CS 4710 - Artificial Intelligence
HW3 - Negotiating

Justin Ingram (jci5kb)
Megan Bishop (mgb5db)

Why do you see the results you see? What does this tell you about the quality of your agents? Briefly discuss some conclusions you make given the data you've presented.

Our Strategy:
All of our bots work under a scheme to try to guess the utility that the opposing negotiator is going to gain from any offer. Based on what we assume their preferences are we are able to calculate what we believe is their gained utility from any offer put forth by either of us. We start by assuming that their first offer is the best and then watch for any offer that would be better than that, at which point we update our list of assumptions.

Our Negotiator:
jci5kb_Negotiator:

This is our most intelligent negotiator, and the one we wish to submit to battle the other negotiators. The bot works by guessing the utility the enemy negotiator is receiving. It does so by assuming that the first offer the bot will make is, or is close to, the enemy's preferences. The bot will only accept an offer if it believes that it will receive more utility than the enemy. While it could be possible to trick this approach alone, the bot has another ability that it learns patterns across runs. Upon receiving the results of a successful negotiation, the negotiator compares the resulting utility to the amount of utility it believed the enemy was receiving. It stores the percentage that the guess was off, averaged with the value for all the runs for the current instantiation. In this way, the negotiator always has a good guess as to what the enemy's utility is, even without knowing the enemy's preferences. With time, this makes the negotiator make better and better guesses about the the enemy's preferences, resulting in more correct estimations of the enemy's utility, so the decision to accept an offer or not improves over time.

The negotiator is tough on opponents it faces in the way it makes offers. With each subsequent offer, the negotiator computes 500 permutations of its preferences list. However, with each additional turn taken, it performs an additional switch in the list. For example, on turn 3, permutations will have 3 switches made, meaning there should be approximately six items in different positions compared to the preferences. Permutations are then sorted, and we return the best offer for us and the worst offer for our opponent as computed by the same estimation of utility based on our prediction of the opponent's preferences.

Our Testing Negotiators:

Dictator:

Our unfriendliest of bots, it always wants to have things go its way. It will start by offering its preference list, and will decrease its gained utility by 5% on each subsequent offer until it hits its lower limit at which point it will continue to offer lists at that utility. To start the lower limit with be 75%, if it is able to come to an agreement then on the next run the lower limit will be set to whatever percentage the previous agreement was made at. The acceptance threshold is that its gained utility must be greater than that of its opponents and that the gained utility must be within 5% of the utility it would have gained from its last offer. If it is the last turn then it will accept if it is ahead.

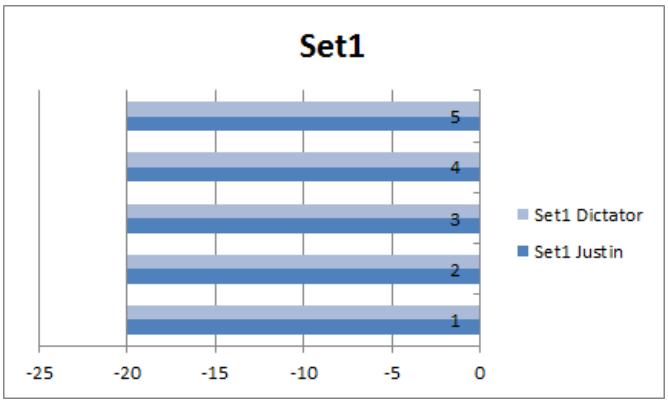
Care Bear:

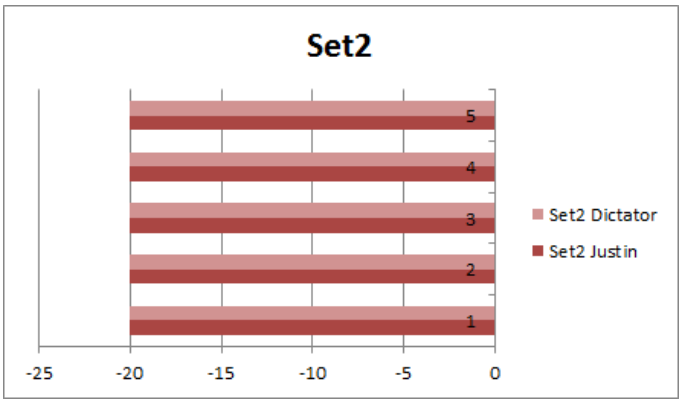
This bot will try to always come to an agreement and on the last turn will always accept the opponents offer. It will accept any offer made that gives it at least 50% of its maximum possible utility. When generating offers if looks to find one within 85% - 65% of whatever its previous offers utility was. This means that it should continually come down in an attempt to appease the opponent.

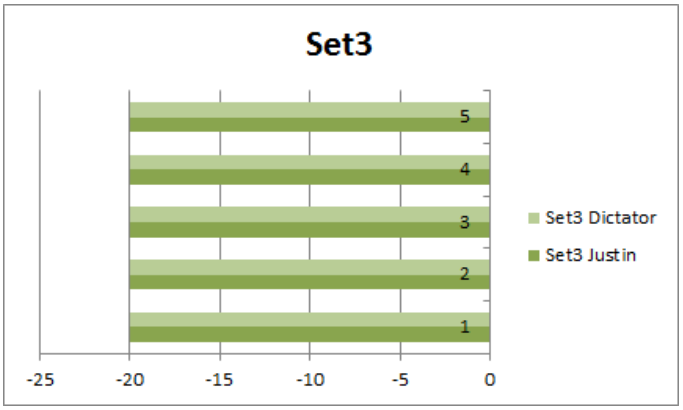
Trials:

We created three sets of data and then ran our negotiator against both of our testing bots on all of them. All sets consisted of twenty items with various rankings for each bot, negotiator A in each situation has the items referenced in order (1-20). Set 1 has negotiator B's preferences in reverse order (20-1), set 2 has every pair of items reversed (2,1,4,3,6,5…), and set 3 has the list split in half and reversed (11-20, 1-10).

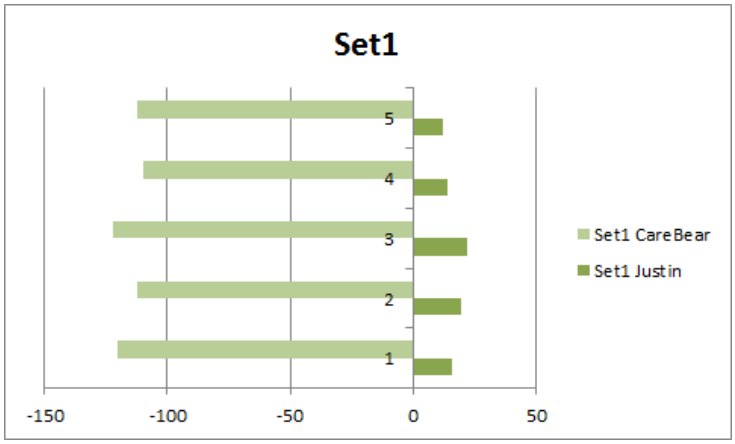
Jci5kb vs. Dictator:

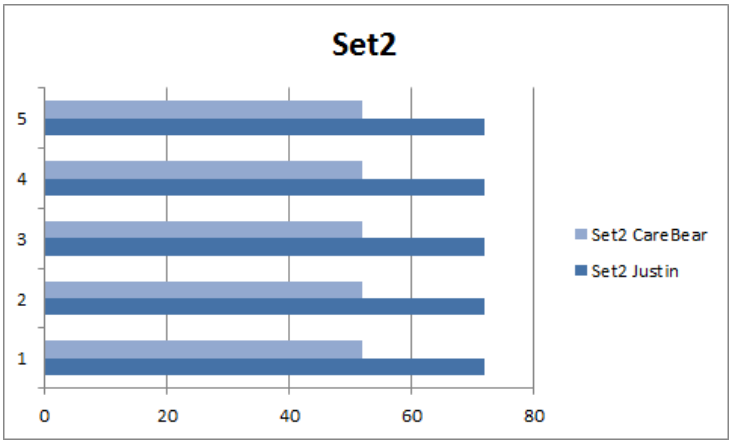


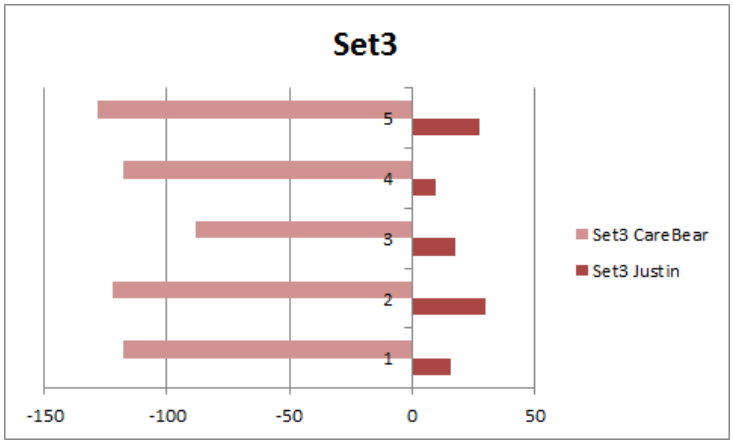




Jci5kb vs. Care Bear:







Result:

Testing jci5kb_Negotiator against Dictator results in failures each time, with each receiving the most negative score possible for the run. Though we expect we may have a few losses as a result of this, we think this is a good quality overall. This guarantees that overly aggressive negotiators will not take advantage of the program, as Professor Floryan's was able to when he did this assignment in college. By only accepting offers we expect to succeed on, we may be overly harsh, but since our negotiator is learning with each run, against a more reasonable negotiator than Dictator, we should start winning as the number of trials increases.

When testing Care Bear (CB) against jci5kb, CB always lost. This was to be expected as our negotiator is a tough one and will continually push for an agreement in its interest while CB will bow down. What was not expected was how CB had difficulties finding offers to put forth, making it a much worse negotiator than intended. This is because of the sheer number of permutations possible in a list of even modest size. When each bot was given a preference list that closely resembled the others then it was able to come to an agreement so that it only lost by a small amount, otherwise its inability to find good offers meant that it lost by a great deal.