# Ad Auction - Diamond Design and Planning

Itai Rosenblatt
Adam Fox
Efi Ben Israel
Eylon Yogev

**Introduction**
In this document we will show our design for the TAC AA agent.
As instructed, our agent consists of three main modules:

1. Modeler - provides the CPC, position, and number of Impressions for each query and bid.
2. Estimator - by using data given by the Modeler, provides profits and conversion estimation. further more, calculated the CTR and conversion rate
3. Optimizer - by utilizing the data from the Estimator, provides the best bid set for the following day.

In our design we chose to implement the above modules using the following approaches:

**Modeler**
In this module we will address two different problems:

1. Estimating the users distribution across the states (NS,IS....) - this is done in order to calculate the number of Impressions. For this we will use MCMC algorithms and particle filtering, using the former at the beginning while there is still little data available, and switching to the last when it's "good enough".
   As another approach we will solve a set of equations constructed from the impressions and choose/construct the best fitting one.

2. Modeling the other advertisers - this is done in order to calculate the CPC and position. This part of the modeler is responsible of estimating the bids submitted by other advertises, for each query, and predict future bids. One possible approach is using a particle filtering approach, that relies on sampling of the rankings of the advertisers, and a model of the bidding behavior of the advertisers, this approach is elaborated in [3].

**Estimator**
In this module we will use several approaches simultaneously:

1. Perform a "straight on" calculation based on the data given by the Modeler
2. use several time-series algorithms.

in order to receive the best results we will calculate the optimal weighted average between the different approaches.

**Optimizer**

The Optimizer needs to solve two separate optimizations problems :

1. Single day
2. Multi day

In order to solve these problems we will test several knapsack algorithms and choose those most fitting.

In the first few days, when the population inhibiting the purchasing states is small, we will try and estimate some of the game parameters (e.g. the minimum bid).

**General Strategy**:
In the following parts we will describe a few approaches for each module.
If by testing them off-line we will come to the conclusion that there is more then one fitting algorithm, we will try and combine all of them in the game, by switching between them according to the success rate in the real games.

We will now elaborate on the algorithms chosen by us for each module.

## Modeler

The first task of the Modeler is to estimate the number of impressions for any possible bid. To complete this task we will want to model the users' distribution over the different states. The users are distributed among the states NS, IS, F0, F1, F2 and T and transfer between them with known probabilities. We would like to estimate this Markov Chain and then use the estimation to calculate the total number of impressions for each query.
We will test a few different approaches:

Particle filtering:
We will start using the Kalman filter (EKF or UKF) first, because we have little sample data. We will run the particle filter in the background and will switch to it when it becomes more accurate than the previous method.
In parallel, we will try using CFTP (coupling from the past) and in particular Read-Once CFTP, for it has some advantages:

a. Save of computer memory
b. Simpler to program.
c. Multiple independent samples from the distribution can be obtained in a single run of the algorithm.

Metropolis-Hastings:
Another approach we will try is trying algorithms as general Metropolis-Hastings algorithm, the Metropolis sampler, and the Gibbs sampler. The HYDRA java library can supply the platform for testing this approach.

Direct approach:

As an additional approach we will construct 3 equations (with 4 variables) given from the impressions and from the parameters data.

$$Xf0 + \frac{1}{3}Xis = Y0$$
$$Xf1 + \frac{1}{3}Xis = Y1$$
$$Xf2 + \frac{1}{3}Xis = Y2$$

we will solve these equations 4 times, by setting a different variable each time, using data from the previous day distribution.
We will create a function to choose the best solution from the above equations- either choosing the best fitting one, or creating a new solution as some combination of the four.

<u>Testing these methods:</u>
We will build a simulation environment for the population distribution with the same values as in the game simulation. The simulation can help us run many quick tests to get to best results without the latency of the game simulation.

Concerning the modeler's second task (advertiser bid estimation) we use particle filtering, as mentioned above. One problem is that we don't have access to the rankings of the bids of the other advertisers; on the other hand we do have access to the average position of the other advertisers. Therefore we will calculate the rankings using the average positions using an approach devised in [4]. In order to model the bidding behavior of the other advertisers we will be using logs of previous games and a machine learning algorithm. The logs will provide actual bids that the other advertisers made. The machine learning algorithm we will probably use is from the library WEKA machine learning toolkit [5].

## Estimator

In this mode we are required to estimate the profit and sales of a bid set of all queries. We assume in this part, the we have the data given from the Modeler part, that is the Impressions and the population distribution.
We will try to estimate the CTR and the Conversion-Rate through a probabilistic calculation assuming the Modeler's data is accurate.
This calculation heavily depends on the accuracy of the Modeler, so it will be wise to use it only when having a good Modeler component. Hence, we will perform a few time series approximations based on the history of the CTR and the Conversion-Rate.
Finally we will calculate a wighted average between the above estimations.
We will want to weight each component based on their success. For that we will test their accuracy, and change the weights accordingly, in run-time during the actual game.
For testing the accuracy we will check their results on the past K days (K will be determined through out testing) to see how good was each approximation, compared to the real results.

Then, penalties and bonus will divided by the results.
The problem of finding the best wights distribution, could be addressed as an optimizations problem, and we will try solving it using simulated annealing.

**Time Serials approximation:**

a. ARIMA: we will try out different parameters for the ARIMA estimation giving out a different approach each time.
b. Statgraphics product: Uses different TSA and picks the one the seems best for the given data.

**Testing**:
We will want to test which methods combinations work better, what starting points should we pick and what constants work best for each method.
We will try out the different strategies against existing agents and see how well we perform.
Furthermore, when targeting an specific parameter/ method combination, we will create 3 different permutations along with the original one, and test it with against a dummy agent and 3 good agents.
Thus giving us a game with 8 agents all together, half of them are our experiments, and the rest are for control.

**Risks:**

- The modeler's estimations can have a large effect on the Estimator's calculations. Hence, error on the modeler's part may cause errors on this part as well.
- Optimizing the wights and calculating the approximations may take a long time. Getting a good result within the time window can be hard.

## Optimizer

The goal of the optimizer is to maximize the profits. It decides what should be the Bids and Limits for all 16 queries for all 60 days. It takes into consideration the constraints that we are encouraged to enforce in order to get the maximum profit by the end of the game. Some of these constraints (those that will prove to be most influential on the implementation of the optimizer) entail a capacity of products we can sell in a "window" of 5 days, before suffering a penalty on additional conversions.

We are going to solve two separate problems: Single day (SD), Multi day (MD).

**Single Day**
We view the SD problem as a PMCKP (Penalized Multi Choice Knapsack Problem), and using GreedyMCKP algorithm to solve it.
Optional methods of solving SD problem [1]:

1. Exhaustive Greedy PMCKP – optimal but slow (consider using a changing discretization

factor)
2. Dynamic Greedy PMCKP (faster, but less optimal). Time complexity: O(n^2*logn)
3. Hybrid Greedy PMCKP (faster than dynamic, less optimal). Time complexity: O(n^2)

We are preferably going to implement the Dynamic Greedy PMCKP (2). WE provide a high level description of Dynamic Greedy PMCKP:
Input: for each query - two vectors: values, weights, and capacity: C.
Output: for each query - which bid was chosen for each query, and the total value for the day.
The algorithm iterates over sets of pairs: (value, weight) and creates incremental items*. It chooses the best incremental item (pair) with regards to efficiency (value/weight), and updates the capacity used for the day. For each query - the first n incremental items chosen correspond to the n'th item in the original set of pairs. The output for each query is the bid that corresponds to the pair that was chosen, and the value, that is the sum of values of all the pairs that were chosen.

* Incremental items are calculated as follows: sort pairs in non decreasing order of weight. Remove dominated, and LP dominated items. Create a differential vector (between successive pairs).

**Multi Day**
Optional methods of solving MD problem [1][2]:
1. Extending SD problem to a MD problem. Viewing MD as a PMCKP problem and solving using Dynamic Greedy PMCKP. (This approach does not separate MD and SD)
2. Hill-Climbing algorithm. (MD uses the function SD).
    a. Initialize daily capacity for all days to be CAPACITY/WINDOW_SIZE
    b. Do while profitable change exists (do not exceed N_HC_ITER iterations)
        i. for t = today…(today + N_DAYS) do
            2)b.i.1. Calculate DeltaT+/- as specified in slides. (Modeling Optimization, slide 19)
        i. Adjust C(t) according to maximum DeltaT.

Further explanation: 2)b.ii: The calculation of Delta T+/- uses the function MD. The function MD (as defined in slide 19), works as follows:
It gets as input the utilized capacity for the last 4 days (window), and allowed capacity for the next M_DAYS.
First: initialize value = 0.
1. for t = today…(today+M_DAYS) do
    1.1 value += SD of last utilized and allowed capacity for the day.
Time complexity: O(M_DAYS * Time complexity(SD))

We are preferably going to implement the Hill-Climbing algorithm using heuristics (which will set the time complexity accordingly to be: O(Time complexity(MD)*N_DAYS*N_HC_ITER).

**Optional heuristics:**
- Restricting the amount of days we calculate:

    - In Hill-Climbing algorithm: N_DAYS, heuristic to be determined.
    - MD will not sum over all days, but for some M_DAYS.
- Time restrictions on calculation (using what we have after a certain time)
- N_HC_ITER – number of iterations in hill-climbing algorithm.

[1] A First Approach to Autonomous Bidding in Ad Auctions/ Jordan Berg, Amy Greenwald, Victor Naroditskiy, and Eric Sodomka
[2]TacTex09: A Champion Bidding Agent for Ad Auctions/ David Pardoe, Doran Chakraborty, and Peter Stone
[3]A Particle Filter for Bid Estimation in Ad Auctions with Periodic Ranking Observations, David Pardoe and Peter Stone
[4]D. Pardoe, D. Chakraborty, and P. Stone. TacTex09: Champion of the first Trading Agent Competition on AdAuctions. Technical Report AI-10-01
[5I. H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. / http://www.cs.waikato.ac.nz/ml/weka/