Model Proposal

### Introduction

PRISM Case Study : **Alternating Offers Protocol** <http://www.prismmodelchecker.org/casestudies/negotiation.php>  
Wikipedia : <https://en.wikipedia.org/wiki/Rubinstein_bargaining_model>

Paper : <http://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/entcs2006.pdf>

### Goals

1. Implement **Alternating Offers Protocol** (AOP) in python.

2. Extend model to have the new objectives and decisions below (deepens problem space explored by AOP).

3. Run optimizers (GA, DE, PSO, etc), collect data and compare to paper’s results.

4. Extend model to have multiple players (multiple buyers/sellers, models supply/demand effects on the AOP).

5. Run Step 3 again.

### Keywords

|  |  |
| --- | --- |
| **Term** | **Description** |
| Agent | Buyer (b) and Seller(s) |
| Action | (i) throwing a proposal (offer), or (ii) accepting the most recent proposal |
| Strategy-conceder | if the player is willing to yield a lot in the early phase of negotiation |
| Strategy-boulware | if a player is willing to concede considerably only when its time deadline is approaching |
| Negotiation Decision Function (NDF) | a function of time that determines a player's strategy |
| T\_b (Time-deadline ) | Agent b quits if no agreement |
| IP\_b (Initial Price) | Agent b first offer |
| RP\_b (Reserved Price) | the threshold above (below) which player b will certainly reject offer player b offers RP\_b only at time T\_b |

### Decisions

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Exists in the PRISM Model** | **Decision** | **Description** |
| 1. | Yes | Strategy | Linear or non-linear (conceder / boulware) |
| 2. | Yes | Continue | If a player wants to continue negotiation |
| 3. | Yes | Reserved Price | Threshold where a player rejects offer 100% of the time |
| 4. | Yes | Time-deadline | Time that player will quit negotiation if no agreement |
| 5. | No | Goods-deadline | Time that the goods being negotiated expire causing pure loss for the seller player |
| 6. | Yes | Bid | Price at which the buyer player wants to buy an item |
| 7. | Yes | Counter Bid | Price at which the seller is willing to sell to a buyer who has bid |
| 8. | Yes | Result | Agreement/non-agreement of the seller to a bid. This can result in the acceptance of a bid or a counter bid offer. |
| 9. | Yes | Boulware Strategy Switch Time | When a player begins to start conceding from a boulware strategy |
| 10. | Yes | Conceder Strategy Switch Time | When a player stops conceding based on the offered value based on reserved price and |
| 11. | No | Goods-Amount | The amount of goods being sold by a seller |
| 12. | No | Goods-Needed | The amount of goods needed by a buyer |
| 13. | No | Players | The amount of buyers and sellers in a marketplace |
| 14. | Yes | Strategy Gradient | Non-linear strategies are approximated and consist of conceder segments followed by a boulware segment in a conceder tactic or in a boulware strategy. Each player can have their own gradient. |

### Objective

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Exists in the PRISM Model** | **Objective** | **Goal** | **Description** |
| 1. | Yes | utility | Maximize | utility depends on the value at which an agreement is reached ( and time) |
| 1.1 | Yes | utility-time | Minimize | For buyer and seller |
| 1.2 | Yes | utility-bought-price | Minimize | For buyer |
| 1.3 | Yes | utility-sold-price | Maximize | For seller |
| 1.4 | No | utility-amount-bought | Maximize | For buyer, as % of need |
| 1.5 | No | utility-amount-sold | Maximize | For seller, as % of stock |
| 2 | Yes | Time | Minimize | Time for outcome |
| 3. | Yes | outcome | A/D | Agreement or disagreement |

Appendix 1 : Discussion

Date: 10/20/16

1. Chose - Alternating Offers Protocol
2. Increase Number of Decisions ( 10-20 ) as per Professor.
3. Avoid binary decisions in the above 10-20
4. Objectives should be around 10 .
5. Add Task Run PRISM Model checker and visualize preliminary results
6. Add Task Read paper to see which optimizers were tried
7. Add Task - Run optimizers and test using stat.py to see clustering.