win.win | a negotiating agent for mutual benefit

JOCHEM BROUWER, SAUL GEBHARDT, WOUTER VAN HARTSKAMP, and OTTO MÄTTAS, Utrecht University, The Netherlands

ACM Reference Format:
Jochem Brouwer, Saul Gebhardt, Wouter van Hartskamp, and Otto Mättas. 2021. win.win | a negotiating agent for mutual benefit. 1, 1 (March 2021), 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ASSIGNMENT 3.2.1

1.1 Evaluating Fairness

There exist many systems where multiple independent agents, or players, may strive to optimize their own utility unilaterally, which can be modeled as non-cooperative games. Given agents' decisions, the utilities of all agents are determined. By an allocation we mean a situation where the decisions of all agents are determined. The allocation where each agent attains its own optimum coincidentally is a Nash equilibrium as shown by [Kameda et al. 2009].

Nash equilibria may, however, be Pareto inefficient (or, simply, inefficient), that is, there may exist another allocation of a system where no agents have less benefit and at least one has more benefit than in the Nash equilibrium of the system. On the other hand, we consider a Nash equilibrium to be fair as it is defined in a context of fair competition.

This also applies to the current assignment, where two agents are negotiating against each other for asserting preferences (Figure 1) over the outcome of a mutually planned holiday. We are showing the most fair arrangement for both agents through Nash equilibrium.

By running a negotiation of two Boulware agents with the round limit of 60, (Figure 2), we can look for the deal which falls on the Nash point and suggests Nash equilibrium. Thus, we have found the most fair holiday arrangement for Agent A and Agent B is as listed below.

- Location: Milan
- Duration: 2 weeks
- Hotel Quality: Hostel

Issue:	Possible Values:
Location	Antalya, Barcelona, Milan
Duration	1 week, 2 weeks
Hotel Quality	Hostel, 3 star hotel, 5 star hotel

Agent A's preferences:

Weights: $W_{location}$= 0.5 $W_{duration}$= 0.2 $W_{hotel\text{-}quality}$=0.3
Evaluation values: 4, 10, 2 (for Antalya, Barcelona and Milan respectively)
Evaluation values: 3, 10 (for 1 week and 2 weeks respectively)
Evaluation values: 10, 2, 3 (for hostel, 3 star hotel and 5 star hotel respectively)

Agent B's preferences:

Weights: $W_{location}$= 0.5 $W_{duration}$= 0.4 $W_{hotel\text{-}quality}$=0.1
Evaluation values: 3, 2, 10 (for Antalya, Barcelona and Milan respectively)
Evaluation values: 4, 10 (for 1 week and 2 weeks respectively)
Evaluation values: 3, 3, 10 (for hostel, 3 star hotel and 5 star hotel respectively)

Fig. 1. Holiday preferences for Agent A & Agent B

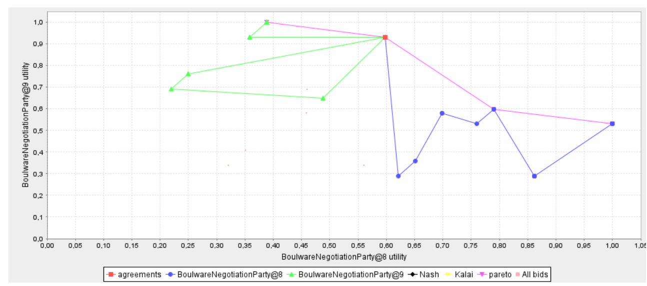

Fig. 2. Holiday negotiation for Agent A & Agent B

1.1.1 *Beyond Nash Equilibrium.*

There may exist innumerably many Pareto-optimal allocations. Choosing which of them to achieve can be controversial among agents. In contrast, each Nash equilibrium is fair among all agents in the sense that it is achieved by the fair competition (with no coalition) among agents. Our purpose in this report is not to consider different paradigms for resource sharing but simply state what is most fair for all agents.

As shown by [Kameda et al. 2009], there are outcomes which are on one hand Pareto efficient, and on the other hand have fairness properties that are related to the Nash equilibrium. We could continue and consider a resource allocation problem in which each of several players has some utility related to the amount of resources it gets. The aim then would be to propose a centralized allocation that is efficient (in the Pareto sense) and fair at the same time. This would arguably bring allocation improvement for the whole system in the form of higher overall benefit or social welfare, without losing the fairness.

1.2 Evaluating different negotiation parties

1.2.1 *RandomBidderExample.*

We now start a negotiation within the party domain, with two agents. Both agents employ the same strategy, `RandomBidderExample`. One agent has preference profile `party1_utility.xml` and the other agent has preference profile `party2_utility.xml`. The strategies and preference profiles are set to the default settings. We run a single negotiation which lasts

Authors' address: Jochem Brouwer, j.brouwer8@students.uu.nl; Saul Gebhardt, s.a.gebhardt@students.uu.nl; Wouter van Hartskamp, w.vanhartskamp@students.uu.nl; Otto Mättas, o.mattas@students.uu.nl, Utrecht University, P.O. Box 80125, Utrecht, Utrecht, The Netherlands, 3508 TC.

XXXX-XXXX/2021/3-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

60 rounds. In this case, Pareto optimal outcome is not reached. This is clear since the agreement is not part of the Pareto Frontier on the graph, and because at the end of the negotiation log, the distance to Pareto is 0.07287, where a distance to Pareto of 0 denotes a Pareto optimal outcome.

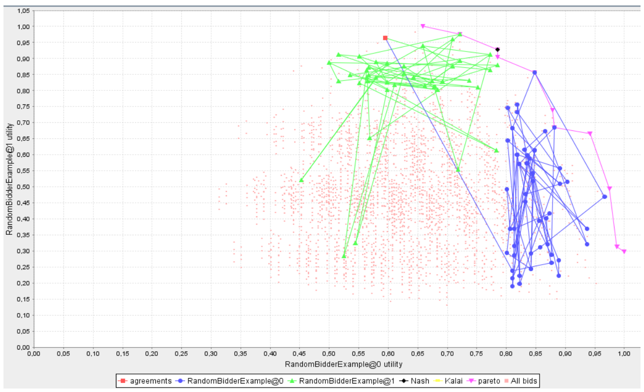

Fig. 3. RandomBidderExample against itself in party_domain

`RandomBidderExample` works as follows: at the start of the negotiation, it sends random offers. The random offers are being generated as follows: the class `RandomBidderExample` contains a value called `MINIMUM_TARGET`. The agent attempts to generate a bid which has a utility higher than 0.8. Once a bid has been found with a utility higher than 0.8, this bid is returned to the other agent. If the agent fails to find such a bid quickly (within 100 tries), the agent just sends the latest random bid to the other agent. The utility of this bid can be lower than 0.8.
Only once 90% has passed, the agent will consider the latest offer made by the opponent. At this point, `RandomBidderExample` evaluates whether the latest offer made by the opponent has a higher utility than its own `ReservationValue` (the 'walk-away point'). This value can be found and set in each preference profile within Genius, and is set to 0.0 by default for both of the preference profiles. This means that both preference profiles would prefer any agreement to no agreement at all.
If the utility is higher than the `ReservationValue`, the agent accepts the bid and the negotiation finishes. If the utility of the latest offer is lower than the `ReservationValue`, the negotiation fails.

In the case of the negotiation of a `RandomBidderExample` agent with another `RandomBidderExample` agent in the party domain, the negotiation goes as follows: for 90% of the negotiation, both agents try to create bids with a utility for them larger than 0.8, without ever considering the bids that the other agent has made. Once 90% of the time has passed, instead of making a new offer, the first agent will look at the bid made by the other agent and will check if the utility for this outcome is higher than its own `ReservationValue`. If this is the case, agent 1 will accept. If the utility of this bid is lower than its own `ReservationValue`, the negotiation will fail.
Since the `ReservationValue` in both preference profiles is 0.0, the agent will always accept this bid, and the negotiation will therefore never fail.

While it is possible due to chance to reach a Pareto optimal outcome, the value of `MINIMUM_TARGET` does not influence this chance. If the value of `MINIMUM_TARGET` is set too high, it becomes too difficult for the agent to generate a bid which has a utility above its `MINIMUM_TARGET` and instead just sends out any bid with a random utility. If the value of `MINIMUM_TARGET` is lowered, both agents will send worse bids, which also lowers the chance of sending out a Pareto optimal bid once 90% of the time has passed.

1.2.2 *ConcederNegotiationParty and BoulwareNegotiationParty.*

We now start a negotiation on the same domain with the same preference profiles. We give one agent the strategy `BoulwareNegotiationParty` and the other agent the strategy `ConcederNegotiationParty`. A Pareto optimal outcome is reached with these two strategies. This is clear, because we can see in the figure below that the agreement is on the Pareto frontier and the distance to Pareto is 0.0 at the end of the negotiation log.

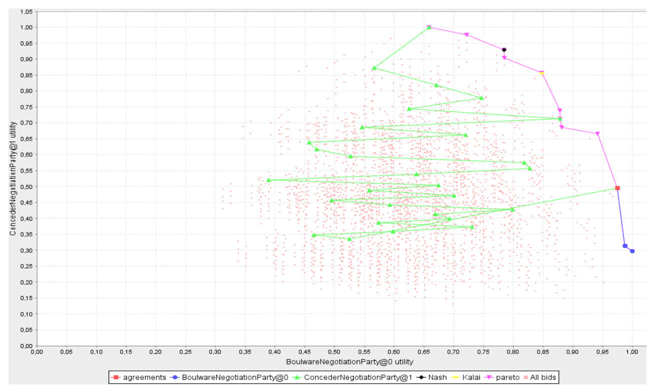

Fig. 4. Boulware versus Conceder in party_domain

`ConcederNegotiationParty` and `BoulwareNegotiationParty` are both strategies that are based on the same java file, `AbstractTimeDependentNegotiationParty.java`. The agents behave in a similar way. Both agents utilize a similar exponential function to calculate their target utility based on the current time instance. For the `ConcederNegotiationParty` this function is $1 - t^{\frac{1}{2}}$, for the `BoulwareNegotiationParty` this function is $1 - t^{\frac{1}{0.2}}$, where t is the current position on the timeline, which scales from 0.0 to 1.0.
When the value of t increases, we can see that the value of the target function of `ConcederNegotiationParty` rapidly decreases, while the target function of the `BoulwareNegotiationParty` stays at 1.0 for quite some time.

REFERENCES

Hisao Kameda, Eitan Altman, Corinne Touati, and Arnaud Legrand. 2009. Nash Equilibrium Based Fairness. *Mathematical Methods of Operations Research* 76, 533 – 539. https://doi.org/10.1109/GAMENETS.2009.5137442