

Phantom8: A phantom decoy agent submitted to the ANAC 2026 ANL league

Ming-lang Qin, Jeremy Hui, Xintong Wang

Rutgers University

`mq158@scarletmail.rutgers.edu`, `jwh139@scarletmail.rutgers.edu`, `xintong.wang@rutgers.edu`

June 24, 2026

Abstract

This agent implements deception by using a phantom phase, where it selects least preferred outcomes to be offered which are also estimated to be poor for the opponent. This ensures that the decoy offers are typically rejected due to being low utility for both parties, but results in the opponent being unable to estimate our utilities. The agent also uses an acceptance strategy that utilizes a dynamic estimated target, and learns opponent preferences from both frequency of observed offers and RankFit to predict opponent utilities.

1 Introduction

In ANL 2026, the goal is to optimize both the advantage scored from the outcome, and the deception they are able to achieve by keeping their own agent's preferences private. Both are equally important, as they contribute an equal weight toward the final score. To this end, we propose a phantom decoy phase, where while the opponent has not offered any high utility outcomes, we repeatedly offer one fixed outcome. This utility will have low utility for our agent, but will be worse for our opponent. This ensures that agreement will never be reached on this outcome, but more importantly will cause any frequency-based model of the opponent to converge to this outcome, and thus deceive the opponent from learning our true utility function.

Additionally, once the phantom deception period has elapsed and the relative time approaches the negotiation deadline, we employ a boulware negotiation strategy and only concede aggressively when we are very late into the negotiation period. The goal in this design is that after the relatively longer phantom decoy phase, our agent will be able to effectively learn the opponent's preferences, and thus settle on an outcome that is best for us but will still clear the opponent's minimum utility for acceptance.

2 The Design of Phantom8

The Phantom8 agent is divided into three main parts: the phantom decoy phase which is utilized to conceal the bidding strategy, an acceptance strategy which generally attempts to accept an offer using a dynamic estimated target based on the current relative time, and an opponent model that uses both a frequency model and a RankFit model that is used for the opponent utility function. These components work together to ensure that the opponent is unable to learn our utility function simply through using a frequency-based model, as well as have our opponent model be resistant to the similar decoy tactics we use and be aware of the based on the current relative time.

3 Concealing Bidding Strategy

As described earlier, the main strength of this agent is in the phantom decoy phase. More specifically, we define a decoy to be safe only when the offer has a utility that is lower than the current predicted acceptance bar (which we set as the value of the opponent’s latest offer) by some margin, as this will make it unlikely for the opponent to settle on an offer that is unfavorable for both us and our opponent.

Our agent also employs a different strategy for the decoy depending on the perceived “toughness”, and how likely the opponent will accept the unfavorable offers proposed. If the opponent is tough and is unlikely to accept the low utility outcomes proposed (and are thus holding a high threshold), we build a pool of outcomes that are in the bottom 25% in value for us while ensuring that these decoys are safe. We then construct a fake concession path to deceive the opponent that we are genuinely negotiating, rather than repetitively offering the decoy, and thus is able to also conceal our bids from opponent agents that is tracking offer progression.

Alternatively, if the opponent is not tough, we only build one fixed decoy that is low in value for both us and our opponent. Because the opponent is likely to concede, we would not want the opponent to agree on an offer that is still relatively low utility for both our agent and the opponent. In practice, typically only this branch is taken, as most opponent agents are likely to concede from testing. Yet by having these two strategies, it makes our agent resistant against different opponent agent strategies.

4 Acceptance Strategy

When deciding whether to accept an offer, we first check that the utility of the proposed offer is at least the reservation value. Then typically, our agent will attempt to accept an offer which is at least a dynamic target value, which varies depending on the relative time in the negotiation (starting high early in the negotiation and decreasing as the relative time increases). This is modeled by the `_target()` helper function, which consists of a Boulware model, a predicted

ceiling of the highest offer that the opponent will eventually offer by performing a curve-fit of their utility versus time.

Finally, once the negotiation is in the final two steps, the agent will accept any offer above the reservation, as not reaching an agreement will typically result in lower utility. An exception however is if we are the first mover, we select an ultimatum offer instead of accepting any offer that is above our reservation value. This ultimatum offer is the outcome with max utility that is both greater than the acceptance bar and the predicted reservation value (lowest self-valued own offer). This is because the agent is able to control the final proposal, and thus takes the chance that the opponent may accept this proposal. Additionally, if the number of outcomes is very small, we accept reasonable outcomes rather than try to wait for an outcome with more value, and thus better avoids not reaching an agreement.

5 Opponent Model

While updating the opponent model, we track the opponent’s predicted best offer along with its utility, which will be used to rebuild two models of the opponent concurrently: a frequency-based model that is used to decide internal decisions such as bids, decoys, and the endgame, while also using a RankFit model which will be used externally for the opponent utility function. This is because the frequency-based model is more effective for optimizing for advantage, while the RankFit model is more optimized for the Kendall Tau-B metric.

In the `_rebuild_model()` function, we first rebuild the frequency model. Here we count only unique offers rather than the raw frequency, as this may be susceptible to deception from the opponent. In addition, we also employ a first-appearance weighting, where we assume that early offers will be of higher utility than those proposed later. We then create a utility table for the opponent, which will become a linear additive utility function.

One initialization that also helped our agent was to set the initial opponent model to be the anti-prior of our own utilities, as most domains are typically competitive. Additionally, for the outcome that the opponent does accept, we also award bonus points, as the opponent will likely not accept an outcome that is low in utility for them.

Finally for the RankFit model, we set each issue-value pair to become binary feature when building RankFit tables. Again, we also assume that early offers are higher in utility than those later, but after fitting the data we also compute the residuals by taking the difference between the actual and the predicted. For offers that have large residuals, we assume these offers are used as decoys, and thus the model will ignore these offers by down-weighting these offers.

6 Evaluation

From our evaluations, it appears that our agent consistently wins against all provided example negotiators, and generally has a higher advantage and deception score against all examples. A local testing pipeline was also developed to better understand how the negotiator would perform under a large number of trials. From our tests, it appears that our agent has a 100% win percentage on all scenarios (with the exception of NiceOrDie) and against all example negotiators. While our local testing pipeline may not be accurate to the population of agents submitted, it provides an effective benchmark that our agent is performing well against simple frequency-based opponent modeling strategies and conceder or Boulware bidding strategies.

7 Lessons and Suggestions

A number of other strategies were also tested, with one in particular being heavily considered is in the bidding strategy. In addition to the general phantom decoy phase, a number of constraints were also chosen to limit which outcomes to propose, and then randomly selecting from the pool of outcomes that fit these constraints. Although this may have occasionally performed better against agents that were aware of the initial phantom decoy phase, this overall did not improve the advantage or Kendall Tau-B score. This also indicated that specifically selecting outcomes that are meant to deceive the opponent agent’s opponent modeling strategy typically works better than using randomization to deceive.

Conclusions

The Phantom8 agent shows how using a phantom decoy phase for a concealing bidding strategy, a dynamic estimated target acceptance strategy, as well as having both a frequency-based and RankFit opponent modeling strategy, can be effective in achieving both a high advantage and Kendall Tau-B score. Given more time, some possible improvements that could be made in the future would be improving the opponent modeling strategy by better inferring their bidding strategy and adjusting the weights given to proposals based on this information, which could possibly be performed using an LLM model. Another improvement would be to predict if the opponent is ignoring our initial decoy phase, and counter this strategy through adapting and not repeatedly offering the same decoy outcome when detected. Overall, this agent performs well against agents that use frequency-based models for opponent modeling and concede easily during our endgame strategy.