NegMAS: A platform for automated negotiations

Yasser Mohammad[1,2,3], Shinji Nakadai[1,2], and Amy Greenwald[4]

[1] NEC Corporation, Japan
{nakadai,y.mohammad}@nec.com
[2] National Institute of Advanced Industrial Science and Technology (AIST), Japan
[3] Assiut University, Egypt
[4] Brown University, USA
amy_greenwald@brown.edu

Abstract. Alongside the widespread adoption of AI technology thoughout the business world, automated negotiation is similarly gaining more interest within the multiagent system (MAS) research community. This interest has prompted the development of research-oriented automated negotiation platforms like GENIUS. This paper introduces NegMAS, Negotiations Managed by Agent Simulations / Negotiation MultiAgent System, which was developed to facilitate research and development of agents that negotiate in dynamic situations characterized by inter-related utility functions with all negotiation related decisions managed by agents.

1 Introduction

Negotiation is one of the most prevalent methods for reaching agreements between self-interested parties. In automated negotiation, autonomous (software) agents negotiate among themselves, or with human negotiators, on behalf of their users. Negotiation research can be traced back to the seminal work of Nash on bargaining theory [17], and Rubinstein's analysis of the alternating offers protocol in the perfect-information case [18], both major game-theoretic advances. More recently, research in automated negotiations has attracted researchers in multi-agent systems (e.g., [6]) and machine learning (e.g., [20]).

Few platforms have been designed to support research in automated negotiations. The de-facto standard platform is the General Environment for Negotiation with Intelligent multi-purpose Usage Simulation (GENIUS) [14]. GENIUS was designed to facilitate research in automated negotiation by providing an extensive analytic toolkit for developers. Since 2010, it has since been the official platform for the Automated Negotiating Agents Competition (ANAC). By serving in this capacity, it has accrued a large number of negotiation strategies and domains, making it an indispensable tool for researchers in automated negotiation. The platform has been available as an open-source project since 2018[5].

A related open-source project that was released in 2019 is the GeniusWeb [2] platform, which provides an open architecture for negotiation over the internet. Based on GENIUS, it shares most of its core strengths (i.e., availability of an extensive analytic

[5] NegMAS is available at https://www.github.com/yasserfarouk/negmas

toolbox, multiple built-in negotiation strategies and domains, etc.), but it further provides flexibility in the way agents can be implemented and deployed.

Yet another negotiation related platform is the *Invite* platform, developed for research and training purposes by Concordia University [12]. Like the pocket negotiator project [11], the main focus of this platform is supporting human-human negotiations.

Commercial platforms in the form of negotiation support systems are also being developed. One such example is ContractRoom [1]. This platform provides easy-to-use tools that enable human negotiators to reach agreements faster, such as a mechanism that facilitates online collaboration. This mechanism is augmented with artificial intelligence, but is still mostly a human-human negotiation support system; it does not venture into the realm of automated negotiations.

All of these platforms assume a static negotiation situation in that the set of issues and the utility functions are fixed throughout the negotiation session. Moreover, they provide little support for interdependent negotiation sessions, which are required to model situations that involve concurrent negotiations and dynamic utility functions. We believe that the primary missing feature that can help overcome most, if not all, of the limitations of existing platforms is the ability to embody negotiations within a rich simulation environment, where utility functions arise endogenously and interdepend across concurrent negotiations. Such *situated negotiations* are closer to reality, so a platform that supports them can provide a bridge between state-of-the-art automated negotiation research and real-world applications.

The main contribution of this paper is the introduction of the Negotiations Managed by Agent Simulations / Negotiation MultiAgent System (NegMAS) platform, which was designed to model situated negotiations, thereby handling most of the aforementioned shortcomings with existing automated negotiation platforms.

NegMAS is intended to complement existing automated negotiation platforms (e.g., GENIUS) by addressing the structural issues stemming from the specific nature of situated negotiations. Moreover, NegMAS is developed as an open-source public project; as such, it is open to contributions from the whole research community. It provides a common API that supports multiple programming language (currently python and Java). Finally, it is designed to work either as a stand-alone system or as a client to a distributed system, implementing the same API, thus providing a scalable solution.

The rest of this paper is organized as follows: Section 2 presents details of the design philosophy and design decisions made in NegMAS to support situated negotiations. Section 3 summarizes the analytic tools available in NegMAS. Section 4 describes an example application developed using NegMAS.

2 System Design

This section presents an overview of NegMAS' key components and their interactions. As a general design principle, NegMAS is intended to make common cases easy to implement, with less common cases still possible, but assuming common default settings for most parameters of its components.

Rational and Runnable The two main entities in NegMAS are *Rational* and *Runnable*. Rational entities have a form of self interest, represented by a utility function, which

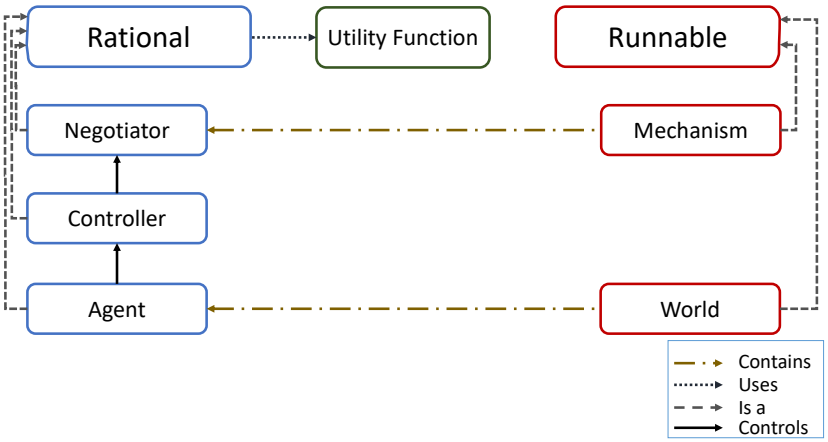

Fig. 1. Main entities in NegMAS and their relationships.

in turns, represents preferences. These include *Negotiators* for carrying out automated negotiations; *Controllers* for orchestrating multiple negotiations; and *Agents* for representing simulated entities (e.g., companies, individuals, etc).

Runnable entities represent processes in the environment. They control the flow of control. Currently NegMAS has two kinds of Runnable entities: *Mechanisms* to represent negotiations (or other agreement-seeking mechanisms like auctions; and *Worlds* to represent simulated worlds within which negotiations take place.

Worlds The simulation environments within which agents operate in NegMAS are called *worlds*. As all worlds use the same interface, some common functionality is provided. These include a public bulletin board on which common information available to all agents in the world is posted, and summary statistics calculations. The world also provides contract persistence (i.e., saving contracts even after a simulation ends), name resolution services, logging, and statistics calculation.

Each world contains a simulator that is responsible for running the environment. Moreover, the world simulator executes all mechanisms within it. Agents can affect the simulation through actions that the world defines.

New world types can be created by inheriting the abstract *World* class and implementing its abstract methods. At a minimum, a single simulation step method needs to be implemented. The designer can also customize the speed of negotiations relative to the simulation speed, and the order of simulation operations (e.g., do negotiations happen throughout a step, only at its start, or only at its end?), among other things.

Mechanisms Negotiations are conducted based on a negotiation *protocol*, which encodes the *rules of engagement* for negotiators. Negotiation protocols are the primary *mechanisms* in NegMAS. A mechanism is an entity that controls the interactions among negotiators. Beyond negotiation protocols, mechanisms can also represent auctions.

Mechanisms define a set of *requirements* that must be satisfied by any negotiator that joins them. In addition to defining a set of requirements, mechanisms also have to define two operations: initialization and a round operation. Mechanisms are run by executing the round operation until it returns a special stop symbol or until a time limit is reached. Time limits can be defined for the complete mechanism session or for each round and each negotiator's action. This feature simplifies implementation of *bounded rationality* negotiators, where the bound is imposed by computational considerations.

Currently, NegMAS includes implementations of the Stacked Alternating Offers Protocol (SAOP), as an example of a non-mediated negotiation protocol [3], the Single Text Negotiation Protocol as an example of a mediated protocol [10], the first-price and second-price auctions, as examples of one-shot mechanisms, and an English auction, as an example of a dynamic auction [19]. Adding new mechanisms to NegMAS involves implementing only a single method.

Agents The main actor in NegMAS is the agent. An agent represents an autonomous entity that has well defined objectives (which can, but need not, be explicitly encoded in a utility function). Fig. 1 shows an example of an agent, which, using a controller and two independent negotiators, is engaged in four simultaneous negotiations.

Agents in NegMAS interact within a simulation that is part of a *world*. Within a world, agents can access public information as well as their own private state, and can execute actions as well as engage in negotiations. Agents are responsible for deciding what negotiations to engage in, which utility functions to use, and how to change their utility functions based on changes in the world simulation, their internal state, or outcomes of other negotiations. Moreover, agents may have other activities not directly related to negotiation that are crucial to achieving their objectives. For example, an agent representing a factory manager needs to control the production lines in that factory based on results of its negotiations.

Negotiators Negotiations occur between *negotiators*. All negotiator types define capabilities that are matched with the requirements of the negotiation protocol before agents are allowed to join negotiation mechanisms. This makes it possible to define negotiation strategies that are applicable across multiple negotiation protocols.

All negotiators define a set of callbacks that can be used to update the negotiator's internal state or behavior based on salient events during the negotiation, including the negotiation's start and end, a round's start and end, errors, and utility function updates.

It is not possible to define general purpose negotiators in NegMAS independent of a negotiation protocol. NegMAS provides implementations of simple negotiation strategies for the SAOP in the bilateral [5] and multilateral cases [3], including the time-based aspiration level strategy with exponential and polynomial aspiration functions [6], and the Naive version of the tit-for-tat strategy described in Baarslag, *et al.* [4].

Beyond these built-in negotiators, NegMAS can also access most negotiation agents defined in the GENIUS platform [14] through a GeniusNegotiator class that allows these agents to participate in negotiation sessions running on NegMAS. Note, however, that since NegMAS supports richer simulation environments than GENIUS, GENIUS negotiators are not always applicable: e.g., they assume static utility functions.

Controllers Negotiators can participate in but one negotiation at a time. This means that they cannot support concurrent negotiations, which are characteristic of real-world negotiations. NegMAS thus provides a *controller* entity capable of orchestrating the behavior of multiple negotiators (its children). Any method that is implemented by the controller takes precedence over the same method implemented by any of its negotiators. This way, controllers can decide to delegate some of their activities to negotiators, while still maintaining centralized control.

Utility Functions NegMAS provides the basic components necessary to model a wide swath of negotiation scenarios, including *Issues* and *Outcomes*, as well as a variety of utility functions. In contrast to existing negotiation environments, utility functions in NegMAS are active entities that evolve over time. They are implemented as objects in the standalone version, and as processes in the distributed version.

NegMAS supports three kinds of utility function interfaces: cardinal, comparative, and ranking. *Cardinal* utility functions need to implement a mapping from any possible outcome (or a partial outcome) to a utility value. Utility values can be real numbers or a probability distribution over real numbers (e.g. uniform, Gaussian, etc.). *Comparative* utility functions need to implement only a comparison operator between any two outcomes, allowing for indifference. *Ranking* utility functions need to implement a ranking function that returns, for any list of outcomes, a partial ordering over them.

Currently NegMAS supports the following types of cardinal utility functions (among others): linear utility functions, generalized additive independence models [7], hyperrectangle utility functions [9] and non-linear combinations thereof, and, more generally, any nonlinear mapping from the outcome space to utilities, implemented, for example, as an arbitrary neural network.

Each of these special cases of cardinal utility functions is represented in NegMAS as its own type, making it possible to implement case-specific algorithms. For example, finding the range of a linear utility function can be done exactly and more efficiently than finding the range of a non-linear utility function. Defining new types of cardinal utility functions involves overriding a single method.

General support for time-discounted cardinal utility functions, with both linear and exponential discounting, is also available. Moreover, NegMAS supports partial outcome specification for multi-dimensional outcome spaces, meaning specification of values for only a partial set of the outcomes, which is used in some mediated protocols [13].

3 Tools and Common Components

To develop effective negotiation algorithms, it is beneficial to designers to have at their disposal a variety of analytic tools for modeling the negotiation scenario and understanding the results of negotiations. NegMAS comes with a growing set of analytic tools that supports developers in this quest. Some of the most important tools available in NegMAS include: Pareto-front evaluation; methods for evaluating the Nash-bargaining, maximum-welfare, and other salient points in the outcome space. Moreover, NegMAS provides tools for outcome space modeling and parameterized generation of utility functions and simulation conditions.

Several visualization primitives are provided in NegMAS for visualizing world simulations, including negotiation requests, negotiation results, and contract signing and execution. Fig. 2 shows the default visualization of a sample negotiation conducted between a seller and a buyer.

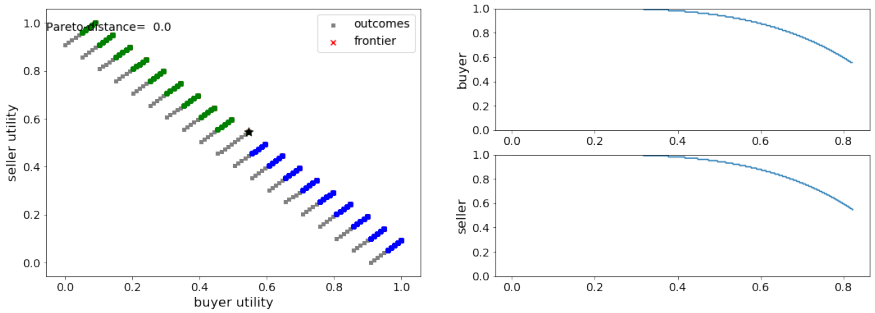

Fig. 2. An example of a negotiation showing the offers exchanged and the final agreement.

Additionally, several visualization options pertaining to the negotiation context are available to the developer. Fig. 3 depicts one such example, where all negotiation-related events between different agents are shown as edges between vertices, the latter of which represent agents within a simulated world.

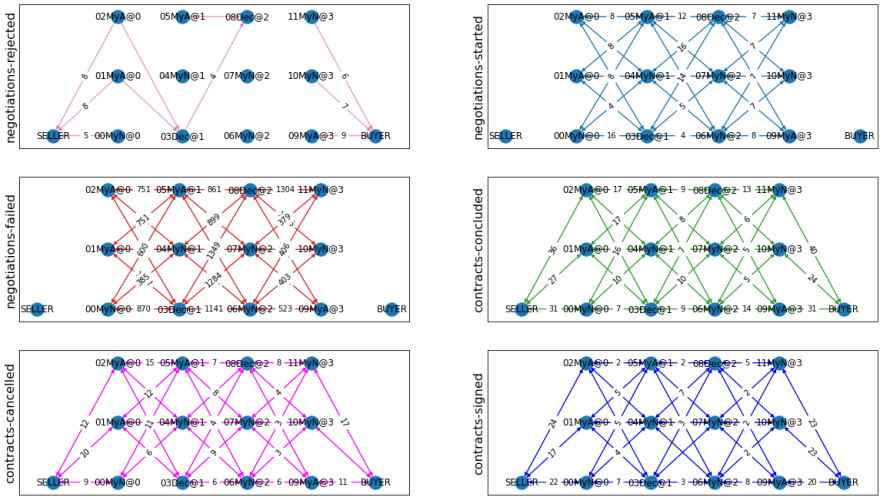

Fig. 3. A visualization of a simulated world in NegMAS. Depicted here are some negotiation and contract-related events. The developer has full control over what events to visualize.

NegMAS provides a common tournament management interface which can be used to run tournaments among agents in any world by implementing just four components: a configuration generator to generate different world configurations, an assigner that assigns competitors to these worlds, a world generator that builds world simulations given configurations together with complete assignments, and a score calculator that calculates the scores of agents based on related world simulations.

4 Applications

NegMAS is still young, yet it is already being used actively for research and development of negotiation agents. It was used in both 2019 [16] and 2020 as the platform for the Supply Chain Management League conducted as part of the Automated Negotiation Agents Competition held in conjunction with IJCAI. In this application, NegMAS was used to implement a genuine situated negotiations situation in which autonomous agents decide when to negotiate, about what, and with whom, using dynamic utility functions that emerge endogenously from the supply chain simulation dynamics rather than being dictated from outside the system.

NegMAS has also been used as a platform for preference elicitation research [15], where the ability to model uncertainties in utility functions is especially important, and in path planning for self-interested robots [8], where it is useful to be able to handle a large number of concurrent negotiations efficiently.

5 Conclusions

This paper presents NegMAS, a new platform for automated negotiations that provides tools for developing negotiation protocols and automated negotiation agents. The proposed system is designed to model realistic negotiation scenarios with dynamic utility functions that are endogenous to the environment within which agents are embedded. Representing dynamic utility functions is essential in real-world applications of automated negotiation. The new platform is being developed as an open-source project with the goal of engaging the automated negotiation research community.

6 Acknowledgments

Amy Greenwald is supported in part by NSF Award CMMI-1761546.

References

1. Contract room platform (2019), https://www.contractroom.com/
2. Geniusweb platform (2019), https://ii.tudelft.nl/GeniusWeb/technicians.html
3. Aydoğan, R., Festen, D., Hindriks, K.V., Jonker, C.M.: Alternating offers protocols for multilateral negotiation. In: Modern Approaches to Agent-based Complex Automated Negotiation, pp. 153–167. Springer (2017)

4. Baarslag, T., Hindriks, K., Jonker, C.: A tit for tat negotiation strategy for real-time bilateral negotiations. In: Complex Automated Negotiations: Theories, Models, and Software Competitions, pp. 229–233. Springer (2013)
5. Chatterjee, K., Samuelson, W.: Bargaining under incomplete information. Operations research **31**(5), 835–851 (1983)
6. Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. Robotics and Autonomous Systems **24**(3-4), 159–182 (1998)
7. Fishburn, P.C.: Interdependence and additivity in multivariate, unidimensional expected utility theory. International Economic Review **8**(3), 335–342 (1967)
8. Inotsume, H., Aggarewal, A., Higa, R., Nakadai, S.: Path negotiation for self-interested multirobot vehicles in shared space. In: Proc. of the 2020 IEEE/RSJ International Converent on Intelligent Robots and Systems (IROS) (2020)
9. Ito, T., Hattori, H., Klein, M.: Multi-issue negotiation protocol for agents: Exploring nonlinear utility spaces. In: IJCAI. vol. 7, pp. 1347–1352 (2007)
10. Ito, T., Klein, M., Hattori, H.: A multi-issue negotiation protocol among agents with nonlinear utility functions. Multiagent and Grid Systems **4**(1), 67–83 (2008)
11. Jonker, C., Aydogan, R., Baarslag, T., Broekens, J., Detweiler, C., Hindriks, K., Huldtgren, A., Pasman, W.: An introduction to the pocket negotiator: A general purpose negotiation support system. pp. 13–27 (06 2017)
12. Kersten, G.E.: Are procurement auctions good for society and for buyers? In: Joint International Conference on Group Decision and Negotiation. pp. 30–40. Springer (2014)
13. Klein, M., Faratin, P., Sayama, H., Bar-Yam, Y.: Protocols for negotiating complex contracts. IEEE Intelligent Systems **18**(6), 32–38 (2003)
14. Lin, R., Kraus, S., Baarslag, T., Tykhonov, D., Hindriks, K., Jonker, C.M.: Genius: An integrated environment for supporting the design of generic automated negotiators. Computational Intelligence **30**(1), 48–70 (2014). https://doi.org/10.1111/j.1467-8640.2012.00463.x, http://dx.doi.org/10.1111/j.1467-8640.2012.00463.x
15. Mohammad, Y., Nakadai, S.: Optimal value of information based elicitation during negotiation. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 242–250. AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems (2019)
16. Mohammad, Y., Viqueira, E.A., Ayerza, N.A., Greenwald, A., Nakadai, S., Morinaga, S.: Supply chain management world. In: Baldoni, M., Dastani, M., Liao, B., Sakurai, Y., Zalila Wenkstern, R. (eds.) PRIMA 2019: Principles and Practice of Multi-Agent Systems. pp. 153–169. Springer International Publishing, Cham (2019)
17. Nash Jr, J.F.: The bargaining problem. Econometrica: Journal of the Econometric Society pp. 155–162 (1950)
18. Rubinstein, A.: Perfect equilibrium in a bargaining model. Econometrica: Journal of the Econometric Society pp. 97–109 (1982)
19. Wurman, P.R., Wellman, M.P., Walsh, W.E.: A parametrization of the auction design space. Games and economic behavior **35**(1-2), 304–338 (2001)
20. Zeng, D., Sycara, K.: Bayesian learning in negotiation. International Journal of Human-Computer Studies **48**(1), 125–141 (1998)