CSE 6369 - *Multiagent Systems*

Project 1- Spring 2018

Due Date: March 26 2018

Resource Bidding

In the following simplified scenario jobs from different agents have to compete for available resources[1]. Multiple (k) computer users (submission agents) want to submit their jobs to a single computing resource. To do so, they have to bid for the resource with cyber money with each bidding agent starting out with $\$m$ in the beginning of the first round. Each agent can only bid integer amounts and the highest bidder wins the resource (in case of a tie, the earlier of the bids wins). Computing resources become available sequentially and each job needs the same resources and has the same utility ($3) if it is successfully put on the resources (i.e. if the corresponding submission agent wins the bidding). To limit the time it takes to determine a winner, each agent is only allowed one bid with subsequent bidders knowing the value of the previous bids. Every round the agent bidding last in the previous round becomes the one starting the next round of bidding. The submission process continues until all n resource slots are taken (i.e. it repeats for n rounds).
Since bidding occurs sequentially and every agent bids only once per round, this scheme can be modeled as a perfect information game and thus contains a subgame perfect pure strategy Nash equilibrium.

1) Consider this game with two submission agents ($k = 2$), a starting purse for every agent of $3 ($m = \$3$), and three available resource slots ($n = 3$). Agent 1 starts the first round.

   a) Show the perfect information game in extensive form for this problem.

   b) Show the induced normal form for this problem. To limit the size of this, you can first list all the pure strategies for each agent (i.e. the action labels of all rows and columns) and then create only a 20x20 submatrix of the full normal form game.

2) Implement a decision agent that computes a subgame perfect Nash equilibrium for the resource bidding problem.

   a) Implement a solution scheme for the specific scenario from part *1*[2].

   b) Implement a more general solution for the two player version of the problem for arbitrary values of m and n (i.e. for arbitrary numbers of rounds and arbitrary start purses).

Note: You can use any programming language for the implementation part as long as it can be compiled and run on *omega.uta.edu* (if it does not you will have to provide the instructor with the necessary software and infrastructure to compile and run the code during office hours). Make sure your code is well documented and that there is sufficient documentation to compile and run the code.

[1] Note: This is not intended to be a realistic system for job submission in practical computer systems

[2] You can either use backward induction or the linear programming notation for the normal form games to compute the equilibrium. In the latter case you can use available linear program solving software to solve the linear programs.