Choosing Multi-Issue Negotiating Object Based on K-Armed Bandit Problem

Liming Wang [12] **Yumei Chai** [1] **Houkuan Huang** [2]

[1]School of Information Engineering, Zhengzhou University, Zhengzhou 450052,China

[2]School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China

Abstract

This paper solves problem of choosing seller before negotiation in order to improve successful rate of the multi-issue negotiation and buyer (agent)'s utility. In order to fully utilize negotiation history, we transform the problem of choosing seller (agent) into a K-armed bandit problem. Several improved algorithms, which are used to learn reward distribution and combine technologies for K-armed bandit problem, are presented. Finally, combination of the improved algorithms and trust and reputation improves accuracy and practicability of choosing a seller. Several experiments prove validity of the work in application.

Keywords: agent; K-armed bandit problem; negotiation

1. Introduction

Multi-issue negotiation is one of the main interactive ways in e-commerce. In Web-based e-market, when a buying agent (called as buyer) carries user's requirements to come to the market, it will face many selling agents (called as sellers). The buyer needs choose an appropriate seller according to the user's requirements, and negotiate with it to maximize its outcome utility in negotiation. Literature [1] and other literatures about negotiation presented respectively some multi-issue negotiation models under different strategies, but they did not take into account choosing seller before negotiation. Under incomplete information of sellers, how does a buyer choose an appropriate seller? Solving the problem before negotiation is very important in applications. The buyer can learn negotiation histories to build the trust on sellers and the probability distribution of reward for choosing seller under no knowledge of sellers. This paper will transform the problem of choosing seller into a multi-armed bandit problem. Literature [2] uses technologies for multi-armed bandit problem to present an algorithm $Hedge(\beta)$ to solve problem of distributing resource.

The remainder of the paper is structured as follows. Transforming the problem of choosing seller into a K-armed bandit problem and several $Hedge(\beta)$ -based improved algorithms are presented in Section 2. Experiments and their analysis are presented in section 3. Finally, in section 4, we present some conclusions.

2. Transforming the Problem of Choosing Seller into K-armed Bandit Problem

2.1. K-Armed Bandit Problem

In original K-armed bandit problem [2], a gambler must choose which of K slot machines to play. At each time step, he pulls the arm of one of the machines and receives a reward or payoff (possibly zero or negative). The gambler's purpose is to maximize his total reward over a sequence of trials. Since each arm is assumed to have a different rewards distribution, the goal is to find the arm with the best-expected return as early as possible, and then keep gambling using that arm.

2.2. Problem of Choosing Sellers

Definition 1 Negotiating Agent Set

$\sum = \sum_B + \sum_S$ denotes set of agents participating negotiation, where $\sum_B$ denotes set of buyers, $\sum_S$ denotes set of sellers.

Definition 2 Multi-Issue Joint Utility

$\forall a \in \sum$, $JU_a : \Omega_1 \times \Omega_2 \times ... \times \Omega_n \to R$ is utility function of agent a over multi-issue. For a and value vector $\vec{V}$, $JU_a(\vec{V})$ integrates the utility of various issues weights: $JU_a(\vec{V}) = \sum_{k=1}^{n} w_a^{i_k} \times \xi_a^{i_k}(v^{i_k})$, where i_k is an issue, $\xi_a^{i_k}(v^{i_k})$ is utility of v^{i_k} , $w_a^{i_k}$ denotes the weights of $\xi_a^{i_k}(v^{i_k})$ of issue i_k in $JU_a(\vec{V})$, and $\sum_{k=1}^{n} w_a^{i_k} = 1$.

For $\forall b \in \sum_B , |\sum_S| = K$ （equivalent to arms number in K-armed bandit problem）, b （equivalent to the gambler） chooses a seller in $\sum_S$ （equivalent to choose arm to play）to negotiate with. b needs to learn seller's negotiation histories under no knowledge of all sellers in $\sum_S$. For $\forall s_i \in \sum_S , i = 1, 2, ..., K$, it has a negotiation history, denoted as $H^{s_i} =< HT_1^{s_i}$

$, HT_2^{s_i}, ..., HT_m^{s_i} >$, where m denotes length of the history, $HT_i^{s_i}$ denotes a negotiation thread in the history. b simulates negotiation course with s_i by learning the history, and obtains the reward distribution from the course. b learns a $HT_i^{s_i}$ to be equivalent to play one time by choosing s_i.

For $HT_j^{s_i} =< O_{j,1}^{s_i}, O_{j,2}^{b'}, ..., O_{j,y}^{s_i} >$, where y is length of the thread, supposing the first element $O_{j,1}^{s_i}$ in the thread is from s_i. If the current offer of s_i is $O_{j,2k+1}^{s_i}$, the counter offer of b is denoted as O_k^b, the counter offer in the history is denoted as $O_{i,2k+2}^{b'}$, $\Delta_k^{s_i} = JU_b(O_{i,2k+2}^{b'}) - JU_b(O_k^b)$ denotes reward of interaction one time. The obtained reward of b's n^{th} play by choosing s_i is: $r^{s_i}(n) = \sum_{k=1}^{\lfloor y/2 \rfloor} \Delta_k^{s_i}$

If there is an algorithm Al, and a buyer obtains the reward by using Al, and $F^{s_i}(N)$ denotes the number of times of choosing s_i in N plays, then the regret of the algorithm is as follows after N plays.

$$R_\Delta^{Al} = r^* N - \max_{1 \le i \le K}(r^{s_i} \sum_{i=1}^K IE[F^{s_i}(N)])$$

where $r^* \overset{def}{=} \max_{1 \le i \le K}(r^{s_i})$, $IE[\cdot]$ denotes expectation. ER_{best} denotes total obtained rewards of the best seller after N plays.

$$ER_{best} = \max_{1 \le i \le K}(r^{s_i} \sum_{i=1}^K IE[F^{s_i}(N)])$$

R_Δ^{Al} denotes the reward loss of the algorithm Al in N plays. The more the number of times of choosing the best seller in N plays, the less the reward loss, the better the performance of the algorithm. The algorithm generates the reward distribution by confirming N plays on $\{s_1, s_2, ..., s_K\}^N$.

2.3. Simulating Negotiation and Algorithm for Estimating Reward Distribution

2.3.1. Algorithm for simulating negotiation

The simulating negotiation is a course that a buyer $b \in \sum_B$ learns the negotiation histories satisfying the user's buying requirements. He learns the offers of the seller and the original buyer during the course, and gives own offer according to the offer of the seller. The utility difference of the offer of the original buyer and own offer is reward of interaction at one time.

[Algorithm for simulating negotiation: $N(s_i)$]

[Explanation] The algorithm performs one time, the buyer b will finish negotiation of a thread with s_i, and the sum of rewards is called as the reward of one play. Supposing the successful thread of s_i is $HT_i^{s_i}$, and there is a pointer to point the first element in $HT_i^{s_i}$, when processing an offer, the pointer moves a position backwards. The element pointed is denoted as $HT_i^{s_i} \uparrow$.

1. $O^{s_i} \leftarrow HT_i^{s_i} \uparrow$, if the O^{s_i} of seller is offer for ending, then the negotiation simulated ends, and returns the sum of rewards.
2. $O^{b'} \leftarrow HT_i^{s_i} \uparrow$, if the $O^{b'}$ of buyer is offer for ending, then the negotiation simulated ends, and returns the sum of rewards.
3. The buyer b generates own offer O^b according to O^{s_i}. $\Delta^{s_i} = JU_b(O^{b'}) - JU_b(O^b)$, if $\Delta^{s_i} > 0$, then reward is positive, i.e. $r^{s_i} \leftarrow r^{s_i} + \Delta^{s_i}$, otherwise, the reward is zero, and then go to 1.

2.3.2. Algorithm for Estimating Reward Distribution

In the past, the bandit problem has almost always been studied with the aid of statistic assumptions on the process generating the rewards for each arm. However, in fact, the assumptions may be difficult or impossible to be determined. Literature [3] solves K-armed bandit problem without statistical assumptions, generates the rewards distribution of the arm in T trial steps. This paper puts this technology into the following algorithms, and generates the rewards distribution of choosing seller. This section presents several improved algorithms based on algorithm $Hedge(\beta)$ in literature [2].

[Algorithm for Estimating Distribution: $H(\alpha, n)$]

[Explanation] The algorithm uses parameter $\alpha > 0$ to determine rewards distribution of choosing seller s_i. It just considers the distribution of choosing seller at n^{th} step. $RS_i(n)$ denotes the sum of rewards of choosing s_i in previous $n-1$ steps.

[Initialization] $for\ i = 1, ..., K\ do \quad RS_i(1) \leftarrow 0$.

$for\ i = 1, ..., K\ do$

{ 1. $P(n)$ is distribution vector, the probability of choosing s_i is $p_i(n)$:

$$p_i(n) = (1+\alpha)^{RS_i(n)} / \sum_{j=1}^K (1+\alpha)^{RS_j(n)}$$

2. Executing $N(s_i)$ to negotiate with s_i, and obtain $r^{s_i}(n)$, $r^{s_i}(n) \leftarrow V_s(r^{s_i}(n))$, $V_s(\,)$ is a score function of the reward, where $r^{s_i}(n) \in [0,1]$.
3. Accumulating reward of s_i

$$RS_i(n+1) \leftarrow RS_i(n) + r^{s_i}(n)$$

}

Algorithm $H(\alpha, n)$ considers just possibility of each chose seller at one step. If executing $for\ n = 1, ..., N\ do$ { $H(\alpha, n)$ }, then obtaining distribution information of each chose seller in N steps. The algorithm chooses the seller according to the distribution and reward as follows.

$$S^* = \arg\max_{1 \le i \le K}(\sum_{j=1}^N p_i(j) r^{s_i}(j))$$

where S^* denotes a seller who can give maximal reward under the distribution.

2.4. Combining the Trust and Reputation with Maximization of Reward

The buyer explores the best seller by simulating negotiation. The rewards from each possible seller are summed respectively in exploration course. The seller with the maximization of the reward is chose after N steps. However, considering the following factors still: (1) The negotiation histories may be not enough. (2) N may be not greater enough. (3) The time distance of negotiation histories from present time will influence choosing seller by total reward. (4) The outside factors are also important to choose seller. In order to combine correlating factors above, several vectors are given as follows.

$\delta^{b_i} =< \delta_1, \delta_2, ..., \delta_K >$ denotes time discount factor vector, where δ_j denotes the discount factor for reward which $b_i \in \sum_B$ learns the histories of $s_j \in \sum_S$ to obtain.

$Tm^{s_i} =< t_1^{s_i}, t_2^{s_i}, ..., t_N^{s_i} >$ denotes negotiation time vector for negotiation histories of $s_i \in \sum_S$ in N steps, where $t_j^{s_i}$ denotes negotiation time for choosing s_i at j step.

$T_S^{b_i} =< T_{b_i \to s_1}, T_{b_i \to s_2}, ..., T_{b_i \to s_K} >$ denotes a trust vector of $b_i \in \sum_B$ on each $s_j \in \sum_S$, where $T_{b_i \to s_j} \in [0,1]$ denotes the trust degree of b_i on s_j. If there is no deals between b_i and s_j, then $T_{b_i \to s_j} = R_{b_i \to s_j}$, where $R_{b_i \to s_j}$ denotes the reputation degree of s_j relative to b_i.

$T_B^{b_i} =< r_{k_1}^{b_i}, r_{k_2}^{b_i}, ..., r_{k_m}^{b_i} >$ denotes the trust degree of $b_i \in \sum_B$ on other $b_j \in \sum_B$, where $i \neq j$, $m = |\sum_B| - 1$ and $k_j \neq i$.

[Extension of $H(\alpha, n) : E(\alpha, \gamma)$]

[Explanation] Combining the time discount factors with outside information based on $H(\alpha, n)$. ss_j denotes the discount total reward.

[Initialization] Initializing $H(\alpha, n)$, $ss_j \leftarrow 0$.

for $n = 1, ..., N$ *do*

{ 1. Executing $H(\alpha, n)$ and obtaining $Q(n) \in [0,1]^K$

2. Obtaining $T_S^{b_i} \in [0,1]^K$, $P(n) \in [0,1]^K$ is another distribution vector.

for $j = 1, ..., K$ *do*

{ $p_j(n) \leftarrow q_j(n) T_{b_i \to s_j}$

3.Overlaying the distribution and the uniform distribution

$$p_j(n) \leftarrow (1-\gamma) p_j(n) + \gamma / K \quad .$$
$$r^{s_j}(n) \leftarrow (\gamma / K) \times (r^{s_j}(n) / p_j(n)) .$$

4. Combining the reward vector at 3th step with $T_S^{b_i} \in [0,1]^K$:

$$r^{s_j}(n) \leftarrow T_{b_i \to s_j} \times r^{s_j}(n)$$

5. Combining time discount factors, and summing rewards of each sellers respectively:

$$ss_j \leftarrow ss_j + p_j(n) \times r^{s_j}(n) \times (\delta_j)^{t - t_n^{s_j}}$$

}

}

In the algorithm above, obtaining distribution $Q(n)$ for choosing seller at 1th step, combining the trust vector to generate new distribution at 2th step, overlaying the distribution and the uniform distribution at 3th step, combining the obtained reward with the trust vectors at 4th step, discount the reward of each seller by time at 5th step, finally, choosing the best seller who has maximal ss_j.

2.5. Choosing parameters α and γ

The parameter α is constraint for generating $P(n)$ in $H(\alpha, n)$, and the parameter γ for overlaying distribution in $E(\alpha, \gamma)$. The following corollary is presented by referring corollary 4.2 in [3].

Corollary If $f \geq ER_{best}$, and algorithm $E(\alpha, \gamma)$ is run with input parameters: $\alpha = \sqrt[3]{(4K \ln K)/f}$, $\gamma = \min\{1, \sqrt[3]{(K \ln K)/(2f)}\}$, then the regret of algorithm $E(\alpha, \gamma)$ is at most:

$$R_{\Delta}^{E(\alpha,\gamma)} \leq \frac{3}{\sqrt[3]{2}} f^{2/3} (K \ln K)^{1/3}.$$

$f \geq ER_{best}$ in the corollary above can be found dynamically. The parameters α and γ that satisfy the relation expression above are called stable point. The maximization of obtained reward that is obtained by negotiating with each seller is compared with f, f is a variable that increases by degrees, and chose as $f \leftarrow 2^n$ in the experiments.

3. Experiments and Analysis

The set of agents in the experiment consists of four buyers, i.e. $|\sum_B| = 4$, and six sellers, i.e. $|\sum_S| = 6$. In experiment, preparing 100 negotiation threads for each seller respectively, and total number of threads are 600.

3.1. Experiment-1

The experiment in this section is based on negotiation histories of s_1, s_2, s_3, s_4, s_5 and s_6, and learns these histories by simulating negotiation. The experiment found dynamically three stable points, as shown in Figure 1, Figure 2 and Figure 3. The 1th stable point is as shown in Figure 1, the stable point maintains in $N = 6$ steps, $\alpha = 1.3903$ and $\gamma = 0.69517$, s_1 has higher chose probability. The 2th stable point is as shown in Figure 2, the stable point maintains in $N = 28$ steps, $\alpha = 1.1035$ and $\gamma = 0.55176$, s_5 has higher chose probability. The 3th stable point is

as shown in Figure 3, the stable point maintains in $N = 100$ steps, it covered all samples of threads, $\alpha = 0.87586$ and $\gamma = 0.43793$, s_3 has higher probability. The 3th stable point maintains the most steps, the more information supports this stable point than the previous two stable points, and so the outcome of the 3th stable point is more accurate than the outcome of the previous two stable points.

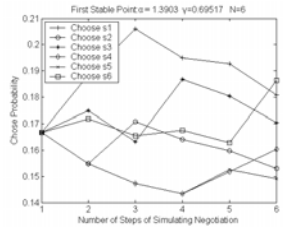

Fig. 1: 1th stable point

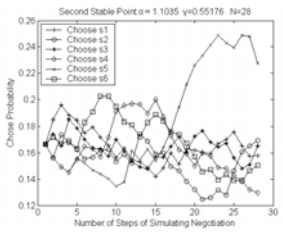

Fig. 2: 2th stable point

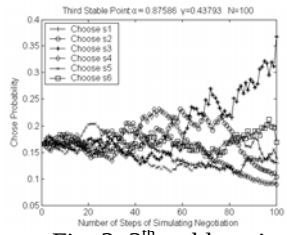

Fig. 3: 3th stable point

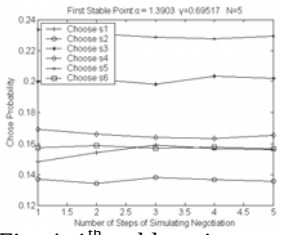

Fig. 4: 1th stable point

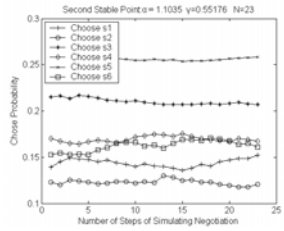

Fig. 5: 2th stable point

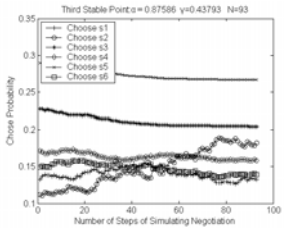

Fig. 6: 3th stable point

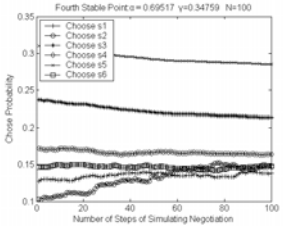

Fig. 7: 4th stable point

3.2. Experiment-2

The experiment in this section considers several factors in section 2.4. The trust vectors $T_S^{b_1}$, $T_S^{b_2}$, $T_S^{b_3}$, $T_S^{b_4}$ and the negotiation histories time vectors Tm^{s_1}, Tm^{s_2}, Tm^{s_3}, Tm^{s_4}, Tm^{s_5}, Tm^{s_6} and the time discount factors δ^{b_1}, δ^{b_2}, δ^{b_3}, δ^{b_4} are combined in the experiment The experiment found dynamically four stable points, as shown in Figure 4, Figure 5, Figure 6 and Figure 7. The 1th stable point is as shown in Figure 4, the stable point maintains in $N = 5$ steps, $\alpha = 1.3903$ and $\gamma = 0.69517$. The 2th stable point is as shown in Figure 5, the stable point maintains in $N = 23$ steps, $\alpha = 1.1035$ and $\gamma = 0.55176$. The 3th stable point is as shown in Figure 6, the stable point maintains in $N = 93$ steps, $\alpha = 0.87586$ and $\gamma = 0.43793$. The 4th stable point is as shown in Figure 7, the stable point maintains in $N = 100$ steps, $\alpha = 0.69517$ and $\gamma = 0.34759$. The four stable points all show s_5 has the highest chose probability and s_3 has higher probability. The experiment-2 has more information than the experiment-1, so the experiment-2 proves also s_3 is the best outcome in the experiment-1. The values of α and γ in previous three stable points are the same respectively, so the previous three stable points between the experiment-1 and the experiment-2 are the same. These all show the outcome of the experiment-2 is superior to the experiment-1. Additionally, the experiment-2 shows also that as long as finding 1th stable point, the best seller will be chose in the given set of sellers.

4. Conclusions

This paper solves the problem of choosing sellers before negotiation from following two main respects in buyer's views: (1) Several improved algorithms are presented. The algorithms learn fully all negotiation histories of the seller based on the simulating negotiation, and combine technologies for solving K-armed bandit problem, transform the problem of choosing seller into a K-armed bandit problem. (2) The paper combines the time factors of negotiation histories further to present corresponding algorithm. The experiments prove the validity and practicability of the algorithms.

5. References

[1] Robert M.Coehoorn, Nicholas R.Jennings. "Learning an Opponent's Preferences to Make Effective Multi-Issue Negotiation Trade-Offs", In Proceeding 6th International Conference on e-Commerce, Delft, The Netherlands 59-68 , 2004.

[2] Yoav Freund, Robert E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting", In Computational Learning Theory: Second European Conference, EuroCOLT'95, pages23-37. Springer-Verlag, 1995.

[3] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, Robert E.Schapire. "Gambling in a rigged casino: The adversarial multi-armed bandit problem", Electronic Colloquium on Computational Complexity, Report No. 68, 2000.