CS4710: Artificial Intelligence
Homework 4: Negotiating
Summer 2015

Introduction:

For this assignment, you will be implementing an agent that negotiates against other agents created by your peers. We will provide you some framework code to work from.

Each negotiation will be over a set of items that have a utility value specific for each negotiator (not necessarily the same). The negotiation structure will consist of alternating offers with a finite turn limit. Each offer will be the set of items a negotiator wishes to take for itself. Both agents will receive a substantial negative reward if no agreement is made. The reward you (and your opponent's) agent receives from an agreement is a function over the associated utility of each item you end up receiving(which your agent will need to reason about at runtime).

Background – The Simulator:

The simulator code (provided on Collab) contains four files. The first is called *negotiator_framework.py*. You should not change the code in this file, but you may look at this code to better understand how the system works if you'd like. The code in *negotiator_framework.py* is the testing harness we will use to exercise your code. It takes two negotiator subclasses (see the note on subclass naming below) and runs them against each other on several problem instances. Problem instances are described in CSV files detailing the list of "items" and the associated utility of each item for each negotiator. The goal of each negotiator is to get the other to agree on a division of items that maximizes its utility received; the utility received is just the summation of each item's utility specific to that negotiator. We will provide you with at least one sample input domain as a CSV file.

The second file is *negotiator_base.py*. The *Negotiator* class in *negotiator_base.py* defines the basic methods which - at a minimum - you will need to implement (as they are used by the framework). The file contains comments describing their purpose. Note that you are welcome to add other methods to the class (you will probably want to do this) for internal use; however, you must not remove any of the methods given in the existing file. We will provide a random negotiator implementation as an example (the third file – negotiator.py).

The final file is GUI.py. This file is responsible for graphing the results of rounds and will be amended mid-way through the assignment to include a Human UI interface for you to play against an AI. You should not be adding anything to this file.

Class Negotiator from negotiator_base.py

__init__(self)	*Constructs a negotiator object. Does not require any parameters.*
initialize(self, preferences)	*Automatically invoked by the framework code. Does*

	any initializations required by the agent before a round of negotiation begins.
make_offer(self, offer)	*Invoked by the framework when it is this agent's turn to make an offer. The offer parameter is the opponent's latest offer for this agent to consider. Returns this agent's new offer. Returning the set_diff of the offer passed in is equivalent to accepting the offer and ending the negotiation.*
utility(self)	*Returns the utility of the current offer.* **You should not be altering this method**. *If you do you will receive no credit for the assignment.*
receive_utility(self, utility)	*Stores the scaled utility the other negotiator received from the last offer it made. This is called BEFORE make_offer() in the negotiation framework.*
receive_results(self, results)	*Store the results of the last series of negotiation (points won, success, etc.)*
set_diff(self)	*Returns the set difference of the total objects and the current offer. Useful for accepting offers made by an opponent.*

GUI.py, Graphs, and User Interface:

A File named GUI.py is provided. Currently, this allows you to see graphical information on how the two AIs performed given a specific round as well as overall for all rounds. The ability to run this requires importing packages (matplotlib and numpy currently) in the project interpreter.

Later in the assignment, a simple User Interface will be made available to you through GUI.py. This will allow you to make negotiations with an AI (either one you developed or another). While not necessary for completing the assignment, the UI may be helpful in collecting information about your AIs performance.

Getting Started:

First, we recommend getting acquainted with the code by implementing a simple negotiator that has little intelligence. Extend the negotiator class and implement all of the required methods. Then, run the code and see how your "dumb" agent negotiates against itself. Then, start to extend the functionality of your agent little by little.

You might also be interested in either building two different agents or testing your code with your peers to see how your agent does against other implementations.

Requirements:

You must submit the following:

- A SINGLE file with all of your code. This file should be called <computingId>.py (comp. id of either of the team members is fine). All of your code should be in this one file. **The class that I will need to instantiate for the contest should have the exact same name as the python file**.

- Produce a pdf documenting your design, implementation, and results of test negotiations.

- Submit the files separately. Please do not zip them up.

Writeup:

Produce a document that describes, at a minimum, the following aspects of the assignment:

- Describe, in detail, the functionality of your agent and defend the motivations for your implementation.

- Present some quantifiable results of your agent negotiating with other agents. Why do you see the results you see? What does this tell you about the quality of your agents? How would you improve your implementation?

- Briefly discuss some conclusions you make given the data you've presented.