# Automated Negotiation for Supply Chain Management

Yasser Mohammad [1, 2, 3]

[1] NEC Corporation, Global Innovation Unit

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

[3] Assiut University, Egypt

December 4, 2023



**ICA 2023 Tutorial** Updated on December 4, 2023

# Outline

# Why Now?

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

# Negotiation in SCM Business

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

[1] KPMG report: https://bit.ly/3kDRy6I
[2] Forrester report: https://bit.ly/3nwXEaY
[3] UN/CEFACT Project website: https://bit.ly/38LOsLX
[4] Y. 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].

[1] KPMG report: https://bit.ly/3kDRy6I
[2] Forrester report: https://bit.ly/3nwXEaY
[3] UN/CEFACT Project website: https://bit.ly/38LOsLX
[4] Mohammad 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]

[1] KPMG report: https://bit.ly/3kDRy6I
[2] Forrester report: https://bit.ly/3nwXEaY
[3] UN/CEFACT Project website: https://bit.ly/38LOsLX
[4] Mohammad et al., "Supply Chain Management World: A benchmark environment for situated negotiations".

# Negotiation in SCM Business

NEC-AIST AI Cooperative Research Laboratory

- 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 come[4].

CONTRACTROOM

pactum

[1] KPMG report: https://bit.ly/3kDRy6I
[2] Forrester report: https://bit.ly/3nwXEaY
[3] UN/CEFACT Project website: https://bit.ly/38LOsLX
[4] Mohammad et al., "Supply Chain Management World: A benchmark environment for situated negotiations".

# The Automated Negotiation Challenge

NEC-AIST AI Cooperative Research Laboratory

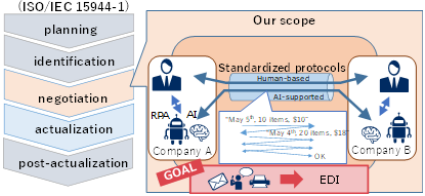
## Why is it hard?

- Mechanism Design Problem:
  - Better than haggling?
- Negotiator Design Problem:
  - Generality $\times$ Effectiveness

## Why is it interesting?

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

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




attribution: UNECE eNegotiation Project


Automated Negotiating Agents Competition: 2010-

# Tutorial Outline

1. Theoretical Session (55min) Break (10min)
   - The Negotiation Problem (35min)
   - Automated Negotiating Agents Competition (ANAC) (5min)
   - Supply Chain Management League 15min
   - Anatomy of an Agent (10min)
2. Development Environment (50min) Break (10min)
   - Installing SCML (5min)
   - Running a simulation (5min)
   - Running Tournaments (5min)
   - Visualization and Logging (15min)
   - Troubleshooting and breathing time (10min)
3. Development Example (55min)
   - SCML Builtin Strategies (20min)
   - Example Agent Strategies (15min)
   - An RL based trading strategy (20min)
4. Conclusions (5min)

# Materials

1. Tutorial Website: http://yasserm.com/tutorial-ica2023/
2. Github Repository: https://github.com/yasserfarouk/ica2023autoneg
3. Handouts: https://github.com/yasserfarouk/ica2023autoneg/raw/main/ica2023autoneg.pdf
4. Negmas Documentation: https://negmas.readthedocs.io
5. SCML Documentation: https://scml.readthedocs.io
6. SCML Competition: https://scml.cs.brown.edu

# Outline

# Outline

# Types of Games

NEC-AIST AI Cooperative Research Laboratory

## Complete Information Game

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

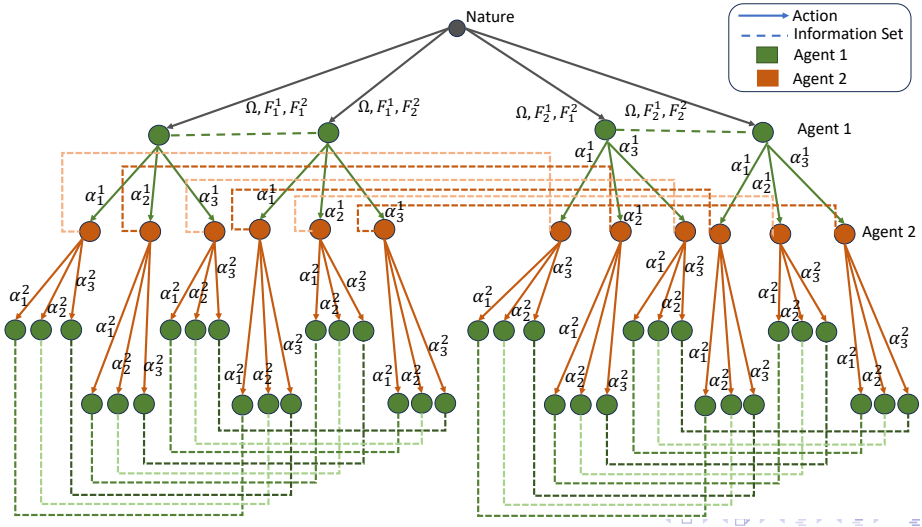
## Imperfect Information Game

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

## Incomplete Information Game

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

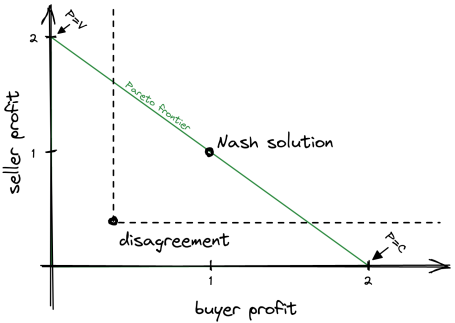
# Induced Game with Incomplete Information



# A Simple Trading Problem

- A buyer values a good at $V$
- A seller can create the good at cost $C$
- If $V > C$, then there is surplus $V - C$ to be gained (value creation)
- Bargaining problem: how much should the buyer pay the seller for the good? (value division)

- We might also assume there is an outside option (e.g., eBay), if the negotiation breaks down (i.e., they do not reach an agreement):
  - The buyer (seller) can buy (sell) the good elsewhere for slightly less than $V$ (more than $C$)




Sketch by Jackson de Campos

# No-go Theorem

NEC-AIST AI Cooperative Research Laboratory

Desiderata:

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

Myerson-Satterthwaite Impossibility Result

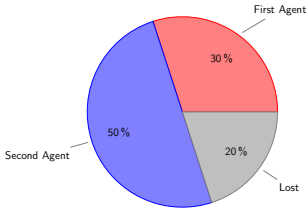
Theorem:[5] No mechanism can achieve all four of these desiderata.

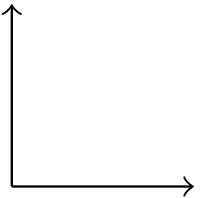
- A buyer values a good at $V$.
- A seller can create the good at cost $C$.
- $V$ ($C$) is private information, known only to the buyer (seller).
- There is no IR, IC, and BB mechanism that results in agreement, for all $V > C$ values.

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

# Nash Bargaining Game: Description

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



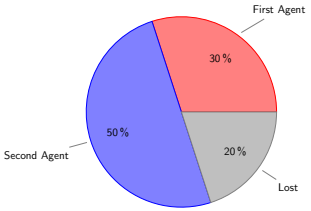



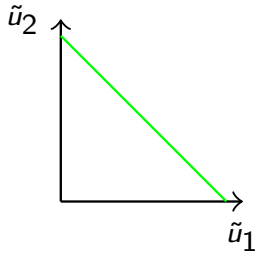
# Nash Bargaining Game: Description

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

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

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



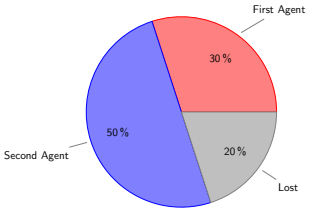



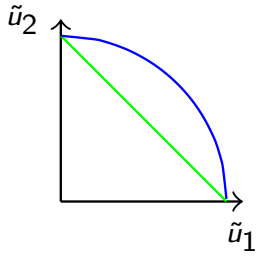
# Nash Bargaining Game: Description

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

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

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



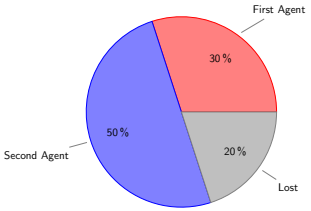



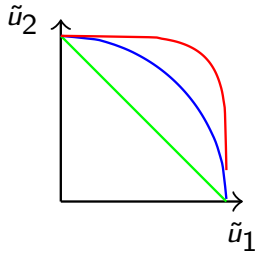
# Nash Bargaining Game: Description

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

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

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



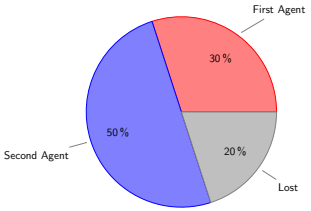
# Nash Bargaining Game: Description

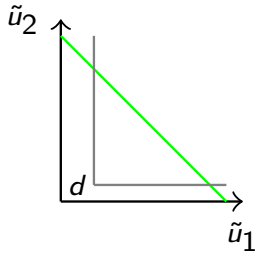
A single-step full-information bilateral negotiation with $\Omega = [0, 1]^2$ and two utility functions $(\tilde{u}_1, \tilde{u}_2)$ such that:

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

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

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



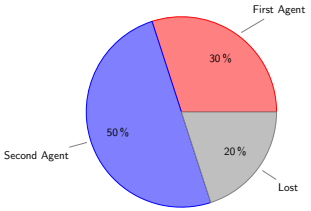
# Nash Bargaining Game: Description

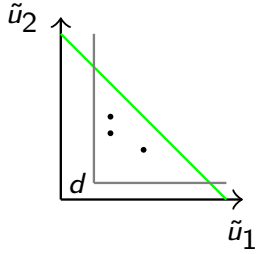
A single-step full-information bilateral negotiation with $\Omega = [0, 1]^2$ and two utility functions $(\tilde{u}_1, \tilde{u}_2)$ such that:

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

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

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



# Other Bargaining Solutions

NEC-AIST AI Cooperative Research Laboratory

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

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

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

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

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

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

13/ 48

# Rubinstein's Bargaining Protocol: Description

NEC-AIST AI Cooperative Research Laboratory

## The Game

- Two agents sharing a pie.

# Rubinstein's Bargaining Protocol: Description



## The Game

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

# Rubinstein's Bargaining Protocol: Description



## The Game

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

# Rubinstein's Bargaining Protocol: Description



## The Game

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

# Rubinstein's Bargaining Protocol: Description



## The Game

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

# Rubinstein's Bargaining Protocol: Description



## The Game

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

# Rubinstein's Bargaining Protocol: Description

NEC-AIST AI Cooperative Research Laboratory

## The Game

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

# Rubinstein's Bargaining Protocol: Description

NEC-AIST AI Cooperative Research Laboratory

## 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

NEC-AIST AI Cooperative Research Laboratory

## Exponential Discounting

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

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

# Rubinstein's Bargaining Protocol: Equilibrium

NEC-AIST AI Cooperative Research Laboratory

## 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:

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

# Negotiation with Complete Information

NEC-AIST AI Cooperative Research Laboratory

## Hick's Paradox

Why do rational parties negotiate when they have full information?

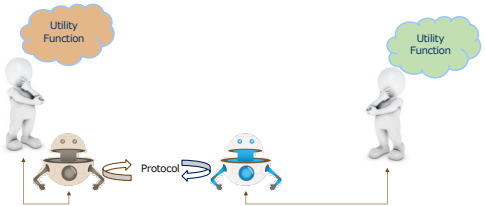
## Labor vs. Management

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

Theorem[6] Subgame perfect equilibria exist in which there is a finite strike followed by agreement.

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

# Components of Negotiation




Negotiation Scenario Defines who is negotiating about what

Negotiation Protocol Defines how negotiation is to be conducted

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

Negotiation Strategy Defines how agents behave during a negotiation

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

# Notation

NEC-AIST AI Cooperative Research Laboratory

Negotiation Scenario $\mathcal{S}$

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

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

- Outcome Space $K$: A set of all possible agreements with $\phi \notin K$ representing disagreement. $K^+ \equiv K \bigcup \{\phi\}$ is called the Extended Outcome Space.
- Preferences Tuple $\mathcal{F}^{\mathcal{S}}$: A tuple of $n_A$ members that defines the preferences of each agent engaged in the negotiation.
- Information Set Tuple $\mathcal{I}^{\mathcal{S}}$: A tuple of $n_A$ information sets $\left(\left\{\mathrm{I}_f^{\mathcal{S}}(\mathcal{F}^{\mathcal{S}}) \mid f \in \mathcal{A}^{\mathcal{S}}\right\}\right)$ where $\mathrm{I}_f^{\mathcal{S}}(\mathcal{F}^{\mathcal{S}})$ represents all the information available to agent $x$ about the preferences $\mathcal{F}^{\mathcal{S}}$ of all agents including itself.

Protocol $\mathcal{P}$: The negotiation protocol.

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

NEC-AIST AI Cooperative Research Laboratory

**Protocol**

**Breaking Rule** when the negotiation ends with a timeout.

**Action Constraints** **what** actions are valid.

**Activation Rule** **which** each agent is activated.

**Evaluation Rule** the next step:

- **Continue** ▷ Goto breaking rule.
- **Agreement** Succeed with $\omega$.
- **Disagreement** Fail with $\phi$.

**Agent (Strategy)**

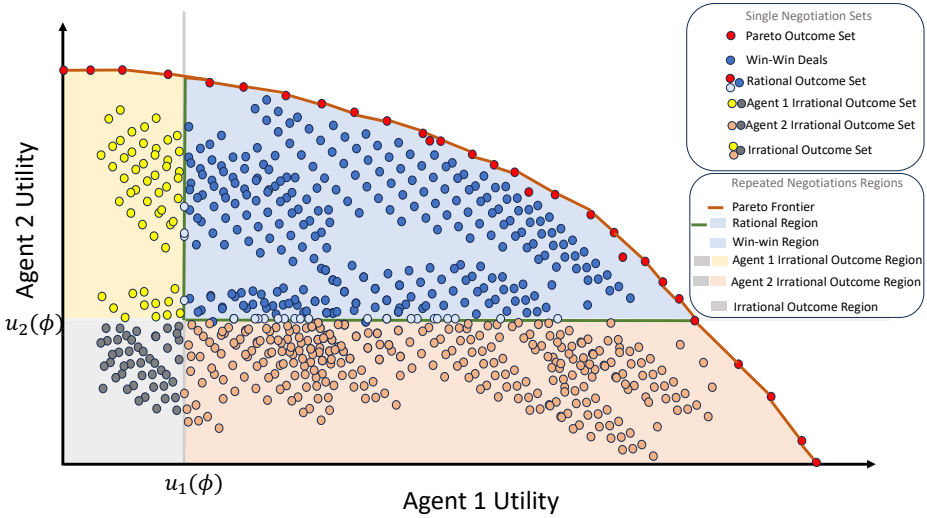
Selects an action from the **valid action set**.

**Algorithm 1** Negotiation Mechanism Execution: $\mathcal{P}_\Upsilon(\lambda)$

**Input:** Scenario $\lambda \equiv \left(\mathcal{N}^\lambda, \left(\Omega, \mathcal{F}^\lambda\right), \mathcal{I}^\lambda \equiv \left\{\mathrm{I}_x^\lambda : x \in \mathcal{N}^\lambda\right\}\right)$,
Mechanism $\mathcal{P}_\Upsilon = (\mathcal{P}, \Upsilon) : \mathcal{P} = (\zeta, \chi, \sigma)$
**Output:** $\omega_* \in \Omega \cup \{\phi\}$

$x \approx \pi_x \forall x \in \mathcal{N}^\lambda \leftarrow \Upsilon(\lambda)$ ▷ Assign strategies
$i, t, t_0, \mathcal{T} \leftarrow 0, 0, clock(), \langle\rangle$ ▷ Init step, time and trace
**while** $\zeta(\mathcal{T}, i, t; \lambda) = 0$ **do** ▷ check the breaking rule
  $\mathcal{N}_i^\lambda, \leftarrow \chi(\mathcal{T}, i; \lambda)$ ▷ Next active agents
  **for** $x \in \mathcal{N}_i^\lambda$ **do** ▷ Activate agents
    $\mathbb{A}_x^v \leftarrow \gamma(\mathcal{T}, x; \lambda)$ ▷ Valid action set
    $\alpha_i^x \leftarrow \pi_x\left(\mathcal{T}, \mathbb{A}_x^v; \mathrm{I}_x^\lambda\right)$ ▷ Receive action
    $\mathcal{T} \leftarrow \mathcal{T} + \langle \alpha_i^x, x \rangle$ ▷ Update the trace
    $z \leftarrow \sigma\left(\mathcal{T}, i; \lambda\right)$ ▷ Evaluation rule
    **if** $z = \phi$ **then return** $\phi$ ▷ End with disagreement
    **else if** $z \in \Omega$ **then return** $\omega_i$ ▷ agreement
    **end if** ▷ Otherwise continue to the next step
  **end for**
  $i, t \leftarrow i + 1, clock() - t_0$ ▷ Update step and time.
**end while**
**return** $\phi$ ▷ Disagreement on breaking rule returning 1

# Visualizing a negotiation



# Outline

# Automated Negotiating Agents Competition (ANAC)

NEC-AIST AI Cooperative Research Laboratory

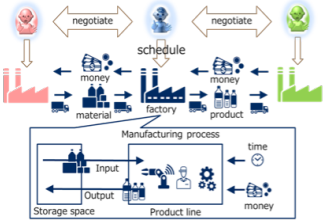
- An international competition dedicated to automated negotiation.
- Started in conjunction with AAMAS in 2010.
- Running every year with AAMAS or IJCAI (14 iterations so far).
- Next ANAC will be in conjunction with AAMAS 2024 (not official yet).

# Outline

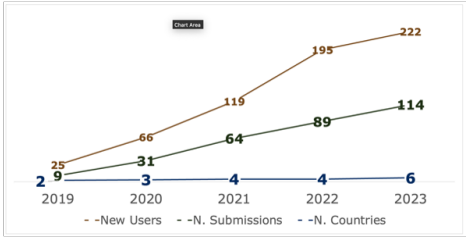
# Supply Chain Management League (SCML)

NEC-AIST AI Cooperative Research Laboratory

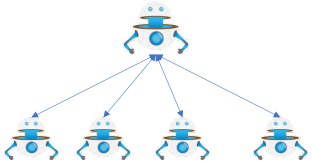
- Running as part of ANAC since 2019.
- **Scenario** A market in which all trade is done through concurrent closed bilateral negotiations:
  - Represents several applications in the real world.
- Agent preferences are endogenous to the simulation.
- **Goal** Maximize your profit.
- **Main Challenges**: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




| Rank | Agent | Earned money |
|---|---|---|
| | | |
| 2 | | |
| 3 | | |
| 4 | | |



# Concurrent Negotiation

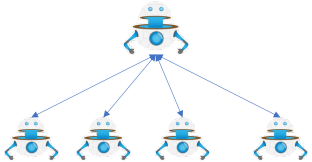


# Concurrent Negotiation

NEC-AIST AI Cooperative Research Laboratory

## Generality

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



# Concurrent Negotiation

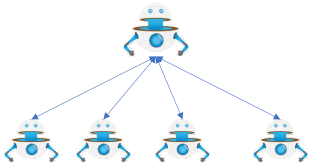
NEC-AIST AI Cooperative Research Laboratory

## Generality

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

## Decommitment

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



# Concurrent Negotiation

NEC-AIST AI Cooperative Research Laboratory

## Generality

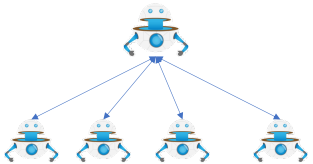
- Specific scenario (buyer-seller).
- General domain

## Decommitment

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

## Timing

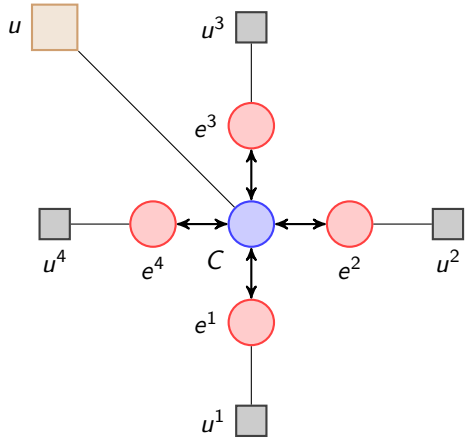
- Synchronous.
- Any-time.



# Concurrent Negotiations

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

The problem of how to negotiate in this setting is reminiscent of how to bid in simultaneous auctions, when bidders' utilities are combinatorial.[7]

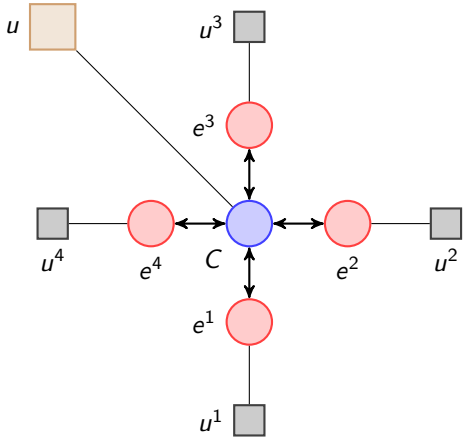



[7] Michael P. Wellman, Eric Sodomka, and Amy Greenwald. "Self-confirming price-prediction strategies for simultaneous one-shot auctions". In: *Games and Economic Behavior* 201 (2017), pp. 339–372.

# Concurrent Negotiations

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

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



# SCML World

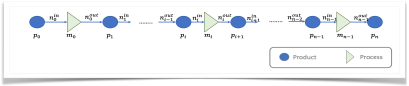
NEC-AIST AI Cooperative Research Laboratory

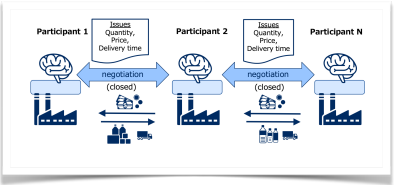
## Challenge

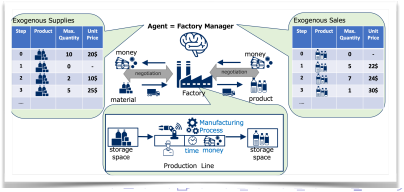
- Negotiation game with imperfect information
- Concurrent negotiations.
- Repeated negotiations → OneShot.
- Sequential negotiations → Standard.

## Information

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



# SCML Competition

NEC-AIST AI Cooperative Research Laboratory

## 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

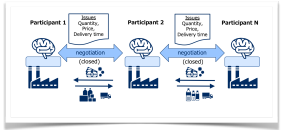
NEC-AIST
AI Cooperative
Research Laboratory

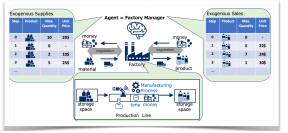
## Competition Details

- 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

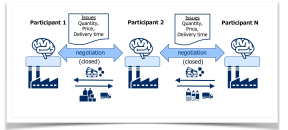
NEC-AIST AI Cooperative Research Laboratory

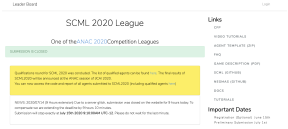
## Competition Details

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

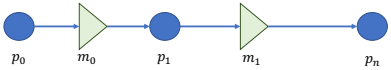
## Flavors

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

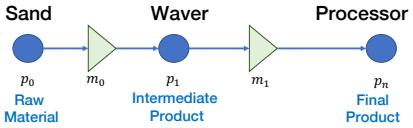



# Overview



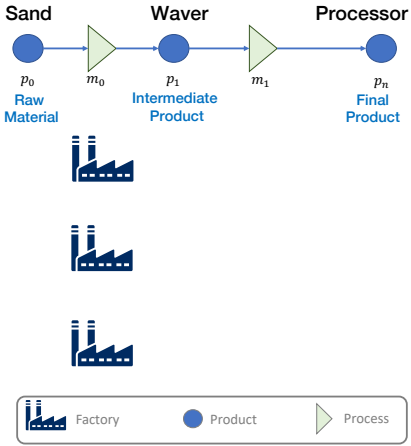

- A **production-graph** defines what can be produced and how.

# Overview



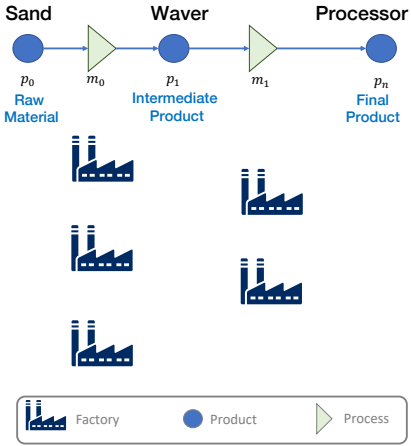

- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.

# Overview



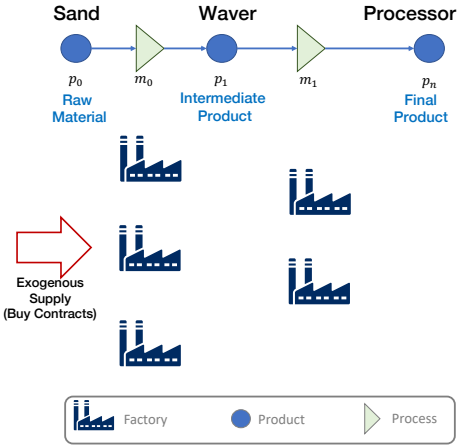

- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.

# Overview



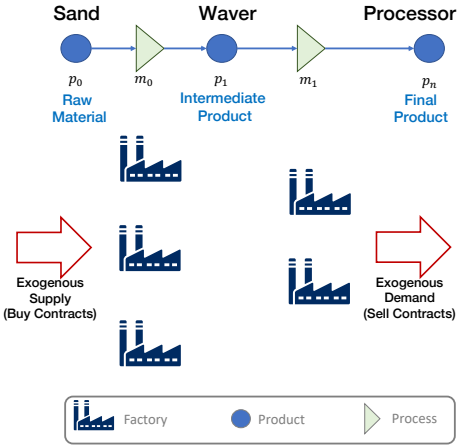

- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.
- We have two layers of factories/agents.

# Overview



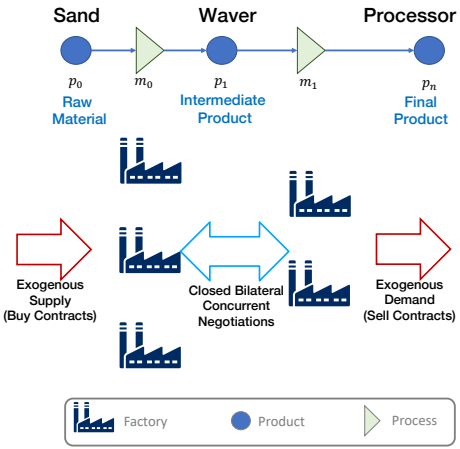

- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.
- We have two layers of factories/agents.
- $L_0$ factories/agents receive exogenous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

# Overview



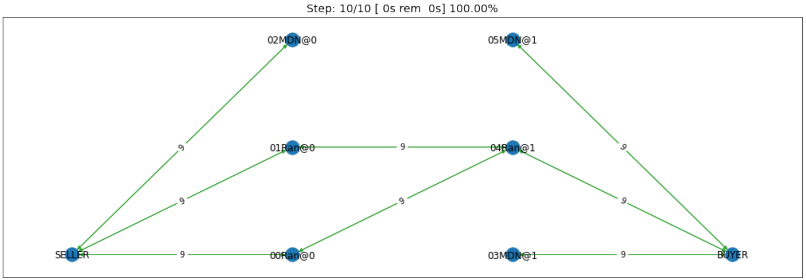

- A **production-graph** defines what can be produced and how.
- We have 3 products, 2 processes.
- Factories can run **manufacturing processes** converting input products into output products on their **production lines**.
- We have two layers of factories/agents.
- $L_0$ factories/agents receive exogenous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

# Overview




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

# SCML-OneShot Track




## Main Idea

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

# 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

NEC-AIST AI Cooperative Research Laboratory

## 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).

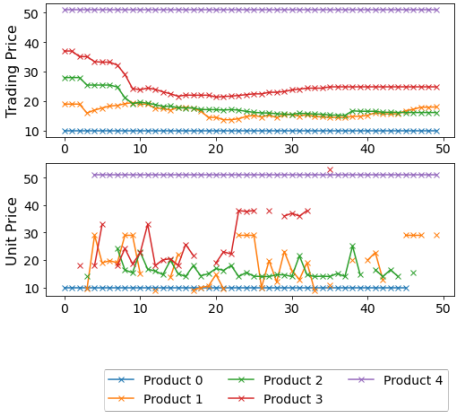
# Market Information

## 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

NEC-AIST AI Cooperative Research Laboratory

## Financial Reports

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

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

# Simulation Steps

Once

Initialize all agents

- init()

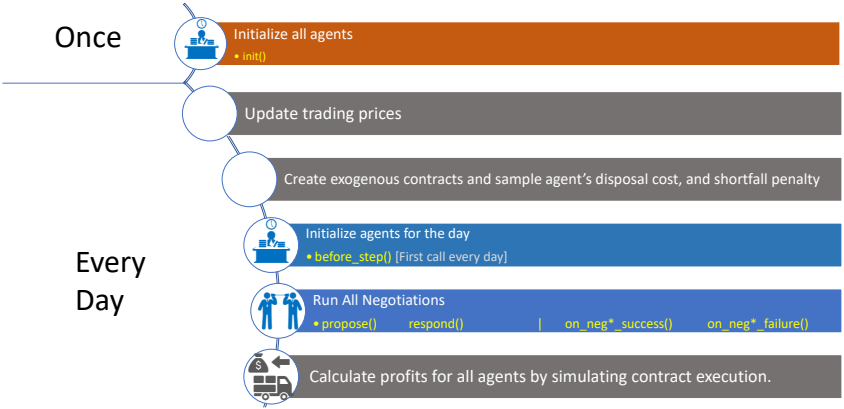
# Simulation Steps

# Simulation Steps

# Simulation Steps

# Simulation Steps

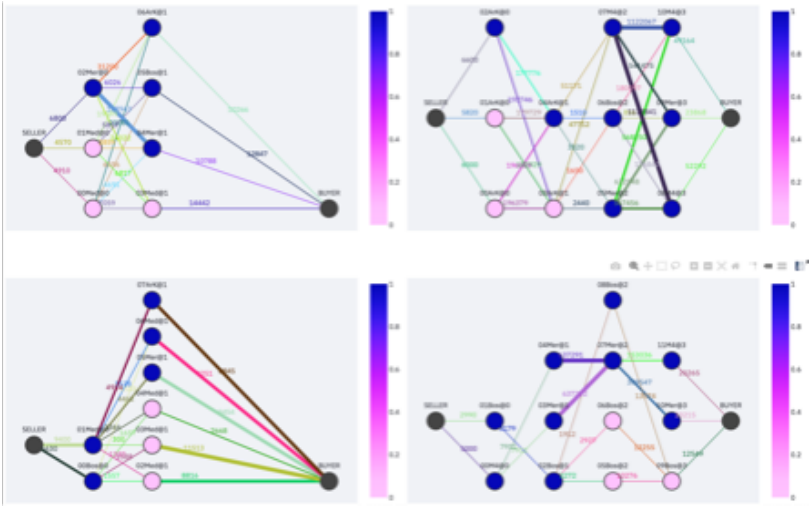
# Simulation Steps



# Simulation Steps

# Simulation Steps

# SCML Environment



# Outline

# Outline

# Installing SCML

NEC-AIST AI Cooperative Research Laboratory

## Pre-Installation

```
> python3 -m venv .venv
> source .venv/bin/activate
> .venv\\Scripts\\activate
```

## Basic Installation

```
> pip install scml
```

## Optional Installations

```
> pip install scml-agents
> pip install scml-vis
```

# Outline

# Running a single simulation (oneshot)

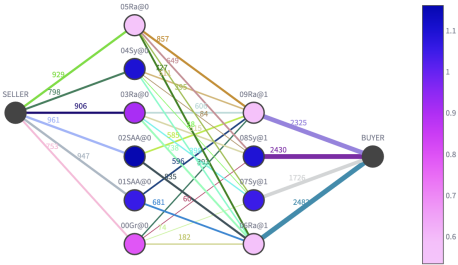
NEC-AIST AI Cooperative Research Laboratory

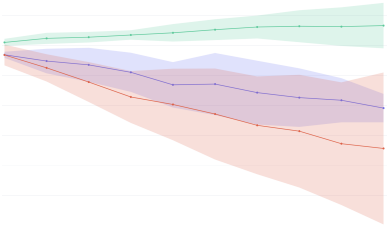
## Command Line Tool

```
> scml run2024 --oneshot
```

## Python

```
SCML2024OneShotWorld(
**SCML2024OneShotWorld.generate(
agent_types=DefaultAgentsOneShot2024
)).run()
```



# Running a single simulation (std)

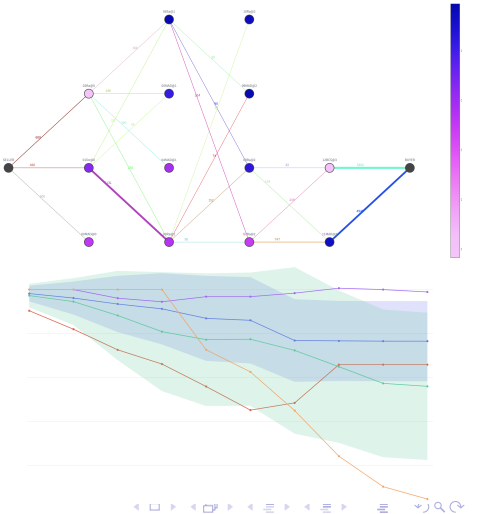
NEC-AIST
AI Cooperative
Research Laboratory

## Command Line Tool

```
> scml run2024
```

## Python

```
SCML2024StdWorld(
**SCML2024StdWorld.generate(
agent_types=DefaultAgents2024
)).run()
```



# Outline

# Running a complete tournament (oneshot)

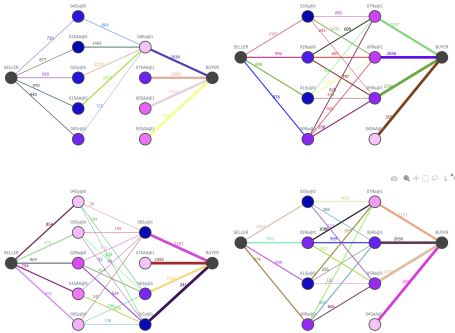
NEC-AIST
AI Cooperative
Research Laboratory

## Command Line Tool

```
> scml tournament2024 --type=oneshot
```

## Python

```
anac2024_oneshot(
    competitors=[
        RandomOneShotAgent,
        SyncRandomOneShotAgent,
        GreedyOneShotAgent
])
```



# Running a complete tournament (std)

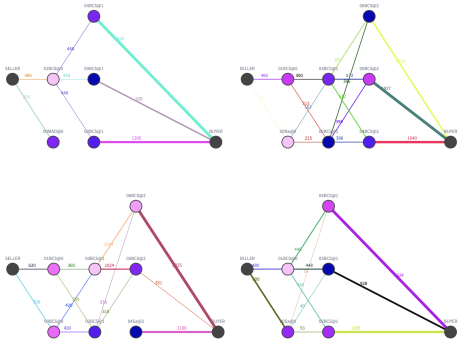
NEC-AIST
AI Cooperative
Research Laboratory

## Command Line Tool

```
> scml tournament2024 --type=std
```

## Python

```
anac2024_std(
    competitors=[
        RandomOneShotAgent,
        SyncRandomOneShotAgent,
        GreedyOneShotAgent
])
```



# Outline

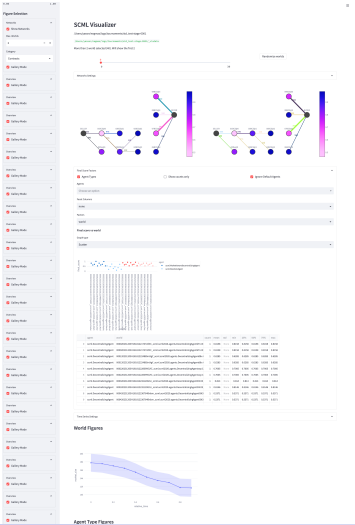
# Visualization CLI

NEC-AIST AI Cooperative Research Laboratory

## Command Line Tool

```
> scmlv show
```

```
20231129H194614733893Ps7-stage-  2.2 K   0000S0_0.n_A-RCFR.0008a55a
20231129H194641342854yLc-stage-  2.2 K   0001S0_1.H_A-R4oJ.016ed8bf
20231129H194846359611Uby-stage-00 1 K   0002S0_2.B_A-RUup.824aa54
20231129H194955076256M84-stage-  2.2 K   0003S0_3.5_A-Rw8m.83165be0
20231129H195118577752sfJ-stage-  2.7 K   0004S1_0.q_A-RNHn.8499dc63
20231129H195216244186qsC-stage-  2.7 K   0005S1_1.p_A-R3ND.05a5a8bb
20231129H194852629346O3k-stage-00 1 K   0006S1_2.A_A-RnFK.868292d7
20231130H880023219868Phm-stage-  960 B   0007S1_3.l_A-RSG8.877b18fc
20231130H075912052957qr6-stage-  1.9 K   agent_stats.csv
20231130H075932117366XNX-stage-  1.1 K   agg_stats.csv
.DS_Store                        14 K    assigned_configs.json
                                         assigned_configs.pickle
                                         attempts
                                         base_configs.json
                                         configs
                                         kstats.csv
                                         params.json
                                         partner_advantage.csv
                                         partner_utility.csv
                                         received_advantage.csv
                                         received_utility.csv
                                         score_stats.csv
                                         scores.csv
                                         stats.csv
                                         total_advantage.csv
                                         total_scores.csv
                                         tstats.csv
                                         type_stats.csv
```



# Outline

# Troubleshooting and breathing time

# Outline

# Outline

# SCML-OneShot Agent: Core

NEC-AIST
AI Cooperative
Research Laboratory

```python
class OneShotSyncAgent(SAOSyncController, OneShotAgent, ABC):
    """
    An agent that automatically accumulate offers from opponents and allows
    you to control all negotiations centrally in the `counter_all` method.

    Args:
+---  4 lines: owner: The adapter owning the agent. You do not need to directly deal···

    """

    def __init__(self, *args, **kwargs):
+---  2 lines: kwargs["global_ufun"] = True·····································
        |
    @abstractmethod
    def counter_all(
        self, offers: dict[str, Outcome], states: dict[str, SAOState]
    ) -> dict[str, SAOResponse]:
+--- 20 lines: """Calculate a response to all offers from all negotiators·····
        |
    @abstractmethod
    def first_proposals(self) -> dict[str, Outcome]:
+--- 10 lines: """·····
```

# SCML-OneShot Agent: Optional Timing Callbacks



```
def init(self):
--- 6 lines: """
def before_step(self):
--- 8 lines: """
def step(self):
--- 8 lines: """
```

# SCML-OneShot Agent: Optional Events

```
    def on_negotiation_failure(
---   5 lines: self,·····
    ) -> None:
--- 12 lines: """·····
    |
    def on_negotiation_success(
    |   self, contract: Contract, mechanism: NegotiatorMechanismInterface
    ) -> None:
---   8 lines: """·····
    |
```

# Outline

# Greedy Random Agent


- Link to builtin agent examples (OneShot)
- Link to builtin agent examples (Std)

# Outline

# Example Winner Agents


- Link to 2023 winner agent descriptions
- Link to source code and description of every agent