Auton Agent Multi-Agent Syst (2016) 30:60–81
DOI 10.1007/s10458-015-9281-9

NegoChat-A: a chat-based negotiation agent with bounded rationality

Avi Rosenfeld · Inon Zuckerman · Erel Segal-Halevi · Osnat Drein · Sarit Kraus

Published online: 17 February 2015

Abstract To date, a variety of automated negotiation agents have been created. While each of these agents has been shown to be effective in negotiating with people in specific environments, they typically lack the natural language processing support required to enable real-world types of interactions. To address this limitation, we present NegoChat-A, an agent that incorporates several significant research contributions. First, we found that simply modifying existing agents to include an natural language processing module is insufficient to create these agents. Instead, agents that support natural language must have strategies that allow for partial agreements and issue-by-issue interactions. Second, we present NegoChat-A's negotiation algorithm. This algorithm is based on bounded rationality, and specifically anchoring and aspiration adaptation theory. The agent begins each negotiation interaction by proposing a full offer, which serves as its anchor. Assuming this offer is not accepted, the agent then proceeds to negotiate via partial agreements, proposing the next issue for negotiation based on people's typical urgency, or order of importance. We present a rigorous evaluation of NegoChat-A, showing its effectiveness in two different negotiation roles.

Keywords Human-agent interactions · Negotiation · Algorithms

A. Rosenfeld (✉)
Department of Industrial Engineering, Jerusalem College of Technology, 91160 Jerusalem, Israel
e-mail: rosenfa@jct.ac.il

I. Zuckerman
Department of Industrial Engineering and Management, Ariel University, 40700 Ariel, Israel
e-mail: inonzu@ariel.ac.il

E. Segal-Halevi · O. Drein · S. Kraus
Department of Computer Science, Bar-Ilan University, 52900 Ramat Gan, Israel
e-mail: erelsgl@gmail.com

O. Drein
e-mail: osnatairy@gmail.com

S. Kraus
e-mail: sarit@cs.biu.ac.il

1 Introduction

Negotiation is a basic task that is intertwined into our daily lives. Most negotiations are mundane, such deciding on a meeting time or even convincing our children to do their homework or eat their vegetables. However, they can also have dramatic effects on the lives of millions, such as negotiations involving inter-country disputes and nuclear disarmament [1].

Due to the importance of the negotiation task, it is unsurprising that a variety of agents have been previously created to negotiate with people within a large range of settings. These agents have addressed issues including: the number of parties, the number of interactions, and the number of issues to be negotiated [2–6]. As the next section details, two key common elements exist in all of these previous agents. First, agents based on notions of equilibrium or optimal methods often did not successfully reach agreements with people and thus alternatives needed to be found [4]. As such, they developed effective negotiation agents that used elements taken from bounded rationality. For example, the QOAgent used the maximin function and the qualitative valuation of offers instead of equilibrium strategies. Second, all agents needed mechanisms for dealing with incomplete information. This is typically done through reasoning about the negotiating partners by learning their preferences and strategies [7–9]. Unfortunately, as has been previously noted [4], natural language processing (NLP) abilities are lacking from current state-of-the-art negotiation agents. This inherent limitation requires people interacting with these agents to be "forced" into interacting via menus or other non-natural interfaces.

To address these limitations, this paper presents NegoChat-A, the product of a research project aimed to push the envelope of automated negotiation research by moving from menu-driven interfaces to chat-based environments. NegoChat-A is specifically designed to help people interact more naturally with agents within repeated interactions. In doing so, it incorporates the following key contributions: First, NegoChat-A integrates natural language into its agent, allowing people to practice his or her negotiation skills from anywhere while still moving away from the more simplistic and less intuitive menu interfaces. Second, NegoChat-A used two bounded rationality theories to help build successful incremental agreements. The first theory, and the key difference between NegoChat-A and the previously developed NegoChat agent [10], is its use of anchoring as the start of all negotiation interactions. Using anchoring has been previously noted to be effective within human-agent negotiations [11,12]. NegoChat-A not only makes use of anchoring for its initial offer, but also "reanchors" its offer if a period of time elapses where it sees that no negotiation progress has been made as no offers have been made by or received from its negotiation partner (e.g. the person). Second, assuming the person has engaged the agent by presenting a counter-offer, the agent then incrementally builds agreements with people, based on aspiration adaptation theory (AAT) [13], a second bounded rationality theory. Accordingly, NegoChat-A first learns the aspiration scale offline, or the order in which people typically discuss issues. During negotiation sessions, the agent uses this ordering to proceeds to propose a value for the issue most people typically aspire for next. This allows NegoChat-A to successfully build incremental agreements with people, something that current automated negotiators do not do [4–6]. Third, we present a rigourous evaluation of NegoChat-A in two negotiation settings demonstrating, how and why it performs better than the current state-of-the-art agent KBAgent [6].

This paper is structured as follows. We first provide background of previous automated negotiation agents and their limitations in Sect. 2. Section 3 presents work on anchoring and an overview of AAT [13] – two bounded rationality theories upon which several key elements of NegoChat-A are built. Section 4 presents the negotiation domain, including

details about the repeated nature of the interactions and the incomplete information within the domain, and presents the KBAgent which is used for evaluation comparison. NegoChat-A also integrates key elements of this state-of-the-art agent, but, as Sect. 5 describes, its negotiation algorithm builds agreements based on anchoring and AAT. Section 6 presents how NegoChat-A generates and responds to natural language, allowing people to practice negotiation in this more intuitive format. Section 7 explores the strengths and weaknesses of NegoChat-A, showing that it achieves significantly better agreements in less time than the current state-of-the-art KBAgent. Section 8 discusses strengths of NegoChat-A and provides future directions. Section 9 concludes.

2 Related work

This paper's contribution lies in its description and evaluation of NegoChat-A, a novel negotiation agent that integrates natural language (chat), anchoring and aspiration adaptation theory (the -A portion of the name) to help facilitate an effective automated negotiation agent. Previous extensive studies in the field of human computer interactions have noted that the goal of any system should be an intuitive interface with the stress being put on creating agents which operate in environments which are as real and natural as possible [14,15]. Thus, following these approaches, it is critical to develop natural language support for negotiation agents to allow for these types of "normal" interactions [16]. This form of typing as natural interaction is referred to as natural-language interaction (NLI) in the literature. There have been numerous informal tests of NLI systems, but few controlled experiments have compared them against a design without NLI support [17].

While automated negotiation agents have been developed for quite some time, even state-of-the-art negotiation agents unfortunately do not yet consider how natural language impacts the agent's strategy. Examples include Byde et al.'s *AutONA* automated negotiation agent [2]. Their problem domain involves multiple negotiations between buyers and sellers over the price and quantity of a given product. Jonker et al. [3] created an agent to handle multi-attribute negotiations which involve incomplete information. However, none of these agents support NLI, and a different interface might profoundly impact performance. In contrast, Traumet et al. [18] presented a model of conversation strategies which they implemented in a virtual human in order to help people learn negotiation skills. Their virtual human can negotiate in complex adversarial negotiation with a human trainee. Their main effort was toward the Dialogue Manager (DM) module and the virtual character module. Their strategy module used rule-based strategies. The focus of our work is on the strategy module that follows anchoring and AAT but our DM is not as advanced as theirs and we do not provide a virtual character. However, we do study how the agent's strategy must be modified to incorporate NLP, and we conduct an exhaustive experimental study on how the strategy we develop is better than the state-of-the-art. As their focus in not on the agent's strategy, they do not report such a test. Another advantage of our agent is its ability to be used by anyone that wants to practice her negotiation skills, from anywhere, without installing any complicated software.

QOAgent [5] is a domain independent agent that can negotiate with people in environments of finite horizon bilateral negotiations with incomplete information. The negotiations consider a finite set of multi-attribute issues and time-constraints. KBAgent builds upon this agent and currently represents the state-of-the-art automated negotiation agent. It also considers negotiations with a finite set of multi-attribute issues and time-constraints, and has been shown to be the most effective agent in achieving agreements with people in several domains

involving multiple attributes [6]. As NegoChat-A is built on several elements of this agent, we provide an overview of how it operates, including how it learns from previous data and how it decides if it should accept proposals.

Multi-attribute negotiation continues to be an important research challenge, with one active research avenue being the ANAC (Automated Negotiating Agents Competition) Workshop. Since 2010, this competition has focused on agents that use the GENIUS[1] interface [19,20]. However, even to date, this competition focuses on agent-agent interactions and the interface supports only menu-based interactions between agents and people. Much research within this field studies exclusively agent to agent negotiations.

It was previously shown [21] that existing negotiation algorithms must be modified to support robust real-world interfaces with people, such as chat. Specifically, it was shown that simply adding chat support to the current state-of-the-art KBAgent was not effective. To address this limitation, we study what logical extensions are needed, if any, to make existing negotiation agents suitable for natural language. Previous economic and behavior research into people's negotiation would suggest that the current agent approach of attempting an agreement on all issues simultaneously will not be effective. For example, Bac and Raff [22] found that simultaneously negotiating a complete package might be too complex for individual buyers. Furthermore, they show that in the context of incomplete information with time discount, the more informed player ("strong" in their terminology) will push towards issue-by-issue negotiation. Busch and Horstmann [23] found that some people might like to decide all issues at once, while others prefer to decide one by one. Chen [24] studied issue-by-issue negotiation with an opt-out factor, and argues that when the opt-out probability is low, agents prefer to negotiate a complete package because intuitively we know that the negotiations can last long enough so that agents can get to a "win-win" situation. However, with a high opt-out probability, agents prefer issue-by-issue negotiation. Sofer et al. have shown that an alternative negotiation scheme which takes an issue-by-issue approach can theoretically form the best protocol between two self-interested rational agents [25]. Thus, one key contribution of this paper is its study of how people react to agents that do not propose issue-by-issue agreements.

3 Anchoring and aspiration adaptation theory

Given the realization that people prefer to negotiate issue-by-issue, we present NegoChat-A, a negotiation algorithm that successfully does this. NegoChat-A begins all negotiation interactions by presenting a full offer which serves as the agent's anchor, or its starting point. Previous research had found that anchors are important within negotiations between people [11,12]. Kahneman previously demonstrated how the initial discussion point, or a negotiation's anchor, can impact the outcome of people's negotiation sessions. Similarly, Kristensen et al. found that when people perceived the initial offer as a gain rather than as a loss they subsequently accepted worse offers, made fewer counter-offers, and encountered fewer negotiation impasses [12]. Based on these works, we reasoned that automated agents could also achieve better agreements with people by selecting an effective anchor. Our previous version of NegoChat-A, which we called NegoChat [10] lacked anchoring. We correct this issue in this version of the agent and, as we present in Sect. 7, we found that anchoring does in fact aid in automated negotiation, as the NegoChat-A agent outperformed NegoChat.

[1] The GENIUS and ANAC websites can be reached though http://mmi.tudelft.nl/negotiation/index.php/Genius.

Assuming that the automated agent's initial anchor is not accepted, as the person negotiating with the agent only accepted parts or none of the anchoring offer, NegoChat-A uses AAT [13] to incrementally build agreements in the remaining issues. Following our previous work, we found that people use key elements from AAT in their negotiations, even when optimal solutions, machine learning techniques for solving multiple parameters, or bounded techniques other than AAT could have been implemented [26]. The premise of AAT is that certain decisions, and particularly our most complex decisions, are not readily modeled based on standard utility theory. For example, assume you need to relocate and choose a new house in which to live. There are many factors that you need to consider, such as the price of each possible house, the distance from your work, the neighborhood and neighbors, and the schools in the area. How do you decide which house to buy? Theoretically, utility based models can be used. However, most of us do not create rigid formulas involving numerical values in order to weigh trade-offs between each of the search parameters. AAT is one way to model this and other similar complex problems.

Despite its lack of utility to quantify rational behavior, AAT is an approach by which bounded agents address a complex problem, $\mathcal{G}$. The complexity within $\mathcal{G}$ prevents the problem from being directly solved, and instead an agent creates n goal variables $G_1, \ldots, G_n$ as a means for solving $\mathcal{G}$. Agents decide how to set the goal variables as follows: The m goal variables are sorted in order of priority, or the variables' *urgency*. Accordingly, the order of $G_1, \ldots, G_n$ refers to goals' urgency, or the priority by which a solution for the goal variables should be attempted. Each of the goal variables has a desired value, or its *aspiration level*, that the agent sets for the current period. Note that the agent may consider the variable "solved" even if it finds a sub-optimal, yet sufficiently desired, value. The agent's search starts with an initial aspiration level and is governed by its *local procedural preferences*. The local procedural preferences prescribe which aspiration level is most urgently adapted upward if possible, second most urgently adapted upward if possible, etc., and which partial aspiration level is *retreated from* or adapted downward if the current aspiration level is not feasible. Here, all variables except for the goal variable being addressed are assigned values based on ceteris paribus, or all other goals being equal, a better value is preferred to a worse one.

For example, referring to the above example of searching for a new home, one would like to find the biggest home, for the cheapest price, in the best location. However, realistically, people have aspiration scales which dictate the order of importance of these issues. One person may be willing to settle for a smaller house in a better location (and thus aspire for location more than price) while another person might have the opposite preferences. Thus, during the search process, while every person will first attempt to maximize all goal variables, assuming this is found to not be possible, they will retreat from, or settle on some of the goals, say by selecting a smaller home, yet satisfy the other goals, such as the home's price and location. AAT provides a framework for quantifying this process. Accordingly, a person might form an *urgency* of the house's price as being the most important goal variable, its size as being the second most important variable, and its location being the third most important variable. Agents, including people, create an *aspiration level* for what constitutes a "good enough" value for these variables. Borrowing from Simon's terminology [27] there is only an attempt to "satisfice" the goal values, or achieve "good enough" values for these values instead of trying to optimize or truly solve them. If it is not possible to achieve all aspirations, the original values will need to be reexamined, or retreated from. Thus, the agent may retreat, or reconsider raising the budget they allocated for buying their house, the size they are willing to settle on, or its location. We now describe NegoChat-A's negotiation algorithm, and specify how anchoring and AAT is used.

4 The negotiation domain

As has been previously described [5], we formally define the negotiation domain as follows: We study *bilateral* negotiation in which two players negotiate to reach an agreement on conflicting issues. The negotiation can end either when (a) the negotiators reach a full or partial agreement, (b) one of the player opts out (denoted as OPT), thus forcing the termination of the negotiation with a predefined opt-out outcome, or (c) a time limit is reached, that results in a predefined status-quo outcome (denoted as SQ).

The negotiations resolve around *multi-attribute* characteristics. There is a set of n issues, denoted as I, and a finite set of values, $domain(i)$ for each issue $i \in I$. An agreement is denoted as $\alpha \in O$, and O is a finite set of values for all issues. The negotiations are sensitive to the negotiation time that impacts the benefits of the negotiating parties. Formally, we define the negotiation rounds $Rounds = \{0, ..., dl\}$, where dl is the deadline limit, and use R to denote the current round. The time duration of each round is fixed and known to both negotiators. Each player is assigned a time cost which influences its benefits as time passes. While in theory the time effect may be negative or positive, we assume it to be negative and therefore it represents a time discount factor.

The negotiation *protocol* is fully flexible. As long as the negotiation has not yet ended, each side can propose a possible agreement, reject a previously offered agreement, opt-out of the negotiation, or communicate a threat, promise, or any general remark. In contrast to the model of alternating offers [28], each player can perform as many interactions with the negotiation player as he wishes.

Last, we consider environments with *incomplete information*. That is, players are not fully aware of the scoring structure of their opponents. We assume that there is a finite set of scoring structures which will be referred to as player types. For example, scoring type might model a long term orientation regarding the final agreement, while another might model a more constrained orientation. Formally, we denote the possible types of the players $Types = \{1, \ldots, k\}$. Given $l \in Types$, we refer to the scoring of the player of type l as S_l, and $S_l : \{O \cup \{SQ\} \cup \{OPT\}\} \times Rounds \rightarrow \mathcal{R}$.[2] Each player is fully aware of its own scoring function, but it does not know the exact type of its negotiating partner.

4.1 The KBAgent

The state-of-the-art automated negotiator for playing against boundedly rational opponents (such as humans) is the KBAgent [6]. It has been shown that KBAgent negotiates efficiently with people and achieves better utility values than other automated negotiators. Moreover, KBAgent achieves significantly better agreements, quantified by these agreements' scores, than the human counterparts playing the same role.

The main difference between KBAgent and other agents is its use of a general opponent model to help shape the offers it will propose. Specifically, KBAgent utilizes past negotiation sessions of other agents as a knowledge base for the extraction of the likelihood of acceptance and offers which will be proposed by the other party. That data is used to determine which offers to propose and accept. One of its main advantages is that it can also work well with small databases of training data from previous negotiation sessions.

[2] We intentionally use the term score instead of utility because the score is the number of points the player can attain from a game's outcome. While our agent maximizes the scoring function, this is not necessarily the case for human players. People's utility functions may be based on factors not encapsulated by the scoring function. As we cannot model these unknowns, we refer to score and not utility.

In order to generate an offer, KBAgent creates a list of offers ordered by their respective QOValues, which quantitatively ranks relative offers and is an alternative to the Nash bargaining solution. The QOValue is based on the player's score and the probability of their acceptance by the other party. It is defined as:

$$\begin{aligned} QOValue(\mathbf{o}) &= min\{\alpha_o, \beta_o\} \\ \alpha_o &= rank_a(\mathbf{o}) * lu_a(\mathbf{o}) \\ \beta_o &= [lu_a(\mathbf{o}) + lu_b(\mathbf{o})] * rank_b(\mathbf{o}) \end{aligned}$$

Where $rank()$ is the ranking value of the offer based on the given scoring function for a given setting (normalized to [0, 1]), and $lu()$ denotes the Luce number of an offer (see [5] for the exact definitions).

KBAgent generates a list of concession offers to be made for every round of the negotiation. The concession offers that are generated change with regard to the round number, as the agent is more willing to reach less beneficial agreements as the round increases. Specifically, offers are ordered by their relative QOValue such that the agent first proposes the offer with the maximal QOValue. The concession list is built based on a database consisting of previous interactions between human players. The agent then uses the average score of previously accepted offers to determine the list of agreements that will be generated for any given timeframe. For more detail, we refer the reader to previous work [6].

There are two main disadvantages to this approach. First, KBAgent only proposes one concession offer per round, and if this offer is rejected, it does not attempt to generate a similar offer during the current round. For the domain they studied, each round lasted 2 minutes, which we found constituted lost opportunities to reach agreements. Thus, if the other side rejects the one offer the KBAgent provides during a given round, an agreement can only be reached if the other side takes the initiative to offer an alternate above the learned threshold. Second, the KBAgent never offers partial offers. We found this was a disadvantage as people found it difficult only to receive full offers [21].

5 The negotiation algorithm

The goal of the NegoChat-A agent is to reach an agreement, α, that maximizes its own score. Following AAT terminology, NegoChat-A creates n goal variables, $G_1, \ldots, G_n$, representing the number of issues needing to be negotiated within the set I. For each goal variable G_k we associate domain values $domain(G_k)$. We use V_k to denote a value in $domain(G_k)$, α_k to denote the value of G_k according to α, and $S(\alpha, R)$ to denote the agent's score from α at round R.

Each value $V_1, \ldots, V_n$ is originally initialized to NULL to represent that no agreement initially exists for the values within G_k. As the two sides negotiate, values for $V_1, \ldots, V_n$ become non-null. An offer is defined as a *full offer* if it has non-null values for $V_1, \ldots, V_n$. An offer is a *partial offer* if $V_1, \ldots, V_n$ contains between 1 and n-1 non-null values. We assume that a full agreement must have non-null values for all values, $V_1, \ldots, V_n$. Note that while the agent's goal is to maximize its own score, an agreement will likely only be achieved through compromise with the person with whom it is negotiating as the person is self-interested and also wishes to maximize the score of her offer.

5.1 NegoChat-A's algorithm

The agent begins by proposing a *full offer*, FA, as its anchor. FA is chosen by selecting the values for $V_1, \ldots, V_n$ with the highest value within the agent's **search cluster**. NegoChat-A builds this search cluster by grouping offers with similar QOValues. Formally, after constructing a concession list using the KBAgent's algorithm [5], at round R we define the **search cluster**, $SC(R)$, as all offers which provide NegoChat a score of at least $S(\alpha_R, R)$ but still yield its opponent a score not lower than $S_{opponent}(\alpha_R, R) - \epsilon$. We arbitrarily set the ϵ to be 200. Thus, the search cluster is a set of offers with all offers in it having QOValue values that are *equal to or lower than* the round R offer's QOVAlue in the original concession list. The assumption behind building this range of possible offers is that offers with too low a value will not be acceptable to the other party as their value is too low, and offers will an insufficiently high value are not acceptable to the agent.

NegoChat-A's **search cluster** is an alternative to KBAgent's concession list. KBAgent's list is built around the assumption that people act rationally, and thus creates a set of offers that are Pareto-optimal for any given time. However, we claim that boundedly rational humans will often be happy to accept offers that provide *similar or lower* utility values than offers presented to them via KBAgent's concession list. Strong cultural or psychological biases of the human player might motivate her to specific values for a specific issue even at the expense of other issues and their values. For instance, in a job-interview domain, a person might prefer (aspire for) a high salary value to a high value for her promotion possibilities. As was previously shown [26], the preferences exhibited by a person are independent of the actual objective score of the issues in the domain. For this reason, the search cluster, which includes sub-optimal bids (taken from the human's perspective) allows for better flexibility for the agent in the face of these human biases when conducting negotiations.

Throughout the course of the negotiation process, the agent must generate offers and decide how to respond to offers. The NegoChat-A agent creates two types of offers: the previously mentioned FA anchoring offer and partial offers. Assuming the person does not accept FA, the agent then notes if any issues were agreed upon. For example, assuming the person says, "I cannot agree to this value for issue x", then we assume all other issues have been agreed upon. As per AAT, the remaining unagreed upon goal variables are ordered by their aspiration scale, whereby the remaining issues that need to be negotiated are ordered by how important these issues are to a typical person within that domain. This ordering is pushed onto a stack, A, where the most aspired for issue (G_k) is at the top of the stack, the second most aspired for issue is the second position from the top, etc. In our approach, we choose this ordering based on previous offline learning from historical data of what issues were typically discussed first, second, third, etc. In theory, other methods of preference elicitation are possible, but this issue is left for future work. Assuming that the aspiration scale is not known, the agent randomly chooses an aspiration scale. NegoChat-A will suggest the maximal value V_k for G_k given that this value is contained within at least one full offer in the agent's search cluster. For example, assume that the agent wishes to discuss the salary issue based on the aspiration scale it learned. It will then propose the maximal value for V_k that exists within its search cluster of all full offers. However, if the search cluster is empty as no possible offers exist with valid values for V_k, the agent will have to retreat, or revisit previous agreed upon issues as no full agreement is possible given the values it had agreed upon for these issues. This retreat process forms a type of backtracking. Please note that NegoChat-A does not learn or utilize any connection between issues. For example, while it may be advantageous to present multiple issues together within an offer as the agent may have identified dependencies between issues, NegoChat-A adheres to AAT's stepwise

search approach and limits itself to only one issue at a time. Conversely, while it may be advantageous to retreat, or backtrack in a more sophisticated fashion, we again leave this issue open for future work.

Ideally, FA offers are only presented once at the very beginning of the negotiation session, after which incremental offers are built based on AAT for the remaining non-agreed upon issues. In practice, we considered a third possibility where the person didn't respond to the agent's offer and also did not generate an offer of their own. In these cases, we assume that the negotiation session has gone "cold" for whatever reason (e.g. the person got distracted) and a new full offer should be set as a new anchor since the previous negotiation interaction had timed out.

As the negotiation interaction progresses, the agent and person agree upon a value V_k, either from the initial anchor offer or subsequent partial offers. The agent then pushes all agreed upon V_k values onto a stack B. Note that in cases where the person agrees, or even partially agrees to FA, the agent will add multiple values to B. In incrementally building subsequent offers, the agent only proposes one issue at a time. In these cases, the agent pops one value from A to determine which goal variable G_k should be discussed next. An agreement is reached when A is empty, as this indicates that no more issues need to be discussed.

5.2 NegoChat-A algorithm definitions

We present three algorithms that comprise NegoChat-A's strategy, which we now explain in greater detail. Algorithm 1 provides the base of the agent, controls how the agent handles offers it receives and generates offers. This is done by keeping two stacks, A and B. Stack A contains the aspiration scale of G_1, …, G_n and represents the order by which the agent should negotiate issues. Stack B is initially set to be empty and represents issues that have been resolved (line 1). The agent then proposes its anchor offer (line 2) and awaits a response from the person. The remainder of the algorithm (lines 3–11) controls the timing of the agent and how it creates and responds to offers. Until the negotiation deadline has been reached, the agent first creates SC, its search cluster. Every round, the agent recalculates the search cluster until the negotiation deadline is reached (lines 3 and 4). This loop is terminated before this point, either by the sides reaching an agreement or the other side opting out (lines 5–7). As our agent never opts out, this can only happen from the person's initiative. As there is no cost for the agent in sending multiple offers, the agent could theoretically send thousands if not millions of proposals every negotiation round. However, as one would expect, people did not appreciate a deluge of offers and we found by trial-and-error that they do not appreciate more than 1 offer every 25 seconds. Thus, if the agent notes 25 seconds of inactivity (line 10), it will then proceed to again issue a full offer to serve as a new anchor for the negotiation (line 11). However, if the person did respond within this time (lines 8–9), it will proceed to respond and incrementally build a full agreement through partial offers as per the following algorithms.

Algorithm 2 receives as its input an offer α for m goal variables, $G_1, \ldots, G_m$, which have the corresponding values V_{k_1}, …, V_{k_m}. Note that m can be the trivial case of 1 issue or n representing all issues (e.g. m <=n). Assuming that every value for every goal issue is contained within the search cluster (line 1), this is an acceptable offer, and the agent responds with an accept message (line 2). The NLP module described in the next section is responsible for generating a natural language response. Once an agreement is reached, the agreed upon values, V_{k_1}, …, V_{k_m}, are added to stack B (line 3), and the agreed upon issues, G_{k_1}, …, G_{k_m}, are removed from stack A (line 4). As previously mentioned, the agent will

Algorithm 1 NegoChat-A

```
1: Create stacks A and B, A ← aspiration scale, B ← ∅
2: ProposeOffer(n, V_1, ..., V_n)
3: while R <= DEADLINE do
4:   SC ← SearchCluster(R)
5:   while NOT(AGREEMENT) do
6:     if AGREEMENT OR OPT-OUT then
7:       EXIT
8:     if Received-Offer(V_{k_1}, ..., V_{k_m}) then
9:       HandleOffer(V_{k_1}, ..., V_{k_m})
10:    if Time − Elapsed > timeout-threshold then
11:      ProposeOffer(n, V_1, ..., V_n)
```

then proceed to incrementally build an agreement with the person by proposing a value for the next most aspired for issue (line 5). This facilitates building incremental offers with the person. However, if the person's offer is not found within the agent's search cluster, it sends a reject message and then proposes an alternative for these issues that is contained within the search cluster (lines 7–8) as per Algorithm 3.

Algorithm 2 HandleOffer($V_{k_1}, \ldots, V_{k_m}$)

```
1: if ∃ (α ∈ SC s.t. ∀k_i, α_{k_i} = V_{k_i} AND ∀V_i ∈ B, α_i = V_i) then
2:   Send-Accept(V_{k_1}, ..., V_{k_m})
3:   Push(V_{k_1}, ..., V_{k_m}, B)
4:   Remove(G_{k_1}, ..., G_{k_m}, A)
5:   ProposeOffer(1, Top(A))
6: else
7:   Send-Reject(V_{k_1}, ..., V_{k_m})
8:   ProposeOffer(m, G_{k_1}, ..., G_{k_m})
```

Algorithm 3 creates the agent's offers. Note that when this algorithm is initially called by the agent in line 2 of Algorithm 1 or for reanchoring discussions in line 11 of Algorithm 1, the anchor offer FA is presented with all n issues. However, in incrementally building offers as per line 5 of Algorithm 2, this algorithm is called to build offers by only proposing one issue at a time ($m = 1$). In all cases, assuming an offer can be found based on these values in the search cluster, SC (line 3), then the proposal is sent as natural language (line 4). Specifics of this module are again described in the next section. Assuming this search cluster is empty, there is nothing for the agent to do except wait for the next time R, during which the search cluster will be recalculated. Assuming the search cluster contains different possibilities, it offers the agreement with the highest score from within the search cluster.

If this offer is accepted, the agent adds these values to B (line 6) and removes these issues from A (line 7). Otherwise, the agent removes all offers with this value (line 9). For example, again assume you wish to buy a house and are in negotiations to purchase a specific house. Your goal variables are: price, payment schedule and occupancy date, and your agent offers a price of 200,000 Euro for a house. If the other side rejects your offer, the agent assumes that all agreements with the same price will also be rejected. Thus, it removes these similar agreements from the search cluster (line 9). If, in another example, you already agreed on the price of 180,000 EURO and your agent offered the seller a payment plan with 4 payments over the next 4 months, and the offer was rejected, NegoChat-A will remove from the search clusters only the agreements that include both the price of 180,000 and 4 payments. Next,

NegoChat-A will look for a different payment schedule to offer consistent with 180,000 EURO (back at line 2). If there is no additional agreement in the search cluster that includes 180,000 EURO for the price, it will backtrack (retreat) to the last agreed upon issue (line 11) and then explore it again as the next issue to pursue (line 12) within the next iteration of Algorithm 1. Referring back to our example, if the person accepted a bid of 180,000 Euro for a house and with this price the agent agrees to a payment plan with four payments, once the offer of four payments is rejected, NegoChat-A will renegotiate the price.

Algorithm 3 ProposeOffer(m, $G_{k_1}, \ldots, G_{k_m}$)

1: **if** SC $\neq \emptyset$ **then**
2: $\alpha \leftarrow \arg\max \{\{S(\alpha', R)\} \mid \alpha' \in \text{SC}, \forall V_i \in B, \alpha_i = V_i\}$
3: **if** $\alpha \neq$ NULL **then**
4: Send-Offer($\alpha_{k_1}, \ldots, \alpha_{k_m}$)
5: **if** Received-Accept($\alpha_{k_1}, \ldots, \alpha_{k_m}$) **then**
6: Push($\alpha_{k_1}, \ldots, \alpha_{k_m}$, B)
7: Remove($G_{k_1}, \ldots, G_{k_m}$, A)
8: **if** Received-Reject($\alpha_{k_1}, \ldots, \alpha_{k_m}$) **then**
9: SC = SC $\setminus \{\alpha' \in \text{SC} \mid \forall k_i \ \alpha'_{k_i} = \alpha_{k_i}$ AND $\forall V_i \in B, \alpha_i = V_i\}$
10: **else**
11: $V_{last} \leftarrow$ Pop(B)
12: Push(G_{last}, A)

6 NegoChat-A's natural language

Our natural language system has a standard dialog system architecture [29], described in Fig. 1. We illustrate the system with a running example from our experimental domain, where the human is an employer and the agent is a job-candidate, and they negotiate over the candidate's job conditions (described further in the next section). Nonetheless, the system itself is general and can be applied to support chat in any system.

The natural language system is composed of several components. The **Natural Language Understander** (NLU) translates the human sentences from natural language to a set of *dialog acts* that represents the users intentions. We represent our dialog acts in the standard JSON format.[3] For example, the human utterance "I accept your salary offer, but only if you work for 10 hours", is translated to a set of two dialog acts: [[{Accept:Salary}, {Offer:{Hours:10}}]]. The NLU is described in detail in Sects. 6.1 and 6.2.

The **Dialog Manager** (DM) has several responsibilities: First, it interprets the human dialog acts based on the current dialog state. For example, it interprets the dialog act {Accept:Salary} based on the salary value in the most recent offer made by the agent, and converts it to an explicit Offer. Second, it responds to human dialog acts that are not directly related to negotiation, such as greetings and questions. Third, it notifies the agent when the human dialog acts are related to negotiation. For example, if one of the human's dialog acts is an offer, then the DM sends a "Received-Offer" notification (see Algorithm 1), and if the human has accepted a full offer, the DM sends a "Received-Accept" notification (see Algorithm 3). Fourth, it controls the timing of conversation. For example, if the human hasn't done anything in a pre-specified time interval (e.g. 25 seconds), then the DM asks

[3] www.json.org.

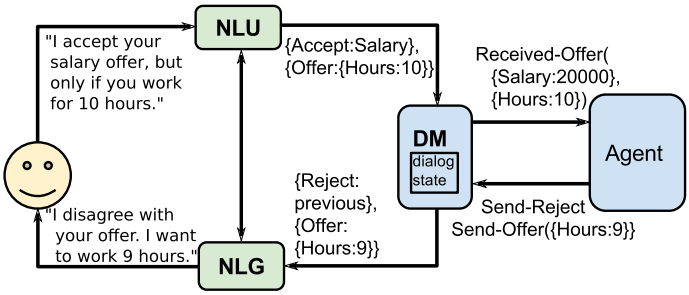

Fig. 1 Dialog system architecture. Example starts at the *top-left* corner

the agent to make an action, e.g., repeat the previous offer or make a new offer. Fifth, it receives commands from the agent and translates them to dialog acts. For example, if the agent issues a Send–Reject command (from Algorithm 2), then the DM creates the dialog act {Reject:previous}. Finally, it combines several dialog acts to a single set. For example, if the agent issues a Send-Reject command, and shortly after that, a Send-Offer command (from Algorithm 3), then the DM creates a set with two dialog acts [{Reject:previous}, {Offer:...}].

The **Natural Language Generator** (NLG) translates the set of dialog acts, created by the DM, to a single natural language sentence, that is sent to the human. Our NLG works in cooperation with our NLU in order to create human-like sentences, as we describe in detail in Sect. 6.3.

6.1 Natural language understander (NLU)

Our NLU component is a **multi-label classifier** (MLC)—a classifier that returns a set of zero or more labels for each input sample. The set of possible labels is the set of dialog acts recognized by our DM, whose total number is 58. They have a hierarchical structure, for example: {Offer:{Salary:20,000}} and {Offer:{Hours:9}} are two different dialog acts. The top level of the hierarchy contains eight different labels: {Offer, Accept, Reject, Append, Insist, Query, Quit, Greet}. In order to take advantage of the hierarchical structure of the dialog acts, we used the HOMER approach (Hierarchy of Multi-label classifiERs, [30]). In this approach, there is a different MLC for each level of the hierarchy. The input sentence is first sent to the top-level MLC, which returns a subset of the top-level labels, e.g., {Offer, Query}. Then the sentence is sent in parallel to all relevant second-level MLCs, e.g., the Offer MLC and the Query MLC. The Offer MLC returns a set of second-level labels from the set relevant to Offer (i.e. Salary, Hours, etc.), and the MLC for Query returns a set of second-level labels from the set relevant to Query. This process continues until the leaves of the hierarchy are reached. Then, the replies of all MLCs are combined to produce the final set of dialog acts.

For the MLCs in each node of the HOMER, we used the One-versus-All approach: each MLC is a collection of binary classifiers, one for each label. For each input sentence, it runs each binary classifier in turn, and returns the set of labels whose classifier returned "true". As the base binary classifier, we used Modified Balanced Winnow [31]—a classifier that supports online training and real-time classification.[4]

[4] The state-of-the-art in NLU for dialog systems is sequence classification [32]. We decided against this option because it requires too much labeling effort: while in multi-label classification you only need to label each sentence, in sequence classification you must label each **fraction** of a sentence. After deciding to use

An input sentence goes through several pre-processing components before it arrives at the MLC. The **normalizer** converts numbers and other common phrases in the input sentence to canonical format. The **splitter** splits the sentence around punctuation marks and creates several sub-sentences. We found out that this simple heuristic greatly improves the performance of the MLC. The **feature extractor** creates a feature vector from each sub-sentence. As features, we use unigrams and bigrams (pairs of adjacent words).[5] As feature values we use the standard TF/IDF metric. The resulting feature vectors are the inputs to the MLC.

6.2 Development and training

As a first step in adding natural language capabilities, we manually wrote a single natural language sentence for each dialog act supported by the agent. This facilitated the coordination between the team working on the agent and the team tagging the training data, and made sure they both understand the negotiation acts in the same way. We also used these sentences as an initial training set for the MLC.

Using this initial NLU component, we let our agent speak with students and Amazon Mechanical Turk workers.[6] During these preliminary experiments, one of the developers acted as a "Wizard-of-Oz": through a web-based GUI, he viewed each set of dialog acts produced by the NLU component, and could edit it before it was sent to the DM. He could also immediately train the classifier with each new sentence, thanks to its fast training abilities. During the online learning process, the sentence-level accuracy of the NLU componenet improved from 18 % (with only the initial 58 manually-written sentences) to 72 % (with 775 tagged sentences).[7]

The total time spent by the Wizard-of-Oz was about 5–10 h. This means that it is relatively cheap to adapt the system to new negotiation domains. In future work we plan to reduce this cost even further using semantic resources such as a textual-inference engine [39].

6.3 Natural language generator (NLG)

The NLG takes as input a set of dialog acts produced by the DM, and returns a natural language sentence that is sent to the human. Usually, NLGs are based on manually-written templates. In contrast, our NLG uses the NLU's training data in the reverse direction. For each dialog act, the NLG asks the NLU for a random sentence tagged with exactly this dialog act, and combines the received sentences to a single output sentence. This approach has several advantages,

Footnote 4 continued
MLC, we did preliminary experiments in which we tried various state-of-the-art approaches to MLC [33–37]. We also tried several kinds of base binary classifiers (Support Vector Machines, Bayesian Classifiers and Random Decision Forests), a classifier based on language models [38] and a spell-checker. We found out that the combination of HOMER with Modified Balanced Winnow, described above, had the best performance in terms of both classification accuracy and run-time.

[5] We tried more sophisticated features, such as pairs of non-adjacent words, but this didn't improve performance.

[6] We used Amazon Turk as a convenient way to get sentences for training the NLU component. We could have used other ways, such as letting experts invent sentences and tag them. However, based on past experience we decided that using Turk is much cheaper. The total cost of gathering data from 22 people was only approximately $30 dollars.

[7] Sentence-level accuracy is the number of sentences whose classification was exactly correct (i.e. the set of dialog acts returned by the MLC is identical to the correct set), divided by the total number of sentences. The 72 % accuracy was calculated using fivefold cross-validation on the set of 775 tagged sentences. Sentence-level accuracy is the strictest possible performance measure. In other measures, such as precision and recall, the performance of our NLU was higher.

which we exemplify with several actual examples from our experiments: First, the agent's replies are versatile, even when the strategy demands that it repeats the same offer again and again. For example: the agent says "I would like to work for 20,000", and 25 s later, "I need to make 20,000". Second, the agent's replies are human-like. They even contain spelling and grammar mistakes that occur naturally in chat conversations between humans. Third, some of the agent's replies contain reasoning and argumentation. For example: "i would like a 20,000 salary. this is mandatory to me to have a good salary as i believe working conditions affect directly my effectiveness" (sic). While we hope to study how to explicitly reason about the use of argumentation in the agent, we currently do not select particular arguments based on context. Instead, we obtain these arguments as part of the sentences uttered by humans in the NLU component that are randomly selected. Last, the agent continuously learns new ways to express itself during the NLU's online learning process.

7 Experiment design and analysis

The main goal of this research was to push the envelope of automated negotiators research by moving from menu-driven interfaces to chat-based environments. As this work transitions from the fruitful work of previously developed agents, [6,19,20], we intentionally chose to base ourselves on these agents and the complex environments they had studied. Thus, we shied away from dealing with overly simplified settings, such as those with full information, single issues, or alternating turn-based offers, and instead considered a complex problem with partial information, multi-attribute negotiations, and an unconstrained interaction protocol.

In order to properly evaluate the influence of natural language input on automated negotiation agents, we picked the *job candidate* domain used in previous research [5,6]. In this domain, a negotiation takes place after a successful job interview between an employer and a job candidate. In the negotiation both the employer and the job candidate wish to formalize the hiring terms and conditions of the applicant. Below are the issues under negotiation: **[Salary]**: This issue dictates the total net salary the applicant will receive per month. The possible values are {7,000, 12,000, 20,000}. **[Job description]**: This issue describes the job description and responsibilities given to the job applicant. The possible values are {QA, programmer, team manager, project manager}. **[Social benefits]**: The social benefits are divided into two categories: company car and the percentage of the salary allocated, by the employer, to the candidate's pension funds. The possible values for a company car are {leased car, no leased car, no agreement}. The possible values for the percentage of the salary deposited in pension funds are {0, 10, 20 %, no agreement}. **[Promotion possibilities]**: This issue describes the commitment by the employer regarding the fast track for promotion for the job candidate. The possible values are {fast promotion track (2 years), slow promotion track (4 years), no agreement} **[Working hours]**: This issue describes the number of working hours required by the employee per day (not including over-time). The possible values are {8, 9, 10 h}.

In this scenario, a total of 1,296 possible agreements exist ($3 \times 4 \times 12 \times 3 \times 3 = 1,296$). Each turn in the scenario equates to two minutes of the negotiation, and the negotiation is limited to 15 rounds of 2 min each (30 min total). If the sides do not reach an agreement by the end of the allocated time, the job interview ends with the candidate being hired with a standard contract, which cannot be renegotiated during the first year. This outcome is modeled for both negotiators as the status quo outcome. Each side can also opt-out of the negotiation if it feels that the prospects of reaching an agreement with the opponent are slim and it is impossible to negotiate anymore. Opting out by the employer entails the

postponement of the project for which the candidate was interviewing, with the possible prospect of its cancelation and a considerable amount of expenses. Opting-out by the job candidate will make it very difficult for him to find another job, as the employer will spread his/her negative impression of the candidate to other CEOs of large companies. Time also has an impact on the negotiation. As time advances the candidate's score decreases, as the employer's good impression has of the job candidate decreases. The employer's score also decreases as the candidate becomes less motivated to work for the company. To facilitate incomplete information there are three possible score structures for each side, which models a long-term candidate, short-term candidate and compromising candidate.

7.1 Experiment design

For our experiments we developed a negotiation system with a chat interface and natural language processing mechanism (see Fig. 2). We have implemented both agents in this system: the current state-of-the-art KBAgent and the newly developed *NegoChat-A* agent.[8] To decide upon the aspiration scale we first ran 16 trials of two people negotiating with each other within the system (32 people total). While these experiments focus on people from Israel, we have since begun studying other cultures and their impact on negotiation styles. Towards this goal, we have also studied similar pilots to find the aspiration scales within the United States and Egypt. Within both Israel and the U.S. we noted that on average people discussed issues in the same order of: Salary, Job Description, Pension, Working Hours, Car Benefit, and Promotion Possibilities. In Egypt the aspiration scales were different with the ordering being: Job Description, Salary, Car Benefit, Pension, Promotion Possibilities, and Working Hours. An interesting open question is when can learning one set of people's preferences—here aspiration scales—be transferred between different groups as evidently is possible between the U.S. and Israel, and when it is not possible as is evident with the Egyptian population. However, this paper focuses on the large population of Israeli participants described below.

We then evaluated the NegoChat-A agent with a total of 126 different Israeli participants within the employer-employee scenario described above to study the strengths and weaknesses of the NegoChat-A as compared to the state-of-the-art KBAgent. All participants were divided randomly to play against either the KB or NegoChat-A agents. Furthermore, we had the agent play both the roles of the employer and employee with the human participant taking the other role. In all, 60 of the 126 people negotiated with the KBAgent and NegoChat-A agents with the agent playing the candidate role and the person playing the role of the employer. Of these 60 people, 27 participants played against the KBAgent and a different 33 people played the same role against the NegoChat-A agent. The remaining group of 66 people played the role of the employer against the agent with 30 of these people playing the role of the candidate against KBAgent and another 33 people playing the same role against the NegoChat-A agent. The participants were typically students between the ages of 21 and 30 with 86 male and 40 female participants. They were motivated through receiving bonus points to their course grade as a function of their final score in the session.

Before starting the negotiation task, all participants were given a full tutorial about the task at hand, the interface and the possible score functions. A short test was issued to verify that the participants understood the instructions and task at hand. The participants did not know any details regarding the automated agent with which they were matched, or the fact that it was not a human player. The outcome of each negotiation task was either reaching a full agreement, opting out or reaching the deadline.

[8] A copy of the KBAgent and NegoChat-A agents are found at: http://biu-ai.com:4014/demo.

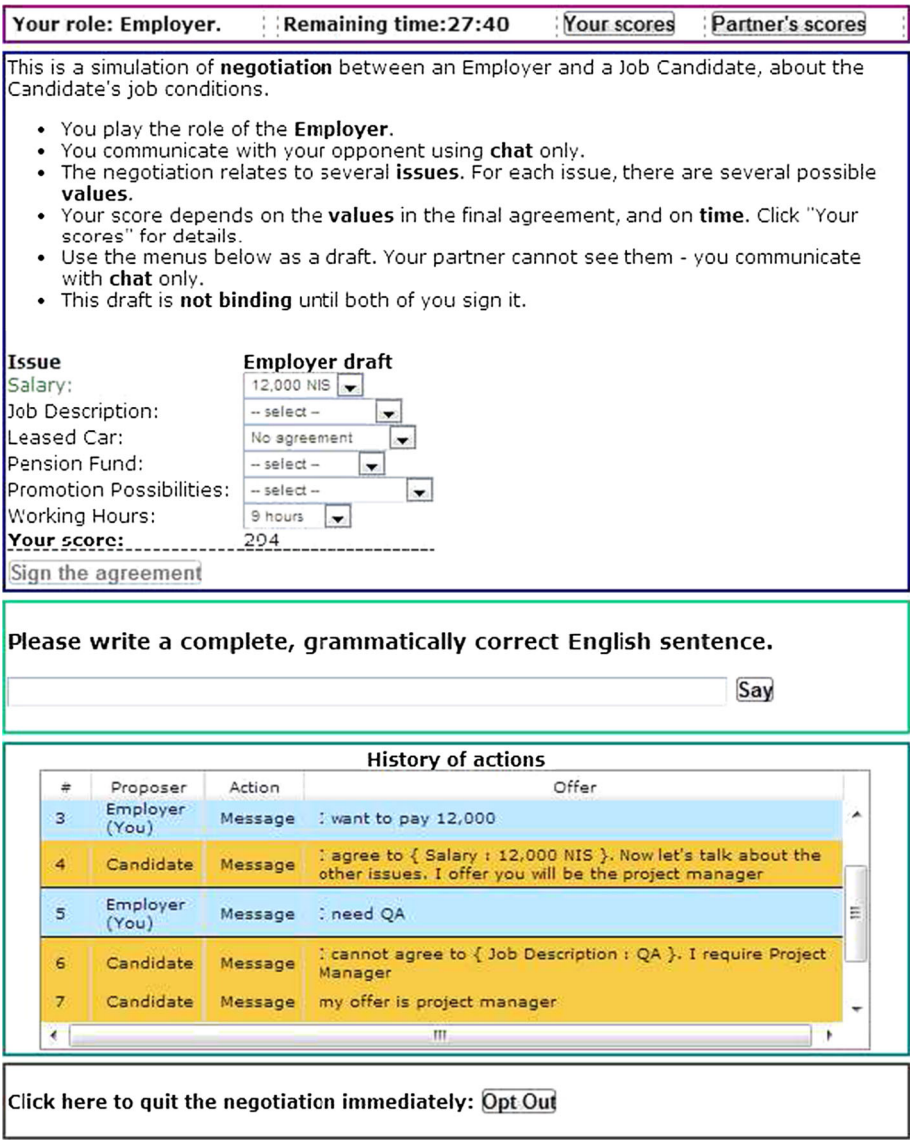

Fig. 2 The negotiation system's interface

7.2 Experimental results

Our first result, found in Table 1, is that the NegoChat-A agent significantly outperformed the state-of-the-art KBAgent when playing the candidate side. As can be seen from the first two rows of the table, when considering all trials including cases where one of the sides opted out, NegoChat-A achieved on average significantly better scores (p-score 0.033 in a two-tailed t-test) and reached agreements in less time (p-score 0.0002 in a two-tailed t-test; less time is better). We also considered analyzing only those cases where agreements were reached and

Table 1 Results of NegoChat-A versus KBAgent negotiation experiments in Israel—agent as candidate

All trials	Person's utility	Agent's utility	Social welfare	Time (sec)	# Games
KBAgent	362.44	409.55	771.99	1,001.29	27
NegoChat-A	333.09	**464.27**	797.36	**622.87**	33
Agreements only					
KBAgent	**378.24**	435.12	813.36	1,017.4	25
NegoChat-A	341.61	**485.32**	826.93	**655.48**	31

Bold values represent statistically significant improvements

Table 2 Results of NegoChat-A versus KBAgent negotiation experiments in Israel—agent as employer

All trials	Person's Utility	Agent's utility	Social welfare	Time (sec)	# Games
KBAgent	314.3	355.73	670.03	1,243	30
NegoChat-A	331.5	**407**	738.5	**953.12**	33
Agreements only					
KBAgent	368.68	398.32	767	1,172.6	25
NegoChat-A	344.4	420.8	765.2	1,005.67	30

Bold values represent statistically significant improvements

removed the sessions were a person chose to opt-out of the negotiation. The results of these cases are found in the last two rows of Table 1. All statistically significant differences between the algorithms are bolded. Please note that the NegoChat-A agent achieved significantly better agreements (p-score 0.0063) in less time (p-score 0.0006). However, people playing against KBAgent on average did better, with the score for the person in the agreement-only cases being significant (p-score 0.01). This implies that some of NegoChat-A's success is evidently at the cost of the person's score and consequently the social welfare score of this agent is not significantly better than that of KBAgent. As our goal is to maximize the agent's utility this should not be seen as a fault. However, future generations of automated agents may decide to implement different strategies to maximize social welfare. We leave this point for future work.

Our second result, found in Table 2, is that the NegoChat-A agent also significantly outperformed the state-of-the-art KBAgent when playing the employer side of the scenario. As can be seen from the first two rows of the table, NegoChat-A achieved on average significantly better scores (p-score 0.048 in a two-tailed t-test) and reached agreements in less time (p-score 0.0004 in a two-tailed t-test). We again also separately analyzed the scores of the sessions with agreements, the results of which are found in the last two rows of the table. However, here the higher scores of NegoChat-A were not significant (p-score 0.055 for the time differences of 0.19 for the agent score differences), likely due to the relatively large number of opt-outs within the KBAgent (5 here versus 2 in Table 1). NegoChat-A again reaches agreements faster (1,005.67 versus 1,172.6 seconds), but not significantly so (p-score 0.08). Please note that the social welfare is nearly the same for both agents in the agreement-only cases. We again point out that this leaves interesting future work for developing against that aim to maximize social welfare.

We then considered how the anchoring element within the NegoChat-A agent improved performance above that of the previously published NegoChat agent [10]. In general, NegoChat-A differs from the previous NegoChat agent in its use of anchoring. Recall from

Table 3 The incremental improvement over three generations of Chat agents—agent as candidate

All trials	Person's utility	Agent's utility	Social welfare	Time (sec)	# Games
NegoChat	312.5714	451.6429	764.2143	662.3214	31
Nego-I	319.25	467.4375	786.6875	601.25	16
NegoChat-A	344.0714	469.5357	813.6071	647.3214	31
Agreements only					
NegoChat	323.9231	480.6154	804.5385	625.9615	28
Nego-I	319.25	467.4375	786.6875	601.25	16
NegoChat-A	349.4815	481.963	831.4444	665.9259	30

Algorithm 1 that NegoChat-A uses anchoring in two situations – for its initial offer (in line 2) and for timeout situations when the person has not responded to its previous offers or counter-offers (line 12). In both of these cases, the NegoChat agent would only have offered a value for the next most aspired for issue, like it still does in line 5 of Algorithm 2, to make incremental agreements. To study the impact of each of these stages, we considered three variations of the NegoChat algorithm. The first variation is the previously published agent [10] that has no anchoring. The second variation, an intermediary version, which we refer to as Nego-I, uses anchoring for only the initial offer (in line 2 of Algorithm 1) but not afterwards (as per line 12 of the algorithm). The final version is the NegoChat-A algorithm published in this paper and it uses anchoring in both cases.

A comparison of all three of these variations is found in Table 3 where we again present the results including opt-outs in the first two rows of the table and the agreement-only cases in the last two rows. As we had hypothesized, the Nego-I variation does perform better than the original NegoChat agent in all categories except the time to reach agreements. The NegoChat-A agent does even better than the Nego-I agent, and surpasses the NegoChat agent in all categories including the time to reach agreements. While the improvements in each of these categories are not statistically significant, they do lend some support as the incremental improvement of anchoring it lends to the original NegoChat agent. Furthermore, while the original NegoChat agent performs better than KBAgent, we did not find this improvement to be significant (p-score 0.14). In contrast, both the Nego-I and NegoChat-A agents significantly outperform the KBAgent baseline (p-score 0.048 and 0.033 respectively), providing additional support of the superiority of the use of anchoring with NegoChat-A.

We then proceeded to evaluate the NegoChat-A's NLU (see Subsection 6.1). To evaluate this component, we employed a human expert that tagged each human sentence from our experiments with its correct set of dialog acts. We then compared the correct set to the set returned by the NLU during the experiments, and calculated the sentence-level accuracy. Some sentences could not have been translated correctly, because they are out-of-domain - their correct meaning is currently not handled by the agent, for example: "What is your work experience?" or "If you will be good you will get better conditions". For each of the two agents, we calculated two accuracy figures: one for only those sentences that could be handled ("In domain") and one for the entire set of sentences ("All").

Table 4 summarizes the results of this analysis. As expected, KBAgent required less interaction from the user, as it only works with full offers, full accepts or full rejects. In contrast, NegoChat-A also adds counter-offers to rejections, accepts partial offers and suggests new issues to discuss, making its language richer. KBAgent typically elicited a smaller range of language, resulting in a smaller average number of utterances in this group (11.3 vs. 28.42 for

Table 4 The NLU sentence-level accuracy (#correct/#all) in KBAgent games versus NegoChat-A games

	Agent	Average #instances	Accuracy (%)
In domain	KB	11.3	195/305 = 64
	NegoChat-A	28.42	305/540 = 56
All	KB	12.48	195/337 = 58
	NegoChat-A	30	305/570 = 54

in-domain utterances, 12.48 vs. 30 overall). The NLU unit was more accurate in the KBAgent games (by either 8 or 4 percent), because of the more limited type of language used by the average participant. These results highlight the centrality of NegoChat-A's success. Not only did this agent perform better in all categories, but it did so despite a less accurate NLU unit, something that inherently should have **hurt** its performance.

To further analyze the impact of the NLU in general, and due to the fact that the KBAgent was not designed initially to work in a chat-based environment, we studied how the KBAgent's performance was impacted by its NLU. We hired workers from Amazon Turk to participate in this experiment: half of them interacted with the KBAgent with the NLU (with its 28 % translation error), while the other half used a "Wizard-of-Oz" approach where an expert manually edited the translations generated by the NLU and corrected them before they were sent to the agent (with very little error if any). Our results show that there were no significant difference between the two (481.47 vs. 468.71, with STD around 60), which allows us to conclude that the KBAgent did not significantly suffer from using the NLU component.

8 Discussion and future directions

As the previous section demonstrates, NegoChat-A is successful in integrating chat within an automated agent. NegoChat-A is effective in achieving better agreements than previous agents, including the previously published NegoChat agent that also uses chat but does not use anchoring [10], as well as the KBAgent that does not build incremental agreements [6]. Despite these successes, we believe it is important to highlight the strengths and potential weaknesses of this agent so that future generations of automated negotiation agents can build upon elements from NegoChat-A. Specifically, we highlight issues regarding the domain considered, the impact of other cultures and how additional research in argumentation and interfaces could be integrated in the future.

First, it is important to note the specific domain we used to build and evaluate NegoChat-A. As previously described, we consider a mixed human-agent negotiation environment with repeated interactions and a time discount factor. Assuming a new domain is considered, such as one without a time discount factor, the strategy of the agent would likely need to be altered. For example, NegoChat-A uses a concept of a search cluster to represent all potential agreements within a given time frame. This group of agreements needs to be recalculated as per the time discount. Furthermore, it may be possible that other approaches are more effective than the search cluster approach we present in determining if a given proposal should be accepted. Work by Baarslag et al. [40] studied a variety of acceptance conditions between automated agents, and we believe that future work should consider if any of these might perform better than the search cluster approach we present. Currently, NegoChat-A uses a pilot study to learn people's aspiration scales for a given domain and culture

(see Sect. 7.1 for details of the pilot used in this study). It is possible that more sophisticated learning methods, particularly based on research in preference elicitation, may be useful instead of or in addition to the approach used in this work. Also, the aspiration scales used by NegoChat-A have similarities to other multi-criteria preference models previously developed [41–43]. It may be that different models will be more effective than the concepts of aspiration scales and search clusters presented in this paper.

It is also important to understand that NegoChat-A was designed to maximize the agent's score within human-agent interactions. As such, our evaluation focused on the agent's score, and not the person's agreements or the system's social welfare. We believe that maximizing these other values can be critical—especially if the system's goal is to support a person's decision-making process or to help both sides reach better agreements. For example, the GENIE and Pocket Negotiator projects [44,45] focused on how to use an agent as decision support during a person's negotiation session. AutoMed and AniMed* are agents that act as a mediator between bilateral negotiations between people [46,47]. An important open question is if or how NegoChat-A's strategy could be changed for decision support or mediation.

Second, we believe additional work is needed to study how NegoChat-A can be applied to additional domains and cultures. In this paper we used participants in Israel who participated in an employer / employee negotiation scenario. An open question which we have begun to study is what extensions, if any, are needed to apply NegoChat-A to other negotiation problems. An initial study with participants from Egypt has led us to the insight that not only are the aspiration scales culturally dependent, but also the frequency with which the agent should present the person its offers. Recall from Sect. 5.2 that NegoChat-A presents offers every 25 seconds. We have found that this value is likely culturally dependent, as Egyptian participants found this to be too frequent and needed more time to consider each offer. As currently this agent only uses chat in English we have hypothesized that this may be due to different peoples' backgrounds in understanding and generating chat in English. We believe that further research might be fruitful in analyzing this aspect as well.

Furthermore, we hope to analyze if other bounded rationality theories, in addition to or instead of Aspiration Adaptation Theory and anchoring, can be used by agents with natural language or more sophisticated interfaces. One possible direction is to explicitly use argumentation theory within the agent. Recall from Sect. 6.3 that NegoChat-A's NLU unit currently includes some elements of argumentation. Currently these arguments are randomly selected as part of the generated Natural Language. We believe that explicitly reasoning about when and how to use arguments may help build stronger agents in the future. For example, it may be useful to integrate previous works on negotiation and argumentation [48] with work on modeling human argumentation [49]. Last, NegoChat-A's main contribution is in supporting chat. While this is definitely a more intuitive interface than the menus previously used, we still hope to study in the future how the agent's strategy must be modified so it can be implemented in more lifelike interfaces such as avatars and virtual humans [16].

9 Conclusions

This paper presents NegoChat-A, a novel negotiation agent that considers a natural language interface and its impact on the agent's strategy. In creating NegoChat-A, we present several novel contributions. First, we describe a new negotiation algorithm based on bounded rationality that uses anchoring to set an initial offer and then facilitates incremental agreements crucial for interacting with people. Second, we present a Natural Language module for interacting with this agent and describe its originality. We describe extensive experiments

highlighting NegoChat-A's ability to reach significantly better agreements in less time than the current state-of-the-art KBAgent. We calculated the accuracy of our natural language understanding unit. We show that it was more difficult to understand humans speaking with NegoChat-A than humans speaking with the KBAgent. We conjecture that it is because NegoChat-A itself uses a more versatile natural language than the KBAgent. The success of NegoChat-A over KBAgent, even when considering the greater difficulty of the task, highlights the challenge NegoChat-A faced, and further emphasizes its success. Last, we provide several directions for future improvements in the hopes that other researchers will further develop additional generations of negotiation agents.

Acknowledgments This work was supported in part by ERC Grant #267523, J-FBI-10-009 and MURI Grant #W911NF-08-1-0144. Sarit Kraus is also affiliated with UMIACS.

References

1. Hoppman, P. T. (1996). *The negotiation process and the resolution of international conflicts*. Columbia: University of South Carolina Press.
2. Byde, A., Yearworth, M., Chen, K.-Y., Bartolini, C., Aut ONA. (2003). A system for automated multiple 1–1 negotiation. In *International Conference on Electronic Commerce*.
3. Jonker, C. M., Robu, V., & Treur, J. (2007). An agent architecture for multi-attribute negotiation using incomplete preference information. *Autonomous Agents and Multi-Agent Systems*, *15*(2), 221–252.
4. Lin, R., & Kraus, S. (2010). Can automated agents proficiently negotiate with humans? *Communications of the ACM*, *53*(1), 78–88.
5. Lin, R., Kraus, S., Wilkenfeld, J., & Barry, J. (2008). Negotiating with bounded rational agents in environments with incomplete information using an automated agent. *Artificial Intelligence*, *172*(6–7), 823–851.
6. Oshrat, Y., Lin, R., Kraus, S. (2009). Facing the challenge of human-agent negotiations via effective general opponent modeling, In *AAMAS*.
7. Peled, N., Gal, Y. K., Kraus, S. (2011). A study of computational and human strategies in revelation games. In *AAMAS*.
8. Peled, N., Gal, Y. K., Kraus, S. (2014). A study of computational and human strategies in revelation games. Autonomous Agents and Multi-Agent Systems 1–25.
9. Reyhan, A., & Pinar, Y. (2012). Learning opponents preferences for effective negotiation: An approach based on concept learning. *Journal of Autonomous Agents and Multi-Agent Systems, 24*(1), 104–140.
10. Rosenfeld, A., Zuckerman, I., Segal-Halevi, E., Drein, O., Kraus, S. (2014). Negochat: a chat-based negotiation agent. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14* (pp. 525–532).
11. Kahneman, D. (1992). Reference points, anchors, norms, and mixed feelings. *Organizational Behavior and Human Decision Processes*, *51*(2), 296–312.
12. Kristensen, H., & Garling, T. (1997). The effects of anchor points and reference points on negotiation process and outcome. *Organizational Behavior and Human Decision Processes*, *71*(1), 85–94.
13. Selten, R. (1998). Aspiration adaptation theory. *Journal of Mathematical Psychology*, *42*, 1910–214.
14. Coen, M.H. (1998). Design principles for intelligent environments. In *AAAI/IAAI*.
15. Cohen, P. R. (1992). The role of natural language in a multimodal interface. In *UIST*.
16. Kenny, P., Hartholt, A., Gratch, J., Swartout, W., Traum, D., Marsella, S., Piepol, D. (2007). Building interactive virtual humans for training environments. In *I/ITSEC*.
17. Shneiderman, B., & Plaisant, C. (2004). *Designing the user interface: Strategies for effective human-computer interaction*. Noida: Pearson Education India.
18. Traum, D., Marsella, S., Gratch, J., Lee, J., Hartholt, A. (2008). Multi-party, multi-issue, multi-strategy negotiation for multi-modal virtual agents. In *Intelligent Virtual Agents* (pp. 117–130).
19. Baarslag, T., Fujita, K., Gerding, E. H., Hindriks, K. V., Ito, T., Jennings, N. R., et al. (2011). Evaluating practical negotiating agents: Results and analysis of the 2011 international competition. *Artificial Intelligence*, *198*(2013), 73–103.
20. Lin, R., Kraus, S., Baarslag, T., Tykhonov, D., Hindriks, K. V., & Jonker, C. M. (2014). Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, *30*(1), 48–70.
21. Zuckerman, I., Rosenfeld, A., Segal-Halevi, E., Drein, O., Kraus, S. (2013). Towards automated negotiation agents that use chat interfaces. In *ANAC*.

22. Bac, M., & Raff, H. (1996). Issue-by-issue negotiations: The role of information and time preference. *Games and Economic Behavior*, *13*(1), 125–134.
23. Busch, L.-A., & Horstmann, I. (1997). A comment on issue-by-issue negotiations. *Games and Economic Behavior*, *19*(1), 144–148.
24. Chen, M. K. (2002). *Agendas in multi-issue bargaining: When to sweat the small stuff*. Cambridge: Tech. rep Harvard Department of Economics.
25. Sofer, I., Sarne, D., Hassidim, A. (2012). Negotiation in exploration-based environment. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.
26. Rosenfeld, A., & Kraus, S. (2012). Modeling agents based on aspiration adaptation theory. *Autonomous Agents and Multi-Agent Systems*, *24*(2), 221–254.
27. Simon, H. A. (1957). *Models of Man*. New York: Wiley.
28. Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory*. Cambridge: MIT Press.
29. Jurafsky, D., & Martin, J. H. (2008). *Speech and language processing* (2nd ed.). Upper Saddle River: Prentice Hall.
30. Tsoumakas, G., Katakis, I., Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of the ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*.
31. Carvalho, V. R., Cohen, W. W. (2006). In *KDD* (pp. 548–553).
32. Hahn, S., Dinarelli, M., Raymond, C., Lefevre, F., Lehnen, P., de Mori, R., et al. (2011). *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(6), 1569–1583.
33. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, *7*, 551–585.
34. Tang, L., Rajan, S., Narayanan, V. K. (2009). Large scale multi-label classification via metalabeler. In *Proceedings of the 18th international conference on World wide web, WWW '09, ACM* (pp. 211–220).
35. Morbini, F., Sagae, K. (2011). Joint identification and segmentation of domain-specific dialogue acts for conversational dialogue systems. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics* (pp. 95–100).
36. Madjarov, G., Kocev, D., Gjorgjevikj, D., & Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, *45*(9), 3084–3104.
37. Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, *46*(3), 817–833.
38. Leuski, A., & Traum, D. (2008). *A statistical approach for text processing in virtual humans*. Tech. rep, University of Southern California, Institute for Creative Technologies.
39. Dagan, I., Roth, D., Sammons, M., & Zanzotto, F. M. (2013). Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, *6*(4), 1–220.
40. Baarslag, T., Hindriks, K., & Jonker, C. M. (2014). Effective acceptance conditions in real-time automated negotiation. *Decision Support Systems*, *60*, 68–77.
41. Jonker, C.M., Robu, V., Treur, J. An agent architecture for multi-attribute negotiation using incomplete preference information. *Autonomous Agents and Multi-Agent Systems Journal* 15.
42. Visser, W., Aydogan, R., Hindriks, K., Jonker, C. M. (2012). A framework for qualitative multi-criteria preferences. In *4th International Conference on Agents and Artificial Intelligence*.
43. Azaria, A., Rabinovich, Z., Kraus, S., Goldman, C. V., Gal, Y. (2012). Strategic advice provision in repeated human-agent interactions. In *AAAI*.
44. Wilkenfeld, J., Kraus, S., Holley, K. M., & Harris, M. A. (1995). Genie: A decision support system for crisis negotiations. *Decision Support Systems*, *14*(4), 369–391.
45. Pommeranz, A., Wiggers, P., Brinkman, W. P., Jonker, C. M. Social acceptance of negotiation support systems: Scenario-based exploration with focus groups and online survey, Cognition, Technology and Work.
46. Lin, R., Gev, Y., & Kraus, S. (2011). Bridging the gap: Face-to-face negotiations with an automated mediator. *IEEE Intelligent Systems*, *26*(6), 40–47.
47. Chalamish, M., & Kraus, S. (2012). Automed: An automated mediator for multi-issue bilateral negotiations. *Autonomous Agents and Multi-Agent Systems*, *24*(3), 536–564.
48. Rahwan, I., Ramchurn, S. D., Jennings, N. R., Mcburney, P., Parsons, S., & Sonenberg, L. (2003). Argumentation-based negotiation. *The Knowledge Engineering Review*, *18*(4), 343–375.
49. Rosenfeld, A., Kraus, S. (2012). Providing arguments in discussions based on the prediction of human argumentative behavior. In *AAAI*.