Jupiter User Guide 1.0.2

公開日　平成30年3月16日

氏名　福井 智哉

概要

Jupiter は自動交渉シミュレーション環境である．

自動交渉という研究分野が存在し，またそのシミュレーション環境として Genius[2] が挙げられる．しかし，Genius はその設計思想や開発時期の点から，昨今研究がさかんである機械学習技術の応用が難しい．そこで，Jupiter という自動交渉シミュレーション環境を構築した．Jupiter では自動交渉のプロトコルとして Stacked Alternating Offers Protocol(SAOP)[1] に基づくエージェントによる自動交渉のシミュレーション環境を提供する．Jupiter の特徴の 1 つとして，先行研究である Genius で開発されたエージェントや交渉ドメインファイルを，用いることを可能としている．

本ユーザーガイドでは，Jupiter の簡単な使い方を説明していく．1 章では，Jupiter で取り扱う自動交渉のプロトコルや交渉ドメインの設定について述べる．2 章では，Jupiter を実行するための環境と，その実行方法について述べる．3 章では，最後にまとめを述べる．

目 次

図 目 次

第1章

Jupiterで取り扱う交渉プロトコルと交渉ドメイン

1.1 序言

Jupiter は，既存研究 [3] を参考に，扱うことのできる交渉環境を設定した．

1.2 節では，交渉プロトコルについて述べる．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．Jupiter では交渉プロトコルとして，Stacked Alternating Offers Protocol(SAOP)[1] を採用している．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 及び EndNegotiation の 3 種類存在する．

1.3 節において，Jupiter で取り扱う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，各交渉エージェントの獲得する効用値について述べる．

1.2 交渉プロトコル

1.2.1 交渉時間の種類

本節では，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かったときまで交渉を続ける．例として，制限時間が 180 ターンで，参加エージェント数が 2 の交渉設定の場合における交渉の終了条件を述べる．交渉過程で合意または EndNegotiation が発生したとき，もしくは 2 つのエージェントがそれぞれ 180 回行動を行なったときが終了条件である．

2. 時間制
   現実時間が交渉の制限時間となる．例として，制限時間が 180 秒で，参加エージェント数が 2 の場合の交渉終了条件について述べる．交渉過程で合意または EndNegotiation が発生したとき，もしくは 2 つのエージェントの行動回数に関係なく現実時間で 180 秒が経過したときに，交渉が終了する．

1.2.2 各エージェントの行動

本節では，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，この 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．合意に至った場合の，各エージェントが取得する効用値については，1.3 節で述べる．

3. EndNegotiation
   交渉の強制的な終了を意味する．他の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．割引効用や，EndNegotiation となった場合における各エージェントが取得する効用値については，1.3 節で述べる．

1.3　交渉ドメイン

1.3.1　交渉ドメインの具体的な中身について

本節では，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．$\boldsymbol{s}$ を合意案候補，n を論点の数，I_i 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 (1.1) により定義される．

$$\boldsymbol{s} = \{(s_1, s_2, \ldots, s_n) | s_i \in I_i, i = 1, 2, \ldots, n\} \tag{1.1}$$

式 (1.1) 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 $U(\boldsymbol{s_k}; \boldsymbol{w})$ は式 (1.2)，式 (1.3) 及び式 (1.4) を用いて表される．

$$U(\boldsymbol{s}; \boldsymbol{w}) = \sum_{i=1}^{n} w_i \cdot eval(s_i) \tag{1.2}$$

$$\sum_{i=1}^{n} w_i = 1 \tag{1.3}$$

$$eval(s_i) = \frac{value(s_i)}{\arg\max_{s_i \in I_i} value(s_i)} \tag{1.4}$$

   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 (1.4) から

$$0 \leq eval(s_i) \leq 1 \tag{1.5}$$

である．式 (1.2)，式 (1.3) 及び式 (1.5) より

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

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 (1.6) で定義される．

$$U(e) = r \tag{1.6}$$

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

$$U_d(\boldsymbol{s}, t; d) = U(\boldsymbol{s}) \cdot d^t \tag{1.7}$$
$$U_d(e, t; d) = U(e) \cdot d^t \tag{1.8}$$

と表される．割引効用が $0.2, 0.4, \ldots, 1.0$ のときの d^t を図 1.1 に示す．

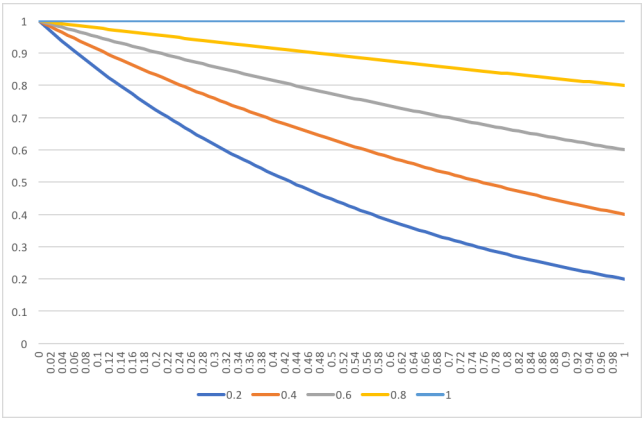

図 1.1: 時間経過と割引効用によるの獲得効用の変化

1.3.2　効用値の計算例

ここまでで交渉ドメインの説明をしてきたが，ここで実際に獲得効用 $U(\boldsymbol{s})$ の計算例を示す．具体的には図 1.2 で表される交渉ドメインにおける計算例を示す．交渉ドメインを式にすると $n = 2$，$|I_1| = 2$，$|I_2| = 2$，$w_1 = 0.3$，$w_2 = 0.7$ であり，各論点におけるそれぞれの選択肢の効用値が以下となる．

$$
\begin{aligned}
value(s_{11}) &= 3 \\
value(s_{12}) &= 5 \\
value(s_{21}) &= 3 \\
value(s_{22}) &= 6
\end{aligned}
$$

合意案候補全体の集合を $\boldsymbol{S}$ とすると，

$$
\boldsymbol{S} = \{(s_{11}, s_{21}), (s_{11}, s_{22}), (s_{12}, s_{21}), (s_{12}, s_{22})\}
$$

である．(s_{11}, s_{21}) について，獲得効用の計算例を示す．

$$
\begin{aligned}
U((s_{11}, s_{21}); \boldsymbol{w}) &= w_1 \cdot \frac{value(s_{11})}{\arg\max_{s_1 \in I_1} value(s_1)} + w_2 \cdot \frac{value(s_{21})}{\arg\max_{s_2 \in I_2} value(s_2)} \\
&= 0.3 \cdot \frac{3}{5} + 0.7 \cdot \frac{3}{6} \\
&= 0.53
\end{aligned}
$$

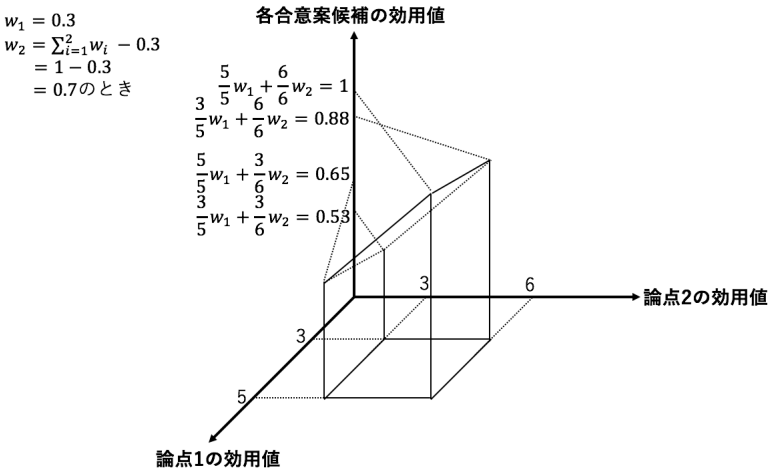

図 1.2: 論点数 2，各論点の選択肢が 2 の場合の効用空間

他の合意案候補についても同様にして

$$
\begin{aligned}
U((s_{11}, s_{22}); \boldsymbol{w}) &= 0.88 \\
U((s_{12}, s_{21}); \boldsymbol{w}) &= 0.65 \\
U((s_{12}, s_{22}); \boldsymbol{w}) &= 1
\end{aligned}
$$

となる．実際に交渉を行った場合に，各エージェントが獲得する効用値には，1.3 章で述べた割引効用が適用される．

第2章

Jupiterの起動方法

2.1 序言

本章では，Jupiter の起動方法について示す．本章で述べるソースコードについては，
`https://gist.github.com/TomoyaFukui/1c98d3ce2dcdb5256ac372e8008be793`
からダウンロードできる．

2.2 動作環境

Jupiter は Python3 で書かれており，以下の環境において動作を保証する．

- python がバージョン 3.5 以上であること，
- 「numpy」，「pandas」，「py4j」，「matplotlib」及び「Cython」モジュールがそれぞれインストールされていること．

また Cython では，C 言語のコンパイラが別途必要となるため注意されたい．mac 環境や linux 環境ではデフォルトで入っていると思われるが，windows では MinGW などにより gcc をインストールする必要がある．
参考として筆者の動作環境を以下に示す．

macOS の動作例

- OS
  - macOS Sierra 10.12.5

- Python のバージョン
  - 3.6.1

- Python3 パッケージ一覧
  - cycler==0.10.0
  - matplotlib==2.1.1
  - numpy==1.13.3
  - pandas==0.21.1
  - py4j==0.10.6
  - pyparsing==2.2.0
  - python-dateutil==2.6.1
  - pytz==2017.3
  - six==1.11.0
  - Cython=0.28

Windows の動作例

- OS
  - Windows 10

- Python のバージョン
  - 3.6.4

- Python3 パッケージ一覧
  - cycler==0.10.0
  - Cython=0.28
  - kiwisolver==1.0.1
  - matplotlib==2.2.0
  - numpy==1.14.2
  - pandas==0.22.0
  - py4j==0.10.6
  - pyparsing==2.2.0
  - python-dateutil==2.7.0
  - pytz==2018.3
  - six==1.11.0

2.3　Jupiter を実行する

この節では Jupiter のプログラムを実行するまでを解説する．

1. ターミナル上で「pip install jupiter-negotiation」と打つ．

2. ターミナル上で「jupiter –test」と打つ．
   実行できた場合，二者間交渉が始まる．

以上の手順により，jupiter を実行することができる．図 2.1 のような画像が表示されれば，実行成功である．

2.4　Jupiter の実行方法

2.4.1　Jupiter の実行ファイル例

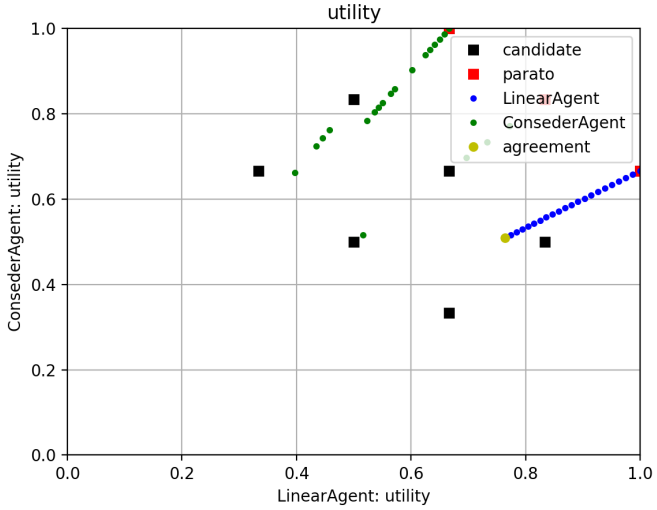

図 2.1: Jupiter の実行例

Listing 2.1: Jupiter の実行ファイル例

```
import os
import jupiter
from jupiter.simulator.jupiter import Jupiter
from jupiter.simulator.negotiationRule import TypeOfNegotiation
from jupiter.agents import linearAgent
from jupiter.agents import concederAgent

if __name__ == '__main__':
    jupiter_path = os.path.abspath(jupiter.__path__[-1])
    domain_path = os.path.join(jupiter_path, "jupiter/domain/")
    jupiter = Jupiter(TypeOfNegotiation.Turn,
                      180,
                      domain_path + 'Atlas3/triangularFight.xml',
                      domain_path + 'Atlas3/triangularFight_util1.xml',
                      domain_path + 'Atlas3/triangularFight_util2.xml')
    jupiter.set_agent(linearAgent, 'LinearAgent')
    jupiter.set_agent(concederAgent, 'ConcederAgent')
    jupiter.do_negotiation(is_printing=True, print_times=1)
    jupiter.display.show()
```

この章では Jupiter の，より詳しい実行方法について述べる．以下の手順により Listing2.1 をまず実行して欲しい．

1. main.py を作成する．
2. main.py の中身を Listing2.1 に書き換える．
3. main.py を実行する．
   実行できた場合，二者間交渉が始まる．

以下，main.pyの解説に加え，Jupiterを実行する際の流れを説明する．

1. jupiterモジュールからJupiterクラス，negotiationRuleモジュールからTypeOfNegotiationクラスをそれぞれimportする．

2. Jupiterクラスを作成する．コンストラクタに指定する引数は以下．
   1つ目の引数として，交渉の種類を指定する．ターン制と時間制がある．
   2つ目の引数として，交渉の長さを指定する．時間の場合，単位は秒である．
   3つ目の引数として，交渉の設定ファイルを指定する．
   4つ目以降の引数として，効用情報の書かれたファイルを指定する．
   Jupiterの仕様として，4つ目以降のファイルの数が，同時交渉エージェントの数となる．

3. 交渉に参加するエージェントを指定する．
   Listing2.1では，LinearAgentとConcederAgentを指定している．
   引数はstr型にして，クラスの名前と同じ変数を渡す．

Jupiterを実行する際は，上記の流れを行うことにより，交渉シミュレーションを行うことができる．より詳しい機能については，jupiter.pyやリファレンスを参照されたい．

2.4.2　ドメインファイルを追加する

Jupiterでは現在，Geniusで作成されたドメインファイルを扱えるようになっており，ドメインファイルの作成機能に関しては，サポートしていない．Geniusで作成したファイルを用いたい場合は，Jupiterのコンストラクタにドメインファイルのパスをstrとして打ち込めば良い．今後ドメインファイルの作成機能については，追加していく予定である．

2.4.3　エージェントを作成する

本章では，エージェントを作成し，Jupiterで動かすところまでを説明する．

1. myAgent.pyをmain.pyと同じディレクトリに作成する．

2. myAgent.py の中身を Listing2.2 に書き換える．

3. main.py 中の 17行目を「jupiter.set_agent(myAgent, 'MyAgent')」に変更する．

上記の手順を踏むことにより，作成したエージェントによる自動交渉実験が可能となる．

Listing 2.2: myAgent.py

```
import sys
import os
from jupiter.simulator import abstractAgent
from jupiter.simulator import agentAction
from jupiter.simulator import abstractUtilitySpace
from jupiter.simulator import negotiationRule


class MyAgent(abstractAgent.AbstractAgent):
    def __init__(self,
                 utility_space: abstractUtilitySpace.AbstractUtilitySpace,
                 negotiation_rule: negotiationRule.NegotiationRule,
                 agent_id: int,
                 agent_num: int):
        self.__utility_space = utility_space
        self.__rule = negotiation_rule
        self.__agent_id = agent_id
        self.__opponent_bid = None

    def receive_action(self, agentAction_: agentAction.AbstractAction):
        if isinstance(agentAction_, agentAction.Offer):
            self.__opponent_bid = agentAction_.get_bid()

    def send_action(self):
        def get_conssetion_value():
            return (1.0 - self.__rule.get_time_now())

        if self.__opponent_bid is not None and \
           get_conssetion_value() < \
           self.__utility_space.get_utility(self.__opponent_bid):
          return agentAction.Accept(self.__agent_id)

        threshold = get_conssetion_value()
        bid_offer = \
    self.__utility_space.get_bid_above_concession_value(threshold)
        return agentAction.Offer(self.__agent_id, bid_offer)

    def receive_start_negotiation(self):
        self.__opponent_bid = None

    def get_name(self):
        return 'MyAgent'
```

2.5 Jupiterに初めから登録されているもの

2.5.1 エージェント

Jupiterではサンプルエージェントとして，譲歩関数を用いたLinearエージェント，Concederエージェント及びBoulwareエージェントが提供されている．

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．交渉戦略について，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．

$$T(t) = u_{max} \cdot (1.0 - t^{\alpha}) \tag{2.1}$$

具体的には式(2.1)で定義される譲歩関数$T(t)$を提案している．u_{max}は最大獲得効用値である．GeniusやJupiterで自動交渉を行う場合は$u_{max} = 1$である．

式(2.1)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．またConcederエージェントは式(2.1)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(2.1)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．

2.5.2 交渉ドメインファイル

過去のANACで用いられた交渉ドメインファイルの一部が初めから登録されている．具体的には，jupiterモジュールがインストールされているディレクトリの中の，ドメインフォルダを見て欲しい．インストール先ディレクトリは「pip show jupiter-negotiation」などのコマンドにより確認することができる．

2.5.3 その他

Jupiter は Jupiter Notebook 上での実行を推奨する.

`http://www.itolab.nitech.ac.jp/Jupiter-HP/en/quick_start/quick_start.html`

などを参考にされたい.

第3章

終わりに

以上でUser Guideを終わりとする．より詳しい説明や機能に関しては，リファレンスを参照されたい．また実際にJupiterを使っていただければ幸いである．

参考文献

[1] Aydoğan, Reyhan, et al.: "Alternating offers protocols for multilateral negotiation." Modern Approaches to Agent-based Complex Automated Negotiation. Springer International Publishing, 2017. 153-167.

[2] Lin, Raz, et al.: "Genius: An integrated environment for supporting the design of generic automated negotiators." Computational Intelligence 30.1 (2014): 48-70.

[3] T. Baarslag, R. Aydogan, K. V. Hindriks, K. Fujita, T. Ito, and C. M. Jonker.: The automated negotiating agents competition 2010-2015. AI Magazine, 2016.