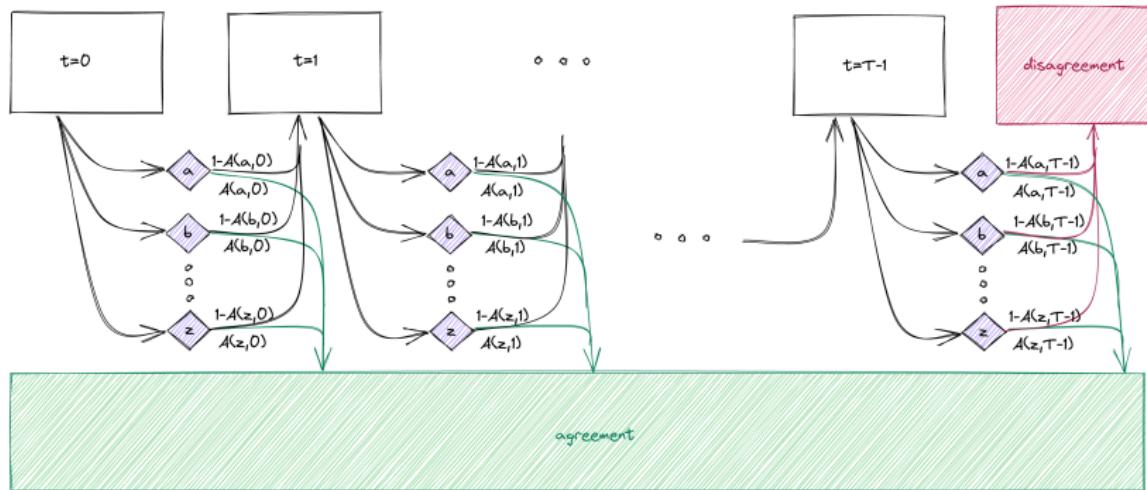
A map from a time to an acceptance probability distribution

TDOP: time-dependent offer policy

A map from a time to opponent offers (or probabilities over offers)

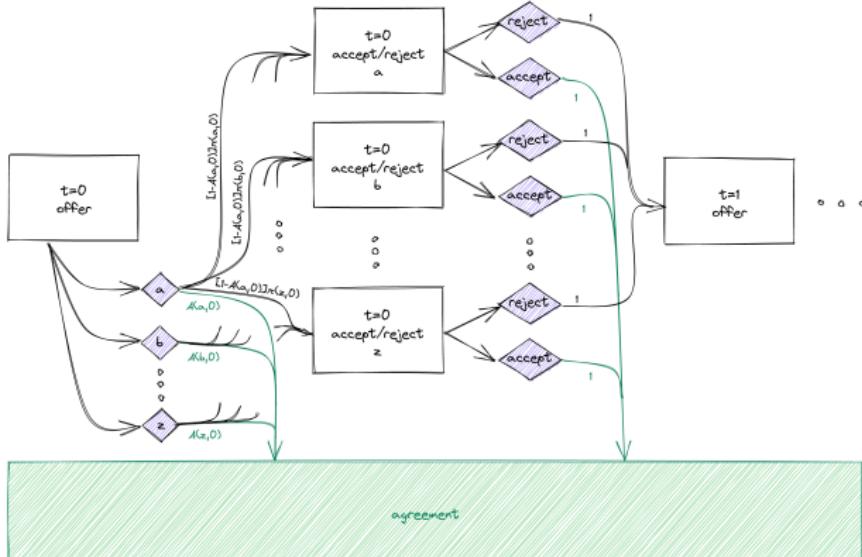
Time = negotiation round.

TDAM MDP



Drawing by Jackson de Campos

TDAM + TDOP MDP



Drawing by Jackson de Campos

EGTA, Revisited

- Goal of EGTA: build agents that are robust, against a population of opponents.
- Each TDAM + TDOP MDP corresponds to an opponent.
A solution to each MDP is an optimal negotiation strategy for playing against that opponent.
- Start from an initial population of opponents (i.e., MDPs) \Rightarrow negotiation strategies.
- Use EGTA to grow the population of negotiation strategies \Rightarrow opponents (i.e., MDP).
- End result should be a robust agent!

Outline

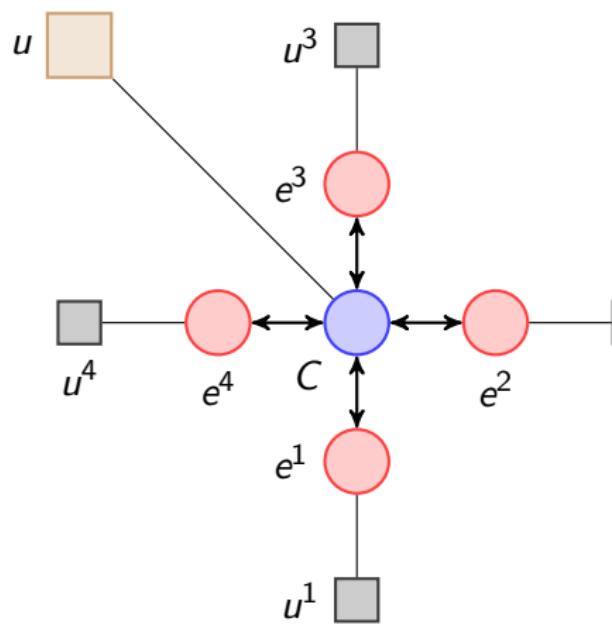
- 1 AI in Games
- 2 Finding a Best Response to a Fixed Opponent
- 3 Finding a Best Response to Multiple Fixed Opponents
- 4 Conclusion
- 5 References

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.⁶ 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



Expected Marginal Utility

An **outcome prediction** is a joint distribution over the outcomes of all (say, n) concurrent negotiations:

i.e., $P(\omega) = P(\omega_1, \dots, \omega_n)$, where $\omega_i \in \Omega_i$ and $\Omega = \prod_{i=1}^n \Omega_i$.

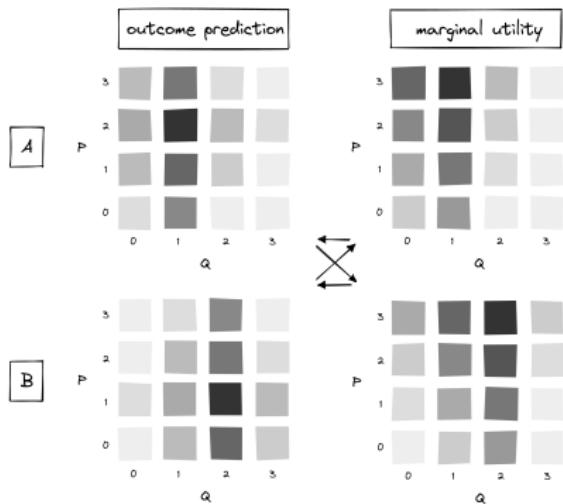
Given an outcome prediction P , the **expected marginal utility** μ_i of outcome $\omega_i \in \Omega_i$ in negotiation i is:

$$\mu_i(\omega_i; P) = \mathbb{E}_{\Omega_{\neg i} \sim P_{\neg i}} [u(\{\omega_i\} \cup \Omega_{\neg i}) - u(\Omega_{\neg i})] ,$$

where $P_{\neg i}$ is the outcome prediction for all negotiations other than i (i.e., P marginalized over i).

SCML OneShot: GodFather Strategy

- We are a seller with three goods to sell
- The arrows indicate dependencies in the expected marginal utility calculations
- Darker squares indicate higher probability predictions and higher marginal utilities
- Since we predict opponent *A* to most likely want 2 goods and *B* to most likely want 1, the highest marginal utility is obtained by contracting to sell 1 to opponent *A* and 2 to opponent *B* (at the highest possible prices)



Outcome Predictions

An outcome prediction is a joint distribution over the outcomes of all concurrent negotiations:

i.e., $P(\omega) = P(\omega_1, \dots, \omega_n)$

- Static model: Depends on factors external to the current negotiation, e.g., the state of the world, past negotiations, etc.

$P(\omega \mid \text{external factors})$

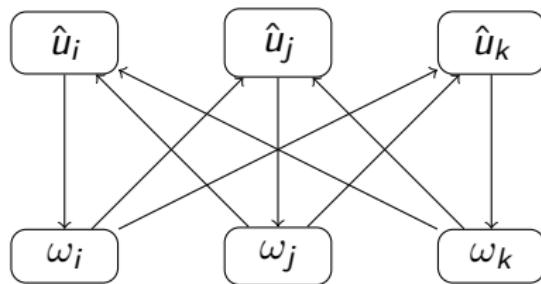
- Dynamic model: Depends on factors both external and internal to the current negotiation,

i.e., the current negotiation trace so far

$P(\omega \mid \text{both external and internal factors})$

- Introspective model: Expected marginal utilities depend on the outcome predictions. Likewise, the outcome predictions depend on the expected marginal utilities (because they affect our behavior).

Introspective Equilibrium



An **introspective equilibrium**⁷ is a situation in which the outcome predictions are consistent with the marginal utilities, meaning $P(\omega | \dots \text{ and expected marginal utilities } \mu(\cdot; f)) = f$.

Equivalently, for all negotiations x ,

$\mu_i(\omega_i; P(\omega | \dots \text{ and expected marginal utilities } f)) = f_i(\omega_i)$.

⁷ wellman17self.

Open Questions

Decoupling negotiations tames the complexity, but is this kosher? How far from optimal is this approach?

How do you build reliable prediction models: static, dynamic, and introspective?

When does simple iteration converge to an introspective equilibrium?

Outline

- 1 AI in Games
- 2 Finding a Best Response to a Fixed Opponent
- 3 Finding a Best Response to Multiple Fixed Opponents
- 4 Conclusion
- 5 References

Summary

Conclusions

Outline

- 1 AI in Games
- 2 Finding a Best Response to a Fixed Opponent
- 3 Finding a Best Response to Multiple Fixed Opponents
- 4 Conclusion
- 5 References

References I

references

Automated Negotiation is important

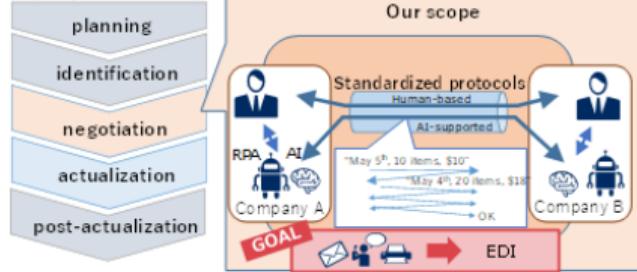
Why is it hard?

- 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

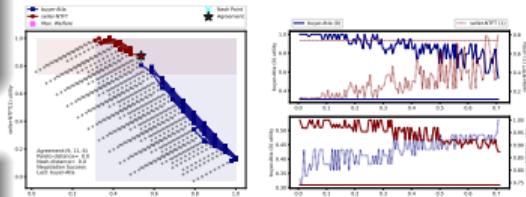


Automated Negotiating Agents Competition: 2010-

Automated Negotiation Has a Long History

Nash Bargaining Game (1950)

How to split a pie when you have one offer? Maximize relative surplus product

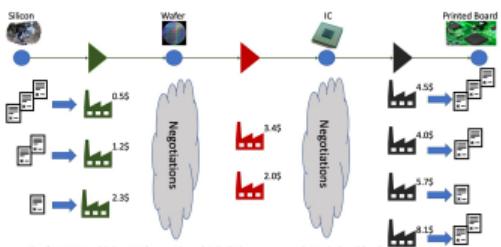


Robenstien Bargaining Game (197s)

Do we need to negotiate if we know everything? Not really

ANAC: 2010 and still running

Can we develop effective agents for automated negotiation in almost any domain? Kind of



An example of an SCM world showing four products (circles), three processes (triangles) and few factories. Each process consumes one item of its input and generates one output of its output in one day. Each factory requires a different cost to run its process (shown in its top right). Factories in the first level have exogenous contracts to buy new material (silicon) and factories at the last level have exogenous contracts to sell the final product (printed boards). These contracts drive the market.

SCML 2019 and still running

Can we develop effective agents that orchestrate multiple related negotiations while being embedded in a business like environment? Kind

We have Nice Platforms?

Genius¹

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

GENIUS

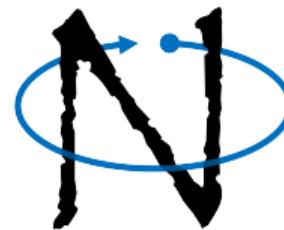
>> General Environment for Negotiation with Intelligent multi-purpose Usage Simulation.

GeniusWeb

A distributed platform for automated negotiation on the internet

NegMAS²

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



You know what it takes to build a negotiator

Offer Policy

What should I offer next?

- Good Heuristics exist
 - Time-based
 - Tit-for-Tat
 - ANAC agents (more than 50)
- Good learners exist but the best ones require unrealistically large amounts of data.

Acceptance Strategy

When should I accept my partner's offer?

- Several heuristics exist.
- We can learn good acceptance Strategies.
- Still, it may not be a good idea to separate acceptance from offering.

Opponent Model

There is too much still to be done

Strategies with Guarantees

Can we develop strategies that combine the guarantees of classic Game theoretic work with the effectiveness of modern heuristics?

Concurrent Negotiation

How coordinate behavior in multiple negotiation threads?

Negotiation under uncertainty

How to behave when I do not know exactly what I want?

Preference Elicitation

Should I ask? And what exactly should I ask about?