Chapter 14
The Simple-Meta Agent

Litan Ilany and Ya'akov (Kobi) Gal

Abstract The Simple-Meta agent uses machine learning to select the negotiation strategy that is predicted to be most successful based on structural features of the domain.

Keywords Algorithm selection • Machine learning • Negotiation

14.1 Introduction

The Simple-Meta agent combines machine learning with known agent strategies for ANAC in order to choose the best existing strategies for different domains. This agent exploits the fact that individual negotiation strategies from the literature vary widely in their best-case performance for different negotiation domains [1]. Our methodology consists of defining a set of features that encapsulate the information about the domain that is available to agents at the onset of negotiation. These features are then used to predict the performance of existing negotiation strategies on a new domain using canonical learning methods including multi-layer neural networks, decision trees, and linear and logistic regression. At run time, we select the negotiation strategy that is predicted to be most successful on a new domain based on its features.

L. Ilany (✉) • Y. Gal
Ben-Gurion University of the Negev, Sheva, Israel
e-mail: kobig@bgu.ac.il; litanil@bgu.ac.il

14.2 Definitions

A *domain* consists of a set of issues L. Each issue $l \in L$ can take one of possible discrete values out of the set V_l. The domain is common knowledge to the negotiating parties. A proposal $p = (v_1, \ldots, v_{|L|})$ is an assignment of values to all issues in L. Let P denote the set of all possible proposals in a domain. A *negotiation round* involves two participants termed Agent1 and Agent2. Each agent has a *profile*, that determines its valuation of a proposal, which is private information. The profile of Agent1 includes (1) a valuation function $o_1 : V_l \to \mathbb{R}$ mapping a value of issue l to the real numbers; (2) a weight vector for all issues $W_1 = (w_{1,1}, \ldots, w_{1,|L|})$ where $w_{1,l}$ is the weight of issue l; (3) a discount factor δ_1; (4) a reservation value r_1. (The profile of Agent2 is defined in a similar way).

In a negotiation round, Agent1 and Agent2 make alternating take-it-or-leave-it offers to each other until a proposal is accepted or a predetermined deadline is reached. Each agent has a *role* that determines whether the agent makes the first or second offer in the negotiation round. If an agreement is reached for a proposal p^t at time t, the utility of Agent1 is

$$u_1(p^t) = \left[\sum_{l \in L} w_{1,l} \cdot o_1(v_l) \right] \cdot \delta_1^t \tag{14.1}$$

Otherwise, the utility of Agent1 is $r_1 \cdot \delta_1^t$ (and similarly for Agent2). The *score* of an agent in a negotiation round is simply the utility it achieved in the round.

14.2.1 Constructing Domain Features

We defined three types of features for any domain d in a tournament. The first type corresponds to domain information that is common knowledge, including the following features:

- the number of issues in the domain ($|L|$),
- the average number of values in each issue $AVG(\{|V_k| \mid \forall k \in L\})$,
- the number of possible proposals in the domain ($|P| = \prod_{k \in L} |V_k|$).

The second type of features corresponds to an agent's profile which is private information. We describe these features from the point of view of Agent1 at time 0 (when it is needed to select an agent):

- the discount factor δ_1 and the reservation value r_1,
- the standard deviation of weights over all possible issues $SD(\{w_{1,k} | \forall k \in L\})$,
- the average utility at time $t = 0$ over all possible proposals in the domain $(AVG(\{u_1(p^0) \mid p^0 \in P\}))$,
- the standard deviation of its utility $(\{SD(\{u_1(p^0) \mid p^0 \in P\})$ over all possible proposals P.

The third type of features corresponds to information that is inferred from the *first* proposal p' that Agent1 receives from Agent2. These features include:

- the utility of agent Agent1 at time 0 from the proposal ($u_1(p')$),
- the average utility at time 0 of all proposals that are preferable to Agent1 than p' (give higher utility) ($AVG(\{u_1(q^0) \mid \forall q^0 \text{ s.t. } u_1(q^0) > u_1(p')\})$),
- the standard deviation over the utility over all such preferable proposals ($SD(\{u_1(q^0) \mid \forall q^0 \text{ s.t. } u_1(q^0) > u_1(p')\})$).

14.2.2 The Simple Meta-Agent

Let $s_{i,j}^d$ denote the score obtained by agent i when negotiating with agent j in any domain d.[1] Let $s_i^d = AVG(\{s_{i,k}^d \mid \forall k \in A, k \neq i\})$ denote the average score for agent i that negotiates in domain d over all training agents A. Let $s^d = AVG(\{s_j^d \mid \forall j \in A\})$ denote the average score of all training agents that negotiate with each other in domain d. The optimal agent in A for a domain d is associated with the highest average score s_*^d when negotiating with all of the testing agents in A':

$$s_*^d = \max_{i \in A}(AVG(s_{i,k}^d \mid k \in A')) \tag{14.2}$$

We used canonical supervised learning algorithms to predict the performance of agents given a domain and profile by the difference between the score of agent i when negotiating with any agent k in domain d and the average score over all negotiations with all agents in the domain ($s_{i,k}^d - s^d$). We used different learning techniques to predict an agent's performance when negotiating in a new domain (adapting standard overfitting avoidance methods for each technique). A regression tree algorithm that selected the tree size for minimizing the cross-validation error [2]; a neural network with a single hidden layer and four hidden nodes, using early stopping after 150 iterations when training; a linear regression model with a forward-backward selection method for choosing the predictive variables [3].

The algorithm used by the simple meta-agent to choose an agent strategy is given in Fig. 14.1 (presented from the point of view of Agent1). We assume knowledge of a set of training domains D and agents A. This training data is used to learn (off-line) the models described above. Given a test domain $d \in D'$ the agent first check whether d is already known ($d \in D$). In this case, the best the meta-agent can do is to select the agent in A that achieved the best performance in d (line 2). Otherwise, the meta-agent will compute the features associated with the domain. These features depend on receiving a proposal from Agent2 (line 5). If the meta-agent is the first

[1]We assume a one-to-one correspondence between an agent $i \in A$ and its negotiation strategy; we use i to refer to either.

Fig. 14.1 Simple meta-agent algorithm

```
Known: Domains $D$, agents $A$
Input: Test domain $d \in D'$
Output: Agent strategy $i^*$
1. If $d \in D$ then
2. return agent $i^*$ such that $i^* \in \text{argmax}_{i \in A} s_i^d$
3. If (Agent1.IsProposer) then
4. make first proposal $p*^0 \in \text{argmax}_{p^0} u_1(p^0)$
5. Receive first proposal $p'$ from Agent2
6. Get the feature list $F(d, p')$
7. For each agent $a \in A$
8. $\widehat{per_a^d}$ = predict performance using $F(d, p')$
9. return agent $i^*$ such that $i^* \in \text{argmax}_{i \in A} \widehat{per_i^d}$
```

proposer, it needs to make a proposal to Agent2. Lacking any information about the profile of Agent2, it makes the proposal that provides it with maximal utility (lines 3–4). In line 6, the meta-agent computes the features associated with domain d and the proposal p' received from Agent2. Finally, in lines 7–8 it predicts the performance of each agent in A on domain d, and returns the agent with the highest predicted performance (breaking ties randomly).

Essentially, the algorithm above describes a class of simple meta-agents that depend on which learning method and performance predicting measure is used. The run-time of the algorithm is dominated by the feature selection process, which is polynomial in the size of the bid space in the domain. In practice, this process terminated in less than a second for each domain on a commodity core i5 computer.

References

1. Lin, R., Kraus, S., Baarslag, T., Tykhonov, D., Hindriks, K.V., Jonker, C.M.: Genius: an integrated environment for supporting the design of generic automated negotiators. Comput. Intell. (2012)
2. Breiman, L., Friedman, J., Stone, C., Olshen, R.: Classification and Regression Trees. Chapman & Hall, New York (1984)
3. Shibata, R.: An optimal selection of regression variables. Biometrika **68**(1), 45–54 (1981)