Chapter 1
Using transfer learning to model unknown opponents in automated negotiations

Siqi Chen, Shuang Zhou, Gerhard Weiss and Karl Tuyls

Abstract Modeling unknown opponents is known as a key factor for the efficiency of automated negotiations. The learning processes are however challenging because of (1) the indirect way the target function can be observed, and (2) the limited amount of experience available to learn from an unknown opponent at a single session. To address these difficulties we propose to adopt two approaches from transfer learning. Both approaches transfer knowledge from previous tasks to the current negotiation of an agent to aid learn the latent behavior model of an opposing agent. The first approach achieves knowledge transfer by weighting the encounter offers of previous tasks and the ongoing task, while the second one by weighting the models learnt from the previous negotiation tasks and the model learnt from the current negotiation session. Extensive experimental results show the applicability and effectiveness of both approaches. Moreover, the robustness of the proposed approaches is evaluated using empirical game theoretic analysis.

1.1 Introduction

In automated negotiations, two or more autonomous agents try to come to a joint agreement in a consumer-provider or buyer-seller set-up [17]. The biggest driving force behind research into automated negotiation is arguably the broad spectrum of potential applications. Negotiation theory typically differentiates negotiation classes through their negotiation settings, for example, the number of participants on the

Siqi Chen, Shuang Zhou and Gerhard Weiss
Department of Knowledge Engineering, Maastricht University, P.O. Box 616, 6200 MD Maastricht, The Netherlands, e-mail: {siqi.chen, shuang.zhou, gerhard.weiss}@maastrichtuniversity.nl

Karl Tuyls
University of Liverpool, Ashton Street, Liverpool L69 3BX, United Kingdom, e-mail: k.tuyls@liverpool.ac.uk

negotiation table (e.g., bilateral or multilateral), or the number of issues being negotiated upon (e.g. whether a single or multiple issues are the subject of each placed bid). Although the contributions from our work are also applicable to multilateral negotiations, the paper concentrates on bilateral, multi-issue negotiation, simply because this makes the proposed techniques easier to explain. The interaction framework enforced in automated negotiations lends itself to the use of machine learning techniques for opponent modeling. The driving force of an (opposing) agent is governed by its hidden utility function as well as its also hidden bidding strategy. Given the opponent behavior model[1], an agent can reach better final (or cumulative) agreement terms by exploiting this private knowledge. But learning an opposing agent's behavior model is not trivial due to the following reasons [4]:

1. the behavior model can only be observed indirectly through offers refused and counter offers made by the opposing agent;
2. the amount of encounter offers available to learn from in a single negotiation session is limited.

Transfer learning (TL) techniques are therefore adopted to alleviate the learning problems. TL is a branch of machine learning that enables the use of knowledge learned in a previous task (so called *source* task) to aid learning in a different, related new task (*target* task) [20]. In its most basic form, there exists a source and a target task where learning in the source task was already accomplished and the knowledge acquired (in whatever form) is available for use in the target task, with the underlying assumption that the source and target task can be found similar in some sense ([11,20,25]). One of the primary goals of TL is to reach better performance in a new task (with few target data being available) by re-using gained knowledge from source task, which is ideally suited for learning settings like that of automated negotiation. More specifically, the learning of an opponent model in an ongoing negotiation (*target* task) could be benefit from transferring knowledge from previous negotiations (*source* tasks). Aiming at applying transfer learning to automated negotiation, this work contributes by:

1. proposing a generic strategy framework for agent-based negotiation;
2. modifying an instance-based transfer algorithm *TrAdaBoost* [11] for multi-issue negotiation problems, and
3. modifying a model-based transfer algorithm — *ExpBoost* [22] for multi-issue negotiation problems.

The first algorithm transfers knowledge between tasks based on data instances. This approach is intuitively appealing – although the whole data from source task may not be reused directly, certain parts of the data can still be reused together with a few labeled data in the target task [20]. One of the most widely used instance transfer learning algorithm is *TrAdaBoost*, which was proposed to transfer instances between a source and a target task for classification. In the automated negotiation

[1] Because both an agent's utility function and bidding strategy are hidden, we will often use the term behavior model to encompass both as the force governing the agents negotiating behavior.

scenario, encountered offers from previous negotiation session are transferred to improve the learning agent's offer proposition scheme in the ongoing negotiation session. In contrast, the second algorithm is based on the model transfer learning. Model transfer approach, by means of discovering the relation between source and target models, transfers knowledge across source and target tasks. Therefore, such kind of approaches aim at retaining and applying the models learned in a single or multiple source tasks to efficiently develop an effective model for the target task, after seeing only a relatively small number of sample situations. In the automated negotiation scenario, instead of directly transfer the encountered offers from previous sessions, the learned models on historical negotiation sessions are transferred to approximate the model of the current session. It can automatically balance an ensemble of models with each trained on one known opponent.

Experiments performed on various challenging negotiation tasks show that transfer can aid target agents in improving their behaviors once encountering new opponents varying in their preference profiles and/or bidding strategies. The contributions, moreover, includes the discussion of performance improvement resulting from transfer, which opens up a few new directions of research.

The remainder of this paper is structured as follows. Section 1.2 underlines the problem of our research. Section 1.3 proposes the generic strategy framework for automated negotiation and the two transfer learning schemes. Section 1.4 offers extensive experimental results and a game-theoretical analysis of the proposed learning approaches. Section 1.5 provides related work. Section 1.6 identifies some important research lines outlined by our paper and concludes the work.

1.2 Bilateral Negotiation Problem

The automated negotiation framework adopted in this work is a basic bilateral multi-issue negotiation model as it is widely used in the agents field (e.g., [5,7–10]). The negotiation protocol is based on a variant of the alternating offers protocol proposed in [23], where the negotiation process is limited by a certain number of rounds.

Let $I = \{a, b\}$ be a pair of negotiating agents, where i ($i \in I$) is used to represent any of the two agents. The goal of a and b is to establish a contract for a product or service, where a contract consists of a vector of issue values. Inherent to the negotiation process is that agents a and b act in conflictive roles. To make this precise, let J be the set of issues under negotiation where j ($j \in \{1, ..., n\}$) is used to represent a particular issue. Contracts are tuples $O = (O_1, \dots, O_n)$ that assign a value O_j to each issue j. A contract is said to be established if both agents agree on it. Following Rubinstein's alternating bargaining model, each agent makes, in turn, an offer in form of a contract proposal.

Each agent i decides to accept or reject a contract based on a weight vector $w^i = (w^i_1, \dots, w^i_n)$ (also called importance vector or preference vector) that represents the relative importance of each issue $j \in \{1, ..., n\}$. These weights are usually normalized (i.e., $\sum_{j=1}^{n}(w^i_j) = 1$ for each agent i).

The utility of an offer for agent i is obtained by the utility function, defined as:

$$U^i(O) = \sum_{j=1}^{n} (w^i_j \cdot V^i_j(O_j)) \tag{1.1}$$

where w^i_j and O are as defined above and V^i_j is the evaluation function for i, mapping every possible value of issue j (i.e., O_j) to a real number. The two parties continue exchanging offers till an agreement is reached or negotiation time runs out.

1.3 Transfer between Negotiation Tasks

This section details the context and the proposed algorithms for automatically transferring between two or multiple negotiation tasks within the same domain. Next, a generic framework of negotiation strategy is first given, which support both transfer and non-transfer modes. The details of the learning in each of the source and the target tasks are then presented.

1.3.1 The Generic Framework of Agent-based Negotiation Strategy

In the present subsection, we detail the generic strategy framework. When the source task data is available, the strategy operates in the transfer learning mode to reuse knowledge from other negotiations tasks; in the other case, it works in the plain mode and decides its negotiation moves solely on the newly learnt model of the target task.

Upon receiving a counter-offer from the opponent at the time t_c, the agent records the time stamp t_c and the utility $U(O_{opp})$ of this counter-offer according to its own utility function. A small change in utility of the opponent can result in a large utility variation for the agent leading to a fatal misinterpretation of the opponent's behavior in the case of multi-issue negotiations. Therefore and in order to reduce that negative impact, the whole negotiation is divided into a fixed number (denoted as ζ) of equal intervals. The maximum utilities at each interval with the corresponding time stamps, are then provided as inputs to the learner *SPGPs* (for more details, see [24]). This also significantly scales down the computation complexity of modeling opponent so that the response speed is improved.

Then, dependent on whether the source task data is available, the agent learns the opponent differently as explained in Section 1.3.2 and 1.3.3, respectively. With the approximated opponent model, the agent adopts a linear concession strategy to avoid further computation load. More specifically, the optimal expected utility $\hat{u}$ (i.e., the expected maximum opponent concession) provided by the opponent model is used to set the target utility to offer at the current time t_c as follows:

$$u' = \hat{u} + (u_{\max} - \hat{u})(\hat{t} - t_c)^{\alpha} \tag{1.2}$$

where $u_{\max}$ is the maximum utility allowed in the negotiation task, $\hat{t}$ is the time when $\hat{u}$ will be reached, α the concession factor determining the way how the agent concedes, e.g., Boulware ($\alpha < 1$), Conceder ($\alpha > 1$) or Linear ($\alpha = 1$).

After the target utility u' has been chosen, the agent has to decide how to respond to the opponent's current counter-offer (this corresponds to line 14 in *Algorithm 1*). The agent first checks whether any of the following two conditions is fulfilled: 1) the utility of the latest counter-offer $U(O_{opp})$ is not smaller than u'; 2) the counter-offer has been proposed by the agent itself to its opponent at some earlier point during the ongoing negotiation process.

The agent settles the deal and the negotiation ends (line 12) if any of these two conditions is satisfied. Otherwise, the agent checks whether u' falls below the best counter-offer received so far. If this holds, then, for the consideration of efficiency, this counter-offer is proposed to the opponent. Proposing such an offer is reasonable because it tends to satisfy the expectation of the opponent. If not, the agent then constructs a new offer following a ε-greedy strategy as used in [3]. According to this strategy, a greedy offer is chosen with probability 1-ε in order to further explore the opponent behavior, and with probability ε a random offer is made (where $0 \leq \varepsilon \leq 1$). The greedy offer is chosen as follows. For a rational opponent, it is reasonable to assume that the sequence of its counter-offers is in line with its decreasing satisfaction. Thus, the more frequent and earlier a value of an issue j appears in counter-offers, the more likely it is that this issue contributes significantly to the opponent's overall utility. Formally, let $F(\cdot)$ be the frequency function defined by:

$$F^{n}(v_{jk}) = F^{n-1}(v_{jk}) + (1-t)^{\psi} \cdot g(v_{jk}) \tag{1.3}$$

where the superscript of $F(\cdot)$ indicates the number of negotiation rounds, ψ is the parameter reflecting the time-discounting effect, and $g(\cdot)$ is a two-valued function whose output is 1 if the specific issue value (i.e., v_{jk}) appears in the counter-offer and 0 otherwise. The new offer to be proposed is the one whose issue values have the maximal sum of frequencies according to the frequency function and whose utility is not worse than the current target utility. In the case of a random offer, an offer whose utility is within a narrow range around u' is randomly generated and proposed.

1.3.2 Learning in the Source Task

When no source tasks are at hand, the agent simply learns the current task using the plain mode of the proposed strategy. The source negotiation task starts by any side of the two parties presenting an offer describing values for the different negotiation issues. If an offer is accepted the negotiation session ends. On the other hand, if the offer is rejected the agent proposes a new offer to the other party. Then, the opponent

Algorithm 1 The generic framework of negotiation strategy is described. Depending on whether source data are available, it operates in two different modes, namely, the plain and transfer mode. The idea of knowledge transfer is to re-weight instances/models from the source task such that it helps the target task agent in learning the target opponent's behaviour model $\boldsymbol{\Theta}$.

```
1: Require: different distribution (i.e., source task) labelled data sets $\mathscr{T}_D$ (if available), same
   distribution data set $\mathscr{T}_s$, the base learning algorithm $\boldsymbol{\Xi}$, the maximum number of iterations $N$,
   maximum time allowed $t_{\max}$, and $t^{(k)}$ is the current time index.
2: while $t^{(k)} < t_{\max}$ do
3:     Collect time stamp and utility and add to $\mathscr{T}_S$
4:     Set n=size($\mathscr{T}_S$)
5:     if no source data available then
6:         $\boldsymbol{\Theta} \Leftarrow SPGPs(\mathscr{T}_S)$
7:     else
8:
           $$\boldsymbol{\Theta} \Leftarrow \begin{cases} TrAdaBoost.Nego(\mathscr{T}_D, \mathscr{T}_S, N) & \text{instance transfer} \\ ExpBoost.Nego(\mathrm{H}, \mathscr{T}_S, m, N) & \text{model transfer} \end{cases}$$
9:     end if
10:    $u' \Leftarrow setTargetU(\boldsymbol{\Theta}, \mathrm{n})$
11:    if Acceptable then
12:        an agreement reached
13:    else
14:        proposes a new offer
15:    end if
16:    Increment time index
17: end while
18: Terminate the negotiation session
```

can decide, according to her own behavior model, whether to accept or reject this new offer.

While the opponent's behavior model is unknown, it can be learned over time. The opponent model is indirectly observed from the utilities of the opponent's counter-offers: every time the opponent proposes a counter-offer, the utility of this offer is computed and added to the data set $\mathscr{T}_S = \{t^{(i)}, u^{(i)}\}_{i=1}^{t_{\max}}$, with $t^{(i)}$ representing the source task time steps running to a maximum of $t_{\max}$. The data set grows dynamically as the negotiation session continues. Every time a new instance is obtained, the model — in this case *SPGPs* — is trained anew to discover a new latent function best describing the new data set. The updated model is then used to propose a new offer to the opponent. This is achieved through the prediction probability distribution of the trained *SPGPs*. Formally, the predicted utility $u_\star$ at a new time step $t_\star$ is calculated according to the following:

$$p(u_\star|t_\star,\mathscr{D},\bar{\mathbf{X}}) = \int p(u_\star|t_\star,\bar{\mathbf{X}},\bar{\mathbf{f}})p(\bar{\mathbf{f}}|\mathscr{D},\bar{\mathbf{X}})d\bar{\mathbf{f}} = \mathscr{N}(u_\star|\mu_\star,\sigma_*^2), \qquad (1.4)$$

where

$$\mu_\star = \mathbf{k}_\star^T \mathbf{Q}_M^{-1}(\boldsymbol{\Lambda}+\sigma^2\mathbf{I})^{-1}u$$
$$\sigma_\star^2 = \mathbf{K}_{\star\star} - \mathbf{k}_\star^T(\mathbf{K}_M^{-1} - \mathbf{Q}_M^{-1})\mathbf{k}_\star + \sigma^2$$
$$\mathbf{Q}_M = \mathbf{K}_M + \mathbf{K}_{MN}(\boldsymbol{\Lambda}+\sigma^2\mathbf{I})^{-1}\mathbf{K}_{NM}$$

The negotiation session ends when either an agreement is reached or the available time steps are exhausted. Finally, the opponent's utility model described by the hyper parameters of the *SPGPs* is returned for later use.

1.3.3 Knowledge Transfer and Target Task Learning

1.3.3.1 Instance-based Transfer Approach

TrAdaBoost is originally designed for instance-based transfer learning, however it overlooks difference among source tasks. In order to make it well suited for the described negotiation setting, an extension of the standard *TrAdaBoost* algorithm is proposed to transfer instances between multiple negotiation tasks. This extended version is achieved by combining the principles of *TrAdaBoost* with the ideas of dealing with multi-task scenarios discussed in [27] and those of modifying a boosting-based classification approach for regression in [12], which together successfully results in the new regression algorithm *TrAdaBoost.Nego*. This new approach is specified in Algorithm 2.

TrAdaBoost.Nego requires two data sets as input. The first is $\mathscr{T}_D$ which represents the different distribution data set from one (or more) previous task(s) $\mathscr{T}_{D_k}$, where $\mathscr{T}_{D_k} \subseteq \mathscr{T}_D$. Since the source and the target opponent's attain their utilities from different distributions, then the different distribution data is that of the source data. Namely, $\mathscr{T}_{D_k} = \{t_k^{(i)}, u_k^{p(i)}\}_{i=1}^{|\mathscr{T}_{D_k}|}$, where $u_k^{p(i)}$ is the predicted source task's utility determined according to Equation 1.4.

The second data set required by the transfer algorithm is the same distribution data set $\mathscr{T}_S$. The same distribution data set is the one from the target task having time steps as inputs and received utilities as outputs. The instances of $\mathscr{T}_S$ are attained automatically from the initial negotiation steps in the target task, where the offer proposition step depends only on the source task's knowledge. In the present case, the behavior of the opponent is monitored similar to the learning in the source task and the utilities of the counter-offers are computed and added to $\mathscr{T}_S$. Please note, that the number of instances from the same distribution data set, $\mathscr{T}_S$, need not be large. In fact, it suffices for to be much less than the number of instances in $\mathscr{T}_D$ for the algorithm to perform well.

Having the above data sets, the weights of each of instances are fitted according to line 12 of Algorithm 2. The principles of weight-updating mechanism remain the

Algorithm 2 TrAdaBoost.Nego ($\mathscr{T}_D$, $\mathscr{T}_S$, N)

```
1: Require: source task data sets $\mathscr{T}_D = \{\mathscr{T}_{D_1}, \ldots, \mathscr{T}_{D_k}\}$, the target task $\mathscr{T}_S$, the base learning
   algorithm $\boldsymbol{\Xi}$, and the maximum number of iterations $N$.
2: $T = \mathscr{T}_{D_1} \bigcup \ldots \bigcup \mathscr{T}_{D_k} \bigcup \mathscr{T}_S$
3: $\mathbf{w} = (\mathbf{w}_{\mathscr{T}_{D_1}}, \ldots, \mathbf{w}_{\mathscr{T}_{D_k}}, \mathbf{w}_{\mathscr{T}_S})$
4: Initialize: the weight vector $\mathbf{w}^{(1)}(x_i) = \frac{1}{|T|}$, for $(x_i, c(x^{(i)})) \in T$. (Alternatively, the initial
   weights could be set with the user's knowledge.)
5: for $t = 1$ to $N$ do
6:     Set $\mathbf{p}^{(t)} = \mathbf{w}^{(t)}/Z^{(t)}$ ($Z^{(t)}$ is a normalizing constant)
7:     Learn a hypothesis $h_j^{(t)} : \mathscr{X} \to \mathscr{Y}$ by calling $\boldsymbol{\Xi}$ with the distribution $\mathbf{p}^{(t)}$ over the combined
       data set $\mathscr{T}_{D_n} \bigcup \mathscr{T}_S$.
8:     Compute the adjusted weighted prediction error of $h_i^{(t)}$ on each instance of $\mathscr{T}_S$ using:
       let $D^{(t)} = \max_{j=1}^{k} \max_{i=1}^{|\mathscr{T}_S|} |h_j^{(t)}(x^{(i)}) - c(x^{(i)})|$
       $e_{j,i}^{(t)} = \frac{|h_j^{(t)}(x^{(i)}) - c(x^{(i)})|}{D^{(t)}}$, where $(x_i, c(x^{(i)})) \in \mathscr{T}_S$
       $\varepsilon_j^{(t)} = \sum_{i=1}^{|\mathscr{T}_S|} \frac{\mathbf{w}_{\mathscr{T}_S}^{(t)}(i) e_{j,i}^{(t)}}{\sum_{q=1}^{|\mathscr{T}_S|} \mathbf{w}_{\mathscr{T}_S}^{(t)}(q)}$
9:     Choose the best hypothesis $\bar{h}^{(t)}$ such that the weighted error is minimal.
10:    Set $\beta^{(t)} = \frac{1}{2} \ln \frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}}$ and $\beta_j = \frac{1}{2} \ln(1 + \sqrt{2 \ln(|\mathscr{T}_{D_j}|/N)})$, where $\mathscr{T}_{D_j} \subseteq \mathscr{T}_D$
11:    Store $\bar{h}^{(t)}$ and $\beta^{(t)}$.
12:    Update the weight vector according to:
       $$\mathbf{w}^{(t+1)} \Leftarrow \begin{cases} \mathbf{w}_{\mathscr{T}_{D_j}}^{(t)}(i) e^{-\beta_j e_{j,i}^{(t)}} & \text{for } x_i \in \mathscr{T}_{D_j} \\ \mathbf{w}_{\mathscr{T}_S}^{(t)}(i) e^{\beta^{(t)} e_{j,i}^{(t)}} & \text{for } x_i \in \mathscr{T}_S \end{cases}$$
13: end for
14: Output: $h^{(f)}(x) = \sum_{t=1}^{N} \beta^{(t)} \bar{h}^{(t)}(x)$.
```

same as the original *TrAdaboost*. The proposed approach, however, no longer trains hypotheses by considering all source instances coming from the different distribution. Instead it generates a hypothesis for each of the source tasks, and then selects the one that appears to be the most closely related to the target. Specifically, at every iteration the i-th source task $\mathscr{T}_{D_i}$, independently from others, proposes a candidate hypothesis using a combined data set consisting of its own instances and those of the target task. The best hypothesis is then chosen such that the weighted error on $\mathscr{T}_S$ can be minimized. In this way, the impact of negative transfer caused by the imposition to transfer knowledge from a loosely related source task can be alleviated. In addition, the proposed extension adopts the way used in [12] to express each error in relation to the largest error at every iteration such that each adjusted error is still in the range [0, 1]. Although a number of loss functions are optional, our implementation employs the linear loss function as shown in line 8 because it is reported to work consistently well ([21]).

Once the extended version of *TrAdaBoost* algorithm fully fits the weights, the agent proposes an offer according to the predicted output of the target task function approximator in line 14 of Algorithm 2. After receiving a counter-offer, the utility of this offer is calculated and added to $\mathscr{T}_S$ so to be used in the next run of TrAdaBoost.Nego.

1.3.3.2 Model-based Transfer Approach

ExpBoost is one of widely used approaches for model-based transfer learning. Different from the instance-based transfer method, *ExpBoost* makes use of prior knowledge through a group of models $\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \dots, \boldsymbol{\Theta}_B$, trained on each source data sets separately. At every iteration, it trains a new hypothesis on the weighted instances of the target task, and then combines different models in order to improve performance even if the target opponent is mostly unknown. In contrast with the instance transfer approach, only the weights of target instances are re-calculated at each run of the algorithm, and the updating rule is according to the performance of the resulting combination of hypotheses. More precisely, target instances are given more importance when they are not correctly predicted. This is because they are believed to be more informative for the next iteration and help the learning algorithm to get better estimators. In the end, the approach returns the target opponent behavior model represented by the weighted median combination of the hypotheses.

When applying the model transfer algorithm *ExpBoost* to automated negotiations introduced in this work, two issues stand out. The first one has been discussed in [21], that is, at each boosting iteration, *ExpBoost* must select between either the newly learned hypothesis on the weighted target instances or a single expert from one of the source tasks, which potentially imposes restrictions on its learning ability. The second issue relates to the one we have already discussed before – how to modify the algorithm for regression. To solve the first issue, we consider to relax the hypothesis selection constraint by allowing a linear combination of the learnt hypotheses from source tasks and the additional hypothesis trained for the target task, to minimize the error at each iteration using least squares regression. To solve the second issue, the ideas of *Adaboost.R2* [12] to deal with the way of computing the adjusted error are then incorporated into the modified version of *ExpBoost*. The modified algorithm is referred to as *ExpBoost.Nego* and present it as Algorithm 3.

1.4 Experimental Results and Analysis

The performance evaluation was done with GENIUS (General Environment for Negotiation with Intelligent multi-purpose Usage Simulation [15]). Known as a famous simulation environment, GENIUS is also adapted by the international Automated Negotiating Agents Competition (ANAC) as the official competition platform. In this simulation environment an agent can negotiate with other opposing agents representing different strategies in a specified negotiation domain, where the utility function is defined by the preference of each negotiating party. The performance of an agent can be evaluated via its utility/score achievements.

Algorithm 3 ExpBoost.Nego (H, $\mathscr{T}_S$, m, N)

```
1: Require: the target task data sets $\mathscr{T}_S$ with size m, the pool of learnt model of previous tasks H
   = $\{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \dots, \boldsymbol{\Theta}_B\}$, the base learning algorithm $\boldsymbol{\Xi}$, and the maximum number of iterations
   $N$.
2: Initialize the weight vector $\mathbf{w}^{(1)}$ with each item equally being $\frac{1}{m}$.
3: for $t = 1$ to $N$ do
4:     Learn a hypothesis $h_{B+1}^{(t)} : \mathscr{X} \to \mathscr{Y}$ by calling $\boldsymbol{\Xi}$ with the distribution $\mathbf{w}^{(t)}$.
5:     $H_t = H \bigcup h_{B+1}^{(t)}$
6:     Find the linear combination $\bar{h^{(t)}}$ of the hypotheses from $H_t$, which minimizes squared error
       on instances of $\mathscr{T}_S$.
7:     Compute the adjusted prediction error $e^{(t)}$ of $\bar{h^{(t)}}$:
       $e_i^{(t)} = \frac{|\bar{h^t}(x^{(i)}) - c(x^{(i)})|}{D^{(t)}}$, where $D^{(t)} = max_{i=1}^{m} |\bar{h^t}(x^{(i)}) - c(x^{(i)})|$
8:     Calculate $\varepsilon^{(t)} = \Sigma_{i=1}^{m} \mathbf{w}_i^{(t)} e_i^{(t)}$
9:     If $\varepsilon^{(t)} \le 0$ or $\varepsilon^{(t)} \ge 0.5$, then stop.
10:    Let $\alpha^{(t)} = \frac{\varepsilon^{(t)}}{1-\varepsilon^{(t)}}$
11:    Update the weight vector $\mathbf{w}_i^{(t+1)} = \frac{\mathbf{w}_i^t (\alpha^t)^{1-e_i^{(t)}}}{Z^{(t)}}$, where $Z^{(t)}$ is a normalizing constant.
12: end for
13: Output: the weighted median of $\bar{h^{(t)}}(x)$ for $\lceil N/2 \rceil \le t \le N$, using the weight of $\ln(1/\alpha^{(t)})$.
```

Table 1.1 Overview of test negotiation domains.

Domain name	Number of issues	Number of values for each issue	Size of the outcome space
Energy	8	5	390,625
Travel	7	4-8	188,160
SuperMarket	6	4-9	98,784
Wholesaler	7	3-7	56,700
Kitchen	6	5	15,625
SmartPhone	6	4-6	12,000

Six domains with the largest outcome space[5] are chosen from the pool of test domains created for the ANAC competitions. For negotiations in large domains, finding an offer that is acceptable to both parties becomes more of a challenge than in a small domain in the sense it is much feasible in small domains to propose a large or even the whole proportion of the possible proposals during the negotiation. These domains are therefore more complicated and computational expensive to explore, placing a big demand on the efficacy of the proposed learning schemes. The main characteristics of these domains are over viewed in Table 1.1 (with a descending order of the size of outcome space).

Next, we first select those source tasks that are similar to the target task for the agent to operate the proposed methods. In so doing, the difference between the behavior models in the source and target tasks is made small such that transfer can be smoothly done. Then, we run tournaments composed of a range of the state-of-the-

[5] Outcome space of a domain refers to the number of possible agreements that could be agreed upon between participants.

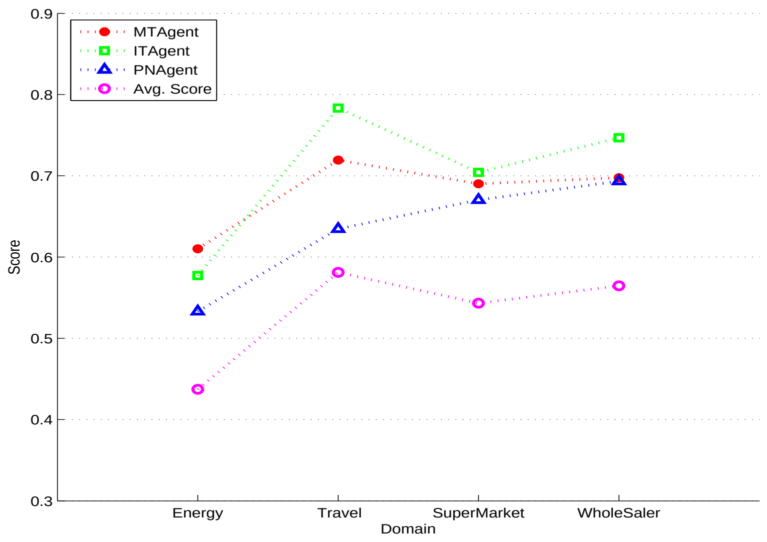

Fig. 1.1 Performance of the transferring and non-transferring agents in four domains when source and target tasks are similar. The average score refers to the mean score of all negotiating agents.

art agents. Moreover, the order of our agents encountering other opposing agents is random, in other words, there is no any guarantee about the similarity between source and target tasks in this setting. Such tournaments are able to provide a measure of effectiveness of negotiation strategies from a realistic perspective. Finally, the empirical game theory (EGT) analysis [18] is applied to the tournaments results. Through this analysis, the strategy robustness can be well examined, which enables us to have a clear view whether the transfer learning schemes also improve the robustness of the plain strategy, or not.

1.4.1 Similar Source and Target Negotiation Tasks

In this set of experiments, we evaluate the performance of the proposed methods given the transferred tasks are closely related to the target task. Toward this end, we manually choose those ANAC agents as the opposing agents in source tasks, which behave in a similar way with the opponent in the target task. To illustrate the difference of the two proposed learning schemes, we implement each algorithm described in Section 1.3.3 with a distinct agent, under name of ITAgent (for instance transfer) and MTAgent (for model transfer). The plain strategy (for the case of no transfer) is implemented by PNAgent.

The results are shown in Figure 1.1. Both the instance-transfer agent and model-transfer agents successfully achieved better performance than the agent using the plain strategy in the four domains. Moreover, the instance-transfer agent seemed to

Table 1.2 Overall tournament performance across all domains in descending order. The letter in bold of each strategy is taken as its identifier for the later EGT analysis.

Agent	*Normalized score*	*95% confidence interval*	
		Lower Bound	*Upper Bound*
ITAgent	**0.711**	**0.694**	**0.728**
MTAgent	0.704	0.687	0.721
AgentLG	0.697	0.679	0.716
CUHKAgent	0.684	0.664	0.704
OMACagent	0.677	0.661	0.694
PNAgent	0.674	0.651	0.698
HardHeaded	0.639	0.623	0.655
IAMhagg**L**er2011	0.588	0.581	0.595
Gahboninho	0.572	0.558	0.584

have a stronger negotiation power than the model-transfer agent. Another interesting observation is that the mean performance of all participants lagged far behind that of our agents. We suspect that is caused by the transfer effect since the improvement made by the transfer is at the cost of (more) benefit loss of the other party, especially when the test domains are fairly competitive.

1.4.2 Negotiation Tournaments

The first set of experiments, while successful, did not tell anything about how the developed strategies perform when the similarity between source and target tasks is not significant. In order to address this issue in a reasonable way, we carried out negotiation tournaments to observe agents' performance in a more realistic setting. In the tournament, each agent competes against all other agents in a random order. For each of the domains, we run a tournament consisting of the nine agents (e.g., the top three agents from respective ANAC 2012 and 2011 plus ITAgent, MTAgent and PNAgent) ten times to get results with high statistical confidence. For convenience of comparing performance across domains, normalization is adopted and done in the standard way, using the maximum and minimum raw score obtained by all agents in each domain.

According to the results given in Table 1.2, ITAgent and MTAgent, benefiting from the novel transfer learning schemes, took the first and second place, respectively. Both of them achieved a better score compared to the plain strategy (PNAgent), with each obtaining an improvement of 5.5% and 4.5%. Although ITAgent performed better than the other transferring agent, the difference was small (less than 1%). The three best agents of ANAC 2012 followed the two transferring agents, thus finishing third to fifth. The worst performers were those from ANAC 2011, whose performance was clearly below the two top agents, namely, 18% on average.

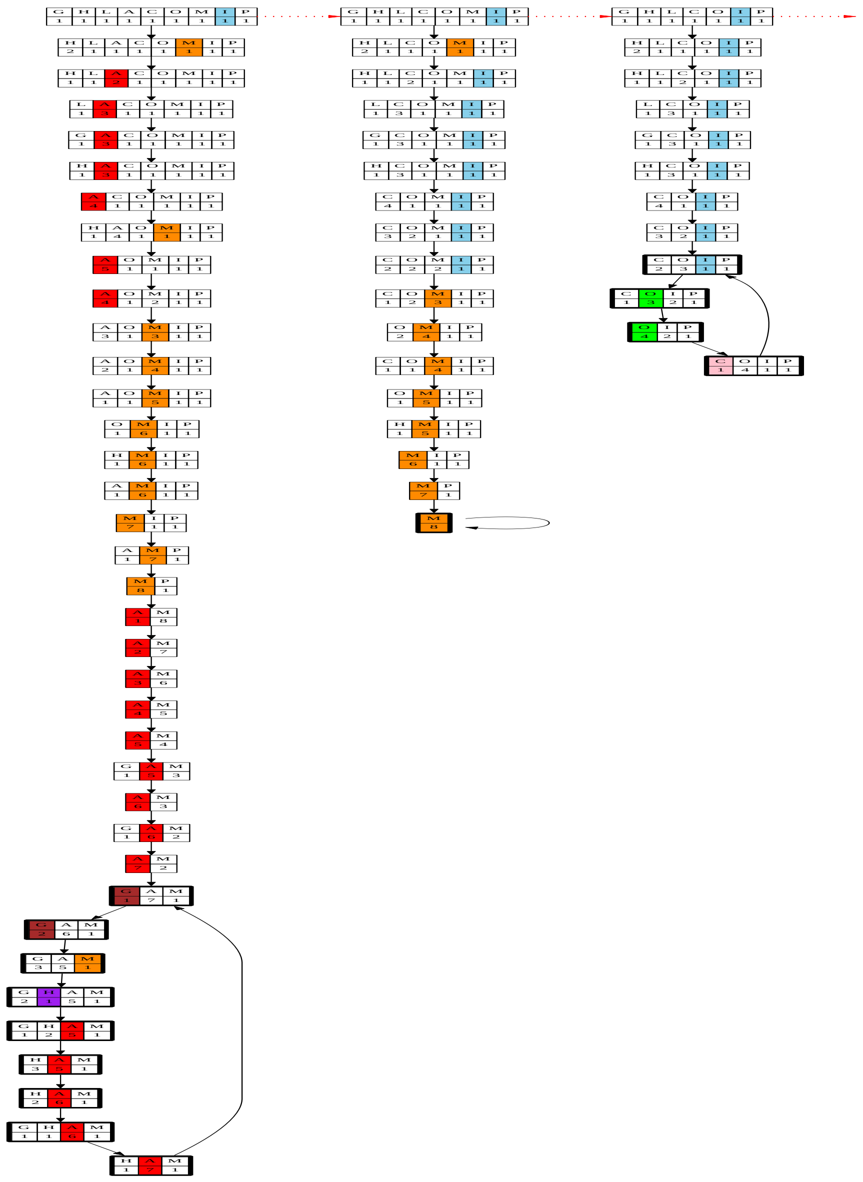

Fig. 1.2 The deviation analysis for the tournament results. Each node shows a strategy profile and the strategy with the highest scoring is highlighted by a background color. An arrow indicates statistically significant deviation from a different strategy profile. The stable states are those nodes with thinker border.

1.4.3 Empirical Game Theoretic Analysis

Till now, the performance of strategies was studied only from the traditional mean-scoring perspective. This, however, did not reveal information about the robustness of these strategies, for example, how would the results change if the players are free to switch their strategies to another one which could bring in a better individual payoff? To address robustness appropriately, the technique of empirical game theoretic (EGT) analysis [18] is applied to the tournaments results. The aim of using EGT is to search for pure Nash equilibria in which no agent has an incentive to deviate from its current strategy, or best reply cycle where there exist a set of profiles (e.g., the combination of strategies chosen by players) for which a path of deviations exists that connect them, with no deviation leading to a profile outside of the set. For brevity, the profiles in both Nash equilibria and best reply cycle are called stable states afterwards. Moreover, we consider deviations as those in [7] (the best single-agent deviation). In such a deviation one agent unilaterally changes the strategy in order to statistically improve its own profit, knowing the configuration of opponent strategies. The abbreviation for each strategy is indicated by the bold letter in Table 1.2. A profile (node) in the resulting EGT graph is defined by the mixture of strategies used by the players in a tournament. The first row of each node lists the three strategies, and the second row shows how many players use each strategy. The initial state of a tournament is the case where each player selects a distinct strategy. In spite of the fact that we cannot directly quantify the strategy robustness, we could rank the robustness by means of the relative sequence of a strategy being involved in stable states. Precisely, we initialize the EGT analysis with the whole set of strategies considered in our experiments. If there exist any stable state(s), the strategy attached (used) by most players is chosen as the most robust one of the current round. In the next round, we remove that winning strategy and restart the analysis to the remaining set of strategies. This process continues till all strategies are ranked or no stable state could be found in a certain round.

We visualize the results of the first three rounds under this EGT analysis in Figure 1.2. As can be seen, there is a best cycle in the first round, which consists of nine stable states. AgentLG represents the most popular strategy, which is used by a total number of 52 players in all stable states of the cycle. Thus, it is the robust strategy of the first round, and also the most robust one among all candidates. Then, we proceed to the second round as indicated by the bold dotted line on the top of the figure. The only difference between the first and second round is that the robust strategy of the prior round has been excluded. The case of the second round is simpler since there exists an unique Nash equilibrium where all players attach to MTAgent. In the third round of the analysis, we again find a best cycle consisting of four states. OMACagent is voted by a number of 10 players, ITAgent attracts six players, and the other two strategies have equal less votes. As a result, OMACagent is the robust strategy selected for the third round.

The final ranking of strategy robustness is illustrated in Table 1.3. Surprisingly, the most stable strategy is AgentLG, even though it failed to reach the highest score achievement in the tournaments. This is because this strategy manages to demonstrate more comprehensive negotiation ability in the sense it is capable of winning more negotiations (even with a relatively smaller advantage). By contrast, ITAgent is merely in the fifth place despite being the best performer of the previous tour-

Table 1.3 The relative robustness ranking.

Agent	*Ranking*	*Round*	*Attracted Players*
AgentLG	1	1	52
MTAgent	2	2	8
OMACagent	3	3	10
PNAgent	4	4	4
ITAgent	5	5	5
CUHKAgent	6	6	4
IAMhagg**l**er2011	7	7	3
HardHeaded	8	8	2
Gahboninho	9	9	1

naments. MTAgent is more robust than the other transferring agent and the plain strategy, finishing in the second place.

The EGT analysis suggests that the proposed model-transfer learning method is robust, and seemingly has the potential of being applied in a wider range of scenarios than the instance-transfer learning method.

1.5 Related Work

Opponent modeling is assumed to be of key importance to performing well in automated negotiations [23]. Learning in the negotiation setting is however hampered by the limited information that can be gained about an opponent during a single negotiation session. To enable learning in this setting, various simplifying assumptions are made. For example, Lin et al. [19] introduce a reasoning model based on a decision making and beliefs updating mechanism which allows the identification of the opponent profile from a set of publicly available profiles. Another work [1] investigates online prediction of future counter-offers by using differentials, thereby assuming that the opponent strategy is defined using a combination of time- and behavior-dependent tactics [13]. Hou [16] employs non-linear regression to learn the opponent's decision function in a single-issue negotiation setting with the assumption that the opponent uses a tactic from the three tactic families introduced in [13]. [2] use a three-layer artificial neural network to predict future counter-offers in a specific domain, but the training process requires a large amount of previous encounters, which is time-consuming as well as computationally expensive.

Recent work has started to focus on learning opponent's strategy in a more practical situation (i.e., without those simplifying assumptions). Some good examples are [3,6,26]. Williams et al. [26] apply Gaussian processes regression model to optimize an agent's own concession rate by predicting the maximal concession that the opponent is expected to make in the future. This approach, known as IAMhaggler2011, made the third place in ANAC 2011. Another successful GPs-based approach is described in [3], where the authors model opponents by means of sparse pseudo-input Gaussian processes to alleviate the computational complexity of opponent modeling. Hao and Leung [14] propose the winning strategy of ANAC 2012. This strategy attempts to avoid concession as much as possible by adjusting the so-called non-exploitation time point. Moreover, it employs reinforcement-learning to increase the acceptance probability of its own proposals. Chen and Weiss [6] develop a negotiating agent, under name of OMAC, which aims at learning an opponent's strategy by analyzing its behavior through discrete wavelet transformation and cubic smoothing spline. OMAC also adapts its concession behavior in response to uncertainties in the environment. OMAC performed better than the five best agents of ANAC 2011 and was awarded the third place in ANAC 2012. Although these methods are proven useful in certain scenarios, they all suffer from insufficient training data when facing an unknown opponent in a new encounter, which results in a more or less inefficient learning process.

1.6 Conclusions and Future Work

This paper proposes the first robust and efficient transfer learning algorithms in negotiation tasks. The transfer technique makes use of adaptation of *TrAdaBoost* and *ExpBoost* — two well known supervised transfer algorithms — to aid learning against a new negotiating opponent. Two strategy variants were proposed. In the first the target task agent makes decisions based on weighting the instances of source and target tasks, while in the second, the agent decides its moves depending on a weighting of the source and the target models. Experimental results show the applicability of both approaches. More specifically, the results show that the proposed agents both outperform state-of-the-art negotiating agents in various negotiation domains with a considerable margin. They further demonstrate that the model transfer learning is not only a boost to the strategy performance, but also results in an improved strategy robustness.

There are several promising future directions of this work. First, the quantification of *negative transfer* is an essential next research step. Furthermore, a thorough analysis of the relationships between the agents' preferences and strategies in order to get a better understanding of transferring behavior among negotiating opponents. This understanding is also of relevance with respect to maximizing transfer stability. Last but not least, the extension of the proposed framework and strategies toward concurrent negotiations is an important issue of practical relevance that needs to be explored. In such settings, an agent is negotiating against multiple opponents simultaneously. Transfer between these tasks can serve as a potential solution for optimizing the performance in each of the negotiation sessions.

Acknowledgements We greatly appreciate the fruitful discussions with our colleagues at the Department of Knowledge Engineering, Maastricht University, especially those suggestions from Kurt Driessens, Haitham Bou Ammar and Evgueni Smirnov.

References

1. J. Brzostowski and R. Kowalczyk. Predicting partner's behaviour in agent negotiation. In *Proceedings of AAMAS '06*, pages 355–361. ACM, 2006.
2. R. Carbonneau, G. E. Kersten, and R. Vahidov. Predicting opponent's moves in electronic negotiations using neural networks. *Expert Syst. Appl.*, 34:1266–1273, February 2008.
3. S. Chen, H. B. Ammar, K. Tuyls, and G. Weiss. Optimizing complex automated negotiation using sparse pseudo-input Gaussian processes. In *Proceedings of AAMAS'2013*, pages 707–714. ACM, 2013.
4. S. Chen, H. B. Ammar, K. Tuyls, and G. Weiss. Using conditional restricted boltzmann machine for highly competitive negotiation tasks. In *Proceedings of the 23th Int. Joint Conf. on Artificial Intelligence*, pages 69–75, Beijing, China, 2013. AAAI Press.
5. S. Chen, J. Hao, G. Weiss, K. Tuyls, and H.-f. Leung. Evaluating practical automated negotiation based on spatial evolutionary game theory. In C. Lutz and M. Thielscher, editors, *KI 2014: Advances in Artificial Intelligence*, volume 8736 of *Lecture Notes in Computer Science*, pages 147–158. Springer International Publishing, 2014.
6. S. Chen and G. Weiss. An efficient and adaptive approach to negotiation in complex environments. In *Proceedings of ECAI'2012*, pages 228–233. IOS Press, 2012.
7. S. Chen and G. Weiss. An efficient automated negotiation strategy for complex environments. *Eng. Appl. Artif. Intell.*, 26(10):2613 – 2623, 2013.
8. S. Chen and G. Weiss. An intelligent agent for bilateral negotiation with unknown opponents in continuous-time domains. *ACM Trans. Auton. Adapt. Syst.*, 9(3):16:1–16:24, Oct. 2014.
9. S. Chen and G. Weiss. An approach to complex agent-based negotiations via effectively modeling unknown opponents. *Expert Systems with Applications*, 42(5):2287 – 2304, 2015.
10. R. M. Coehoorn and N. R. Jennings. Learning on opponent's preferences to make effective multi-issue negotiation trade-offs. In *Proceedings of the 6th Int. conf. on Electronic commerce*, pages 59–68, New York, NY, USA, 2004. ACM.
11. W. Dai, Q. Yang, G. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.
12. H. Drucker. Improving regressors using boosting techniques. In *Proceedings of ICML '97*, pages 107–115, 1997.
13. P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Rob. Autom. Syst.*, 24(4):159–182, 1998.
14. J. Hao and H. Leung. ABiNeS: An adaptive bilateral negotiating strategy over multiple items. In *Proceedings of the 2012 IEEE Conference on IAT*, pages 95–102, 2012.
15. K. Hindriks, C. Jonker, S. Kraus, R. Lin, and D. Tykhonov. Genius: negotiation environment for heterogeneous agents. In *Proceedings of AAMAS'2009*, pages 1397–1398, 2009.
16. C. Hou. Predicting agents tactics in automated negotiation. In *Proceedings of the 2004 IEEE Conference on IAT*, pages 127–133, 2004.
17. N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: prospects, methods and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.
18. P. R. Jordan, C. Kiekintveld, and M. P. Wellman. Empirical game-theoretic analysis of the tac supply chain game. In *Proceedings of AAMAS'2007*, pages 1188–1195, 2007.
19. R. Lin, S. Kraus, J. Wilkenfeld, and J. Barry. Negotiating with bounded rational agents in environments with incomplete information using an automated agent. *Artificial Intelligence*, 172:823–851, April 2008.
20. S. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, 2010.
21. D. Pardoe and P. Stone. Boosting for regression transfer. In *Proceedings of the 27th international conference on Machine learning*, pages 863–870, 2010.
22. A. Rettinger, M. Zinkevich, and M. Bowling. Boosting expert ensembles for rapid concept recall. In *Proceedings Of AAAI'2006*, volume 21, pages 464 – 469, 2006.

23. A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):97–109, January 1982.
24. E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances In Neural Information Processing Systems*, pages 1257–1264. MIT press, 2006.
25. M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, December 2009.
26. C. Williams, V. Robu, E. Gerding, and N. Jennings. Using gaussian processes to optimise concession in complex negotiations against unknown opponents. In *Proceedings of IJCAI'2011*, pages 432–438. AAAI Press, 2011.
27. Y. Yao and G. Doretto. Boosting for transfer learning with multiple sources. In *2010 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1855–1862. IEEE, 2010.