

# 機械学習技術を用いた交渉エージェントのための 自動交渉シミュレータ Jupiter の提案

## A proposal of automatic negotiation simulator Jupiter for negotiation agents using machine learning.

福井智哉 \*<sup>1</sup>  
Tomoya Fukui

伊藤孝行 \*<sup>1</sup>  
Takayuki Ito

\*<sup>1</sup>名古屋工業大学大学院 情報工学専攻  
Nagoya Institute of Technology, Department of Computer Science

The purpose of this paper is to propose Jupiter, a new environment for automated negotiations in which we can easily create agents that is able to use machine learning. Genius is cited as a prior study of the environment where automated negotiation can be simulated. It provides an environment for automated negotiations that aim to solve multi-issue negotiation problems. In the field of automated negotiation, it is expected that agreement results are optimized by machine learning. However, it is difficult to use Genius to simulate automated negotiations with agents using machine learning, because the past negotiations information provided by Genius is insufficient. As above, in this paper we propose Jupiter as a new automated negotiation environment in which we can easily create agents that is able to use machine learning. In addition, we compare Jupiter with Genius and show the superiority of Jupiter.

## 1. はじめに

本論文の目的は、機械学習技術を用いたエージェントの作成が容易な交渉シミュレーション環境として Jupiter を提案することである。

交渉は互いの利害の調整のために、社会において必要不可欠な行為である。自動交渉を模擬的に行う環境の先行研究として、Genius が挙げられる。Genius は複数論点交渉問題における、エージェントによる自動交渉シミュレーション環境を提供し、自動交渉の合意結果から自動交渉エージェントの性能を定量的に測ることができる。

近年、ハードウェア技術の向上に伴い、機械学習技術の発展が目覚ましい。しかし、Genius では繰り返し交渉を行う際に、過去の交渉に関する情報が一部分しか取得できないため、機械学習技術を用いる際に必要となる学習データが不足しており、機械学習技術を応用したエージェントの自動交渉実験が困難である。以上の点から、本論文では機械学習技術を用いたエージェントの作成が容易な環境として Jupiter を提案する。また提案システムである Jupiter と Genius で比較と評価実験を行うことにより、Jupiter の有用性を示す。

## 2. 関連研究

### 2.1 ANAC 概要

本論文で提案する Jupiter では、既存研究 ANAC[Baarslag 2016] を参考に、扱うことのできる交渉プロトコルと交渉ドメインを設定した。交渉プロトコルとしては Stacked Alternating Offers Protocol(SAOP)[Reyhan 2017] を採用している。交渉プロトコルとは、交渉の進め方や各エージェントの行動の定義をまとめた概念である。SAOP では、交渉参加エージェントが、交互に合意候補案の中から 1 つ提案を行うことにより、交渉を進行させる。交渉時間の種類にはターン制と時間制の 2 種類が存在し、各エージェントの行動には Offer, Accept および EndNegotiation の 3 種類が存在する。

本節では Jupiter で取り扱う交渉プロトコルと交渉ドメインについて述べる。

#### 交渉時間の種類

交渉の制限時間についてターン制と時間制の 2 種類述べる。

#### 1. ターン制

全参加エージェントの合意または EndNegotiation が発生したとき、もしくはあらかじめ決められたターンまで合意が得られなかったときまで交渉を続ける。

#### 2. 時間制

全参加エージェントの合意または EndNegotiation が発生したとき、もしくはあらかじめ決められた制限時間まで合意が得られなかったときまで交渉を続ける。

### 各エージェントの行動

各エージェントが取ることの出来る行動について述べる。エージェントの行動には Offer, Accept 及び EndNegotiation の 3 種類が存在する。各交渉参加エージェントは、自分の手番が回ってくる度に、3 種類の行動から 1 つ選ぶ。また、交渉参加エージェントは、交渉参加エージェントの順番に関係なく、他エージェントの行動を公開情報として全て認知することができる。しかし、行動を起こすには、自分の手番が回ってくるまで待たなければならない。

#### 1. Offer

合意案候補から 1 つ選び、提案する。他エージェントからの Offer をされたあとに、自エージェントが Offer を行なった場合、その他のエージェントの Offer は自動的に却下されたことになる。

#### 2. Accept

最近の他エージェントからの Offer を受け入れることを意味する。あるエージェントの Offer を、提案したエージェント以外の全エージェントが連続で Accept した場合、その交渉は合意に至ったとして判定され、交渉が終了する。

#### 3. EndNegotiation

交渉の強制的な終了を意味する。他エージェントの行動に関わらず、交渉を終了することができる。一般的に他エージェントが交渉において、割引効用が大きいにもかかわらず、まったく妥協しない場合に有用な選択肢となる。

### 交渉ドメイン

ANAC で取り扱われる交渉ドメインについて述べる。交渉ドメインには、交渉結果における、各エージェントが取得する効用値に関する情報が記されている。

交渉ドメインに記述されている内容を説明する前に、合意案候補 (bid) を数式で示す。s を合意案候補、n を論点の数、 $I_i$

連絡先: 福井智哉, 名古屋工業大学工学部情報工学科,  
fukui.tomoya@itolab.nitech.ac.jp

を  $i$  番目の論点で取りうる選択肢の集合と定義すると、各合意案候補 ( $bid$ ) はの式 (1) により定義される。

$$\mathbf{s} = \{(s_1, s_2, \dots, s_n) | s_i \in I_i, i = 1, 2, \dots, n\} \quad (1)$$

式 (1) をふまえて、交渉ドメインに記述されている各合意案候補の効用値、留保価格及び割引効用の 3 つを説明する。

1. 各合意案候補の効用値  
各合意案候補の効用値  $U(\mathbf{s}; \mathbf{w})$  は式 (2)、式 (3) 及び式 (4) を用いて表される。

$$U(\mathbf{s}; \mathbf{w}) = \sum_{i=1}^n w_i \cdot eval(s_i) \quad (2)$$

$$\sum_{i=1}^n w_i = 1 \quad (3)$$

$$eval(s_i) = \frac{value(s_i)}{\arg \max_{s_i \in I_i} value(s_i)} \quad (4)$$

$w_i$  は  $i$  番目の論点における重みを表し、 $0 \leq w_i \leq 1 (1 \leq i \leq n)$  である。 $value(s_i)$  は  $I_i$  において  $s_i$  を選んだ場合の効用値を表す。 $w_i (1 \leq i \leq n)$  と  $value(s_i) (s_i \in I_i)$  の値は交渉ドメインに記されている。補足として、式 (4) から

$$0 \leq eval(s_i) \leq 1 \quad (5)$$

である。式 (2)、式 (3) および式 (5) より

$$0 \leq U(\mathbf{s}; \mathbf{w}) \leq 1 (\forall \mathbf{s})$$

である。

2. 留保価格  
EndNegotiation や合意が得られずに交渉が終了した際に得られる効用値を表す。留保価格を  $r$ 、合意失敗を  $e$  とすると、式 (6) で定義される。

$$U(e) = r \quad (6)$$

3. 割引効用  
交渉が終わった際に、交渉にかかった時間に応じて、各エージェントが得られる効用値を割り引くために使われる値である。各エージェントが得られる割引済みの効用値を  $U_d$ 、割引効用を  $d$ 、正規化された交渉の経過時間を  $t (0 \leq t \leq 1)$  とすると

$$U_d(\mathbf{s}, t; d) = U(\mathbf{s}) \cdot d^t \quad (7)$$

$$U_d(e, t; d) = U(e) \cdot d^t \quad (8)$$

と表される。

## 2.2 Genius 概要

既存の自動交渉シミュレーション環境に関する先行研究として、Genius [Lin 2014] について述べる。汎用的な自動交渉エージェントに関する研究を支援し、Java API を用いて交渉エージェントの開発や自動交渉シミュレーションを行うことができる。Genius の主な機能は、以下の 3 つである。

1. 交渉問題を定義する交渉ドメインファイルの作成
2. 自動交渉エージェントによる交渉シミュレーション
3. 交渉の過程と結果の記録と解析

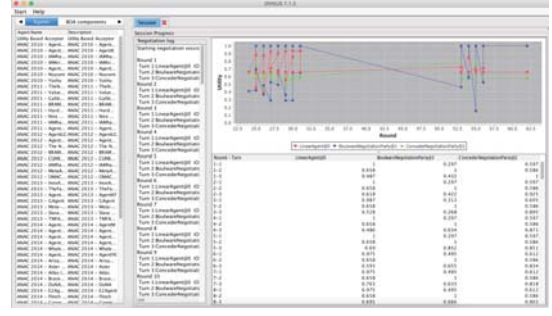


図 1: Genius の実行例

Genius はパレートフロント、ナッシュ解、およびカライ・スモロデンスキー解といった合意案候補点をグラフ上に描画するため、自動交渉の結果を視覚的に分析することができる。また特定の交渉問題ではなく、本章で説明した交渉プロトコルを採用することにより、汎用的な自動交渉全般に関する研究を支援している。よって Genius は、異なる交渉戦略をもつエージェント同士を、客観的な指標を用いて評価することが可能であり、自動交渉エージェント及び交渉戦略の研究において重要な役割を果たしている。Genius の実行例として、図 1 を添付しておく。

### Genius の問題点

Genius を用いた自動交渉エージェントの研究開発では、さまざまな分野の技術が用いられている。森ら [森 2015] の提案するエージェントでは、ゲーム理論の観点からエージェントを作成している。Williams の提案するエージェント [Williams 2011] では、ガウス過程という機械学習技術を用いたエージェントを作成している。

一方、昨今のエージェントの研究分野では、深層学習技術を用いた研究が注目されている。2017 年には Lewis らが自然言語により自動交渉を行うエージェントの研究 [Lewis 2017] を行なっている。今後も深層学習などの機械学習技術を用いたエージェントの研究が期待される。しかし、Genius は Java 実装である点や、2010 年より開発されているという経緯から、昨今の深層学習技術を用いたエージェントの開発に向けて作成されていない。一般的に深層学習技術を実装する場合には、Python で実装されることが多い。

## 3. 自動交渉シミュレーション環境 Jupiter

### 3.1 Jupiter の概要

Jupiter は本論文で提案する自動交渉シミュレーション環境である。Jupiter は 2.1 章で述べた交渉プロトコルと交渉問題を扱うことができる。またプロセス間通信を行うことにより、Genius 上で動作する自動交渉エージェントを Jupiter 上で動かすことができる。本章では、Jupiter の実行方法と動作例を示す。

### 3.2 Jupiter で自動交渉を行うまでの流れ

本章では、本論文で提案する自動交渉シミュレーション環境 Jupiter により、自動交渉を行う際の処理の流れを説明する。

1. ユーザが交渉の長さ、交渉ドメインファイルおよび交渉参加エージェントを指定する。
2. Jupiter が交渉ドメインファイルを読み込む。またエージェントを呼び出し、エージェントに交渉ルール、交渉ドメイン情報を渡す。
3. 交渉を実行する。

また交渉を実行する際のパラメータ次第で、交渉過程を動的に表示させることや交渉履歴を json 形式で保存することができる。

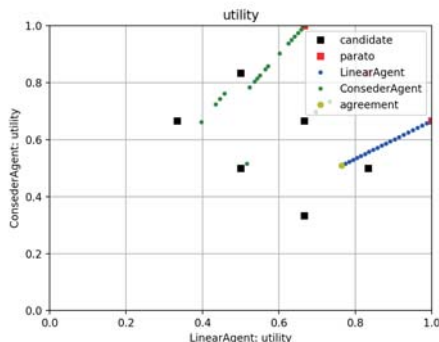


図 2: Jupiter の実行例

## abstractAgent module

```
class abstractAgent.AbstractAgent(utility_space: abstractUtilitySpace.AbstractUtilitySpace, negotiation_rule:
negotiationRule.NegotiationRule, agent_id: int, agent_num: int)
    ベースクラス: object

    自動交渉エージェントの抽象クラス 自動交渉エージェントを作成する際はこのクラスを継承すること

    パラメータ:
    • utility_space (AbstractUtilitySpace) – 効用空間の情報が取得できる
    • negotiation_rule (NegotiationRule) – 交渉の時間やタイプ、現在の正規化時間が取得できる
    • agent_id (int) – 自分のエージェントに割り振られたid
    • agent_num (int) – 交渉参加エージェントの数

    receive_action(agent_action: agentAction.AbstractAction)
        他エージェントが行動を起こした場合に、その行動が通知される

    パラメータ: agent_action (AbstractAction) – 他エージェントが起こした行動

    send_action() → agentAction.AbstractAction
        自分のターンが回ってきた際に呼び出され、どの行動を起こすか返す

    戻り値の型: AbstractAction
    戻り値: Accept, Offer, EndNegotiationのいずれかを返す

    get_name() → str
        自分エージェントの名前を返す。クラス名と同じにすることを推奨

    戻り値の型: str
    戻り値: エージェントの名前。クラス名と同じにすることを推奨

    receive_start_negotiation()
        提案応答ゲームを行う際に、提案応答ゲームが開始される際に呼び出される

    receive_end_negotiation()
        提案応答ゲームを行う際に、提案応答ゲームが終了される際に呼び出される
```

図 3: abstractAgent

Jupiter の実行例として、図 2 を載せる。緑色の点が ConcederAgent の提案した合意案候補であり、青色の点が Linear エージェントの提案した合意案候補である。横軸が Linear エージェントの獲得できる効用値であり、縦軸が Conceder エージェントの獲得できる効用値である。また黒色の四角形と赤色の四角形が合意案候補点である。

## 3.3 Python による自動交渉エージェントの作成

Jupiter でエージェントを実装するには、図 3 に表示されている、自動交渉エージェントの抽象クラスである abstractAgent クラスを継承したクラスを作成すればよい。継承した子クラスについて、親クラスの各メソッドをオーバーライドすることにより、エージェントの交渉戦略を定義できる。

ここでエージェントを実装する際の、Genius との差異として、receive.start\_negotiation メソッドと receive.end\_negotiation メソッドがあることを挙げておく。繰り返し提案応答ゲームにおいて機械学習技術に応用する際に、提案応答ゲームと提案応答ゲームの間に、パラメータの調整などの処理を行いたい場合がある。receive.start\_negotiation メソッドは提案応答ゲームの開始時に、receive.end\_negotiation メソッドは終了時に呼ばれる。そのため、提案応答ゲームが終了した際の後処理を実装したい場合、Jupiter では容易に行うことができる。

表 1: Genius と Jupiter によって実装された Linear エージェントの比較

| シミュレーション環境 | コードの行数 | コードの文字数 |
|------------|--------|---------|
| Genius     | 57     | 120     |
| Jupiter    | 36     | 83      |

## 4. Jupiter と Genius の比較

## 4.1 概要

本章では Jupiter と Genius について、定性的な評価と定量的な評価を行うことにより、その差異を示す。

まず Genius 実装のエージェントと、Jupiter 実装のエージェントについて、そのコードの長さを比較する。次に Genius によって行われる自動交渉と、Jupiter によって行われる自動交渉で、実行にかかる時間を比較する。実行にかかる時間が短ければ、自動交渉エージェントの実験にかかる時間を削減することができる。よって、自動交渉シミュレータにとって重要な課題である。次に定性的な評価について比較を行う。最後に、本章で行なった比較のまとめと考察を行う。

## 4.2 自動交渉エージェントのコードの長さの比較

本実験では、Linear エージェントを Jupiter 環境と Genius 環境で動くようにそれぞれ実装し、そのコードの行数と文字数を定量的に比較する。以下、Linear エージェントについて解説する。

エージェントを作成する際には、交渉戦略を設計する必要がある。自動交渉エージェントはその交渉戦略に基づいて受容 (Accept)、提案 (Offer) 及び交渉打ち切り (EndNegotiation) から 1 つ選択する。特に他エージェントからの提案を受容するかどうか、に関する戦略を Acceptance Strategy と呼ぶ。Acceptance Strategy を設計する際に、自身が獲得できる効用値からその判断を行うエージェントが多く存在する。その判断の手法には、提案された内容の効用値を入力として、受容するかどうかの 2 値を出力する、譲歩関数を用いる手法が存在する。文献 [C.Sierra 1999] では、譲歩関数に交渉の経過時間  $t(0 \leq t \leq 1)$  を用いたエージェントを提案している。

$$T(t) = 1.0 - t^\alpha \quad (9)$$

具体的には式 (9) で定義される譲歩関数  $T(t)$  を提案している。式 (9) で表される譲歩関数を持ち、特に  $\alpha = 1$  となるような譲歩関数を持つエージェントを Linear エージェントと呼ぶことにする。Linear エージェントは、譲歩関数の出力する値以上の効用値を持つ提案または受容を行う。

比較の結果を表 1 にまとめた。一般的に Python 言語は Java 言語よりも、コードを簡潔に記述できるとされている。本比較の結果においても、定量的なコードの記述量による比較から、コードの簡潔さを示すことができた。

## 4.3 実行速度の比較

本実験では Linear エージェント、Conceder エージェント及び Boulware エージェントによる 3 者間交渉を行う。Conceder エージェントは式 (9) において  $\alpha = 0.5$  となるような譲歩関数を持つエージェントである。Boulware エージェントは式 (9) において  $\alpha = 10$  となるような譲歩関数を持つエージェントである。

交渉の設定としてはターン制で、180 ターンを上限とした。また、交渉ドメインファイルとして、ANAC2016 で用いられた triangularFight, KDomain 及び NewDomain を用いる。交渉ドメインファイルを選んだ基準として、合意案候補の数を考えており、合意案候補の数が少ないもの、中くらいのもの及び多いものを選んだ。それぞれの効用空間の大きさは表 2 を参照されたい。

交渉参加エージェントの順列が  ${}_3P_3 = 6$  通りであり、それぞれ 10 回ずつ自動交渉を行うため、各々の効用情報について合計  $6 \times 10 = 60$  回ずつ自動交渉を行う。

表 2: 実験で用いる効用空間の大きさ

| ドメイン名           | 論点数 | 合意案候補の数 |
|-----------------|-----|---------|
| triangularFight | 2   | 9       |
| KDomain         | 5   | 1280    |
| NewDomain       | 7   | 35840   |

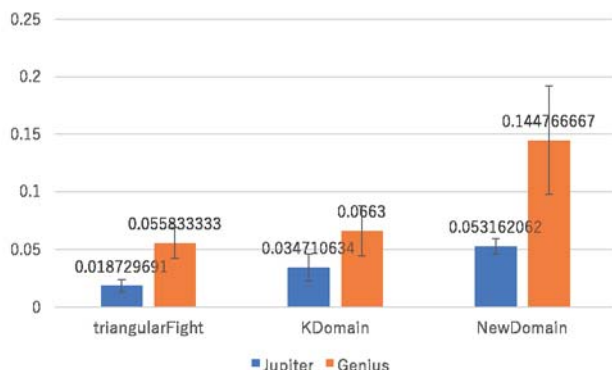


図 4: 平均交渉時間とその分散

各交渉ドメインファイルにおける 60 回の、平均交渉時間とその分散を示したものが図 4 である。効用空間の大きさに関わらず、Jupiter の方が自動交渉にかかる時間が短いことが示された。Jupiter では一部の処理を Cython で実装することにより、処理の高速化を行なっている。

#### 4.4 定性的な評価による比較

本節では定性的な評価により比較を行う。

Genius ではパレートフロント、ナッシュ解、及びカライ・スモロデンスキー解などの分析を自動で行う。Jupiter では現在、パレートフロントのみを提供している。ナッシュ解及びカライ・スモロデンスキー解の分析については実装中である。

Genius は Java 言語による実装であり、Jupiter は Python 言語による実装である。そのため、Jupiter は scikit-learn や tensorflow などのライブラリを用いることにより、機械学習技術を応用したエージェントの実装が容易である。

また Genius では GUI 上で自動交渉が行えるが、Jupiter では基本的に Jupyter Notebook というシェル上での実行を推奨している。Jupyter Notebook 上での実行が可能のため、実行ログの視覚的な確認が容易であり、エージェントの各種パラメータの調整が容易である。

#### 4.5 比較のまとめ

以下、本章で行った比較評価に関して、まとめる。簡単にまとめたのが表 3 である。

Genius と提案する自動交渉シミュレーション環境 Jupiter について比較評価を行った。定量的な比較項目として、交渉の実行時間と、エージェントを実装する際のコードの記述量を比較した。定性的な評価として、実装言語や使用方法を比較した。

#### 4.6 考察

考察をもって、本章の終わりとする。

Python 言語はインタラクティブな実行が可能である、スクリプト言語である。よって、一般的には Java 言語の方が実行速度は速いと言われている。しかし、Genius は 2010 年から現在に到るまで、長期間運用されてきた自動交渉シミュレータである。内部が最適化されていないために、動作が遅いと考えている。一方で、長期間運用されてきたことからその研究分野はゲーム理論、心理学及びマルチエージェントなど多岐に渡る。例えば 4.4 節で述べた、ゲーム理論の枠組みによる、合意結果の分析は大変興味深い。また ANAC で毎年用いられていることから、留保価格や割引効用が取り入れられるようになり、より現実の交渉問題に近い交渉設定を扱えるようになって来ている。しかし現状ではまだ現実の交渉問題を扱うことは難しいと

表 3: Genius と Jupiter の比較まとめ

| 比較項目          | Genius | Jupiter |
|---------------|--------|---------|
| 実装言語          | Java   | Python3 |
| CUI           | △      | ○       |
| 交渉の実行時間       | △      | ○       |
| エージェントのコード記述量 | △      | ○       |

感じている。よって、交渉プロトコルや交渉ドメインについての研究を今後の課題としたい。

## 5. おわりに

本研究では、自動交渉シミュレーション環境 Jupiter を提案した。SAOP に基づく自動交渉シミュレーションを Python 言語により行うことができる。また実験により、自動交渉にかかる時間とエージェントのコード記述量において、先行研究である Genius よりも短いことを示した。しかし、Genius と比較して、交渉ドメインファイルの作成機能や GUI の実装など未実装である機能が存在することが問題点として挙げられる。今後の課題としていきたい。

## 参考文献

- [Williams 2011] Williams, Colin R., et al.: "Using gaussian processes to optimise concession in complex negotiations against unknown opponents.", IJCAI Proceedings-International Joint Conference on Artificial Intelligence. Vol. 22. No. 1. (2011).
- [Reyhan 2017] Aydoan, Reyhan, et al.: "Alternating offers protocols for multilateral negotiation.", Modern Approaches to Agent-based Complex Automated Negotiation. Springer International Publishing, 153-167. (2017).
- [Lin 2014] Lin, Raz, et al.: "Genius: An integrated environment for supporting the design of generic automated negotiators.", Computational Intelligence 30.1, 48-70. (2014).
- [森 2015] 森頭之, and 伊藤孝行.: "二者間交渉問題における進化的安定戦略を考慮した交渉戦略の提案.", 研究報告知能システム (ICS) 2015.19, 1-8. (2015).
- [Baarslag 2016] T. Baarslag, R. Aydog an, K. V. Hindriks, K. Fujita, T. Ito, and C. M. Jonker.: The automated negotiating agents competition 2010-2015., AI Magazine(2016).
- [Lewis 2017] Lewis, Mike, et al. : "Deal or no deal? end-to-end learning for negotiation dialogues." arXiv preprint arXiv:1706.05125 (2017).
- [C.Sierra 1999] Sierra, Carles, Peyman Faratin, and Nick R. Jennings.: "A service-oriented negotiation model between autonomous agents." Collaboration between human and artificial societies. Springer, Berlin, Heidelberg, 201-219, (1999) .