Equilibrium agent for playing the Weighted Voting Game

Tal Pascal, Matan Kintzlinger

Introduction

***** HAPPINESS *****

Once upon a time in a land far far away the “Coalition Game” landed in the borders of the northern castle. Many men from all around the kingdom traveled to participate in the upcoming event revealing the incredible “Coalition Game” to have a chance and negotiate their way into a fully functional and working coalition to control the northern kingdom.

The “Coalition Game” is quite simple:

- The game may contain a variable number of rounds, which are either known or unknown to the participants.
- Each player has his party’s size which is visible to all.
- For each round a player is chosen (for that round only), and proposes a distribution of the price (100 points for example) for the players. If enough players agree to the proposal the game ends with the players who agreed to the proposal as the coalition. Else, the next round is played.

Whoever is in the coalition receives a price as the price he was proposed. Everyone else receives 0. The game might end if there are no more turns and not enough people agree to the proposal at every step in the game.

A single round of this game can be solved easily - every player should accept every offer, since it will be better than receiving 0 when the game fails.

However, for a higher number of rounds the NE of the game is quite different and is displayed in [1] very thoroughly for a game of known game length (number of rounds).

Before this NE was displayed, an experiment was conducted regarding multiple humans playing the “Coalition Game”, sometime with other humans and sometimes an agent also played the game with them. The experiment was conducted with a web-based system for playing the “Coalition Game”.

We will test this NE agent against humans using the same system with a few changes.

What did we get

We received the original web-based system which includes a management panel, a gui for playing the game and the server of the game. It is written in c# and uses .asp for the web features. Furthermore, configuration files are used to initialize the different aspects of the game - the size of the players parties, existence of an agent, the propositions and acceptance policy for the agent.

The second module we received was the code which computes the NE displayed in [1], it is written in Matlab.

What we have done

1. We created a format of configuration file for the equilibrium agent for the coalition game. The new format is as follow:

    [

#players => number of players
#rounds => number of rounds for the game
[players weight] => weight for each player (in the order of the players' index)
AI devision[#rounds][#players] => a devision for the AI player, the devision depends on the index of the AI player, (thus depends on the weight of the AI)
proposer-timeout => the timeout for the proposer to propose
responder-timeout => the timeout for the responder to respond

]

For example:
#players: 4
Weights: (6,8,2,2).
#Round: 5
The table below consist the minimum acceptance amount of the equilibrium agent, which was created by the Matlab code written by Moshik. Each column is the round and each row is the player. (e.g. if the agent is player 2 and the current round is 2, the agent will accept minimum amount of 47.9166666666667 [which will be rounded to 48] as an offer).

Round1	Round2	Round3	R4	R5
26.4467592592593	17.3611111111111	20.8333333333333	25	0
20.6597222222222	47.9166666666667	37.5000000000000	25	0
26.4467592592593	17.3611111111111	20.8333333333333	25	0
26.4467592592593	17.3611111111111	20.8333333333333	25	0

The output configuration of this game will be:
[4,5,6,8,2,2,27,20,27,27,18,48,18,18,21,38,21,21,25,25,25,25,0,0,0,0,30,20]

2. We created changes of the C# code to be able to handle the new format, in addition to the old format, and we make it possible to add as much new configuration formats as needed. The instructions for adding new agent format are:

1. The new format configuration file should contain a string-signature of the format, like the "NORMAL" string in the beginning of RoomComf.json (Appendix 1)
2. Add new class which extends "Configuration" abstract class.
3. If you need to add more fields to the "Configuration" abstract class, you can do this freely
4. The constructor of the new class should parse the new configuration file according to the new format. The constructor should get as a parameter the number of players in the room, and get the appropriate raw configuration (an array of double values) from DAL.GetSingleRoomConfigurationForSize(num_of_playes), as can be seen in "NormalConfiguration.cs" or "NEConfiguration.cs"
5. Add the new type of format to the "Type" enum
6. Add the new type of format to the functions "createConfiguration" and "setTypeFromString" of the "ConfigurationType" class. The strings of the switch-case of the function "setTypeFromString" is the name from (1).
7. add additional condition to the function "ResponseAsAi" in "CRoom" class. This function is responsible for the case where the agent is a responder. For example, if

the agent is NE-Agent, we accept only if item.Value.playerOffer >= room.Value.configuration.AiDevision[room.Value.currentRound][aiIndex]
8. add additional condition to the function "PlayAi" in "CRoom" class. This function is responsible for the case which the agent is the proposer. For example, if the agent is NE-Agent, it will offer from room.Value.configuration.AiDevision[room.Value.currentRound].
9. For the game to be played with the new format, the configuration file should be located in: C:\Sites\Coalition\RoomConf.json. This file is being loaded when the admin enters the "ManagementPanel" page.

What do we need to do

Now, after we have the coalition platform working, we need to arrange some experiments to test the equilibrium agent, and then analyze the results. Some result that could be interesting are:

1. Who played better, the agent or the real players (Who got more money)
2. How close are real players to the NE of the game (played by the agent).
3. Different expectations requesting for yourself (as a proposer) between an agent and a human player (per round)
4. Average utility (over all rounds, over rounds as proposer, over rounds as responder.
5. Accaptence rate for the agents propositions (did human players consider the NE a good offer?).
    - How many players accepted his offer (even when a coalition wasn't agreed upon).
    - How many Coalitions did the agent achieve.

Appendices

```
NORMAL
[
    [3,2,4,6,20,20,60,0,50,50,0,10,90,20,50,80,90,30,5],
    [3,1,1,9,10,0,90,0,10,90,0,5,95,5,5,90,90,30,5],
    [3,3,4,4,40,30,30,10,80,10,10,10,80,10,50,50,90,30,5],
    [4,3,4,4,4,40,30,30,0,10,80,10,10,10,10,80,10,10,50,50,10,10,50,50,10,30,20,5],
    [5,8,2,2,2,2,90,10,0,0,0,50,50,0,0,0,50,0,50,0,0,50,0,0,50,0,50,0,0,0,50,80,20,20,20,20,90,30,5],
    [5,6,4,2,2,2,60,40,0,0,0,40,60,0,0,0,40,0,50,10,0,40,0,10,50,0,40,0,10,0,50,50,10,10,10,10,90,30,5],
    [5,9,1,1,1,1,95,5,0,0,0,60,40,0,0,0,60,0,40,0,0,60,0,0,40,0,60,0,0,0,40,90,10,10,10,10,90,30,5]
]
```

Appendix 1 - RoomConf.json

Bibliography

[1] Mash, Moshe, Yoram Bachrach, and Yair Zick. "How to Form Winning Coalitions in Mixed Human-Computer Settings."
[2] http://www.dummies.com/programming/c-sharp/
[3] https://martinfowler.com/books/refactoring.html