The Bargaining Game

A case study in Prolog Programming
This is also your assignment
Join the **Tournament**!

 Edward Tsang (All rights reserved)

The Bargaining Game 3.3 (2004)

1. Seller is given (a) cost;
2. Buyer is given (a) utility;
3. Both players know: cost ∈ [MinC, MaxC], utility ∈ [MinU, MaxU], Cost_per_iteration, MaxDays.
4. The seller doesn't know the buyer's utility;
5. The buyer doesn't know the seller's cost;
6. Either play can terminate the bargain at any time by bidding "0"
7. The players make alternate bids, starting with the seller;
8. When both players bid for the same price, a sale is agreed;
9. Bargaining terminates when any player decides to stop, or maximum day has been reached;
10. Past bargaining is forgotten;
11. The history is given to the player (in reverse order), e.g.:
    [+106, -106, +192, -53, +215]
    [+0, -100, +200, -100, +250, -100, +300]

 Edward Tsang (All rights reserved)

Design of the mediator

- Ask: what information is relevant?
- Minimum information:
  - mediator(Seller, Buyer, ParamFile, Logfile, NoI)
- For convenience:
  - mediator(Seller, Buyer, ParamFile, Logfile, NoI, SellerProfits, BuyerProfits)

 Edward Tsang (All rights reserved)

Predicate mediator/7 (simplified)

```
mediator(Seller, Buyer, Param, Logfile, NoI, SProfits, BProfits ) :-
    mediator_retract_all( mediator_log(_, _, _, _, _) ),
    concat_strings( 'buyers/', Buyer, BFile ), consult( BFile ),
    concat_strings( 'sellers/', Seller, SFile ), consult( SFile ),
    consult(Param ),
    negotiation( NoI ),
    mediator_writes_to_log_file( Seller, Buyer, Logfile, NoI ),
    summarize_results(Seller, Buyer, NoI, SProfits, BProfits ).
```

 Edward Tsang (All rights reserved)

Design of negotiate/?

- Assume: buyer, seller & parameters loaded
- negotiate(NoI) or
  negotiation(NoI, SellerProfit, BuyerProfit)?
  - NoI in '0,000, hence used *assert* & *retract*
  - mediator_log(iteration(25), seller(142, 8, 9), buyer(252, 10, 101), History, sales_agreed(151)).
  - History: [+151, -151, ..., +452, -100, +454]

 Edward Tsang (All rights reserved)

Simplified negotiation/1

```
negotiation( 0 ).
negotiation( NoI ) :-
    NoI > 0, N1 is NoI - 1, negotiation( N1 ),
    /* generate random Cost, Utility, DTB and DTS */
    negotiate( Cost, Utility, DTB, DTS, [ ], History, Result ),
    profits(Cost, Utility, Result, SellerProfit, BuyerProfit ), !,
    assert( mediator_log( ... ) ),
    mediator_reports_one_negotiation( ... ).
```

 Edward Tsang (All rights reserved)

Predicate negotiate/7

```
negotiate( _, _, _, _, History, History, sales_agreed(AgreedPrice) ) :-
    sales_agreed( History, AgreedPrice ), !.
negotiate( Cost, Util, DTB, DTS, [ ], History, Result ) :-
    DTS > 0, DTS1 is DTS - 1, seller( Cost, DTS, [ ], AskingPrice ),
    negotiate( Cost, Util, DTB, DTS1, [+AskingPrice], History, Result ).
negotiate( Cost, Util, DTB, DTS, [-P | BidSF], History, Result ) :-
    DTS > 0, DTS1 is DTS - 1, seller( Cost, DTS, [-P | BidSF], AskP ),
    negotiate( Cost, Util, DTB, DTS1, [+AskP, -P | BidSF], History, Result ).
negotiate( Cost, Util, DTB, DTS, [+P | BidSF], History, Result ) :-
    DTB > 0, DTB1 is DTB - 1, buyer( Util, DTB, [+P | BidSF], BidP ),
    negotiate( Cost, Util, DTB1, DTS, [-BidP, +P | BidSF], History, Result ).
negotiate( _, _, 0, 0, BidHistory, BidHistory, timeout(both) ).
negotiate( _, _, 0, _, BidHistory, BidHistory, timeout(buyer) ).
negotiate( _, _, _, 0, BidHistory, BidHistory, timeout(seller) ).
```


Selected predicates

- sales_agreed([-P, +P | _], P).
- sales_agreed([+P, -P | _], P).

Tournament

```
neg_game( SellersList, BuyersList, Exp_Para, NoI ) :-
    neg_game( SellersList, BuyersList, Exp_Para, NoI, [ ] ).
neg_game( SellersList, BuyersList, Exp_Para, NoI, Parameters ) :-
    'neg_game: delete'( logfile(Logfile), Parameters, OtherParameters ), !,
    tell(Logfile),
    neg_game( SellersList, BuyersList, Exp_Para, NoI, OtherParameters ),
    told.
neg_game( SellersList, BuyersList, Exp_Para, NoI, Parameters ) :-
    ( 'neg_game: delete'( seed(Seed), Parameters, _) ;
     'mediator: random'(1, 10000, Seed) ), !,
    'neg_game: experiments'( SellersList, BuyersList, Exp_Para, NoI, Seed ).
```


Running Experiments (Simplified)

```
'neg: experiments'( Sellers, Buyers, Exp_Param, NoI, Seed ) :-
    neg_game_signature(Signature),
    'neg: writelist'([nl, Signature, nl]),
    findall( (Seller-SellerProfit, Buyer-BuyerProfit),
            'neg: match_one_pair'( Sellers, Buyers,
                    Exp_Param, NoI,
                    Seed, Seller, Buyer, SellerProfit, BuyerProfit ),
            Results),
    'neg: summarize_results'( Results, Sellers, Buyers, NoI ).
```


Match one pair

```
'neg: match_one_pair'( SellersList, BuyersList, Param_file, NoI,
        Seed, Sellerfile, Buyerfile, SellerProfit, BuyerProfit ) :-
    member( Seller, SellersList),      /* rely on backtracking */
    member( Buyer, BuyersList),
    mediator(seed(Seed)),              /* reset the seed */
    mediator( Seller, Buyer, Param_file, nologfile, NoI,
            SellerProfit, BuyerProfit ).
```


Jacob-Seller (dgjaco_s)

- Accepts bids that are above the cost by a predefined margin,
- Or when it judges (based on the bids history) that the buyer has reached its limit.
- Start offer: cost plus a predefined premium
- General rule: This offer is reduced linearly until 4th final day. It then offers cost plus a target profit (parameter to the program) for one move. The penultimate move makes an obvious drop in price to tempt the buyer. A minimum profit is demanded in the final offer.

Stacey-Seller (rpstac_s)

- Complex seller: 18 rules for various situations
- Drive hard bargains by various sensible means.
- When the bid is above cost, the bid is accepted if
  (i) the last two bids are 50% above cost;
  (ii) the last three bids are 25% above cost; or
  (iii) the last four bids are 15% above cost.
- Final 2 days' strategies fine tuned with 7 rules
  - depending on its predetermined margin thresholds
  - and the buyer's latest offer.
- General rule: reduce offer by 7.5% of the cost per round, as long as the offer is above cost.

Monday, 02 February 2009 Edward Tsang (All rights reserved) 39

Bargaining

Research Frontier

Monday, 02 February 2009 Edward Tsang (All rights reserved) 47

Bubinstein's Two players alternative offering game

Player 1: How about 70% for me 30% for you?

t = 0, Player 1's pay off is 70%

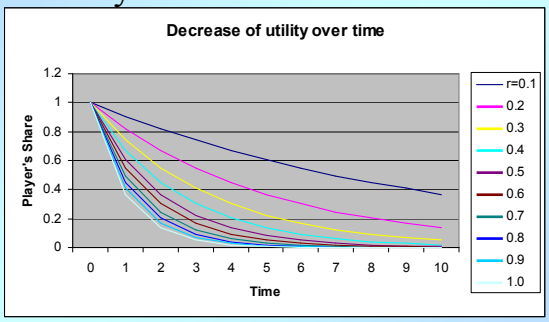
Player 1: No, how about 50-50?

t = 2, Player 1's pay off is $50\% \times e^{-0.1 \times 2} = 41\%$

- If neither players have any incentive to compromise, this can go on for ever
- Payoff drops over time – incentive to compromise
- A's Payoff = $x_A \exp(-r_A t\Delta)$ Let r_1 be 0.1, Δ be 1

Monday, 02 February 2009 Edward Tsang (All rights reserved) 48

Payoff decreases over time



Monday, 02 February 2009 Edward Tsang (All rights reserved) 49

Evolutionary Computation for Bargaining

- Conventional methods find strategies mathematically
  - However, it is laborious
  - It also assumes perfect rationality
- Co-evolution is an *alternative approximation* method to find game theoretical solutions
  - Perfect rationality assumption relaxed
  - Relatively quick for approximate solutions
  - Relatively easy to modify for new models
- Genetic Programming with incentive / constraints
  - Constraints helped to focus the search in promising spaces

See http://www.bracil.net/finance/bargain for details

Monday, 02 February 2009 Edward Tsang (All rights reserved) 50