Noname manuscript No.
(will be inserted by the editor)

Strategic Negotiations for Extensive-Form Games

Dave de Jonge · Dongmo Zhang

the date of receipt and acceptance should be inserted later

Abstract When studying extensive-form games it is commonly assumed that players make their decisions individually. One usually does not allow the possibility for the players to negotiate their respective strategies and formally commit themselves to future moves. As a consequence, many non-zero-sum games have been shown to have equilibrium outcomes that are suboptimal and arguably counter-intuitive.

For this reason we feel there is a need to explore a new line of research in which game-playing agents are allowed to negotiate binding agreements before they make their moves. We analyze what happens under such assumptions and define a new equilibrium solution concept to capture this. We show that this new solution concept indeed yields solutions that are more efficient and, in a sense, closer to what one would expect in the real world.

Furthermore, we demonstrate that our ideas are not only theoretical in nature, but can also be implemented on bounded rational agents, with a number of experiments conducted with a new algorithm that combines techniques from Automated Negotiations, (Algorithmic) Game Theory, and General Game Playing. Our algorithm, which we call Monte Carlo Negotiation Search, is an adaptation of Monte Carlo Tree Search that equips the agent with the ability to negotiate. It is completely domain-independent in the sense that it is not tailored to any specific game. It can be applied to any non-zero-sum game, provided that its rules are described in Game Description Language. We show with several experiments that it strongly outperforms non-negotiating players, and that it closely approximates the theoretically optimal outcomes, as defined by our new solution concept.

Dave de Jonge
Western Sydney University, School of Computing, Engineering and Mathematics
Locked Bag 1797, Penrith, NSW 2751, Australia
E-mail: d.dejonge@westernsydney.edu.au

Dongmo Zhang
Western Sydney University, School of Computing, Engineering and Mathematics
Locked Bag 1797, Penrith, NSW 2751, Australia
E-mail: d.zhang@westernsydney.edu.au

Keywords Automated Negotiations · Non-zero-sum Games · Extensive-form Games · General Game Playing · Monte Carlo Tree Search

1 Introduction

Games have always been an important research topic in Artificial Intelligence because they provide a controlled environment with clear-cut rules. They can be seen as simplified metaphors for real-world problems. In a sense, any multiagent system in which each agent has its own private goals that may be conflicting with the other agents' goals can be seen as a game. However, research on game-playing algorithms has mostly focused on zero-sum-games, such as Chess, Poker, and Go. This is striking, because many problems encountered in our daily lives are better modeled as non-zero-sum games. For example, the concept of a market economy is essentially a non-zero-sum game; each participant in the economy has its own personal goals and by exchanging goods and services with one another, participants mutually benefit.

Although traditional game-playing algorithms such as Minimax and Monte Carlo Tree Search can be adapted easily to non-zero-sum games, they do not take into account the ability of the agent to negotiate its actions with its opponent(s). These algorithms may therefore yield inefficient outcomes. Typical examples of games where individual play is inefficient are the (Iterated) Prisoner's Dilemma [1], the Centipede Game [35], and the Dollar Auction [39]. If in these games the players were allowed to negotiate and make binding agreements then they could achieve much better results, as we will show.

Non-zero-sum games have been studied extensively from a formal point of view, but when studying their equilibrium properties, such as the Nash Equilibrium or the Subgame Perfect Equilibrium, it is commonly assumed that players decide their actions individually. Again, one does not take into account that agents may be able to negotiate and formally commit themselves to future actions. We argue that for this reason such equilibrium concepts sometimes predict outcomes that are different from those outcomes one would expect in real-life situations. After all, in the real world people do communicate and coordinate their actions. For example, when two people find themselves in a situation comparable to the Prisoner's Dilemma, the natural thing to do would be to discuss the situation and agree to cooperate. Even if they have no reason to trust each other they can still cooperate, by signing a formal contract which legally binds them to obey their agreements.

We should remark that extensive-form games and Subgame Perfect Equilibria have been used as tools to study negotiations, for example in [31], but in those cases the negotiations *themselves* were modeled as extensive-form games. This is not what this paper is about. In this paper, we study the use of Automated Negotiations as an additional tool to manipulate the outcome of any extensive-form game that, by itself, does not have anything to do with negotiations.

The field of Automated Negotiations, on the other hand, has largely ignored the high complexity of extensive-form games. Most work on Automated Negotiations assumes the utility of any deal is known almost instantaneously, and the process of evaluating the proposal is almost completely abstracted away. This is in sharp contrast to games such as Chess or Go, which are so complex that in most cases it is unfeasible to determine the exact value of any action taken.

Therefore, in this paper we aim to bridge the gap between Automated Negotiations and the theory of extensive-form games. We aim to answer the following question: *"Given some extensive-form game, we know that without negotiations two perfectly rational players would play the Subgame Perfect Equilibrium, but what would happen if we did allow those players to negotiate their actions and sign binding agreements?"* In order to answer this question we define a new solution concept for extensive-form games, which takes into account the possibility that players negotiate binding agreements about their future moves. We calculate the expected utility outcomes under this new equilibrium concept for three traditional games, and show that they dominate the outcomes prescribed by the traditional Subgame Perfect Equilibrium.

The main idea is that we define a new concept which we call an Extensive-Form Game with Negotiations. Specifically, given an extensive-form game G we define the associated Extensive-Form Game with Negotiations NG, which is an adaptation of G that allows the players to negotiate their future actions before every round of the game. Although these negotiations can be modeled as extensive-form games themselves, it is easier to analyze them if we model them as normal-form games. This then allows us to describe the optimal behavior of a perfectly rational agent not only in terms of its moves, but also in terms of the agreements it makes with its opponent.

Furthermore, we present a new solution concept to predict the outcome of a bilateral negotiation, which is an alternative to the Nash Bargaining Solution [30]. The difference is that our solution works for discrete agreement spaces, whereas Nash's solution assumes convex agreement spaces.

In order to show that our concepts are not merely theoretical in nature, but can also be efficiently approximated in a setting with limited computational resources we present the results of a number of experiments we have performed using a new algorithm that combines techniques from General Game Playing (GGP) with techniques from Automated Negotiations. It is an adaptation of the commonly used Monte Carlo Tree Search (MCTS), and equips the agent with the ability to negotiate. We therefore call our algorithm Monte Carlo Negotiation Search (MCNS). We have tested our algorithm on three traditional games and show that is indeed able to play and negotiate in such a way that it approximates our proposed equilibrium concept. This work is a continuation of the earlier work presented in [21] and [22].

The rest of this paper is organized as follows. In Section 2 we give an overview of existing work on Automated Negotiations and GGP. In Section 3 we state the assumptions we make. In Section 4 we introduce a number of formal definitions related to extensive-form games and illustrate these definitions using the *Centipede Game*, which we use as a running example throughout the paper. In Section 5 we formally define the notion of a Negotiation Domain and introduce our solution concept for negotiations over discrete agreement spaces. Section 6 forms the core of this paper. In this Section we define our new equilibrium solution concept for Extensive Form Games with Negotiations. Then, in Section 7 we briefly describe how we have adapted the MCTS algorithm for games with negotiations and we present the results of our experiments. In Section 8 we discuss a number of assumptions we made in this paper and their motivations, and in Section 9 we discuss future work. Finally, in Appendix A and Appendix B we provide the proofs of a number of theorems presented in the main section of the paper, and in Appendix

C we give a complete formalization of our notion of an Extensive-Form Game with Negotiations

2 Related Work

The earliest work on Automated Negotiations has mainly focused on proving formal properties of idealized scenarios. A seminal paper of this type is by Nash [30] in which it was shown that under certain axioms the rational outcome of a bilateral negotiation is the solution that maximizes the product of the players' utilities. Many papers have been written afterwards that generalize or adapt some of these axioms. A non-linear generalization has been made for example in [9]. A general overview of such theoretical studies is made in [38].

In later work focus shifted more towards heuristic approaches for domains where one cannot expect to find any formal equilibrium results, or where such equilibria cannot be determined in any reasonable amount of time. It is usually not possible to give hard guarantees about the success of such algorithms, but they are more suitable to real-world settings. Important examples are [7] and [8], which propose a strategy that amounts to determining for each time t which utility value should be demanded from the opponent (the *Aspiration Level*). However, they do not take into account that it may be hard to find a contract that yields that aspired utility value. They simply assume that such a contract always exists, and that the negotiator can find it without effort.

In general, these heuristic approaches still often make many simplifying assumptions. They may for example assume there is only a small set of possible agreements, or that the utility functions are linear additive functions which are explicitly given or which can be calculated without much computational cost. Such assumptions were made, for example, in the first four editions of the annual Automated Negotiating Agent Competition (ANAC 2010-2013) [2].

Recently, more attention has been given to more realistic settings in which the number of possible deals is so large that one needs to apply search algorithms to find good deals to propose, and where utility functions are non-linear [14, 28, 29]. Although in these works the utility functions are indeed non-linear over the vector space that represents the set of possible contracts, the value of any given contract can still be calculated quickly by solving a linear equation. Theoretically speaking, one can argue that any non-linear function can indeed be *modeled* in such a way. However, in practice the utility functions are not always *given* in this way (e.g. there is no known closed-form expression for the utility function over the set of all possible configurations of a Chess game). In order to apply their method one would first need to transform the given expression of the utility function into the expression required by their model, which may easily turn out unfeasible.

Therefore, the idea of complex utility functions was taken a step further in [18], which studied domains for which the utility functions are not only non-linear, but also computationally hard to calculate.

An even more complex negotiation scenario is the game of Diplomacy [6]. This is an extensive-form game that involves negotiations before each round. These negotiations are very complex, because the players' utility functions are not directly defined in terms of the agreements they make, but more indirectly through the moves they make in the game. The players negotiate with one another about

which moves each will make, which in turn influences the outcome of the game. Determining the effect of an agreement on the player's final utility is a hard problem that involves Game Theory and Constraint Satisfaction. Pioneering work on negotiations in Diplomacy was presented in [25,36]. New interest in Diplomacy as a test-bed for negotiations has sparked with the introduction of the DipGame platform [6] and its extension BANDANA [20], making the development of Diplomacy agents for scientific research easier. Several negotiating agents have been developed on these platforms [5,10,16]. Also, the 2017 and 2018 editions of ANAC hosted a special Diplomacy League [17]. Unfortunately, the agents developed for Diplomacy highly depend on details specific for this game and are therefore hard to generalize to other settings.

Search algorithms such as Genetic Algorithms (GA)[19,33] and Simulated Annealing [14,28] have been used to search for good proposals in complex negotiation domains. Unfortunately, GAs cannot be applied straightforwardly to games, because the search space is not closed under the two major operators of GA ('mutation' and 'crossover'). When applying these operators to a legal sequence of joint actions, the result will generally contain illegal moves. A similar problem occurs with Simulated Annealing.

The field of General Game Playing (GGP) studies algorithms for game playing agents, under the restriction that the rules of those games are only known at run-time. Therefore, when developing a GGP agent, one cannot use any game-specific heuristics. GGP is a relatively new topic. Although earlier work has been done, it only started to draw widespread attention in the AI community after the introduction of Game Description Language (GDL) [27] and the first edition of the annual AAAI GGP competition in 2005 [13]. GDL is a formal language to write down the rules of a game in the form of a machine-readable logic program. Common techniques applied by GGP players are minimax [42], alpha-beta pruning [23] and Monte Carlo Tree Search (MCTS) [24]. Many of the winners of the AAAI GGP competition applied variants of MCTS, such as FluxPlayer [37], Cadia Player [11], Sancho,[1] and Galvanise.[2] MCTS is also the basic algorithm applied by the AlphaGo program that recently defeated the world champion in Go [40].

One of the main contributions of this paper is the definition of an *Extensive-Form Game with Negotiations*. We have already defined a similar concept earlier in [16] and [20], which we called a *Negotiation Game*, and which was essentially a one-shot game preceded by a negotiation stage. In [21] we extended this definition to also include games over multiple rounds, but where only the first round was preceded by a negotiation stage. In this paper, we extend it even further, by allowing negotiations before *every* round of the game. Note, however, that here we do not use the term 'Negotiation Game' because this name might be confusing in combination with a number of other definitions in this paper.

Finally, we remark that in this paper we define some concepts in a slightly different way than in our earlier paper [22], but they are essentially the same.

[1] `http://sanchoggp.blogspot.co.uk/2014/05/what-is-sancho.html`

[2] `https://bitbucket.org/rxe/galvanise_v2`

3 Assumptions

We consider a setting in which two agents play a non-zero-sum game that takes place over multiple rounds. In each round each player chooses an action from a finite set of legal actions, with the goal of ending the game in a final state that maximizes that player's utility. Unlike in traditional studies of extensive-form games we assume that in each round, before selecting their actions, the players have the opportunity to negotiate binding agreements about which strategies they will follow throughout the rest of the game.

The goal of our work is to implement an algorithm for one of these two agents that aims to maximize that agent's utility by strategically selecting the right moves to make, and by negotiating the right deals with its opponent.

In summary, we are making the following assumptions, which we discuss in more detail in Section 8.

1. **Agents are self-interested.** Each agent is only interested in maximizing its own utility, and is not interested in maximizing any form of 'social utility'. Therefore, a rational player is only willing to accept a proposal if it expects that doing so will not decrease its own utility.
2. **The opponent is unknown.** When implementing our algorithm we cannot make any assumptions about the other agent it will be negotiating with. The other agent may be running the same algorithm, or a completely different algorithm, or may even be human.
3. **Agreements are always obeyed.** It is very important to understand that we assume the agents always obey the agreements they make, even if it would be rational for one or both of the agents to unilaterally deviate from it.
   We can make this assumption, because we assume there is some external mechanism in place that enforces these agreements to be obeyed. This is similar to the real-world case where you sign a legally binding contract, which is enforced by the judicial system.
   We should stress, however, that this enforcement mechanism does not make any decisions. It merely enforces the agreements made by the players, but the players themselves have full autonomy to decide which agreements they make. The fact that agents must obey their agreements is exactly what makes the analysis of our games-with-negotiations different from classical extensive-form games. For example, if the players of an (iterated) Prisoner's Dilemma were allowed to make binding agreements prior to choosing their actions, they could agree to play 'cooperate', and hence reach a better outcome than in the classical Prisoner's Dilemma. This only works, however, if such agreements are truly binding.
   In Section 8.3 we discuss our arguments to support this assumption in more detail.
4. **Bounded rationality.** Throughout most of this paper we focus on the theoretical aspects of our setting, and therefore ignore the problem of bounded rationality. However, in Section 7, we present an anytime algorithm that is able to approach the theoretically optimal solution with limited resources.
5. **General games.** The agents only receive the rules of the game at run-time, so our algorithm cannot make use of any pre-defined game-specific heuristics.

6. **The agents play games of full information.** We only consider games of full information and with no random events. More precisely, this means that agents have a complete description of each others' utility functions. We should note, however, that these descriptions may be very complex, so it will often still be too hard to calculate the utility values exactly. Also, we should note that 'full information' only refers to the rules of the game, but it does *not* mean that the agents know each others' implementations or strategies.

Note that points 1, 2, 4 and 6 are pretty much standard assumptions in (algorithmic) Game Theory, and in addition, point 5 is standard in the field of General Game Playing. Point 3, on the other hand is not common in Game Theory, but is a standard assumption in the field of Automated Negotiations (although not often mentioned explicitly).

4 Extensive Form Games

In this section we formalize the notion of an *extensive-form game* (without negotiations), and several related concepts. These definitions are by no means novel, but we need to state them in order to establish notation, and to take away any ambiguity in these definitions that may exist in the literature.

We start by defining the notion of a *first-order protocol* and then define an extensive form game as a first-order protocol together with a pair of utility functions (in Appendix C we will also define *higher-order protocols*, which are essentially protocols that are built up from smaller protocols). Although our definitions can be extended easily to multiple agents we will for simplicity restrict ourselves to protocols for only two agents.

Throughout this paper, for any quantity q, we use the notation $\vec{q}$ as a shorthand for a pair (q_0, q_1).

4.1 Protocols and Games

Definition 1 A **protocol of order 1** or **first-order protocol** is a tuple $Pr = \langle \vec{\alpha}, \vec{\mathcal{A}}, W, w_0, T, \vec{L}, u, O, out \rangle$ where:

- $\vec{\alpha} = (\alpha_0, \alpha_1)$ a pair of **agents**.
- $\vec{\mathcal{A}} = (\mathcal{A}_0, \mathcal{A}_1)$ in which each $\mathcal{A}_i$ is a set of **actions** or **moves** of agent α_i.
- W is a finite set of **states**.
- $w_o \in W$ is the **initial state**
- $T \subset W$ is the set of **terminal states**.
- $\vec{L} = (L_0, L_1)$, where each L_i is the **legality function** of α_i, which assigns to each non-terminal state a nonempty set of actions $L_i : (W \setminus T) \to 2^{\mathcal{A}_i}$.
- $u : (W \setminus T) \times \mathcal{A}_0 \times \mathcal{A}_1 \to W$ is the **update function** that maps each non-terminal state and action-pair to a new state.
- O is a finite set of **outcomes**.
- $out : T \to O$ is the **outcome function** that labels every terminal state with an outcome.

We say an action a is **legal** in w for agent α_i iff $a \in L_i(w)$. In general, agents have multiple legal moves in each state, because they can choose which move to make, so the co-domain of L_i is the power set of $\mathcal{A}_i$, which we denote by $2^{\mathcal{A}_i}$.

A pair of actions $\vec{a} = (a_0, a_1) \in \mathcal{A}_0 \times \mathcal{A}_1$, one for each player, is called a **joint move**. A joint move (a_0, a_1) is **legal** in w if a_0 is legal for α_0 in w and a_1 is legal for α_1 in w.

The update function u defines how the protocol transitions from one state to the next as a consequence of the actions chosen by the agents.[3]

Definition 2 An **extensive-form game** G is tuple $\langle Pr, \vec{U} \rangle$ consisting of a first-order protocol Pr and a pair of utility functions $\vec{U} = (U_0, U_1)$ that map each outcome of Pr to a real number $U_i : O \to \mathbb{R}$.

In this definition each terminal state is labeled with an object we call the *outcome* and each outcome is in turn mapped to a pair of utility values. For example, in Chess, the set of outcomes could be $O = \{white\_wins, black\_wins, draw\}$, while the utility functions would assign a number of points to each of these outcomes. At this point the notion of an 'outcome' may seem a bit redundant, and it is indeed more common in the literature to map terminal states directly to utility values, but we will see later that it allows us to elegantly model negotiations as a special kind of extensive form game, and it will make it easier to define higher-order protocols.

If t is a terminal state then we will sometimes simply use the notation $U_i(t)$ as a shorthand for $U_i(out(t))$.

Example 1 The Prisoner's Dilemma can be modeled as an extensive-form game[4] as follows:

- $\mathcal{A}_0 = \mathcal{A}_1 = \{cooperate, defect\}$
- $W = \{w_0, t_{cc}, t_{cd}, t_{dc}, t_{dd}\}$
- $T = \{t_{cc}, t_{cd}, t_{dc}, t_{dd}\}$
- $L_1(w_0) = L_2(w_0) = \{cooperate, defect\}$
- $u(w_0, cooperate, cooperate) = t_{cc}, \quad u(w_0, cooperate, defect) = t_{cd},$
  $u(w_0, defect, cooperate) = t_{dc}, \quad u(w_0, defect, defect) = t_{dd}$
- $O = T$
- $out = Id$
- $U_0(t_{cc}) = 3 \quad U_1(t_{cc}) = 3$
  $U_0(t_{cd}) = 0 \quad U_1(t_{cd}) = 5$
  $U_0(t_{dc}) = 5 \quad U_1(t_{dc}) = 0$
  $U_0(t_{dd}) = 1 \quad U_1(t_{dd}) = 1$

Here, w_0 is the initial state, t_{cc} is the terminal state that is reached when both players play *cooperate*, t_{cd} is the terminal state that is reached when α_0 plays *cooperate* and α_1 plays *defect*, etcetera. The outcome space is simply identified with the set of terminal states and the map *out* between T and O is the identity function.

[3] The update function does not always need to be defined completely, because it is only relevant on those triples (w, a_0, a_1) for which a_0 and a_1 are legal in w anyway.

[4] It is easier and more custom to define the Prisoner's Dilemma as a normal-form game, but the point is that we want to give a simple example of an extensive-form game.

Note that a first-order protocol can be seen as a directed graph with W its set of vertices, and for which there is an edge from w to w' iff there are legal actions $a_0 \in L_0(w)$ and $a_1 \in L_1(w)$ such that $u(w, a_0, a_1) = w'$. In this paper we always assume for any protocol or game that this underlying graph is acyclic. This implies that each game has a maximum number of rounds, because the number of states is assumed finite.

Informally, a *turn-taking game* is a game in which in every turn only one player (the *active player* of that turn) makes a move. In our definitions above, however, we have followed the standard convention of GGP that players always make moves simultaneously. This is not a restriction, because any turn-taking game can be modeled equivalently as a game with simultaneous moves. One can achieve this by adding a dummy move to the game and requiring that the non-active player makes that dummy move instead of making no move at all. This dummy move is usually referred to as *noop*. Therefore, we formally define turn-taking games as follows.

Definition 3 A first-order protocol is called a **turn-taking protocol** if one can define a function *active*, that maps each non-terminal state to one of the two agents: $active : W \setminus T \to \{\alpha_0, \alpha_1\}$, such that for every non-terminal state w we have:

$$\text{If } active(w) \neq \alpha_i \text{ then } L_i(w) = \{noop\}$$

If $\alpha_i = active(w)$ we say that α_i is the **active player** of w, while the other player is called the **non-active player** of w. A **turn-taking game** is a game for which its protocol is a turn-taking protocol.

This definition says that in every non-terminal state either α_0 or α_1 is the active player, and that the non-active player always has exactly one legal move, called *noop*. In the rest of this paper we will always assume every game is a turn-taking game, unless specified otherwise. We will use the notation $u(w, a)$ as a shorthand for $u(w, a, noop)$ or $u(w, noop, a)$ where a is an action of the active player of w.

4.2 Client-Server Model

We will now explain the semantics of the above definitions by modeling the agents as clients in a client-server architecture. This architecture is largely the same as the one commonly used in General Game Playing [13]. We note, however, that any other type of architecture could be used just as well.

In order to play a game, the two agents need to connect to a server. The server then sends a message to each player containing the description of some game, written in GDL, which directly encodes the components of Def. 1 and 2. The agents are given some initial time to parse the game description and initialize their algorithms. Next, the game starts when the server sends a message to each player indicating the initial state w_0. In general, whenever the server sends a message with some non-terminal state w_r each player α_i must reply with a message containing a legal move $a_i \in L_i(w_r)$. The server will then send a new message with the new state w_{r+1}, which is determined by the game's update function and the actions chosen by the players: $w_{r+1} = u(w_r, a_0, a_1)$. This process repeats until the state sent by the server is a terminal state $t \in T$. Each player α_i then obtains the utility

value $U_i(t)$ corresponding to that terminal state. If a player does not respond to the server within a specified deadline, or responds with an illegal action, the server will instead pick a random legal action for that player.

Whenever we say that the game is *in round* r or that the game is *in state* w_r, we mean that the last message from the server contained the state w_r.

4.3 The Centipede Game

We now present an existing game called the Centipede Game [35], which we will use as a running example throughout this paper.

The Centipede Game is a typical non-zero-sum game often discussed in Game Theory text books. The details of its definition may vary per text book, so here we follow the definition according to its GDL description implemented by Sam Schreiber, which we downloaded from the GDL database.[5] This is mainly so that the results of our experiments are in line with the definitions here.

The Centipede Game is a turn-taking game for two players, with a maximum of 19 rounds. In each round the active player of that round can choose between two possible moves: *finish* or *continue*, except in the 19th round, in which case the only possible move is *finish*. In any round, if the active player chooses *finish* the game stops. Otherwise, the game continues to the next round. The utilities of the players are purely defined by the round in which the game is stopped, as indicated in Figure 1.

Formally, we can define its set of terminal states as: $T = \{t_1, t_2 \dots t_{19}\}$ and its complete set of states as: $W = \{w_0, w_1 \dots w_{18}\} \cup T$. In each round the active player can play *continue* or *finish*, while the non-active player can only play *noop*:

$$\mathcal{A}_0 = \mathcal{A}_1 = \{noop, continue, finish\}$$

$$L_i(w_k) = \begin{cases} \{noop\} & \text{if } i \neq k \pmod 2 \\ \{continue, finish\} & \text{if } i = k \pmod 2 \text{ and } k \neq 18 \\ \{finish\} & \text{if } i = k \pmod 2 \text{ and } k = 18 \end{cases}$$

After playing *continue* the next state will be another non-terminal state:

$$u(w_k, continue) = w_{k+1}$$

After playing *finish* the next state will be a terminal state:

$$u(w_k, finish) = t_{k+1}$$

Like in our example of the Prisoner's Dilemma, we simply identify the outcome set with the set of terminal states.

$$O = T \quad out = Id$$

The utility values for a terminal state t_k are given as follows:

$$\vec{U}(t_k) = \begin{cases} (5k \ , \ 5k-5) & \text{if } k \text{ is odd} \\ (5k-10 \ , \ 5k+5) & \text{if } k \text{ is even} \end{cases}$$

[5] `http://games.ggp.org`

Specifically, this means that for $k = 17, 18, 19$ we have:

$$\vec{U}(t_{17}) = (85, 80)$$
$$\vec{U}(t_{18}) = (80, 95)$$
$$\vec{U}(t_{19}) = (95, 90)$$

At first sight, one might think that the best strategy is to always play *continue*. After all, the higher the value of k, the higher the scores for both players. However, there is a catch. For any k, the active player of w_k always prefers t_{k+1} over t_{k+2}. In other words: she prefers to finish directly rather than to let the opponent finish the game in the next round. Therefore, if the game is in state w_{17} then player α_1 prefers to finish immediately, yielding terminal state t_{18} and hence 95 utility points, rather than to continue and let the game finish in state t_{19} which only yields 90 utility points. However, suppose the game is in state w_{16}. Active player α_0 will now reason that if the game continues to w_{17} then α_1 will play *finish* and the game will end in t_{18}. This means that α_0 would prefer to finish the game immediately, yielding state t_{17} which is more profitable to α_0 than t_{18}. Continuing this reasoning we come to the counter-intuitive conclusion that if α_0 is perfectly rational she would finish the game immediately, at state t_1.

This is formalized in the following, well-known, lemma from [35], which we will need later on.

Lemma 1 *In the Subgame Perfect Equilibrium of the Centipede Game, for any non-terminal state w_k the active player always plays 'finish'.*

This can be proved easily using the technique of backward induction (see for example [31]). However, since it is a well-known result we omit the proof.

5 Negotiations

In this section we will give a short review of the topic of Automated Negotiations and show how it fits within the framework of our earlier definitions.

5.1 Informal Introduction

In a typical, classical domain for Automated Negotiations two agents α_0 and α_1 are bargaining to agree on some contract. The agents have a fixed amount of time to make proposals to one another according to some negotiation protocol, such as the Alternating Offers protocol [34]. That is, an agent may propose a contract x from some predefined set of possible agreements Agr (known as the *contract space* or *agreement space*) to the other agent and then the other agent may either accept the proposal, or make a counter proposal $y \in Agr$. The agents continue making proposals to one another until either a given deadline t_{dead} has passed, or one of the agents accepts a proposal made by the other.

Each agent α_i has a utility function U_i which assigns to each contract $x \in Agr$ a utility value $U_i(x) \in \mathbb{R}$. If a proposed contract x is accepted, then both agents receive their respective utility values, $U_0(x)$ and $U_1(x)$, corresponding to this contract. However, if no proposal is accepted before the deadline, then each agent

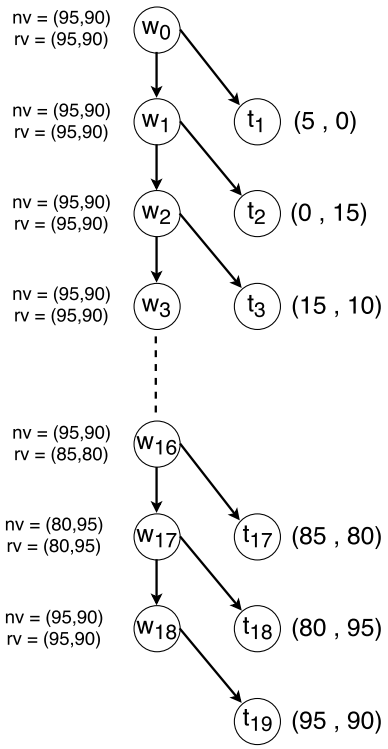

Fig. 1 State space of the Centipede Game. Next to each terminal state t_k, on the right-hand side, we have indicated its utility vector. Next to each non-terminal state w_k, on the left-hand side, we have indicated its Negotiation Values $nv_{w,\vec{r},i}$ and Reservation Values $rv_{w,\vec{r},i}$ (Sec. 6.3), assuming no deal has been made in any earlier round.

α_i will receive a pre-defined amount of utility which is known as its *Reservation Value* rv_i. Clearly, a perfectly rational agent would never accept any proposal that yields an amount of utility less then its Reservation Value.[6]

The vector $\vec{U}(x) = (U_0(x), U_1(x))$ is referred to as the *utility vector* of x and the set of all utility vectors is called the *utility space*. The negotiation protocol, the agreement space, the utility functions, and the reservation values taken together are referred to as a *Negotiation Domain*.

A typical example of a negotiation domain would be the case of a car salesman and a client negotiating over the price of the car. In that case the contract space would be the set of all possible prices the customer could possibly pay. However, more complex negotiation scenarios are possible in which the agreement space is

[6] In the case the negotiator is *bounded* rational we should say that it would never accept any proposal it *expects* to yield utility less than its reservation value.

multi-dimensional, meaning that they do not only negotiate the price, but also other aspects of the deal, such as the type of car, or any additional features.

5.2 Formal Definition a Negotiation Domain

In this section we formalize the notion of a Negotiation Domain as a special kind of extensive-form game for which the underlying protocol is a negotiation protocol. We should caution the reader, however, that later on we will mostly model negotiations as a *normal* form game instead. Also, we should note that in the rest of the paper we are mainly concerned with negotiation domains N for which the agreement space is the set of strategies of some *other* extensive-form game G.

Many different protocols have been defined in the automated negotiations literature, and it is not hard to see that most of them can indeed be modeled as a special case of a first-order protocol in the sense of Definition 1. The set of actions of each negotiating agent would typically be something like $\mathcal{A}_0 = \mathcal{A}_1 = \{propose(x), accept(x), noop \mid x \in Agr\}$, where Agr is the set of possible contracts the negotiators may agree upon, which can be any kind of set, depending on the domain. The 'actions' of a negotiator in a negotiation domain consist of accepting or proposing contracts. Furthermore, the outcome of a negotiation protocol is either a contract $x \in Agr$ which the agents agree upon, or no agreement at all, which we call the *conflict outcome* and which we denote by η.

We will not attempt to give a very precise definition of a negotiation protocol, but instead just stick with the following, rather broad, definition, which is sufficient for our purposes.

Definition 4 A first-order protocol is called a **negotiation protocol**, if its outcome set O can be decomposed as $O = Agr \cup \{\eta\}$ for some set Agr that we call the **agreement space**, and some element η that we refer to as the **conflict outcome**, and such that for every $x \in Agr$ and every $i \in \{0, 1\}$ we have $propose(x) \in \mathcal{A}_i$ and $accept(x) \in \mathcal{A}_i$.

This definition just says there must be some *propose* action as well as an *accept* action for every possible contract. It does not say when it is or when it is not legal to make such proposals or acceptances, because this would be different for every negotiation protocol.

Definition 5 A **negotiation domain** is a negotiation protocol together with a pair of utility functions $\vec{U}$ that map each outcome in $Agr \cup \{\eta\}$ to a real number. In other words, it is an extensive form game for which the underlying protocol is a negotiation protocol.

Definition 6 For a negotiation domain N, we define the **Reservation Values** $rv_i(N)$ as $rv_i(N) := U_i(\eta)$, where η is the conflict outcome.

We will just use the notation rv_i instead of $rv_i(N)$ whenever N is clear from context.

Definition 7 For any negotiation domain N an agreement $x \in Agr$ is said to be **rational** if for all $i \in \{0, 1\}$ we have $U_i(x) \geq rv_i$ and for at least one $i \in \{0, 1\}$ we have $U_i(x) > rv_i$.

5.3 Negotiations as a Normal Form Game

In this section we will present our first main result. Namely, an alternative to the Nash Bargaining Solution for discrete agreement spaces. This solution will play an important role in our analysis of extensive-from games with negotiations later on in the paper. The definitions in this section have already be published before, in [22], but we present them here again, to keep this paper self-contained.

Analyzing a negotiation domain can be difficult because its negotiation protocol may consist of many states. However, we can simplify the analysis by ignoring the details of the concession strategies applied by the negotiators, and instead looking only at the minimum amount of utility the negotiators are willing to accept at the end of the negotiations.

In a typical time-based concession strategy a negotiator would slowly decrease its demanded utility over time, until a minimally acceptable value is reached near the deadline. The question is then, is how low this minimally acceptable value should be. One could simply choose it to be equal to the Reservation Value rv_i, but that would be a weak strategy, because any opponent could easily exploit it by not accepting anything until the deadline. On the other hand, choosing this value to be very high is also risky, because if the opponent does the same, it is likely that the agents will never concede enough to come to any agreement at all. A strong negotiator therefore needs to strike a balance. One should determine which utility value one can realistically expect to obtain and concede no further than that.

If the utility space is convex, then the solution to this problem is already given by Nash [30]. According to the Nash Bargaining Solution both negotiators should concede towards the contract x that maximizes the product $U_0(x) \cdot U_1(x)$. However, in this paper we deal with discrete, and therefore non-convex, agreement spaces. This could be solved by allowing the agents to also negotiate 'lotteries' of contracts. That is: the agents could agree to flip a (biased) coin, and if the outcome is 'heads' they will obey contract x, while if the outcome is 'tails' they will obey contract y. If we allow the coin to be biased with any probability P, then the utility space becomes convex. Unfortunately, the problem with this solution is that one needs to trust that the coin is indeed exactly biased with the agreed probability P. One would need some external source of randomness that is trusted by both agents. Such a source is not always available.

Therefore, we propose another solution, in which the agents only make deterministic agreements. Each player itself will still flip a coin in order to choose whether it will insist on some agreement x with high utility, or concede to an alternative agreement y with lower utility, but the important difference is that the agent only uses a coin internally, so there is no problem with trust.

Let us explain this with an example. Suppose that the agents have to choose between two Pareto-optimal contracts x and y with corresponding utility vectors $\vec{U}(x) = (60, 40)$ and $\vec{U}(y) = (40, 60)$. We see that player α_0 has the choice to either *persist* and demand at least 60 utility points (meaning that he is only willing to accept contract x), or to *yield* and accept to receive only 40 utility points (meaning he is willing to accept either contract x or contract y). The opponent α_1 has exactly the same two options, but with the roles of x and y reversed.

The choice which strategy to follow, to persist or to yield, may itself also be seen as a game, which we refer to as the *concession game*. If both players persist, the negotiations fail, and the players obtain their respective reservation values. If one

player persists and the other yields, then the persisting player receives 60 points, while the other receives 40 points. If both players yield, the outcome depends on how fast the players concede. The player who concedes fastest will receive 40 points, while the other will receive 60 points. Here we will simply assume that in that case there is a 50% chance that they will agree on contract x, and 50% chance it will be y, so the expected utility for both players is 50.

In general, if we have contracts x and y with utility vectors $\vec{U}(x) = (A, B)$ and $\vec{U}(y) = (C, D)$ and the Reservation Values for the players are rv_0 and rv_1, then the concession game can be modeled as a normal-form game with the following payoff matrix:

	$Persist$	$Yield$
$Persist$	rv_0, rv_1	A, B
$Yield$	C, D	$\frac{1}{2}(A+C), \frac{1}{2}(B+D)$

One can easily calculate (see e.g. [32]) that the Mixed Strategy Nash Equilibrium of this game is given by:

$$P_0 = \frac{D - B}{B + D - 2 \cdot rv_1} \qquad P_1 = \frac{A - C}{A + C - 2 \cdot rv_0} \tag{1}$$

where P_i represents the probability that player α_i will play *persist*. Agent α_0 can now determine its minimum acceptable utility by flipping a coin and setting it equal to A with probability P_0 and to C with probability $1 - P_0$.

Furthermore, we can now also calculate the expectation values of the utility both agents will receive from the negotiations. If we apply this to our example, then we find that $P_0 = P_1 = \frac{1}{5}$, and for their expected utilities we find:

$$E(U_0) = \frac{1}{5} \cdot \frac{1}{5} \cdot 0 + \frac{1}{5} \cdot \frac{4}{5} \cdot 60 + \frac{4}{5} \cdot \frac{1}{5} \cdot 40 + \frac{4}{5} \cdot \frac{4}{5} \cdot 50 = 48$$
$$E(U_1) = \frac{1}{5} \cdot \frac{1}{5} \cdot 0 + \frac{1}{5} \cdot \frac{4}{5} \cdot 40 + \frac{4}{5} \cdot \frac{1}{5} \cdot 60 + \frac{4}{5} \cdot \frac{4}{5} \cdot 50 = 48$$

We see that both players can expect to receive 48 utility points. Interestingly, this is lower than the expected outcome in the traditional approach where the players negotiate lotteries. In that case, following Nash, both players would expect to receive 50 utility points. The reason for this difference is that the Nash solution calculates the expected outcome under the assumption that the negotiations will succeed. However, in our case, the non-convexity means that rational players must sometimes persist, which will result in failure if they both do that at the same time. Of course, this means that if the players do have access to a trusted coin, then it is preferable for them to negotiate lotteries.

In the following, we say that α_i prefers x over y if $U_i(x) > U_i(y)$. If x and y are Pareto-optimal then if α_0 prefers x over y this implies that α_1 prefers y over x.

Definition 8 Let N be a negotiation domain. Then the **Concession Game** $C(N)$ is a normal form game in which for each player its set of actions is exactly the set of Pareto-optimal and rational contracts of Agr. Furthermore, for any pair of actions x_0, x_1, for which α_0 prefers x_0 over x_1, the payoff vector $\vec{\mathcal{U}}$ is defined as:

$$\mathcal{U}_i(x_0, x_1) = rv_i$$
$$\mathcal{U}_i(x_0, x_0) = U_i(x_0)$$
$$\mathcal{U}_i(x_1, x_1) = U_i(x_1)$$
$$\mathcal{U}_i(x_1, x_0) = \frac{1}{2} \cdot (U_i(x_0) + U_i(x_1))$$

Here, the U_i represent the utility functions of N, while the $\mathcal{U}_i$ represent the utility functions of the concession game $C(N)$. Note that when a player chooses a contract x as the action to play, it means that he is willing to accept any contract that is at least as good as x.

The first line in this definition represents the case that both agents persist, so each is only willing to accept its own preferred contract. Therefore, they cannot come to an agreement, and they receive their respective reservation values. In the second and third lines the negotiators agree on the contracts x_0 and x_1 respectively. The last line represents the case that each negotiator is willing to concede to its less preferred contract, so they may agree on either of the two.

For a negotiation domain N and a negotiator α_i we define the *negotiation value* $nv_i(N)$ as the expected utility of α_i in the concession game $C(N)$. In order to calculate this expected utility we need to make some assumptions about the strategies played by the two agents. For example, we could assume the agents play a Nash Equilibrium. However, in general, $C(N)$ can have multiple Nash Equilibria (see Appendix B). For the rest of this paper we will ignore the question how the agents choose their concession strategy, and simply assume there is *some* solution concept that the agents adhere to. For the proofs of the various theorems in this paper it does not matter, because in all cases the negotiation domains will have only 0 or 1 rational agreements.

In the case that N does not have any rational agreements $C(N)$ is not defined, and we define the negotiation values to be equal to the reservation values of N. If there is only 1 rational agreement x, then obviously, the expected utility for α_i is exactly $U_i(x)$.

Definition 9 For a negotiation domain N the **Negotiation Values** $nv_i(N)$ are defined as:

$$nv_i(N) = \begin{cases} rv_i(N) & \text{if } N \text{ has no rational agreements} \\ U_i(x) & \text{if } N \text{ has exactly 1 rational agreement } x \\ E(\mathcal{U}_i) & \text{otherwise} \end{cases} \tag{2}$$

where $E(\mathcal{U}_i)$ is the expectation value of $\mathcal{U}_i$ under some game theoretical solution concept applied to $C(N)$.

We propose the Negotiation Value as an alternative to the Nash Bargaining solution for negotiation domains with discrete agreement spaces.

Lemma 2 *For any negotiation domain N we have: $nv_i(N) \geq rv_i(N)$.*

Proof Note that, by definition, the game $C(N)$ only involves agreements that are rational, which means that for each agreement x of N, we have $U_i(x) \geq rv_i(N)$. Therefore, from Def. 8 it follows that for every pair of pure strategies (x, y) we have $\mathcal{U}_i(x, y) \geq rv_i(N)$. This in turn, means that the same holds for any pair of mixed strategies. □

Of course, this result is not surprising, because it simply says that each negotiator can expect to achieve a utility that is at least as high as its reservation value, which should be obvious from the informal definitions. However, it will be useful to us later on to have this lemma.

6 Extensive-Form Games with Negotiations

In this section we define the main domain of interest of this paper, which we call an *Extensive-Form Game with Negotiations*. This consists of two agents that are playing some extensive-form game G, but before each round they are able to negotiate binding agreements about the strategies they will follow during the rest of the game.

Note that in Section 5.2 a negotiation domain itself was modeled as an extensive-form game. Here, however, we take some existing extensive-form game G without negotiations, and turn it into a game-with-negotiations NG by allowing the agents to negotiate.

In Section 6.1 we first go through a number of necessary definitions, and then in Section 6.2 we can finally present the definition of this new concept.

6.1 Strategies

Definition 10 Given a game G, a **strategy** σ_i for player α_i is a map that maps every non-terminal state of G to a nonempty set of legal moves for that player. Thus, σ_i is a map:

$$\sigma_i : W \setminus T \to 2^{\mathcal{A}_i} \quad \text{such that} \quad \forall w \in W \setminus T : \quad \emptyset \neq \sigma_i(w) \subseteq L_i(w).$$

We say a player α_i **follows** a strategy σ_i if it always chooses its actions from $\sigma_i(w)$, for any non-terminal state w. The **trivial strategy** τ_i for player α_i is the strategy given by $\tau_i(w) = L_i(w)$ for all $w \in W \setminus T$.

A strategy σ is essentially a commitment of an agent to choose its actions only from a certain *subset* $\sigma(w) \subseteq L_i(w)$ of all its legal actions. In that sense, the trivial strategy represents the case that the agent is not committed to anything at all, because it allows the agent in any state to choose any legal action $a \in \tau(w) = L_i(w)$.

Note that what we here call a strategy, is actually an *indeterministic* strategy, because it only partially fixes the agent's choice for each state. In the special case that for each $w \in W \setminus T$ we have $|\sigma(w)| = 1$ we could call it a *deterministic strategy* which is perhaps more close to what one intuitively would call a strategy.

Definition 11 A pair $\vec{\sigma} = (\sigma_0, \sigma_1)$ consisting of one strategy for each player is called a **joint strategy**. We use $\vec{\tau}$ to denote the **trivial joint strategy**, which is the pair (τ_0, τ_1). The set of all joint strategies of an extensive form game G is denoted $\mathcal{S}^G$.

If w is a non-terminal state then we define $next(w, \vec{\sigma})$ as the set of all states that could be the next state if the players follow the joint strategy $\vec{\sigma}$:

$$next(w, \vec{\sigma}) = \{w' \in W \mid \exists a \in \sigma_j(w) : w' = u(w, a)\} \tag{3}$$

with α_j being the active player of w.

Definition 12 For any state w, joint strategy $\vec{\sigma}$ and any player α_i in a turn-taking game we recursively define the **Subgame Perfect Equilibrium Value** $spe_{w,\vec{\sigma},i}$ as follows:

$$spe_{w,\vec{\sigma},i} = \begin{cases} spe_{w^{**},\vec{\sigma},i} & \text{if } w \in W \setminus T \\ U_i(w) & \text{if } w \in T \end{cases}$$

where

$$w^{**} = \underset{w' \in next(w,\vec{\sigma})}{\arg\max} \; spe_{w',\vec{\sigma},j} \tag{4}$$

and where α_j is the active player of w.

This is the utility a perfectly rational player can expect to obtain against another perfectly rational player when the game is in state w, and both players are restricted to follow the joint strategy $\vec{\sigma}$.

Note that this definition is slightly different from the tradition definition of SPE (e.g. [31]) because we take into account that the agents may be committed to some joint strategy $\vec{\sigma}$. The traditional SPE value of any state w would correspond to the special case $spe_{w,\vec{\tau},i}$. If the game is a zero-sum game then the value $spe_{w,\vec{\tau},i}$ is also known as the MiniMax value. We will use the notation SPE_i as a shorthand for the special case $spe_{w_0,\vec{\tau},i}$ (i.e. the traditional SPE value for the game as a whole).

6.2 Games with Negotiations

We will now explain how we combine extensive-form games with negotiations over the strategies of such games. The idea is that we define a new kind of structure, which we call an *Extensive-form Game with Negotiations*, which is neither a game, nor a negotiation domain, but rather a hybrid of the two. Specifically, for any extensive-form game G we can define a corresponding Extensive-Form Game with Negotiations NG. However, since the formal definition of NG is rather complex, we only provide a complete formal definition in Appendix C. In this Section we only provide formal definitions of the most essential components, which should be sufficient to understand the rest of this paper.

In terms of the client-sever model the idea is that after the server has sent the current state w_r to the players and before the players reply with their next actions, we allow the two players to exchange messages between each other according to some negotiation protocol. These negotiation messages are of the type $propose(\vec{\sigma})$ or $accept(\vec{\sigma})$ in which $\vec{\sigma}$ can be any joint strategy. If one player proposes a joint strategy $\vec{\sigma} = (\sigma_0, \sigma_1)$ and the other accepts it, then both players must obey it. This means that for each w_k with $k \geq r$ each player α_i must play a move $a \in \sigma_i(w_k)$. However, even if the players have already agreed to play some joint strategy $\vec{\sigma}$, the players may continue to negotiate and agree on a new joint strategy $\vec{\sigma}'$. In that case the first agreement is discarded, and instead the players are only required to obey the newly agreed $\vec{\sigma}'$. The details of the negotiation protocol are irrelevant for this paper. It could be the Alternating Offers protocol, but we may just as well assume any other bilateral negotiation protocol.

More formally, for any state w of G and any joint strategy $\vec{\sigma}$ we define a Negotiation Domain $N_{w,\vec{\sigma}}$ for which the agreement space is the set of all joint strategies of G. This negotiation domain represents the situation that the game has

reached state w, while the agents have previously agreed to obey the joint strategy $\vec{\sigma}$, and they are currently negotiating whether they can agree on some better joint strategy $\vec{\sigma}'$. Since $\vec{\sigma}$ is the standing agreement, it means that if negotiations in $N_{w,\vec{\sigma}}$ fail, the players remain committed to $\vec{\sigma}$. In other words, the conflict outcome of $N_{w,\vec{\sigma}}$ is $\vec{\sigma}$.

Furthermore, note that if no agreement has been made yet, this is equivalent to saying that the players are "committed" to the trivial joint strategy $\vec{\tau}$ (which is just another way of saying they are not committed to anything at all). Therefore, at the start, when the game is still in the state w_0 the players are negotiating over the negotiation domain $N_{w_0,\vec{\tau}}$.

Informally, for any given extensive-form game G the corresponding Extensive-Form Game with Negotiations NG consists of G together with a negotiation domain $N_{w,\vec{\sigma}}$ for every state w of G and every joint strategy $\vec{\sigma}$ of G.

6.3 Reservation Values and Negotiation Values

As explained above, in each round of the game, before choosing their respective moves, the players negotiate their strategies, which is formalized as a negotiation domain $N_{w,\vec{\sigma}}$. In this section we discuss how to define the utility functions $U_{w,\vec{\sigma},i}$ of these negotiation domains. That is, we need to assign a value $U_{w,\vec{\sigma},i}(\vec{\sigma}')$ to every possible joint strategy $\vec{\sigma}'$.

Suppose the agents agree on some joint strategy $\vec{\sigma}'$, and let us for a moment assume that they will not make any new agreements afterwards. In that case the agents will be obliged to follow $\vec{\sigma}'$ throughout the rest of the game and, if the agents are perfectly rational, this means that the agents will at the end receive a utility value equal to $spe_{w,\vec{\sigma}',i}$. However, we cannot simply set $U_{w,\vec{\sigma},i}(\vec{\sigma}')$ equal to $spe_{w,\vec{\sigma}',i}$, because the agents do have the option to continue negotiating, which means they could come to a better agreement in a later round. Therefore, $spe_{w,\vec{\sigma}',i}$ is only a lower bound for the true utility $U_{w,\vec{\sigma},i}(\vec{\sigma}')$.

Now, suppose that after agreeing on $\vec{\sigma}'$ the best possible action for the active player leads the game to transition to a non-terminal state w'. Then, in the next round, the agents will be negotiating over the negotiation domain $N_{w',\vec{\sigma}'}$. This means that the value of the agreement $\vec{\sigma}'$ in $N_{w,\vec{\sigma}}$ should be equal to the expected utility for the negotiations in $N_{w',\vec{\sigma}'}$, which, as explained in Section 5.3, is $nv_i(N_{w',\vec{\sigma}'})$. On the other hand, if the best possible action for the active player leads to a terminal state $t \in T$ then the value of $\vec{\sigma}'$ for α_i should be equal to the value $U_i(t)$ of that terminal state. Here, when we say 'the best possible action' we mean the action that leads to the state w^* with the highest expected value for the active player. This means we define the utility functions for $N_{w,\vec{\sigma}}$ as follows:

$$U_{w,\vec{\sigma},i}(\vec{\sigma}') := nv_{w^*,\vec{\sigma}',i} \tag{5}$$

with

$$nv_{w,\vec{\sigma},i} := \begin{cases} nv_i(N_{w,\vec{\sigma}}) & \text{if } w \in W \setminus T \\ U_i(w) & \text{if } w \in T \end{cases} \tag{6}$$

and with

$$w^* := \underset{w' \in next(w,\vec{\sigma}')}{\arg\max}\ nv_{w',\vec{\sigma}',j} \tag{7}$$

where j is the index of the active player. Note that U_i is a utility function of the game G, while $U_{w,\vec{\sigma},i}$ is a utility function of the negotiation domain $N_{w,\vec{\sigma}}$.

We see that in order to calculate the utility values $U_{w,\vec{\sigma},i}(\vec{\sigma}')$ we need to calculate the values $nv_{w',\vec{\sigma}',j}$ for all w' in $next(w,\vec{\sigma}')$, but in order to calculate these values, we need to calculate the negotiation values $nv_i(N_{w',\vec{\sigma}'})$ for each such state, which in turn requires (see Sec. 5.3, Def. 9) the utility functions $U_{w',\vec{\sigma}',i}$ of $N_{w',\vec{\sigma}'}$. In other words, the utility functions are recursively defined and this recursion ends with the terminal states of G (recall that the underlying graph of G was assumed to be finite and acyclic, so the recursion indeed terminates).

In practice, it will often be infeasible to calculate these values exactly, because the number of states of G may be too large for exhaustive exploration. In Section 7, however, we show it is possible to implement an anytime algorithm that can approximate these values without exhaustive exploration.

Now that we have defined the utility functions of $N_{w,\vec{\sigma}}$, we may ask what the reservation values of $N_{w,\vec{\sigma}}$ are, which we will denote by $rv_{w,\vec{\sigma},i}$. Since the reservation values are, by definition, the players' utility values for the conflict outcome, we have: $rv_{w,\vec{\sigma},i} = U_{w,\vec{\sigma},i}(\eta)$. Furthermore, if the negotiations in $N_{w,\vec{\sigma}}$ fail, then $\vec{\sigma}$ remains the standing agreement, which means it is equivalent to agreeing on joint strategy $\vec{\sigma}$. Therefore, we have that $U_{w,\vec{\sigma},i}(\eta) = U_{w,\vec{\sigma},i}(\vec{\sigma})$. Combining this we have:

$$rv_{w,\vec{\sigma},i} = U_{w,\vec{\sigma},i}(\vec{\sigma}) \tag{8}$$

and if we then combine this with Equation (5) we obtain:

$$rv_{w,\vec{\sigma},i} = nv_{w^*,\vec{\sigma},i} \tag{9}$$

For the following, recall that for any state w we have defined a state w^* (Eq. 7) and a state w^{**} (Eq. 4). Informally, w^* is the state that the active player would choose assuming it has the possibility to negotiate further agreements, while w^{**} is the state that the active player would choose if it does not have the option to make new agreements.

Theorem 1 *Let G be an extensive form game. Then for any state $w \in W$, any joint strategy $\vec{\sigma}$, and any player α_i we have:*

$$rv_{w,\vec{\sigma},i} \geq spe_{w,\vec{\sigma},i}$$

Proof Let k_w denote the length of the longest path from w to any terminal state of G (recall from Sec. 4.1 that G can be interpreted as a graph). The proof then goes by induction on k_w. Note that if $k_w = 1$ it means that for all w' in $next(w,\vec{\sigma})$ we have $w' \in T$. This in turn means that $nv_{w',\vec{\sigma},j} = U_j(w')$, but also $spe_{w',\vec{\sigma},j} = U_j(w')$, and thus that $nv_{w',\vec{\sigma},j} = spe_{w',\vec{\sigma},j}$. It follows that $w^* = w^{**} \in T$. Therefore, for $k_w = 1$ we have:

$$\begin{aligned} rv_{w,\vec{\sigma},i} &= nv_{w^*,\vec{\sigma},i} = U_i(w^*) \\ spe_{w,\vec{\sigma},i} &= spe_{w^{**},\vec{\sigma},i} = U_i(w^{**}) \end{aligned}$$

where the first line follows from Eq. (6), Eq. (9), and the fact that $w^* \in T$, while the second line follows from Def. 12, the definition of w^{**} and the fact that $w^{**} \in T$. So indeed, since $w^* = w^{**}$ the theorem holds for the base case.

Now suppose that for some integer k we know that the inequality holds for all states w' with $k_{w'} < k$. Then, for any state w with $k_w = k$:

$$\begin{aligned} rv_{w,\vec{\sigma},i} &= nv_{w^*,\vec{\sigma},i} \\ &\geq nv_{w^{**},\vec{\sigma},i} \\ &\geq rv_{w^{**},\vec{\sigma},i} \\ &\geq spe_{w^{**},\vec{\sigma},i} \\ &= spe_{w,\vec{\sigma},i} \end{aligned}$$

The first line is simply Eq. (9), the second line follows from the definition of w^* and the fact that $w^{**} \in next(w, \vec{\sigma})$, the third line is a special case of Lemma 2, the fourth line is the induction hypothesis (note that w^{**} is a child of w and therefore $k_{w^{**}} < k_w$), and the last line holds by Definition 12. □

Intuitively, this result should be clear. If the players previously agreed on some joint strategy $\vec{\sigma}$ and the game is currently in state w, then $spe_{w,\vec{\sigma},i}$ is the minimum value each player can guarantee himself for the rest of the game by playing the equilibrium strategy, *without further negotiations*. Indeed, this means that an agreement would only be rational if it yields higher utility for both players, and therefore the reservation values must be at least as high as the equilibrium values.

In some cases it may happen that $rv_{w,\vec{\sigma},i}$ is strictly larger than $spe_{w,\vec{\sigma},i}$. The reason for this is that even if negotiations fail in the current round, the players may still come to an agreement in the following rounds, so they may still achieve higher utilities than $spe_{w,\vec{\sigma},i}$. Later on we will see some examples of this.

We will now define a new solution concept which is an adaptation of the Subgame Perfect Equilibrium, for games with negotiations. It represents the utility values the agents can expect to achieve in some game G if they are allowed to negotiate binding agreements regarding to their strategies.

Definition 13 Given an extensive-form turn-taking game G and a player α_i we define its **Negotiation Value** as

$$NV_i(G) := nv_{w_0,\vec{\tau},i}$$

Note that in Def. 9 we defined the Negotiation Values of a negotiation domain N, while here we have defined the Negotiation Values of an extensive-form game G. The two concepts are related, because the Negotiation Values of G are defined as the Negotiation Values $nv_{w_0,\vec{\tau},i}$ of the negotiation domain $N_{w_0,\vec{\tau}}$ corresponding to the negotiations at the start of the game.

Intuitively, it should be clear that allowing players to negotiate binding agreements allows them to achieve higher utilities than they would obtain without negotiations. We formalize this with the following theorem.

Theorem 2 *For any extensive-form game G we have $NV_i(G) \geq SPE_i(G)$*

Proof We know by Lemma 2 that $nv_{w_0,\vec{\tau},i} \geq rv_{w_0,\vec{\tau},i}$ and we know from Theorem 1 that $rv_{w_0,\vec{\tau},i} \geq spe_{w_0,\vec{\tau},i}$. □

For some games, the inequality in this theorem is a strict inequality. For example, as we will show in the next section, for the Centipede Game we have:

$$(NV_0(G), NV_1(G)) = (95, 90)$$
$$(SPE_0(G), SPE_1(G)) = (5, 0)$$

and in Appendix A we show that for the Iterated Prisoner's Dilemma [1] we have:[7]

$$(NV_0(G), NV_1(G)) = (60, 60)$$
$$(SPE_0(G), SPE_1(G)) = (20, 20)$$

6.4 The Centipede Game with Negotiations

In this section we will illustrate the definitions of the previous subsections with the Centipede Game. We will calculate the players' Negotiation Values and Reservation Values of the negotiation domains $N_{w_k,\vec{\tau}}$ for all non-terminal states w_k, from which we then conclude that $(NV_0, NV_1) = (95, 90)$. While reading this Section the reader may find it helpful to simultaneously keep an eye of Figure 1.

In the following, we sometimes say the agents agree on a "deal t" or an "agreement t", for some terminal state t of the Centipede Game. In that case we technically speaking mean that they agree to play a joint strategy $\vec{\sigma}_t$ such that, if both agents follow it, the game will end in state t.

First, let us note that:

$$\forall k \in [0, 17] : next(w_k, \vec{\tau}) = \{w_{k+1}, t_{k+1}\} \quad \text{and} \quad next(w_{18}, \vec{\tau}) = \{t_{19}\}$$

Recall from Eq. (6) that for terminal states t_k we have:

$$nv_{t_k,\vec{\tau},i} = U_i(t_k) \tag{10}$$

Also note that in this game, in any state, the only Pareto-optimal deals are t_{18} and t_{19}. We will proceed by calculating $nv_{w_k,\vec{\tau},i}$, for every non-terminal state w_k, starting with $k = 18$ and then working our way back to $k = 0$.

Let us look at w_{18}. Since there is only one next state for w_{18}, we have $w^* = t_{19}$, and thus using Eq. (9) and (10) we get $rv_{w_{18},\vec{\tau},i} = U_i(t_{19})$. Since any joint strategy played from w_{18} will lead to t_{19}, there is no $\vec{\sigma}$ that could dominate $\vec{\tau}$ so there is no rational contract in $N_{w_{18},\vec{\tau}}$. Therefore, by Def. 9 we have:

$$\forall i \in \{0, 1\} : \quad nv_{w_{18},\vec{\tau},i} = rv_{w_{18},\vec{\tau},i} = U_i(t_{19})$$

Now, let us look at w_{17}. The active player is α_1, and we have two possible next states: w_{18} and t_{18}. We already know that $nv_{w_{18},\vec{\tau},1} = U_1(t_{19}) = 90$ and $nv_{t_{18},\vec{\tau},1} = U_1(t_{18}) = 95$. Therefore, $w^* = t_{18}$. Then, from Eq. (9) and (10) we have $rv_{w_{17},\vec{\tau},i} = nv_{t_{18},\vec{\tau},i} = U_i(t_{18})$. This means that in state w_{17} the deal t_{18} is not rational, but also t_{19} is not rational, because $U_1(t_{19}) = 90 < rv_{w_{17},\vec{\tau},1} = U_1(t_{18}) = 95$. There are no rational deals, so we have:

$$\forall i \in \{0, 1\} : \quad nv_{w_{17},\vec{\tau},i} = rv_{w_{17},\vec{\tau},i} = U_i(t_{18})$$

[7] Strictly speaking SPE_i and NV_i are not defined for the Iterated Prisoner's Dilemma, because it is not a turn-taking game, but we show in Appendix A how this is can be resolved with a minor adaptation.

Now, let us look at state w_{16}. The active player is α_0. For the two next states we have: $nv_{w_{17},0} = U_0(t_{18}) = 80$ and $nv_{t_{17},\vec{\tau},0} = U_0(t_{17}) = 85$, so $w^* = t_{17}$, so:

$$\forall i \in \{0,1\}: \quad rv_{w_{16},\vec{\tau},i} = nv_{t_{17},\vec{\tau},i} = U_i(t_{17})$$

The deal t_{18} is not rational, because $U_0(t_{18}) = 80 < 85 = U_0(t_{17}) = rv_{w_{16},\vec{\tau},0}$, the deal t_{19} however, is rational. Therefore, the only rational agreement of $N_{w_{16},\vec{\tau}}$ is t_{19} and hence, by Def. 9, we have:

$$\forall i \in \{0,1\}: \quad nv_{w_{16},\vec{\tau},i} = U_i(t_{19})$$

Now, let us look at state w_{15}. For the two next states we have: $nv_{w_{16},\vec{\tau},1} = U_1(t_{19}) = 90$ and $nv_{t_{16},\vec{\tau},1} = U_1(t_{16}) = 85$, so $w^* = w_{16}$, and therefore $rv_{w_{15},\vec{\tau},i} = nv_{w_{16},\vec{\tau},i} = U_i(t_{19})$. This means that neither t_{18} nor t_{19} are rational deals, and therefore:

$$\forall i \in \{0,1\}: \quad nv_{w_{15},\vec{\tau},i} = rv_{w_{15},\vec{\tau},i} = U_i(t_{19})$$

For states with $k < 15$ the analysis goes analogously, so:

$$\forall k < 15 \quad \forall i \in \{0,1\}: \quad nv_{w_k,\vec{\tau},i} = rv_{w_k,\vec{\tau},i} = U_i(t_{19})$$

The values of NV_i now follow by setting $k = 0$ in this equation and observing that $\vec{U}(t_{19}) = (95, 90)$

Theorem 3 *There exist extensive-form games G for which we have:*

$$\forall i \in \{0,1\}: \quad NV_i(G) > SPE_i(G).$$

Proof The Centipede Game is an example of such a game. We see from Lemma 1 and from the utility functions as defined in Section 4.3 that for this game we have $(SPE_0, SPE_1) = (5, 0)$. Furthermore, we have just shown that for this game we have $(NV_0, NV_1) = (95, 90)$. □

We see that the Centipede Game with negotiations has a number of interesting properties. Clearly, the possibility of making binding agreements removes the players' incentives to break the game off at an early stage. However, what may be more surprising, is that such agreements do not have to be made immediately. In fact, it is perfectly rational for the players to play *continue* without making any agreements at all, until they reach the state w_{16}. It is the mere fact that they know they can make agreements *later on*, which makes it rational to continue playing. This is reflected by the fact that for each state before w_{16} we have that the reservation values are equal to the negotiation values (see Fig. 1). Only once w_{16} is reached it becomes necessary to sign a binding agreement, because only at that point any threat to finish the game early becomes credible.

A second surprising property, is the fact that although the game has two terminal states that are Pareto-optimal, t_{18} and t_{19}, which both dominate the SPE, the only reasonable outcome for the players is to agree on t_{19}. The reason for this lies in the way the game is structured.

Proposition 1 *If the players are allowed to negotiate in the Centipede Game, then it is never rational for player α_0 to agree with t_{18}.*

Proof We see from the calculations above (also displayed in Fig. 1) that for all $k \leq 16$ we have $U_0(t_{18}) < rv_{w_k,\vec{\tau},0}$. So indeed, in any of those states the proposition holds. In state w_{16}, if the players do not come to any agreement, then α_0 will play *finish* because $U_0(t_{17}) = 85 > 80 = nv_{w_{17},\vec{\tau},0}$. If they do come to an agreement, then that agreement would be t_{19}, because for α_0 that is the only rational deal (because $U_0(t_{18}) < rv_{w_{16},\vec{\tau},0}$). This means that either the game never reaches state w_{17}, or the players agree on t_{19}, and therefore, for $k = 17$ and $k = 18$ the proposition only needs to be proved under the assumption that the players have agreed on t_{19}. Indeed, once t_{19} is accepted, it is also not rational for α_0 to accept t_{18}, because $U_0(t_{19}) = 95 > 80 = U_0(t_{18})$. □

Intuitively, the idea is that once w_{16} is reached, player α_0 would prefer to abort the game than to agree on t_{18}, so t_{19} is the only outcome α_0 could agree with. So, although t_{18} seems rational at the beginning of the game (with respect to the traditional SPE), it is no longer rational once w_{16} is reached. Furthermore, α_1 cannot threaten to abort the game before reaching w_{16}, because that would only yield less utility than to agree with t_{19}.

7 Implementation with Bounded Rationality

In the previous sections we have presented a purely theoretical framework. From this section onward we will look at things from a more practical point of view, taking into account *bounded* rationality. We will present a working algorithm that allows an agent to play an extensive form game whilst negotiating a joint strategy with the opponent.

Our algorithm runs on one of the two agents and aims to maximize the utility of that agent both by selecting the best moves to make and by trying to negotiate the best possible deals for that agent. Its opponent may be running the same algorithm, but may just as well be running any other algorithm, or may even be human.

7.1 Monte Carlo Negotiation Search

Our algorithm is based on a commonly used algorithm called Monte Carlo Tree Search (MCTS). MCTS is an anytime algorithm for extensive-form games. It is an anytime algorithm that returns an estimation of the best move to make, whenever time runs out. This is especially useful for games in which the state space is too large to be explored exhaustively. Given a turn-taking game G, MCTS iteratively generates a tree in which every node represents a state w of the game. For every such node it randomly samples a number of possible sequences of joint actions that could be played starting from w until a terminal state is reached. The results of this sampling procedure are used to estimate the players' utility values for the state w. A key feature of MCTS is the fact that if the algorithm is run long enough, the moves it selects correspond with the Subgame Perfect Equilibrium. For more details about MCTS we refer to [24].

MCTS is an algorithm for traditional games *without* negotiations, but we have implemented a new variant of this algorithm, which not only selects the best

actions for the player it represents, but also determines which joint strategies to propose to its opponent, and which proposals to accept from the opponent. We have further refined it with a Branch and Bound technique, similar to the one described in [3]. We call this algorithm *Monte Carlo Negotiation Search* (MCNS).

Just like regular MCTS our algorithm generates a tree in which every node represents a possible future state. However, we now not only estimate a pair of expected utility values for each node, but also a pair of Reservation Values and and a pair of Negotiation Values. Thanks to the random sampling procedure we are able to make estimations of these values without having to exhaustively explore the entire search space.

Every sequence of joint moves generated by the sampling procedure is stored as a potential agreement to propose. Although currently our algorithm only proposes such linear sequences of moves, our algorithm could in principle also work for more complex types of joint strategies. Our current implementation does not allow that, because for that we would first need to establish a communication language that allows the agent to express more complex proposals. We leave this as future work.

7.2 Experiments

We now present the experiments we have performed with our algorithm.[8] We have tested it on the following games [1,35,39]

- Iterated Prisoner's Dilemma (IPD)
- The Centipede Game (CG)
- Dollar Auction (DA)

These are classic games that are often discussed in Game Theoretical text books. Of course, they were originally not intended to be used with negotiations and they are usually analyzed under the assumption that the players do not communicate or make binding agreements. Therefore, one could argue that by allowing negotiations we have actually changed the rules of these games so we are in fact playing different games. Nevertheless, we still refer to our games-with-negotiations by their original names.

We have chosen these 3 games, because these are the only games we could find that satisfy the following criteria:

1. The game has an existing GDL description which can be found in the standard GDL repository at `http://games.ggp.org/`.
2. The game is non-zero-sum, and its SPE is far away from the Pareto-Frontier, so that there is enough room for negotiation.

We feel it is very important to stress here that writing a negotiating agent *specifically* for any of these games would be an easy task. However, the point of our algorithm is that it is entirely *generic*. We are using exactly the same algorithm for every game, without even changing a single parameter, and it could just as well be used for any other non-zero-sum game, as long as it is described in GDL.

Our agent does not have any knowledge about these games, other than their GDL descriptions. In particular, this means there is no straightforward, generic,

[8] These experiments have already been published previously in [22].

way for our agent to determine their Pareto Frontiers or their equilibrium values. Furthermore, even if an agent does know which utility vectors are Pareto-optimal, this is still not enough to negotiate successfully, because it still needs to find out which strategies the agent and its opponent need to play in order to obtain such a Pareto-optimal outcome.

For each of these games, we have let two instances of our algorithm play 100 matches against each other with negotiations and 100 matches without negotiations. When the agents play without negotiating they just apply a plain MCTS. For each game we have given the players a deadline to negotiate of 5 seconds per round. Our algorithm is implemented in Java and the experiments were performed on a HP Z1 G2 workstation with Intel Xeon E3 4x3.3GHz CPU and 8 GB RAM. We have downloaded the GDL descriptions of the above games from the standard GDL repository.

The results are displayed in Table 1. Each row represents one of the games. The first column shows the classical Subgame Perfect Equilbrium values for each game, and the fourth column shows the Negotiation Values for each game. The two center columns show the average utilities obtained by the two players over 100 matches, with and without negotiations respectively. We see that without negotiations, the outcome of every match in every game is exactly the Subgame Perfect Equilibrium, as one would expect from theory. When the players do negotiate they obtain much higher scores. In the case of the CG and the DA, the players obtain exactly their Negotiation Values in every match, again exactly as predicted by theory. That is, in the CG without negotiations in every game the active player immediately played 'finish', while with negotiations they agreed to play until t_{19}, exactly as explained in Section 6.4.

In the version of the DA we used, both players start with \$80 in the pocket and they are bidding for a prize of \$25. Each turn the active player can choose to increase his bid by \$5, or to stop the game, in which case the other player receives the prize, and both players lose the amount of money they bid. In the Appendix we show that without negotiations, in the SPE the player α_0 immediately terminates the game without making any bid (in which case no one wins the prize), so both players keep the money they originally had, i.e. both players end with \$80. On the other hand, with negotiations, the only rational deal they can make is for player α_1 to allow α_0 to make a bid and after which α_1 will terminate the game. This means that α_0 will win the prize after only 1 bid of only \$5 and α_1 will not loose any money at all, so they will end with \$100 and \$80 respectively.

In the case of the IPD the players score an average of 55 and 56 points respectively, with standard errors of 0.3 and 0.4 respectively, which is close to the theoretical values. This means that in almost every round both players played 'cooperate'. The reason that the algorithm does not achieve the theoretical optimum in the IPD is simply because the IPD has a very large search space, so it does not always find the optimal contract. Each player has 2 legal actions in each round and the game lasts for 20 rounds, so there are 4^{20} possible joint-move sequences from the initial state to any terminal state.

Overall, we conclude that our algorithm enables the players to significantly increase their scores by means of negotiation, and obtain results that are very close to optimal.

To explore a bit further how our algorithm behaves with increasing complexity of the game it is playing we have repeated the experiment with the IPD several

	SPE	Without Negotiations	With Negotiations	*NV*
IPD	$(20, 20)$	$(20, 20)$	$(56 \pm 0.3, 55 \pm 0.4)$	$(60, 60)$
CG	$(5, 0)$	$(5, 0)$	$(95, 90)$	$(95, 90)$
DA	$(80, 80)$	$(80, 80)$	$(100, 80)$	$(100, 80)$

Table 1 Without negotiations the results are exactly the Subgame Perfect Equilibrium values. With negotiations the results clearly dominate the results without negotiations and are close to optimal.

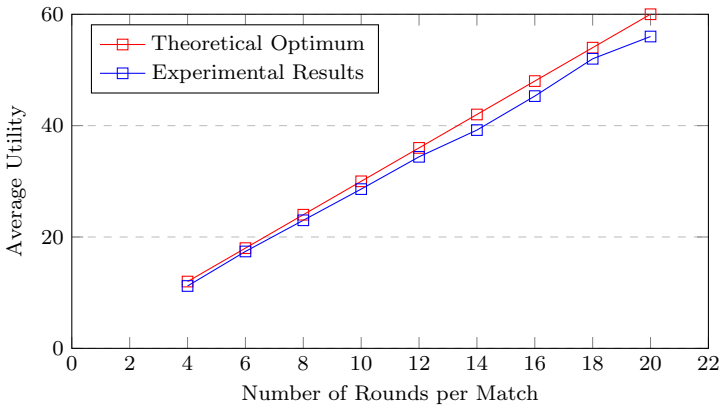

Fig. 2 The Iterated Prisoner's Dilemma, with varying number of rounds per match. The value on the vertical axis is the utility averaged over 50 matches and over the two agents.

times with an increasing number of rounds per match. The results are displayed in Figure 2. Interestingly, we see that, although the size of the agreement space increases exponentially, the results do not get much worse as the number of rounds increase. One reason for this is that many of the contracts in the agreement space are non-pareto-optimal, and many of the contracts that are Pareto-optimal have identical utility vectors, which means that many of those agreements can be ignored by the algorithm. The set of Pareto-optimal utility vectors actually only increases linearly. It just becomes harder to find the few (near-)optimal agreements. Another reason that the results only get slightly worse, is that as the number of rounds increases, the number of negotiation sessions also increases (because there is one negotiation session before each round), and therefore the total time the agents have to find good deals also increases.

Unfortunately, we cannot easily extend this experiment beyond 20 rounds per match, because in that case the maximum possible score for a player would be more than 100 points, while in GGP it is assumed that 100 is always the maximum score, so all our code is based on that assumption.

The GDL repositories do contain a number of other (more complex) non-zero-sum games, such as Free-For-All, Skirmish, Chinese Checkers, and Chinook. Unfortunately, it turns out that these games are not suitable for negotiations, because even without negotiating our pure MCTS algorithm already achieves near-optimal scores. That is, their Subgame Perfect Equilibria are already near Pareto-optimal, so there is not much benefit to be obtained from negotiations.

8 Discussion

In this section we further discuss some of the assumptions made in this paper.

8.1 Self-interested Agents

In this paper we have always assumed that agents are purely self-interested. That is, each has its own utility function, and only acts with the intention of maximizing its own utility. Therefore, there is no form of 'social utility', and agents are not implemented with the goal of maximizing any form of social welfare. Nevertheless, agents still had the incentive to make agreements and cooperate, because it was mutually beneficial.

The reason that agents still have the incentive to negotiate, is that the games we have considered are non-zero-sum games. In fact, we need a slightly stronger criterion, namely that we have games in which the Subgame Perfect Equilibrium is not Pareto-optimal. Indeed, if the SPE were Pareto-optimal, then each agent could simply play its equilibrium strategy and there would be no possible alternative outcome that improves the utility of both players with respect to the equilibrium outcome. Note that in a zero-sum game every outcome is Pareto-optimal, so the requirement that the SPE is not Pareto-optimal is indeed strictly stronger than the requirement that the game be non-zero-sum.

On the other hand, this restriction may in practice be overly strict, because even if the game is zero-sum, the players may not be aware of this because they are only bounded rational. This means that if two agents are playing a zero-sum game, they could still make an agreement if they both *believe* that this is beneficial, even though at least one of them must be wrong.

8.2 Social Welfare Criteria

Even though our agents do not aim to maximize social welfare, one might still wonder to what extent our algorithm does achieve a form of optimal social welfare. For example, one might ask whether certain criteria such as *fairness* and *envy-freeness* [4, 15] are met. However, we argue that nothing in general can be said about this, because these criteria mainly depend on the specific game that is played. We can easily show this with an example we call the 'dictatorial split-the-pie game'. In this game α_0 gets to decide how to split a pie between himself and its opponent, while the opponent gets nothing to say. Clearly, if α_0 is purely self-interested and perfectly rational, then α_0 will take the entire pie for himself and give nothing to α_1, regardless of whether they negotiate or not. This is obviously neither fair nor envy-free. If α_0 did give some portion of the pie to its opponent it would mean that either it is not purely self-interested (contradicting our assumptions) or that it is just running a really bad algorithm (for not maximizing its utility).

On the other hand, it is not hard to imagine that we could, in a similar fashion, construct a game with exactly the opposite characteristics, meaning that envy-freeness and fairness are always guaranteed.

8.3 Binding Agreements

One critical assumption we have made is that the agents are able to make binding agreements, which will always be obeyed, even if a breach of contract would increase a player's utility. We can make this assumption because we assume there is some external mechanism in place that enforces the agents to obey their agreements.

We do not think that this assumption is unrealistic, or overly restrictive, because in the real world it is very common to make binding agreements. This may be achieved by signing a contract, but in many jurisdictions even a mere verbal agreement counts as legally binding (this is known as an *oral contract*). Furthermore, when making an online purchase on the Internet, a formal contract can be agreed upon by nothing more than a few mouse clicks.

If one prefers a more traditional agent model, then one might argue that a truly autonomous agent always takes its actions purely based on a utility function, and cannot be forced by any external mechanism to do anything. In such a model, whenever an agent signs a binding contract, it only obeys that contract because the legal consequences of breaking it would impose a negative utility onto that agent. However, we argue that this model is not fundamentally different from ours. For example, we could imagine that an agent's utility function is actually the sum of a *payoff* function which is determined by the game it plays, and a *penalty* function which has a large negative value whenever the agent breaks any agreement, and is zero as long as the agent obeys all agreements. Now, if such penalties are high enough then it is never rational for any agent to break any agreements. In that case we may just as well ignore the penalty function altogether and simply assume the agent only cares about maximizing its payoff, but under the hard constraint that it always obeys its agreements. This is exactly the model that we have been using.

Another objection one might raise against our assumptions, is that we do not specify how the external enforcement mechanism would work. However, although we acknowledge that this is a very important question, we argue that this question lies beyond the scope of our work. We have simply abstracted it away and just assume it exists and works perfectly. We justify this abstraction with the following arguments.

- In traditional Game Theory the enforcer of the rules is usually also abstracted away. When programming a Chess algorithm, for example, you simply assume the opponent will only make legal moves, even if it would be rational for the opponent to make an illegal move (e.g. move a rook diagonally). The question how these rules are enforced is not relevant to the programmer.
- In real-life situations, we also constantly 'abstract away the enforcer'. For example, when opening a restaurant, you could wonder how to make sure that your customers will always pay their bills. However, we believe that in general restaurant owners do not worry about this, and simply assume that most customers will. They can do this because there are already several enforcement mechanisms in place. Firstly, people pay because they have to by law. If they do not pay they risk being caught and being prosecuted. Secondly, even if the risk of getting caught is low, customers usually still pay their bills, because they adhere to social norms. Our point is, however, that the restaurant owner

does not need to worry about how these mechanisms work exactly. Just like us, the owner simply assumes they are there and that they work.
- Of course, one might argue that real-world enforcement systems do not always work. For example, sometimes people do walk away from a restaurant without paying. However, we do not claim that our assumptions are always perfectly valid. We only claim that our assumptions are reasonable in most real-life situations. After all, we think it is safe to say that the far majority of customers in restaurants do pay their bills.
- Furthermore, even though we acknowledge that it is not always realistic to assume agreements are perfectly enforced, one can just as well turn that argument around: it is also not always realistic to assume that agents do not communicate or make binding agreements at all. Nevertheless, this assumption is commonly made in traditional Game Theory (e.g. the Prisoner's Dilemma). If one is not allowed to make such assumptions, then a vast amount of work in Game Theory would be invalid.
- Similarly, the question how agreements are enforced is rarely answered in the field of Automated Negotiations.

For our experimental setup the enforcement of agreements is taken care of by the server (Section 4.2). It checks which agreements the players make, and enforces them in the same way that the regular GGP game server enforces the rules of a game. Whenever a player submits a move that does not obey a previously established agreement, the server will ignore this move, and instead pick a random move for that player which does obey the agreement.

Finally, we remark that if agreements were not binding then they would be essentially meaningless. After all, a rational player would then always intend to play its Subgame Perfect Equilibrium strategy, regardless of any agreements it made.

8.4 Full Information

Arguably less realistic is the assumption that the agents have full information about each other's utility functions. After all, in the real world, a negotiator usually does not have perfect information about its opponent's utility. However, we still argue in favor of this assumption with three arguments.

Firstly, we should note that *formally* the agent has full information about its opponent's utility function, but this utility is not given as a closed form expression that can be calculated easily. Instead, the opponent's utility is given in terms of a GDL game description. Therefore, for any given proposal it may be computationally very hard to determine its exact utility value. Given any proposed joint strategy, our agent would first need to determine which terminal states would likely be reached given that joint strategy. Next, our agent would need to determine which terminal states would likely be reached if no agreement were made. Both tasks involve determining the Subgame Perfect Equilibrium of an extensive form game. This means that in practice the agent's knowledge of its opponent's utility function is far from perfect.

Our second argument, is that we find the opposite assumption that the agent does not know anything about its opponent's utility function at all, equally unrealistic (this assumption is commonly made in the field of Automated Negotiations).

For example, when negotiating a car deal you may not know the car salesman's reservation value, but you certainly do have some idea of which price range would be realistic. Moreover, you do know that the car salesman prefers a higher price over a lower price. In fact, we think that the skill of negotiating well is primarily based on the ability to correctly estimate what the opponent wants, and which price he or she is willing to accept. Therefore, we think our assumption is not less realistic than the assumptions made in other work.

Our third argument, is that in order to adapt our algorithm to deal with imperfect knowledge one could implement some module that approximates the opponent's utility. This module could base its estimations on information given beforehand (e.g. a domain description in GDL-II instead of GDL) as well as on information inferred from the behavior of the opponent during the negotiations. The point is, that the output of this module could be used by our algorithm to replace the perfect information it currently uses. This would make no difference for the algorithm itself. In other words: the question whether information is perfect or not may affect the accuracy of the input and the output of the algorithm, but it does not affect the implementation of the algorithm itself.

Nevertheless, we still consider it important to take imperfect information into account, but we will leave this as future work. We should also note that even in regular GGP, without negotiations, very little research has been done on games with imperfect information.

9 Future Work

In this paper we have assumed the negotiators have full information about the game. In particular, this means that for any proposed deal, our agent was able to perfectly determine the opponent's utility value of this deal. Although we do think that in real life a negotiator does have some information about the opponent's utility function, it is unrealistic to assume this information is perfect. Therefore, we plan to generalize our algorithm to games described in GDL-II [41], which allows hidden information. The games studied in this paper were highly theoretical. It would be more interesting to see if we can implement some real-world negotiation scenarios in GDL and apply our algorithm to them.

Furthermore, we would like to implement an agent that can truly negotiate joint strategies, rather than just linear sequences of joint moves. For this however, we need a language that allows the agents to communicate such joint strategies in a concise way. This could be potentially be achieved by making use of a recently introduced logical language called SGL [43], which indeed allows agents to describe (joint) strategies.

Finally, we will investigate how our algorithm can be generalized to games for more than 2 players, such as Diplomacy and to see to what extent it can be applied to other negotiation domains, such as the Genius framework [26], and the Colored Trails game [12].

Appendix A

In this section we describe the other two games that we used for our experiments. Just as with the Centipede Game, their precise definitions may vary per text book, so we follow the definitions according to their GDL descriptions implemented by Sam Schreiber, which can be found in the GDL database at `http://games.ggp.org`.

The Dollar Auction

The Dollar Auction (DA) [39] is another classic game with a somewhat counter-intuitive outcome. The idea is that an auctioneer will put up a 100-dollar bill for auction. The players of the game may make increasing bids. The player with the highest bid will pay his bid, and receive the 100-dollar bill in return. The loser however, must also pay his bid, but will receive nothing in return.

Again, the counter-intuitive result is that the best strategy is to stop the game immediately and never bid more than 0 dollars. The problem is, that if α_0 bids 98 dollars, and α_1 bids 99 dollars, then if α_0 gives up he will lose 98 dollars. Therefore, he prefers to bid 100 dollars. However, this means α_1 will lose 99 dollars if he gives up, so he will bid 101 dollars, but then α_0 will lose 100 dollars, so he will bid 102 dollars. Clearly, both players are going to lose money, so they would have been better off simply bidding 0 dollars and no more.

The implementation of the DA in GDL is a bit different from the description above, but the idea is the same. In this case both players start with 80 dollars in their pockets, and they bid for a prize of 25 dollars. The game is a turn-taking game, and in each round the active player can either choose to 'lay a claim to the prize', which costs him 5 dollars, or to finish the game in which case the other player will receive the prize (if that other player has laid at least 1 claim).

We can formalize it as follows:

$$T = \{t_k \mid k \in \{1, 2 \dots 33\}\}$$

$$W = \{w_k \mid k \in \{0, 1 \dots 32\}\} \cup T$$

$$\mathcal{A}_0 = \mathcal{A}_1 = \{\mathit{noop}, \mathit{lay\_claim}, \mathit{finish}\}$$

$$L_i(w_k) = \begin{cases} \{\mathit{noop}\} & \text{if } i \neq k \pmod 2 \\ \{\mathit{lay\_claim}, \mathit{finish}\} & \text{if } i = k \pmod 2 \text{ and } k \neq 32 \\ \{\mathit{finish}\} & \text{if } i = k \pmod 2 \text{ and } k = 32 \end{cases}$$

After playing *lay_claim* the next state will be another non-terminal state:

$$u(w_k, \mathit{lay\_claim}) = w_{k+1}$$

After playing *finish* the next state will be a terminal state:

$$u(w_k, \mathit{finish}) = t_{k+1}$$

If we let $cl_{i,k}$ denote the number of claims that have been made by player α_i when the game ends in t_k, then the utility functions are given by.

$$\vec{U}(t_k) = \begin{cases} (80, 80) & \text{if } k = 1 \\ (80 - 5 \cdot cl_{0,k} + 25 \ , \ 80 - 5 \cdot cl_{1,k}) & \text{if } k \text{ is even} \\ (80 - 5 \cdot cl_{0,k} \ , \ 80 - 5 \cdot cl_{1,k} + 25) & \text{if } k \text{ is odd and } k > 1 \end{cases}$$

which is equivalent to:

$$\vec{U}(t_k) = \begin{cases} (80, 80) & \text{if } k = 1 \\ (105 - 5 \cdot \frac{k}{2} \quad , \quad 85 - 5 \cdot \frac{k}{2}) & \text{if } k \text{ is even} \\ (80 - 5 \cdot \frac{k-1}{2} \quad , \quad 105 - 5 \cdot \frac{k-1}{2}) & \text{if } k \text{ is odd and } k > 1 \end{cases}$$

Specifically:

$$\vec{U}(t_1) = (80, 80)$$
$$\vec{U}(t_2) = (100, 80)$$
$$\vec{U}(t_3) = (75, 100)$$
$$\vec{U}(t_4) = (95, 75)$$

Note that this implementation of the DA is slightly unconventional, because if the game stops after both players have made the same number of bids then α_1 wins the prize, even though both players have bid the same amount of money.

In order to calculate the NV_i of the DA we need the following two lemmas (note that although the DA is a well-known game in the literature, these lemmas apply strictly to slightly unconventional version of the DA as defined above, so these lemmas are not taken from literature).

Lemma 3 *In the Subgame Perfect Equilibrium of the Dollar Auction, α_0 always plays 'finish', while α_1 always plays 'lay_claim'*

Proof This can be seen easily seen using the technique of backward induction (see also Figure 3). □

By combining this lemma with the utility functions of the DA as define above we immediately obtain that $(SPE_0, SPE_1) = (80, 80)$.

Lemma 4 *Let G be the dollar auction then we have:*

$$spe_{w_k,\vec{\tau},i} = \begin{cases} U_i(t_{k+1}) & \textit{if } k \textit{ is even} \\ U_i(t_{k+2}) & \textit{if } k \textit{ is odd} \end{cases}$$

Proof Suppose the game is in state w_k and assume both players follow the subgame perfect equilibrium. If k is even then α_0 is the active player, which according to Lemma 3 will play *finish*, and hence the game will end in the terminal state t_{k+1}. If k is odd then α_1 is the active player, which which according to Lemma 3 will play *lay_claim*, so the game will advance to w_{k+1}. Since $k + 1$ is even, we know that the game will end in state t_{k+2}. □

We are now ready to show that for the Dollar Auction we have $(NV_0, NV_1) = (100, 80)$.

We need to calculate $nv_{w_0,\vec{\tau},i}$. By Lemma 4 we know that for w_1 we have $\vec{spe}_{w_1,\vec{\tau}} = \vec{U}(t_3) = (75, 100)$. There is no other terminal state for which the utility vector dominates this result. Therefore, $nv_{w_1,\vec{\tau},i} = rv_{w_1,\vec{\tau},i} = U_i(t_3)$.

For w_0, the active player is α_0, so we have:

$$w^* = \underset{w' \in \{w_1, t_1\}}{\arg\max} \{nv_{w_1,\vec{\tau},0}, nv_{t_1,\vec{\tau},0}\}$$

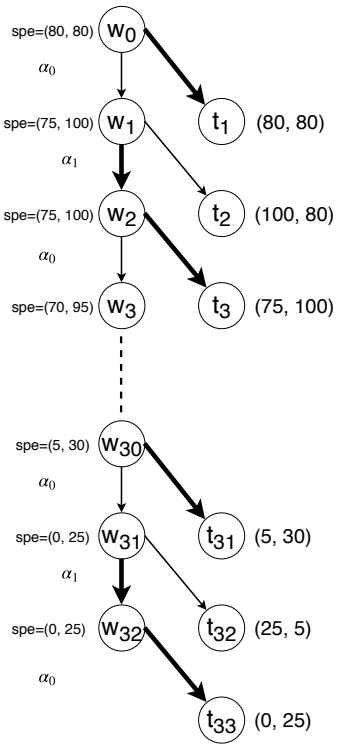

Fig. 3 State space of the Dollar Auction. Next to each terminal state t_k, on the right-hand side, we have indicated its utility vector. Next to each non-terminal state w_k, on the left-hand side, we have indicated its Subgame Perfect Equilibrium values $spe_{w,\vec{r},i}$, assuming no agreements have been made. Also, on the left-hand side next to the arrows we have indicated the active player. The thick arrows indicate the optimal moves.

with $nv_{w_1,\vec{r},0} = U_0(t_3) = 75$ and $nv_{t_1,\vec{r},0} = U_0(t_1) = 80$. So we see that $w^* = t_1$, and thus $rv_{w_0,\vec{r},i} = nv_{t_1,\vec{r},i} = U_i(t_1)$, which means the reservation values are given by $\vec{U}(t_1) = (80, 80)$. The only terminal state for which the utility vector dominates $(80, 80)$ is t_2 which has utility vector $(100, 80)$.

The Iterated Prisoner's Dilemma

The Iterated Prisoner's Dilemma (IPD) [1] is simply the Prisoner's Dilemma repeated n times. The number of repetitions n is known to the agents, and the utility functions of the game is simply defined as the sum of the outcomes of each individual iteration.

The IPD has a very large search space. Each player has 2 legal actions in each round, so if the game lasts for n rounds, there are 4^n possible sequences from the

initial state to any terminal state. Here, following the GDL description, we have $n = 20$.

We can model each state as a triple (d_0, d_1, m) where d_i is the number of points obtained so far by player α_i and m is the total number of rounds that have been played. We then have the following formalization.

$$\begin{aligned}
\mathcal{A}_i &= \{cooperate, defect\} \quad \forall i \in \{0,1\} \\
W &= \{(d_0, d_1, m) \mid d_0, d_1 \in \{0, 1, \dots 100\}, m \in \{0, 1 \dots 20\}\} \\
T &= \{(d_0, d_1, 20) \mid d_0, d_1 \in \{0, 1 \dots 100\}\} \\
w_0 &= (0,0,0) \\
L_i(w) &= \{cooperate, defect\} \quad \forall i \in \{0,1\}, \forall w \in W \setminus T \\
O &= T \\
out &= Id \\
U_i(d_0, d_1, 20) &= d_i \quad \forall i \in \{0,1\}
\end{aligned}$$

$$u((d_0, d_1, m), a_0, a_1) = \begin{cases} (d_0 + 3, d_1 + 3, m + 1) & \text{if } a_0 = cooperate, a_1 = cooperate \\ (d_0 + 0, d_1 + 5, m + 1) & \text{if } a_0 = cooperate, a_1 = defect \\ (d_0 + 5, d_1 + 0, m + 1) & \text{if } a_0 = defect, a_1 = cooperate \\ (d_0 + 1, d_1 + 1, m + 1) & \text{if } a_0 = defect, a_1 = defect \end{cases}$$

In order to calculate the SPE_i and NV_i values of the IPD we need the following lemmas. Of course, we do not claim that any of these lemmas are novel results. We only state them because we need them for our calculations.

Lemma 5 *In the Subgame Perfect Equilibrium of the Iterated Prisoner's Dilemma, every player always plays 'defect'.*

Proof This follows directly from backward induction. □

By combining this lemma with the update function of the IPD as defined above we obtain that the terminal state reached by two perfectly rational players is the state $(20, 20, 20)$ (starting from $w_0 = (0,0,0)$ apply the update function with $a_0 = defect, a_1 = defect$ twenty times). Therefore, we conclude that we have $(SPE_0, SPE_1) = (20, 20)$.

The rest of this section is dedicated to showing that $(NV_0, NV_1) = (60, 60)$.

Lemma 6 *Let $s = (\vec{a}_0 \dots \vec{a}_{19})$, be a legal sequence of joint moves of the IPD starting at the initial state, for which there is at least one $\vec{a}_j$ in which α_0 plays defect, and at least one $\vec{a}_m$ in which α_1 plays defect. Then the resulting terminal state is not Pareto-optimal.*

Proof We will just prove this for the case that: $\vec{a}_j = (defect, cooperate)$ and $\vec{a}_m = (cooperate, defect)$. Let t_s denote the resulting terminal state of s. Furthermore, let s' denote the sequence that is identical to s except that both $\vec{a}_j$ and $\vec{a}_m$ are replaced with $(cooperate, cooperate)$, and let $t_{s'}$ denote the resulting state of s'. A simple calculation yields:

$$U_0(t_{s'}) = U_0(t_s) + 1 \quad U_1(t_{s'}) = U_1(t_s) + 1$$

We see that $t_{s'}$ dominates t_s. We leave the other cases for the reader to verify. □

Let $t_{i,k}$ denote the terminal state that results after a game in which α_i has played 'defect' k times, and its opponent has always has played 'cooperate'. Then one can easily verify from the above definition of the IPD that the players' payoffs are given by:

$$U_i(t_{i,k}) = k \cdot 5 + (20-k) \cdot 3 = 60 + 2k \tag{11}$$

$$U_{-i}(t_{i,k}) = (20-k) \cdot 3 = 60 - 3k \tag{12}$$

where U_{-i} is the utility function of the opponent of α_i.

Lemma 7 *The set of Pareto-optimal terminal states of the IPD consists of exactly those terminal states that result from a legal action sequence in which at least one of the two players always plays 'cooperate'.*

Proof We already know from Lemma 6 that if a sequence of joint moves s results in a Pareto-optimal terminal state t, then at least one of the two players only plays 'cooperate' in s. So we just need to prove the converse: any terminal state resulting from such a sequence is Pareto-optimal. We use the same notation as above and denote such a terminal state by $t_{i,k}$. Without loss of generality we can assume that $i = 0$, because the proof for $i = 1$ goes analogously. If $t_{0,k}$ is not Pareto-optimal then it must be dominated by some Pareto-optimal solution. Therefore, from Lemma 6 we know that $t_{0,k}$ must be dominated either by a state of the form $t_{0,l}$, or by a state of the form $t_{1,l}$. First we prove that $t_{0,k}$ cannot be dominated by $t_{0,l}$. This follows because if $l < k$ then:

$$U_0(t_{0,l}) \quad = \quad 60 + 2l \quad < \quad 60 + 2k \quad = \quad U_0(t_{0,k}),$$

while if $k < l$ we have:

$$U_1(t_{0,l}) \quad = \quad 60 - 3l \quad < \quad 60 - 3k \quad = \quad U_1(t_{0,k}).$$

Next, we prove that $t_{0,k}$ cannot be dominated by $t_{1,l}$. This follows because for all k, l:

$$U_0(t_{1,l}) \quad = \quad 60 - 3l \quad < \quad 60 + 2k \quad = \quad U_0(t_{0,k}).$$

□

Lemma 8 *Suppose the game is in a state (d, d, k), then the only terminal state that is reachable from (d, d, k) which is Pareto-optimal and for which both players obtain a score higher than $d + 58 - 3k$ is the state that results from both players always playing cooperate.*

Proof If the game is in state (d, d, k) then there are $20 - k$ rounds to go. If both players continue by only playing *cooperate* in every remaining round, then each player will gain 3 points in each round, so each will obtain a final of $d+3\cdot(20-k) = d + 60 - 3k$. Indeed, this is greater than $d + 58 - 3k$.

Now, suppose α_i always plays *cooperate*, and α_j plays *defect* n times, with $1 \leq n \leq 20 - k$. It is easy to calculate that α_i will end with $d + 3 \cdot ((20 - k) - n)$ points, and since $n \geq 1$ we have:

$$d + 3 \cdot ((20-k) - n) \quad = \quad d + 60 - 3k - 3n \quad < \quad d + 58 - 3k.$$

Finally, if both players play *defect* at least once, then we can use similar reasoning as in Lemma 6 to show that this is not Pareto-optimal. □

We will now calculate the values of NV_i. However, before doing this we should remark that strictly speaking the quantities NV_i and SPE_i are not defined for the IPD, because the IPD is not a turn-taking game. However, with a small adaptation specific to the IPD we can make these concepts make sense.

If G is the IPD, and $N_{w,\vec{\sigma}}$ the negotiation domain for any state w and any standing agreement $\vec{\sigma}$. Then, we define the utility value $U_{w,\vec{\sigma},i}(\vec{\sigma}')$ of any agreement $\vec{\sigma}'$ by assuming both players will always play *defect*, except when it is in contradiction with the agreement $\vec{\sigma}'$. Note that this definition make sense, because indeed, we can assume that without any agreements a rational player would always play *defect*. From this, it follows that we have $rv_{w,\vec{\tau},i} = nv_{w^*,\vec{\tau},i}$, where $w^* := u(w, \mathit{defect}, \mathit{defect})$.

Let $w_{d,k}$ denote the state (d, d, k). We will show that for all $i \in \{0, 1\}$ and all integers d we have:

$$nv_{w_{d,k},\vec{\tau},i} = d + 60 - 3k$$

We will prove this by 'reversed induction'. That is, we first prove it for $k = 19$, and then we prove that if it holds for some k, then it must also hold for $k - 1$. Note that we have $w^*_{d,k} = (d + 1, d + 1, k + 1)$.

First, let k be 19, so $w^*_{d,19} = (d + 1, d + 1, 20) \in T$. Then, because this is a terminal state, for any integer d we have

$$rv_{w_{d,19},\vec{\tau},i} = nv_{w^*_{d,19},\vec{\tau},i} = U_i(w^*_{d,19}) = U_i(d + 1, d + 1, 20) = d + 1. \tag{13}$$

Now, if the game is in state $w_{d,19}$ there is only one more round to go, so it is easy to verify that the only terminal state with a utility vector that dominates this one would be the state resulting from both players playing *cooperate*. Thus, the agreement to do that is the only rational agreement, and its utility vector is $(d + 3, d + 3)$, so we have

$$nv_{w_{d,19},\vec{\tau},i} = d + 3$$

which is indeed equal to $d + 60 - 3k$, with $k = 19$. This proves the base case.

Now suppose we have proved the theorem for $k + 1$. We now want to prove that it also holds for k. We have:

$$rv_{w_{d,k},\vec{\tau},i} = nv_{w^*_{d,k},\vec{\tau},i} = nv_{w_{d+1,k+1},\vec{\tau},i} = (d + 1) + 60 - 3 \cdot (k + 1)$$

which can be rewritten as:

$$rv_{w_{d,k},\vec{\tau},i} = d + 58 - 3k$$

Now, using Lemma 8 we see that the only possible agreement that is rational and Pareto-optimal, is the deal in which both players agree to always play *cooperate*. In this case both players will receive $d + 3 \cdot (20 - k)$ points, from which it follows that:

$$nv_{w_{d,k},\vec{\tau},i} = d + 3 \cdot (20 - k) = d + 60 - 3k$$

If we now set $d = k = 0$ then we obtain our result that $(NV_0, NV_1) = (60, 60)$

Appendix B

In this section we prove that concession games in general have multiple Nash Equilibria.

A normal-form game for two players is called a *concession game* if both players have the same set of actions, this set is an ordered set $\mathcal{A} = \{a_0, a_1, \dots a_n\}$, and whenever $k < m$ the payoff functions satisfy the following (in)equalities:

$$0 < \mathcal{U}_0(a_m, a_m) < \mathcal{U}_0(a_k, a_k) \tag{14}$$

$$0 < \mathcal{U}_1(a_k, a_k) < \mathcal{U}_1(a_m, a_m) \tag{15}$$

$$\forall i \in \{0,1\}\ \mathcal{U}_i(a_k, a_m) = 0 \tag{16}$$

$$\forall i \in \{0,1\}\ \mathcal{U}_i(a_m, a_k) = \frac{1}{2}(\mathcal{U}_i(a_k, a_k) + \mathcal{U}_i(a_m, a_m)) > 0 \tag{17}$$

We now aim to characterize the Nash Equilibria for such a game.

Lemma 9 *If $k < l \leq m$ then we have $\mathcal{U}_1(a_m, a_k) < \mathcal{U}_1(a_m, a_l)$.*

Proof By (15) we have:

$$\mathcal{U}_1(a_k, a_k) < \mathcal{U}_1(a_l, a_l)$$

from this it follows that:

$$\frac{1}{2}(\mathcal{U}_i(a_k, a_k) + \mathcal{U}_i(a_m, a_m)) < \frac{1}{2}(\mathcal{U}_i(a_l, a_l) + \mathcal{U}_i(a_m, a_m))$$

then, by (17) we have that the left-hand side is equal to $\mathcal{U}_1(a_m, a_k)$ while the right-and side is equal to $\mathcal{U}_1(a_m, a_l)$ so we have:

$$\mathcal{U}_1(a_m, a_k) < \mathcal{U}_1(a_m, a_l)$$

□

Theorem 4 *Let S be some subset of $\mathcal{A}$ and let a_k be any action that is not in S, i.e. $a_k \in \mathcal{A} \setminus S$. If one player plays a mixed strategy with support S, then playing a_k is not a best response for the other player.*

Proof Suppose agent α_0 is playing a mixed strategy in which each action a_i has a probability P_i of being played. If its support is S this means that $P_i = 0$ iff $a_i \notin S$. The expected utility of α_1, when playing a_k is then given by:

$$E(\mathcal{U}_1(a_k)) = \sum_{i=0}^{n} P_i \cdot \mathcal{U}_1(a_i, a_k) \tag{18}$$

We need to prove that there is some other action a_l for which $E(\mathcal{U}_1(a_l)) > E(\mathcal{U}_1(a_k))$. In order to prove this we need to consider two different cases, namely the case that $i < k$ for all $a_i \in S$, and the case that there is at least one integer i such that $k < i$ and $a_i \in S$.

Case 1 Suppose that for all $a_i \in S$ we have $i < k$. Note that if $i < k$ then by (16) we have $\mathcal{U}_1(a_i, a_k) = 0$, while if $i \geq k$ we have $P_i = 0$ because $a_i \notin S$. Therefore, every term in the summation in Eq. (18) equals 0, so we have:

$$E(\mathcal{U}_1(a_k)) = 0$$

Now let l be largest integer such that $a_l \in S$. Then we have:

$$E(\mathcal{U}_1(a_l)) = \sum_{i=0}^{n} P_i \cdot \mathcal{U}_1(a_i, a_l) = P_l \cdot \mathcal{U}_1(a_l, a_l) > 0$$

Here, the second equality holds because for all $i < l$ we have $\mathcal{U}_1(a_i, a_l) = 0$, and for all $i > l$ we have $P_i = 0$, so the only term that does not vanish is the term with $i = l$. We have now shown that $E(\mathcal{U}_1(a_l)) > E(\mathcal{U}_1(a_k))$ so we have proven the proposition for this case.

Case 2 Now suppose there is at least one integer j such that $a_j \in S$ and $k < j$. Let l be the smallest such integer. Since $\mathcal{U}_1(a_i, a_k) = 0$ for all $i < k$, and $P_i = 0$ for all i with $k \leq i < l$ we can rewrite Equation (18) as:

$$E(\mathcal{U}_1(a_k)) = \sum_{i=l}^{n} P_i \cdot \mathcal{U}_1(a_i, a_k)$$

Note that for each term in this summation we have that $k < l \leq i$, so we can apply Lemma 9 and conclude that

$$E(\mathcal{U}_1(a_k)) = \sum_{i=l}^{n} P_i \cdot \mathcal{U}_1(a_i, a_k) < \sum_{i=l}^{n} P_i \cdot \mathcal{U}_1(a_i, a_l) = E(\mathcal{U}_1(a_l))$$

We have proven that the proposition also holds for this case, so it holds in general. □

Corollary 1 *In any Mixed Strategy Nash Equilibrium of a Concession Game, α_0 and α_1 must choose exactly the same support.*

Proof Let S_0 denote the support of α_0 and S_1 the support of α_1. In a mixed strategy Nash Equilibrium, every action in the support of α_0 must be a best response to α_1, and vice versa. Therefore, if $a \in S_1$ then it must be a best response to α_0 and by Theorem 4 we then must have that $a \in S_0$. Similarly, if $a \in S_0$ then it must be a best response to α_1 and therefore it must be in S_1. □

We now know that if the players play a Nash Equilibrium with supports S_0 and S_1 respectively, then $S_0 = S_1$. However, that does not mean that any subset $S \subseteq \mathcal{A}$ can be the support of some Nash Equilibrium. The following three propositions show that at least if $0 < |S| \leq 3$ then there is a Nash Equilibrium with support S.

Proposition 2 *For any integer i with $0 \leq i \leq n$ the pure strategy profile defined by both players choosing action a_i is a pure Nash Equilibrium.*

Proof This follows directly from Theorem 4 by setting $S = \{a_i\}$.

Proposition 3 *For any subset $S \subseteq \mathcal{A}$ of size 2 there exists a Mixed Strategy Nash Equilibrium with supports $S_0 = S_1 = S$.*

Proof Assume α_0 plays a strategy with support $S = \{a_i, a_j\}$. Let P_1 denote the probability of playing a_i and P_2 the probability of playing a_j. Furthermore, let us

define $A = \mathcal{U}_1(a_i, a_i)$, and $B = \mathcal{U}_1(a_j, a_j)$, with $0 < A < B$ The question now is whether there exists a solution that makes the following two expressions equal:

$$E(\mathcal{U}_1(a_i)) = P_1 \cdot A + P_2 \cdot \frac{1}{2}(A + B)$$
$$E(\mathcal{U}_1(a_j)) = P_1 \cdot 0 + P_2 \cdot B$$

with $P_1 > 0$, $P_2 > 0$, and $P_1 + P_2 = 1$. It is easy to verify that the following solution solves the equation:

$$P_1 = \frac{B - A}{A + B} \qquad P_2 = \frac{2A}{A + B}$$

We should still prove that every action in S is also a best response to α_1 playing a strategy with support S, but this goes analogously. □

Proposition 4 *For any subset $S \subseteq \mathcal{A}$ of size 3 there exists a Mixed Strategy Nash Equilibrium with supports $S_0 = S_1 = S$.*

Proof Assume α_0 plays a strategy with support $S = \{a_i, a_j, a_k\}$. Let P_1, P_2 and P_3 denote their respective probabilities. Furthermore, let us define $A = \mathcal{U}_1(a_i, a_i)$, $B = \mathcal{U}_1(a_j, a_j)$ and $C = \mathcal{U}_1(a_k, a_k)$, with $0 < A < B < C$. We need to equate these three expressions:

$$E(\mathcal{U}_1(a_i)) = P_1 \cdot A + P_2 \cdot \frac{1}{2}(A + B) + P_3 \cdot \frac{1}{2}(A + C)$$
$$E(\mathcal{U}_1(a_j)) = P_1 \cdot 0 + P_2 \cdot B + P_3 \cdot \frac{1}{2}(B + C)$$
$$E(\mathcal{U}_1(a_k)) = P_1 \cdot 0 + P_2 \cdot 0 + P_3 \cdot C$$

with all P_i positive and $P_1 + P_2 + P_3 = 1$. One can verify that the following is a solution:

$$P_1 = \frac{CB + B^2 - AC - AB}{AB + AC + B^2 + BC}$$
$$P_2 = \frac{2AC - 2AB}{AB + AC + B^2 + BC}$$
$$P_3 = \frac{4AB}{AB + AC + B^2 + BC}$$

Again, the case that α_1 plays with support S goes analogously. □

We suspect that the Propositions 2, 3 and 4 can be generalized to subsets of any size, but we leave this as an open conjecture.

Conjecture 1 *Let S be any subset of $\mathcal{A}$. Then there exists a Mixed Strategy Nash Equilibrium with supports $S_0 = S_1 = S$.*

Appendix C

In this Appendix we give a complete formalization of the concept of an Extensive-Form Game with Negotiations.

Definition 14 Let $\vec{\mathcal{A}} = (\mathcal{A}_0, \mathcal{A}_1)$ be some pair of sets of actions. A **simple protocol** for $\vec{\mathcal{A}}$, denoted $Pr^{\vec{\mathcal{A}}}$, is a first-order protocol $Pr = \langle \vec{\alpha}, \vec{\mathcal{A}}, W, w_0, T, \vec{L}, u, O, out \rangle$ such that:

- $W = \{w_0\} \cup T$
- $T = \{t_{a_0,a_1} \mid (a_0, a_1) \in \mathcal{A}_0 \times \mathcal{A}_1\}$
- $L_i(w_0) = \mathcal{A}_i$ for all $i \in \{0, 1\}$
- $u(w_0, a_0, a_1) = t_{a_0,a_1}$ for all $(a_0, a_1) \in \mathcal{A}_0 \times \mathcal{A}_1$.
- $O = \mathcal{A}_0 \times \mathcal{A}_1$
- $out(t_{a_0,a_1}) = (a_0, a_1)$ for all $(a_0, a_1) \in \mathcal{A}_0 \times \mathcal{A}_1$.

A simple protocol is indeed in a certain sense the simplest possible protocol, since it only consists of each player picking one action, each terminal state directly corresponds to the chosen pair of actions, and is also labeled with that same pair of actions.

We will now define the concept of a 'higher order protocol', which is essentially a nested protocol in which each state corresponds to a lower-order protocol.

Definition 15 Let n be an integer with $n > 1$. A **protocol of order n** is a tuple $Pr = \langle W, w_0, T, \mathcal{P}, O, u, out \rangle$, where:

- W is a finite set of **states**.
- $w_0 \in W$ is the **initial state**.
- $T \subset W$ is the set of **terminal states**.
- $\mathcal{P}$ is the **protocol map** that assigns to each non-terminal state $w \in W \setminus T$ a protocol $\mathcal{P}(w)$ of order $n - 1$.
- u is the **update function** that maps each pair (w, o) consisting of a non-terminal state w and an outcome o of the corresponding protocol $\mathcal{P}(w)$ to a new state $w' = u(w, o) \in W$.
- O is the **outcome set**.
- out is the outcome function $out : T \to O$ that maps each terminal state to an outcome.

Informally, this definition means that if we have a protocol of order 2, then in each state w the agents need to choose a sequence of joint actions according to some first-order protocol $\mathcal{P}(w)$. The outcome of this first-order protocol will determine the next state of the second-order protocol.

In the following, the notation W^G, and u^G represent the set of states and the update function of the game G respectively, and similarly for T^G, O^G, out^G and $\vec{U}^G$.

Definition 16 Let G be an extensive-form game. Then we define a **Extensive-Form Game With Negotiations** NG over G as a protocol of order 2 together with a pair of utility functions, with the following properties:

1. The set of non-terminal states of NG is partitioned into a set of *negotiation states*, and a set of *action states*: $W \setminus T = W_{nego} \cup W_{action}$, s.t. $W_{nego} \cap W_{action} = \emptyset$. For each pair $(w, \vec{\sigma})$ there is one action state, denoted $act_{w,\vec{\sigma}}$, and one negotiation state, denoted $neg_{w,\vec{\sigma}}$:

- $W_{nego} = \{neg_{w,\vec{\sigma}} \mid w \in W^G \setminus T^G, \quad \vec{\sigma} \in \mathcal{S}^G\}$
- $W_{action} = \{act_{w,\vec{\sigma}} \mid w \in W^G \setminus T^G, \quad \vec{\sigma} \in \mathcal{S}^G\}$

2. The initial state of NG is the negotiation state $neg_{w_0,\vec{\tau}}$ where w_0 is the initial state of G and $\vec{\tau}$ is the trivial joint strategy of G.
3. $T = T^G$.
4. $\mathcal{P}$ assigns to each negotiation state $neg_{w,\vec{\sigma}}$ a negotiation protocol:
$$\mathcal{P}(neg_{w,\vec{\sigma}}) = N_{w,\vec{\sigma}}$$
for which $Agr = \mathcal{S}^G$.
5. $\mathcal{P}$ assigns to each action state $act_{w,\vec{\sigma}}$ the simple protocol (Def. 14) defined by the actions $\mathcal{A}_i = \sigma_i(w)$:
$$\mathcal{P}(act_{w,\vec{\sigma}}) = Pr^{\vec{\sigma}(w)}$$
6. The update function is defined as follows:
   (a) $u(neg_{w,\vec{\sigma}}, \vec{\sigma}') = act_{w,\vec{\sigma}'}$
   (b) $u(neg_{w,\vec{\sigma}}, \eta) = act_{w,\vec{\sigma}}$.
   (c) $u(act_{w,\vec{\sigma}}, a_0, a_1) = neg_{w',\vec{\sigma}}, \quad$ where $w' = u^G(w, a_0, a_1), \quad$ if $w' \notin T$
   (d) $u(act_{w,\vec{\sigma}}, a_0, a_1) = w' \quad$ where $w' = u^G(w, a_0, a_1), \quad$ if $w' \in T$.
7. $O = O^G$
8. $out = out^G$
9. $\vec{U} = \vec{U}^G$

For each action state $act_{w,\vec{\sigma}}$ or negotiation state $neg_{w,\vec{\sigma}}$ we call $\vec{\sigma}$ the **standing agreement**. Let us now discuss a number of properties of NG:

- This second-order protocol has three types of states: actions states, negotiation states and terminal states (Line 1).
- Every negotiation state is followed by an action state (Lines 6a and 6b).
- Every action state is followed by either a negotiation state or a terminal state (Lines 6c and 6d).
- In each action state $act_{w,\vec{\sigma}}$ each agent selects an action from the state w of the game G (Line 5).
- In each action stage $act_{w,\vec{\sigma}}$ the agents must obey the standing agreement, that is: each α_i must choose its actions from $\sigma_i(w)$ (Line 5).
- In each negotiation state $neg_{w,\vec{\sigma}}$ the agents negotiate a joint strategy for G (Line 4).
- When the negotiators agree on some joint strategy σ' then that will become the new standing agreement (Line 6a).
- When the negotiators do not come to an agreement then the currently standing agreement remains the standing agreement (Line 6b).

Intuitively, this means that NG alternates between action states and negotiation states, where in each action state $act_{w,\vec{\sigma}}$ the players choose some action from the original game G, but under the restriction that they have to obey the earlier agreement $\vec{\sigma}$, and where in each negotiation stage $neg_{w,\vec{\sigma}}$ the players have the change to re-negotiate a new agreement.

The initial state of NG is $neg_{w_0,\vec{\tau}}$, which means that initially, before any agreement has been made, the trivial joint strategy $\vec{\tau}$ is considered the standing agreement. This is not a restriction, because the trivial joint strategy, by definition, does not pose any restrictions on the agents' actions at all. In other words, it is equivalent to saying that there is no standing agreement.

References

1. R Axelrod and WD Hamilton. The evolution of cooperation. *Science*, 211(4489):1390–1396, 1981.
2. Tim Baarslag, Koen Hindriks, Catholijn M. Jonker, Sarit Kraus, and Raz Lin. The first automated negotiating agents competition (ANAC 2010). In Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo, editors, *New Trends in Agent-based Complex Automated Negotiations, Series of Studies in Computational Intelligence*. Springer-Verlag, 2010.
3. Tristan Cazenave and Abdallah Saffidine. Score bounded monte-carlo tree search. In H. Jaap van den Herik, Hiroyuki Iida, and Aske Plaat, editors, *Computers and Games - 7th International Conference, CG 2010, Kanazawa, Japan, September 24-26, 2010, Revised Selected Papers*, volume 6515 of *Lecture Notes in Computer Science*, pages 93–104. Springer, 2010.
4. Yann Chevaleyre, Paul E. Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaître, Nicolas Maudet, Julian A. Padget, Steve Phelps, Juan A. Rodríguez-Aguilar, and Paulo Sousa. Issues in multiagent resource allocation. *Informatica (Slovenia)*, 30(1):3–31, 2006.
5. Angela Fabregues. *Facing the Challenge of Automated Negotiations with Humans*. PhD thesis, Universitat Autònoma de Barcelona, 2012.
6. Angela Fabregues and Carles Sierra. Dipgame: a challenging negotiation testbed. *Engineering Applications of Artificial Intelligence*, 2011.
7. Peyman Faratin, Carles Sierra, and Nicholas R. Jennings. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3-4):159 – 182, 1998. Multi-Agent Rationality.
8. Peyman Faratin, Carles Sierra, and Nicholas R. Jennings. Using similarity criteria to make negotiation trade-offs. In *International Conference on Multi-Agent Systems, ICMAS'00*, pages 119–126, 2000.
9. Shaheen Fatima, Michael Wooldridge, and Nicholas R. Jennings. An analysis of feasible solutions for multi-issue negotiation involving nonlinear utility functions. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 1041–1048, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.
10. André Ferreira, Henrique Lopes Cardoso, and Luís Paulo Reis. Dipblue: A diplomacy agent with strategic and trust reasoning. In *7th International Conference on Agents and Artificial Intelligence (ICAART 2015)*, pages 398–405, 2015.
11. Hilmar Finnsson. *Simulation-Based General Game Playing*. PhD thesis, School of Computer Science, Reykjavik University, 2012.
12. Y. Gal, B. Grosz, S. Kraus, A. Pfeffer, and S. Shieber. Agent decision-making in open-mixed networks. *Artificial Intelligence*, 2010.
13. M. Genesereth, N. Love, and B. Pell. General game playing: Overview of the AAAI competition. *AI Magazine*, 26(2):62–72, 2005.
14. Takayuki Ito, Mark Klein, and Hiromitsu Hattori. A multi-issue negotiation protocol among agents with nonlinear utility functions. *Multiagent Grid Syst.*, 4:67–83, January 2008.
15. Karthik Iyer and Michael N. Huhns. Negotiation criteria for multiagent resource allocation. *The Knowledge Engineering Review*, 24(2):111–135, 2009.
16. Dave de Jonge. *Negotiations over Large Agreement Spaces*. PhD thesis, Universitat Autònoma de Barcelona, 2015.
17. Dave de Jonge, Tim Baarslag, Reyhan Aydoğan, Catholijn Jonker, Katsuhide Fujita, and Takayuki Ito. The challenge of negotiation in the game of diplomacy. In Marin Lujak, editor, *Agreement Technologies 2018, Revised Selected Papers*, pages 100–114, Cham, 2019. Springer International Publishing.
18. Dave de Jonge and Carles Sierra. NB3: a multilateral negotiation algorithm for large, non-linear agreement spaces with limited time. *Autonomous Agents and Multi-Agent Systems*, 29(5):896–942, 2015.
19. Dave de Jonge and Carles Sierra. GANGSTER: an automated negotiator applying genetic algorithms. In Naoki Fukuta, Takayuki Ito, Minjie Zhang, Katsuhide Fujita, and Valentin Robu, editors, *Recent Advances in Agent-based Complex Automated Negotiation*, Studies in Computational Intelligence, pages 225–234. Springer International Publishing, 2016.
20. Dave de Jonge and Carles Sierra. D-brane: a diplomacy playing agent for automated negotiations research. *Applied Intelligence*, pages 1–20, 2017.

21. Dave de Jonge and Dongmo Zhang. Using GDL to represent domain knowledge for automated negotiations. In Nardine Osman and Carles Sierra, editors, *Autonomous Agents and Multiagent Systems: AAMAS 2016 Workshops, Visionary Papers, Singapore, Singapore, May 9-10, 2016, Revised Selected Papers*, pages 134–153, Cham, 2016. Springer International Publishing.
22. Dave de Jonge and Dongmo Zhang. Automated negotiations for general game playing. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 371–379. ACM, 2017.
23. Donald E. Knuth and Ronald W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293 – 326, 1975.
24. Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg, 2006. Springer-Verlag.
25. Sarit Kraus and Daniel Lehmann. Designing and building a negotiating automated agent. *Computational Intelligence*, 11:132–171, 1995.
26. Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M. Jonker. Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, 30(1):48–70, 2014.
27. Nathaniel Love, Michael Genesereth, and Timothy Hinrichs. General game playing: Game description language specification. Technical Report LG-2006-01, Stanford University, Stanford, CA, 2006. http://logic.stanford.edu/reports/LG-2006-01.pdf.
28. Ivan Marsa-Maestre, Miguel A. Lopez-Carmona, Juan R. Velasco, and Enrique de la Hoz. Effective bidding and deal identification for negotiations in highly nonlinear scenarios. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 1057–1064, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.
29. Ivan Marsa-Maestre, Miguel A. Lopez-Carmona, Juan R. Velasco, Takayuki Ito, Mark Klein, and Katsuhide Fujita. Balancing utility and deal probability for auction-based negotiations in highly nonlinear utility spaces. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 214–219, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.
30. J.F. Nash. The bargaining problem. *"Econometrica"*, "18":155–162, 1950.
31. Martin J. Osborne and Ariel Rubinstein. *Bargaining and Markets*. Academic Press, 1990.
32. M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.
33. Li Pan, Xudong Luo, Xiangxu Meng, Chunyan Miao, Minghua He, and Xingchen Guo. A two-stage win-win multiattribute negotiation model: Optimization and then concession. *Computational Intelligence*, 29(4):577–626, 2013.
34. J. S. Rosenschein and G. Zlotkin. *Rules of Encounter*. The MIT Press, Cambridge, USA, 1994.
35. Robert W Rosenthal. Games of perfect information, predatory pricing and the chain-store paradox. *Journal of Economic Theory*, 25(1):92 – 100, 1981.
36. E. Ephrati S. Kraus, D. Lehman. An automated diplomacy player. In D. Levy and D. Beal, editors, *Heuristic Programming in Artificial Intelligence: The 1st Computer Olympia*, pages 134–153. Ellis Horwood Limited, 1989.
37. Stephan Schiffel and Michael Thielscher. M.: Fluxplayer: A successful general game player. In *In: Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 1191–1196. AAAI Press, 2007.
38. Roberto Serrano. bargaining. In Steven N. Durlauf and Lawrence E. Blume, editors, *The New Palgrave Dictionary of Economics*. Palgrave Macmillan, Basingstoke, 2008.
39. Martin Shubik. The dollar auction game: A paradox in noncooperative behavior and escalation. *The Journal of Conflict Resolution*, 15(1):109–111, 1971.
40. David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
41. Michael Thielscher. A general game description language for incomplete information games. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

42. John von Neumann. On the theory of games of strategy. In A.W. Tucker and R.D. Luce, editors, *Contributions to the Theory of Games*, pages 13–42. Princeton University Press, 1959.
43. Dongmo Zhang and Michael Thielscher. A logic for reasoning about game strategies. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 1671–1677, 2015.