

Auction Agent: Model Solution

By Adam Richardson

Steps

1. Modeling
2. Strategize

Steps

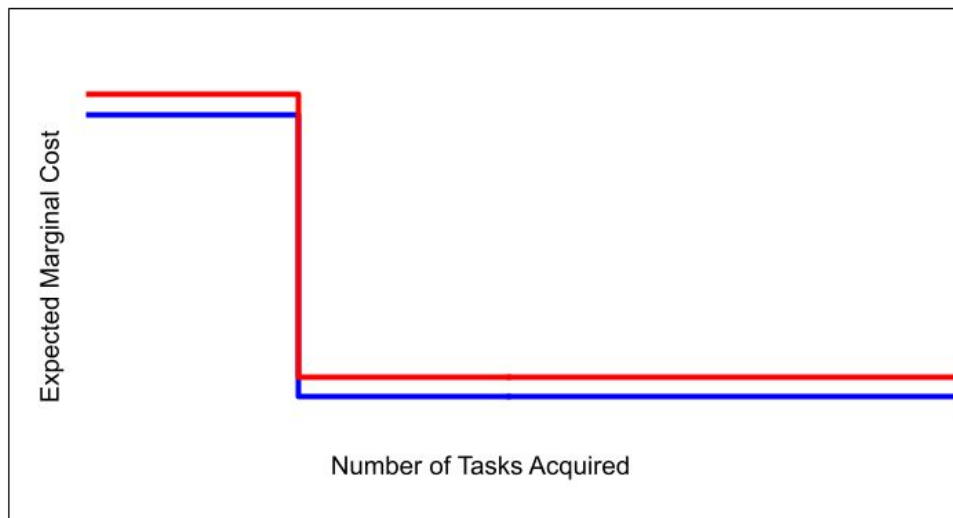
1. Modeling
2. Strategize

Gather Information

- What information is at our disposal a priori?
 - Shared topology, task distribution
 - Our vehicles
- What additional information do we gain during a bid?
 - Auctioned task
 - Opponent's bidding history
- What can we do with this?
 - Estimate marginal cost of delivering the auctioned task (Stochastic Local Search, Simplex)
 - Model the future
 - Model opponent behavior

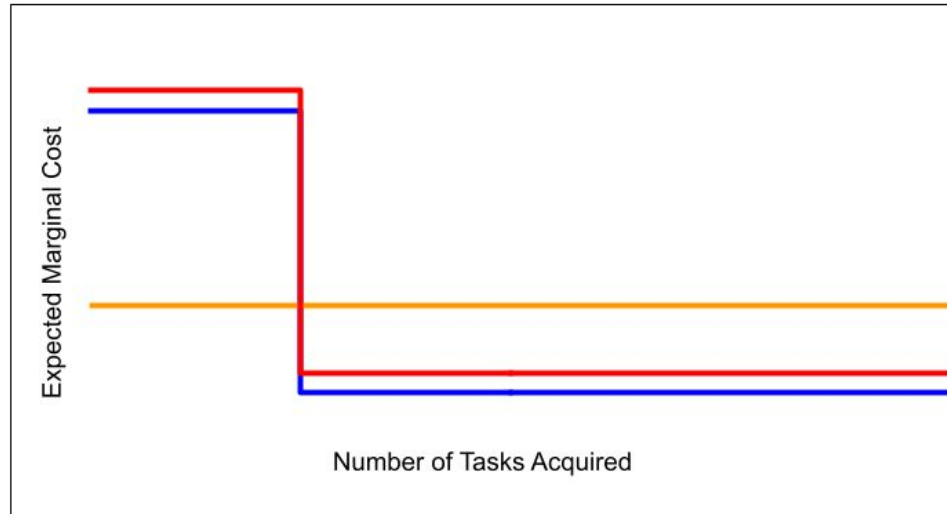
Modeling

- Model the future
 - How will future marginal costs change when we accept a task?
 - A task might have a high current marginal cost, but significantly lowers future marginal costs, resulting in a lower long term marginal cost



Modeling

- Model the future
 - How will future marginal costs change when we accept a task?
 - A task might have a high current marginal cost, but significantly lowers future marginal costs, resulting in a lower long term marginal cost



Modeling

- Model the future

```
public double estimatePotentialForFutureCostReduction(Candidate solution, Task task, int numberOfFutureTasks, double timeout) {  
    int numberOfRuns = 4;  
    double savings = 0;  
    Candidate augmentedSolution = new Candidate(solution);  
    augmentedSolution.addTask(task);  
    for (int i = 0; i < numberOfRuns; i++) {  
        List<Task> additionalTasks = new ArrayList<Task>();  
        for (int j = 0; j < numberOfFutureTasks; j++) {  
            Task potentialTask = distribution.createTask(); // Sample potential future tasks  
            additionalTasks.add(potentialTask);  
        }  
  
        double marginalCostSolution = marginalCostEstimation(solution, additionalTasks, timeout / numberOfRuns / 2); // Compute the ma  
        double marginalCostAugmentedSolution = marginalCostEstimation(augmentedSolution, additionalTasks, timeout / numberOfRuns / 2);  
  
        savings += (marginalCostAugmentedSolution - marginalCostSolution) / numberOfRuns;  
    }  
    return savings;  
}
```

Modeling

- Modified marginal cost estimation

```
@Override
public Long askPrice(Task task) {
    double discount = 0.1;
    System.out.println("Auctioning task: " + task);

    List<Task> additionalTasks = new ArrayList<Task>();
    additionalTasks.add(task);
    double myMarginalCost = marginalCostEstimation(mySolution, additionalTasks, timeout_bid * 0.25);
    double opponentMarginalCost = marginalCostEstimation(opponentSolution, additionalTasks, timeout_bid * 0.25);

    double potentialFutureSavings = Math.min(estimatePotentialForFutureCostReduction(mySolution, task, numberOfFutureTasks: 2, timeout_bid * 0.25), b: 0);
    double myCost = Math.max(a: 0, myMarginalCost + discount * potentialFutureSavings); // potentialFutureSavings is <= 0

    potentialFutureSavings = Math.min(estimatePotentialForFutureCostReduction(opponentSolution, task, numberOfFutureTasks: 2, timeout_bid * 0.25), b: 0);
    double opponentCost = Math.max(a: 0, opponentMarginalCost + discount * potentialFutureSavings);
}
```


Modeling

- Model Opponent's Behavior
 - VERY unconstrained, but can make some reasonable assumptions
 - Opponents want to be profitable, opponents will behave similarly to us:
 - Opposing bids will be on average slightly higher than modified marginal cost
 - We don't know the opponent's initial vehicles!
 - Naive: model opponents using our own vehicle configuration
 - Model Solution: estimate decaying average of ratio of opponent bid to opponent marginal cost.

```
public void updateOpponentCostMultiplier(long opponentBid) {  
    double discount = 0.2;  
    if (opponentBidEstimate == 0)  
        return;  
    double ratio = opponentBid / opponentBidEstimate;  
    opponentCostMultiplier = discount * ratio + (1 - discount) * opponentCostMultiplier;  
    return;  
}
```

Modeling

- Model Opponent's Behavior
 - VERY unconstrained, but can make some reasonable assumptions
 - Opponents want to be profitable, opponents will behave similarly to us:
 - Opposing bids will be on average slightly higher than modified marginal cost
 - We don't know the opponent's initial vehicles!
 - Advanced: Probabilistic Modeling
 - Opponents bid some multiple $1+\epsilon$ of their marginal cost with some prior on ϵ .
 - Solve for initial configuration with maximum likelihood.
 - Solve for maximum likelihood ϵ .

Steps

1. Modeling
2. Strategize

Strategize

- Balancing Risk

- Advanced: with a probabilistic model of opponent's configuration and bidding strategy, use regret matching
- Naive Solution: bid Modified Marginal cost when necessary, try to grab a bit of profit when you

can:

```
opponentCost = Math.round(Math.max(opponentCostMultiplier * opponentCost, 0));
opponentBidEstimate = opponentCost;

System.out.println("My cost estimate: " + myCost);
System.out.println("Opponent's cost estimate: " + opponentCost);

double bid;
if (opponentCost < myCost)
    bid = myCost;
else
    bid = myCost + discount * (opponentCost - myCost);

if (bid == 0)
    bid = 250 + random.nextInt(bound: 500);
```