Learning in automated negotiation

Renting, B.M.

LEARNING IN AUTOMATED NEGOTIATION

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op donderdag 11 december 2025
klokke 11.30 uur

door

Bram Matthias RENTING

geboren te Zwolle, Nederland

Promotores:
Prof.dr. H.H. Hoos — Leiden University, RWTH Aachen University
Prof.dr. C.M. Jonker — Leiden University, Delft University of Technology

Co-promotor:
Dr. T.M. Moerland

Promotiecommissie:
Prof.dr. M.M. Bonsangue
Prof.dr. A. Plaat
Prof.dr. M.T.J. Spaan — Delft University of Technology
Prof.dr. S. Trimpe — RWTH Aachen University
Prof.dr. P. Yolum — Utrecht University
Dr. A.V. Kononova

Keywords: Automated Negotiation, Multi-agent systems, Algorithm configuration, Reinforcement learning

Printed by: proefschriftspecialist.nl

An electronic version of this dissertation is available at
https://scholarlypublications.universiteitleiden.nl/

The work in this dissertation was funded by the Netherlands Organisation for Scientific Research (NWO) through the Hybrid Intelligence Centre via the Zwaartekracht grant (024.004.022).

SUMMARY

This dissertation investigates the design and evaluation of automated negotiation agents within the area of multi-agent systems, with an emphasis on the development of learning negotiation agents to improve performance and generalisability across diverse negotiation settings. Traditional approaches to developing negotiation agents rely on manually designed heuristics and strategies. While effective in specific scenarios, such methods suffer from limitations including human development cost, poor generalisation across different negotiation problems and opponent types, and the introduction of human-induced biases in strategy design. The first part of this dissertation addresses these limitations by exploring methods for learning negotiation strategies. We progress through several methods that enable autonomous agents to learn effective negotiation behaviours.

Chapter 3 presents an automated algorithm configuration technique, specifically Sequential Model-based Algorithm Configuration (SMAC) [75], for the optimisation of the parameters of a manually defined negotiation strategy. This approach demonstrates the potential to improve upon manually tuned or literature-based configurations through computational optimisation, while still relying on a predefined, parametrised strategy structure and manually engineered instance features to guide the configuration process.

Recognising that a single optimised strategy may not be universally optimal, Chapter 4 extends this work by exploring portfolio-based approaches. Using methods such as Hydra [162] for portfolio construction and AutoFolio [100] for per-instance algorithm selection, a portfolio of complementary negotiation strategies is automatically generated, and a selector is trained. This allows the agent to dynamically choose the most suitable strategy from its portfolio based on the characteristics of the current negotiation scenario and opponent, leading to improved adaptiveness and overall performance compared to relying on a single best strategy.

To further reduce the reliance on manual design and potentially mitigate human-induced biases more substantially, Chapter 5 investigates an end-to-end reinforcement learning approach. This method employs Proximal Policy Optimisation (PPO) [135] and uses Graph Neural Networks (GNNs) [86] to handle the variable dimensionality in the observation and action spaces of diverse negotiation problems. This allows the negotiation policy to be learned directly from interaction data without explicit feature engineering or manually designed parametrised strategies. This facilitates further mitigation of human-induced biases and generalisation across negotiation problems of varying sizes and complexities, although with remaining challenges in effective adaptation to opponent types.

In the second part of this dissertation, we move beyond the development of learning agents by critically examining the methods used for their evaluation. Chapter 6 presents an extensive empirical analysis, using data from the Automated

Negotiation Agents Competition (ANAC) 2022, which we organised ourselves and specifically challenged participants to develop learning agents. The analysis shows limitations in current evaluation methods. It demonstrates that agent rankings depend on the choice of performance criteria, such as individual utility, social welfare, or game-theoretic equilibria. It also shows that the commonly used average utility metric is sensitive to group composition and cannot handle non-transitive performance relations between agents. The conclusion drawn is that there is no single, universally robust metric for evaluating negotiation agents, particularly for learning agents that exhibit non-stationary behaviour.

Addressing the identified need for clear evaluation criteria and research challenges grounded in negotiation application domains, Chapter 7 proposes multi-agent calendar scheduling as a rich, relevant, and complex real-world application area. This task contains many challenges for the automated negotiation community, potentially guiding future research efforts to push progress beyond the current boundaries in the automated negotiation community.

In summary, this dissertation contributes methods for developing negotiation agents capable of learning and adapting their strategies, pushing towards reduced human bias and increased generalisability. It also provides a critical analysis of evaluation methodologies in automated negotiation, highlighting their shortcomings and advocating for research into evaluation methods, potentially using application areas like calendar scheduling.

SAMENVATTING

Dit proefschrift onderzoekt het ontwerp en de evaluatie van geautomatiseerde onderhandelingsagenten binnen het domein van multi-agent systemen, met de nadruk op de ontwikkeling van lerende onderhandelingsagenten om de prestaties en generaliseerbaarheid van zulke agenten te verbeteren. Traditionele benaderingen voor het ontwikkelen van onderhandelingsagenten zijn gebaseerd op handmatig ontworpen heuristieken en strategieën. Hoewel dergelijke methoden effectief zijn in specifieke scenario's, hebben ze te kampen met beperkingen, waaronder menselijke ontwikkelingskosten, slechte generaliseerbaarheid voor verschillende onderhandelingsproblemen en soorten tegenstanders, en menselijke voorkeuren in het ontwerp van strategieën. Het eerste deel van dit proefschrift gaat in op deze beperkingen door methoden voor het leren van onderhandelingsstrategieën te onderzoeken. We behandelen verschillende methoden waarmee autonome agenten effectief onderhandelingsgedrag kunnen leren.

Hoofdstuk 3 presenteert een geautomatiseerde algoritmeconfiguratietechniek, specifiek Sequential Model-based Algorithm Configuration (SMAC) [75], voor de optimalisatie van de parameters van een handmatig gedefinieerde onderhandelingsstrategieruimte. Deze aanpak toont een verbetering aan ten opzichte van handmatig ontworpen strategieën en configuraties die volgen uit literatuur, terwijl nog steeds wordt vertrouwd op een vooraf gedefinieerde onderhandelingsstrategieruimte en handmatig ontworpen instantiekenmerken om het configuratieproces te begeleiden.

Omdat we beseffen dat één enkele geoptimaliseerde strategie waarschijnlijk niet optimaal is voor elke situatie, gaan we in Hoofdstuk 4 verder met het onderzoeken van portfolio-gebaseerde benaderingen. Met behulp van methoden zoals Hydra [162] voor het samenstellen van portefeuilles van strategieën en AutoFolio [100] voor het selecteren van de beste strategie per situatie, wordt automatisch een portefeuille van complementaire onderhandelingsstrategieën gegenereerd en wordt een selector getraind. Hierdoor kan de agent dynamisch de meest geschikte strategie uit zijn portefeuille kiezen op basis van de kenmerken van het huidige onderhandelingssituatie en de tegenstander, wat leidt tot een verbeterde aanpassingsvermogen en algehele prestaties in vergelijking met het gebruik van één enkele beste strategie.

Om de afhankelijkheid van handmatig ontwerp verder te verminderen en door mensen ingebouwde voorkeuren mogelijk nog substantiëler te beperken, onderzoekt Hoofdstuk 5 een benadering met behulp van reinforcement learning. We maken gebruik van Proximal Policy Optimisation (PPO) [135] en Graph Neural Networks (GNN's) [86] om de variabele dimensionaliteit in de observatie- en actieruimtes van diverse onderhandelingssituateis te kunnen hanteren. Hierdoor kan de onderhandelingsstrategie rechtstreeks geleerd worden uit historische interacties, zonder expliciet ontwerp van instantiekenmerken of een handmatig gedefinieerde

onderhandelingsstrategieruimte. Dit maakt het mogelijk om door mensen ingebouwde voorkeuren verder te reduceren en te generaliseren naar onderhandelingsproblemen van verschillende omvang en complexiteit. Er zijn nog wel uitdagingen wat betreft het effectief aanpassen van de strategie ten opzichte van verschillende tegenstanders.

In het tweede deel van dit proefschrift gaan we verder dan de ontwikkeling van lerende agenten door de methoden die worden gebruikt voor hun evaluatie kritisch te bekijken. Hoofdstuk 6 presenteert een uitgebreide empirische analyse, waarbij gebruik wordt gemaakt van gegevens van de Automated Negotiation Agents Competition (ANAC) 2022, die we zelf hebben georganiseerd en waarbij we deelnemers specifiek hebben uitgedaagd om lerende agenten te ontwikkelen. De analyse toont de beperkingen van de huidige evaluatiemethoden aan. Ze laat zien dat de rangschikking van agenten afhankelijk is van de keuze van prestatiecriteria, zoals individuele score, gesommeerde groepsscore of speltheoretische evenwichten. Ze toont ook aan dat de veelgebruikte maatstaf, van gemiddelde individuele score, gevoelig is voor de samenstelling van de groep en niet geschikt is voor het beschrijven van niet-transitieve verhoudingen tussen agenten. De conclusie is dat er momenteel geen enkele, universeel robuuste maatstaf bestaat voor de evaluatie van onderhandelingsagenten, met name voor lerende agenten die niet-stationair gedrag vertonen.

Om tegemoet te komen aan de vastgestelde behoefte aan duidelijke evaluatiecriteria en onderzoeksuitdagingen binnen toepassingen, stelt Hoofdstuk 7 multi-agent kalenderplanning voor als een rijk, relevant en complex toepassingsgebied. Deze taak bevat veel uitdagingen voor de automatische onderhandelingen onderzoeksgemeenschap en kan mogelijk toekomstig onderzoek sturen om vooruitgang te boeken buiten de huidige grenzen van deze gemeenschap.

Samengevat levert dit proefschrift methoden voor het ontwikkelen van onderhandelingsagenten die in staat zijn om te leren en hun strategieën aan te passen, waardoor menselijk ingebouwde voorkeuren worden verminderd en de generaliseerbaarheid wordt vergroot. Het biedt ook een kritische analyse van evaluatiemethodologieën in geautomatiseerde onderhandelingen, waarbij de tekortkomingen ervan worden benadrukt en wordt gepleit voor onderzoek naar evaluatiemethoden, mogelijk met behulp van toepassingsgebieden zoals agendabeheer.

ACKNOWLEDGEMENTS

What a ride it was! Before I started, someone told me that PhD life is not all sunshine and roses and I naively thought my experience would be different. I now understand why people say that, and why many PhD candidates (half-jokingly) dream of becoming baristas, bakers or farmers. These years have taught me a lot about myself, both my strengths and my pitfalls. It was challenging at times, but there were also plenty of rewarding moments and unforgettable memories. And now this milestone has been achieved and I hope there will be many more.

Holger often said that research is a group effort. That always resonated with me, not only because we "stand on the shoulders of giants", but also because so many supervisors, mentors, colleagues, friends, and family are (directly or indirectly) part of the work and deserve to be acknowledged. So here we go.

First and foremost, my supervisors, Catholijn, Holger, and Thomas. Catholijn, your broad knowledge on topics even far beyond computer science was incredibly helpful. Thank you for the personal touch you had in your supervising style, often asking me how I was doing and celebrating my successes, more than I was capable of. Holger, I have always admired your strong scientific morals and your preciseness has made me a better scientist. You also shared your enjoyment of social gatherings with our group, fostering a group bond that made it a joy to be in the office everyday. Thank you for that. Thomas, I much needed someone appreciative of me barging in the door in Leiden. You always took time for me, even before you officially joined the supervision team, which helped me more than you might realise. Thank you for that and for the personal relationship we had.

Matthias, my longest and closest colleague. A PhD can be a lonely job. It makes having a great colleague sitting next to you all the more valuable. I have especially great memories of me and Annelot visiting you in Oxford. All the drinks at the Foo Bar and other bars, the festivals we attended together, the Jiujitsu classes we took, you visiting me in Edinburgh, me visiting you in Berlin. Thank you, as the PhD journey would have been a lot less fun without you.

Annelot, we also spent many days in the office and outside of it. Our week of combining a conference in Vienna with drinking cocktails in the sun on a rooftop bar and skipping a day to go to Prater with our supervisors will forever be a fond memory.

Michiel, my HI buddy at Leiden University. I remember the summer school in Berlin, the first trip after COVID, of which I have more memories of social activities than of lectures, going to Jacob Collier together in Munich, and being roommates on Ameland. Your endless optimism and capability to get a thousand things done is admirable. Thank you for being a great friend and colleague during these years.

Enrico and Mani, also part of the Berlin crew and the food club sequel. Enrico, we are lucky you still have not been found by the pasta police for your kimchi-

Gruyère risotto (to all readers: try it and be amazed). You are one brave Italian with a great sense of humour. Mani, for such a well-versed and thoughtful person, you somehow experience an incredible number of extremely random adventures. I will never forget your stories. Thank you both for all the fun, at work and beyond.

To my fellow members of the Leiden ADA group: Anna Latour, Marie, Koen, Mitra, Jan, Matthias, Annelot, Mike, Laurens, Julia, Andreas, Khashayar, and Can. And the members of RLG: Aske, Thomas, Álvaro, Andreas, Michiel, Andrius, Zhao, Matthias, Tom, Joost, Bernard, Alan, Annie, Felix, Koen, Lindsay. Daily life at the office became so normal, I will miss the chats, the laughter, coffee breaks, faculty bar drinks, and the Bonte Koe evenings. Also to the rest of the great staff at LIACS, thank you for making this the workplace it is, which felt like a tight-knit community. Special thanks also to Aske for showing me the business perspective on academia and for the light-hearted relationship we had.

To everyone in our Hybrid Intelligence research consortium, especially also the PhD students who started alongside me in COVID times and tried to make the best out of it. I feel privileged having been part of a national research consortium resulting in so many connections across universities. I sincerely enjoyed all the gatherings, conferences, and retreats we had, both academically and socially. A special thanks to Roel for being my mentor and providing me with outside perspectives on my PhD. To our sister research group at the RWTH Aachen, thanks for the great workshops we had. I have especially fond memories of the Christmas market evenings we had in Aachen.

Special thanks to my Master students: Thijs, Hadar, and Antonio. I enjoyed working with every one of you and it was great that we could also hang out outside of university. I am happy to see where you all ended up. Also thanks to Stefano and Kobi for allowing me to come to the University of Edinburgh for a research visit. My daily interactions in Edinburgh were mostly with the Autonomous Agents Research Group and the visiting researchers with whom I shared desks. I had an amazing, though way too short, time in Edinburgh, thank you all for that.

It is hard to know where to stop mentioning names, as there are so many people who have contributed to this journey, and I hope I have not overlooked anyone. If I did, please know that this gratitude is meant for you as well. One of the aspects of academia that was dear to me was the opportunity to travel, sometimes halfway across the globe, and walk into a conference or gathering to immediately recognise faces. People you might only see for a week each year, yet you intensively spend a week together with, explore a new place together, and then part ways again. That sense of slowly building and belonging to a global community, of feeling strangely at home while being far from it, was truly unique. To all those I connected with over the years, thank you for creating that feeling of community.

To my friends outside of the academic bubble: Thijs, Thijs, Jeroen, Maarten, Roland, Bart, Sander, Sebastiaan (and partners). Nothing beats spending a week in the Alps when it comes to resetting yourself. Our summer hut tours and winter ski trips were (and are) a welcome distraction from the PhD work and something I always looked forward to. Academics are often unable to talk about anything else than work, so having a group to casually hang out with is a blessing. This

gratitude also extends to my Club crew: Joost, Matthijs, Ron, Dave, Sebastiaan, Jeroen, Tijmen, and Suzanne, my Milan group: Amanda, Laura, and Jesse, and my Embedded buddies, Lourens, Jeroen, Vito, and Thijmen. Thank you all for being an awesome bunch to hang out with.

To my parents, Aaldert and Rian. You never really think about growing up. And then you turn 30 and you start to reflect, and you realise that your skills and curiosities were nurtured in your youth. I deeply appreciate my appreciation of many things in life, far beyond computer science. This dissertation is a result of how I was raised. You also taught me by example to put society's needs before one's own, sometimes a bit too much, but the world is a better place with people like you. To my sister and brother, Anneke and Kees. Thank you for riding the roller coaster of growing up with me and to Arnoud and Danielle for joining the ride. It is great to have people who have your back unconditionally. An imperturbability that is underestimated and certainly not a given. Special thanks to Kees for letting me abuse your house as a writing retreat.

And finally, to my partner, Suzanne. A decade together, of which nearly half was spent with me working on the PhD. You have seen it all, start to finish. It is difficult to keep work at the office when you do a PhD due to the heavy variable workload. I think we both realise that this did not always make me the most fun person. And yet you were always supportive when I needed it, enthusiastic when I decided to spend months abroad, and patient when I once again wanted to do too many things concurrently. You are an exceptionally kind person and I am lucky to have you. A mere "thank you" does not capture it, this one is for you. Ik hou van je.

CONTENTS

1

INTRODUCTION

Ants and bees can also work together in huge numbers, but they do so in a very rigid manner and only with close relatives. Wolves and chimpanzees cooperate far more flexibly than ants, but they can do so only with small numbers of other individuals that they know intimately. Sapiens can cooperate in extremely flexible ways with countless numbers of strangers. That's why Sapiens rule the world, whereas ants eat our leftovers and chimps are locked up in zoos and research laboratories.

Yuval Noah Harari, *Sapiens* (2015)

1

We rule the world. In other words, the human species is the most dominant species on earth (according to us humans). Yuval Harari [65] stated that this could be attributed to our unique ability to cooperate both flexibly and with very large numbers. Crucial to our cooperative behaviour is our ability to communicate and, as a special case of communication, negotiate [36, 105]. Negotiation enables humans to resolve conflicts over resources, improving collective productivity through task specialisation. For instance, a hunter can exchange surplus food for a blacksmith's traps, allowing each to focus on their expertise. Now, the hunter can keep hunting, and the smith can keep smithing instead of both having to do both. This mutually beneficial agreement, facilitated by negotiation, improves the overall efficiency and productivity of a community. Humans are not the only species that negotiate, as studies have shown that, e.g., chimpanzees also possess this ability [105]. However, human communication and negotiation are far more complex and likely play a crucial role in our dominant position.

Negotiation is an important part of human interaction in modern society, from interpersonal relationships to international diplomacy. It serves as a critical mechanism for resolving conflicts, allocating resources, and enabling cooperation between parties with divergent interests. Negotiations are so embedded into our daily lives that we do not always notice them [117]. Discussing the details of a contract is clearly a negotiation, but other negotiations are not that explicit. For example, finding a meeting slot with colleagues, deciding where to go on a city trip with a friend, or navigating busy bike lanes with non-verbal signals can all be considered forms of negotiation.

Negotiation has seen interest from the scientific community, for example, by social scientists [131, 149], economists [118, 115] and by mathmaticians [111, 132]. With the advent of artificial intelligence, it is envisioned that artificial intelligent agents also have conflicts of interest, which should be studied. This dissertation, therefore, studies negotiation in computer science, more specifically, in multi-agent systems.

1.1 NEGOTIATION IN MULTI-AGENT SYSTEMS

As artificial intelligence becomes more embedded into society and daily life, the question arises of how autonomous or semi-autonomous agents interact. This is a central focus of Multi-Agent Systems (MAS), a broad research area within AI concerned with "systems that include multiple autonomous entities with either diverging information or diverging interests, or both" [140]. MAS research encompasses a range of research areas, such as multi-agent learning [151], communication and consensus [121], organisational structures [72], and distributed constraint satisfaction [164].

In multi-agent interactions, situations arise where agents, whether AI-AI or AI-human, must cooperate despite (partially) conflicting interests. Such conflicts of interest should then be resolved to improve payoff or even obtain payoff in the first place (e.g., surveying drone swarms or transporting goods using multiple robots). Resolving the conflicting interests can be done in several ways. In some cases, conflicts are handled through adherence to pre-defined institutions or social

norms (e.g., traffic rules governing right-of-way). In others, predefined coordination mechanisms or protocols suffice for enabling coordination.

If there is no predefined method of handling conflicts of interest, then negotiation or bargaining can be used to enable cooperation. As it was important for humans to develop negotiation skills, AI researchers envisioned that this is also an important skill for intelligent agents to resolve conflicting interests. This led to the research field of automated negotiation [79]. Automated negotiation plays a role in real-world applications, such as traffic light coordination [62], calendar scheduling [125], or balancing energy demand and production in local power grids, and also in games, such as Diplomacy [107] and Werewolves.

Automated negotiation is a multidisciplinary area at the intersection of artificial intelligence, game theory, and decision science. Early work in this field, pioneered by scholars such as Smith [143], Rosenschein [130], Klein and Lu [88], and Sycara [147], laid the groundwork for computational approaches to negotiation. The field saw significant advancements with the development of more sophisticated negotiation strategies and frameworks. Notably, the work of Faratin et al. [48] introduced time-dependent and behaviour-dependent tactics, which have become fundamental components of many negotiation strategies.

Over the years, the research community has developed a wide array of negotiation strategies, protocols, and evaluation frameworks, leading up to initiatives like the Automated Negotiating Agents Competition (ANAC) [16] and the General Environment for Negotiation with Intelligent multi-purpose Usage Simulation (GENIUS) negotiation platform [99]. The combined effort of GENIUS and ANAC provided a standardised test bed with more than 100 negotiating agents and negotiation scenarios that are readily accessible for research on automated negotiation [9]. Both the development of new agents and structured evaluation of these agents are important, which we will discuss in Sections 1.1.1 and 1.1.2, respectively.

1.1.1 DEVELOPING NEGOTIATION AGENTS

The goal of automated negotiation research is the development of agents that are capable of negotiating quickly and effectively. The negotiating agents are generally hard-coded strategy algorithms with parameters that are tuned during design to optimise their performance. Such agents must navigate large outcome spaces, deal with incomplete information, and engage in strategic interactions with other parties, all while attempting to achieve optimal results for themselves or the humans or organisations they represent. Traditionally, developing negotiation agents relied on manually designed heuristics and strategies, optimised and tested on constrained sets of negotiation settings, to make it manageable. This is still a commonly seen approach in recent editions of ANAC [5]. While such methods have proven effective in specific scenarios, they struggle to generalise across diverse negotiation scenarios and opponent types. This limitation becomes problematic as we envision deploying automated negotiation agents in real-world applications, where they may encounter a wide variety of negotiation problems and counterparts.

To alleviate the difficulty in designing negotiation agents manually and to obtain better performance, optimisation methods were later used to (partially) automate

the design. This allows agents to be reconfigured in various negotiation settings, improving generalisability. Genetic algorithms have been used, but optimising negotiation agents is computationally expensive, due to, e.g., searching for potential agreements in outcome spaces exponential in the objectives for which a decision must be made. To circumvent the computational complexity issue, agents were configured on small sets of scenarios and opponent types [104]. For instance, agents were only tested in one or two scenarios [71] or merely optimised against themselves [46, 43]. The resulting agents are highly specialised and have unpredictable performance when negotiating "out-of-distribution".

The problem we address is the development of agents capable of learning to negotiate effectively across large and diverse sets of negotiation settings. Learning to negotiate involves agents optimising policies for actions, such as making proposals, accepting offers, or walking away, to maximise their performance. We adopt a learning-based approach because hand-designed negotiation agents impose human-induced bias, leading to suboptimal performance, limitations we want to overcome. Furthermore, as traditional hand-crafted negotiation state representations used for agent input can discard informational value, we also aim to learn representations directly from (near-) raw observational data.

Part one of this dissertation aims to address the problem of learning autonomous agents to negotiate on large and diverse sets of opponent types and negotiation scenarios. We explore approaches to learn negotiation strategies more efficiently (Chapter 3). We also explore strategy-switching mechanisms that are learned based on characteristics of a negotiation setting in an effort to further improve the adaptiveness of negotiation strategies (Chapter 4). Finally, we leverage machine learning, more specifically reinforcement learning, to push the boundaries of what is possible in automated negotiation (Chapter 5). We discuss all three approaches briefly below.

ALGORITHM CONFIGURATION

Chapter 3 focuses on enhancing existing negotiation strategies through automated algorithm configuration. Many negotiation agents in the literature use strategies with fixed parameters or ones manually tuned by their designers. This approach, while straightforward, often leads to suboptimal performance and poor generalisation across different negotiation settings.

We propose to address this limitation by applying general-purpose automated algorithm configuration techniques, specifically Sequential Model-based Algorithm Configuration (SMAC) [75], to the task of optimising negotiation strategies. SMAC is a generally well-performing algorithm that is often used for such tasks. Our approach automatically configures the parameters of commonly used negotiation strategy components, including those related to bidding strategies, accepting strategies, and opponent modelling. Our approach employs a set of features that characterise negotiation scenarios and opponent behaviours, enabling more efficient configuration across diverse negotiation settings. This enhanced efficiency enables us to optimise performance on larger and more diverse sets of negotiation simulations than previously achievable in automated negotiation systems. We have

been the first to pursue this approach, which goes beyond simple application of algorithm configuration.

ALGORITHM SELECTION

Building on the insights gained from automated configuration, we next explore the potential of portfolio-based methods in automated negotiation in Chapter 4. This direction is motivated by the observation that there is often no single best strategy for all negotiation settings — different strategies may be more or less effective, depending on the specific negotiation scenario and opponent type[99].

Our portfolio construction method, based on Hydra [162], iteratively generates complementary strategies that specialise on sub-spaces of the negotiation game distribution. We build on top of the previous work where we configure a single-best strategy and, instead, now configure a portfolio of complementary strategies. We then employ a learned strategy selection model, using AutoFolio [100], that selects a strategy from the portfolio for a given negotiation game. This portfolio-based approach offers the advantage of better adaptation to different opponents and negotiation problems and outperforms the single-best strategy across diverse settings.

REINFORCEMENT LEARNING

In Chapter 5, we investigate the potential of end-to-end reinforcement learning approaches for automated negotiation. The previous algorithm configuration and selection approaches still rely on a high degree of manual design, as the parameterised strategies are a prerequisite for both approaches. This introduces bias, which can help in reducing the search space of negotiation strategies. However, it also potentially introduces suboptimality to the method. To further mitigate potential biases introduced by human design, we propose a conceptually straightforward reinforcement learning approach, based on Proximal Policy Optimisation (PPO) [135], in an effort to obtain better performance across a broader set of negotiation settings.

A key challenge in applying reinforcement learning to negotiation is handling the variability in problem structures. Negotiation outcome spaces are arbitrarily sized, leading to widely varying observation and action spaces. This variability poses difficulties for traditional neural network architectures used in reinforcement learning. To address this challenge, we developed a policy network using Graph Neural Networks (GNNs) [26] that operates on a custom graph-structured representation of negotiation game data and trained it through reinforcement learning. Our policy can process negotiation scenarios of varying sizes and structures, enabling generalisation across diverse scenarios. This approach opens up avenues for more sophisticated and adaptable negotiation agents in automated negotiation by leveraging reinforcement learning techniques. It significantly reduces human-induced biases in learned strategies while facilitating the integration of advanced learning capabilities into policy networks.

1.1.2 EVALUATING NEGOTIATION AGENTS

An essential part of developing negotiation agents is evaluating them empirically. Negotiation agents are tested by running computational experiments where agents

compete against each other on sets of negotiation scenarios. Without a good performance measuring method, it is difficult to judge whether a designed strategy is effective or not. Over the years, multiple such evaluation methods were developed and tested on selected strategies [9]. Despite the efforts, there is still no clear answer to the question of what makes a negotiation agent good or not. To make matters more complex, we see an increase in the deployment of machine learning techniques in recently developed agents, which makes them change their behaviour over time. This inherent non-stationarity poses a challenge to empirical evaluation, as evaluation methods often require repeated trials under stable conditions. Evaluating learning agents via repetition yields non-stationary performance metrics, complicating such an evaluation. In part two of this dissertation, we study this in more depth.

EMPIRICAL ANALYSIS OF NEGOTIATION AGENTS

We organised ANAC 2021 and 2022, and set the challenge explicitly on the design of negotiation agents that learn and adapt their behaviour over time. The agents submitted to the 2022 competition are used for an extensive evaluation study of negotiation agents comparing various evaluation criteria in Chapter 6. We observe that multiple factors influence the performance of an agent. The evaluation criteria have an influence, but so does the composition of the group of opponents. We conclude that there is no single evaluation criterion to evaluate negotiation agents, and that outperforming a fixed group of opponents is no guarantee that an agent will perform well in general. However, a common practice in ANAC is to evaluate agents based on a single evaluation criterion in a fixed group of agents.

PROPOSED APPLICATION DOMAIN

We argue that the lack of a good performance measuring method goes hand in hand with the lack of a straightforward application domain. Different application domains likely require different performance criteria. Without a clear application domain, the community could be running in circles in pursuit of better strategies. To this extent, in Chapter 7, we propose an application domain for negotiation agents that we think is rich and useful. We propose the use of negotiation agents in calendar scheduling problems and map out the necessary steps to achieve this. We hope this sets an example and a direction for the automated negotiation community.

1.2 RESEARCH QUESTIONS

We formulate two main research questions and several sub-questions for this dissertation. The questions are in line with what was discussed in Section 1.1.1 and Section 1.1.2:

Q1 How do we design agents that can learn to negotiate?

- **SQ1.1** How can we reduce human-induced biases and conceptual complexity in learned negotiation strategies?
- **SQ1.2** Can we learn to generalise over diverse negotiation instances?

Q2 Is there a uniform way of evaluating negotiation agents?

- **SQ2.1** Is there a single-best metric?
- **SQ2.2** What is the value of the average utility metric for evaluating negotiation agents?

Investigating these questions will advance machine learning within automated negotiation, leading to more sophisticated negotiation agents while reducing design overhead. Furthermore, improved insight into evaluation metrics is essential for refining research goals and assessment methodologies. Progress in automated negotiation is important to stimulate the wider adoption of agent-based AI systems.

1.3 DISSERTATION STRUCTURE

The remainder of this dissertation is structured as follows. Chapter 2 provides background information on commonly used concepts and topics covered in this dissertation. Chapter 3 presents our work on automated configuration of negotiation strategies. We detail our application of SMAC to strategy optimisation, which is not straightforward due to the combined nature of negotiation problems consisting of opponents and scenarios. We introduce our feature representation for negotiation problems and provide empirical results demonstrating the effectiveness of this approach. Chapter 4 describes our portfolio-based approach for strategy selection and adaptation. We explain our portfolio construction method, detail the AutoFolio-based strategy selection process, and present results from ANAC-like tournaments, showing the advantages of this approach. Chapter 5 introduces our end-to-end reinforcement learning method for general negotiation. We describe our graph-based policy architecture, explain the training process using PPO, and provide extensive evaluations against both baseline and state-of-the-art negotiation agents. Chapter 6 contains our extensive analysis of negotiation agents that were submitted to ANAC 2022. The 2022 edition focused specifically on learning negotiation agents. We compare various performance criteria and study the effect of non-stationary behaviour. We argue that evaluating negotiation agents is challenging and that there is no single best method to do so. Chapter 7 resurfaces a realistic real-world application domain for negotiation agents that has been studied in the past, namely, agent based calendar scheduling. We thoroughly dissect the complexity of the domain into manageable sub-problems to solve in the hope of setting the stage for future work in this application domain. Chapter 8 concludes the dissertation with a discussion of the broader implications of our work, its limitations, and promising

directions for future research.

2

Background

This chapter contains background knowledge for readers who are less familiar with some of the topics discussed in this dissertation. We start with an introduction to automated negotiation (Section 2.1) and further discuss algorithm configuration (Section 2.2), algorithm selection (Section 2.3), and reinforcement learning (Section 2.4).

2

2.1 AUTOMATED NEGOTIATION

The research topic of automated negotiation studies autonomous agents that participate in a negotiation game [50]. Here, negotiation can be seen as a distributed search through a space of potential outcomes [79]. The software agents used in this context can negotiate with other agents in multi-agent systems or with humans in human-agent negotiations. Automated negotiation combines concepts and insights from multiple disciplines, such as artificial intelligence, game theory, and social psychology [79].

In this dissertation, we focus exclusively on bilateral negotiation between two autonomous agents. We define a negotiation game as a tuple $N = \langle \mathcal{I}, \Omega, \mathcal{U}_\mathcal{I} \rangle$, where $\mathcal{I}$ denotes the set of agents, Ω denotes the set of potential outcomes the agents negotiate over, also known as the domain or outcome space, and $\mathcal{U}_\mathcal{I}$ is the set of utility functions for agents $\mathcal{I}$ such that $\mathcal{U}_\mathcal{I} = \{u_i | i \in \mathcal{I}\}$. $u_i : \Omega \rightarrow [0,1]$ is the utility function of agent i over the outcome space where higher utilities indicate more desirable outcomes. From the perspective of an agent, we often refer to its own utility function with u and to the opponent's utility function with u_o. How agents communicate to agree is specified through a negotiation protocol that dictates the rules of encounter and permissible actions.

2.1.1 NEGOTIATION PROTOCOL

Communication between agents is required to enable a negotiation. Humans primarily negotiate using natural language. This is also possible for negotiation agents, but adds additional challenges in generating and interpreting natural language [128, 95, 67, 90, 98]. Most of the work in automated negotiation uses a negotiation protocol that dictates what actions agents can perform and when. A few examples of protocols are:

- The Ultimatum Game [24]: One agent makes an offer and the other agent may only accept or refuse it, after which the negotiation is ended.
- Alternating Offers Protocol (AOP) [132]: Two agents take turns in making offers. An extended version of this protocol that supports more than two agents is the Stacked Alternating Offers Protocol (SAOP) [7].
- Monotonic Concession Protocol [129]: Requires agents to make progressively more attractive offers to their opponents.
- Multiple Offers Protocol for multilateral negotiations with Partial Consensus (MOPaC) [110]: Allows for multilateral negotiations where only a subset of negotiating agents can reach an agreement, enabling partial consensus through bidding, voting, and opt-in phases.

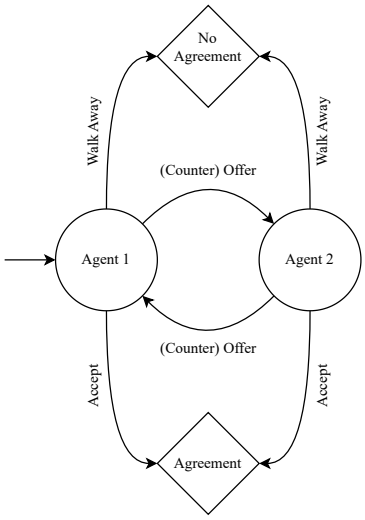


Figure 2.1: Visualisation of the Alternating Offers Protocol (AOP). Agent 1 starts the negotiation by making an offer. Note that the action "Accept" is inaccessible in the first step.

- Simulated Annealing Protocol [87]: A negotiation approach where agents or a mediator use simulated annealing techniques to explore the outcome space, allowing temporary concessions to escape local optima and ultimately achieve near-optimal agreements.

The choice of protocol can significantly impact negotiation dynamics and outcomes. For instance, the Alternating Offers Protocol (AOP) allows for more complex strategies and learning during negotiation, while one-shot protocols may encourage more truthful revelation of preferences.

In this dissertation, we exclusively use the AOP, due to its simplicity and its wide use in the automated negotiation literature. Here, agents take turns making a (counter) offer, accepting the opponent's offer, or walking away from the negotiation. The protocol is visualised in Figure 2.1. A (counter) offer is made by selecting one of the potential outcomes $\boldsymbol{\omega} \in \Omega$ and sending it to the opponent. A deadline T is imposed on the negotiation to prevent agents from negotiating indefinitely. Such a deadline is defined either in a maximum number of rounds that the agents can negotiate for or in seconds of wall-clock time or a combination of the two. The wall-clock time limit is often used as searching large (often combinatorial) outcome spaces for suitable candidate offers can take a long time.

The negotiation ends with an agreement if one of the agents accepts the offer received from the opponent. Both agents then receive a pay-off according to their (discounted) utility function. The negotiation ends without an agreement if one of the agents decides to walk away or if the deadline passes. In that case, agents are awarded their reservation value or 0 if there is no reservation value. The reservation value can be used to represent a Best Alternative To a Negotiation Agreement (BATNA) [53], e.g. an agreement reached with another negotiation partner, or

deciding to build something yourself instead of negotiating with a contractor.

2.1.2 NEGOTIATION SCENARIO

A negotiation scenario is a combination of an outcome space and utility functions that reflect the preferences of the agents. The outcome space generally consists of a set of objectives, known as issues $B = \{1, \cdots, n\}$, that are being negotiated. For example, in a GPU compute node purchase negotiation, issues might include price, support period, GPU type, number of GPUs, and delivery date. Issues can be:

- Categorical (e.g., GPU type: A6000, A100, or H100)
- Continuous (e.g., price of the compute node)
- Integer (e.g., number of GPUs: 2, 4, or 8)

We focus solely on categorical issues in this dissertation, which is the most general type, as continuous issues can also be discretised. The set of possible values for an issue $b \in B$ is denoted as Ω_b. The Cartesian product of all the issue values comprises the total outcome space $\Omega = \Omega_1 \times \cdots \times \Omega_n$ of the negotiation scenario. During a negotiation, the agents attempt to agree upon an outcome $\boldsymbol{\omega} = (\omega_1, \omega_2, \cdots, \omega_n) \in \Omega$, where n is the number of issues and $\omega_b \in \Omega_b$. Simple scenarios might have 2–3 issues with a small number of categorical values each. Complex scenarios can have dozens of issues, continuous values, and large numbers of categorical issue values, increasing the computational complexity of searching for suitable outcomes in the outcome space.

UTILITY FUNCTION

Utility functions are mathematical representations of an agent's preferences over the outcome space. They map potential outcomes to a real number, indicating the relative desirability of different outcomes. Both agents have a utility function $u_i : \Omega \to [0, 1]$, where 1 is the best possible utility for an agent. Such utility functions can be complex non-linear functions that capture interdependencies between issues, but in this dissertation, we focus on linear additive utility functions like the one shown in Equation 2.1. Here, weights are assigned to all values and issues through weight functions $w : B \mapsto [0, 1]$ and $w_b : \Omega_b \mapsto [0, 1]$ such that $\sum_{b \in B} w(b) = 1$, $\max_{\omega_b \in \Omega_b} w_b(\omega_b) = 1$, and $\min_{\omega_b \in \Omega_b} w_b(\omega_b) = 0$.

$$u(\omega) = \sum_{b \in B} w(b) \cdot w_b(\omega_b) \tag{2.1}$$

RESERVATION VALUE & DISCOUNT FACTOR

A negotiation scenario can optionally be extended with a reservation value and a discount factor. A reservation value is specified per agent and is the utility that an agent obtains when they fail to reach an agreement. This can be used to simulate the Best Alternative to a Negotiated Agreement (BATNA) [53], which is a commonly used concept in negotiation literature. This can be used to, for example, simulate an alternative agreement that is achieved with another party.

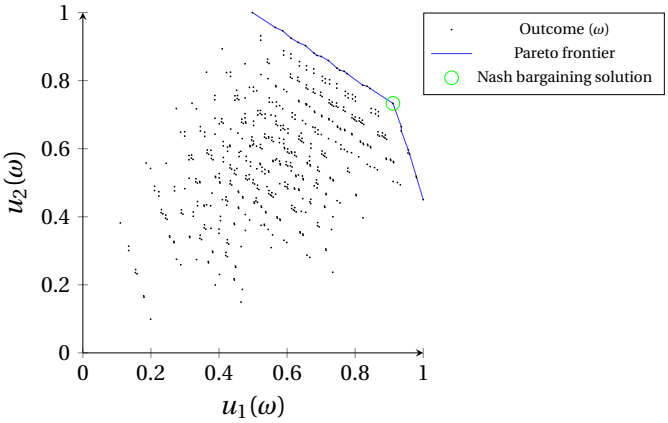


Figure 2.2: Visualisation of example negotiation scenario. The dots represent outcomes, which are plotted according to their utility value for two agents. The blue line connects the Pareto efficient possible outcomes, which form the Pareto frontier. The Nash Bargaining solution, the possible outcome that maximises the product of utilities, is circled in green.

The discount factor can be used to apply time pressure in a negotiation game. Here, the obtained utility is discounted as follows:

$$u_i^{\gamma}(\boldsymbol{\omega}) = (\gamma)^t \cdot u_i(\boldsymbol{\omega}) \tag{2.2}$$

where u_i is given in Equation 2.1, $\gamma \in [0,1]$ is the discount factor and t is a discrete time step variable. We do not consider reservation values and discount factors as the increased complexity they cause is outside of the scope of this dissertation.

EXAMPLE SCENARIO

To provide some intuition, we visualise an example negotiation scenario with categorical issues in Figure 2.2. All possible outcomes of the scenario are plotted based on their utility value for agent 1 (x-axis) and agent 2 (y-axis). We also visualise the Pareto frontier, which is the set of possible outcomes that are not strictly dominated by another possible outcome based on its utility value for both agents. Finally, we visualise the Nash bargaining solution [111], which is the outcome that maximises the product of utilities of all involved agents and is often used as a reference outcome for performance analysis. The Nash bargaining solution is defined as:

$$\omega_{Nash} \in \arg\max_{\omega \in \Omega} (u_1(\omega) \cdot u_2(\omega)) \tag{2.3}$$

2.1.3 NEGOTIATION AGENTS

Negotiation strategies determine an agent's behaviour during the negotiation process within the used protocol. They encompass decision-making about which offers to make, when to accept offers, and how to model the opponent's behaviour and respond to it. Strategies are at the core of automated negotiation research, with a

wide variety of approaches developed over the years. A commonly used framework to structure components of negotiation strategies is the Bidding, Opponent modelling, and Accepting (BOA) framework [8]. Although we will not always use this framework in this dissertation, we will use it in this section to provide insights into negotiation strategies.

BIDDING STRATEGY

The bidding strategy decides which offers to make. This involves balancing between pursuing the interests of the agent (e.g. maximising utility) and making concessions to reach an agreement. Common approaches include:

- Time-dependent tactics (see, e.g. [48], [51]): Concession is based on the remaining time.
  - Boulware: Concedes slowly at first, then rapidly as the deadline approaches.
  - Conceder: Makes large concessions early.
  - Linear: Concedes at a constant rate.
- Behavior-dependent tactics (see, e.g. [4]): Adapt based on the opponent's actions. These might mimic, reciprocate, or exploit the opponent's concession pattern.
- Trade-off strategies: Attempt to generate offers of similar utility to the previous offer but more attractive to the opponent.

ACCEPTANCE STRATEGY

The acceptance strategy determines when to accept an offer. Common criteria include:

- Threshold-based: Accept if the offer's utility exceeds a certain threshold.
- Time-dependent: Lower the acceptance threshold as the deadline approaches.
- Aspiration-based: Accept if the offer exceeds the utility of the planned counter-offer.

Acceptance strategies are often combinations of these criteria or more complex designed heuristic-based methods. For a comparison of acceptance strategies, we refer the reader to Baarslag and Hindriks [17].

OPPONENT MODELLING

With opponent modelling, the agent attempts to learn about the opponent's preferences, which are commonly considered to be private information. Being able to accurately predict the opponent's utility function helps in finding mutually beneficial outcomes, i.e., find outcomes that are (close to) Pareto efficient, which generally improves the pay-off of the agent. Techniques to predict the opponent's utility function include:

- Frequency analysis: Infer the importance of outcomes based on how often they are offered.
- Regression: Assume the concession behaviour (e.g., Boulware) of the opponent and fit a regression model to the opponent's offers based on that concession behaviour.
- Bayesian learning: Fit a probability model to the observed offers of the opponent.

We refer the reader to Baarslag et al. [12] for a more elaborate review and comparison of opponent modelling techniques.

2.2 ALGORITHM CONFIGURATION

Negotiation agents commonly have parameters that influence the behaviour or strategy of the agent. Algorithm configuration, also known as parameter tuning or hyperparameter optimisation, is the task of selecting the best parameters for an algorithm to optimise its performance on a given set of problem instances. The need for algorithm configuration arose from the observation that many algorithms, particularly in areas such as optimisation and machine learning, have parameters that significantly affect their performance. Traditionally, experts set these parameters manually through a process of trial and error. However, as algorithms became more complex and included more parameters, manual tuning became increasingly time-consuming and often suboptimal [74]. Early approaches to configure parameters included simple techniques like grid search and random search [52]. However, these methods scale poorly with the number of parameters, also known as the curse of dimensionality [22]. This led to the development of more sophisticated, automated algorithm configuration approaches, of which we will mention a few.

2.2.1 PROBLEM DEFINITION

The algorithm configuration problem can be formally defined as follows [74]. Given an algorithm with a configuration space Θ, a set of problem instances $\mathcal{P}$, and a performance metric $m(\theta, p)$ for $\theta \in \Theta$ and $p \in \mathcal{P}$, find $\theta^* \in \Theta$ that maximises (or minimises depending on m):

$$\theta^* \in \arg\max_{\theta \in \Theta} \sum_{p \in \mathcal{P}} m(\theta, p) \tag{2.4}$$

CONFIGURATION SPACE

The configuration space Θ represents all possible parameter settings for the algorithm. It can include various types of parameters, such as;

- Categorical parameters (e.g., choice of heuristic).
- Continuous parameters (e.g., learning rate in neural networks).
- Integer parameters (e.g., population size in genetic algorithms).

The configuration space can be highly complex, with dependencies between parameters and varying impacts on algorithm performance.

PERFORMANCE METRICS

The choice of performance metric $m(\theta, p)$ is crucial and depends on the specific application. Common metrics include;

- Solution quality: For example, the distance travelled for optimisation problems like the travelling salesperson problems or the prediction accuracy for machine learning methods on classification or regression tasks.
- Running time: Time taken to solve a problem or reach a certain quality threshold.
- Composite measures: e.g., PAR (Penalised Average Running time), which assigns a penalty to unsuccessful runs in, e.g., Boolean satisfiability or travelling salesperson problems.

2.2.2 CONFIGURATION METHODS

Several methods have been developed for automated algorithm configuration. Two separate parts within these methods can be identified: how new configurations are selected for evaluation and how a set of configurations is compared. In the following, we briefly outline examples of prominent, high-performance configuration approaches:

ParamILS uses Iterated Local Search (ILS) to find high-performance parameter settings for a given target algorithm. It navigates the configuration space by iteratively applying a local search phase, typically involving single parameter changes to find improvements and perturbation steps to escape local optima. Key features include always accepting better-performing configurations and using random restarts for diversification.

- ParamILS [74] uses Iterated Local Search (ILS) to find high-performance parameter settings for a given target algorithm. It navigates the configuration space by iteratively applying a local search phase, typically involving single parameter changes to find improvements and perturbation steps to escape local optima. Key features include always accepting better-performing configurations and using random restarts for diversification.
- F-Race [25] is a statistical racing algorithm that eliminates suboptimal configurations by first using the Friedman test to detect significant overall performance differences among remaining candidates across multiple instances. If such differences exist, it then performs pairwise comparisons against the best-performing configuration (incumbent) to discard those that are statistically significantly worse. This saves computational budget, as not all configurations have to be tested on the full target instance set. The set of configurations to test can be selected either manually, as a grid search, or at random. Balaprakash et al. [20] extended upon F-Race by implementing

it as a model-based search [166], which iteratively models and samples the configuration space in search of promising candidate configurations.

- Sequential Model-based optimisation for general Algorithm Configuration (SMAC) [75] builds a random forest model to predict algorithm performance based on parameter settings and instance features. It uses this model to find promising configurations to evaluate. It also incorporates an early elimination mechanism for new configurations by comparing them with a dominant incumbent configuration on individual problem instances.

- Gender-based Genetic Algorithm (GGA) [3] uses a genetic algorithm approach. It maintains a population of configurations, evolves them over generations, and incorporates gender separation to maintain diversity.

- Bayesian Optimisation HyperBand (BOHB) [47] is designed for hyperparameter optimisation of machine learning algorithms. It combines Bayesian optimisation with hyperband [96], where Bayesian optimisation is used to find promising configurations and multi-armed bandit strategies are used to allocate resources to evaluate configurations.

2.2.3 INSTANCE FEATURES

Some model-based configuration methods (e.g., SMAC and BOHB) allow the use of problem instance features to guide the search process. These features aim to capture relevant properties of the instances to better model the relation between parameter settings, problem instances, and performance. Examples of such features are:

- SAT solving: Number of variables, clause-to-variable ratio, etc.

- Classification tasks: Dataset size, number of features, class distribution, etc.

- Optimisation tasks: Dimensionality, constraint density, objective function properties, etc.

- Negotiation games: Size of the outcome space, average utility of outcomes, last obtained utility against opponent, etc.

2.3 ALGORITHM SELECTION

It is a common observation that a single negotiation strategy does not necessarily work well in every negotiation game [99]. This raises the question of whether selecting between negotiation strategies depending on the negotiation game could improve performance. Algorithm selection is the problem of choosing the best algorithm from a set of candidates for a given problem instance. It is based on the idea that no single algorithm is superior for all problem instances and aims to exploit the complementary strengths of different algorithms. The algorithm selection problem was first formally described by Rice [127], who recognised that for many computational problems, different algorithms performed best on various

instances and proposed a framework for selecting the best algorithm based on features of the problem instance.

The research area gained significant traction with the advent of portfolio-based approaches in areas such as Boolean satisfiability (SAT) solving and Constraint Programming (CP). Notable early systems include SATzilla [161] for SAT solving and CPHydra [113] for CP.

2.3.1 PROBLEM DEFINITION

The per-instance algorithm selection problem can be formally defined as follows. Given a set of algorithms $\Psi = \{\psi_1, \psi_2, \cdots, \psi_n\}$, a set of problem instances $\mathcal{P}$, a performance metric $m(\psi, p)$ for $\psi \in \Psi$ and $p \in \mathcal{P}$, and an instance feature mapping $f : \mathcal{P} \mapsto \mathbb{R}^k$ (Section 2.2.3) that characterise each problem instance p, find a selection mapping $M : f(\mathcal{P}) \mapsto \Psi$ that optimises the performance metric on the set of problem instances:

$$\min_{M} \sum_{p \in \mathcal{P}} m(M(f(p)), p) \tag{2.5}$$

2.4 REINFORCEMENT LEARNING

The negotiation game defined in Section 2.1 can be formulated as a Markov Decision Problem (MDP). We discuss the similarities in Section 2.4.2. This enables the usage of Reinforcement Learning (RL) in negotiation games. RL is a form of machine learning where an agent learns by interacting with an environment. Unlike supervised learning, where the agent learns from labelled examples, or unsupervised learning, where the agent finds patterns in unlabeled data, RL agents learn from the consequences of their actions.

The research area gained significant momentum with the development of algorithms like Q-learning [157] and the publication of a book by Sutton and Barto [146]. More recently, the combination of RL with deep learning, known as deep reinforcement learning, has led to breakthroughs in areas such as game playing (e.g., AlphaGo [141]) and robotics [89].

2.4.1 MARKOV DECISION PROCESS

RL problems are typically formalised as Markov Decision Processes (MDPs). An MDP is defined as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ the set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto p(\mathcal{S})$ denotes the transition function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ the reward function. The transition function describes how the environment transitions to a new state based on the current state the environment is in and the action taken. The reward function describes the reward that the agent obtains by taking an action in the state that the environment is in.

The goal in an MDP is to find the optimal policy $\pi^* : \mathcal{S} \mapsto \mathcal{A}$ that maximises the expected cumulative discounted reward:

$$\mathbb{E}_{\pi, \mathcal{T}} \left[\sum_{k=0}^{H-1} \gamma^t \cdot \mathcal{R}(s_t, a_t) \right] \tag{2.6}$$

Table 2.1: Similarities between a negotiation game and a Markov Decision Process (MDP). Each row describes a similar concept.

Negotiation game		**MDP**	
Utility	u	Reward	$\mathcal{R}$
Outcomes	Ω	Actions	$\mathcal{A}$
Discount	γ	Discount	γ
Deadline	T	Horizon	H
Time step	t	Time step	t

where H denotes the horizon of the MDP (the number of rounds we select an action), $\gamma \in [0,1]$ is a discount factor that discounts future reward, and t is the time step.

2.4.2 MDPS AND AUTOMATED NEGOTIATION

The negotiation game, as defined in Section 2.1, shows similarities with the definition of an MDP. We provide an overview of similarities in Table 2.1. Although unifying the notation is possible, we intentionally kept them separate to maintain the relation in this dissertation to the specific literature of the automated negotiation and reinforcement learning communities. We mention this relation here to make the reader aware of the close resemblance between both research areas.

I

Learning to Negotiate

3

AUTOMATED CONFIGURATION OF NEGOTIATION STRATEGIES

This chapter has been published at AAMAS 2020 [124].

3.1 INTRODUCTION

In this chapter, we use algorithm configuration techniques to configure negotiating agents on large and diverse sets of negotiation scenarios and opponent types. We recreate a negotiation agent from literature [92] that is configured manually, combine it with contemporary opponent learning techniques and create a configuration space of its strategic behaviour. To automatically configure this conceptually rich and highly parametric design, we use Sequential Model-based optimization for general Algorithm Configuration (SMAC), a general-purpose automated algorithm configuration procedure that has been used previously to optimize the performance of cutting-edge solvers for Boolean Satisfiability (SAT), Mixed Integer Programming (MIP) and other NP-hard problems. We note that here, we apply automated algorithm configuration for the first time to a multi-agent problem.

Earlier attempts to solve the automated configuration problem in automated negotiation mostly used basic approaches, such as random and grid search. More advanced methods, in the form of genetic algorithms, have also been attempted. Matos et al. [104] encoded a mix of baseline tactics as a chromosome and deployed a genetic algorithm to find the best mix. They assumed perfect knowledge of the opponent's preferences, and their strategy is only tested against itself in a single negotiation scenario. Eymann [46] encoded a more complex strategy as a chromosome with six parameters, again only testing its performance against itself and using the same scenario. Dworman et al. [43] implement the genetic algorithm in a coalition game with three players, with a strategy in the form of a hard-coded if-then-else rule. The parameters of the rule are implemented as a chromosome. The strategy is tested against itself in a coalition game with varying coalition values. Lau et al. [92] use a genetic algorithm to explore the outcome space during a negotiation session but do not use it to change the strategy.

This work aims to automatically configure a negotiation algorithm with no fixed or pre-defined strategy. This agent can be configured to perform well on a user-defined set of training problem instances, with little restrictions on the size of the instances or instance sets. To demonstrate its performance, we configure the agent in an attempt to win an Automated Negotiating Agents Competition (ANAC)-like bilateral tournament [10].

We show that we can win such a tournament with a comfortable margin of 5.1% in increased negotiation payoff compared to the number two. These margins are not observed in a tournament without our negotiation agent, where the winning strategy obtains a marginal improvement in negotiation payoff of 0.012%.

3.2 PROBLEM DESCRIPTION

In this section, we discuss the problem that we attempt to solve. We first describe the parameterised agent that we will configure (Section 3.2.1) and then give a formal problem definition using this parameterised agent (Section 3.2.2). For background on automated negotiation and algorithm configuration, we refer the reader to Section 2.1 and Section 2.2, respectively.

3.2.1 DYNAMIC AGENT

We first create a Dynamic Agent with a flexible strategy equivalent to a configuration space. We implement a few popular components and add their design choices to the configuration space, increasing the chances that it contains a successful strategy. We refer to this configuration space (or strategy space) with Θ. We name the constructed agent Dynamic Agent $DA(\theta)$, with strategy $\theta \in \Theta$.

The dynamic agent is constructed on the basis of the BOA-architecture [8]. We use this structure to give a brief overview of the workings of the dynamic agent and its configuration space.

BIDDING STRATEGY

The implemented bidding strategy applies a fitness value to the outcome space Ω and selects the outcome with the highest fitness as the offer, which is an approach used by Lau et al. [92]. This fitness function $f(\boldsymbol{\omega}, t)$ balances between our utility, the opponent's utility and the remaining time towards the deadline. Such a tactic is also known as a time-dependent tactic, and it generally concedes to the opponent as time passes.

The fitness function in Equation 3.1 has three parameters:

- A trade-off factor δ that balances between the importance of our own utility and the importance of reaching an agreement.
- A factor to control an agent's eagerness to concede e relative to time, where the behaviour is Boulware if $0 < e < 1$, linear conceder if $e = 1$, and conceder if $e > 1$.
- A categorical parameter n that sets the outcome where the fitness function concedes towards over time (Equation 3.2). Here, x^{last} is the last offer made by the opponent, and x^+ is the best offer the opponent made in terms of our utility.

$$\begin{aligned} f(\boldsymbol{\omega}, t) &= F(t) * u(\boldsymbol{\omega}) + (1 - F(t)) * f_n(\boldsymbol{\omega}) \\ F(t) &= \delta * (1 - t^{\frac{1}{e}}) \end{aligned} \tag{3.1}$$

$$\begin{aligned} f_1(\boldsymbol{\omega}) &= 1 - |\hat{u}_o(\boldsymbol{\omega}) - \hat{u}_o(x^{last})| \\ f_2(\boldsymbol{\omega}) &= \min(1 + \hat{u}_o(\boldsymbol{\omega}) - \hat{u}_o(x^{last}), 1) \\ f_3(\boldsymbol{\omega}) &= 1 - |\hat{u}_o(\boldsymbol{\omega}) - \hat{u}_o(x^+)| \\ f_4(\boldsymbol{\omega}) &= \min(1 + \hat{u}_o(\boldsymbol{\omega}) - \hat{u}_o(x^+), 1) \\ f_5(\boldsymbol{\omega}) &= \hat{u}_o(\boldsymbol{\omega}) \end{aligned} \tag{3.2}$$

Outcome space exploration The outcome space is potentially large. To reduce computational time and to ensure a fast response time for our agent, we apply a genetic algorithm to explore the outcome space in search of the best outcome. Standard procedures such as elitism, mutation and uniform crossover [109] are

Table 3.1: Configuration space in bidding strategy

Description	Symbol	Domain
Trade-off factor	δ	$[0,1]$
Conceding factor	e	$(0,2]$
Conceding goal	n	$\{1,2,3,4,5\}$
Population size	N_p	$[50,400]$
Tournament size	N_t	$[1,10]$
Evolutions	E	$[1,5]$
Crossover rate	R_c	$[0.1,0.5]$
Mutation rate	R_m	$[0,0.2]$
Elitism rate	R_e	$[0,0.2]$

Table 3.2: Configuration space in acceptance strategy

Description	Symbol	Domain
Scale factor	α	$[1,1.1]$
Utility gap	β	$(0,0.2]$
Accepting time	t_{acc}	$[0.9,1]$
Lower boundary utility	γ	$\{MAX^W, AVG^W\}$

applied, and the parameters of the genetic algorithm are added to the configuration space.

Configuration space The configuration space of the bidding strategy is summarized in Table 3.1.

OPPONENT MODEL

The Smith Frequency model [59] is used to estimate the opponent utility function $\hat{u}_o(\boldsymbol{\omega})$. According to an analysis by Baarslag et al. [11], the performance of this opponent modelling method is already quite close to that of the perfect model. No parameters are added to the configuration space of the Dynamic Agent.

ACCEPTANCE STRATEGY

The acceptance strategy decides when to accept an offer from the opponent. Baarslag et al. [14] performed an isolated and empirical research on popular acceptance conditions. They combined acceptance conditions and showed that a combined approach outperforms its parts. Baarslag et al. defined four parameters and performed a grid search in search of the best strategy. We adopt the combined approach and add its parameters (Table 3.2) to the configuration space of the Dynamic Agent. For more details on the combined acceptance condition, see Baarslag et al. [14].

3.2.2 PROBLEM DEFINITION

The negotiation agents in the General Environment for Negotiation with Intelligent multi-purpose Usage Simulation (GENIUS) environment [1] [99] are mostly based

[1] https://ii.tudelft.nl/genius/

on manually configured strategies by competitors in ANAC. These agents almost always contain parameters that are set by trial and error, despite the abundance of automated algorithm configuration techniques (e.g. Genetic Algorithm [71]). Manual configuration is a difficult and tedious job due to the dimensionality of both the configuration and the negotiation instance space.

A few attempts were made to automate this process as discussed in Section 3.1, but only on very specific negotiation settings with few configuration parameters. The main reason for this is that many automated configuration algorithms require to evaluate a challenging configuration on the full training set. To illustrate, evaluating the performance of a single configuration on the full training set that we use in this paper would take approximately 18.5 hours, regardless of the hardware due to the real-time deadline. These methods of algorithm configuration are, therefore, impractical.

Automated strategy configuration We have an agent called Dynamic Agent $DA(\theta)$, with strategy θ. We want to configure this agent such that it performs generally well using automated configuration methods. More specifically, we want the agent to perform generally well in bilateral negotiations with a real-time deadline of 60$[s]$. To do so, we take a diverse and large set of both agents $\mathcal{I}_{train}$ of size $|\mathcal{I}_{train}| = 20$ and scenarios S_{train} of size $|S_{train}| = 56$ that we use for training, making the total amount of training instances $|\mathcal{P}_{train}| = |\mathcal{I}_{train}| * |S_{train}| = 1120$. Running all negotiation settings in the training set would take 1120 minutes or $\sim$ 18.5 hours, regardless of the hardware, as we use real-time deadlines.

Now suppose we have a setting for the Dynamic Agent based on the literature θ_l and a setting that is hand tuned based on intuition, modern literature and manual tuning θ_m that we consider baselines. Can we automatically configure a strategy $\theta_{opt} \in \Theta$ that outperforms the baselines and wins an ANAC-like bilateral tournament on a never before seen test set of negotiation instances $\mathcal{P}_{test}$?

3.3 AUTOMATED CONFIGURATION

The goal of our work is to create an agent that can be configured to obtain a negotiation strategy that performs well in a given setting. This requires us to define what it means for a strategy to perform well. An obvious performance metric $m(\theta, p)$ is the utility obtained using strategy θ in negotiation instance p. As we are interested in optimizing performance on the full set of training instances rather than for a single instance, we define the performance of a configuration on an instance set as the average utility:

$$M(\theta, \mathcal{P}) = \frac{1}{|\mathcal{P}|} \cdot \sum_{p \in \mathcal{P}} m(\theta, p), \tag{3.3}$$

where:

m : utility obtained by using strategy θ in negotiation instance p
M : average utility of configuration θ on instance set $\mathcal{P}$
$\theta \in \Theta$: parameter configuration
p : single negotiation instance consisting of opponent agent $i \in \mathcal{I}$ and scenario $s \in S$, where $p = \langle a, s \rangle \in \mathcal{P}$
$\mathcal{P}$: set of negotiation instances

As stated in Section 3.2.2, automated configuration methods that require evaluation on the full training set of instances, thus requiring Equation 3.3 to be calculated, are impractical for our application. A second component that influences the amount of required evaluations is the mechanism that selects configurations for evaluation. This is not a straightforward problem, as the configuration space is large, and simple approaches, such as random search and grid search, suffer from the curse of dimensionality.

3.3.1 SMAC

To solve the problem defined in Section 3.2.2, we bring SMAC, a prominent, general-purpose algorithm configuration procedure [75], into the research area of automated negotiation. We note that SMAC is well suited for tackling the configuration problem arising in the context of our study:

1. It can handle different types of parameters, including real- and integer-valued as well as categorical parameters.
2. It can configure on subsets of the training instance set, reducing the computational expense.
3. It has a mechanism to terminate poorly performing configurations early, saving computation time. If it detects that a configuration is performing very poorly on a small set of instances (e.g., a very eager conceder), it stops evaluating and drops the configuration.
4. It models the relationship between parameter settings, negotiation instance features and performance, which tends to significantly reduce the effort of finding good configurations.
5. It permits parallelisation of the configuration process by means of multiple independent runs, which leads to significant reductions in wall-clock time.

SMAC keeps a run history (Equation 3.4), consisting of a configuration θ_i with its associated utility m_i on a negotiation instance that is modelled by a feature set $\mathcal{F}(p)$. A random forest regression model is fitted to this run history, mapping the configuration space and negotiation instance space to a performance estimate $\hat{m}$ (Equation 3.5). This model is then used to predict promising configurations, which are subsequently raced against the best configuration found so far, until an overall time budget is exhausted. We refer the reader to Hutter et al. [75] for further details on SMAC.

$$R = \{(\langle \theta_1, \mathcal{F}(p) \rangle, o_1), \ldots, (\langle \theta_n, \mathcal{F}(p) \rangle, o_n)\} \tag{3.4}$$

Table 3.3: Scenario features

Feature type	Description	Equation	Notes
Domain	Number of issues	$\|B\|$	
Domain	Average number of values per issue	$\frac{1}{\|B\|}\sum_{b\in B}\|\Omega_b\|$	
Domain	Number of possible outcomes	$\|\Omega\|$	
Preference	Standard deviation of issue weights	$\sqrt{\frac{1}{\|B\|}\sum_{b\in B}(w(b)-\frac{1}{\|B\|})^2}$	
Preference	Average utility of all possible outcomes	$\frac{1}{\|\Omega\|}\sum_{\omega\in\Omega}u(\omega)$	denoted by $u(\bar{\omega})$
Preference	Standard deviation utility of all possible outcomes	$\sqrt{\frac{1}{\|\Omega\|}\sum_{\omega\in\Omega}(u(\omega)-u(\bar{\omega}))^2}$	

$$\mathcal{M} : (\Theta \times \mathcal{P}) \rightarrow \hat{m} \tag{3.5}$$

In order for SMAC to be successful in predicting promising configurations, it requires an accurate feature description of the negotiation instances that captures differences in complexity between these instances.

Automated algorithm configuration Suppose we have a set of opponent agents $\mathcal{I}$ and a set of negotiation scenarios S, such that combining a single agent $i \in \mathcal{I}$ and a single scenario $s \in S$ creates a new negotiation setting or instance $p \in \mathcal{P}$. Can we derive a set of features for both the opponent and the scenario that characterize the complexity of the negotiation instance?

We approach this question empirically by analyzing if a candidate feature set helps the automated algorithm configuration method find better configurations within the same computational budget.

3.4 INSTANCE FEATURES

The negotiation instances consist of an opponent and a scenario. We will extract features for both components separately and then combine them as a feature set of an instance (Equation 3.6). This feature description is used by the configuration method to predict promising strategies for our Dynamic Agent $DA(\theta)$.

$$\mathcal{F} : \mathcal{P} \rightarrow (X_{sc} \times X_{opp}) \tag{3.6}$$

3.4.1 SCENARIO FEATURES

A negotiation scenario consists of a shared domain and individual preference profiles. Ilany and Gal [76] specified a list of features to model a scenario that they used for strategy selection in bilateral negotiation. Although the usage differs in their paper, the goal to model the scenario is the same, so we will follow Ilany et al.. The features are fully independent of the opponent's behaviour. An overview of the scenario features is provided in Table 3.3.

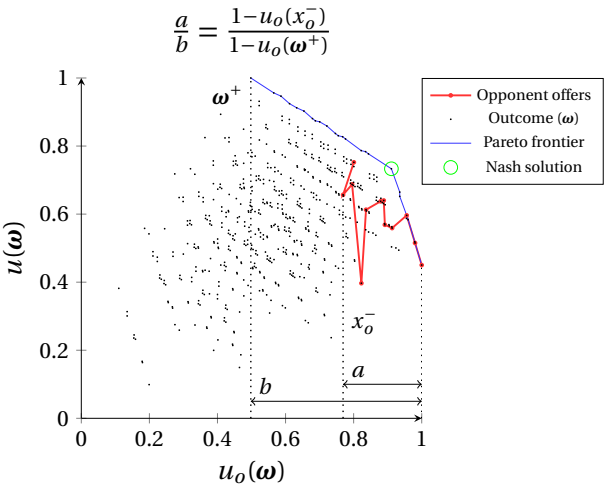


Figure 3.1: Visualisation of Concession Rate (CR)

3.4.2 OPPONENT FEATURES

This section describes the opponent features in detail. For each opponent, we store both the *mean* and the *Coefficient of Variance (CoV)* of all features.

NORMALIZED TIME

The time $t \in [0,1]$ it takes to reach an agreement with the opponent.

CONCESSION RATE

To measure how much an opponent is willing to concede towards our agent, we use the notion of Concession Rate (CR) introduced by Baarslag et al. [15]. The CR is a normalized ratio $CR \in [0,1]$, where $CR = 1$ means that the opponent fully conceded and $CR = 0$ means that the opponent did not concede at all. By using a ratio instead of an absolute value (utility), the feature is disassociated from the scenario.

To calculate the CR, Baarslag et al. [15] used two constants. The minimum utility an opponent has demanded during the negotiation session $u_o(x_o^-)$ and the Full Yield Utility (FYU), which is the utility that the opponent receives at our maximum outcome $u_o(\omega^+)$.

We present a formal description of the CR in Equation 3.7 and a visualisation in Figure 3.1.

$$CR(x_o^-) = \begin{cases} 1 & \text{if } u_o(x_o^-) \leq u_o(\omega^+), \\ \frac{1-u_o(x_o^-)}{1-u_o(\omega^+)} & \text{otherwise.} \end{cases} \tag{3.7}$$

AVERAGE RATE

We introduce the Average Rate (AR) that indicates the average utility an opponent has demanded as a ratio depending on the scenario. The two constants needed are the FYU ($u_o(\omega^+)$) as described in the previous section and the average utility an

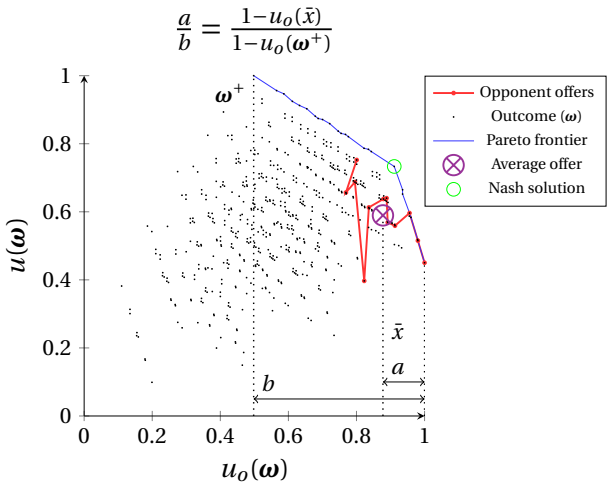


Figure 3.2: Visualisation of Average Rate (AR)

opponent demanded ($u_o(\bar{x})$). The AR is a normalized ratio $AR \in [0,1]$, where $AR = 0$ means that the opponent only offered his maximum outcome and $AR = 1$ means that the average utility the opponent demanded is less than or equal to the FYU. We present a definition of the AR in Equation 3.8 and a visualisation in Figure 3.2.

$$AR(\bar{x}) = \begin{cases} 1 & \text{if } u_o(\bar{x}) \leq u_o(\boldsymbol{\omega}^+), \\ \frac{1-u_o(\bar{x})}{1-u_o(\boldsymbol{\omega}^+)} & \text{otherwise.} \end{cases} \tag{3.8}$$

The AR is another indication of competitiveness of the opponent based on average utility demanded instead of minimum demanded utility as the CR is.

DEFAULT CONFIGURATION PERFORMANCE

According to Hutter et al. [75], the performance of any default configuration on a problem works well as a feature for that specific problem. For negotiation, this translates to the obtained utility of a hand-picked default strategy on a negotiation instance. The obtained utility is normalized and can be used as a feature for that negotiation instance.

We implement this concept as an opponent feature by selecting a default strategy and using it to obtain an agreement $\boldsymbol{\omega}_{agree}$ with the opponent. We then normalize the obtained utility and use it as the Default Configuration Performance (DCP) feature. We present the formal definition of this feature in Equation 3.9 and a visualisation in Figure 3.3.

$$DCP(\boldsymbol{\omega}_{agree}) = \begin{cases} 0 & \text{if } u(\boldsymbol{\omega}_{agree}) \leq u(\boldsymbol{\omega}^-), \\ \frac{u(\boldsymbol{\omega}_{agree})-u(\boldsymbol{\omega}^-)}{1-u(\boldsymbol{\omega}^-)} & \text{otherwise.} \end{cases} \tag{3.9}$$

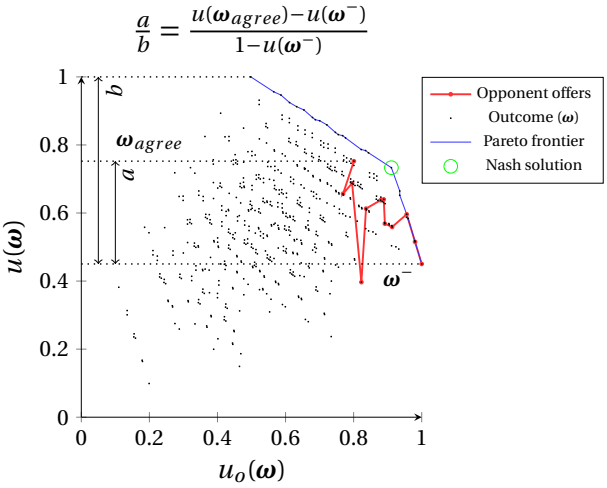


Figure 3.3: Visualisation of Default Configuration Performance (DCP)

Table 3.4: Opponent utility function usage

	Training	**Testing**
$DA(\theta)$	$\hat{u}_o(\omega)$	$\hat{u}_o(\omega)$
SMAC	$u_o(\omega)$	N/A

3.4.3 OPPONENT UTILITY FUNCTION

As can be seen in Figure 3.1, Figure 3.2, and Figure 3.3, the actual opponent utility function $u_o(\omega)$ is used to calculate the opponent features. SMAC is only used to configure the Dynamic Agent on the training set. As the opponent features are only used by SMAC, we can safely use the opponent's utility function to construct those features (Equation 3.7, Equation 3.8 and Equation 3.9) without giving the Dynamic Agent an unfair advantage during testing. The Dynamic Agent always uses the predicted opponent utility $\hat{u}_o(\omega)$ obtained through the model (Table 3.2.1), as is conventional in the ANAC.

We provide an overview of when the predicted opponent utility function and when the actual opponent utility function is used in Table 3.4.

3.5 EMPIRICAL EVALUATION

We must set baseline configurations to compare to the result of the optimisation. The basis of our Dynamic Agent is derived from a paper by Lau et al. [92]. Though some functionality is added, it is possible to set our agent's strategy to resemble that of the original agent. We refer to this configuration from the literature as θ_l, its parameters can be found in Table 3.5.

Another baseline strategy is added, which is configured manually, as the literature configuration is outdated. A combination of intuition, past research, and

Table 3.5: Baseline configurations parameters

	Accepting				Fitness function			Space exploration					
θ	α	β	t_{acc}	γ	n	δ	e	N_p	N_t	E	R_c	R_m	R_e
θ_l	1	0	1	MAX^W	1	0.5	0.5	200	3	3	0.6	0.05	0.1
θ_m	1	0	0.98	MAX^W	4	0.95	0.05	300	5	4	0.6	0.05	0.05

manual search is used for this manual configuration, which we consider the default method for current ANAC competitors. We present the manually configured parameters θ_m in Table 3.5 and an explanation below:

- *Accepting*: The acceptance condition parameters of θ_l set a pure AC_{next} strategy with parameters $\alpha = 1$, $\beta = 0$. Baarslag et al. [14] performed empirical research on a variety of acceptance conditions and showed that there are better alternatives. We set the accepting parameters of our configuration to the best-performing condition as found by Baarslag et al. [14].
- *Fitness function*: Preliminary testing showed that the literature configuration concedes much faster than the average ANAC agent, resulting in a poor-performing strategy. We set a more competitive parameter configuration for the fitness function by manual search to match the competitiveness of the ANAC agents.
- *Space exploration*: The domain used in the paper has a relatively small set of outcomes. We increased the population size, added an extra evolution to the genetic algorithm and made some minor adjustments to cope with larger outcome spaces.

3.5.1 METHOD

SMAC is run in *embarrassingly parallel* mode on a computing cluster by starting a separate SMAC process on chunks of allocated hardware. SMAC selects a negotiation instance and a configuration to evaluate that instance and calls the negotiation environment GENIUS through a wrapper function.

Input The training instances were created by selecting a diverse set of opponents and scenarios from the GENIUS environment. The scenarios have non-linear utility functions and vary in competitiveness and outcome space size (between 9 and 400 000). The scenario features were calculated in advance as described in Section 3.4.1, and the configuration space is defined in Section 3.2.1.

The opponent features, as defined in Section 3.4.2, can only be gathered by performing negotiations against the opponents. We gather these features in advance by negotiating 10 times in every instance with the manual strategy θ_m.

Hardware & configuration budget We perform 300 independent parallel runs of SMAC for 4 hours of wall-clock time each, on a computing cluster running Simple

Table 3.6: Configurations overview

	Accepting				Fitness function			Space exploration					
θ	α	β	t_{acc}	γ	n	δ	e	N_p	N_t	E	R_c	R_m	R_e
θ_l	1	0	1	MAX^W	1	0.5	0.5	200	3	3	0.6	0.05	0.1
θ_m	1	0	0.98	MAX^W	4	0.98	0.05	300	5	4	0.4	0.05	0.05
θ_1	1.001	0.048	0.901	AVG^W	3	0.879	0.00183	345	10	4	0.437	0.003	0.176
θ_2	1.041	0.001	0.904	AVG^W	4	0.913	0.00130	384	5	4	0.431	0.126	0.198
θ_3	1.009	0.026	0.910	MAX^W	1	0.977	0.00113	361	2	5	0.279	0.181	0.072
θ_4	1.032	0.022	0.931	AVG^W	3	0.914	0.00429	311	8	3	0.251	0.082	0.132
θ_5	1.015	0.017	0.925	AVG^W	5	0.961	0.00105	337	5	3	0.192	0.090	0.138
θ_6	1.027	0.022	0.943	AVG^W	3	0.985	0.00227	283	7	4	0.294	0.057	0.156

Linux Utility for Resource Management (SLURM). To ensure consistent results, all runs were performed on Intel® Xeon® CPU, allocating 1 CPU core, with 2 processing threads and 12 GB RAM to each run of SMAC.

Output Every parallel SMAC process outputs its best configuration θ_{inc} after the time budget is exhausted. As there are 300 parallel processes, a decision must be made on which of the 300 configurations to use. To do so, the SMAC random forest regression model conform Equation 3.5 is rebuilt and used to predict the performance of every θ_{inc}. The configuration with the best-predicted performance is selected as the best configuration θ_{opt}.

3.5.2 RESULTS

The configuration process, as described, is run three times without instance features and three times with instance features, under identical conditions. There is now a total of 8 strategies: 2 baselines $[\theta_l, \theta_m]$, 3 optimized without features $[\theta_1, \theta_2, \theta_3]$, and 3 optimised with features $[\theta_4, \theta_5, \theta_6]$. An overview of the final configurations is presented in Table 3.6.

The obtained configurations are now analyzed with an emphasis on the following three topics:

1. The influence of the instance features on the convergence of the configuration process.

2. The performance of the obtained configurations on a never-before-seen set of instances.

3. The performance of the best configuration in an ANAC-like bilateral tournament.

INFLUENCE OF INSTANCE FEATURES

To study the influence of the instance features on the configuration process, we compare the strategies obtained by configuring with features and configuring without features. Only the training set of instances is used for the performance comparison, as we are purely interested in the convergence towards a higher utility.

Table 3.7: Performance of configurations on $\mathcal{P} = \mathcal{P}_{train}$

θ	$M(\theta, \mathcal{P})$	$\frac{M(\theta,\mathcal{P})-M(\theta_m,\mathcal{P})}{M(\theta_m,\mathcal{P})}$	**Description**
θ_l	0.533	-0.307	Literature
θ_m	0.769	0	Manually configured
θ_1	0.785	0.020	Configured without features
θ_2	0.770	0.000	Configured without features
θ_3	0.792	0.029	Configured without features
θ_4	0.800	0.040	Configured with features
θ_5	0.816	0.060	Configured with features
θ_6	0.803	0.044	Configured with features

Table 3.8: Performance of configurations on $\mathcal{P} = \mathcal{P}_{test}$

θ	$M(\theta, \mathcal{P})$	$\frac{M(\theta,\mathcal{P})-M(\theta_m,\mathcal{P})}{M(\theta_m,\mathcal{P})}$	**Description**
θ_l	0.563	-0.261	Literature
θ_m	0.763	0	Manually configured
θ_1	0.779	0.021	Configured without features
θ_2	0.760	-0.004	Configured without features
θ_3	0.774	0.015	Configured without features
θ_4	0.792	0.038	Configured with features
θ_5	0.795	0.042	Configured with features
θ_6	0.789	0.034	Configured with features

Every configuration is run 10 times on the set of training instances $\mathcal{P}_{train}$, and the average obtained utility is calculated by Equation 3.3. The results are presented in Table 3.7, including an improvement ratio over θ_m.

SMAC is capable of improving the performance of the Dynamic Agent above our capabilities of manual configuration. We observe that configuration without instance features potentially leads to marginal improvements on the training set. Finally, we observe that the usage of instance features leads to less variation in final configuration parameters (Table 3.6) and to a significant improvement of obtained utility.

PERFORMANCE ON TEST SET

Testing the configurations on a never-before-seen set of opponent agents and scenarios is needed to rule out potential overfitting. We selected a diverse set of scenarios and opponents for testing, such that $|\mathcal{P}_{test}| = |A_{test}| * |S_{test}| = 16 * 28 = 448$.

Every configuration is once again run 10 times on the set of training instances $\mathcal{P}_{test}$ and the average obtained utility is calculated by Equation 3.3. The results are presented in Table 3.8, including an improvement ratio over θ_m.

It is now clear that strategy configuration without instance features is undesirable as it potentially leads to a worse-performing strategy. Configuration with instance feature, on the other hand, still leads to a significant performance increase on a never-before-seen set of negotiation instances.

ANAC TOURNAMENT PERFORMANCE OF BEST CONFIGURATION

The strategy configuration method is successful in finding improved configurations, but the results are only compared against the other configurations of our Dynamic Agent. No comparison is yet made with agents built by ANAC competitors. We now compare the performance of the best configuration that we found to the ANAC agents in the test set of opponents.

We select θ_5 as the best strategy based on performance on the training set and enter the Dynamic Agent in an ANAC-like a bilateral tournament with a 60-second deadline. The Dynamic Agent is combined with the test set of opponents and scenarios. Every combination of 2 agents negotiated 10 times on every scenario, for a total amount of 38080 negotiation sessions. The averaged results are presented in Table 3.9. We elaborate on the performance measures found in the table:

- **Utility:** The utility of the agreement.
- **Opp. utility:** The opponent's utility of the agreement.
- **Social welfare:** The sum of utilities of the agreement.
- **Pareto distance:** Euclidean distance of the agreement to the nearest outcome on the Pareto frontier in terms of utility.
- **Nash distance:** Euclidean distance of the agreement to the Nash solution in terms of utility (Equation 2.3).
- **Agreement ratio:** The ratio of negotiation sessions that result in an agreement.

Using the Dynamic Agent with θ_5 results in a successful negotiation agent that is capable of winning a ANAC-like bilateral tournament by outperforming all other agents (two-tailed t-test: $p < 0.001$). It managed to obtain a $\frac{0.795-0.756}{0.756} * 100\% \approx 5.1\%$ higher utility than SimpleAgent, the number two in the ranking, while also outperforming it on every other performance measure.

Since the presence of our agent in the tournament also influences the performance of other agents, we also ran the full tournament without our Dynamic Agent as a sanity check. The top 5 performers of this tournament are presented in Table 3.10, along with their margins over the respective next lower-ranking agent in terms of utility.

3.6 CONCLUSION

The two main contributions of this chapter are (1) the success of automated configuration of negotiation strategies using a general-purpose configuration procedure (here: SMAC), and (2) an investigation of the importance of the features of negotiation settings.

Table 3.9: Bilateral ANAC tournament results using $DA(\theta_5)$ (bold = best, underline = worst)

Agent	**Utility**	**Opp. utility**	**Social welfare**	**Pareto distance**	**Nash distance**	**Agreement ratio**
RandomCounterOfferParty	0.440	**0.957**	1.398	0.045	0.415	1.000
HardlinerParty	0.496	0.240	0.735	0.507	0.754	0.496
AgentH	0.518	0.801	1.319	0.118	0.408	0.904
ConcederParty	0.577	0.848	1.425	0.047	0.358	0.964
LinearConcederParty	0.600	0.831	1.431	0.046	0.350	0.964
PhoenixParty	0.625	0.501	1.125	0.263	0.468	0.748
GeneKing	0.637	0.760	1.396	0.061	0.383	0.993
Mamenchis	0.651	0.725	1.377	0.087	0.360	0.927
BoulwareParty	0.662	0.786	1.448	0.043	0.319	0.968
Caduceus	0.677	0.486	1.163	0.241	0.453	0.784
Mosa	0.699	0.640	1.339	0.113	0.385	0.902
ParsCat2	0.716	0.671	1.386	0.108	**0.286**	0.904
RandomDance	0.737	0.716	**1.453**	**0.024**	0.344	0.998
ShahAgent	0.744	0.512	1.256	0.188	0.389	0.821
AgentF	0.751	0.605	1.356	0.100	0.367	0.918
SimpleAgent	0.756	0.437	1.194	0.212	0.470	0.801
$DA(\theta_5)$	**0.795**	0.566	1.361	0.087	0.407	0.922

Table 3.10: Bilateral ANAC tournament without $DA(\theta_5)$

Agent	**Utility**	**Margin**
Mosa	0.715	3.01%
ShahAgent	0.736	2.43%
RandomDance	0.754	0.65%
AgentF	0.759	0.01%
SimpleAgent	0.759	

3.6.1 CONFIGURATION

Two baseline strategies were selected for our comparison. The first configuration, θ_l, is based on publications from which we derived the agent [92, 14]. The second configuration, θ_m, is configured based on intuition, recent literature and manual search, which we considered the default approach for current ANAC competitors. In Section 3.5, we automatically configured our dynamic Agent using SMAC.

The configuration based on earlier work θ_l [92] performed poorly compared to the manually configured configuration θ_m, and achieved 26.1% lower utility on our test set. The best automatically configured strategy θ_5 outperformed both baseline configurations and achieved a 4.2% increase in utility compared to θ_m. From this, we conclude that the automated configuration method is successful in outperforming manual configuration.

Our experiments show that the automated configuration method can produce a strategy that can win an ANAC-like bilateral tournament by a margin of 5.1% (Table 3.9). This is particularly striking when considering that without our agent, the winner of the same tournament beats the next-based agent only by a margin of 0.01%.

3.6.2 FEATURES

We consider a set of features that characterizes the negotiation scenario as well as the opponent. Our empirical results indicate that when using the negotiation instance features, SMAC is able to find good configurations faster.

Overall, using SMAC in combination with instance features leads to less variation in the parameter settings between the final configurations obtained in multiple independent runs (Table 3.6, Table 3.7), as well as significant and consistent performance improvement. Furthermore, our results show that automated configuration without features does not always outperform manual configuration. Therefore, we conclude that the instance features presented in this chapter are a necessary ingredient for the successful automated configuration of negotiation strategies.

3.6.3 NEXT STEPS

For this initial step towards automated configuration of negotiation agents, the negotiation scenarios were simplified by removing the reservation utility and the discount factor. Now that we have demonstrated that our general approach can be successful, additional validation should be performed in more complex and different negotiation environments.

Over the years, it became clear that there is no single best negotiation strategy for all negotiation settings [99]. In this chapter, we have presented a method to automatically configure an effective strategy for a specific set of negotiation settings. However, if this set becomes too diverse, we inherently end up in a situation where the automatically configured best strategy may not perform too well. In the next chapter, we exploit the strategy space of the dynamic agent by extracting multiple complementary strategies for specific settings, along with an online selection mechanism that determines the strategy to be used in a specific instance.

4

Configuration of Strategy Portfolios

This chapter has been published at the Cooperative AI workshop at NeurIPS 2021 [122] and AAMAS 2022 [123].

4.1 INTRODUCTION

The strategies of negotiation agents almost always remain monolithic, i.e. single strategy with fixed behaviour for every setting, the exceptions we found are, e.g., [76, 137]. It has been observed that no single strategy is optimal for all negotiation instances [76, 99] and that the success of a negotiator also depends on the strategy of the opponent [10]. In Chapter 3, we successfully showed that we can use automated configuration techniques to optimise negotiation strategies. However, the obtained strategy is also fixed in that it does not adapt to various opponent types or negotiation scenarios. A good way to further improve pay-off would be to select from a portfolio of strategies based on the negotiation game. This introduces the problem of algorithm selection [127] into negotiation. An early attempt to apply algorithm selection in automated negotiation was made by Ilany and Gal [77, 76], but they only selected a strategy based on the negotiation scenario, without considering the opponent, which we know to be an essential factor [10]. Furthermore, they relied on a portfolio of existing strategies to select from, which potentially limits robustness.

Our contributions in this chapter are as follows: (i) we apply automated algorithm configuration techniques to not only create a single negotiation strategy, but a portfolio of complementary negotiation strategies; and (ii) we introduce a procedure to learn and exploit opponent and scenario characteristics during a simulated Automated Negotiating Agents Competition (ANAC) tournament. Our method uses the approach from Chapter 3 to automatically configure negotiation strategies, which we extend by implementing HYDRA [162] for portfolio construction and AutoFolio [100] to create a portfolio selector. Empirical results on a variety of negotiation instances show that our method beats the runner-up agent by a (comfortable) margin of 5.6%.

4.2 RELATED WORK

Thanks to ANAC, new negotiation strategies are developed every year and collected in the General Environment for Negotiation with Intelligent multi-purpose Usage Simulation (GENIUS) test-bed [99], to support future research; they are categorised and empirically evaluated [10, 8] to provide a basis for new strategies.

As there is no single best strategy for all negotiation instances [76, 99], we should be able to improve pay-off by exploiting differences in instances by selecting different strategies per negotiation instance. We see this as a variation of the algorithm selection problem [127]. While algorithm selection methods have been successfully applied to other problems, only few attempts have been made to apply them in the area of automated negotiation. Ilany and Gal [77, 76] and Güneş et al. [64] used a set of past ANAC strategies and predicted which strategy would perform best on a given negotiation instance; they then entered that strategy into the negotiation session. Although they managed to improve the pay-off of the agent in this manner, they were unable to win ANAC. Kawata and Fujita [84] used a portfolio of 7 strategies that previously competed in ANAC. They applied a multi-armed bandit approach to find the best-performing strategy for every combination of an opponent and scenario while repeating precisely the same negotiation setting 100 times. Unfortunately,

this strategy does not generalise to unseen negotiation instances. Sengupta et al. [137] trained both a set of strategies and a selection mechanism. Strategy selection on a more fine-grained level by selecting modular components of a negotiation strategy using reinforcement learning has also been attempted [18].

4.3 PRELIMINARIES

This chapter requires background knowledge in automated negotiation (Section 2.1), algorithm configuration (Section 2.2), and algorithm selection (Section 2.3).

We continue the work of the previous chapter and extend it with portfolio-building methods and algorithm selection techniques. The dynamic agent $DA(\theta)$ with parameter configuration space Θ from Section 3.2.1 is used as a basis. We also use the same scenario features and opponent features as described in Section 3.4. A deadline of 60 seconds is used for the Alternating Offers Protocol (AOP) [132] in this chapter, normalised to $t \in [0,1]$, after which negotiation is aborted without agreement. The performance metric $m(\theta, p)$ of a configuration θ on negotiation instance p is the obtained utility, where the performance on a set of negotiation instances is:

$$M(\theta, \mathcal{P}) = \frac{1}{|\mathcal{P}|} \cdot \sum_{p \in \mathcal{P}} m(\theta, p), \tag{4.1}$$

4.3.1 PROBLEM DEFINITION

Note that in this chapter, we consider algorithm selection to be performed on portfolios of parameter configurations of the same algorithm, thus replacing ψ in Section 2.3 with θ.

Strategy portfolio creation. We have an agent with a dynamic strategy $DA(\theta)$ based on configuration space Θ. Can we create a portfolio of configurations $\boldsymbol{\theta} \subset \Theta$ using a training set of negotiation instances $\mathcal{P}$ consisting of configurations that outperform each other on specific subsets of a test set of negotiation instances $\mathcal{P}'_{test} \subset \mathcal{P}_{test}$ that have never been encountered before?

Algorithm selection. We have an agent with a dynamic strategy $DA(\theta)$, and a portfolio of configurations $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_n\}$, where θ_1 is the single best-performing configuration (Equation 4.3). Can we apply an algorithm selection method $\theta_p = AS(\boldsymbol{\theta}, p)$ that selects a configuration θ_p from $\boldsymbol{\theta}$ based on negotiation setting p, such that $M(AS(\boldsymbol{\theta}, p), \mathcal{P}_{test}) > M(\theta_1, \mathcal{P}_{test})$. The real goal here is to let $M(AS(\boldsymbol{\theta}, p), \mathcal{P}_{test})$ approach the performance of the oracle selector (Equation 4.2) $M(OR(\boldsymbol{\theta}, p), \mathcal{P}_{test})$ as closely as possible.

$$OR(\boldsymbol{\theta}, p) \in \arg\max_{\theta \in \boldsymbol{\theta}} m(\theta, p) \tag{4.2}$$

$$\theta_1 \in \arg\max_{\theta \in \boldsymbol{\theta}} M(\theta, \mathcal{P}) \tag{4.3}$$

4.4 PORTFOLIO CREATION

As a basis for algorithm selection, we need a portfolio of negotiation strategies to select from. A simple approach is to build a portfolio of negotiation strategies that already exist within the GENIUS environment, which is the approach used by Ilany and Gal [76]. However, for several reasons, we consider this a less-than-ideal approach:

1. It relies on strategies that already exist, thus limiting our choices for a portfolio to strategies that have been previously implemented and are available to be re-used.

2. The strategies might not be optimised or optimised for a different objective, resulting in a low-performance portfolio.

3. There might be dominated strategies in the portfolio, which are outperformed in all cases by some other strategy in the portfolio, needlessly complicating the selection problem.

4. The portfolio might not be robust. There can be negotiation settings for which all the negotiation strategies fail to achieve decent performance, causing "weak spots" in our portfolio.

4.4.1 PORTFOLIO CREATION

We aim to expand upon the work of Chapter 3, by not only automatically configuring a single negotiation strategy, but by building a portfolio of complementary strategies to better exploit differences between negotiation instances. The portfolio of strategies $\boldsymbol{\theta}$ we create is thus a portfolio of configurations for our $DA(\theta)$. In our method we will therefore enforce that every strategy must add value to the portfolio:

$$\forall \theta \in \boldsymbol{\theta},\ \exists p \in \mathcal{P},\ \forall \theta' \in (\boldsymbol{\theta} \setminus \theta)\ :\ m(\theta, p) > m(\theta', p) \tag{4.4}$$

The portfolio can be viewed as a set of strategies that each specialise on a region within the negotiation instance space. Similarities in this space are found by mapping the space to the feature space. One could obtain such a portfolio by automatically configuring strategies on sets of negotiation instances that are separated in feature space by dividing the feature space either manually or using clustering techniques. However, both methods rely on human input without clear insight into the effects. The quality of the sets is disputable, as they are created based on similarities in the given feature space without regard for the performance gains thus achieved. Therefore, instead, we chose to automate the portfolio creation method by using HYDRA [162], removing the requirement of human input in feature space separation.

4.4.2 HYDRA

HYDRA automatically generates a portfolio given only a parameterised strategy (Section 3.2.1) and a set of negotiation instances with features (Section 3.4) while

Algorithm 4.1 HYDRA [162]

Input	Θ	Configuration space
	$\mathcal{P}$	Training set of negotiation instances
	m	Performance metric
Variables	θ_k	Configuration
	$\boldsymbol{\theta}$	Portfolio of configurations
	m_k	Modified performance metric
Output	$\boldsymbol{\theta}$	Portfolio of configurations
	AS	Algorithm selector

1: $\boldsymbol{\theta} \leftarrow \emptyset$; $m_k \leftarrow m$
2: **for** $k = 1$; $k = k + 1$ **do**
3: $\quad \theta_k \leftarrow SMAC(\Theta, \mathcal{P}, m_k)$
4: $\quad TestPerformance(\mathcal{P}, \theta_k)$
5: $\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} \cup \{\theta_k\}$
6: $\quad AS \leftarrow FitAlgorithmSelector(\boldsymbol{\theta}, \mathcal{P})$
7: $\quad m_k \leftarrow GetModifiedPerformanceMetric(m, AS)$
8: $\quad$ **if** θ_k is not contributing to $\boldsymbol{\theta}$ on S **then**
9: $\quad\quad$ End for loop
10: **return** AS, $\boldsymbol{\theta}$

using an algorithm configurator and an algorithm selector (Section 4.5). We provide a pseudo-code description of HYDRA in Algorithm 4.1, modified for this work.

The main idea of HYDRA is to perform multiple configurator runs on an identical set of training instances while only modifying the performance metric. Due to the modifications to the metric, the configurator produces different strategies. In Algorithm 4.1, the modified performance metric is computed by "*GetModifiedPerformanceMetric*" and formally defined as:

$$m_k(\theta, p) = \max\{m(\theta, p),\ m(AS(\boldsymbol{\theta}, p), p)\}. \tag{4.5}$$

The modified performance is the better of the performance of the strategy that is assessed and the performance of the strategy that the algorithm selector selects. By optimising the increase of performance as compared to the current portfolio, the configurator aims to find a configuration that adds the most value to the portfolio. In the first configurator run, the default performance metric is used. The resulting configuration θ_1 is therefore a locally optimal configuration over the full set of training settings, also known as the *single best strategy* in the portfolio.

4.5 STRATEGY SELECTION

The next step in our approach is strategy selection. We now have a portfolio of strategies $\boldsymbol{\theta}$, but still need to decide which of these strategies best fits our current scenario and opponent. We, therefore, desire a mapping from the feature space X to a one-hot distribution over the possible strategies. This is an algorithm selection problem [127] and is illustrated in Figure 4.1, modified for our work. Essentially, it is a classification problem for which we can train a classifier on examples generated from our training set. Subsequently, we hope the learned function will generalise to new negotiation scenarios and unknown opponents in the test set, allowing us to select the most suitable strategy from our portfolio.

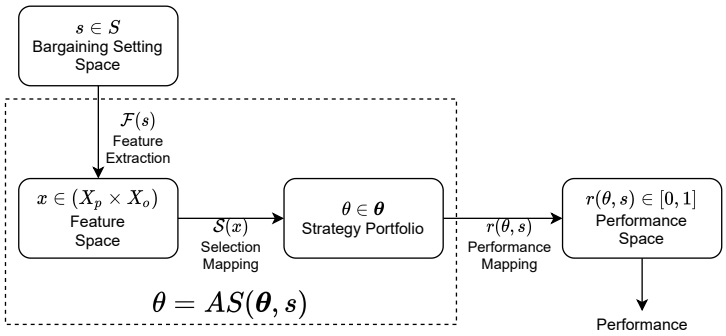


Figure 4.1: Algorithm selection schematics [127], modified for this work.

4

Ilany and Gal [76] also considered this algorithm selection problem and analysed the performance of multiple classifiers that map feature vectors to algorithms. The process of selecting a classifier and configuring the accompanying parameters can again be seen as an algorithm configuration problem. In line with the rest of this paper, we chose to automate the configuration of an algorithm selector by using AutoFolio [100], leveraging the power of a broad range of algorithm selection methods and removing human bias.

4.5.1 AUTOFOLIO

The algorithm selection system AutoFolio is used to construct the algorithm selector. It has a range of regression and classification methods to choose from and uses Sequential Model-based optimization for general Algorithm Configuration (SMAC) to determine both the selection method to use and the settings of its hyperparameters. The data AutoFolio requires as input is the performance $m(\theta, p)$ of every strategy ($\theta \in \boldsymbol{\theta}$) on every setting ($p \in \mathcal{P}$) in the training set and a set of features. Its goal is to select the best-performing strategy for every negotiation setting.

4.5.2 CROSS VALIDATION.

AutoFolio uses 10-fold cross-validation during optimisation to avoid overfitting by dividing the negotiation instances in the training set into 10 subsets and leaving one subset out for performance testing. However, due to the nature of a negotiation instance being a combination of an opponent and a negotiation scenario, this leads to overfitting of the algorithm selector. The training set of negotiation instances is the Cartesian product of the training set of opponents and scenarios, so both components are included multiple times in the training set.

To address this issue, we modified AutoFolio to split the cross-validation folds based on the set of opponents and scenarios that build the negotiation instances. The set of opponents and the set of scenarios are each split into 4 subsets, such that we obtain a total of $4 \cdot 4 = 16$ folds. When selecting a fold ($|\mathcal{P}_{fold}| = \frac{1}{16} \cdot |\mathcal{P}|$), we must eliminate the part of the remaining training set ($|\mathcal{P}_{elim}| = \frac{6}{16} \cdot |\mathcal{P}|$) that overlaps with the fold based on opponents and scenarios and use the remaining

instances ($|\mathcal{P}_{fit}| = \frac{9}{16} \cdot |\mathcal{P}|$) to fit the algorithm selector. This cross-validation approach reduces the workable size of the training set, but it does prevent training on test opponents/scenarios.

4.5.3 PERFORMANCE BASELINES

The oracle selector (Equation 4.2) always makes the perfect choice for every negotiation setting and is an upper bound on the performance of a selector using the given portfolio. It is obtained by simply trying every strategy on every setting and selecting the best strategy. The *single best strategy* is the strategy in the portfolio that obtains the highest performance on the full set of negotiation settings (Equation 4.3). We refer to this strategy as θ_1, as it is the first strategy in the portfolio produced by HYDRA. The performance of the single best strategy is considered to be the baseline.

4.6 EMPIRICAL EVALUATION

We will first describe the method that was used to obtain the results of this work before we show the results.

4.6.1 METHOD

The first configurator run with the default performance metric results in the single best strategy θ_1 on the training set of negotiation settings. We iterated through HYDRA until $k = 4$. At that point, the Hydra loop was terminated, as the last strategy that was added did not contribute to the portfolio based on the training set, which will be shown in Section 4.6. This also allows us to analyse the performance of portfolios of sizes 1, 2 and 3, due to the incremental approach of HYDRA. The configurations thus obtained were tested 10 times on every negotiation setting in the training set to capture performance variation due to randomness in the negotiation strategies. Finally, the portfolio and the performance data were used along with the setting features to configure an algorithm selector using AutoFolio.

INPUT

An overview of the opponents that are used in this work can be found in Table 4.1. The test set of opponents $\mathcal{I}_{opp,test}$ consists of the bug-free ANAC 2017 agents. More recent ANAC agents are not compatible with this work, due to different challenges, such as partially defined preferences and a change of benchmarking platform since 2020. In line with the competition, we allow ourselves access to the agents of previous ANAC editions (before 2017) that we use as a training set $\mathcal{P}$ for the HYDRA procedure (Algorithm 4.1). Two additional agents are added to the test set in order to compare our work to the work of Ilany and Gal [76], which adopted a similar portfolio selection method. 36 agents from the ANAC are used, split up into 20 training agents and 16 test agents.

The set of scenarios is provided in Table 4.2. A total of 42 scenarios is used, of which both sides can be played by our agent, resulting in 84 playable scenarios. The set of negotiation scenarios is selected based on diversity using the features

Table 4.1: Overview of opponent set used in this work. The last column indicates in which year the opponent participated in ANAC.

Training set		**Test set**	
Agent	**ANAC**	**Agent**	**ANAC**
ParsCat	2016	SimpleAgent	2017
YXAgent	2016	Rubick	2017
Terra	2016	PonPokoAgent	2017
MyAgent	2016	ParsCat2	2017
GrandmaAgent	2016	ShahAgent	2017
Farma	2016	Mosa	2017
Caduceus	2016	Mamenchis	2017
Atlas3201	2016	MadAgent	2017
AgentHP2_main	2016	Imitator	2017
RandomDance	2015	GeneKing	2017
PokerFace	2015	Farma17	2017
PhoenixParty	2015	CaduceusDC16	2017
ParsAgent	2015	AgentKN	2017
kawaii	2015	AgentF	2017
Atlas3	2015	MetaAgent2013	2013
AgentX	2015	MetaAgent	2012
AgentH	2015		
AgentBuyogMain	2015		
Gangster	2014		
DoNA	2014		

as described in Section 3.4, and their discount factor and reservation utility are removed. The set is split up into 56 training scenarios and 28 test scenarios. The training set is of size $|\mathcal{P}| = 20 \times 56 = 1120$ and the test set is of size $|\mathcal{P}_{test}| = 16 \times 28 = 448$.

The negotiation scenario features were calculated in advance, as described in Section 3.4. The opponent features can only be gathered by performing negotiations against the opponents. We gathered these features in advance for the first configurator run, by negotiating 10 times on every setting with a manually set strategy. After the first configurator run, opponent features are extracted based on negotiations with strategies that are already in the portfolio. Note that during training, we use the actual opponent's utility function (u_o) to calculate the features in Section 3.4 to reduce estimation noise.

HARDWARE & BUDGET

We followed Renting et al. [124] in terms of computational budget, in order to be able to compare results. Each run of SMAC was given a 1200-hour budget, divided over 300 parallel runs. Every run was performed on a single Intel® Xeon® CPU core with 2 threads and 12 GB of RAM. Running AutoFolio for our problem is not computationally expensive, so we chose not to run it in parallel for convenience. We used a single dual-core processor on the same computing cluster, assigned it 4 GB of RAM, and provided it with a budget of 0.5 hours.

Table 4.2: Overview of the negotiation scenarios sets used in this work. These scenarios are part of the GENIUS package.

Train/Test	Preference Profile 1	Preference Profile 2
train	ItexvsCypress_Cypress.xml	ItexvsCypress_Itex.xml
train	laptop_buyer_utility.xml	laptop_seller_utility.xml
train	Grocery_domain_mary.xml	Grocery_domain_sam.xml
train	Amsterdam_party1.xml	Amsterdam_party2.xml
train	camera_buyer_utility.xml	camera_seller_utility.xml
train	energy_consumer.xml	energy_distributor.xml
train	EnergySmall-A-prof1.xml	EnergySmall-A-prof2.xml
train	Barter-A-prof1.xml	Barter-A-prof2.xml
train	FlightBooking-A-prof1.xml	FlightBooking-A-prof2.xml
train	HouseKeeping-A-prof1.xml	HouseKeeping-A-prof2.xml
train	MusicCollection-A-prof1.xml	MusicCollection-A-prof2.xml
train	Outfit-A-prof1.xml	Outfit-A-prof2.xml
train	RentalHouse-A-prof1.xml	RentalHouse-A-prof2.xml
train	Supermarket-A-prof1.xml	Supermarket-A-prof2.xml
train	Animal_util1.xml	Animal_util2.xml
train	DogChoosing_util1.xml	DogChoosing_util2.xml
train	Icecream_util1.xml	Icecream_util2.xml
train	Lunch_util1.xml	Lunch_util2.xml
train	Ultimatum_util1.xml	Ultimatum_util2.xml
train	DefensiveCharms_util1.xml	DefensiveCharms_util2.xml
train	SmartEnergyGrid_util1.xml	SmartEnergyGrid_util2.xml
train	DomainAce_util1.xml	DomainAce_util2.xml
train	Smart_Grid_util1.xml	Smart_Grid_util2.xml
train	DomainTwF_util1.xml	DomainTwF_util2.xml
train	ElectricVehicle_profile1.xml	ElectricVehicle_profile2.xml
train	PEnergy_util1.xml	PEnergy_util2.xml
train	JapanTrip_util1.xml	JapanTrip_util2.xml
train	NewDomain_util1.xml	NewDomain_util2.xml
test	England.xml	Zimbabwe.xml
test	travel_chox.xml	travel_fanny.xml
test	IS_BT_Acquisition_BT_prof.xml	IS_BT_Acquisition_IS_prof.xml
test	AirportSiteSelection-A-prof1.xml	AirportSiteSelection-A-prof2.xml
test	Barbecue-A-prof1.xml	Barbecue-A-prof2.xml
test	EnergySmall-A-prof1.xml	EnergySmall-A-prof2.xml
test	FiftyFifty-A-prof1.xml	FiftyFifty-A-prof2.xml
test	Coffee_util1.xml	Coffee_util2.xml
test	Kitchen-husband.xml	Kitchen-wife.xml
test	Wholesaler-prof1.xml	Wholesaler-prof2.xml
test	triangularFight_util1.xml	triangularFight_util2.xml
test	SmartGridDomain_util1.xml	SmartGridDomain_util2.xml
test	WindFarm_util1.xml	WindFarm_util2.xml
test	KDomain_util1.xml	KDomain_util2.xml

Table 4.3: Final configurations in the portfolio. These are the final parameter settings that make up the different negotiation strategies in the portfolio.

	Accepting				**Bidding**			**Searching**					
θ	α	β	t_{acc}	γ	n_{fit}	δ	e	N_{pop}	N_{tour}	E	R_c	R_m	R_e
θ_1	1.038	0.03201	0.942	AVG^W	3	0.927	0.00199	262	6	4	0.290	0.140	0.085
θ_2	1.001	0.00166	0.935	AVG^W	3	0.998	0.06232	94	2	5	0.168	0.002	0.108
θ_3	1.007	0.01970	0.912	AVG^W	4	0.917	0.01093	305	10	1	0.107	0.063	0.184
θ_4	1.056	0.00003	0.900	MAX^W	5	0.997	0.02090	139	10	4	0.463	0.176	0.101

Table 4.4: Individual configuration performance on $\mathcal{P}$ and $\mathcal{P}_{test}$. The two left columns show the average utility of every individual strategy in the portfolio on the training and test set of negotiation settings. The next four columns show the fraction of the amount settings in the test set for which a single strategy belongs to a set of best-performing strategies.

	$M(\theta,\cdot)$		**Best performing on $\mathcal{P}_{test}$ by ratio**				
θ	$\mathcal{P}$	$\mathcal{P}_{test}$	Single best	In top 2	In top 3	In top 4	Sum
θ_1	0.815	0.742	0.281	0.100	0.016	0.123	0.520
θ_2	0.788	0.734	0.167	0.022	0.020	0.123	0.333
θ_3	0.789	0.754	0.154	0.065	0.031	0.123	0.373
θ_4	0.773	0.721	0.118	0.058	0.033	0.123	0.333

Output. The final algorithm selector was saved as a binary file at the final step of HYDRA, along with the parameter settings of every strategy configuration (Table 4.3). We use both when faced with a new negotiation setting for which we want to select a configuration.

4.6.2 RESULTS

We now present the results using a test set of negotiation instances $\mathcal{P}_{test}$. More specifically, we investigated two aspects:

1. the quality of the portfolio;

2. the performance of the algorithm selector.

QUALITY OF THE PORTFOLIO

We assessed the quality of the portfolio by measuring the performance (Equation 4.1) of every configuration in the portfolio on the training and testing sets of negotiation settings. The results can be found in Table 4.4. We included ratios that indicate how often a strategy is part of the set of best strategies per setting ("Sum" in Table 4.4). As a final quality check, the performance of the oracle selector (Equation 4.2) is evaluated for varying sizes of the portfolio. We present the results in Table 4.5.

Table 4.4 shows the results per strategy in the portfolio in the form of individual performance over a set of settings $M(\theta,\mathcal{P})$. It is evident that θ_1 is the single best strategy over the full training set $\mathcal{P}$. Furthermore, as every strategy is at least once the single best on individual settings (single best ratio > 0), we can conclude that

Table 4.5: Algorithm selector performance compared to oracle performance. The two columns on the left show the upper limit in average utility for various sizes of the portfolio on the training and test set of negotiation settings. The right two columns show the average utility obtained by applying the trained algorithm selector on every setting in both sets.

	$M(OR(\boldsymbol{\theta}, p), \cdot)$		$M(AS(\boldsymbol{\theta}, p), \cdot)$	
$\boldsymbol{\theta}$	$\mathcal{P}$	$\mathcal{P}_{test}$	$\mathcal{P}$	$\mathcal{P}_{test}$
$\{\theta_1\}$	0.815	0.742	0.815	0.742
$\{\theta_1, \theta_2\}$	0.870	0.824	0.865	0.785
$\{\theta_1, \theta_2, \theta_3\}$	0.875	0.832	0.869	0.776
$\{\theta_1, \theta_2, \theta_3, \theta_4\}$	0.879	0.840	0.868	0.784

every strategy contributes to the portfolio, thus satisfying our requirement from Section 4.3.1.

Finally, Table 4.5 shows us that, at every iteration of HYDRA, the oracle performance of the portfolio increases on both $\mathcal{P}$ and $\mathcal{P}_{test}$. The improvement decreases on $\mathcal{P}$ as the number of iterations increases, indicating that HYDRA fills the largest "weaknesses" in the portfolio first.

PERFORMANCE OF THE ALGORITHM SELECTOR

Table 4.5 shows that there is potential in the portfolio to improve the utility of $DA(\theta)$ by $\frac{0.840-0.742}{0.742} \cdot 100\% \approx 13.0\%$ on the test set, if we use the oracle selector rather than θ_1. We now replace the oracle selector with the actual selector and test its performance in two ways.

Performance against known opponents. We test the absolute performance of the algorithm selector by assuming perfect knowledge of opponent features of the opponents in the test set of negotiation setting $\mathcal{P}_{test}$. The opponent features are gathered by running 10 negotiation sessions with configuration θ_1 on the test set.

We trained and tested multiple algorithm selectors on different portfolio sizes by extending the portfolio, starting with the single best strategy θ_1. We report the performance in Table 4.5. For the oracle selector, the performance of $DA(\theta)$ increases with the size of the portfolio. However, the performance increase plateaus on $\mathcal{P}_{test}$ after adding the fourth strategy to the portfolio. Based on the results on the training set, we conclude that the fourth strategy in the portfolio is redundant and needlessly complicates the strategy selection procedure; we therefore omitted it in the final evaluation step reported in the following.

Performance with unknown opponents. Opponent features, in contrast to the scenario features, must be learned from previous encounters. Up to this point, we assumed the opponents to always be known in advance, which is not realistic. We now simulate a realistic negotiation tournament where this problem occurs. The agents in $\mathcal{P}_{test}$ can also learn from their opponents, but we cannot guarantee fair learning chances due to parallelisation. To address this issue, we negotiate once against all of them and then clean up and restart our agent, giving every opponent a head start, favouring a handicap over any advantage for our agent.

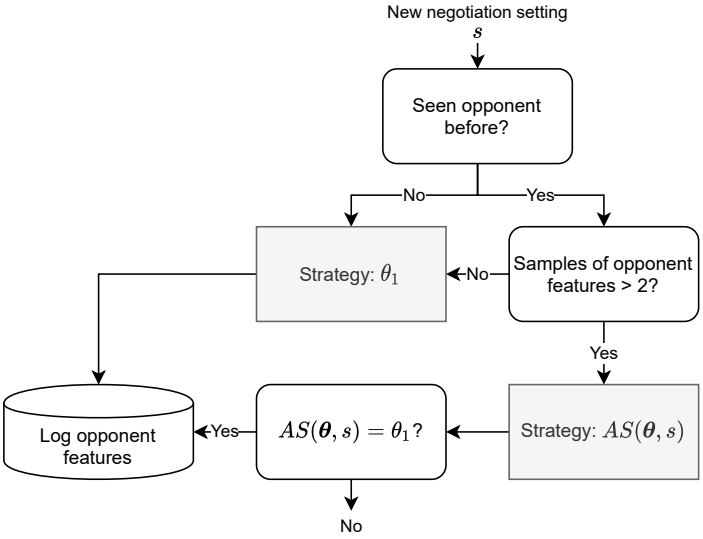


Figure 4.2: Realistic strategy selection of $DA(AS(\boldsymbol{\theta}, p))$

The question arises of what strategy to select at first encounters with opponents when no opponent features are available. If strategy selection is not possible, we select the single best strategy θ_1. Opponent features are influenced by the strategy that is selected by $DA(\theta)$, so we simplify the feature extraction process and only gather features when strategy θ_1 is selected. This aligns with the decision to select θ_1 at first opponent encounters. The coefficient of variation of an opponent feature (Section 3.4) needs at least two samples to be meaningful, so we set a second condition to select strategy θ_1 for the first two encounters with an opponent to "sample" the opponent. We illustrate this behaviour in Figure 4.2.

To obtain the results, we iterate randomly through the test settings $\mathcal{P}_{test}$ and use $DA(AS(\boldsymbol{\theta}, p))$ with $\boldsymbol{\theta} = \{\theta_1, \theta_2, \theta_3\}$ to negotiate, following the procedure as described in Figure 4.2. Additionally, we let every opponent in the test set negotiate with every other opponent in the test set on every test scenario and combine the results with the results of the $DA(\theta)$. This procedure is repeated 10 times to reduce the influence of variance for a total of 38 080 negotiations. The results averaged per agent show that we are capable of winning an ANAC-like bilateral tournament with our $DA(\theta)$ using the strategy selector, see Table 4.6. We beat the runner-up agent (MetaAgent) by $\frac{0.788-0.752}{0.752} \cdot 100\% \approx 5.6\%$ (significant at $\alpha = 0.05$ according to a one-tailed t-test p-value of $p = 0.0022$).

Finally, we compare the performances including error bars of $DA(\theta)$ with θ_1 and with a portfolio of strategies in a realistic ANAC tournament setup, see Figure 4.3. Notice that our utility improved with $\frac{0.788-0.742}{0.742} \cdot 100\% \approx 6.2\%$ by using a portfolio instead of a single fixed strategy and that the portfolio approach also improves all other performance measures.

Table 4.6: ANAC tournament results using $DA(AS(\boldsymbol{\theta}, p))$ where all scores are averaged over all negotiation instances. The goal of ANAC is to obtain the highest utility. We show the top 5 agents and all the outliers for every performance measure. Here, social welfare is the summation of utility and opponent utility, Pareto distance is the smallest distance to a Pareto efficient negotiation outcome, Nash distance is the distance to the Nash bargaining solution [111] of the scenario, and agreement ratio represents the fraction of settings that resulted in an agreement. (bold = best, underline = worst)

Agent	**Utility**	**Opponent utility**	**Social welfare**	**Pareto distance**	**Nash distance**
Imitator	0.446	**0.901**	1.347	0.091	0.428
GeneKing	0.612	0.783	1.396	0.065	0.378
Mamenchis	0.636	0.863	**1.498**	**0.016**	0.272
ParsCat2	0.642	0.773	1.414	0.090	0.273
MadAgent	0.669	0.536	1.204	0.232	0.383
Farma17	0.676	0.690	1.366	0.115	0.311
CaduceusDC16	0.688	0.599	1.287	0.181	0.327
AgentKN	0.690	0.757	1.447	0.065	**0.252**
SimpleAgent	0.699	0.531	1.230	0.204	0.398
Mosa	0.702	0.781	1.483	0.026	0.271
Rubick	0.716	0.715	1.431	0.070	0.282
PonPokoAgent	0.730	0.589	1.320	0.158	0.307
AgentF	0.738	0.679	1.417	0.076	0.301
ShahAgent	0.741	0.554	1.296	0.172	0.342
MetaAgent2013	0.746	0.659	1.405	0.092	0.284
MetaAgent	0.752	0.634	1.386	0.106	0.296
$DA(AS(\boldsymbol{\theta}, p))$	**0.788**	0.627	1.414	0.074	0.314

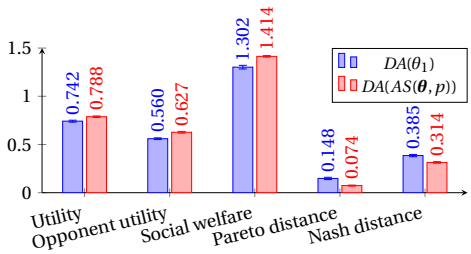


Figure 4.3: Comparison of two $DA(\theta)$ strategies in an ANAC tournament setting. Here, $DA(\theta_1)$ is comparable to the agent configured by Renting et al. [124] and $DA(AS(\boldsymbol{\theta}, p))$ represents this work. See Table 4.6 for an explanation of the measures.

4.7 CONCLUSION

In previous work [124], automatic algorithm configuration was used to obtain a single best strategy. Here, we have introduced a method to configure and use a portfolio of strategies for negotiation agents, adding a combination of HYDRA, AutoFolio, and a procedure to learn opponent behaviour. Our approach is fully automated and represents a significant step beyond the use of single best strategies in automated negotiation. In principle, it can be applied to any negotiation agent with a flexible, parameterised strategy.

We created a portfolio of 4 strategies $\boldsymbol{\theta}$ and tested the performance of every strategy on a broad set of negotiation settings. In Table 4.4, we showed that every configured strategy contributes to the portfolio by specialising on separate sets of negotiation settings. By adding algorithm selection to the Dynamic Agent to exploit differences between settings in a realistic tournament, we increased the performance of Dynamic Agent by 6.2% compared to the single best strategy and won the tournament by a margin of 5.6%. We note that the single best strategy is comparable to the agent configured by Renting et al. [124], indicating that a portfolio-based agent provides another significant boost to negotiation pay-off.

Limitations lie in the required mutual agreement on the norms of how to conduct a negotiation. In this work, a predefined protocol is used that is supported by all used agents. Agents that do not support this protocol cannot participate in the negotiation. Another important limitation is that this method has no safeguards to detect whether the strategy portfolio is still performing well and that we are not being exploited. Finally, due to the train-then-test principle of our method, we still rely on a training set that is reasonably representative of the actual application. Ethical concerns arise in the design of negotiation agents for use in real-life applications. Persons who have more resources to design quality negotiation agents can gain even more resources in the process, leading to more inequality. There are risks of exploitation, unfair play, and deception due to a lack of explainability and a high level of complexity for laypersons.

In future work, we intend to study the influence of the strategies employed by the Dynamic Agent on the opponent characteristics that we learn during negotiation to improve opponent learning. Secondly, strategy selection could be improved for first encounters with opponents, where currently, the single best strategy is selected without regard to the instance characteristics. We intend to investigate strategy selection for negotiation instances through neural networks to relax the reliance on manually designed instance features. Finally, it would be interesting to explore the use of reinforcement learning for training negotiation strategies instead of the algorithm configuration approach that we leveraged here.

5

Towards General Negotiation Strategies with End-to-End Reinforcement Learning

This chapter has been published in the Reinforcement Learning Journal (RLJ) [126].

5.1 Introduction

Traditionally, negotiating agents were manually designed algorithms based on heuristics, which is still a commonly seen approach in recent editions of the Automated Negotiation Agents Competition (ANAC) [5]. However, manually designing such negotiation strategies is time-consuming and results in highly specialised and fixed negotiation strategies that do not generalise over a broad set of negotiation settings. In later work, optimisation methods were used to optimise the parameters of negotiation strategies using evolutionary algorithms [46, 43, 92], or algorithm configuration techniques [124].

In Chapters 3 and 4, we showed that algorithm configuration and portfolio selection methods can be used to learn autonomous agents to negotiate. The proposed approaches allow negotiation strategies to be more easily adaptable to different negotiation settings. However, they still require a relatively high degree of manual design to obtain a parameterised negotiation strategy, making them time-consuming to build, limiting their generalisability by specialisation on specific negotiation settings, and inducing human bias in strategy design.

With the advent of reinforcement learning (RL) [146], there have been attempts at using RL-based methods for creating negotiation agents [19]. There is, however, still an open challenge. In automated negotiation, it is common for agents to deal with various negotiation scenarios that would cause differently sized observation and action vectors for default linear layer-based RL policies. Up until now, this issue has been dealt with by either abstracting the observations and actions into a fixed-length vector (see, e.g., Bakker et al. [19]) or by fixing the negotiation scenario, such that the observation and action space remain identical (see, e.g., Higa et al. [68]). The first approach causes information loss due to feature design, and the latter renders the obtained policy non-transferable to other negotiation scenarios.

We set out on the idea that a more general RL-based negotiation strategy capable of dealing with various negotiation scenarios is achievable and that such a strategy can be learned using end-to-end reinforcement learning without using state abstractions and without the human bias induced in the design of parameterised agents. Developing such an RL negotiation strategy would open up new avenues for RL in automated negotiation as policy networks are easily extendable. End-to-end methods might also be able to learn complex relations between observations and actions, minimising the risk of information loss that is often imposed by (partially) manual design strategies.

To this extent, we designed a graph-based representation of a negotiation scenario. We applied graph neural networks in the RL policy to deal with the changing dimensions of both the observation and action space. We show that our method performs about as well as a recent end-to-end RL-based method designed to deal only with a fixed negotiation scenario. More importantly, we show that our end-to-end method can successfully learn to negotiate with other agents and that the obtained policy also performs well on previously unseen, randomly generated linear-additive negotiation scenarios.

5.2 RELATED WORK

Bakker et al. [19] applied RL to decide what utility to demand in the next offer. They abstracted the state to utility values of the last few offers and time towards the deadline. Translating utility to an offer, estimating opponent utility, and deciding when to accept were done without RL. Bagga et al. [18] also abstracted the state into a fixed representation with utility statistics of historical offers. They used an RL policy to decide whether to accept and a separate policy that outputs offers based on a non-RL opponent utility estimation model.

Sengupta et al. [137] encoded the state into a fixed length of past utility values. The action is the target utility of the next offer, translated to an actual offer through an exhaustive search of the outcome space. They trained a portfolio of policies and tried to select effective counterstrategies by classifying the opponent type. Li et al. [97] also build a portfolio of RL-based negotiation strategies by incrementally training best responses based on the Nash bargaining solution. During evaluation, their method searches for the best response in an effort to improve cooperativity. They only applied their method to fixed negotiation scenarios.

Another line of research on negotiation agents includes natural language. An environment for this was developed by Lewis et al. [95]. Kwon et al. [90] used this environment and applied a combination of RL, supervised learning, and expert annotations (based on a dataset) to iteratively train two agents through self-play. The negotiation scenarios considered are fixed, except for the preferences.

Takahashi et al. [148] and Higa et al. [68] are closest to our work, as they also train an end-to-end RL method for negotiation games. Their approach does not use state abstractions and linearly maps the negotiation scenario and actions in a policy. The policy obtained can only be used for a fixed scenario. They trained and tested only against single opponents

Graph Neural Networks (GNNs) [86] have been used to handle graph-structured input in policy networks, for example, in molecular design [165]. Wang et al. [156] and Yang et al. [163] applied them to transfer learn over variable action spaces of various multi-joint robots. However, they aimed to speed up learning on unseen tasks, while we strive for complete transferability without additional learning.

5.3 METHODS

We formulate the negotiation game as a turn-based Partially Observable Stochastic Game (POSG), a partially observable extension of a stochastic game [139]. We model the game as a tuple $\mathcal{M} = \langle \mathcal{I}, \mathcal{S}, \mathcal{O}_i, \mathcal{A}_i, \mathcal{T}, \Omega_i, \mathcal{R}_i \rangle$, where $\mathcal{I} = \{1, \cdots, n\}$ denotes the set of agents, $\mathcal{S}$ the set of states, $\mathcal{O}_i$ the set of possible observations for agent i, and $\mathcal{A}_i$ the set of actions for agent i. For convenience, we write $\mathcal{A} = \mathcal{A}_i$, as we consider a turn-based game where the set of actions is identical for each agents. Furthermore, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto p(\mathcal{S})$ denotes the transition function, $\Omega_i : \mathcal{S} \times \mathcal{A} \mapsto p(\mathcal{O}_i)$ the observation function for agent i, and $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ the reward function for agent i.

The game starts in a particular state s. Then, at timestep t, an agent i selects an action $a_{t,i}$ independently of other agents. Based on this action, the state of the

POSG changes according to $s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)$. Subsequently, each agent receives its own observation $o_{t,i} \sim \Omega_i(o_{t,i}|s_t, a_t)$ and associated reward $r_{t,i} \sim \mathcal{R}_i(r_{t,i}|s_t, a_t)$.

Each agent i selects actions according to its own policy $\pi_i : \mathcal{O}_i \times \mathcal{O}_i \times \cdots \rightarrow p(\mathcal{A})$. At timestep t, agent i samples an action $a_t \sim \pi_i(a_t|o_{t,i}, o_{t-1,i}, \cdots)$. Note that we can vary the length of the historical observations by which we condition the policy for each agent. The more history we include, the more we can overcome partial observability.

Our goal is to find a policy π_i for agent i that maximizes cumulative expected return:

$$\pi_i^\star \in \arg\max_{\pi_i} \mathbb{E}_{\pi,\mathcal{T}} \left[\sum_{k=0}^{H-1} \mathcal{R}_i(s_{t+k}, a_{t+k}) \right], \tag{5.1}$$

where H denotes the horizon of the POSG (the number of rounds we select an action). Crucially, the performance of a particular policy π_i depends on the policies of the other agents.

5.3.1 Proximal Policy Optimisation

We will use reinforcement learning to optimize the policy π_i of our own agent i in the negotiation scenario. For simplicity, we will drop the subscript i and simply write π for the policy of our own agent. We also simplify by writing o instead of $\langle o_{t,i}, o_{t-1,i}, \cdots \rangle$. To optimize this policy, we use Proximal Policy Optimisation (PPO) [135] due to its empirical success and stability.

At each update iteration k, PPO optimises π relative to the last policy π_k by maximising the PPO clip objective:

$$\pi_{k+1} \in \arg\max_{\pi} \mathbb{E}_{o,a\sim\pi_k} \left[\min\left(\frac{\pi(a|o)}{\pi_k(a|o)} A_{\pi_k}(o,a),\ \text{clip}\left(\frac{\pi(a|o)}{\pi_k(a|o)}, 1 \pm \epsilon \right) A_{\pi_k}(o,a) \right) \right] \tag{5.2}$$

where ϵ denotes a clip parameter, and $A_\pi(a,o)$ denotes the advantage function of taking action a in observation o [146]. The ratio gets clipped to ensure that the new policy does not change too quickly from the policy at the previous step. Our PPO implementation is based on the CleanRL repository [73].

5.3.2 Graph neural networks

We aim to learn to negotiate across randomly generated scenarios where the number of objectives and values differ. This forces us to design a policy/value network where the shape and number of parameters are independent of the number of objectives and values. Networks of linear layers, often the default in RL, do not fit this criterion, as they require fixed input dimensions. We chose to represent the input of the policy network as a graph and make use of Graph Neural Networks (GNN) to deal with the changing size of the input space, more specifically, Graph Attention Networks (GAT) [152].

The input graph contains nodes that have node features. A layer of GNN encodes the features x_u of node u into a hidden representation h_u based on the features of the set of neighbour nodes $\mathcal{N}_u$ and on its own features. The specific case of GATs is defined in Equation 5.3. Here, neighbour features are encoded by a linear layer ψ

and then weighted summed through a learned attention coefficient $a(x_u, x_v)$. The weighted sum is concatenated with x_u and passed through another linear layer ϕ to obtain the embedding of the node h_u.

$$h_u = \phi\left(x_u, \sum_{v \in \mathcal{N}_u} a(x_u, x_v) \cdot \psi(x_v)\right) \tag{5.3}$$

5.3.3 IMPLEMENTATION

At each timestep, the agent receives observations that are the actions of the opponent in the negotiation game. Based on these observations, the agent must select an action. The action space combines multiple categorical actions: the accept action and an action per objective to select one of the values in that objective as an offer. If the policy outputs an accept action, then the offer action becomes irrelevant as the negotiation will be ended.

A negotiation scenario has objectives B and a set of values to decide on per objective Ω_b. We represent the structure of objectives and values as a graph and encode the history of observations $\langle o_{t,i}, o_{t-1,i}, \cdots \rangle$ of a negotiation game in this structure to a single observation o (see the left side of Figure 5.1). Each objective and value is represented by a node, where value nodes are connected to the objective node to which they belong. An additional head node is added that is connected to all objective nodes. The node features of each node are:

- 5 features for each value node: the weight $w_b(\omega_b)$ of the value, a binary value to indicate the opponent's last offer, a binary value to indicate the last offer of the agent itself, the fraction of times this value was offered by the opponent, and the fraction of times this value was offered by itself. Note that it might have been better to implement a recurrent network to condition the policy on the full history of offers instead of summary statistics. However, the added computational complexity would have rendered this work much more difficult. Our approach enables efficient learning, but future work should explore the use of the raw history of offers.

- 2 features for each objective node: the number of values in the value set of this objective $|\Omega_b|$, and the weight of this objective $w(b)$.

- 2 features for the head node: the number of objectives $|B|$, and the progress towards the deadline scaled between 0 and 1.

As illustrated in Figure 5.1, we apply GAT layers to the observation graph to propagate information through the graph and embed the node features (Equation 5.3). The size of the representation is a hyperparameter. We then take the representation of the head node and pass it to a linear layer that predicts the state value V. The head representation is also passed through a linear layer to obtain the two accept action logits. Finally, we take the representation of every value node and apply a single linear layer to obtain the offer action logits. These logits are concatenated per action and used to create the probability distribution over the action space. As we

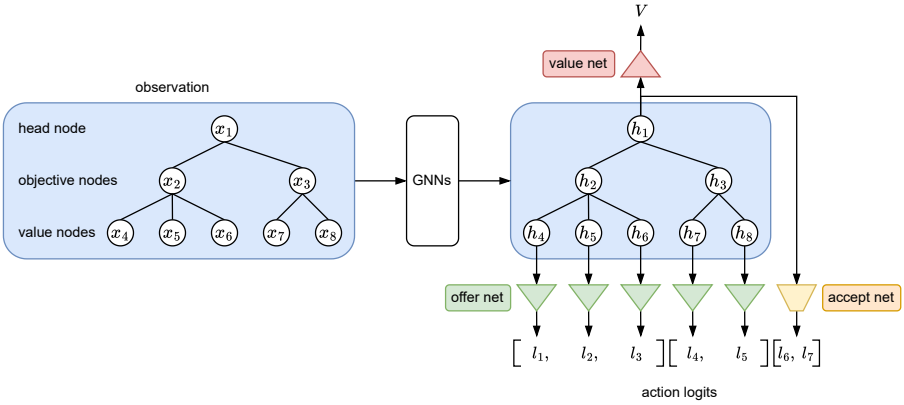


Figure 5.1: Overview of our designed policy network based on GNNs. Observations are encoded in a graph representation (left) and passed through GNNs. Action distribution logits and state value are obtained by passing the learned representation of the head node and value nodes through linear layers.

Table 5.1: Description of baseline negotiation agents used for benchmarking.

Name	Type	Description
BoulwareAgent	Time-dependent	Utility target decreases concave with time
ConcederAgent	Time-dependent	Utility target decreases convex with time
LinearAgent	Time-dependent	Utility target decreases linearly with time
RandomAgent	Random	Makes random offers, accepts any utility > 0.6

use the same linear layer for all value nodes, the number of output logits is independent of the number of parameters in the policy, thus satisfying our requirement. We also note that although the size of the outcome space suffers heavily from the curse of dimensionality when the number of objectives increases, our approach does not. Our code implementation can be found on GitHub[1].

5.4 Empirical evaluation

To train our agent, we need negotiation scenarios as well as opponents to negotiate against. The negotiation scenarios were randomly generated with an outcome space size $|\Omega|$ between 200 and 1000. As opponents, we used baseline agents and agents developed for the 2022 edition of the Automated Negotiation Agents Competition (ANAC). The baseline agents are simple negotiation strategies often used within automated negotiation to evaluate and analyse new agents. We provide a description of the opponents in Table 5.1. All agents were originally developed for the GENIUS negotiation software platform [99].

We set a negotiation deadline of 40 rounds. An opponent is randomly selected during the rollout phase, and a negotiation scenario is randomly generated. The policy is then used to negotiate until the episode ends, either by finding an agree-

[1] https://github.com/brenting/RL-negotiation/tree/RLC-2024

Table 5.2: Hyperparameter settings

Parameter	Value
total timesteps	$2 \cdot 10^6$
batch size	6000
mini batch size	300
policy update epochs	30
entropy coefficient	0.001
discount factor γ	1
value function coefficient	1
GAE λ	0.95
# GAT layers	4
# GAT attention heads	4
hidden representation size	256
Adam learning rate	$3 \cdot 10^{-4}$
Learning rate annealing	True
activation functions	ReLU

ment or reaching the deadline. The episode is added to the experience batch, which is repeated until the experience batch is full. We apply 4 layers of GATs with a hidden representation size of 256. A complete overview of the hyperparameter settings can be found in Table 5.2.

5.4.1 FIXED NEGOTIATION SCENARIO

As a first experiment, we compared our method to a recent end-to-end RL method by Higa et al. [68] that can only be used on a fixed negotiation scenario. Their method was originally only trained and evaluated against single opponents. We chose to train the agent against the set of baseline players instead, as we consider that a more realistic scenario. The baseline agents show relatively similar behaviour, making this an acceptable increase in difficulty.

We generated a single negotiation scenario and trained a reproduction of their and our own method for 2 000 000 timesteps on 10 different seeds. The training curve is illustrated in Figure 5.2, where we plot both the mean of the episodic return and the 99% confidence interval based on the results from 10 training sessions. Every obtained policy is evaluated in 1000 negotiation games against every opponent on this fixed negotiation scenario. We report the average obtained utility of the trained policy and the opponent, including a confidence interval based on the 10 evaluation runs in Figure 5.3.

We can see in Figure 5.3 that our method performs similarly to the method proposed by Higa et al. [68]. This result is mostly a sanity check that our method can successfully learn to negotiate in a relatively simple setup despite being more complex and broadly usable.

5.4.2 RANDOM NEGOTIATION SCENARIOS

We now evaluate the performance of our end-to-end method on randomly generated negotiation scenarios. Negotiation scenarios will continuously change during both training and evaluation.

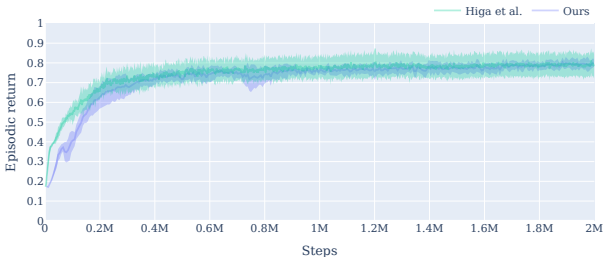


Figure 5.2: Mean and 99% confidence interval of episodic return during training based on results from 10 random seeds . The results of the policy designed by Higa et al. [68] and our policy are plotted.

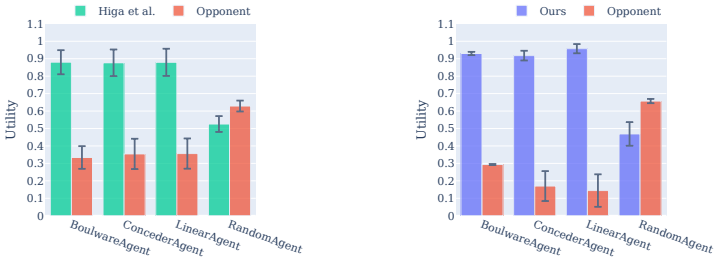


Figure 5.3: Evaluation results of the policy designed by Higa et al. [68] and our GNN-based policy. Results are obtained by evaluating each trained policy for 1000 negotiation games against the set of baseline agents. Mean and 99% confidence interval are plotted based on 10 training iterations.

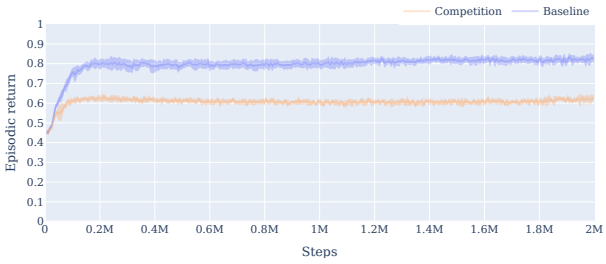
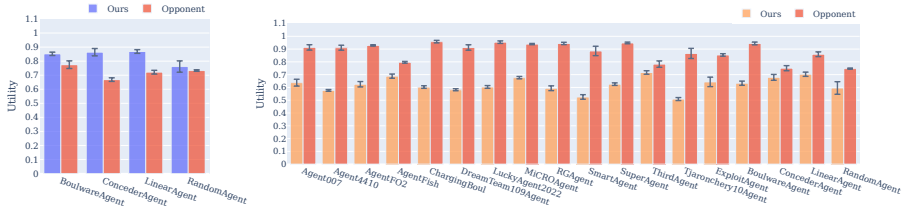


Figure 5.4: Mean and 99% confidence interval of episodic return during training of our GNN policy based on results from 10 different random seeds. The results from training against the baseline agents and training against the competition agents are plotted.



(a) (b)

Figure 5.5: Evaluation results of our GNN-based policy on randomly generated negotiation scenarios both against the set of baseline opponents (left) and against the full set of opponents (right). Results are obtained by evaluating each trained policy for 1000 negotiation games against the set of agents. Mean and 99% confidence interval are plotted based on 10 training iterations.

BASELINE OPPONENTS

We first train and evaluate against the set of baseline agents as described in Table 5.1. We train our method for 2 000 000 steps on 10 random seeds. The learning curve is plotted in Figure 5.4. Results are again obtained by running 1000 negotiation sessions against the set of baseline opponents, but this time, all negotiation scenarios have been randomly generated and were never seen before. We note that the observation and action space sizes are constantly changing. Results are plotted in Figure 5.5a.

As seen in Figure 5.5a, our method performs well against all baseline agents while negotiating on various structured negotiation scenarios it has never seen before. It is promising that an end-to-end learned GNN-based policy appears to generalise over such different scenarios.

COMPETITION OPPONENTS

We now repeat the experiments, but increase the set of agents we negotiate against. More specifically, we add the agents of the 2022 edition of the Automated Negotiation Agents Competition (ANAC)[2]. The learning curve and results are plotted in Figure 5.4 and Figure 5.5b, respectively.

The results show much lower performance against all opponents, including those previously outperformed. Our current method of encoding the observations and design of the policy likely leads to limited capabilities of learning opponent characteristics. Past work has shown that adapting to opponents is important to improve performance [76, 137, 123], which is currently impossible. However, this goes beyond the core contribution of this work, which is about handling different-sized negotiation scenarios in end-to-end RL methods. We discuss potential solutions in Section 5.5.

5.5 CONCLUSION

We developed an end-to-end RL method for training negotiation agents capable of handling differently structured negotiation scenarios. We showed that our method

[2] https://web.tuat.ac.jp/~katfuji/ANAC2022/

performs as well as a recent end-to-end method that is not transferable beyond a single fixed negotiation scenario. We see the latter as a restriction since, in real-world applications, it would be unlikely to encounter the exact same negotiation scenario more than once.

In this chapter, we have demonstrated how the difficulty of dealing with changing negotiation scenarios in end-to-end RL methods can be overcome. Specifically, we have shown how an agent can learn to negotiate on diverse negotiation scenarios in such a way that performance generalises to never-before-seen negotiation scenarios. Our method is conceptually simple compared to previous work on reinforcement learning in negotiation agents. Our agent performs well against strong baseline negotiation strategies, but leaves room for improvement when negotiating against a broad set of highly competitive agents.

Our approach is based on encoding the stream of observations received by our agent into a graph whose node features are designed to capture historical statistics about the episode. This manual feature design likely leads to information loss and goes against the end-to-end aim of our approach. For example, the expressiveness of history is limited, as the graph only encodes the last offer and frequency of offers. This likely also causes limited adaptivity to a broad set of opponent strategies, which in turn may well cause the low performance observed in Figure 5.4.2.

We note that, due to the competition setup of ANAC, competitive agents often play a game of chicken. Performing well against such strategies means that a policy must also learn this game of chicken. This can be challenging for RL, due to exploration problems, as it must learn a long sequence of relatively meaningless actions before having a chance to select a good action. We could attempt to improve upon this, but it might be more beneficial to prioritize mitigating this game of chicken behaviour, as it is inefficient and (arguably) undesirable.

The negotiation scenarios we generated have additive utility functions and outcome spaces that are comparable in size and competitiveness to the benchmarks used in the ANAC competition. Real-world negotiation scenarios, however, can have huge outcome spaces [82]. Our designed policy can be applied to larger scenarios without increasing the trainable parameters, and the effects on the performance of doing this should be investigated in future work.

Further promising avenues for future work include extending end-to-end policies with additional components that, e.g., learn opponent representations based on the history of observations in the current or previous encounter. Changing a negotiation strategy based on the opponent characteristics has been shown previously to improve performance [76, 137, 123], but is likely difficult to learn through our current policy design. Furthermore, improving our method to handle continuous objectives would eliminate the necessity of discretizing them.

Overall, we believe that the work in this chapter is a substantial step towards the effective use of end-to-end RL for the challenging and important problem of training negotiation agents whose performance generalises to new negotiation scenarios and opens numerous exciting avenues for future research in this area.

II

Evaluating Learning Negotiation Agents

5

6

Analysis of Learning Agents in Automated Negotiation

This chapter is under review at the time of writing.

6.1 Introduction

The Automated Negotiating Agents Competition (ANAC) was first organised in 2010 to support the development and benchmarking of automated negotiating agents [10]. Since 2017, ANAC has been extended with a number of additional leagues that each focus on a more specialised challenge, such as the game of Diplomacy [81], supply chain environments [6], the game of Werewolves [6] or negotiations between agents and humans [106]. The main league, which focuses on more classical agent-based negotiations, has since then been called Automated Negotiation League (ANL). In ANL, the participating agents are designed to bargain with other agents over a collective agreement in scenarios with conflicting interests.

Over the years, the main league of ANAC has evolved to incorporate new challenges, such as multi-lateral negotiation, preference elicitation and large intractable solution spaces. In most earlier editions, the negotiations were considered largely single-shot sessions, in which the agents would be re-initialised for every new negotiation, making it impossible for them to use any knowledge from previous interactions. However, in some future applications of negotiating agents like the ones provided before, it is imaginable that agents would encounter opponents multiple times, making the negotiation a repeated game. In such scenarios, it is also realistic that agents would use information from previous encounters in order to optimise their performance. This adds a learning dynamic between negotiation sessions to the negotiating agents. Agents generally also learn within a single negotiation session (e.g., for opponent preference estimation), but in this work, we exclusively mean agents that learn over multiple negotiation sessions when we refer to "learning agents". We intended to study such learning behaviour further using the ANL.

It is beneficial for a negotiating agent to implement a measure of adaptivity to the environment in which it carries out negotiation. Negotiation strategies can be adapted depending on the characteristics of the negotiation scenario and the opponent. For example, a negotiation scenario in which the preferences of agents are strongly conflicting might require an agent to behave differently than a scenario in which preferences are largely overlapping. Also, if an opponent drives a hard bargain, it might not be smart to adopt a cooperative strategy, as this risks being extorted. We have seen agents that successfully adapt to negotiation scenarios [76] and opponents [137, 123], but not yet in environments where opponents are also learning.

We set the challenge of the 2021 and 2022 editions of ANL with the goal to study learning negotiating agents better. The challenge was to improve performance by learning and adapting to the behaviour of the other agents submitted to ANL. This article provides an overview of learning agents in the history of ANL in general and the submissions and results of the 2022 edition of ANL, held in conjunction with the International Joint Conference on Artificial Intelligence (IJCAI) 2022, specifically. We consider the competition and its design part of the novelty of this work, which we now extend with a thorough analysis. We aim to answer the following questions:

1. Can we design a negotiation competition where participants manage to submit strategies including learning mechanisms?

2. Given that the negotiation games are general sum, do agents that perform well in the social welfare performance criteria also perform well in terms of individual utility and vice versa?

3. Do learning strategies benefit negotiation agents?

4. What is the effect of the negotiation scenario generator on the performance of the agents?

5. Does the standard approach of averaging performance, used in earlier editions of ANL, provide a robust ranking of agents?

We have analysed to which extent the agents are sensitive to the characteristics of the given negotiation scenarios. We observed that agents perform noticeably better in scenarios with strong mutually beneficial outcomes or a high variance of utility over the outcomes for both agents. Furthermore, we have analysed the results in depth and explored to what extent the learning algorithms positively affected the agents' negotiation performance.

We draw three main conclusions. Firstly, we conclude that agents that apply learning techniques clearly outperform those that do not, which shows that learning can improve the performance of a negotiating agent. Secondly, however, we also observe that a naïve strategy that does not learn at all outperforms all other agents when we look at the results from a game-theoretical perspective, forming an empirical Nash equilibrium. Finally, we conclude that the current approach of ranking agents through average scores is not sufficiently robust and that there is no clear alternative to ranking the agents. We hope this work serves as a useful starting point of this last issue within the automated negotiation agents community.

6.2 RELATED WORK

6.2.1 THE AUTOMATED NEGOTIATING AGENTS COMPETITION

The annual Automated Negotiating Agents Competition (ANAC) was first organized in 2010. The first three editions of ANAC were focused on the simplest scenario only, in which two agents negotiate with each other over a domain with linear utility functions [16]. In these tournaments, each negotiation session was completely independent of previous sessions, so the agents were not allowed to learn from previous encounters. This changed in 2013 when the option was added for agents to store information between sessions and, hence, to learn and evolve over the course of the tournament [1], but opponents were anonymous. In 2014, this option was removed again, and the focus shifted to very large domains, where the number of possible deals was of the order 10^{30} and in which the utility functions were non-linear [58]. From 2015 onward, the competition returned to smaller domains and linear utility but focused on multi-lateral negotiations involving more than two agents at a time [57]. In the 2017 and 2018 editions, for the second time, the opportunity was provided to the agents to maintain an internal state and to learn from previous encounters with opponents. However, this was limited to a single negotiation setting, which was repeated six times with the same opponent and

scenario. In 2019 and 2020, the focus was on bilateral negotiations again, but this time with *partially* known utility functions simulating an agent estimating human preferences [6]. Then, in 2021, the preferences of the agents were again represented by linear utility functions and the option to learn from previous negotiation sessions was re-introduced, similar to the setting adopted in 2013. The difference between the 2021 and 2022 editions and the 2013 edition is that in 2013, opponents were anonymous; this means that they were not able to adapt to specific opponents.

6.2.2 Learning agents in automated negotiation

As mentioned previously, there are multiple opportunities for learning in the context of automated negotiation. Within a single negotiation, the stream of proposals received from the opponent contains information about the preferences and tactics of opponents [12]. In repeated encounters, agreements and observations of previous encounters with the opponent can also be used to reason about the opponent's tactics. It is important to note that learning and adapting to opponents can benefit all agents involved in a negotiation rather than merely improve individual performance. Specifically, adapting to opponents can improve the chances of reaching an agreement and finding mutually beneficial (i.e., Pareto efficient) outcomes in settings where preferences are partially aligned.

Another option is offline learning, where the performance of an agent is optimised in a controlled environment through training on a given set of agents and negotiation scenarios. A distinction can be made in the way these agents are trained. Some take an approach where the behaviour of the agent is parameterised, and these parameters are optimised either through reinforcement learning (see Chapter 5) or other algorithmic optimisers (see Chapter 3). Others take an algorithm selection approach as we discussed in Chapter 4.

6.2.3 Learning agents in ANL

Many agents in the history of ANL have implemented a form of preference estimation, which attempts to learn opponent preferences. Accurate preference estimation helps find mutually beneficial outcomes and thus potentially improves performance. The simple frequency model that was part of the SmithAgent [59] submitted in the 2010 edition of ANAC is often used; this model estimates preferences based on the frequency an opponent has offered an outcome. Besides frequency models, Bayesian models based on Bayesian inference are also commonly seen [69, 159]. Baarslag et al. [11] created an overview of preference estimation methods that have been applied in ANL and demonstrated that frequency models show better performance than Bayesian models. As frequency models are also performant and conceptually easy, most agents in the ANL competition implement a frequency model.

Aside from learning opponent preferences, learning opponent strategy can also help improve performance. However, few agents in the past of ANL have implemented such a mechanism, with one notable exception. In the 2012 edition, an agent adopted an algorithm selection approach using previously submitted agents called the MetaAgent [77]. An offline-trained classifier was then used to

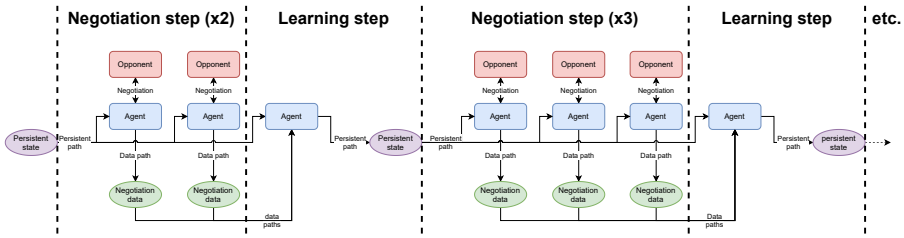


Figure 6.1: ANL tournament flow for the 2021 edition. The learning and negotiating phases were strictly separated. As agents were killed after execution, strategy-specific settings were stored in a persistent state that was fed to the agent at initialisation. During the negotiation phase, agents were allowed to store observation data, but not to update their strategy settings. Learning from this data and modifying the strategy settings of the agent was only allowed during the learning phases.

select an agent based on manually designed features of the negotiation scenario and the opponent's first few observations. This agent was also submitted to ANAC 2013.

6.3 ANL 2021

The ANL is intended to stimulate the advance of research in automated negotiation. Every year, a specific topic on the research agenda is chosen as the basis for the challenge. As mentioned before, we challenged the participants to learn in repeated negotiation games, where we tried to restrict the agents as little as possible in their learning methods. The 2021 and 2022 editions of ANL were organised around this topic.

In the 2021 edition, we decided to use a fixed set of negotiation scenarios such that every agent would encounter every other agent on the exact same set. This makes the competition fair in the sense that the impact on agent performance of using a different set of scenarios per agent is eliminated. We designed a complex data-saving structure that participants could use for learning purposes. The complexity was required to prevent potential unfair play caused by repeated use of the same negotiation scenarios.

All submitted agents would negotiate against each other on identical negotiation scenarios concurrently. Without restrictions, this could lead to unfair play by agents saving data on the negotiation scenario that they would face once more against another agent. The designed competition flow provides us control of the location where agents could save their data. Only after all negotiation sessions on the same scenario finished, the agents were given access to their data and a chance to use it for learning and changing its behaviour. This mechanism made the agents blind to their history until we allowed them to access it and thus prevented unfair play; it is visualised in Figure 6.1.

In retrospect, this structure added too much complexity to the competition for participants, causing a lower-than-expected number of submissions. Moreover, only few of the submissions implemented a learning mechanism, failing the goal of the competition. The ANL 2021 edition received 8 submissions, of which 2 imple-

mented a learning mechanism.[1] This prevented us from performing a meaningful analysis of such learning negotiation agents. Our main insight from ANL 2021 was to keep future editions as simple as possible from the perspective of participants.

In the 2022 ANL edition, we simplified the rules by allowing agents to save and load data files in a specifically provided directory without any restrictions. This resulted in more participants. Fair play was ensured by never repeating negotiation scenarios, while the number of negotiation rounds played was massively scaled up to minimise the stochastic impact caused by the randomly generated negotiation scenarios. We also moved from Java to Python as the default implementation language in order to allow for the use of the plethora of machine learning packages available in Python.

6.4 COMPETITION SETUP OF ANL 2022

Participants of ANL must design and submit an agent that can perform bilateral negotiation with other submitted agents following a finite-horizon Alternating Offers Protocol (AOP) [132]. We used GeniusWeb[2] as a platform for the negotiations, which is a software package that was specifically designed as a test-bed for agent-based negotiation.

This section describes the competition setup and the specifics of the negotiation games that are played between agents.

6.4.1 NEGOTIATION SCENARIO

The negotiation scenarios used in this competition follow the format described in Section 2.1. The utility functions are considered private information to the agents, making the negotiation an imperfect information game and exclusively categorical issues are used.

RANDOM GENERATION

In editions of ANL prior to 2021, the negotiation scenarios used to be manually designed. However, the challenge we set this year includes never repeating a negotiation scenario, which requires many such scenarios. The negotiation domains and utility functions in ANAC 2022 were randomly generated to accommodate this.

Outcome space To generate the negotiation scenarios, a goal outcome space size between 200 and 10 000 is sampled uniformly at random. The number of issues between 4 and 10 is also randomly sampled uniformly. Finally, the number of values per issue must be set so that the product of the number of values per issue is close to the goal size of the outcome space. This is done by distributing the number of values per issue according to a Dirichlet distribution, which creates a vector of random values summing up to 1. The probability density function of the Dirichlet distribution is

$$f(x_1, \cdots, x_n \mid \alpha_1, \cdots, \alpha_n) = \frac{1}{C} \cdot \prod_{i=1}^{n} x_i^{\alpha_i - 1} \tag{6.1}$$

[1] https://tracinsy.ewi.tudelft.nl/pubtrac/GeniusWebThirdParties
[2] https://ii.tudelft.nl/GeniusWeb/

where C is a normalising constant. We set all parameters α_i in this distribution to 1, i.e. $(\alpha_1, \cdots, \alpha_n) = \mathbf{1}_n$ such that the individual values are sampled from a uniform distribution. The full generation method is provided as pseudocode in Algorithm 6.1.

Algorithm 6.1 Outcome space generation

$g \leftarrow$ random integer $\in [200, 10\,000]$
while true **do**
  $m \leftarrow$ random integer $\in [4, 10]$
  $\boldsymbol{x} \leftarrow Dirichlet(\mathbf{1}_m)$
  $\boldsymbol{x} \leftarrow \boldsymbol{x} \cdot \left(\frac{g}{\prod_{b=1}^{m} x_b}\right)^{\frac{1}{m}}$ ▷ After 5: $\prod_{b=1}^{m} x_b = g$
  for $b \leftarrow 1$ to m **do**
    $|\Omega_b| \leftarrow \max\{round(x_b), 2\}$
  if $\prod_{b=1}^{m} |\Omega_b| - g < 0.1 \cdot g$ **then**
    break
$\Omega_1, \cdots, \Omega_m \leftarrow create\_values(\{|\Omega_1|, \cdots, |\Omega_m|\})$
$\Omega \leftarrow \{\Omega_1 \times \cdots \times \Omega_m\}$
return Ω

Utility functions Only bilateral negotiations are considered in this competition, so two utility functions that express preferences over the outcome space must be generated. The utility is obtained through a linear weighted sum of the values per issue, with weight factors $w(b)$ for every issue. These weights are again sampled from a Dirichlet distribution parameterized by $(\alpha_1, \cdots, \alpha_m) = \mathbf{1}_m$. Finally, for each issue b, the scores of the values ω_b within that issue are also sampled from a Dirichlet distribution and scaled to the range $[0, 1]$. These scores are expressed through the value weight function $w_b(\omega_b)$.

6.4.2 ANL 2022 CHALLENGE

In 2022, all submitted agents repeatedly negotiated against each other in one-on-one negotiation sessions with a deadline of 60 seconds in wall clock time to ensure a finite horizon. Failing to reach an agreement resulted in 0 utility for both agents involved in the negotiation. The negotiation scenarios were randomly generated and are likely always different in terms of size, number of issues, and utility functions.

The challenge in 2022 was to learn from previous encounters with other agents. The name of the opponent was made known to the agent. Agents were allowed to save any data files in a provided directory while encountering every other submitted agent 50 times throughout the tournament. One challenging part was effectively using information extracted from previous encounters with the same opponents, while the negotiation scenarios changed between each negotiation session.

6.4.3 EVALUATION

Agents were ranked based on two performance measures: individual utility and social welfare, both averaged over all negotiation sessions. Social welfare is the sum

Table 6.1: Computing hardware and resources per negotiation session.

Description	Type	Quantity
CPU	Intel® Xeon® CPU E5-2620 v4	2 cores
Memory	RDIMM DDR4-2400	10GB
OS	CentOS 7.9.2009	-

of utilities obtained by both agents involved in the negotiation session and is thus identical for both agents. Prize money was to be awarded to the two best-performing agents according to each of these performance measures. This resulted in multiple optimisation criteria for the participants of this competition. Maximising individual utility is selfish, and maximising social welfare could be considered social.

6.4.4 SIMULATION SPECIFICS

Every submitted agent encountered every other agent 50 times sequentially. For each negotiation session, a new negotiation scenario was generated randomly (see Section 6.4.1). As preferences over the negotiation domains can be unbalanced and might favour one of the agents, we decided to repeat the full tournament once more while switching the utility functions. The storage directory of every agent was completely erased when we restarted the tournament with switched utility functions to rule out the possibility of foul play. To further reduce the stochastic influence in the results, we repeated the previously mentioned procedure 5 times for the competition and 25 times for the analysis provided later in this article. The 19 submissions received required us to run a total of $19 \cdot 18 \cdot 50 \cdot 5 = 85\,500$ and $19 \cdot 18 \cdot 50 \cdot 25 = 427\,500$ negotiation sessions, respectively.

The sessions were run parallelized on a compute node. Details of the hardware and resources per session can be found in Table 6.1. The speed of the system is important as negotiation sessions are run with a wall-clock deadline. We ensured that no agent would ever face the same opponent concurrently so that the sequential encounter requirement of our challenge was satisfied. The participants were notified of potential file race issues due to parallel negotiation sessions and were suggested to save files based on the name of their current opponent to avoid this.

6.5 SUBMISSIONS TO ANL 2022

An overview of the submissions is provided in Table 6.2. The competition received a total of 20 submissions, of which 1 was invalid, resulting in a total of 19 agents that participated in the competition. The code of these agents can be found on the GeniusWeb webpage[3].

6.5.1 LEARNING CAPABILITIES

As mentioned earlier, the challenge was designed to encourage participants to develop learning methods implemented by providing the agents with a directory

[3] https://tracinsy.ewi.tudelft.nl/pubtrac/GeniusWebThirdParties

Table 6.2: Overview of agents that were submitted to ANL 2022

Name	Affiliation	Learning
Agent007	Bar Ilan University	
Agent4410	College of Management Academic Studies	
AgentFish	Tokyo University of Agriculture and Technology	
AgentFO2	Tokyo University of Agriculture and Technology	x
BIUagent	Bar Ilan University	
ChargingBoul	University of Tulsa	x
CompromisingAgent	Bar Ilan University	x
DreamTeam109Agent	College of Management Academic Studies	x
GEAAgent	College of Management Academic Studies	
LearningAgent	Bar Ilan University	x
LuckyAgent2022	Babol Noshirvani University of Technology	x
MiCROAgent	IIIA-CSIC	
PinarAgent	Siemens	x
ProcrastinAgent	University of Tulsa	x
RGAgent	Bar Ilan University	
SmartAgent	College of Management Academic Studies	x
SuperAgent	Bar Ilan University	x
ThirdAgent	College of Management Academic Studies	
Tjaronchery10Agent	College of Management Academic Studies	x

to save and load data. When we refer to learning, we mean changing behaviour between sessions based on previously recorded information. Preference estimation of an opponent within a negotiation session could also be considered learning, but we do not refer to it as such in this article. As opponents are repeatedly encountered during the competition, observations about their past behaviour could be exploited to improve negotiation capabilities. However, not all submitted agents implemented such a mechanism, making their strategies single-shot-based. Table 6.2 indicates which agents implemented a learning mechanism using the storage location to save data. As can be seen, more than half of the agents actually implemented a learning mechanism. We were successful in designing a competition that enables participants to actually implement such a mechanism. The effects of the implemented learning mechanisms are studied in more detail in Table 6.6.2.

6.5.2 SUBMITTED AGENT STRATEGIES

This section describes the strategies of selected agents: AgentFO2, DreamTeam109Agent, SuperAgent, Tjaronchery10Agent, and MiCROAgent. In general, the behaviour of the submitted agents can be considered a black box due to heavy manual design and parameter tuning. These agents were selected because they implemented an intuitively describable mechanism, and the respective participants submitted a report with their agent code. We summarize the main components based on these reports, which can be found in the repository of submitted agents[4].

[4] https://tracinsy.ewi.tudelft.nl/pubtrac/GeniusWebThirdParties

AgentFO2 This agent tries to reason over the opponent based on the Hamming distance between offers that it received. It classifies opponents as time-dependent converters, i.e., agents that concede based on the time towards the deadline, random agents, or other strategies. It then applies a time-dependent strategy while changing the minimum utility goal depending on the classified opponent strategy. This minimal utility to aim for is based on historical observations of the opponent.

DreamTeam109Agent This agent focuses on obtaining high utility first, and if that does not work, tries to minimize negotiation sessions that end in no agreement. It keeps track of the speed of the opponent in the past and tries to estimate how many rounds still can be played. If it is likely that this is the last round, then it simply accepts the offer. It also maintains a percentage of top outcomes it accepts per opponent and increases this percentage if past sessions result in low utility. The reasoning is that a low utility could result from the agent accepting a bad offer and that utility could be improved if it was more lenient towards the opponent in accepting offers with higher utility but were out of the top percentage pool.

SuperAgent This agent splits the negotiation session into timeslots and saves the average self-utility and average estimated opponent utility of all received offers in this timeslot for future use. It uses these values as utility thresholds for generating offers in the corresponding timeslot by demanding the average obtained utility as a minimum threshold and making offers above the opponent's threshold at the end of the session. The friendly behaviour at the end of the negotiation session is randomized to prevent opponents from exploiting it.

Tjaronchery10Agent This agent also adopts a time-dependent conceding strategy but does not concede in the first few encounters with an opponent. It attempts to force opponents to accept bad offers from them by being a hardliner. However, if this strategy does not appear fruitful after 3 sessions with an opponent, the strategy towards this opponent is modified to be slightly more conceding. These modifications can be repeated.

MiCROAgent MiCROAgent is an implementation of the recently introduced MiCRO strategy [40]. It is a very simple strategy that employs no form of learning or opponent modelling. It sorts all possible outcomes in order of decreasing individual utility and then proposes them in this order as long as the opponent also keeps making new proposals. That is, whenever the opponent makes a *new* proposal, MiCRO replies by proposing the next offer from its list. Whenever the opponent repeats an offer it has already made before, MiCRO replies by also proposing a (random) offer it has already proposed before. MiCRO accepts an offer from the opponent when that offer is better than or equal to the next offer that MiCRO will make. The idea is that it is a tit-for-tat-like strategy that assumes no knowledge about the utility function of the opponent. The agent always makes the smallest possible concession whenever it notices that the opponent is making a concession,

Table 6.3: Top 5 agents of the submitted agents to the ANL. Both prize categories are displayed. Here, individual utility is the average individual utility that agents obtained over all their negotiation sessions, and social welfare is the sum of the utilities of agents averaged over all negotiation sessions. Note that these are the original competition results, which differ from the results in the paper for reasons described in Section 6.6.1.

Agent	Individual Utility	Rank
DreamTeam109Agent	0.7247	1st
ChargingBoul	0.7238	2nd
SuperAgent	0.7040	3rd
CompromisingAgent	0.6857	4th
RGAgent	0.6819	5th

Agent	Social Welfare	Rank
DreamTeam109Agent	1.4605	1st
Agent007	1.4564	2nd
CompromisingAgent	1.4563	3rd
AgentFish	1.4396	4th
Agent4410	1.3993	5th

regardless of the magnitude of that concession. A negotiation between two such agents guarantees a Pareto-efficient agreement.

6.6 RESULTS & ANALYSIS

In this section, the agents are thoroughly empirically evaluated using multiple approaches. The 2021 edition results of ANL showed that agent scores depend on the other submitted agents, which we explore further. We answer the question of the influence of the learning mechanism on the performance of the negotiating agents, as this was the ANL challenge of the competition. Finally, we perform a game theoretical analysis of the competition and see how that relates to the official results of the competition.

6.6.1 DIFFERENCES TO ACTUAL COMPETITION

The results presented in this section are not fully in line with the actual results of the competition. The "LuckyAgent2022" was underperforming during the competition because of bugs. We allowed a resubmission of this agent to be included in this article. As this also affects the performance of other agents, the ranking of agents differs slightly compared to the official ranking presented after the competition. Finally, the entire competition was rerun to gather additional results that we use in our analysis presented in the following. The actual competition winners can be found in Table 6.3.

6.6.2 TOURNAMENT RESULTS

The top part of Figure 6.2 shows the tournament results. The agents are sorted based on the average utility obtained during the tournament, where the leftmost agent is the best-performing agent. A more elaborate results table is provided in Table 6.4. Notice that there are no large differences between the individual utility scores, as the difference between the maximum and minimum scores is only 0.2. The difference in social welfare score is much more apparent, with a maximum difference of nearly 0.6. We emphasise that most of the top-performing agents implemented a learning mechanism.

The results also generally show a higher social welfare score for agents with higher individual utility. Still, it is not evidently true that a higher individual utility

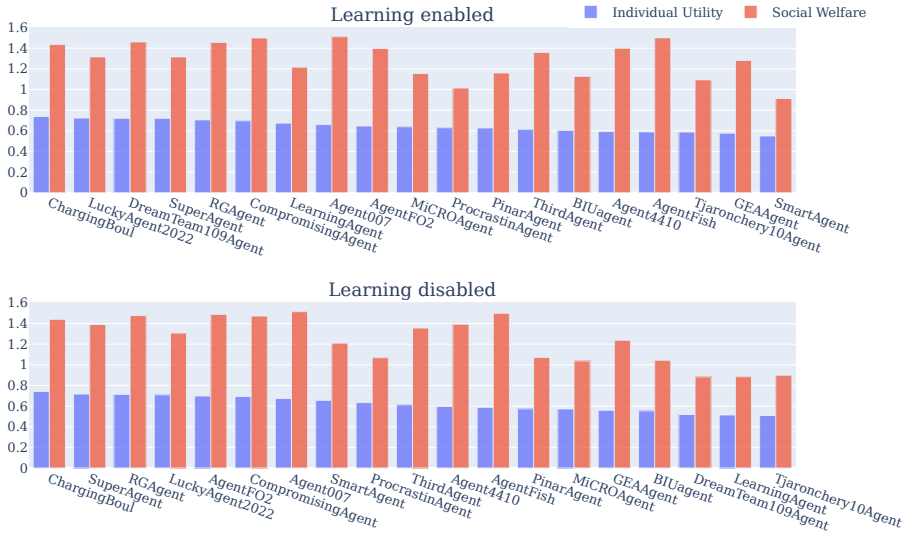


Figure 6.2: Results of two tournaments where learning is **enabled** (top) and learning is **disabled** (bottom). Both individual utility and social welfare are presented and agents are sorted from highest individual utility to lowest.

also leads to higher social welfare. Table 6.4 shows that the top-performing agent in utility fails to reach an agreement in more than 10% of the negotiation session. This clearly indicates wasted potential, as finding an agreement will always result in a higher utility than finding no agreement. On the contrary, the two highest-scoring agents in social welfare have a near 100% success rate in finding agreements yet do not obtain the highest utility. Being more selfish apparently leads to a higher utility at the cost of a lower agreement ratio. Paradoxically, a lower agreement ratio also, in turn, leads to lower utility. In terms of our research question, our empirical results indicate that agents that perform well in utility also perform well in social welfare, but that top performance in one of the categories tends to correlate with a lower score in the other.

IMPACT OF LEARNING

To determine to what extent the ability to learn influences the performance of the agents, we ran another experiment in which the agents' learning capability was disabled. This is achieved by emptying the storage directory of the agents after every negotiation, returning them to their initial state. The agents have no knowledge about previous negotiation sessions when initialised for all negotiation sessions. The results are found in the lower part of Figure 6.2.

The utility and ranking of every agent are different compared to the tournament where learning is enabled. This is more apparent for some agents, e.g. DreamTeam109Agent. We visualise this difference in ranking between a tournament where learning is enabled and a tournament where learning is disabled in Figure 6.3. We see that the top four highest-ranking agents when learning is enabled

Table 6.4: More results from the Automated Negotiation League (ANL) where **bold** is best and underline is worst. The results are averaged over all the negotiation sessions that each agent participated in. Distances are calculated by taking the Euclidian distance between the utilities of two outcomes. The Pareto front is the set of outcomes that are not strictly dominated by other outcomes in terms of utility. The Nash bargaining solution [111] is the outcome that maximizes the product of utilities. The Kalai-Smorodinsky bargaining solution [83] is the outcome on the Pareto front that is closest to equal utility.

Agent	Learning	**Individual Utility**	**Opponent Utility**	**Nash Product**	**Social Welfare**	**Number of Offers**	**Distance to Pareto Front**	**Distance to Nash Bargaining Solution**	**Distance to Kalai-Smorodinsky Bargaining Solution**	**Distance to Max Social Welfare**	**Agreement Ratio**
ChargingBoul	x	**0.739**	0.696	0.570	1.435	4286	0.113	0.275	0.261	0.282	0.891
LuckyAgent2022	x	0.724	0.593	0.522	1.317	2500	0.186	0.357	0.342	0.362	0.814
DreamTeam109Agent	x	0.721	0.740	0.540	1.461	**4733**	0.069	0.306	0.294	0.311	0.945
SuperAgent	x	0.721	0.595	0.522	1.316	4669	0.185	0.361	0.347	0.365	0.812
RGAgent		0.707	0.750	**0.593**	1.457	1036	0.111	**0.246**	**0.240**	**0.252**	0.886
CompromisingAgent	x	0.698	0.802	0.567	1.500	997	0.051	0.277	0.271	0.282	0.953
LearningAgent	x	0.675	0.542	0.487	1.217	1026	0.248	0.425	0.410	0.429	0.741
Agent007		0.660	0.851	0.549	**1.511**	2144	**0.036**	0.280	0.272	0.285	0.990
AgentFO2	x	0.647	0.751	0.553	1.398	2448	0.137	0.298	0.282	0.305	0.873
MiCROAgent		0.640	0.515	0.464	1.155	4620	0.291	0.451	0.426	0.458	0.709
ProcrastinAgent	x	0.630	0.385	0.364	1.015	3792	0.336	0.598	0.576	0.601	0.658
Pinar_Agent	x	0.628	0.532	0.467	1.161	3865	0.292	0.438	0.407	0.446	0.718
ThirdAgent		0.615	0.744	0.530	1.359	915	0.154	0.328	0.310	0.335	0.858
BIU_agent		0.604	0.523	0.461	1.128	1245	0.316	0.459	0.438	0.465	0.682
Agent4410		0.593	0.807	0.522	1.400	1195	0.114	0.336	0.322	0.340	0.900
AgentFish		0.590	**0.911**	0.530	1.501	1894	0.033	0.300	0.287	0.304	**0.997**
Tjaronchery10Agent	x	0.589	0.505	0.425	1.094	775	0.319	0.515	0.496	0.518	0.681
GEAAgent		0.577	0.705	0.497	1.282	31	0.207	0.375	0.360	0.379	0.814
SmartAgent	x	0.551	0.362	0.344	0.913	1238	0.418	0.644	0.620	0.647	0.574

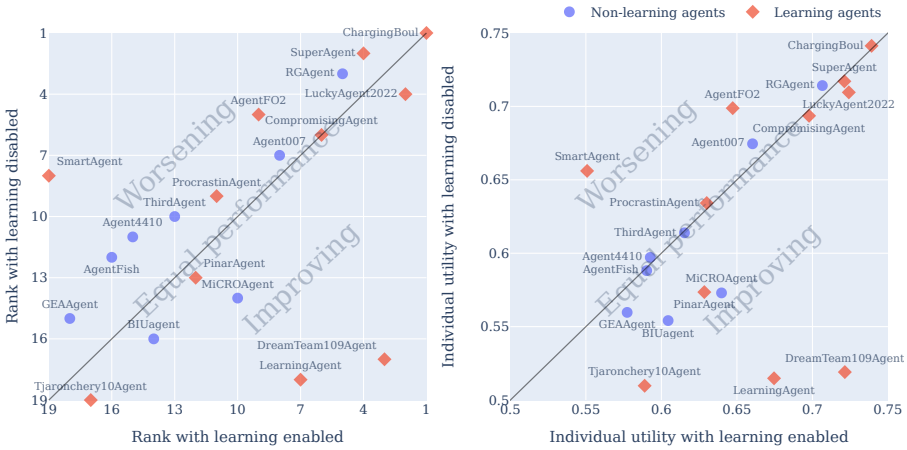


Figure 6.3: Ranking (left) and individual utility (right) comparison of a tournament where learning was disabled and one where learning was enabled. Agents in the lower right improved their ranking when learning was enabled.

are, in fact, agents that learn between sessions. One of them actually made a significant jump from 17th position to 3rd. On the contrary, we also see that SmartAgent performed significantly worse when learning was allowed. Finally, one might expect the non-learning agents to be on the equal performance line as their behaviour is agnostic to disabling the learning capabilities. However, their individual utility is affected by opponents being more capable of finding agreements with them.

PERFORMANCE CONVERGENCE OF LEARNING AGENTS

One problem when evaluating a group of learning agents is that their strategies continuously change, and their scores may not converge. This makes a given ranking dependent on the number of iterations a tournament is run, impacting its robustness. We analyse whether this behaviour can indeed be observed for the agents that were submitted to the competition. To do so, instead of the competition tournament of 50 rounds, a tournament of 1,000 rounds was run for a total of 342 000 negotiation sessions. We report the moving average of the individual utility of the agents against the number of rounds played in Figure 6.4. The window size used is 100 rounds to smooth out the stochastic influence of the random negotiation scenario generator.

Figure 6.4 shows that the ranking and individual utility of the agents keeps changing in the later rounds but that the differences are minor and reasonably stable, but still influence the ranking. Before round 400, differences are more pronounced. We also clearly see a difference between agents with and without learning mechanisms, where the latter exhibit more stable behaviour. The learning mechanisms do not always work out to the benefit of the agents, which we also saw in Figure 6.3; especially AgentFO2 stands out in terms of worsening performance as the rounds progress.

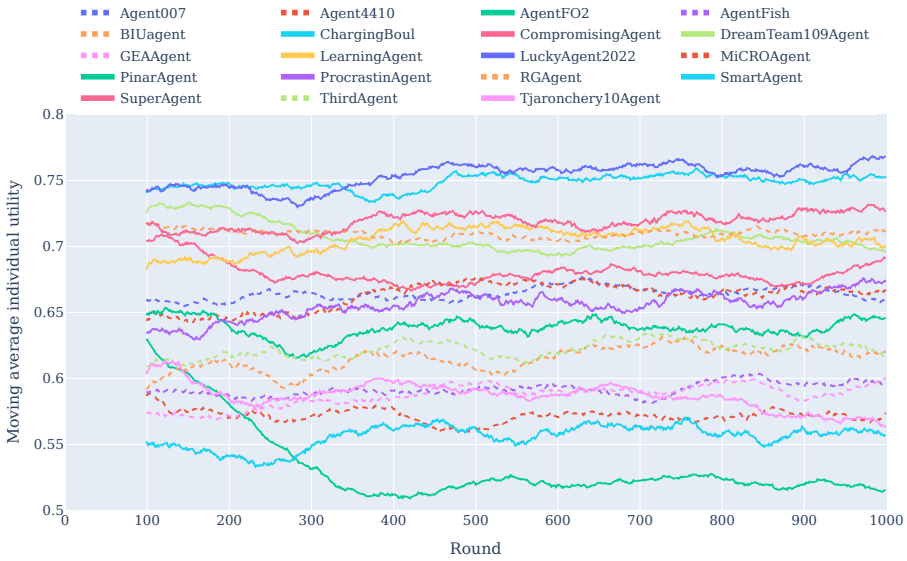


Figure 6.4: Results of a tournament of 1 000 rounds. The moving average of the individual utility with a window size of 100 rounds is visualised. Agents without a learning mechanism are indicated with a dotted line.

6

IMPACT OF GROUP COMPOSITION

We observed during this competition that the composition of the group of agents also influences the final ranking. To explore this further, we evaluated the performance of the submitted agents in every possible group of minimum size 2. Out of the 19 submitted agents, we created all possible 524 268 sub-tournaments (Equation 6.2). Separately running all these tournaments would have been computationally intractable, so we obtained the results naïvely by filtering the result from a full tournament. To the best of our knowledge, none of the participating agents' behaviour is influenced by this naïve approach, as the agents only reason about the current opponent they are facing and their history with that agent. We counted the ranking of every agent in all of the sub-tournaments for both individual utility and social welfare. Both results are plotted in Figure 6.5.

$$\sum_{i=2}^{19} \binom{19}{i} = 524\,268 \tag{6.2}$$

As we can see in the heatmaps, there was a chance for all submitted agents to win the tournament, depending on which opponents were also submitted. This chance was low, but greater than zero, for the agents that obtained a low ranking in the full tournament. This observation is more pronounced for the higher-ranking agents, as chances to win a sub-tournament are much more similar. This means that, at least in part, winning the competition was a matter of chance, depending on the submitted opponents.

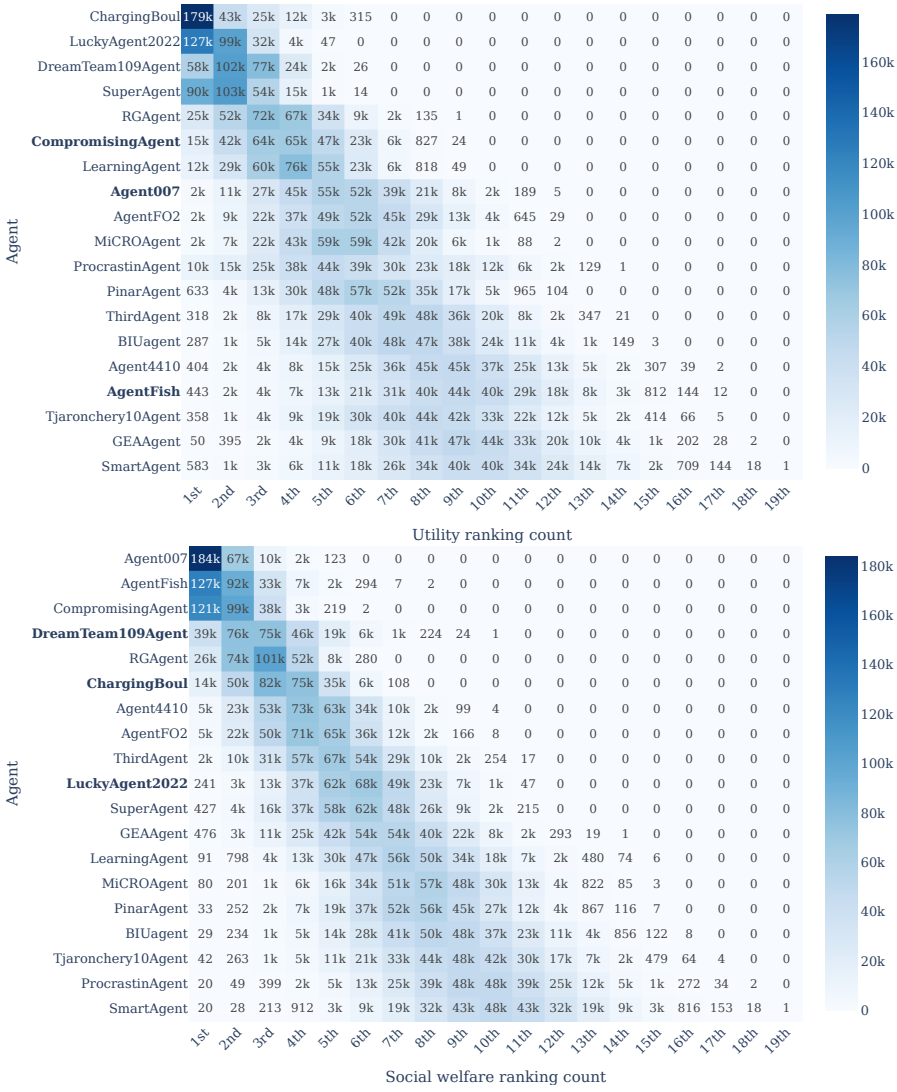


Figure 6.5: Heatmap of the number of times an agent obtained a certain rank in average individual utility (top) and social welfare (bottom). Results are counted over a total of 524 267 tournament setups that could be created with the submitted agents. The agents are sorted based on their ranking in average individual utility (top) and social welfare (bottom) in Figure 6.2. The top three agents in the other category are boldfaced.

In fact, simply averaging the performance of the agents provides a reasonable ranking under the assumption that all opponents are equally likely to be encountered. One could argue that this is unlikely to be the case as, especially after obtaining these tournament results, the underperforming agents are not likely to be used. This degrades the value of the obtained ranking. Therefore, we take a step towards analysing the performance of the agents beyond average scores in a tournament.

6.6.3 GAME-THEORETICAL ANALYSIS OF THE AGENTS

In the following, we evaluate the agents from a more game-theoretical point of view through empirical game theory analysis [158]. Specifically, we construct a meta-game of the underlying negotiation game where agents must select one of the ANL 2022 competition agents to negotiate on their behalf. This would automatically mitigate the previously described issue that underperforming agents are unlikely to be used in practice and could be more in line with a realistic scenario. We analyse which agents are likely to be picked and whether Nash equilibria can be found in such a meta-game.

This analysis assumes that agents are perfectly rational, which may be too strong an assumption for real-world applications. However, our previous evaluation method is also based on an unrealistic assumption that all opponents are equally likely to be encountered. We argue that tournament evaluation and game-theoretical evaluation both have their advantages and their disadvantages. For the same reason, other authors also performed game-theoretical evaluations [160, 10, 28].

We averaged the bilateral result of every agent against every opponent separately and combined these results into a matrix. This matrix can then be seen as the payoff matrix of a symmetric normal-form game in which the two players choose one of the agents as their strategy. Note that, in order to obtain the full matrix, we also need to have, for each agent, the score it would obtain when negotiating against itself, while the ANL 2022 tournament did not involve self-play. We therefore repeated the tournament, but this time including self-play, to obtain those scores. The payoff matrix U we obtained is displayed in Table 6.5. For readability, we multiplied the scores and their standard errors by 1000. Each entry $U_{A,B}$ represents the average individual utility obtained by agent A when playing against agent B (averaged over 2500 negotiation sessions).

NASH EQUILIBRIA

The meta-game has two pure Nash equilibria: SuperAgent against SuperAgent $U_{4,4}$, and MiCROAgent against MiCROAgent $U_{10,10}$. Of these two equilibria, MiCROAgent against MiCROAgent achieves the highest payoff for both players and is therefore preferred.

We performed several statistical tests to verify critical results in Table 6.5. First, we verified both pure Nash equilibria by checking whether the respective agent playing against itself actually results in the highest average individual utility. That is, for each agent $B \in \{\text{MiCROAgent}, \text{SuperAgent}\}$ we performed a one-sided Welch t-test against the null hypothesis that $\bar{U}_{A,B} \geq \bar{U}_{B,B}$ for each opponent A (with $A \neq B$)

Table 6.5: Results of game-theoretical evaluation. Each cell displays the average score of the agent indicated in the row header, along with its standard error, obtained against the agent in the corresponding column. The scores and standard errors are multiplied by 1000 for readability. In each column, the highest score is indicated in boldface.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
1 ChargingBoul	809	674	705	684	863	798	602	881	884	764	490	**725**	**918**	622	**978**	943	560	785	427
	± 3	± 5	± 5	± 5	± 3	± 3	± 7	± 2	± 3	± 4	± 6	± 5	± 3	± 8	± 1	± 2	± 7	± 6	± 7
2 LuckyAgent2022	813	671	828	690	832	929	605	935	808	526	499	593	861	511	966	964	500	777	397
	± 5	± 7	± 4	± 7	± 5	± 2	± 8	± 1	± 6	± 8	± 8	± 8	± 5	± 9	± 2	± 1	± 8	± 7	± 8
3 DreamTeam109Agent	797	**698**	743	744	852	916	**659**	955	732	623	435	619	821	478	946	957	569	739	445
	± 5	± 5	± 5	± 5	± 4	± 3	± 5	± 1	± 7	± 5	± 6	± 6	± 5	± 7	± 2	± 1	± 6	± 5	± 6
4 SuperAgent	816	674	802	**767**	776	929	598	954	862	607	**547**	555	766	502	955	966	534	737	402
	± 5	± 7	± 4	± 5	± 7	± 2	± 8	± 1	± 5	± 7	± 7	± 8	± 7	± 9	± 2	± 0	± 9	± 8	± 8
5 RGAgent	761	679	736	619	817	746	589	864	803	591	515	692	851	663	830	930	**573**	**811**	**465**
	± 3	± 5	± 3	± 6	± 3	± 2	± 6	± 2	± 5	± 7	± 6	± 6	± 3	± 7	± 4	± 1	± 7	± 5	± 7
6 CompromisingAgent	787	644	610	641	**879**	790	585	807	**892**	613	395	640	819	**694**	791	980	558	792	434
	± 4	± 4	± 5	± 5	± 3	± 5	± 5	± 4	± 3	± 5	± 5	± 6	± 5	± 5	± 6	± 0	± 6	± 6	± 6
7 LearningAgent	733	623	885	630	759	957	620	985	808	426	444	506	668	525	645	979	452	734	382
	± 8	± 8	± 3	± 8	± 8	± 2	± 8	± 1	± 7	± 8	± 8	± 9	± 9	± 9	± 9	± 0	± 8	± 8	± 8
8 Agent007	702	631	529	573	780	797	462	791	821	622	393	651	698	650	907	931	558	760	424
	± 3	± 5	± 5	± 5	± 3	± 3	± 5	± 5	± 3	± 5	± 5	± 4	± 6	± 6	± 3	± 2	± 6	± 5	± 5
9 AgentFO2	724	631	565	618	726	682	565	746	734	745	474	631	847	557	799	850	549	553	389
	± 3	± 5	± 6	± 5	± 5	± 4	± 6	± 2	± 5	± 4	± 5	± 6	± 3	± 8	± 6	± 2	± 7	± 8	± 7
10 MiCROAgent	**822**	577	865	657	728	935	500	916	847	**820**	104	497	719	336	894	875	383	655	206
	± 4	± 9	± 5	± 8	± 8	± 2	± 9	± 1	± 3	± 2	± 6	± 9	± 7	± 9	± 2	± 2	± 9	± 9	± 8
11 ProcrastinAgent	683	587	916	688	747	**981**	540	**995**	781	85	377	87	605	322	874	**1000**	444	733	272
	± 8	± 9	± 5	± 8	± 8	± 2	± 9	± 1	± 8	± 5	± 9	± 5	± 10	± 9	± 7	± 0	± 10	± 8	± 8
12 PinarAgent	750	582	830	555	762	744	512	880	784	488	90	384	758	432	857	881	433	704	267
	± 5	± 8	± 5	± 8	± 6	± 6	± 9	± 2	± 5	± 8	± 5	± 9	± 7	± 9	± 4	± 2	± 9	± 7	± 8
13 ThirdAgent	614	569	683	553	739	693	515	595	753	612	363	598	648	671	708	731	524	730	422
	± 4	± 5	± 5	± 6	± 4	± 5	± 7	± 5	± 4	± 7	± 7	± 6	± 6	± 7	± 5	± 4	± 8	± 6	± 7
14 BIUagent	641	503	744	512	716	851	531	896	627	308	321	416	712	420	798	898	478	703	227
	± 8	± 9	± 7	± 9	± 7	± 4	± 9	± 3	± 8	± 8	± 9	± 9	± 7	± 9	± 5	± 1	± 9	± 7	± 8
15 Agent4410	472	513	541	532	737	670	496	639	595	643	377	602	798	660	766	777	478	752	384
	± 5	± 5	± 5	± 5	± 5	± 6	± 7	± 5	± 6	± 5	± 5	± 5	± 5	± 6	± 4	± 4	± 8	± 6	± 7
16 AgentFish	540	532	504	509	644	493	519	646	708	715	389	638	696	614	796	772	543	700	438
	± 5	± 6	± 6	± 6	± 5	± 5	± 5	± 5	± 4	± 4	± 5	± 5	± 5	± 5	± 2	± 4	± 6	± 5	± 5
17 Tjaronchery10Agent	685	529	898	501	668	811	504	959	723	261	293	322	607	387	548	956	527	652	297
	± 8	± 9	± 4	± 9	± 8	± 7	± 9	± 1	± 8	± 7	± 7	± 8	± 9	± 8	± 9	± 1	± 8	± 8	± 7
18 GEAAgent	630	548	758	514	682	614	522	669	537	457	525	537	668	563	631	787	511	717	240
	± 5	± 6	± 4	± 6	± 5	± 5	± 6	± 5	± 7	± 7	± 6	± 6	± 5	± 7	± 6	± 3	± 7	± 5	± 7
19 SmartAgent	556	477	**921**	492	611	890	456	994	545	181	281	274	578	232	608	994	452	370	219
	± 9	± 9	± 5	± 10	± 9	± 6	± 9	± 1	± 9	± 7	± 8	± 8	± 9	± 8	± 9	± 0	± 10	± 9	± 8

and set a maximum significance level of $p = 0.05$ to reject the null hypothesis. Note that we use $\bar{U}$ to denote the *true* expected utility that agent A would obtain against agent B, whereas U represents the *measured* average individual utility. We found that the highest p-value for this hypothesis was $4 \cdot 10^{-34}$ for MiCROAgent and $6 \cdot 10^{-4}$ for SuperAgent. Each of these p-values still needs to be multiplied by 18, to take into account that for only one of the 18 opponents, the null hypothesis needs to be true to reject our conclusion, but then they still stay well below the threshold of $p = 0.05$, so our claims that MiCROAgent and SuperAgent both form a Nash equilibrium are statistically significant. Furthermore, we inverted this test to verify that none of the other agents forms a Nash equilibrium when playing against itself. That is, for each agent $B \notin \{\text{MiCROAgent}, \text{SuperAgent}\}$, and each agent A (with $A \neq B$) we performed a one-sided Welch t-test against the null hypothesis that $\bar{U}_{A,B} \leq \bar{U}_{B,B}$. Indeed, for each agent B we found at least one opponent A for which the p-value for this hypothesis was far below 0.05. Therefore, we can reject the hypothesis that $\bar{U}_{A,B} \leq \bar{U}_{B,B}$ for *all* opponents A.

Apart from pure Nash equilibria, we also found 21 *mixed* Nash equilibria using the Gambit software package (v.16.2) [134]. However, for each of these mixed equilibria, the payoff was lower than for the two pure equilibria. The top mixed equilibrium found has a probability of 66% for SuperAgent and 34% for MiCROAgent. In this mixed equilibrium, both players receive an expected utility of 0.712, which is significantly lower than the utility they would achieve if they both played SuperAgent (0.767) or if they both played MiCROAgent (0.820).

We note that MiCROAgent does not perform well in the tournament evaluation but does perform strongly in the game-theoretical evaluation. A quick analysis shows that MiCROAgent works particularly well against competitive opponents and less so against weaker opponents. As the game-theoretical approach depends on selecting the best possible response, it emphasises results obtained against stronger opponents. As mentioned earlier, the average scoring used in the tournament evaluation is based on the assumption that all opponents are equally likely to be encountered, which could be considered unrealistic. This also suggests that learning is especially beneficial in the presence of weaker agents that can be exploited. If only stronger agents are present, the learning agents may lose their advantage over a simpler approach such as MiCROAgent.

6.6.4 ANALYSIS OF THE NEGOTIATION SCENARIOS

The characteristics of the randomly generated negotiation scenarios can have a significant influence on the performance of the submitted agents [70]. We analyze the scenarios based on characteristics often used in the automated negotiation literature: the opposition, distribution, and balance scores described in the sections below. The cumulative distribution functions of these metrics for the negotiation scenarios used in this paper are visualised in Figure 6.6 and Figure 6.7. The maximum social welfare and maximum Nash product are included in Figure 6.6.

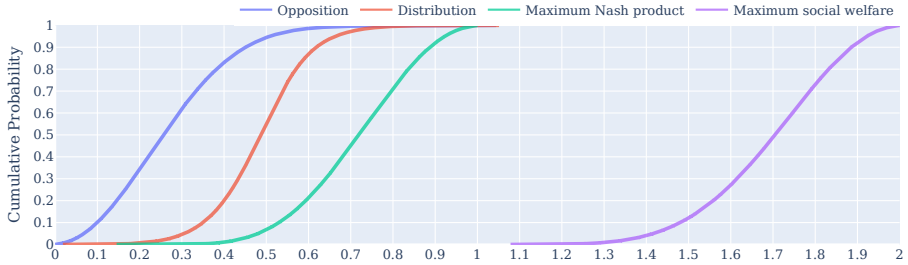


Figure 6.6: Cumulative distribution functions of the characteristics of the randomly generated negotiation scenarios used in this paper.

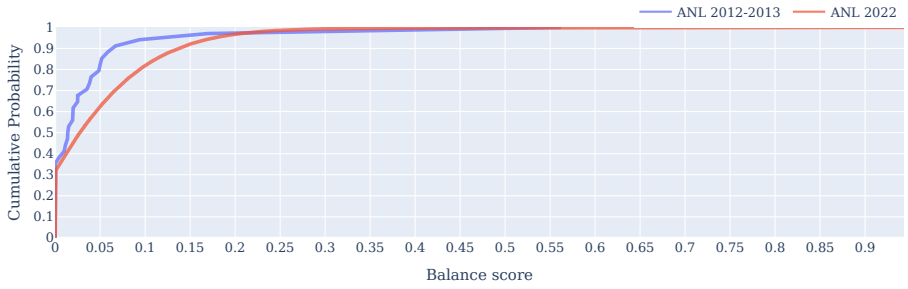


Figure 6.7: Cumulative distribution functions of the balance scores of the negotiation scenarios of the ANL 2012-2013 and ANL 2022 editions. The scenarios used for ANAC 2022 are less balanced than those used for ANAC 2012/2013. For example, we see that 80% of the ANAC 2012/2013 scenarios had a balance score less than or equal to 0.050, while among the ANAC 2022, this fraction was only 62%

Table 6.6: Average individual utility and success rate for all agents, compared between negotiation scenarios with low, medium and high opposition.

	low opposition $s_{opp} < 0.20$	medium opposition $0.20 \leq s_{opp} \leq 0.325$	high opposition $0.325 < s_{opp}$
Av. util. all sessions	0.814	0.647	0.471
Av. util. sessions with agreement	0.877	0.784	0.690
Agreement ratio	0.93	0.83	0.68

OPPOSITION

A commonly used measure to quantify the competitiveness of a negotiation scenario is the opposition value, which indicates how easy it is to find agreements that are satisfactory to both agents [10]. Before calculating the opposition value, the subset of Pareto efficient outcomes Ω_p must be extracted. An outcome is Pareto efficient if no other outcome improves the utility of at least one of the agents while not decreasing the utility for the others (Equation 6.3). From this Pareto-efficient set, we select the Kalai-Smorodinsky bargaining solution [83] (Equation 6.4) and use it to calculate the opposition based on Equation 6.5.

$$\Omega_p = \left\{ \omega \in \Omega \mid \neg\exists \omega' \in \Omega : \begin{bmatrix} (u_A(\omega') > u_A(\omega) \wedge u_B(\omega') \geq u_B(\omega)) \\ \vee \\ (u_A(\omega') \geq u_A(\omega) \wedge u_B(\omega') > u_B(\omega)) \end{bmatrix} \right\} \tag{6.3}$$

$$\omega_{Kalai} \in \arg\min_{\omega \in \Omega_p} |u_A(\omega) - u_B(\omega)| \tag{6.4}$$

$$s_{opp}(\Omega) = \sqrt{(1 - u_A(\omega_{Kalai}))^2 + (1 - u_B(\omega_{Kalai}))^2} \tag{6.5}$$

We split the negotiation scenarios into three roughly equal-sized categories with low, medium and high opposition values. The results of this analysis are displayed in Table 6.6. We observe that the lower the opposition values, the better the agents perform in terms of agreement rate and utility obtained from the agreement. Another interesting observation is that the opposition value has a noticeable influence on the ranking of the agents. Most notably, DreamTeamAgent109 ends in seventh place on the scenarios with low opposition, while it ends in first place on the scenarios with high opposition.

DISTRIBUTION

The same analysis was carried out for the distribution value, $s_{dist}(\Omega)$, which is defined in Equation 6.6; it is the average Euclidian distance in utility of every outcome to its closest Pareto-efficient outcome. The higher this value, the more difficult it becomes to find an agreement that is (close to) Pareto efficient. Estimating the preferences of the opponent accurately is essential in negotiation scenarios with a high distribution value.

$$s_{dist}(\Omega) = \sum_{\omega \in \Omega} \left[\min_{\omega' \in \Omega_p} \sqrt{(u_A(\omega) - u_A(\omega'))^2 + (u_B(\omega) - u_B(\omega'))^2} \right] \tag{6.6}$$

Table 6.7: Average individual utility and success rate for all agents, compared between negotiation scenarios with low, medium and high distribution values.

	low distribution $s_{dist}, 0.450$	medium distribution $0.450 \leq s_{dist} \leq 0.525$	high distribution $0.525, s_{dist}$
Av. util. all sessions	0.516	0.656	0.777
Av. util. sessions with agreement	0.720	0.793	0.857
Agreement ratio	0.72	0.83	0.91

The results are displayed in Table 6.7. Again, we clearly observe a difference in the performance of the agents depending on the distribution value. The higher the distribution value, the higher the score of the agents, both in terms of the quality of the deals made and the percentage of negotiations that end with a deal.

These results may initially seem surprising since we argued that finding Pareto-efficient outcomes in scenarios with high distribution is harder. However, if a negotiation scenario has a high distribution, the outcomes are also more scattered throughout the utility space and may, in turn, contain outcomes with high utility for both agents. A change in the ranking of the agents is also observed. Most notably, DreamTeamAgent109 ends in second place on scenarios with low distribution, while it ends in seventh place on scenarios with a high distribution.

BALANCE SCORE

Recent work stated that the negotiation scenarios used for ANAC 2012 and 2013, also used in many later editions of ANAC, were too simplistic [40]. They showed that many of these scenarios display a certain type of symmetry, which makes them easy to tackle by a naïve strategy called the MiCRO strategy (which also participated in ANL 2022, see Section 6.5.2). This was demonstrated by the fact that MiCRO was able to outperform some of the best agents in these scenarios, even though MiCRO is a much simpler strategy that does not apply any form of opponent modelling or learning [40].

To quantify this symmetry, De Jonge [40] defined the notion of the "balance values" of a negotiation scenario, which are the individual utilities of the outcome ω_b that two agents would agree upon if they both apply the MiCRO strategy. They showed that many of the ANAC 2012 and 2013 negotiation scenarios are "balanced", meaning that the balance values lie very close to the Nash Bargaining Solution [111] (NBS). De Jonge [40] argued that a more versatile set of negotiation scenarios should be used to test agents.

We performed the same analysis to see to what extent the scenarios used in ANAC 2022 are balanced. Similar to De Jonge [40], the balancedness of a negotiation scenario is measured by comparing the balance values to the utilities associated with the optimal outcome. However, we do not consider the NBS as the optimal outcome but rather argue that the optimal solution is the one that maximizes social welfare.

The motivation for using the Maximum Social Welfare Solution (MSWS) instead of the NBS as the optimal outcome is twofold. First, maximizing social welfare was explicitly one of the goals of the competition. Second, even if the goal is to maximize

individual utility, each scenario has two utility functions, u_A and u_B, randomly assigned to the two agents. Maximising the expected individual utility before being assigned a utility function would equal agreeing to the MSWS in advance (see Jonge [80] for a more detailed discussion).

$$s_{bal}(\Omega) = \max_{\omega \in \Omega} \{u_A(\omega) + u_B(\omega)\} - (u_A(\omega_b) + u_B(\omega_b)) \tag{6.7}$$

We define the *balance score* s_{bal} in Equation 6.7, where ω_b is the outcome that would be obtained by 2 MiCRO strategies. The utilities are assumed to be normalized to fall within the range [0,1]. The lower the balance score, the closer the balance values are to the MSWS. If the balance score is exactly 0, the balance values coincide with the MSWS, which in turn means that in such a negotiation scenario the agreement made between two MiCRO agents would always be exactly the MSWS.

The result of the analysis is shown in Figure 6.7. We observe that the negotiation scenarios of ANL 2022 are less balanced than those of ANAC 2012 and 2013 and should therefore be preferred for research. The fact that we see different values between the two types of scenarios can be explained by the fact that the ANAC 2012/2013 scenarios were handcrafted by participants, while the scenarios of ANL 2022 were randomly generated.

However, ANL 2022 scenarios still seem to display a high degree of balance. Around half of the scenarios have a balance score of 0.025 or less, and for around one-third of the scenarios, the balance score is exactly 0. It remains an open question why exactly the randomly generated scenarios are so balanced and how this compares to real-world negotiation scenarios.

We recalculated the scores of all the agents while only counting negotiation sessions involving negotiation scenarios with low balance scores ($s_{bal} = 0$), with medium balance scores ($0 < s_{bal} \leq 0.05$) and with high balance scores ($0.05 < s_{bal}$). The balance scores were chosen such that each category contained roughly one-third of all scenarios.

While we expected that MiCRO would perform better on the balanced scenarios, we noticed that this was actually the case for *all* agents. The agents make better and more agreements on the balanced scenarios, while the opposite is true for the unbalanced scenarios. The results are summarized in Table 6.8. The differences are lower than in Table 6.6 and Table 6.7, suggesting that the opposition value and the distribution value are better indicators of the level of difficulty of a scenario than the balance score.

Finally, not much difference was observed in the outcome of the tournament. The final ranking in the tournament evaluation remains more or less the same, with a few agents moving one or two positions up or down the ranking.

6.7 DISCUSSION

The extensive analysis we performed using the agents that were submitted to ANL 2022 led to insights into their behaviour that we will now discuss. We will discuss some general observations first, and after that, we will discuss two core topics more in-depth.

Table 6.8: Average individual utility and success rate for all agents, compared between negotiation scenarios with low, medium, and high balance scores.

	low balance score $s_{bal} = 0$	medium balance score $0 < s_{bal} \leq 0.050$	high balance score $0.050 < s_{bal}$
Av. util. all sessions	0.689	0.652	0.611
Av. util. sessions with agreement	0.817	0.796	0.774
Agreement ratio	0.84	0.82	0.79

Regarding the negotiation scenarios used, we observed that the randomly generated scenarios from ANL 2022 are slightly less balanced than the handcrafted scenarios used in many of the earlier editions of ANL. This indicates that the randomly generated scenarios are slightly more challenging than the handcrafted ones. We also showed that the agents performed better on the balanced scenarios, but this did not significantly influence the outcome of the tournament. On the other hand, the more classical notions, such as opposition and distribution of the outcome space, correlated more strongly with the performance of the agents.

We can calculate from Table 6.4 that 18.4% of the negotiation sessions ended without agreement. This is undesirable, as it lowers the social welfare obtainable by the group, which is to no one's benefit. However, competitive strategies that cause these failures to reach an agreement are beneficial for individual utility, which we observed more often in the history of ANL, and which is also a general observation in partially cooperative (general-sum) games, such as the Prisoner's Dilemma. It would, therefore, seem useful to investigate the design of mechanisms that push agents towards cooperating strategies [112].

6.7.1 LEARNING IN NEGOTIATION AGENTS

We have outlined and analysed agents that learn from repeated encounters with opponents and presented the results of a competition between such learning agents. The main focus of the competition was to assess the strength of algorithms that can learn from previous encounters in groups with other learning agents and use this knowledge to adapt their strategies to individual opponents. Since not every participant submitted an agent that implemented such a learning approach, we were able to compare the results of learning and non-learning agents and showed that the learning agents indeed performed better than the non-learning agents for the challenge set in the competition.

When the learning capabilities are disabled, some learning agents drop substantially in performance. While this suggests that learning is beneficial, it could also indicate that these agents became dependent on their learning mechanism. Ideally, agents should be more robust and perform well in single-shot and repeated encounters with opponents. This could have been achieved by evaluating agents in the competition based on their single-shot performance, which we believe is an interesting idea for future competitions.

A notable observation from this competition was that the agent that scored highest in the tournament, ChargingBoul, did not produce the best response against

itself. If one is to select one of the agents that participated in the competition, choosing the winner may not be the best option. Instead, we observed that two agents formed a pure Nash equilibrium when playing against themselves (MiCRO and SuperAgent), of which MiCRO scored the highest. This is remarkable, as MiCRO is a naïve strategy that does not use any form of learning or opponent modelling. However, while MiCRO was the strongest participant in the game-theoretical sense, it only ended in 10th place in the tournament evaluation. We can explain this based on the fact that MiCRO fails to exploit weaker opponents and/or fails to make agreements with them.

To our knowledge, none of the agents let the group composition influence their strategy. Still, it greatly impacts overall performance within a competition setting, and we believe that this could be an interesting topic for future research.

6.7.2 RANKING NEGOTIATION AGENTS

Our analysis of the competition results used various methods to analyse the performance of negotiation agents. One is based on average performance, the default method in the automated negotiation community, and one is based on Nash equilibria found using empirical game theory in a meta-game.

The ranking based on average performance strongly depends on the submitted agents, as shown in Figure 6.6.2. Surely, this is the case in any AI competition, but it is even more apparent in agent-versus-agent competitions. In such competitions, a single added agent influences the score of all the other agents instead of merely contributing another score that is added to the ranking. Intuitively, in a group of defecting strategies, a conceding strategy is needed to win a tournament, as some utility is better than no utility, and the defecting opponents will also not obtain agreements with each other. Conversely, a hardheaded strategy is needed to win in a group of conceding strategies, as it will exploit all opponents, obtaining the highest utility.

The game theoretic analysis did not give us a full ranking but a selection of strategies that form Nash equilibria in the meta-game. Such equilibria rely on flawless rational agents that all know the full structure of the game, which is a disputable assumption. The equilibria also depend on the strategy set included, but many more strategies will likely exist than were submitted to this competition. There can be more than one pure equilibrium, which makes it unclear which equilibrium is played. Equilibria can be unfair to one of the agents involved. The last two points apply, for example, to the game of chicken. All in all, there are many counterarguments for analysing agent performance based on Nash equilibria.

We attempted to find a ranking method that would be a natural fit for this competition but were unsuccessful. We considered Elo ranking, which assumes that relative skill is transitive and is sensitive to copies of the same strategy [21]; unfortunately, both of these assumptions are problematic for ranking negotiation agents. Nash Averaging [21] is also a popular ranking method but can only deal with zero-sum games and is sensitive to the set of included agents [91]. In contrast, our setting is a general sum game, and we attempt to avoid methods sensitive to the set of included agents. On the other hand, α-rank [114] is suitable for general

sum games. It uses replicator dynamics to find a dynamical solution concept and extracts a ranking based on that. However, the main motivation behind this method was computational tractability, which is not an issue in our case, and the obtained ranking depends on the α parameter setting. We attempted to obtain a stable ranking using this method but were unhappy with the sensitivity to the α parameter. We checked ranking methods based on social choice theory and voting [91], but these methods consider ordinal pairwise comparisons, where our meta-game is cardinal. Finally, the ranking method that we believe comes closest to our needs has been proposed by Marris et al. [103] and was developed to rank N-player general sum games. However, it proposes a ranking based on the (Coarse) Correlated Equilibrium obtained through the maximum entropy objective. Agents require an external correlation device to be able to optimise for a correlated equilibrium, which is a requirement that might not be realistic for negotiation games.

This leaves us without a convincing ranking method for negotiation agents in competitions like ANL 2022. We see the development of such a ranking method as a worthwhile yet highly non-trivial undertaking that is beyond the scope of our work presented here. We hope that our discussion here draws attention to this important topic and serves as a good starting point for discussion within the negotiation community, as much of the recently published work is still benchmarked on the average performance of a given agent.

6.8 CONCLUSION

In this chapter, we discussed learning agents in iterated negotiation games and provided an in-depth analysis of groups of such agents competing in a tournament. We utilised ANL to obtain a diverse set of learning negotiating agents by making learning over repeated games the challenge of the 2021 and 2022 editions. We ran experiments with these agents and extensively analysed the results. To the best of our knowledge, this is the first analysis of learning negotiating agents competing in a tournament.

The agents were designed for the performance metric we set for the competition. Regarding this metric, we found that the agents equipped with a learning mechanism performed better than those that did not, and we conclude that the challenge we set was successful. We also observed that complex strategies are sometimes outperformed by relatively naïve strategies, such as the MiCRO strategy, which managed to outperform every other agent in being a pure Nash equilibrium action in a strategy selection meta-game.

Agents that performed well in the competition show more competitive (defecting) behaviour, despite this sometimes causing failures to reach an agreement and thus hurting both their own utility and the social welfare of the group. We should aim to prevent such behaviour in negotiation games if we care about social welfare.

We showed that the ranking of the agents depends on the submitted opponents when using simple average performance-based ranking methods. Such ranking methods assume that opponents are equally likely to be encountered. We attempted other methods to evaluate agents and showed that the performance of agents does not transfer across these evaluation methods. All ranking methods in this paper are

based on assumptions that can be disputed, and obtained rankings vary depending on the chosen method. This is not unexpected, but still unsatisfactory; as discussed in Section 6.7, this suggests room for further improvements in evaluation criteria and the automated negotiation competition.

To conclude, we designed and analysed a first negotiation competition with a focus on learning across negotiation sessions. We made significant progress in setting up a framework for analysing the performance of learning negotiating agents. Despite this advancement, we note that challenges remain in obtaining a definitive answer to the question of what a good performing negotiating agent is. There might not be a single answer to this question, as it also depends on a conscious choice of which objectives are important and the environment the agent is in. We should push for more (empirical) research into diverse adaptive and learning negotiating agents to gain a more robust understanding of the performance of such agents.

7

MULTI-AGENT MEETING SCHEDULING

In multi-agent systems (MAS), applications that directly interface with daily human activities represent a rich avenue for exploration. This paper dives into a potentially impactful application of MAS, targeting a well-known real-world challenge: meeting scheduling. While there have been previous efforts to address this challenge, we believe that the time is right to revisit this task as a blue-sky challenge for the MAS community.

Traditional scheduling methodologies rely on static, sub-optimal support tools that are susceptible to inefficiencies that include repeated rescheduling, and the overhead for the humans affected per scheduling attempt remains substantial. This opens an intriguing challenge for the MAS community: What if a collection of autonomous agents could extend human capabilities, designed to adapt and negotiate, making scheduling more dynamic and less time-consuming? The potential of collective time saved is substantial, not only in a reduction of human effort due to fewer rescheduling attempts, but also in better alignment of schedules. Furthermore, the privacy of participants can be better preserved.

We argue that the richness of this domain is of interest to the MAS community and that recent advances in AI open up new ways for tackling this challenge. In this paper, we set the stage for this research direction, focussed on the use of MAS to support an age-old, yet fundamental and pervasive task.

This chapter has been published at the Adaptive and Learning Agents (ALA) workshop at AAMAS 2024 [125].

7.1 INTRODUCTION

Meetings with others, for social and work-related interactions, form a crucial part of our daily lives. A 2014 survey by Ovum[1] found that employees meet eight times per week on average and that this number has been rising over the years. More specifically, executive management and higher meet on average 12 times a week and VPs, directors and C-level roles in highly collaborative industries reach an even higher average of 17 meetings per week.

These business meetings need to be scheduled, which takes an average of 26–30 minutes per meeting per participant according to a blog by Doodle[2]. This makes scheduling meetings a major time investment for the average employee, who likely has to schedule their own meetings. Higher-ranking roles often have assistants who perform this scheduling task on their behalf and only occasionally ask their bosses for confirmation. Furthermore, manual scheduling can lead to sub-optimal schedules, due to the complexity of the problem. Tools that support solving this problem are popular[3], but are often suboptimal as they only solve part of the problem. There is much to be gained from improving the process of scheduling, which is also recognised by industry[4].

We informally define the meeting scheduling problem (MSP) as the problem of finding a time slot of a desired duration in which the intended set of participants (or an acceptable subset thereof) commit to attending the meeting at an agreed location. We note that the location can be on-site, online, or a mixture thereof. Furthermore, the notion of an "acceptable subset" makes this *de facto* a family of problems, as it leaves unspecified who determines what defines the acceptability of that subset. In terms of complexity, the problem becomes easy, if this is determined by the one that initiates the scheduling (authority), and most complex if acceptability is determined by a group process amongst the intended participants.

The meeting scheduling problem, being such a major part of daily human life, has seen decades of attention from the computer science community, first appearing in the 80s [63, 102], often modelled as a (form of a) constraint satisfaction problem (CSP) [150]. Researchers have attempted solving the multi-agent MSP using market-based approaches [45] and negotiation approaches, where agents, representing users, negotiate over meeting time slots [136, 78]. In the years after, the problem consistently continued to receive attention among researchers (e.g., [60, 55, 34, 167, 94, 85, 155]) across several communities.

Despite the MSP being a common and relatable problem that has seen considerable effort from the research community, we are still not close to a system that alleviates most of the burden. Difficulties in learning human preferences, communication with humans, and the complexity of decentralized mixed-motive multi-agent problems render the MSP challenging. Many of these challenges are recognised as open problems in cooperative AI [36]. With the recent successes in (multi-agent)

[1] Ovum 2014 - Collaboration 2.0: Death of the Web Conference (As We Know It)

[2] https://doodle.com/en/resources/blog/study-reveals-time-spent-with-scheduling/

[3] https://www.gartner.com/reviews/market/scheduling-automation-software

[4] https://www.mckinsey.com/capabilities/operations/our-insights/smart-scheduling-how-to-solve-workforce-planning-challenges-with-ai

(deep) reinforcement learning [146], large language models (LLM), and reinforcement learning from human feedback (RLHF) [144], we believe that now the time is right to revisit the MSP as a rich and rewarding real-world challenge for the MAS community. We also believe that the various communities within computer science that have studied this problem can come together to meet this challenge and jointly achieve far better solutions than currently available.

In this paper, we lay out the necessary groundwork for tackling this problem. We discuss the characteristics of the problem and try to isolate its distinct components. We believe that a decentralised negotiation-based solution is the best-fitting approach to solving the MSP for scalability and practical reasons; this, therefore, forms the basis of our effort.

7.2 THE MEETING SCHEDULING PROBLEM

The following anecdotical meeting scheduling process illustrates the richness of the MSP: Alice must schedule a meeting with 4 colleagues, of which 2 are notoriously busy. In his role as organiser, Alice first asks the 2 busy participants about their constraints and options. The first slot found is too far in the future, so the organiser reduces the meeting duration, allowing for two earlier slots. Alice proposes these slots to the other participants. One of them, Bob, already has other obligations conflicting with both slots, but might be able to reschedule a meeting and requests the others to agree on one of the two slots. After the agreement is made, Bob commits to that slot after rescheduling his previous commitment.

In the introduction, we provided a simple and informal definition of MSP that nonetheless already introduces complexity by referring to “an acceptable subset”. Furthermore, under the hood of this definition lurk additional complexities, as mentioned, *e.g.*, by Berger et al. [23]: Each participant must be able to reach the meeting location, attend for the entire duration and reach the next meeting location on time. This refers to travel time between meetings and to means of transportation. Even in online meetings, one must be in a place where one is allowed to speak, and that is quiet enough to hear what is being discussed. Aside from such practicalities that complicate MSPs, there are also numerous human aspects to consider, e.g., participants having ulterior motives and/or hidden agendas, strategic voting, powerplay, and incomplete revelations of potential meeting slots. Any and all of these have an impact on what information they are willing to share and when, how much importance they attach to the meeting and some of its intended participants, and how many attempts have to be made to arrive at a feasible solution.

7.2.1 EARLIER FORMALISATIONS

The MSP naturally lends itself to be formalised as a constraint satisfaction problem (CSP) [150]. Formalised in this manner, all the techniques for solving CSPs are applicable. This includes centralised and distributed approaches. Centralised approaches boil down to efficient search strategies in the solution space defined by means of hard constraints or strategies for optimising the utility when using soft constraints. In the distributed approaches, solving the CSP is distributed to

Table 7.1: Characteristics of the Meeting Scheduling Problem

	Problem	Participant			
Description	**Type**	**Type**	**Values**		
Substitutability	Boolean	Set	Other participants		
Importance of attendance	Boolean	Continuous	0 - Irrelevant	1 - Crucial	
Calendar observability	Boolean	Categorical	None	Availability	Full
Rescheduling	Boolean	Boolean			
Role	Boolean	Categorical	Organiser	Participant	Observer
Repeated encounters	Boolean	Categorical	Yes	Maybe	No
Multiple rounds	Boolean	N/A			
Preferences	Boolean	Categorical	Invisible	On options	Visible
Arguments	Boolean	Categorical	None	Restricted format	Free text

the agents as a local problem [164]. This often scales better than centrally solving a given CSP instance and is more sensitive to information sharing on a need-to-know basis. Examples of models of specific MSPs as CSP include the assignment problem [37], private incremental multi-agent agreement problem (piMAP) [49], group activity selection problem (GASP) [38, 39], valued constraint satisfaction optimisation problem (VCSOP) [133], group scheduling problem (GSP) [93], and stable group scheduling problem (SGSP) [94].

All of the above-mentioned formalisations of the MSP simplify part of the problem and fail to capture the full richness of the MSP. Such simplifications include full visibility of other agents' preferences and assuming that decisions are made centrally. In the following, we attempt to describe the full richness of the MSP as a basis for future research on the topic. In doing so, we refrain from fully formalising the problem, as multiple viable approaches exist. Instead, we focus on the characteristics of the MSP that must be considered when solving this problem.

7.2.2 CHARACTERISTICS

We identified a set of characteristics of the problem of scheduling meetings as well as of the participants involved in the meetings. An overview of these characteristics can be found in Table 7.1. We deliberately make a distinction between these two sets of characteristics. The problem characteristics map the different aspects of the problem that we can either consider or exclude when scheduling meetings. The participant characteristics describe the potential differences that participants have in relation to others within the considered characteristics of the problem. As soon as at least one participant has a certain characteristic, the problem must accommodate this.

To give a few examples of participant characteristics; within an MSP, a participant might have full visibility of the calendar of only part of the group of participants. Participants might also have different views on the importance of other participants' attendance and substitutability. For example, Alice might feel that Bob can be substituted by Carol, whereas Bob feels he cannot be substituted at all. As another example, Alice might feel that Dan's attendance is crucial, whereas Dan does not think the meeting is that important and will only attend if Erin will.

The characteristics we listed can be used to approach the problem in a systematic manner. Due to the richness of the problem, the complexity is also high, and we might want to approach it in incremental steps of complexity. Our characteristics form a structure of challenges that can be attempted in isolation. We will describe the characteristics one by one.

- **Substitutability**: If included, allows for the substitution of (some or all) participants by others. This information is to be observable by the participants. As a participant characteristic, we consider two cases: `Simple`; the set of participants that this participant can be substituted with. This models only the view of a given participant on who can substitute them. `Full`; for this participant, their full view on who (including themselves) can be substituted by whom. Differences in opinion need to be evaluated by all participants.

- **Importance of attendance**: If included, considers that some participants are more important for the meeting than others. Per participant, this is a value in $[0, 1]$ representing the importance of attendance of this or another participant, according to this participant.

- **Calendar observability**: If included, allows sharing of calendars between participants. Per participant, there are three possibilities: `None` if no part of this participant's calendar can be directly observed. `Availability` if only the availability of this participant can be observed. `Full`: the participant's calendar is fully observable by others.

- **Rescheduling**: If included, meetings can be rescheduled to clear slots for other more important meetings. Per participant, whether this participant has the authority and capability to reschedule existing meeting commitments to free a slot.

- **Roles**: If included, participants can have different roles in the MSP. Per participant, these are: `Organiser` if this participant is the organiser. `Participant` if this participant is intended to attend the meeting as a participant. `Observer` if this participant is intended to attend the meeting as an observer.

- **Repeated encounters**: If included, allows for multiple encounters between agents over the course of scheduling different meetings. Per participant, whether the participant is encountered repeatedly.

- **Multiple rounds**: If included, allows for multiple rounds of back-and-forth communication before an agreement is made.

- **Preferences**: If included, consider preferences over meeting slots instead of simple 'yes' or 'no' answers on availability. `none` if this participant only answers 'yes'/'no' to offered slots. `on options` if this participant provides preferences over offered slots. `full` if this participant provides access to their full preference profile.

- **Arguments**: if included, permits arguments to be added to answers regarding availability. Per participant, these are `None`: the participant neither has the capability or authority to add arguments to their answers, nor the ability to interpret arguments made by others. `Restricted format`: the participant can only add or interpret arguments of a prespecified restricted format. `Free text`: the participant can add and interpret text arguments free of other restrictions. The arguments characteristic can be used, for example, to model that a participant provides a conditional answer, *e.g.*, needs consultation with the human user, or that a 'yes' is only valid if an agreement can be found within a given time.

7.2.3 PERFORMANCE CRITERIA

We argue that the following abstract criteria should be considered when evaluating the performance of a solution to the MSP:

- **Obtained utility**: When considering preferences in the MSP, one can measure properties over the utilities attained by all agents when used as a global performance measure. Examples are average utility, Pareto-optimality, distance to Nash product or Rawls point, [117, 120]. One can also look at the utility attained by an individual agent as a local measure.

- **Scheduling success**: The percentage of the meetings that could be scheduled. This measures the effectiveness of a meeting scheduling solution in finding common slots and aligning calendars. It should be easier to obtain a perfect score when the number of agents involved is lower or when the density of meetings is low, but becomes an interesting measure when the opposite is true.

- **Privacy preservation** [56, 54, 49]: When observability is (partially) enabled, which is likely true for real-world scenarios, then it becomes important not to reveal too much information, nor share that information with others.

- **Need for rescheduling**: This can be considered an efficiency measure. If a need for rescheduling meetings arises frequently, this could indicate that the agents are not good at estimating future conflicts and that they schedule meetings too easily.

- **Time investment of humans**: As we advocate to approach the MSP via a human-in-the-loop hybrid intelligent approach, humans must be included in the scheduling process, *e.g.*, for preference elicitation or permission in exceptional situations. However, not bothering the human too much is essential for any system to achieve advantages over conventional meeting scheduling methods.

- **Trust and acceptance**: If humans do not trust that the agent will properly schedule their meetings, adoption will be compromised. We note that asking for too little input from humans might be detrimental to trust in the system and the quality of its solutions.

- **Computational cost**: Considering the complexity of the many variants of the MSP, it is important to pay attention to the computational cost incurred by systems for solving this problem.

7.3 NEGOTIATION IN MEETING SCHEDULING

As mentioned in Section 7.1, we can distinguish between market-based and negotiation-based approaches to solving the MSP from a multi-agent perspective. Market-based approaches assume that agents are self-interested [45] and are generally based on the ideal that fairness (*e.g.*, maximum social welfare) can be guaranteed through mechanism design, where the goal is to design a mechanism that satisfies both the incentive-compatible (IC) property (*i.e.*, agents are truthful about preferences) and the individually rational (IR) property (*i.e.*, you cannot receive a negative pay-off from the mechanism). Some success has been achieved using, for example, Clarke tax [30, 44] under simplified conditions, which do not hold up in real-world applications. Designing effective mechanisms for real-world multi-agent systems is theoretically challenging [34].

We argue that negotiation-based approaches are a good fit for the MSP. Firstly, a negotiation approach fits naturally with how humans agree on meeting times. Delegating the legwork to AI agents does not interfere with this and would enable an effective hybrid intelligent solution to this problem, where human capability is extended with AI. Secondly, negotiation is distributed in nature and does not *per se* require a trusted central authority. Thirdly, in negotiation, the practice is only to reveal information on a need-to-know basis, which promotes privacy and is part of the responsibility by-design approach we subscribe to [42].

If we do not require participants to reveal all their preferences and constraints and allow multiple scheduling attempts, then we are basically in a negotiation setting. This is how humans schedule meetings without tooling, often via email, which is cumbersome and inefficient. Tools like Doodle and When2meet can be considered single-shot negotiations [2] as they eliminate the multiple-round component while lowering participants' privacy. We believe negotiation methods make the most sense as we aim for multiple-round, privacy-preserving scheduling.

7.3.1 NEGOTIATION PROTOCOLS

Agents must communicate with each other to find agreements. Open communication with other agents in the form of "cheap talk" [35] or with humans in the form of natural language is possible but renders the problem more complex. We deem it more efficient to use negotiation protocols to aid the negotiation process in finding cooperative solutions. Such protocols restrict the type of messages and order in which they are sent [143].

We are not the first to propose negotiation for MSP; examples of proposed protocols for MSP are the single proposer mechanism (SPM) [93] and the distributed score-based multi-round (DSM) negotiation mechanism [49].

7.3.2 HUMAN PREFERENCES

Agents representing humans in negotiations should attempt to optimise outcomes based on human preferences. We, therefore, consider preference elicitation and estimation ([27, 154, 153, 138]) as core components in negotiation-based approaches for solving the MSP.

In general, a preference model can be bootstrapped based on available historical data in the form of preference pairs through direct preference optimisation (DPO) [116]. In the case of MSP, it can be based on historical calendar data [85] and on the current state of the calendar [33]. Estimating the preferences of other humans can help in finding mutually beneficial outcomes. Opponent modelling techniques can be used to estimate these preferences while negotiating, see *e.g.*, [12].

7.3.3 LEARNING TO NEGOTIATE

Given a protocol and preference profile, agents need to learn how to negotiate with other agents, focusing on maximising individual utility, optimising for cooperativeness (*e.g.*, social welfare), or a mixture of those, depending on the characteristics of the MSP at hand. Such agents can be trained using, *e.g.*, automated algorithm configuration [124] or reinforcement learning [19, 137, 97].

In MSP, one can assume that the environment is highly dynamic. New agents will be encountered, other agents will change their behaviour, human preferences over preferred slots will change, etc. Optimising performance means that continuous adaptation is required. An agent's policy can be retrained at fixed times based on historical interactions [95]. After training on a dataset, an agent can be guided by expert annotations to improve exploration online [90].

7.4 DISCUSSION

The MSP is a challenging problem. Finding optimal Nash equilibrium solutions in such cooperative AI problems is known to be NP-hard [61, 32, 31]. We therefore believe emphasis should be placed on finding solutions that are good enough, but not necessarily optimal, to avoid the need for exponential time solvers. Finding sufficiently good solutions also avoids problems caused by agents aiming to maximise utility deviating or rescheduling for minuscule improvements, which hurts mutually beneficial cooperation [119].

Another important point concerns interaction with humans that are not represented by agents. In the adoption of automated meeting scheduling systems, there will be a transition period during which some humans are represented by agents and others are not. Communication methods change and humans are likely to be less responsive, both in terms of the frequency of interaction and response time. Naturally, agents need to consider this in their scheduling behaviour.

We also have to ensure a degree of fairness in such systems. It cannot be the case that the calendar of some users will be inefficient or that they are being exploited by other agents, simply because they are not properly represented by their agent. Extra care must be taken when a group of agents is dealing with a single participant who is not represented by an agent. A lack of scheduling capabilities should not

lead to a drastically less desirable outcome compared to other participants.

If we solve this problem, a societal implication is that humans might change their view on appointments as being somewhat more fluent than is currently the case. Whether this is net beneficial remains to be seen, but an effort should be made to be alert to potential negative side effects.

Finally, we reiterate that, in our view, the time is right to take on this challenge. Recent interest and advances in dealing with human preferences, aligning AI systems, cooperative AI and multi-agent systems can all come together within the domain of meeting scheduling. After a long period of off-and-on attention, the tools might now be available to tackle this problem in a manner that brings substantial benefits to the many individuals who have to regularly schedule meetings and to their organisations.

III

CONCLUSIONS

8

Conclusion

This dissertation investigated the design and evaluation of automated negotiation agents, with a particular focus on learning capabilities and performance across diverse negotiation settings. In this final chapter, we revisit and answer the research questions from Section 8.1. We discuss our results in Section 8.2 and discuss its limitations and suggestions for future work in Section 8.3.

8.1 RESEARCH QUESTIONS & CONTRIBUTIONS

We now address the research questions from Section 1.2 per question. We will reiterate the questions and their sub-questions on the design of learned negotiation strategies and the evaluation of such agents, and answer them.

RESEARCH QUESTION 1

Q1 How do we design agents that can learn to negotiate?

SQ1.1 How can we reduce human-induced biases and conceptual complexity in learned negotiation strategies?

SQ1.2 Can we learn to generalise over diverse negotiation instances?

Our work has demonstrated significant progress in creating agents that can learn to negotiate, using a series of approaches, namely automated algorithm configuration (Chapter 3), portfolio selection methods (Chapter 4), and reinforcement learning (Chapter 5).

This dissertation makes a substantial contribution towards reducing the reliance on manual human design in developing negotiation agents. Approaches involving manual strategy design inherently introduce biases, for example, through the selection of specific algorithms or the creation of feature representations, which can limit agent capabilities or lead to information loss. By employing machine learning and optimisation methods, as explored in this dissertation, substantial aspects of the strategy design process can be automated, thereby mitigating these human-imposed biases. This leads to a reduction of human-induced biases in the negotiation strategies but it does not eliminate them entirely (e.g., the selection of the learning algorithm itself remains a human decision). Furthermore, integrating such learning methods into agents can increase the conceptual complexity of obtaining effective negotiation strategies.

In Table 8.1, we subjectively compare the human-induced bias and conceptual complexity of the methods proposed in this dissertation relative to each other. We also added manually designed strategies, which have a high degree of human-induced bias, as their entire logic is explicitly encoded by developers, limiting the strategy space explored. Their conceptual complexity is low, as such agents are generally simple heuristic-based strategies with limited capabilities.

The algorithm configuration approach (Chapter 3) reduces the bias somewhat but still has a medium-high degree of bias because the parameterised agent and the instance features guiding configuration remain manually designed. We classify

Table 8.1: Relative comparison of the design of negotiation agents presented in this dissertation.

Chapter	Main method	Human-induced bias	Conceptual complexity
-	Manual design	high	low
Chapter 3	Algorithm configuration	medium-low	medium-high
Chapter 4	Algorithm selection	medium-high	high
Chapter 5	Reinforcement learning	low	medium-low

its conceptual complexity as medium-low, due to the combination of the parameterised agent with the SMAC [75] automated configuration algorithm.

In Chapter 4, the portfolio selection method lowers the bias further to medium-high. While still utilising the parameterised agent, the portfolio construction (using Hydra [162]) and strategy selection (using AutoFolio [100]) broadens the strategy space of the agent. However, this layering of methods results in high conceptual complexity due to the need to manage multiple sophisticated methods.

Finally, the end-to-end reinforcement learning approach (Chapter 5) has a low degree of human-induced bias. By learning directly from negotiation interactions using GNNs, it avoids the manual design of feature representations, but still has some bias in the design of the policy architecture. We classify its conceptual complexity as medium-low, as it operates as a single, unified learning framework integrating various negotiation strategy challenges within the learned policy. We further discuss the conceptual complexity and human-induced bias in Section 8.2.

We have shown that it is possible to develop agents capable of learning to negotiate on a diverse set of negotiation scenarios and opponents, thus answering SQ1.2. This is a substantial improvement over previous attempts at learning negotiation agents that were trained and tested in strictly scoped settings, which hurts generalisability. We have shown that the performance of learned strategies transfers to unseen opponents in Chapter 3, Chapter 4, and Chapter 5. We have also shown that performance transfers to unseen negotiation scenarios both on composed sets (Chapter 3, Chapter 4) as well as on randomly generated scenarios (Chapter 5).

RESEARCH QUESTION 2

Q2 Is there a uniform way of evaluating negotiation agents?

- **SQ2.1** Is there a single-best metric?
- **SQ2.2** What is the value of the average utility metric for evaluating negotiation agents?

Our investigation into evaluation methodologies for negotiation agents, detailed in Chapter 6, revealed the limitations of relying on single-best metrics. We examined several commonly used metrics and found that agent rankings often varied depending on the specific criterion chosen (e.g., individual utility versus social welfare), demonstrating the lack of a single consistent metric. Critically, answering SQ2.2, the commonly used average utility metric in automated negotiation

literature and competitions is not reliable, as the average utility cannot represent non-transitive rankings and is influenced by the composition of the group that the agent is evaluated in (Section 6.7.2). Both are undesirable properties in the context of automated negotiation.

In answer to SQ2.1, regarding the possibility of discovering a single-best metric, our work suggests this is improbable. Negotiation is complex, involving trade-offs between individual utility, social welfare, Pareto efficiency, and other goals. The observed non-transitivity implies that agent performance cannot always be linearly ordered, which is required for ranking by a single scalar metric. Therefore, the challenge might be less about finding the right metric and more about recognising the complexity of negotiation strategy evaluation.

8.2 DISCUSSION

With the research questions addressed, we now contextualise the findings of this dissertation, beginning with a discussion. In addition to the results obtained, there are further insights and lessons learned that are worth discussing. We discuss the challenges in building learning negotiation agents and the implications of achieving such agents. We also discuss the difficulty of evaluating negotiation agents.

REDUCING HUMAN-INDUCED BIAS

A central theme of this dissertation has been the progressive reduction of human-induced biases in negotiation strategy learning. Our trajectory from manually crafted parameterised negotiation strategies in combination with algorithm configuration (Chapter 3) and portfolio selection (Chapter 4) to end-to-end reinforcement learning (Chapter 5) exemplifies this principle. It is our opinion that human-induced bias through manual design should be limited. Such biases reduce the reachable subspace of the total negotiation strategy space that is used for learning/optimising a strategy. High-performing strategies might not even be part of the strategy space that is considered.

Chapter 3 and Chapter 4 both present a parameterised agent and negotiation instance features that were manually designed as prerequisites for the developed methods. Both were designed based on task-specific knowledge and past research, which gave them merit. As the findings in Chapter 6 exemplify, negotiation dynamics are so complex that these are not easily understood by human experts. Chapter 3 suffers most of this effect, as only a single fixed strategy is obtained. In contrast, Chapter 4 improves upon this by allowing for a portfolio of complementary strategies to be used, thus widening broadening the possible strategy space. However, we think that there is room for improvement in terms of further decreasing the degree of human-induced bias. We achieve this by removing manual feature design and using deep learning methods instead.

We can draw a parallel here with the development of the research areas of computer vision and natural language procession, where, in the early days, there was a large focus on manually designed features and tools such as SIFT features [101] and bag-of-words models [66]. With the advent of deep learning, such manually designed methods have been dominated by deep learning methods, which are

capable of learning complex relations not captured through manual design. Our research shows that the automated negotiation research area can also benefit from making this step by demonstrating that agents can learn to negotiate without relying on manually designed features. The graph-based reinforcement learning approach presented in Chapter 5 represents a significant advancement in this direction, allowing negotiation strategies to emerge with minimal human-induced bias.

CONCEPTUAL COMPLEXITY

In our definition, conceptual complex methods layer several learning methods, optimisation methods, or manually designed algorithms. Note that we deviate from the conceptual complexity definition that is used in psychology as a personality variable reflecting information processing tendencies [145]. Conceptual complexity generally makes methods both difficult to reproduce and hard to understand. From the perspective of Occam's razor, layering algorithmic methods increases the number of entities perhaps beyond necessary, whereas conceptually simpler, more unified approaches are preferable if they achieve comparable results. Conceptual simplicity, in line with the principle of parsimony (Occam's razor), is essential. It improves reproducibility and lowers the barrier for understanding, facilitating critique, extension, and further research.

The commonly used negotiation framework General Environment for Negotiation with Intelligent multi-purpose Usage Simulation (GENIUS) [99], many of the developed strategies in the automated negotiation community are hard to reproduce, especially the strategies that are learned or optimised. Many strategies entail various techniques stacked on top of each other (e.g. Sengupta et al. [137], Bagga et al. [18], and Chen et al. [29]). Part of this originates from the common practice of designing strategies to be modular (see, e.g., the BOA framework [13]), and part of it is due to the difficulty of dealing with differently sized scenarios, causing actions to be abstracted to utility goals commonly. This adds an additional search problem, as negotiation agents must find outcomes that fit the utility goals (also discussed in Chapter 5).

In retrospect, we also propose two conceptually complex methods in Chapter 3 and Chapter 4. Where Chapter 3 already combines algorithm configuration methods, a parameterised agent, an opponent preference estimation method, a random search method, a feature logging mechanism, and the negotiation platform GENIUS [99], Chapter 4 adds portfolio construction methods and an algorithm selection method. The work in both chapters can be seen as initial steps towards learned negotiation strategies, but it would be challenging to further advance those lines of research. The work in Chapter 5 represents a substantial step towards conceptual simplicity: It combines all the traditionally considered separate problems, such as opponent preference estimation, strategy learning, and outcomes space searching, into a single reinforcement learning problem on a clearly defined and close to raw observation/action space. Chapter 5 opens up many new avenues for further research, which we discuss in the next paragraph.

The discussed BOA structure aligns with the long-standing "divide-and-conquer" tradition in AI, i.e., to partition complex problems into more manageable subproblems. This makes such problems easier to understand and developed methods

better explainable. However, the RL approach presented in this dissertation (Chapter 5) deliberately steps away from this, treating the negotiation problem holistically. This approach prioritises simplicity, in the form of a unified learning process, over the intrinsic explainability of divide-and-conquer approaches. The question is whether this trade-off is desirable. With the advancement of the automated negotiation research area, strategies are becoming more complex, challenging explainability regardless of whether a divide-and-conquer or holistic approach is used. Recent work, for example, took a divide-and-conquer approach while using RL to solve sub-problems [19, 18]. We argue that the research area might need to accept that negotiation strategies for complex negotiation tasks will likely be hard to explain regardless of the chosen approach. Therefore, stepping away from this divide-and-conquer approach and focusing research on conceptually simple methods is a potentially more productive path forward for the automated negotiation community, even if it leads to challenges in understanding the resulting agent behaviours.

REINFORCEMENT LEARNING FOR NEGOTIATION AGENTS

As discussed previously, the reinforcement learning-based strategy learning method presented in Chapter 5 is conceptually simple and less affected by human-induced bias. It integrates many of the problems faced in a negotiation game into a single-policy learning problem. This is in contrast with the complexity of the modular methods in the literature (see, e.g., [13, 19]), as we already discussed in Section 8.2. A further benefit is that more elaborate strategies can be achieved by extending the policy network, where, in modular approaches, this is achieved by adding another module to the agent. Learning to negotiate will then simply remain a single policy learning problem, which might be more difficult to learn, but only adds to the complexity by including more trainable neural network layers. We think that reinforcement learning is a fruitful direction for learning negotiation strategies based on this flexibility.

PROGRESS IN NEGOTIATION STRATEGY DEVELOPMENT

The lack of a universally applicable evaluation metric for negotiation agents remains a significant obstacle in the progress of the research area. Our analysis and discussion in Chapter 6 concluded that there currently is no clear evaluation method for negotiation agents and that the most commonly used metric, average individual utility, has problematic characteristics. The standard procedure of developing such agents is to take a set of top-performing agents of the Automated Negotiating Agents Competition (ANAC) or literature and to create an agent that outperforms them, often based on average utility. Such newly developed state-of-the-art agents are then included in benchmarks for future work. However, we discussed in Section 6.7.2 that average utility performance in a tournament with a set of benchmark agents depends on the composition of that set. Adding a well-performing agent to a benchmark set does not per se improve the benchmark but merely changes it. Any newly designed agent is, therefore, not strictly better, but just specialised on the new benchmark. We argue that not much progress is made in negotiation strategy

design using this approach. As a community, we need to step outside the boundaries of evaluation methods and tasks that we have considered to make meaningful progress.

We think that research into automated negotiation application areas is the way forward. A clear task specification helps in defining the negotiation setting, such that performance criteria might change. In Chapter 7, we proposed multi-agent calendar scheduling as a concrete, real-world application area for automated negotiation. This problem encapsulates many of the challenges faced in negotiation research. It serves as an excellent example of a task-specific challenge that can potentially drive progress in the research area. We might find out that performance criteria are always different per agent, or that we should not care much about determining "winners" in groups of negotiation agents. We encourage the community to adopt this or similar challenge problems to drive progress, similar to how benchmark datasets like ImageNet [41] have driven progress in computer vision.

EMPIRICAL EXPERIMENTATION AT SCALE

The automated negotiation community needs to mature in its approach to large-scale experimentation. Our work, particularly in part one and Chapter 6, goes beyond what is commonly seen in the automated negotiation literature in terms of the scale of training and evaluation iterations. Learning negotiation strategies on large and diverse sets of negotiation instances is computationally intensive but also important in the pursuit of generalisability. This also shifts the burden from manual algorithm design to computational resources, aligning with the increasing availability of compute in recent years. The added benefit of such learning methods is the partial removal of human bias in the negotiation strategies. Furthering this line of research requires training at scale.

Scaling is a challenge in evaluating automated negotiation agents. The Automated Negotiation League (ANL) faces the curse of dimensionality in its tournaments. For instance, a bilateral tournament with 20 agents and 10 negotiation scenarios requires 1 900 evaluations, excluding repetitions commonly performed to minimise stochastic influence. To manage this, competition editions often adopted a two-phase approach: an initial group phase followed by a finalists phase. While this reduces the number of evaluations required, it introduces a new problem. As shown in Chapter 6, group composition significantly influences agent rankings. Consequently, the final rankings are affected by the initial, arbitrary group divisions, introducing undesirable inconsistency in the evaluation process.

This brings us to our statement that the automated negotiation community needs to mature in experimentation at scale and use the abundance of computing power that has become readily available in recent years. This is a prerequisite to improve the evaluation of negotiation agents, as well as to further the line of research into learning negotiation agents.

COOPERATION AND NEGOTIATION

Chapter 1 opened with the statement that human success is due to our ability to cooperate, which is partially enabled by our ability to negotiate. However, negotiation agents in the automated negotiation community often show limited cooperative

behaviour. Many agents are designed to optimise their individual utility, which is supported by individual utility being one of the main evaluation criteria in the ANAC leagues (including the editions that we organised). The behaviour frequently comes at the expense of finding mutually beneficial outcomes that would maximise social welfare and causes many negotiations to end without an agreement. For example, approximately 11% of the negotiations ended in no agreement for the top performing agent in Table 6.4. This myopic approach is particularly problematic in mixed-motive scenarios where there is potential for joint gains. As we demonstrated in Chapter 6, in a tournament of competitive agents, the social welfare of the group is sub-optimal due to negotiations ending without an agreement. This is wasteful and undesirable for some potential applications, such as energy grid management, where prosumers must cooperate to distribute energy effectively.

How to move away from this competitive behaviour is, however, unclear. Agents can specifically be designed to optimise for social welfare, but that makes them sensitive to exploitation. After all, in a group of cooperators, it is often beneficial to be a defector [112]. This is already a challenge for groups of humans, but it is an even more significant challenge in groups of artificial agents, as they are insensitive to social constructs and norms. In other words, negotiation agents can afford to negotiate more aggressively than we think human negotiators will be comfortable with, which is a problem if cooperation is ultimately the goal.

8.3 LIMITATIONS & FUTURE WORK

Aside from the contributions of this dissertation to the research area of automated negotiation, there are also limitations to our work, as well as potential directions for future work. We discuss both in this section many of which follow from Section 8.2.

SIMPLIFICATION OF NEGOTIATION INSTANCES

Our studies primarily focused on bilateral negotiations with linear utility functions and categorical issues. While these scenarios are common in the literature and provide a good starting point, they do not capture the full complexity of real-world negotiations. Multi-party negotiations, negotiations with continuous issues, and scenarios with non-linear utility functions or interdependencies between issues were not extensively explored. This is a limitation as such additional complexity is common. For example, in the calendar scheduling task, there are often more than two parties involved in setting meetings, time is continuous, and issues such as location and availability can be connected.

LIMITED OPPONENT DIVERSITY

Although we used a variety of opponent strategies in our experiments, including those from ANAC competitions, this set may not fully represent the diversity of strategies encountered in real-world negotiations. Human negotiators, in particular, may employ tactics and exhibit behaviours that are not well-represented by our current set of artificial opponents. Evaluating against a broader range of opponents, including human negotiators, would provide a more comprehensive assessment of agent performance. As a result, the performance of the agents developed in this

dissertation could be unpredictable in real-world situations. Future work could train negotiation agents using self-play, which avoids the requirement for a set of opponent agents and has seen previous successes in, e.g., the game of Go [142].

PREFERENCES OF AGENTS

We assumed that preferences were clearly defined in every negotiation scenario by means of the utility function. However, eliciting human preferences and capturing them in a model (or utility function) is difficult and a research area by itself [27]. The assumption of having such a clearly defined cardinal ranking over the entire outcome space is likely to be unrealistic. We also assumed that preferences won't change over time but are static through a negotiation session. This does not have to be the case, as negotiations might stretch out over longer periods when situations change.

Unclear preferences would add significant challenges to developing negotiation strategies, but should be studied in future work, as they are realistic. For example, in the calendar scheduling task, preferences are more typically expressed as partial constraints ("busy Tuesday morning"), ordinal statements ("prefer earlier slots"), or fuzzy priorities. Furthermore, the static preference assumption directly contradicts the dynamic reality of calendars, as availability changes and meeting importance shifts.

LIMITED EXPLORATION OF MULTI-AGENT DYNAMICS

Our research primarily focused on the performance of individual agents rather than the emergent behaviours in multi-agent systems where all participants are learning agents. The long-term dynamics and potential equilibria in such systems were touched upon in Chapter 6, but not extensively studied. This is an important area for future research, as it could reveal insights into the stability and efficiency of negotiation strategies in more realistic, adaptive environments.

LACK OF INTERPRETABILITY

Particularly in our end-to-end reinforcement learning approach, the learned strategies can be difficult to interpret or explain. This lack of interpretability could be a barrier to adoption in tasks where transparency and accountability are important. Future work should explore methods for making learned negotiation strategies more interpretable. While achieving interpretability for deep learning models remains an open research area across AI, progress is needed for reinforcement learning based negotiation agents. Future work could investigate techniques from explainable AI [108] to explore methods that extract simplified representations, such as knowledge technology-based models that capture simple aspects of the learned strategy. We realise that complete transparency might be unrealistic, but simplified high-level insight could be acceptable.

EVALUATION METRICS

Future work should focus on developing more robust and comprehensive evaluation metrics for negotiation agents. As demonstrated in Chapter 6, current metrics such as average utility have significant limitations, particularly in their sensitivity to

opponent groups and inability to capture non-transitive relationships. Directions could be exploring multi-criteria evaluation frameworks that consider factors beyond just utility, such as fairness, stability of outcomes, and adaptability to different negotiation settings. This could provide a more holistic assessment of agent performance. Additionally, research into game-theoretic metrics that are more suitable for handling general-sum games could yield valuable insights.

COOPERATION

As discussed in Section 8.2, the level of cooperativeness is lacking in negotiation agents, as they are often designed to maximise individual utility without much risk for repercussions. Negotiation agents often have restricted communication capabilities, preventing them from engaging in the rich, nuanced dialogue that facilitates human cooperation. There is a lack of sophisticated mechanisms for building and maintaining trust, which is crucial for fostering cooperation, especially in repeated interactions or when dealing with partial information. There is a general lack of meta-strategic reasoning, with agents typically not engaging in higher-level thinking about the broader context of negotiations or the impact of their actions on future interactions. Note that implementing a memory mechanism to detect and reciprocate uncooperative behaviour might not be enough to enforce cooperative behaviour. Sometimes, individuals need to be excluded from the group based on the experiences of others to foster cooperation. Addressing these limitations represents a significant opportunity for future research to develop more cooperative negotiation agents. A potential starting point would be implementing reciprocity mechanisms that foster cooperation, such as described by Nowak [112].

REINFORCEMENT LEARNING IN NEGOTIATION AGENTS

The application of reinforcement learning in automated negotiation research remains relatively uncommon, with only a few notable attempts [19, 137, 18, 97]. While the challenge of managing variable-sized action spaces may have contributed to this scarcity, we addressed this issue in Chapter 5. The introduction of single policy-based agents now allows for the integration of multiple learning problems rather than the potentially noisy stacking of learning methods. This development opens up new possibilities and avenues for future work. Based on the policy proposed in Chapter 5, we suggest several further advancements.

- The policy network could be modified to include the full history in the observation (using, e.g., recurrent nets or transformers), potentially enabling it to identify and adapt to different opponent types, a capability we have shown to improve performance (Chapter 4).
- The loss function can include information that is typically unavailable during evaluation. For instance, the loss can be set to maximise social welfare, which requires knowledge of the opponent's utility function. This approach could help steer agents towards agreements with higher social welfare. The loss function can also be modified to explicitly incorporate the task of learning the opponent's utility function. Explicitly adding this task could improve the

agent's capabilities of finding mutually beneficial agreements. Note that this is only possible if we train the agent in simulations with full observability.

- To explore the possibility of handling continuous actions with our proposed policy network, the node outputs can be interpreted directly as bounded continuous distributions. Some negotiation objectives, such as price, are inherently continuous and would be better represented by continuous actions. This could also reduce the number of trainable parameters.

We think that the first point, including a history in the policy input, is a promising direction for future work. As demonstrated in Chapter 5, the reinforcement learning agent currently struggles to effectively adapt its strategy to diverse opponent types. Enabling the policy to learn from the history of interactions could directly address this limitation, which would be a significant step towards adaptive negotiation agents.

8.4 CLOSING REMARKS

In the introduction of this dissertation, we began by reflecting on Yuval Harari's insight from "Sapiens" that humans' unique ability to cooperate flexibly and at scale is what sets us apart and has led to our dominance as a species. Negotiation, as a special form of communication, plays a crucial role in enabling this cooperation. Throughout this work, we have explored how to create artificial agents capable of negotiating effectively and mirroring, in some ways, the development of this essential human skill. From the automated configuration of negotiation strategies to portfolio-based approaches and end-to-end reinforcement learning, we have taken notable steps to develop more flexible and adaptive negotiating agents.

As we conclude, it is worth considering how the advancement of negotiating AI agents might impact human cooperation and society at large. Just as the development of human negotiation skills enabled more complex forms of collaboration and social organisation, sophisticated AI negotiators could potentially facilitate new forms of AI-AI cooperation and AI-human cooperation at an unprecedented scale. However, as our analysis of evaluation metrics and the challenges in multi-agent dynamics reveal, there is still much to learn about the creation of effective negotiation strategies. As we continue to develop AI that can negotiate, we must ensure that it enhances rather than diminishes cooperative capabilities.

BIBLIOGRAPHY

[1] Kobi (Ya'akov) Gal and Litan Ilany. "The Fourth Automated Negotiation Competition". In: *Next Frontier in Agent-Based Complex Automated Negotiation*. Studies in Computational Intelligence. Tokyo: Springer Japan, 2015, pp. 129–136. ISBN: 978-4-431-55525-4 (cit. on p. 67).

[2] D. Anbar and E. Kalai. "A one-shot bargaining problem". In: *International Journal of Game Theory* 7.1 (Mar. 1, 1978), pp. 13–18. ISSN: 1432-1270. DOI: 10.1007/BF01763116 (cit. on p. 99).

[3] Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. "A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms". In: *Principles and Practice of Constraint Programming - CP 2009*. Ed. by Ian P. Gent. Berlin, Heidelberg: Springer, 2009, pp. 142–157. ISBN: 978-3-642-04244-7. DOI: 10.1007/978-3-642-04244-7_14 (cit. on p. 17).

[4] Robert Axelrod. "The Emergence of Cooperation among Egoists". In: *American Political Science Review* 75.2 (June 1981), pp. 306–318. ISSN: 0003-0554, 1537-5943. DOI: 10.2307/1961366 (cit. on p. 14).

[5] Reyhan Aydoğan, Tim Baarslag, Katsuhide Fujita, Holger H. Hoos, Catholijn M. Jonker, Yasser Mohammad, and Bram M. Renting. "The 13th International Automated Negotiating Agent Competition Challenges and Results". In: *Recent Advances in Agent-Based Negotiation: Applications and Competition Challenges*. Ed. by Rafik Hadfi, Reyhan Aydoğan, Takayuki Ito, and Ryuta Arisaka. Studies in Computational Intelligence. Singapore: Springer Nature, 2023, pp. 87–101. ISBN: 978-981-9905-61-4. DOI: 10.1007/978-981-99-0561-4_5 (cit. on pp. 3, 54).

[6] Reyhan Aydoğan, Tim Baarslag, Katsuhide Fujita, Johnathan Mell, Jonathan Gratch, Dave de Jonge, Yasser Mohammad, Shinji Nakadai, Satoshi Morinaga, Hirotaka Osawa, Claus Aranha, and Catholijn M. Jonker. "Challenges and Main Results of the Automated Negotiating Agents Competition (ANAC) 2019". In: *Multi-Agent Systems and Agreement Technologies*. Ed. by Nick Bassiliades, Georgios Chalkiadakis, and Dave de Jonge. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 366–381. ISBN: 978-3-030-66412-1. DOI: 10.1007/978-3-030-66412-1_23 (cit. on pp. 66, 68).

[7] Reyhan Aydoğan, David Festen, Koen V. Hindriks, and Catholijn M. Jonker. "Alternating Offers Protocols for Multilateral Negotiation". In: *Modern Approaches to Agent-based Complex Automated Negotiation*. Ed. by Katsuhide Fujita, Quan Bai, Takayuki Ito, Minjie Zhang, Fenghui Ren, Reyhan Aydoğan,

and Rafik Hadfi. Studies in Computational Intelligence. Cham: Springer International Publishing, 2017, pp. 153–167. ISBN: 978-3-319-51563-2 (cit. on p. 10).

[8] Tim Baarslag. "What to bid and when to stop". PhD thesis. Delft University of Technology, Sept. 18, 2014. 338 pp. (cit. on pp. 14, 25, 40).

[9] Tim Baarslag, Reyhan Aydoğan, Koen V. Hindriks, Katsuhide Fujita, Takayuki Ito, and Catholijn M. Jonker. "The Automated Negotiating Agents Competition, 2010–2015". In: *AI Magazine* 36.4 (Dec. 31, 2015), pp. 115–118. ISSN: 2371-9621. DOI: 10.1609/aimag.v36i4.2609 (cit. on pp. 3, 6).

[10] Tim Baarslag, Katsuhide Fujita, Enrico H. Gerding, Koen Hindriks, Takayuki Ito, Nicholas R. Jennings, Catholijn Jonker, Sarit Kraus, Raz Lin, Valentin Robu, and Colin R. Williams. "Evaluating practical negotiating agents: Results and analysis of the 2011 international competition". In: *Artificial Intelligence* 198 (May 1, 2013), pp. 73–103. ISSN: 0004-3702. DOI: 10.1016/j.artint.2012.09.004 (cit. on pp. 24, 40, 66, 81, 85).

[11] Tim Baarslag, Mark Hendrikx, Koen Hindriks, and Catholijn Jonker. "Predicting the Performance of Opponent Models in Automated Negotiation". In: *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 02*. WI-IAT '13. USA: IEEE Computer Society, Nov. 17, 2013, pp. 59–66. ISBN: 978-0-7695-5145-6. DOI: 10.1109/WI-IAT.2013.91 (cit. on pp. 26, 68).

[12] Tim Baarslag, Mark J. C. Hendrikx, Koen V. Hindriks, and Catholijn M. Jonker. "Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques". In: *Autonomous Agents and Multi-Agent Systems* 30.5 (Sept. 1, 2016), pp. 849–898. ISSN: 1573-7454. DOI: 10.1007/s10458-015-9309-1 (cit. on pp. 15, 68, 100).

[13] Tim Baarslag, Koen Hindriks, Mark Hendrikx, Alexander Dirkzwager, and Catholijn Jonker. "Decoupling Negotiating Agents to Explore the Space of Negotiation Strategies". In: *Novel Insights in Agent-based Complex Automated Negotiation*. Ed. by Ivan Marsa-Maestre, Miguel A. Lopez-Carmona, Takayuki Ito, Minjie Zhang, Quan Bai, and Katsuhide Fujita. Tokyo: Springer Japan, 2014, pp. 61–83. ISBN: 978-4-431-54758-7. DOI: 10.1007/978-4-431-54758-7_4 (cit. on pp. 109–110).

[14] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. "Effective acceptance conditions in real-time automated negotiation". In: *Decision Support Systems*. Automated Negotiation Technologies and their Applications 60 (Apr. 1, 2014), pp. 68–77. ISSN: 0167-9236. DOI: 10.1016/j.dss.2013.05.021 (cit. on pp. 26, 33, 38).

[15] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. "Towards a Quantitative Concession-Based Classification Method of Negotiation Strategies". In: *Agents in Principle, Agents in Practice*. Ed. by David Kinny, Jane Yung-jen Hsu, Guido Governatori, and Aditya K. Ghose. Berlin, Heidelberg: Springer,

2011, pp. 143–158. ISBN: 978-3-642-25044-6. DOI: 10.1007/978-3-642-25044-6_13 (cit. on p. 30).

[16] Tim Baarslag, Koen Hindriks, Catholijn Jonker, Sarit Kraus, and Raz Lin. "The First Automated Negotiating Agents Competition (ANAC 2010)". In: *New Trends in Agent-Based Complex Automated Negotiations*. Ed. by Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo. Berlin, Heidelberg: Springer, 2012, pp. 113–135. ISBN: 978-3-642-24696-8. DOI: 10.1007/978-3-642-24696-8_7 (cit. on pp. 3, 67).

[17] Tim Baarslag and Koen V. Hindriks. "Accepting optimally in automated negotiation with incomplete information". In: *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. AAMAS '13. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 6, 2013, pp. 715–722. ISBN: 978-1-4503-1993-5 (cit. on p. 14).

[18] Pallavi Bagga, Nicola Paoletti, and Kostas Stathis. "Deep Learnable Strategy Templates for Multi-Issue Bilateral Negotiation". In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '22. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 9, 2022, pp. 1533–1535. ISBN: 978-1-4503-9213-6 (cit. on pp. 41, 55, 109–110, 114).

[19] Jasper Bakker, Aron Hammond, Daan Bloembergen, and Tim Baarslag. "RL-BOA: A Modular Reinforcement Learning Framework for Autonomous Negotiating Agents". In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 8, 2019, pp. 260–268. ISBN: 978-1-4503-6309-9 (cit. on pp. 54–55, 100, 110, 114).

[20] Prasanna Balaprakash, Mauro Birattari, and Thomas Stützle. "Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement". In: *Hybrid Metaheuristics*. Ed. by Thomas Bartz-Beielstein, María José Blesa Aguilera, Christian Blum, Boris Naujoks, Andrea Roli, Günter Rudolph, and Michael Sampels. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 108–122. ISBN: 978-3-540-75514-2. DOI: 10.1007/978-3-540-75514-2_9 (cit. on p. 16).

[21] David Balduzzi, Karl Tuyls, Julien Perolat, and Thore Graepel. "Re-evaluating evaluation". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018 (cit. on p. 89).

[22] Richard Bellman and Robert Kalaba. *Dynamic Programming*. Academic Press, Feb. 11, 1966. 112 pp. ISBN: 978-0-12-084856-0 (cit. on p. 15).

[23] Florian Berger, Rolf Klein, Doron Nussbaum, Jörg-Rüdiger Sack, and Jiehua Yi. "A meeting scheduling problem respecting time and space". In: *GeoInformatica* 13.4 (Dec. 1, 2009), pp. 453–481. ISSN: 1573-7624. DOI: 10.1007/s10707-008-0053-4 (cit. on p. 95).

[24] K. G. Binmore. *Fun and Games: A Text on Game Theory*. D.C. Heath and Company, 1992. 602 pp. ISBN: 978-0-669-24603-2 (cit. on p. 10).

[25] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. "F-Race and Iterated F-Race: An Overview". In: *Experimental Methods for the Analysis of Optimization Algorithms*. Ed. by Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss. Berlin, Heidelberg: Springer, 2010, pp. 311–336. ISBN: 978-3-642-02538-9. DOI: 10.1007/978-3-642-02538-9_13 (cit. on p. 16).

[26] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. May 2, 2021. DOI: 10.48550/arXiv.2104.13478. arXiv: 2104.13478[cs,stat] (cit. on p. 5).

[27] Li Chen and Pearl Pu. *Survey of Preference Elicitation Methods*. EPFL, 2004 (cit. on pp. 100, 113).

[28] Siqi Chen and Gerhard Weiss. "An efficient automated negotiation strategy for complex environments". In: *Engineering Applications of Artificial Intelligence* 26.10 (Nov. 1, 2013), pp. 2613–2623. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2013.08.012 (cit. on p. 81).

[29] Siqi Chen, Jianing Zhao, Gerhard Weiss, Ran Su, and Kaiyou Lei. "An effective negotiating agent framework based on deep offline reinforcement learning". In: *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*. Uncertainty in Artificial Intelligence. PMLR, July 2, 2023, pp. 324–335 (cit. on p. 109).

[30] Edward H. Clarke. "Multipart Pricing of Public Goods". In: *Public Choice* 11 (1971), pp. 17–33. ISSN: 0048-5829 (cit. on p. 99).

[31] Vincent Conitzer and Caspar Oesterheld. "Foundations of Cooperative AI". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.13 (Sept. 6, 2023), pp. 15359–15367. ISSN: 2374-3468. DOI: 10.1609/aaai.v37i13.26791 (cit. on p. 100).

[32] Vincent Conitzer and Tuomas Sandholm. "New complexity results about Nash equilibria". In: *Games and Economic Behavior*. Second World Congress of the Game Theory Society 63.2 (July 1, 2008), pp. 621–641. ISSN: 0899-8256. DOI: 10.1016/j.geb.2008.02.015 (cit. on p. 100).

[33] E. Crawford and M. Veloso. "Learning dynamic preferences in multi-agent meeting scheduling". In: *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. International Conference on Intelligent Agent Technology, 2005. (IAT 2005). Compiegne, France: IEEE, Sept. 2005, pp. 487–490. ISBN: 0-7695-2416-8. DOI: 10.1109/IAT.2005.94 (cit. on p. 100).

[34] E. Crawford and M. Veloso. “Mechanism design for multi-agent meeting scheduling including time preferences, availability, and value of presence”. In: *Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004. (IAT 2004).* International Conference on Intelligent Agent Technology, 2004. (IAT 2004). Beijing, China: IEEE, Sept. 2004, pp. 253–259. ISBN: 0-7695-2101-0. DOI: 10.1109/IAT.2004.1342952 (cit. on pp. 94, 99).

[35] Vincent P. Crawford and Joel Sobel. “Strategic Information Transmission”. In: *Econometrica* 50.6 (1982), pp. 1431–1451. ISSN: 0012-9682. DOI: 10.2307/1913390 (cit. on p. 99).

[36] Allan Dafoe, Edward Hughes, Yoram Bachrach, Tantum Collins, Kevin R. McKee, Joel Z. Leibo, Kate Larson, and Thore Graepel. *Open Problems in Cooperative AI*. Dec. 15, 2020. DOI: 10.48550/arXiv.2012.08630 (cit. on pp. 2, 94).

[37] Panayiotis Danassis, Florian Wiedemair, and Boi Faltings. “Improving Multi-agent Coordination by Learning to Estimate Contention”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Twenty-Ninth International Joint Conference on Artificial Intelligence. Vol. 1. Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 125–131. DOI: 10.24963/ijcai.2021/18 (cit. on p. 96).

[38] Andreas Darmann, Edith Elkind, Sascha Kurz, Jérôme Lang, Joachim Schauer, and Gerhard Woeginger. “Group Activity Selection Problem”. In: *Internet and Network Economics*. Ed. by Paul W. Goldberg. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 156–169. ISBN: 978-3-642-35311-6. DOI: 10.1007/978-3-642-35311-6_12 (cit. on p. 96).

[39] Andreas Darmann, Edith Elkind, Sascha Kurz, Jérôme Lang, Joachim Schauer, and Gerhard Woeginger. “Group activity selection problem with approval preferences”. In: *International Journal of Game Theory* 47.3 (Sept. 1, 2018), pp. 767–796. ISSN: 1432-1270. DOI: 10.1007/s00182-017-0596-4 (cit. on p. 96).

[40] Dave De Jonge. “An Analysis of the Linear Bilateral ANAC Domains Using the MiCRO Benchmark Strategy”. In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. Thirty-First International Joint Conference on Artificial Intelligence {IJCAI-22}. Vienna, Austria: International Joint Conferences on Artificial Intelligence Organization, July 2022, pp. 223–229. ISBN: 978-1-956792-00-3. DOI: 10.24963/ijcai.2022/32 (cit. on pp. 74, 86).

[41] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009 IEEE Conference on Computer Vision and Pattern Recognition. June 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 111).

[42] Virginia Dignum. *Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way*. Artificial Intelligence: Foundations, Theory, and Algorithms. Cham: Springer International Publishing, 2019. ISBN: 978-3-030-30370-9 (cit. on p. 99).

[43] Garett Dworman, Steven O. Kimbrough, and James D. Laing. "Bargaining by artificial agents in two coalition games: a study in genetic programming for electronic commerce". In: *Proceedings of the 1st annual conference on genetic programming*. Cambridge, MA, USA: MIT Press, July 28, 1996, pp. 54–62. ISBN: 978-0-262-61127-5 (cit. on pp. 4, 24, 54).

[44] Eithan Ephrati and Jeffrey S. Rosenschein. "The Clarke tax as a consensus mechanism among automated agents". In: *Proceedings of the ninth National conference on Artificial intelligence - Volume 1*. AAAI'91. Anaheim, California: AAAI Press, July 14, 1991, pp. 173–178. ISBN: 978-0-262-51059-2 (cit. on p. 99).

[45] Eithan Ephrati, Gilad Zlotkin, and Jeffrey S. Rosenschein. "Meet your destiny: a non-manipulable meeting scheduler". In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. CSCW '94. New York, NY, USA: Association for Computing Machinery, Oct. 22, 1994, pp. 359–371. ISBN: 978-0-89791-689-9. DOI: 10.1145/192844.193049 (cit. on pp. 94, 99).

[46] Torsten Eymann. "Co-Evolution of Bargaining Strategies in a Decentralized Multi-Agent System". In: *symposium on negotiation methods for autonomous cooperative systems*. AAAI. Jan. 2001, pp. 126–134. ISBN: 978-1-57735-137-5 (cit. on pp. 4, 24, 54).

[47] Stefan Falkner, Aaron Klein, and Frank Hutter. "BOHB: Robust and Efficient Hyperparameter Optimization at Scale". In: *Proceedings of the 35th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 3, 2018, pp. 1437–1446 (cit. on p. 17).

[48] Peyman Faratin, Carles Sierra, and Nick R. Jennings. "Negotiation decision functions for autonomous agents". In: *Robotics and Autonomous Systems*. Multi-Agent Rationality 24.3 (Sept. 30, 1998), pp. 159–182. ISSN: 0921-8890. DOI: 10.1016/S0921-8890(98)00029-3 (cit. on pp. 3, 14).

[49] Farzaneh Farhadi and Nicholas R. Jennings. "A Faithful Mechanism for Incremental Multi-Agent Agreement Problems with Self-Interested and Privacy-Preserving Agents". In: *SN Computer Science* 2.4 (May 11, 2021), p. 272. ISSN: 2661-8907. DOI: 10.1007/s42979-021-00650-4 (cit. on pp. 96, 98–99).

[50] Shaheen Fatima, Sarit Kraus, and Michael Wooldridge. "The Negotiation Game". In: *IEEE Intelligent Systems* 29.5 (Sept. 1, 2014), pp. 57–61. ISSN: 1541-1672. DOI: 10.1109/MIS.2014.90 (cit. on p. 10).

[51] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. "Multi-issue negotiation under time constraints". In: *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. AAMAS '02. New York, NY, USA: Association for Computing Machinery,

July 15, 2002, pp. 143–150. ISBN: 978-1-58113-480-3. DOI: 10.1145/544741.544775 (cit. on p. 14).

[52] Matthias Feurer and Frank Hutter. "Hyperparameter Optimization". In: *Automated Machine Learning: Methods, Systems, Challenges*. Ed. by Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Cham: Springer International Publishing, 2019, pp. 3–33. ISBN: 978-3-030-05318-5. DOI: 10.1007/978-3-030-05318-5_1 (cit. on p. 15).

[53] Roger Fisher, William L. Ury, and Bruce Patton. *Getting to Yes: Negotiating Agreement Without Giving In*. Penguin, May 3, 2011. 150 pp. ISBN: 978-1-101-53954-5 (cit. on pp. 11–12).

[54] M. S. Franzin, F. Rossi, E. C. Freuder, and R. Wallace. "Multi-Agent Constraint Systems with Preferences: Efficiency, Solution Quality, and Privacy Loss". In: *Computational Intelligence* 20.2 (2004), pp. 264–286. ISSN: 1467-8640. DOI: 10.1111/j.0824-7935.2004.00238.x (cit. on p. 98).

[55] Maria Sole Franzin, E C Freuder, F Rossi, and R Wallace. "Multi-Agent Meeting Scheduling with Preferences: Efficiency, Privacy Loss, and Solution Quality". In: *Proceedings of the AAAI 2002 Workshop on Preferences in AI and CP: Symbolic Approaches*. Edmonton, Canada: AAAI Press, 2002, p. 8 (cit. on p. 94).

[56] Eugene C Freuder, Marius Minca, and Richard J Wallace. "Privacy/Efficiency Tradeoffs in Distributed Meeting Scheduling by Constraint-Based Agents". In: *Proceedings of the IJCAI-01 Workshop on Distributed Constraint Reasoning*. Seattle, USA: AAAI Press, 2001, p. 9 (cit. on p. 98).

[57] Katsuhide Fujita, Reyhan Aydoğan, Tim Baarslag, Koen Hindriks, Takayuki Ito, and Catholijn Jonker. "The Sixth Automated Negotiating Agents Competition (ANAC 2015)". In: *Modern Approaches to Agent-based Complex Automated Negotiation*. Ed. by Katsuhide Fujita, Quan Bai, Takayuki Ito, Minjie Zhang, Fenghui Ren, Reyhan Aydoğan, and Rafik Hadfi. Cham: Springer International Publishing, 2017, pp. 139–151. ISBN: 978-3-319-51563-2. DOI: 10.1007/978-3-319-51563-2_9 (cit. on p. 67).

[58] Katsuhide Fujita, Reyhan Aydoğan, Tim Baarslag, Takayuki Ito, and Catholijn Jonker. "The Fifth Automated Negotiating Agents Competition (ANAC 2014)". In: *Recent Advances in Agent-based Complex Automated Negotiation*. Ed. by Naoki Fukuta, Takayuki Ito, Minjie Zhang, Katsuhide Fujita, and Valentin Robu. Cham: Springer International Publishing, 2016, pp. 211–224. ISBN: 978-3-319-30307-9. DOI: 10.1007/978-3-319-30307-9_13 (cit. on p. 67).

[59] Niels van Galen Last. "Agent Smith: Opponent Model Estimation in Bilateral Multi-issue Negotiation". In: *New Trends in Agent-Based Complex Automated Negotiations*. Ed. by Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2012, pp. 167–174. ISBN: 978-3-642-24696-8 (cit. on pp. 26, 68).

[60] Leonardo Garrido and Katia Sycara. “Multi-Agent Meeting Scheduling: Preliminary Experimental Results”. In: *Proceeding of the Second International Conference on Multiagent Systems 1996*. The Second International Conference on Multiagent Systems 1996 (ICMAS-96). Kyoto, Japan: AAAI Press, Dec. 1996, p. 8 (cit. on p. 94).

[61] Itzhak Gilboa and Eitan Zemel. “Nash and correlated equilibria: Some complexity considerations”. In: *Games and Economic Behavior* 1.1 (Mar. 1, 1989), pp. 80–93. ISSN: 0899-8256. DOI: 10.1016/0899-8256(89)90006-7 (cit. on p. 100).

[62] Russell T. Graves, Zachariah E. Nelson, and Subhadeep Chakraborty. “A decentralized intersection management system through collaborative negotiation between smart signals”. In: *Journal of Intelligent Transportation Systems* 27.2 (Mar. 4, 2023), pp. 272–294. ISSN: 1547-2450. DOI: 10.1080/15472450.2021.2016405 (cit. on p. 3).

[63] Irene Greif. *PCAL: A Personal Calendar*. MIT LCS, 1982 (cit. on p. 94).

[64] Taha D. Güneş, Emir Arditi, and Reyhan Aydoğan. “Collective Voice of Experts in Multilateral Negotiation”. In: *PRIMA 2017: Principles and Practice of Multi-Agent Systems*. Ed. by Bo An, Ana Bazzan, João Leite, Serena Villata, and Leendert van der Torre. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 450–458. ISBN: 978-3-319-69131-2. DOI: 10.1007/978-3-319-69131-2_27 (cit. on p. 40).

[65] Yuval Noah Harari. *Sapiens: A Brief History of Humankind*. New York, NY, USA: Harper Perennial, 2015. ISBN: 978-0-06-231609-7 (cit. on p. 2).

[66] Zellig S. Harris. “Distributional structure”. In: *Word* 10 (1954), pp. 146–162. DOI: 10.1080/00437956.1954.11659520 (cit. on p. 108).

[67] He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. “Decoupling Strategy and Generation in Negotiation Dialogues”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2018. Ed. by Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2333–2343. DOI: 10.18653/v1/D18-1256 (cit. on p. 10).

[68] Ryota Higa, Katsuhide Fujita, Toki Takahashi, Takumu Shimizu, and Shinji Nakadai. “Reward-based negotiating agent strategies”. In: *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*. Vol. 37. AAAI’23/IAAI’23/EAAI’23. AAAI Press, Feb. 7, 2023, pp. 11569–11577. ISBN: 978-1-57735-880-0. DOI: 10.1609/aaai.v37i10.26367 (cit. on pp. 54–55, 59–60).

[69] Koen Hindriks and Dmytro Tykhonov. “Opponent modelling in automated multi-issue negotiation using Bayesian learning”. In: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 1*. AAMAS ’08. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 12, 2008, pp. 331–338. ISBN: 978-0-9817381-0-9 (cit. on p. 68).

[70] Koen V. Hindriks and Dmytro Tykhonov. “Towards a Quality Assessment Method for Learning Preference Profiles in Negotiation”. In: *Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis*. Ed. by Wolfgang Ketter, Han La Poutré, Norman Sadeh, Onn Shehory, and William Walsh. Lecture Notes in Business Information Processing. Berlin, Heidelberg: Springer, 2010, pp. 46–59. ISBN: 978-3-642-15237-5. DOI: 10.1007/978-3-642-15237-5_4 (cit. on p. 83).

[71] John Henry Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992. 236 pp. ISBN: 978-0-262-58111-0 (cit. on pp. 4, 27).

[72] Bryan Horling and Victor Lesser. “A survey of multi-agent organizational paradigms”. In: *The Knowledge Engineering Review* 19.4 (Dec. 2004), pp. 281–316. ISSN: 1469-8005, 0269-8889. DOI: 10.1017/S0269888905000317 (cit. on p. 2).

[73] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G. M. Araújo. “CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms”. In: *Journal of Machine Learning Research* 23.274 (2022), pp. 1–18. ISSN: 1533-7928 (cit. on p. 56).

[74] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stuetzle. “ParamILS: An Automatic Algorithm Configuration Framework”. In: *Journal of Artificial Intelligence Research* 36 (Oct. 30, 2009), pp. 267–306. ISSN: 1076-9757. DOI: 10.1613/jair.2861 (cit. on pp. 15–16).

[75] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. “Sequential Model-Based Optimization for General Algorithm Configuration”. In: *Learning and Intelligent Optimization*. Ed. by Carlos A. Coello Coello. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 507–523. ISBN: 978-3-642-25566-3. DOI: 10.1007/978-3-642-25566-3_40 (cit. on pp. iii, v, 4, 17, 28, 31, 107).

[76] Litan Ilany and Ya’akov Gal. “Algorithm selection in bilateral negotiation”. In: *Autonomous Agents and Multi-Agent Systems* 30.4 (July 1, 2016), pp. 697–723. ISSN: 1573-7454. DOI: 10.1007/s10458-015-9302-8 (cit. on pp. 29, 40, 42, 44–45, 61–62, 66).

[77] Litan Ilany and Ya'akov (Kobi) Gal. "The Simple-Meta Agent". In: *Novel Insights in Agent-based Complex Automated Negotiation*. Ed. by Ivan Marsa-Maestre, Miguel A. Lopez-Carmona, Takayuki Ito, Minjie Zhang, Quan Bai, and Katsuhide Fujita. Studies in Computational Intelligence. Tokyo: Springer Japan, 2014, pp. 197–200. ISBN: 978-4-431-54758-7 (cit. on pp. 40, 68).

[78] N. R. Jennings and A. J. Jackson. "Agent-based Meeting Scheduling: A Design and Implementation". In: *Electronics Letters* 31.5 (1995). In collab. with N. R. Jennings and A. J. Jackson, pp. 350–352. ISSN: 0013-5194 (cit. on p. 94).

[79] Nicholas R. Jennings, Peyman Faratin, Alessio R. Lomuscio, Simon Parsons, Michael Wooldridge, and Carles Sierra. "Automated Negotiation: Prospects, Methods and Challenges". In: *Group Decision and Negotiation* 10 (2001), pp. 199–215 (cit. on pp. 3, 10).

[80] Dave de Jonge. "A new bargaining solution for finite offer spaces". In: *Applied Intelligence* 53.23 (Dec. 1, 2023), pp. 28310–28332. ISSN: 1573-7497. DOI: 10.1007/s10489-023-05009-1 (cit. on p. 87).

[81] Dave de Jonge, Tim Baarslag, Reyhan Aydoğan, Catholijn Jonker, Katsuhide Fujita, and Takayuki Ito. "The Challenge of Negotiation in the Game of Diplomacy". In: *Agreement Technologies*. Ed. by Marin Lujak. Cham: Springer International Publishing, 2019, pp. 100–114. ISBN: 978-3-030-17294-7. DOI: 10.1007/978-3-030-17294-7_8 (cit. on p. 66).

[82] Dave de Jonge and Carles Sierra. "NB^3: a multilateral negotiation algorithm for large, non-linear agreement spaces with limited time". In: *Autonomous Agents and Multi-Agent Systems* 29.5 (Sept. 1, 2015), pp. 896–942. ISSN: 1573-7454. DOI: 10.1007/s10458-014-9271-3 (cit. on p. 62).

[83] Ehud Kalai and Meir Smorodinsky. "Other Solutions to Nash's Bargaining Problem". In: *Econometrica* 43.3 (1975), pp. 513–518. ISSN: 0012-9682. DOI: 10.2307/1914280 (cit. on pp. 77, 85).

[84] Ryohei Kawata and Katsuhide Fujita. "Meta-Strategy for Multi-Time Negotiation: A Multi-Armed Bandit Approach". In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 8, 2019, pp. 2048–2050. ISBN: 978-1-4503-6309-9 (cit. on p. 40).

[85] Donghyeon Kim, Jinhyuk Lee, Donghee Choi, Jaehoon Choi, and Jaewoo Kang. "Learning User Preferences and Understanding Calendar Contexts for Event Scheduling". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM '18. New York, NY, USA: Association for Computing Machinery, Oct. 17, 2018, pp. 337–346. ISBN: 978-1-4503-6014-2. DOI: 10.1145/3269206.3271712 (cit. on pp. 94, 100).

[86] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: International Conference on Learning Representations. Nov. 3, 2016 (cit. on pp. iii, v, 55).

[87] M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam. "Protocols for negotiating complex contracts". In: *IEEE Intelligent Systems* 18.6 (Nov. 2003), pp. 32–38. ISSN: 1941-1294. DOI: 10.1109/MIS.2003.1249167 (cit. on p. 11).

[88] Mark Klein and Stephen C. -Y. Lu. "Conflict resolution in cooperative design". In: *Artificial Intelligence in Engineering* 4.4 (Oct. 1, 1989), pp. 168–180. ISSN: 0954-1810. DOI: 10.1016/0954-1810(89)90013-7 (cit. on p. 3).

[89] Jens Kober, J. Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11 (Sept. 1, 2013), pp. 1238–1274. ISSN: 0278-3649. DOI: 10.1177/0278364913495721 (cit. on p. 18).

[90] Minae Kwon, Siddharth Karamcheti, Mariano-Florentino Cuellar, and Dorsa Sadigh. "Targeted Data Acquisition for Evolving Negotiation Agents". In: *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning. Virtual: PMLR, July 2021, pp. 5894–5904 (cit. on pp. 10, 55, 100).

[91] Marc Lanctot, Kate Larson, Yoram Bachrach, Luke Marris, Zun Li, Avishkar Bhoopchand, Thomas Anthony, Brian Tanner, and Anna Koop. *Evaluating Agents using Social Choice Theory*. Dec. 6, 2023. DOI: 10.48550/arXiv.2312.03121. arXiv: 2312.03121[cs] (cit. on pp. 89–90).

[92] Raymond Y. K. Lau, Maolin Tang, On Wong, Stephen W. Milliner, and Yi-Ping Phoebe Chen. "An evolutionary learning approach for adaptive negotiation agents: Research Articles". In: *International Journal of Intelligent Systems* 21.1 (Jan. 1, 2006), pp. 41–72. ISSN: 0884-8173 (cit. on pp. 24–25, 32, 38, 54).

[93] Hooyeon Lee. "Algorithmic and Game-theoretic Approaches to Group Scheduling". In: *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*. The 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014). Paris, France: IFAAMAS, 2014, pp. 1709–1710 (cit. on pp. 96, 99).

[94] Hooyeon Lee and Yoav Shoham. "Stable group scheduling". In: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. AAMAS '14. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 5, 2014, pp. 1347–1348. ISBN: 978-1-4503-2738-1 (cit. on pp. 94, 96).

[95] Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. "Deal or No Deal? End-to-End Learning of Negotiation Dialogues". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2017. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 2443–2453. DOI: 10.18653/v1/D17-1259 (cit. on pp. 10, 55, 100).

[96] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *Journal of Machine Learning Research* 18.185 (2018), pp. 1–52. ISSN: 1533-7928 (cit. on p. 17).

[97] Zun Li, Marc Lanctot, Kevin R. McKee, Luke Marris, Ian Gemp, Daniel Hennes, Paul Muller, Kate Larson, Yoram Bachrach, and Michael P. Wellman. *Combining Tree-Search, Generative Models, and Nash Bargaining Concepts in Game-Theoretic Reinforcement Learning*. Feb. 1, 2023. DOI: 10.48550/arXiv.2302.00797 (cit. on pp. 55, 100, 114).

[98] Austen Liao, Nicholas Tomlin, and Dan Klein. *Efficacy of Language Model Self-Play in Non-Zero-Sum Games*. June 26, 2024. DOI: 10.48550/arXiv.2406.18872. arXiv: 2406.18872[cs] (cit. on p. 10).

[99] 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". In: *Computational Intelligence* 30.1 (2014), pp. 48–70. ISSN: 1467-8640. DOI: 10.1111/j.1467-8640.2012.00463.x (cit. on pp. 3, 5, 17, 26, 38, 40, 58, 109).

[100] Marius Lindauer, Holger H. Hoos, Frank Hutter, and Torsten Schaub. "AutoFolio: An Automatically Configured Algorithm Selector". In: *Journal of Artificial Intelligence Research* 53 (Aug. 31, 2015), pp. 745–778. ISSN: 1076-9757. DOI: 10.1613/jair.4726 (cit. on pp. iii, v, 5, 40, 44, 107).

[101] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (Nov. 1, 2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94 (cit. on p. 108).

[102] Thomas W Malone, Kenneth R Grant, Franklyn A Turbak, Stephen A Brobst, and Michael D Cohen. "Intelligent information-sharing systems". In: *Communications of the ACM* 30.5 (May 1, 1987), pp. 390–402. ISSN: 0001-0782. DOI: 10.1145/22899.22903 (cit. on p. 94).

[103] Luke Marris, Marc Lanctot, Ian Gemp, Shayegan Omidshafiei, Stephen McAleer, Jerome Connor, Karl Tuyls, and Thore Graepel. *Game Theoretic Rating in N-player general-sum games with Equilibria*. Oct. 5, 2022. DOI: 10.48550/arXiv.2210.02205. arXiv: 2210.02205[cs] (cit. on p. 90).

[104] N. Matos, C. Sierra, and N.R. Jennings. "Determining successful negotiation strategies: an evolutionary approach". In: *Proceedings International Conference on Multi Agent Systems (Cat. No.98EX160)*. Proceedings International Conference on Multi Agent Systems (Cat. No.98EX160). July 1998, pp. 182–189. DOI: 10.1109/ICMAS.1998.699048 (cit. on pp. 4, 24).

[105] Alicia P. Melis, Brian Hare, and Michael Tomasello. "Chimpanzees coordinate in a negotiation game". In: *Evolution and Human Behavior* 30.6 (Nov. 1, 2009), pp. 381–392. ISSN: 1090-5138. DOI: 10.1016/j.evolhumbehav.2009.05.003 (cit. on p. 2).

[106] Johnathan Mell, Jonathan Gratch, Tim Baarslag, Reyhan Aydoğan, and Catholijn M. Jonker. "Results of the First Annual Human-Agent League of the Automated Negotiating Agents Competition". In: *Proceedings of the 18th International Conference on Intelligent Virtual Agents*. IVA '18. New York, NY, USA: Association for Computing Machinery, Nov. 5, 2018, pp. 23–28. ISBN: 978-1-4503-6013-5. DOI: 10.1145/3267851.3267907 (cit. on p. 66).

[107] META FUNDAMENTAL AI RESEARCH DIPLOMACY TEAM (FAIR), Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. "Human-level play in the game of Diplomacy by combining language models with strategic reasoning". In: *Science* 0.0 (Nov. 22, 2022), eade9097. DOI: 10.1126/science.ade9097 (cit. on p. 3).

[108] Dang Minh, H. Xiang Wang, Y. Fen Li, and Tan N. Nguyen. "Explainable artificial intelligence: a comprehensive review". In: *Artificial Intelligence Review* 55.5 (June 1, 2022), pp. 3503–3568. ISSN: 1573-7462. DOI: 10.1007/s10462-021-10088-y (cit. on p. 113).

[109] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Mar. 2, 1998. 226 pp. ISBN: 978-0-262-63185-3 (cit. on p. 25).

[110] Pradeep K. Murukannaiah and Catholijn M. Jonker. *MOPaC: The Multiple Offers Protocol for Multilateral Negotiations with Partial Consensus*. May 13, 2022. DOI: 10.48550/arXiv.2205.06678 (cit. on p. 10).

[111] John F. Nash. "The Bargaining Problem". In: *Econometrica* 18.2 (1950), pp. 155–162. ISSN: 0012-9682. DOI: 10.2307/1907266 (cit. on pp. 2, 13, 51, 77, 86).

[112] Martin A. Nowak. "Five Rules for the Evolution of Cooperation". In: *Science* 314.5805 (Dec. 8, 2006), pp. 1560–1563. DOI: 10.1126/science.1133755 (cit. on pp. 88, 112, 114).

[113] Eoin O'Mahony, Emmanuel Hebrard, Alan Holland, Conor Nugent, and Barry O'Sullivan. "Using Case-based Reasoning in an Algorithm Portfolio for Constraint Solving". In: *Proceedings of the 19th Irish Conference on Artificial Intelligence and Cognitive Science*. Lecture Notes in Computer Science. Springer, 2008 (cit. on p. 18).

[114] Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M. Czarnecki, Marc Lanctot, Julien Perolat, and Remi Munos. "$\apha$-Rank: Multi-Agent Evaluation by Evolution". In: *Scientific Reports* 9.1 (July 9, 2019), p. 9937. ISSN: 2045-2322. DOI: 10.1038/s41598-019-45619-9 (cit. on p. 89).

[115] Martin J. Osborne and Ariel Rubinstein. *Bargaining and markets*. Academic Press Limited, 1990 (cit. on p. 2).

[116] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. May 29, 2023. DOI: 10.48550/arXiv.2305.18290 (cit. on p. 100).

[117] Howard Raiffa. *Negotiation Analysis: The Science and Art of Collaborative Decision Making*. USA: Harvard University Press, Mar. 31, 2007. 567 pp. ISBN: 978-0-674-25569-2 (cit. on pp. 2, 98).

[118] Howard Raiffa. *The Art and Science of Negotiation*. USA: Harvard University Press, 1982. 390 pp. ISBN: 978-0-674-04813-3 (cit. on p. 2).

[119] Senthil Rajasekaran, Suguman Bansal, and Moshe Y. Vardi. *Multi-Agent Systems with Quantitative Satisficing Goals*. May 17, 2023. DOI: 10.48550/arXiv.2305.00953 (cit. on p. 100).

[120] John Rawls. *A Theory of Justice: Revised Edition*. USA: Harvard University Press, July 27, 2020. 495 pp. ISBN: 978-0-674-25767-2 (cit. on p. 98).

[121] Wei Ren, R.W. Beard, and E.M. Atkins. "A survey of consensus problems in multi-agent coordination". In: *Proceedings of the 2005, American Control Conference, 2005.* Proceedings of the 2005, American Control Conference, 2005. June 2005, 1859–1864 vol. 3. DOI: 10.1109/ACC.2005.1470239 (cit. on p. 2).

[122] Bram M. Renting, Holger H. Hoos, and Catholijn M. Jonker. "Automated Configuration and Usage of Strategy Portfolios for Bargaining". In: The Second Cooperative AI workshop, NeurIPS 2021. arXiv, Dec. 14, 2021. DOI: 10.48550/arXiv.2212.10228 (cit. on p. 39).

[123] Bram M. Renting, Holger H. Hoos, and Catholijn M. Jonker. "Automated Configuration and Usage of Strategy Portfolios for Mixed-Motive Bargaining". In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '22. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 9, 2022, pp. 1101–1109. ISBN: 978-1-4503-9213-6 (cit. on pp. 39, 61–62, 66).

[124] Bram M. Renting, Holger H. Hoos, and Catholijn M. Jonker. "Automated Configuration of Negotiation Strategies". In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '20. Auckland: International Foundation for Autonomous Agents and Multiagent Systems, May 5, 2020, pp. 1116–1124. ISBN: 978-1-4503-7518-4 (cit. on pp. 23, 46, 51–52, 54, 100).

[125] Bram M. Renting, Holger H. Hoos, and Catholijn M. Jonker. "Multi-Agent Meeting Scheduling: A Negotiation Perspective". In: The Sixteenth Workshop on Adaptive and Learning Agents. Mar. 13, 2024 (cit. on pp. 3, 93).

[126] Bram M. Renting, Thomas M. Moerland, Holger H. Hoos, and Catholijn M. Jonker. "Towards General Negotiation Strategies with End-to-End Reinforcement Learning". In: *Reinforcement Learning Journal* 5 (2024), pp. 2059–2070. ISSN: 2996-8577 (cit. on p. 53).

[127] John R. Rice. “The Algorithm Selection Problem”. In: *Advances in Computers*. Ed. by Morris Rubinoff and Marshall C. Yovits. Vol. 15. Elsevier, Jan. 1, 1976, pp. 65–118 (cit. on pp. 17, 40, 43–44).

[128] Avi Rosenfeld, Inon Zuckerman, Erel Segal-Halevi, Osnat Drein, and Sarit Kraus. “NegoChat: a chat-based negotiation agent”. In: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. AAMAS ’14. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 5, 2014, pp. 525–532. ISBN: 978-1-4503-2738-1 (cit. on p. 10).

[129] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. Cambridge, MA, USA: MIT Press, July 1994. 268 pp. ISBN: 978-0-262-18159-4 (cit. on p. 10).

[130] Jeffrey Solomon Rosenschein. “Rational interaction: cooperation among intelligent agents”. PhD thesis. Stanford, CA, USA: Stanford University, 1986 (cit. on p. 3).

[131] Jeffrey Z. Rubin and Bert R. Brown. *The Social Psychology of Bargaining and Negotiation*. Elsevier, Oct. 22, 2013. 372 pp. ISBN: 978-1-4832-8907-6 (cit. on p. 2).

[132] Ariel Rubinstein. “Perfect Equilibrium in a Bargaining Model”. In: *Econometrica* 50.1 (1982), pp. 97–109. ISSN: 0012-9682. DOI: 10.2307/1912531 (cit. on pp. 2, 10, 41, 70).

[133] Hajer Salem and Ahlem Ben Hassine. “Meeting Scheduling based on Swarm Intelligence”. In: *Procedia Computer Science*. Knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings 60 (Jan. 1, 2015), pp. 1081–1091. ISSN: 1877-0509. DOI: 10.1016/j.procs.2015.08.153 (cit. on p. 96).

[134] Rahul Savani and Theodore L. Turocy. *Gambit: The package for computation in game theory*. Version 16.2.0 (cit. on p. 83).

[135] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal Policy Optimization Algorithms*. Aug. 28, 2017. DOI: 10.48550/arXiv.1707.06347. arXiv: 1707.06347[cs] (cit. on pp. iii, v, 5, 56).

[136] Sandip Sen and Edmund H. Durfee. “A formal study of distributed meeting scheduling: preliminary results”. In: *Conference proceedings on Organizational computing systems - COCS ’91*. Conference proceedings. Atlanta, Georgia, United States: ACM Press, 1991, pp. 55–68. ISBN: 978-0-89791-456-7. DOI: 10.1145/122831.122837 (cit. on p. 94).

[137] Ayan Sengupta, Yasser Mohammad, and Shinji Nakadai. “An Autonomous Negotiating Agent Framework with Reinforcement Learning based Strategies and Adaptive Strategy Switching Mechanism”. In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS ’21. Richland, SC: International Foundation for Autonomous Agents

and Multiagent Systems, May 3, 2021, pp. 1163–1172. ISBN: 978-1-4503-8307-3 (cit. on pp. 40–41, 55, 61–62, 66, 100, 109, 114).

[138] Burr Settles. *Active Learning Literature Survey*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences, 2009 (cit. on p. 100).

[139] Lloyd S. Shapley. "Stochastic Games". In: *Proceedings of the National Academy of Sciences* 39.10 (Oct. 1953), pp. 1095–1100. DOI: 10.1073/pnas.39.10.1095 (cit. on p. 55).

[140] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. USA: Cambridge University Press, 2008. ISBN: 978-0-521-89943-7 (cit. on p. 2).

[141] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. ISSN: 1476-4687. DOI: 10.1038/nature16961 (cit. on p. 18).

[142] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (Dec. 7, 2018), pp. 1140–1144. DOI: 10.1126/science.aar6404 (cit. on p. 113).

[143] Reid G. Smith. "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver". In: *IEEE Transactions on Computers* C-29.12 (Dec. 1980), pp. 1104–1113. ISSN: 1557-9956. DOI: 10.1109/TC.1980.1675516 (cit. on pp. 3, 99).

[144] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. "Learning to summarize with human feedback". In: *Advances in Neural Information Processing Systems*. NeurIPS 2020. Vol. 33. Virtual: Curran Associates, Inc., 2020, pp. 3008–3021 (cit. on p. 95).

[145] Peter Suedfeld and Stanley Coren. "Cognitive correlates of conceptual complexity". In: *Personality and Individual Differences* 13.11 (Nov. 1, 1992), pp. 1193–1199. ISSN: 0191-8869. DOI: 10.1016/0191-8869(92)90255-N (cit. on p. 109).

[146] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, second edition: An Introduction*. USA: MIT Press, Nov. 13, 2018. 549 pp. ISBN: 978-0-262-35270-3 (cit. on pp. 18, 54, 56, 95).

[147] Katia Sycara. “Resolving goal conflicts via negotiation”. In: *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence*. AAAI’88. Saint Paul, Minnesota: AAAI Press, Aug. 21, 1988, pp. 245–250 (cit. on p. 3).

[148] Toki Takahashi, Ryota Higa, Katsuhide Fujita, and Shinji Nakadai. “VeNAS: Versatile Negotiating Agent Strategy via Deep Reinforcement Learning (Student Abstract)”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.11 (June 28, 2022), pp. 13065–13066. ISSN: 2374-3468. DOI: 10.1609/aaai.v36i11.21669 (cit. on p. 55).

[149] Leigh L. Thompson. *The mind and heart of the negotiator*. Pearson, 2015. ISBN: 978-1-292-07333-0 (cit. on p. 2).

[150] Edward Tsang. *Foundations of Constraint Satisfaction*. UK: Academic Press, 1993. 446 pp. ISBN: 978-0-12-701610-8 (cit. on pp. 94–95).

[151] Karl Tuyls and Gerhard Weiss. “Multiagent Learning: Basics, Challenges, and Prospects”. In: *AI Magazine* 33.3 (Sept. 20, 2012), pp. 41–41. ISSN: 2371-9621. DOI: 10.1609/aimag.v33i3.2426 (cit. on p. 2).

[152] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. “Graph Attention Networks”. In: International Conference on Learning Representations. Feb. 15, 2018 (cit. on p. 56).

[153] Ivan Vendrov, Tyler Lu, Qingqing Huang, and Craig Boutilier. “Gradient-Based Optimization for Bayesian Preference Elicitation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.6 (Apr. 3, 2020), pp. 10292–10301. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i06.6592 (cit. on p. 100).

[154] Toby Walsh. “Uncertainty in preference elicitation and aggregation”. In: *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*. AAAI’07. Vancouver, British Columbia, Canada: AAAI Press, July 22, 2007, pp. 3–8. ISBN: 978-1-57735-323-2 (cit. on p. 100).

[155] Dujuan Wang, Yugang Yu, Yunqiang Yin, and Tai Chiu Edwin Cheng. “Multi-agent scheduling problems under multitasking”. In: *International Journal of Production Research* 59.12 (June 18, 2021), pp. 3633–3663. ISSN: 0020-7543. DOI: 10.1080/00207543.2020.1748908 (cit. on p. 94).

[156] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. “NerveNet: Learning Structured Policy with Graph Neural Networks”. In: International Conference on Learning Representations. Feb. 15, 2018 (cit. on p. 55).

[157] Christopher J. C. H. Watkins and Peter Dayan. “Q-learning”. In: *Machine Learning* 8.3 (May 1, 1992), pp. 279–292. ISSN: 1573-0565. DOI: 10.1007/BF00992698 (cit. on p. 18).

[158] Michael P. Wellman. “Methods for empirical game-theoretic analysis”. In: *proceedings of the 21st national conference on Artificial intelligence - Volume 2*. AAAI’06. Boston, Massachusetts: AAAI Press, July 16, 2006, pp. 1552–1555. ISBN: 978-1-57735-281-5 (cit. on p. 81).

[159] Colin R. Williams, Valentin Robu, Enrico H. Gerding, and Nicholas R. Jennings. "IAMhaggler: A Negotiation Agent for Complex Environments". In: *New Trends in Agent-Based Complex Automated Negotiations*. Ed. by Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo. Berlin, Heidelberg: Springer, 2012, pp. 151–158. ISBN: 978-3-642-24696-8. DOI: 10.1007/978-3-642-24696-8_10 (cit. on p. 68).

[160] Colin R. Williams, Valentin Robu, Enrico H. Gerding, and Nicholas R. Jennings. "Using Gaussian processes to optimise concession in complex negotiations against unknown opponents". In: *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume One*. IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press, July 16, 2011, pp. 432–438. ISBN: 978-1-57735-513-7 (cit. on p. 81).

[161] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. "SATzilla: Portfolio-based Algorithm Selection for SAT". In: *Journal of Artificial Intelligence Research* 32 (July 1, 2008), pp. 565–606. ISSN: 1076-9757. DOI: 10.1613/jair.2490 (cit. on p. 18).

[162] Lin Xu, Holger Hoos, and Kevin Leyton-Brown. "Hydra: Automatically Configuring Algorithms for Portfolio-Based Selection". In: *Twenty-Fourth AAAI Conference on Artificial Intelligence*. Twenty-Fourth AAAI Conference on Artificial Intelligence. July 3, 2010 (cit. on pp. iii, v, 5, 40, 42–43, 107).

[163] Tianpei Yang, Heng You, Jianye Hao, Yan Zheng, and Matthew E. Taylor. "A Transfer Approach Using Graph Neural Networks in Deep Reinforcement Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.15 (Mar. 24, 2024), pp. 16352–16360. ISSN: 2374-3468. DOI: 10.1609/aaai.v38i15.29571 (cit. on p. 55).

[164] M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara. "The distributed constraint satisfaction problem: formalization and algorithms". In: *IEEE Transactions on Knowledge and Data Engineering* 10.5 (Sept. 1998), pp. 673–685. ISSN: 1558-2191. DOI: 10.1109/69.729707 (cit. on pp. 2, 96).

[165] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. "Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018 (cit. on p. 55).

[166] Mark Zlochin, Mauro Birattari, Nicolas Meuleau, and Marco Dorigo. "Model-Based Search for Combinatorial Optimization: A Critical Survey". In: *Annals of Operations Research* 131.1 (Oct. 1, 2004), pp. 373–395. ISSN: 1572-9338. DOI: 10.1023/B:ANOR.0000039526.52305.af (cit. on p. 17).

[167] Alejandro Zunino and Marcelo Campo. "Chronos: A multi-agent system for distributed automatic meeting scheduling". In: *Expert Systems with Applications* 36.3 (Apr. 1, 2009), pp. 7011–7018. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2008.08.024 (cit. on p. 94).

Curriculum Vitæ

Bram Matthias Renting was born on September 12, 1991, in Zwolle, the Netherlands. After a pre-university education, he pursued a Bachelor in Marine Technology at the Delft University of Technology which he completed in 2016 followed by a Master of Science in Embedded Systems with specialisation in Control Systems, also at the Delft Unversity of Technology and partially at Politecnico di Milano, which he completed in 2019. During his academic career, Bram was actively involved in extracurricular activities. He was a member of the DUT Racing team, was a competitive rower, had various roles in the organisation of the rowing club, including coaching, and was a teaching assistant for various courses at the university. During his studies, he also spend 2 years working in the marine industry as a (project) engineer. After finishing his Master, Bram worked as a data scientist at DNV in Schiedam, the Netherlands, where he build machine learning models for various applications. In 2020 Bram started his PhD studies at Leiden University. During his PhD studies, he took courses in communication and scientific conduct, among others. Starting from June 2025, Bram is a senior research engineer at Det Norske Veritas in Oslo, Norway, working on the intersection of AI research and marine technology.

List of Publications

Submitted

1. **Bram M. Renting**, Dave de Jonge, Holger H. Hoos, and Catholijn M. Jonker. Analysis of Learning Agents in Automated Negotiation. (Chapter 6)

2024

3. Thijs Snelleman, **Bram M. Renting**, Holger H. Hoos, and Jan N. van Rijn. 2024. Edge-Based Graph Component Pooling. In *Proceedings of the 21st International Workshop on Mining and Learning with Graphs*, ECMLPKDD '24.

2. **Bram M. Renting**, Thomas M. Moerland, Holger H. Hoos, and Catholijn M. Jonker. 2024. Towards General Negotiation Strategies with End-to-End Reinforcement Learning. In *Reinforcement Learning Journal* volume 5, pages 2059–2070. (Chapter 5)

1. **Bram M. Renting**, Holger H. Hoos, and Catholijn M. Jonker. 2024. Multi-Agent Meeting Scheduling: A Negotiation Perspective. In *Proceedings of the 16th Workshop on Adaptive and Learning Agents*. AAMAS '24. (Chapter 7)

2023

1. Reyhan Aydoğan, Tim Baarslag, Katsuhide Fujita, Holger H. Hoos, Catholijn M. Jonker, Yasser Mohammad, and **Bram M. Renting**. 2023. The 13th International Automated Negotiating Agent Competition Challenges and Results. In *Recent Advances in Agent-Based Negotiation: Applications and Competition Challenges*. Studies in Computational Intelligence.

2022

1. **Bram M. Renting**, Holger H. Hoos, and Catholijn M. Jonker. 2022. Automated Configuration and Usage of Strategy Portfolios for Mixed-Motive Bargaining. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '22. (Chapter 4)

2021

1. **Bram M. Renting**, Holger H. Hoos, and Catholijn M. Jonker. 2021. Automated Configuration and Usage of Strategy Portfolios for Bargaining. In *Proceedings of the 2nd Cooperative AI Workshop*. NeurIPS '21. (Chapter 4)

2020

1. **Bram M. Renting**, Holger H. Hoos, and Catholijn M. Jonker. 2020. Automated Configuration of Negotiation Strategies. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '20. (Chapter 3)

PREPRINT

1. Thomas M. Moerland, Matthias Müller-Brockhausen, Zhao Yang, Andrius Bernatavicius, Koen Ponse, Tom Kouwenhoven, Andreas Sauter, Michiel van der Meer, **Bram M. Renting**, and Aske Plaat. 2023. EduGym: An Environment Suite for Reinforcement Learning Education. Preprint.

Included in this Dissertation.
Won a best paper.