AgentNeko23

Ryota Arakawa*

1 Negotiation Strategy

The behavior of our agent is mainly based on the negotiation environment. We define a boolean a as Eq.(1). Here, Q^{exin}, Q^{exout} are the total quantity of exogenous input/output contracts. EM is a constant and we use $EM = 1.05$. If a is true and our agent is a seller, it means that partners should concede to fulfill their exogenous contracts because the quantity of supply is less than the quantity of demand. Thus, our agent can adopt a bullish strategy. On the contrary, if a is false and our agent is a seller, it means that our agent has to concede to fulfill its exogenous contract for the same reason. Thus, our agent should adopt a bearish strategy. The same can be said if our agent is a buyer. We found that our agent works better when it is a bit bullish, so we set $EM = 1.05$ to make it more bullish.

$$a = \begin{cases} true & (seller, Q^{exin} \leq Q^{exout} EM) \\ false & (seller, Q^{exin} > Q^{exout} EM) \\ true & (buyer, Q^{exout} \leq Q^{exin} EM) \\ false & (buyer, Q^{exout} > Q^{exin} EM) \end{cases} \tag{1}$$

We use a to switch the behavior of our agent. At the beginning of each day, our agent updates a based on Q^{exin} and Q^{exout}.

Our agent inherits OneShotSyncAgent and uses the default OneShot utility function.

1.1 Offering Strategy

Quantity

We implemented two strategies: Random Quantity Strategy (RQS) and Expected Quantity Strategy(EQS). Based on the result of the evaluation shown in Section 2, the submitted agent uses EQS.

We use Algorithm 1 to decide the offer quantities Q. $Q[i]$ is the offer quantity from our agent to agent i. RQS and EQS use the same procedure, but the functions (Eqs.(2)

* Tokyo University of Agriculture and Technology, arakawa@katfuji.lab.tuat.ac.jp

and (4)) are different. Here, FV, FN, RM is a constant. We set $FV = 0.7, FN = 3$, and $RM = 0.3$.

Algorithm 1 Algorithm to decide the offer quantities

Require: Negotiation partners P, needed quantity n, Environment flag a

```
Q[i] ← 1 for all i ∈ P
Let A = {i|i ∈ P}
while n > f(Q, a) do
    Sample an agent i from A with weights w(a, i)
    Q[i] ← Q[i] + 1
    if Q[i] ≥ n then
        Remove i from A
    end if
end while
return Q
```

$$f(Q,a) = \begin{cases} \sum_i Q[i] & (RQS) \\ \sum_i Q[i] g(s[a,i], d[a,i], a) & (EQS) \end{cases} \tag{2}$$

$$g(s,d,a) = \begin{cases} FV & (d[a,i] \leq FN) \\ \max\left(RM, \dfrac{s[a,i]}{d[a,i]}\right) & (d[a,i] > FN) \end{cases} \tag{3}$$

$$w(a,i) = \begin{cases} \dfrac{1}{|A|} & (RQS) \\ \dfrac{g(s[a,i], d[a,i])}{\sum_j g(s[a,j], d[a,j])} & (EQS) \end{cases} \tag{4}$$

We use $s[a,i]$ and $d[a,i]$ to calculate the agreement rate of agent i in the environment a. At the beginning of a simulation, our agent initializes $s[a,i]$ and $d[a,i]$ with all zeros. At the end of the day, our agent updates $s[a,i]$ and $d[a,i]$ using Algorithm 2.

Algorithm 2 Algorithm to update $s[a, i]$ and $d[a, i]$

Require: Environment flag a, Negotiation partners P

for all $i \in P$ **do**
  if failed to make a contract with i **then**
    $d[a, i] \leftarrow d[a, i] + 1$
  else if i agreed to my proposal **then**
    $d[a, i] \leftarrow d[a, i] + 1$
    $s[a, i] \leftarrow s[a, i] + 1$
  end if
end for

There are two differences between RQS and EQS. The first is the total offer quantity determined by Eq.(2). With RQS, the total offer quantity is equal to n. With EQS, on the other hand, our agent takes the agreement rates into account, and offers more than n to avoid the risk of shortfall. The second is the sampling weights determined by Eq.(4). RQS distributes the offer quantity uniform-randomly. In contrast, EQS also distributes it randomly, but gives more weight to agents with high agreement rates. This trick makes our agent to make contracts with stable quantity by prioritizing friendly agents.

Unit Price

If a is true, our agent offers at the bullish unit price. Otherwise, it offers at the bearish unit price.

1.2 Responding Strategy

Responding Strategy has three modes: Last Response Mode, Advantageous Mode, Disadvantageous Mode.

Last Response Mode

This mode is used when our agent is unable to present counter-offers to partners after creating a response. Our agent chooses proposals to maximize the utility, and accepts them.

Advantageous Mode

This mode is used when a is true. First, our agent selects proposals whose price is favorable to it and makes them P_c. Next, it selects proposals from P_c so that the total quantity of proposals is equal to the needed quantity. If such a combination exists, it accepts them. Otherwise, it rejects all proposals and makes counter-offers.

Disadvantageous Mode

This mode is used when a is false. Our agent chooses proposals so that the total quantity of proposals q_{sum} and the needed quantity q_{need} satisfy the condition Eq.(5).

Here, QT is a constant. We set $QT = 0.7$. It accepts or rejects proposals the same way as Advantageous Mode.

$$QT \times q_{need} \leq q_{sum} \leq q_{need} \tag{5}$$

2 Evaluation

We tested AgentNeko23 by running a tournament. We use following agents in the tournament. The number of worlds is 50, the number of runs per world is 1, and the number of days in each run is 100.

- AgentNeko23: the agent which we implemented. This variant uses EQS.
- AgentNeko23Random: almost as same as AgentNeko23, but this variant uses RQS.
- SimpleAgent[1]: one of the tutorial agent.
- Neko[2]: the agent which we submitted last year.

Table 1 shows the result. The result shows that AgentNeko23 outperforms other agents.

Table 1: The test result

agent_type	mean	min	25%	median	75%	max
AgentNeko23	1.109	0.670	0.998	1.095	1.216	1.534
SimpleAgent	1.103	0.828	0.998	1.093	1.200	1.466
AgentNeko23Random	1.085	0.555	0.988	1.084	1.185	1.516
Neko	1.031	0.102	0.923	1.051	1.149	1.613

References

[1] Developing an agent for scml2023 (oneshot) — scml 0.5.6 documentation. http://www.yasserm.com/scml/scml2020docs/tutorials/02.develop_agent_scml2020_oneshot.html#simple-oneshotagent. Accessed on 04/28/2023.

[2] Ryota Arakawa. AgentNeko. https://github.com/yasserfarouk/scml-agents/blob/master/scml_agents/scml2022/oneshot/team_123/report.pdf, 2022. Accessed on 04/28/2023.