Keyword Bidding in Sponsored Search Using an Estimation of Distribution Algorithm

Aparna Sundara Rajan

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Master of Science

University of Washington

2011

Program Authorized to Offer Degree: Computing and Software Systems

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

Aparna Sundara Rajan

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Dr Ankur Teredesai

Dr Matthew Alden

Dr Daniel Zimmerman

Date: _____

Signature_____

Date_____

University of Washington

Abstract

Keyword Bidding in Sponsored Search Using an Estimation of Distribution Algorithm

Aparna Sundara Rajan

Chair of the Supervisory Committee:
Professor Dr Ankur Teredesai
Computing and Software Systems

Sponsored search is an important form of Internet advertising today. With very limited space available for such advertisements, search engines use an auction mechanism to divide the space among multiple advertisers. Advertisers place bids on desirable keywords, which are then ranked to determine the position of each advertiser's ads. Determining an optimal bid is a challenge for an advertiser with a limited marketing budget. High bids can lead to higher advertising costs, and thus loss of profit, while low bids risk poor ad position, and thus loss of revenue.

In this thesis we introduce a new method for developing profitable bidding strategies over a fixed set of keywords. We evolve bidding strategies using the Markovian Learning Estimation of Distribution Algorithm (MARLEDA), a type of search algorithm that optimizes solutions through statistical modeling. To validate our method, we implemented our strategies as agents to compete in the Trading Agent Competition's Ad Auctions (TAC/AA) game. We observed that the evolved bidding strategies were able to make a significant profit and were able to perform better than some of the leading strategies from the 2009 and 2010 TAC/AA competitions.

TABLE OF CONTENTS

Page

LIST OF FIGURES

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to University of Washington, where I have had the opportunity to do my graduate studies and successfully complete this thesis work. I would like to thank Dr Matthew Alden whose enthusiasm, inspiration and his great efforts to explain things clearly and simply, helped me understand and enjoy evolutionary algorithms. I would also like to thank Dr Ankur Teredesai who motivated me to take up this research, and whose support made it all possible.

DEDICATION

to my dear husband, Srivatsan

Chapter 1

INTRODUCTION

When we use Google or Bing or any other search engine, we often see advertisements or "sponsored links" corresponding to our search keywords (see Figure 1.1 for an example). Advertisers buy ad space on the web pages produced by popular search engines and place advertisements to promote their products alongside the regular search results. The allocation of these advertising slots and their pricing is done via auctions run by the search engine ad platform. Since the introduction of this concept in 1998 [4], sponsored search has evolved into a major source of revenue for the search engine giants such as Google, Yahoo!, and Bing. The success of sponsored search can be attributed partly to its effectiveness at producing relevant (targeted) advertising, and partly to the appealing framework that allows even small-scale advertisers to use it easily and effectively while only paying when their ad is clicked upon.

When a user enters a keyword query into a search engine, he gets back a page with results, containing both the links most relevant to the query and the sponsored links, i.e., paid advertisements. When a user clicks on a sponsored link, he is sent to the respective advertiser's web page. The advertiser then pays the search engine for sending the user to his web page.

As Figure 1.1 illustrates, sponsored links are typically clearly distinguishable from the actual search results. However, the visibility of a sponsored search link depends on its location (slot) on the result page. Typically, a number of advertisers naturally prefer a slot with higher visibility, i.e., towards the top of the list of sponsored links. Hence, search engines need a system for allocating the slots to advertisers and deciding on a price to be

charged to each advertiser. Due to increasing demands for advertising space, most search engines are currently using auction mechanisms for this purpose. These auctions are called *sponsored search auctions*. In a typical sponsored search auction, advertisers are invited to submit bids on keywords, i.e., the maximum amount they are willing to pay for a user clicking on the advertisement displayed on the search results page when a user submits a query containing those keywords. Based on the bids submitted by the advertisers, the search engine picks a subset of advertisements along with the order in which to display them. The actual price charged, i.e., the *cost per click (CPC)*, also depends on the bids submitted by the advertisers.

Due to a practically limited campaign budget for each advertiser, strategic bidding behavior plays a crucial role in sponsored search auctions. From an advertiser's perspective, an ideal strategy has to prevent oneself from overbidding, while at the same time bidding just enough to obtain his desired position, thereby beating competing advertisers. Even though advertisers generally want to increase the number of users clicking on their ads, they don't want user clicks that do not result in conversions (sales). For every click an advertisement receives, the advertiser is bound to pay the CPC value to the publisher (the search engine). Therefore, an ad is profitable only if the revenue from click-to-sale conversions is greater than the cost of all user clicks. Similarly, a publisher generates revenue from user clicks, and thus is interested in maximizing CPC values while maintaining a trustworthy environment for the search user and the advertiser.

In this thesis we introduce a new design for an automated intelligent bid-generating agent (i.e., an algorithm, to be implemented as a computer program) that will help an advertiser maximize his return on investment by optimizing the position his ads get and the corresponding traffic to his website.

1.0.1 Problem Statement

In this thesis, we look at the sponsored search problem from the perspective of an advertiser. For any auction mechanism followed by the publisher, advertisers face the problem of how

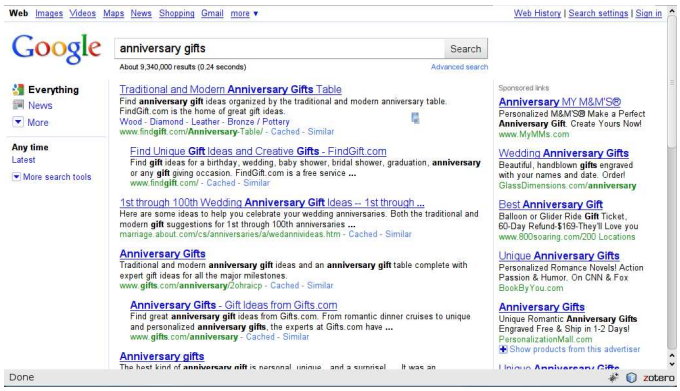

Figure 1.1: Result of a search performed on Google for the query *anniversary gifts*. The sponsored search results (on the right) are clearly distinguishable from the algorithmic search results (on the left). Each time a user clicks on a sponsored link, the advertiser pays the search engine a *cost per click (CPC)* price for sending that user to his web page. Higher slot positions have a higher CPC.

to generate bids over time in a competitive and highly dynamic ad market. We assume that the advertiser has already selected a set of keyword queries that he wishes to place bids on, and we focus on solving the problem of coming up with the best bids for these keywords. Note that this problem is different from identifying the most profitable set of keywords among millions of available keywords as is done e.g. in [23]. In an actual search engine marketing campaign, both problems need to be solved.

> *The goal of this thesis is to learn and devise an optimal bidding strategy. Such a strategy automatically determines how much to bid for a given keyword query on a given day in order to maximize profits by keeping costs per click down while achieving high conversion rates from potential customers.*

To this end we have:

- Formulated the task of finding optimal bids as a combinatorial optimization problem which aims to find near optimal solution from a set of candidate solutions that are

composed of individual parameters. These parameters are finite valued, and hence the solution space is combinatorial in the number of parameters.

- Evolved intelligent bidding strategies using an Estimation of Distribution Algorithm (EDA) and developed an algorithm to use these bidding strategies as agents that can submit meaningful bid values(see chapter 5).

- Implemented this algorithm as an agent (computer program) for participation in the an instance of Ad Auctions game of the Trading Agent Competitions (TAC/AA)[1].

- Compared the performance of these evolved strategies with a set of known good strategies in this domain

- Outlined the challenges we faced and suggested improvements that could add value

[1]http://www.sics.se/tac/, http://aa.tradingagents.org/

Chapter 2

TRADING AGENTS COMPETITION - AD AUCTION

To encourage the research community to develop bidding strategies, in 2009 a new game was introduced in the Trading Agent Competition (TAC), a series of international research tournaments to promote research and education in the technology underlying trading agents.

Participants, or agents in the TAC Ad Auctions game (TAC/AA) [12], representing Internet advertisers, bid for search engine advertisement placement over a range of interrelated keyword combinations. A back-end search user model translates placement over each simulated day to impressions, clicks, and sale conversions, yielding revenue for the advertiser. Advertiser strategies need to combine online data analysis and bidding tactics to compete for maximum profit over the simulated campaign horizon. The TAC/AA scenario is designed to include many of the interesting strategic aspects of sponsored search auctions, while being repeatable and computationally amenable to empirical analysis. This is why we choose to develop a bidding strategy within this framework, but the approach that we will develop will be applicable outside of the game setting too. Recall that there are three entities involved in a sponsored search scenario: users, advertisers, and a publisher (or search engine) that brings them together. The game environment simulates the behavior of the users and the search engine: the search users and the publisher follow fixed (stochastic) policies built into the game environment. The agents, on the other hand, follow policies implemented by competition entrants.

In the game, participating advertisers (agents) have an interest in the keywords Lioneer, PG, Flat, TV, Audio, and DVD. The first three correspond to the names of manufacturers, while the others are components in the home entertainment market. The game environment simulates users who launch queries that contain one, two, or none of these keywords. Mentioning neither a component nor a manufacturer is called an F0 level query. Mentioning

Query Focus level	Query (manufacturer,product)
F0	— , —
F1	(Lioneer, —),(PG, —),(Flat, —) (—, TV),(—, Audio),(—, DVD)
F2	(Lioneer, TV),(Lioneer, Audio), (Lioneer, DVD) (PG, TV),(PG, Audio),(PG, DVD) (Flat, TV),(Flat, Audio),(Flat, DVD)

Table 2.1: The 16 different query types in the TAC/AA game, with their focus levels

one or the other, but not both, is denoted an F1 level query. Mentioning both component and manufacturer is called an F2 level query. In total, there are 16 distinct queries, as illustrated in Table 2.1.

A game instance corresponds to a period of 60 days [1]. At the beginning of each day, each participating advertiser submits a bid for each of the 16 query kinds from Figure 2.1. In addition, each advertiser also informs the publisher of the kind of ad to display for each of the 16 query kinds: either a targeted ad mentioning a particular product, or a generic ad. Along with the bid bundles, advertisers may also submit individual daily spend limits for each of the 16 query types, as well as a limit for the total expenditure across all queries. When a spend limit is reached, the affected ads will no longer be shown to users for the rest of the current day.

After all advertisers have submitted their bids for the day, the publisher executes an ad auction for each query type to determine the ad positions and cost per click. Simulated users then issue queries, receive search results, and consider clicking on ads. When a user

[1] In the game environment, the length of one such day takes a few seconds in real time

clicks on an ad, the corresponding advertiser is charged the cost per click determined by the ad auction process. The simulated user then determines whether or not to purchase a product from that advertiser. If the user makes a purchase, the event is called a *conversion*, and the advertiser earns revenue. As new queries are launched throughout the day, the publisher monitors the spend limits of the advertisers and reruns ad auctions as necessary.

Although every advertiser sells every product, there are two differences between individual advertisers assigned by the game environment at the beginning of each game instance. First, each advertiser specializes in a particular manufacturer and a particular component (say PG, TV). The specialization affects conversion probability and sales profit. Users with a component preference matching the advertiser's specialization (e.g. TV) are more likely to purchase once they reach the advertiser's landing page, i.e., the odds of conversion are higher. Furthermore, an advertiser receives a manufacturer specialist bonus for every sale of a product from the manufacturer matching its specialization (e.g. PG). Second, each advertiser is assigned a distribution capacity of low, medium, or high. This is used to model the phenomenon that if advertisers sell too much products in a short period, their inventories run short and they have to put items on backorder. As a result, shoppers will be less inclined to purchase, and conversions suffer, i.e. the odds for converting decrease. Roughly speaking, the distribution capacity of an advertiser directly relates to the number of products he can sell in an aggregation window of $W = 5$ days without risking a backorder penalty. For more details, refer to [12].

Finally, at the beginning of each day, each advertiser receives three reports based on events from the previous day, namely a daily query report from the publisher, a daily account status report from the bank, and a daily sales report from the sales analyst enumerating the sales for each of the 16 query kinds [12]. The daily query report for an advertiser contains, among other information, the average position of the slot obtained by that advertiser for each of the 16 query kinds. The reports about day d-1 are only made available after advertisers have placed their bids for day d. As a result, information about events that took place on day d-1 can be used at the earliest to influence bids for day d+1.

Chapter 3

ESTIMATION OF DISTRIBUTION ALGORITHMS

Estimation of Distribution Algorithms (EDAs) ([16],[14]) are a new approach to solving combinatorial optimization problems. EDAs are non-deterministic search algorithms that maintain a population of individuals similar to Genetic Algorithms (GAs) [9]. While in GAs crossover and mutation operations are used to produce a new population of individuals, in EDAs these operations have been replaced by the learning and sampling of a probability distribution (refer to Figure 3.1). This distribution is estimated from selected individuals of the previous population.

Such probabilistic search algorithms forgo systematic exploration of a candidate solution space in favor of randomized search strategies. Guided by an objective function, such techniques attempt to explore the search space to find near optimal solutions by building probabilistic models of promising solutions found so far [21]. However, because probabilistic search algorithms do not search the entire search space, they are generally not *complete*, i.e. they are not guaranteed to identify optimal solutions. In spite of this weakness, by searching only a fraction of the solution space, probabilistic search algorithms are practical methods for producing near-optimal (and occasionally optimal) solutions quickly. In classic GAs, the genetic operators such as selection, recombination and mutation work well under the assumption that high-fitness values (which is desirable) chromosomes are located 'near' other high-fitness chromosomes or their recombination. This assumption is generally termed as a 'building block' of desirable solutions. This assumption might often mislead the search space if the optimal solution is a function of dependent or correlated genes. Such dependencies can be expressed in a statistical model that can be sampled to produce the next generation of candidate solutions. EDAs address the building block problem by statistically inferring dependencies among genes. These dependencies are expressed in a statistical model, which

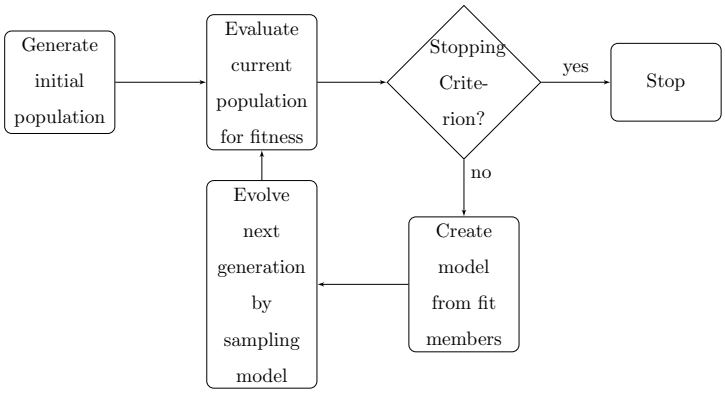

Figure 3.1: The general flow of an EDA

can then be sampled to produce a new population. Through the sampling process, EDAs are likely to preserve high-fitness combinations of alleles, making the search process more efficient. For a comparative evaluation of GAs and EDAs, see e.g. [19].

The power of these algorithms depends on two factors: (1) how appropriate the statistical model is to the domain, and (2) how well the system can learn it. The majority of EDAs incorporate models that organize domain structure in directed acyclic graphs (DAGs), such as Bayesian networks or Gaussian networks. There are many established learning mechanisms for DAGs and simple methods for sampling the resulting model. However, directed graph models are not necessarily the most natural representation of domain structure for all problems. For example, a DAG cannot represent, by definition, bi-directional or cyclic dependencies. Several researchers have proposed using undirected graph models in EDAs ([26], [25]). In particular, Markov random fields (MRFs) have been shown to be a promising basis for EDA models [27]. However, learning and sampling MRFs is more difficult than for DAGs, and had been so far constrained their implementation. MARkovian Learning EDA (MARLEDA) [2] is a new EDA that makes use of Markov Random Fields to model the domain. More on implementing this algorithm can be found in chapter 5.

In this problem of finding an optimal bidding strategy for Ad Auctions, the solution space consists of chromosomes that represent agents that are evolved through generations

to get optimal bidding strategies. Individual genes (refer Figure 5.1) represent bid adjustments for specific advertising scenarios. We do not assume that the bid for all scenarios can be independently optimized. For example, assume that in a particular generation, a set of chromosomes that faired better than the rest of the population had a striking bias towards bidding higher for the specialty matching feature. Since each allele value in a chromosome is considered as a function of three parameters (chapter 5 and [17]), it is highly possible that the overall fitness of a chromosome that determines the bid adjustments, is influenced by combinations of allele values in it. In other words, there could be interdependencies between the genes of every chromosome that leads to a final bid value for that chromosome. Therefore, an optimization algorithm that identifies and exploits gene interdependencies should be more effective than an optimization algorithm that is ignorant of gene interdependencies. Without any specific knowledge about the structure of the interdependencies, a more generic model (undirected rather than directed) is a good starting point.

EDAs have been successfully used in complex real-world applications in place of GA based optimization systems ([20, 14]). However, application of an EDA to solve a problem with the scope of TAC/AA is completely new to the best of our knowledge.

Chapter 4

RELATED WORK

Rapid growth in the online-advertising industry and its relationship with search engines is attracting interest in the topic of online keyword based ad auctions. In a sponsored search scenario, the publisher (search engine) invites the advertisers to bid for the ad space, which is typically the position for placing their ads. Advertisers would bid based on the amount they are willing to pay to the publisher if a user clicks on their ad or if their ad is able to make an impression. A number of research work have been done towards such pricing strategies used by search engines, particularly towards per-click pricing rather than per-impression strategies [11].

Ongoing research efforts in this area include the study of sponsored search auction mechanisms with ranking of advertisements as a whole ([1], [6]) as well as the proposal of different strategies to optimize the advertisers profit. Bidding strategies in ad auctions play a critical role in deciding the revenue for advertisers. Kitts and Leblanc [13] for instance treat the problem of finding CPC offers or bids that will optimize the profit as a linear programming problem with unknown parameters that need to be estimated, such as the expected number of clicks generated by users overall, and the slot position that can be expected for a specific CPC offer. Cary et al. [7] study the revenue, convergence and robustness of greedy bidding strategies, while Liang and Qi [15] investigate cooperative and vindictive strategies. The TAC Ad Auctions game, first held in 2009, has certainly attracted more attention to this problem. Among the final competitors of the first game, the TacTex agent created by Pardoe et. al [18] at the University of Texas, Austin, estimates the full game state from limited information, such as various game parameters, and uses these estimates to make optimal action predictions. This agent was the winner at the 2009 international TAC/AA game.

Besides the work by Munsey et al.[17], who use a GA to develop an intelligent bidding strategy, several other papers have been published that give an insight into competitive strategies used in the game so far. A particularly elegant approach, developed by Vorobeychik [28], estimates the value v of a click and then bids a fraction $\alpha \cdot v$ of this value, where an equilibrium value for α is found via simulation-based game theoretical analysis. The corresponding QuakTAC agent obtained the 4th position in the 2009 international TAC/AA game. Another approach by Greenwald et al. [5] decomposed the agents problem into a modeling sub-problem, where values such as click probability and cost per click were estimated, and an optimization sub-problem, where a bid value for given these estimates was determined. They used a mixture of rule-based algorithms and multi-choice knapsack problem (MKCP) algorithm. The corresponding Schlemazl agent obtained 2nd position in the 2010 international TAC/AA game. Chang et al., [8] developed a adaptive bid value generating strategy based on a Market-based Value per Click that is estimated based on available market reports. Their agent AstonTAC was the runner-up in the 2009 international TAC/AA game.

Although evolutionary algorithms have been used before to solve problems in auction space ([22, 24, 29]), to the best of our knowledge, [17] is the first proposal to use Genetic Algorithm to find optimal bids for sponsored search auction. Even though their agent performed relatively well and came at fourth place in the competition, their approach motivated us to improvise the evolutionary algorithmic approach with more statistical learning and less randomness. We hope to investigate how an EDA might improve the performance. One drawback of their approach was that they had selected features (parameters that would affect advertiser's revenue) based on intuition. We think that performance can be improved if we have a stronger learning algorithm to learn the policy and hence derive influencing features to increase revenue. Archer et al. [3] extended [17] and tried to investigate the best features that would increase advertiser's revenue. They used data-mining tools to come up with the best features that they think would help increase revenue.

One common problem with the aforementioned GA based approaches lay in evaluating

the populations. The evaluation method used in [17] is to run a tournament of two games, between eight randomly selected chromosomes per game. After the first pair of games, a set of two more games are run with randomly selected chromosomes for each game, and the fittest chromosome is selected as the one that had highest average profit over two games. Even though this method can reveal a general trend in increasing capacity to defeat prior strategies, there are two major issues: (1) The initial population size of 16 limits the ability to effectively explore the solution space, given the size of each chromosome (Figure 5.1) (2) Fitness evaluation is a score per generation and not an absolute score over generations, which does not reveal whether a specific ranking of increasingly sophisticated strategies exist among the competing chromosomes as they evolve.

Chapter 5

EVOLVING INTELLIGENT STRATEGIES

In this chapter we formulate the keyword bidding problem as a combinatorial optimization problem by specifying the chromosome structure of candidate solutions as well as the fitness function, in the same way as one would do for a GA. We also describe the details of our application of MARLEDA to solve this optimization problem. Finally, we discuss our strategies for placing bids on the first 2 days (when no daily report information is available yet), how we choose between generic and targeted ads, and how we set spend limits.

5.1 Chromosome structure and Fitness

Each bidding strategy is evolved as a chromosome consisting of an array of real-valued genes. Each gene corresponds to a bidding situation that could occur during an Ad Auction game instance. A chromosome is mapped to an agent competing in such a game instance. When submitting a bid for query type q at the beginning of game day d, the agent identifies the set of genes corresponding to that day's situation. The value of a gene, its allele, is a bid adjustment value a in the range $(-1, 1)$ that instructs the agent how to adopt its bid $b(q, d-1)$ for the same query type q from the previous day to obtain an appropriate bid for the current day, i.e. $b(q, d) = (1 + a) \cdot b(q, d-1)$. Initially the chromosomes are populated with random values. As evolution progresses during training, the bid adjustment values are optimized for the corresponding bidding situations.

This chromosome representation effectively forms a lookup table for bid corrections, based on the current backorder situation of the advertiser, how well the query type matches with the advertiser's specialty, and the return on investment accumulated for the query type so far. Note that the latter two features are query type specific, while the first one is not. The following subsections describe each of the features in detail.

5.1.1 Backorder feature

In the game environment, at the beginning of a game instance, an advertiser is assigned a distribution capacity of either $C = 300$ units (low), 450 units (medium), or 600 units (high), corresponding to the number of units he can sell in a sliding window of $W = 5$ days without the risk of a backorder penalty. The more an advertiser exceeds his distribution capacity within a sliding window, the smaller the odds that users will place additional orders. It seems therefore worthwhile to take into account the number of items already sold in the current sliding window. We compute this as

$$B = \left(\sum_{i=(d-W)}^{d-2} c_i \right) - C$$

where c_i is the total number of conversions made by the advertiser on day i, a number which is readily available from the daily report of the sales analyst. Note that B can be positive or negative. A positive value denotes that the agent has made more conversions than what the distribution capacity allows, resulting in a true backorder. A negative value for backorder denotes that there is room for more conversions before the distribution capacity over the aggregation window is reached.

To facilitate encoding of the backorder feature in a chromosome, we discretize backorder into the set of 7 intervals used by Munsey et al. [17], namely $(-\infty, -100]$, $(-100, -51]$, $(-51, -26]$, $(-26, -11]$, $(-11, 0]$, $(0, 10]$, $(10, +\infty)$.

5.1.2 Specialty match feature

Just like advertisers "specialize" in a particular manufacturer/component pair, each simulated user in the TAC/AA game environment has a specific product preference (e.g. Lioneer, TV). A user will only buy the product he has a preference for, and not issue queries for other products in the game. Furthermore, for users with preference for a component matching the advertiser's specialization (e.g. TV), the odds of converting after an ad has been clicked-through are increased. Finally, a manufacturer receives a bonus whenever selling a product of the manufacturer in which he specializes (e.g. PG). All of the above suggests

that the degree to which a user's query matches the advertiser's specialization is useful to take into account when making bidding decisions. The specialty match feature does just that.

In other words, specialty match feature represents the degree to which a user's query matches with the agent's manufacturer specialty [12]. A keyword from the query that occurs in the agents specialization is called a hit; e.g. Lioneer is a hit in Lioneer, Audio. A keyword from the query that does not occur in the agents specialization is called a miss. The level of matching between a query and the agents specialization is defined as follows:
2 misses: VERY BAD
1 miss, 0 hits: BAD
1 miss, 1 hit: WEAK
0 hits, 0 misses: NEUTRAL
1 hit, 0 misses: GOOD
2 hits: PERFECT
We adopted this discretization for specialty match feature from [17]. This seemed to be a reasonable set of discretization for this feature to ensure that the agent bids for different queries with either a generic ad or a targeted ad. Later it was seen that agents made some significant revenue through generic ads displayed for less than 'perfect' matches.

5.1.3 Return on Investment feature

Measure of Return on Investment (ROI) characterizes the performance of a bidding strategy for a particular query on a particular day. If we need a chromosome to make decisions about the next bid for a query based on its previous performance for that query, ROI values from all the previous days will need to be used.

Return on Investment is a dynamic value that changes from day to day, whereas chromosome characteristics remain static for the entire period of a game. In order to make use of this dynamic information we define ROI parameter per query as, Let $Revenue(q, i)$ denote the revenue obtained for query q on day i. Furthermore, let $Cost(q, i) = CPC(q, i) \cdot Clicks(q, i)$,

with $CPC(q,i)$ the average cost per click for query q on day i, and $Clicks(q,i)$ the number of clicks obtained for q on day i. Note that all of this information can be derived from the daily reports. Return on investment is then defined as

$$ROI(q) = \sum_{i=0}^{d-1} \frac{Revenue(q,i) - Cost(q,i)}{Cost(q,i)}$$

ROI values determine if the agent was able to make significant profit for the queries for which it sent bid values. A negative value for the ROI would indicate that the agent incurred more cost than revenue, and hence no profit. If value of the factor is between 0 and 1, it indicates that the agent was able to balance the cost and revenue for a query. If the value is more than 1, then it indicates that the agent was able to make significant profit for the query. To use ROI as a gene characteristic, we discretized this feature into three intervals: [-∞, 0) indicating no profit, $[0,1]$ indicating balanced revenue and cost and (1, ∞] indicating significant profit made.

Position feature and its drawbacks One of the key features used to define a chromosome by Munsey et al., [17] was *ad position feature*. It was discretized as the position values in the TAC/AA game that the agent would desire to be in. The goal of this feature was to ensure that agent's advertisement stays in *impressive* positions in the ad space to increase the probability of clicks from the user. During the initial stages of developing an EDA agent, we had adopted the position feature with the same discretization used in [17]. Later during experiments we found that even if we excluded the position feature completely, agents behaved equally competent in the games. Average position for a query type is an effect of high bid values rather than a means to obtain higher profit. Features like specialty match and backorder contributes to higher conversion probability (specialty match through Manufacturer Specialty bonus) and higher conversion probabilities per query increases revenue per query. But the position feature as used by Michael would only cause the agent to bid higher to obtain higher position thus increasing CPC. It would have been helpful if we were to estimate CPC values of other bidding agents.

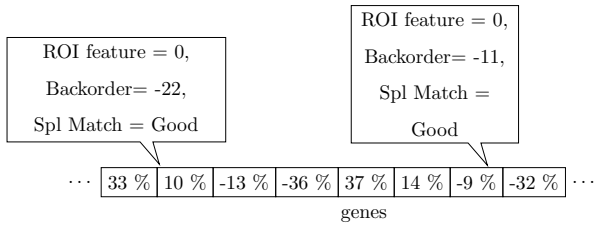

Figure 5.1: A sample Chromosome representation. Each allele value represents bid adjustment (represented here as percentage values) for each possible situation a bidding agent can be in

Also, the average positions given in the query reports for the advertisers is not exact, but rather based on a small sample average (average over sample size of 10). Moreover the only reason Michael mentions in his paper about why he chose position is because position is directly related to the number of impressions made. Even though higher number of impressions ensure that agent's ad was displayed to the search users, impressions do not affect conversion probability, and so we cannot conclude that higher positions would mean better profit.

5.1.4 Fitness of a chromosome

All chromosomes include a gene for every combination of possible values of the 3 discretized features described above, thus each chromosome is composed of 126 real valued genes (7 values for backorder, 6 values for specialty match, 3 values for ROI). Figure 5.1 depicts part of a chromosome and an example of what each gene position signifies. Initially the chromosomes are populated with random values. As evolution progresses during training, the bid adjustment values are optimized for the corresponding bidding situations. Throughout the evolution, we compute the fitness of a chromosome as the average profit value obtained by the corresponding agent in a set of 3 Ad Auction games. We chose to run at least 3 games per generation to ensure that every corresponding agent gets a fair chance of being assigned one of the 3 distribution capacities at least once.

5.2 Application of MARLEDA

Chapter 3 illustrates the general flow of an estimation of distribution algorithm. Initially, a random set of candidate solutions are generated as arrays of random real numbers within a range (-1,1). This represents the seeding population from which further evolution is carried out. Each candidate solution is a linear representation of parameters, called a chromosome that defines a bidding strategy.

5.2.1 Chromosome Selection

Every MARLEDA generation consists of a population with 24 chromosomes. The process of creating the next generation of chromosomes begins with the selection of a subset of the current population. To bias selection towards the best known chromosomes, MARLEDA uses truncation selection [10]. In truncation selection the chromosomes are ordered by fitness, and some proportion p (e.g. $p = 1/2$) of the fittest individuals are selected on an average $1/p$ times. Even though truncation selection is not generally used in the most popular evolutionary algorithms, one important property of truncation selection proved advantageous to our application. Truncation selection is known to have a lower loss of diversity compared to alternatives such as tournament selection. This property reduces the risk of premature convergence in the small populations we used. Note that MARLEDA can be set to use either tournament selection or truncation selection. As we experimented with tournament selection, we observed that the evolved population converged prematurely, and hence we adopted truncation selection instead.

5.2.2 Generating Models

MARLEDA uses a set of random variables $\{X_1, ..., X_n\}$, such that each variable X_i corresponds to a gene x_i ($i = 1 \ldots n$, and, in our case, $n = 126$). Statistical dependencies among the random variables, and therefore among the genes, are recorded in a neighborhood system, thus forming a Markov Random Field (MRF) model. The neighbor relation between

any two random variables is grounded in an observable statistical dependence within the members of the selected subpopulation. Like many EDAs, MARLEDA tests for these dependencies to learn its model. Consider a "partial" MRF whose neighborhood system does not yet fully capture all the observable dependencies. Let X_i and X_j be non-neighbors in the current neighborhood system, each with their own "partial" set of neighbors. If a dependence between X_i and X_j is observable, the neighborhood system should be updated to make X_i and X_j neighbors. Conversely, if X_i and X_j began as neighbors and a dependence is not observable, they should become non-neighbors.

In the original implementation of MARLEDA, Pearson's χ^2 test is used to calculate the confidence level of dependence between two genes. This statistical testing method is appropriate for nominal random variables modeling nominal genes. However, our chromosomes consist of real valued genes, hence we replaced the statistical testing method in MARLEDA with Spearman's ρ test. The ρ value between two random variables is calculated as follows:

$$\rho = \frac{\sum(X_i - \bar{X}_i)(X_j - \bar{X}_j)}{\sqrt{\sum(X_i - \bar{X}_i)^2 \sum(X_j - \bar{X}_j)^2}}$$

where X_i and Y_i model two distinct random genes over the selected subset of chromosomes in the current population. When two variables are in perfect correlation, their ρ value will be 1. However for our implementation, we set a relaxed threshold of 0.7, a ρ value above which indicates a good correlation, and hence suggests that x_i and x_j be neighbors.

MARLEDA constructs the MRF neighborhood system via a greedy search approach starting from a null neighborhood system. At each iteration pairs of non-neighbor genes are tested. If the confidence level of the pair is at least 0.7, the model is updated to make the pair neighbors. Similarly, pairs of neighbors are tested, and if the confidence level is below the threshold the pair is made non-neighbors. The order of all tests is randomized.

5.2.3 Generating New Chromosomes

New chromosomes are created in MARLEDA by sampling the MRF model. Sampling is performed via a Markov chain Monte Carlo process:

1. $x^{new} \leftarrow$ random chromosome from the selected sub-population
2. $i \leftarrow$ a random value uniformly sampled from [1,126]
3. Compute $\mathrm{P}(x_i|x_k, k \in \partial(i))$
4. $x_i^{new} \leftarrow$ sample from $\mathrm{P}(x_i|x_k, k \in \partial(i))$
5. Unless the iteration limit has been reached, return to step 2

5.2.4 Mutation

Even though EDAs frequently do not incorporate a mutation operation, MARLEDA has provisions to retain mutation [2]. We use a decaying mutation factor that modifies alleles as follows:

$$x_i^{new} = x_i \pm (r \times \delta)$$

where $\delta = (1 - \frac{CurrentGeneration}{TotalNumberOfGenerations}) \times 0.10$ and r is a random value in the interval $[0, 1]$. δ value thus helps decrease the effect of mutation linearly as the number of generations increases.

5.2.5 Replacement

The chromosomes in the population with lowest fitness values are replaced with the newly generated chromosomes. This step implements an elitist strategy where the "best" chromosomes are always preserved from generation to generation. We chose to have $3/8^{th}$ of the population remain as elitists.

5.3 Ad Choice and Spend Limits

Choosing ads to display and setting spend limits were part of the bidding strategy, even though they were not a part of the evolution process. For queries of type F2 (Table 2.1), corresponding targeted ads are chosen. This is to take advantage of the targeting factor [12]

that influences click probability for a displayed ad. For F0 and F1 queries, corresponding generic ads are chosen.

Daily spend limits were set for three intervals during a game. For the first 5 days, we set a fixed amount of 500 as daily spend limit. For the next 5 days, we increased spend limit by the available balance amount in the advertiser's bank account. From day 11 to the final day in the game, we did not set any spend limit.

5.4 Bidding for first two days

Since daily reports from the game server are not available before sending bid bundles for the first 2 days in the game, we had to use a slightly different strategy for these 2 days. We initialize the ROI feature value at 0, and Backorder feature value at 5 indicating that the advertiser has 0 backorder or a positive backorder upto 10. This helps the advertiser to be conservative in choosing the bid values. Then we take the manufacturer-component specialization of the advertiser which would be given before the first day (day 0) begins, to find the exact gene position with the help of specialty match discretization. For day 0, allele values are used as absolute bid values, whereas on day 1, allele values are bid adjustments to the bid values sent on day 0.

Chapter 6

EXPERIMENTS AND OBSERVATIONS

We co-evolved 24 bidding strategies through MARLEDA. During training, we decided to respect the default setting of 8 participating agents per game instance as used in the 2009 and 2010 TAC/AA competition. To speed up the fitness evaluation we therefore installed 3 instances of the game server (version 10.1.0.1) so we could run game instances in parallel. In each generation, the 24 chromosomes created by MARLEDA were linked to 24 agents split across the 3 servers, where they each participated in 3 consecutive games, accounting for a total of 9 games per generation. Figure 6.1 depicts the fitness value of the best agent per generation, for 400 generations. The agent to which the best chromosome out of the 24 after 400 generations is linked to, is referred to as the *EDAAgent* in our experiments below.

6.0.1 Performance

To evaluate the performance of our evolved EDAAgent, we lined up a team including the toughest competitors from the 2009 and 2010 TAC/AA competition that are available in the TAC online agent repository, a collection of agent binaries. The top 5 of the 2009 competition consisted of — in this order — *TacTex*, *AstonTAC*, *Schlemazl*, *QuakTAC* and *DNAgents*. Out of these, *TacTex* and *Schlemazl* also participated in the 2010 competition, obtaining respectively the 1st and the 2nd place. For our experiments, we decided to include the most recent versions available of all of these agents. For clarity, in Table 6.1 we have appended the year of the version to their name. Furthermore we selected *Mertacor*, the agent that came 3rd in the 2010 competition. The final spot was given to *epflagent*, an agent that obtained 6th place in 2009 but did not make it into the finals in the 2010 competition.

Table 6.1 shows the average profit obtained by all of these agents over a total of 40

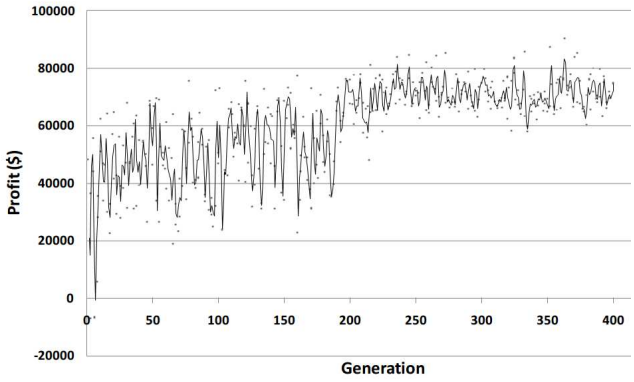

Figure 6.1: Fitness curve: average profit value of the best MARLEDA agent per generation. The *EDAAgent* in Table 6.1 corresponds to the best agent from generation 400.

TAC/AA test games. TacTex remains the undefeated champion. The other strategies from the top 3 of 2009 and 2010, namely Mertacor, AstonTAC and Schlemazl also do well. The lower than expected result of QuakTAC is due to too low bid values which resulted in infrequent display of ads and hence a low impression rate. Even though QuakTAC had a very high return on investment for queries for which its ads were displayed, its low impression rate affected the overall profit score. The epflagent did really well in some of the games, but ended up getting negative scores in some of the other games too, accounting for a low average score.

The most interesting observation for this paper is however that our EDAAgent managed to obtain a comfortable 3rd position among the leaders, and, in particular, managed to outperform DNAgents2009.

Figure 6.2 and 6.3 offer some more insight into the performance of the EDAAgent in comparison with the stronger agents TacTex and Mertacor. From Figure 6.2(a), which depicts the average clickthrough rate for all 3 agents in the 40 test games, it becomes clear that the ads of the EDAAgent attract a lot more clicks than those of the other agents, and, as Figure 6.2b documents, these clicks are also substantially more expensive on average. However, they do not lead to more, or more profitable, conversions. As is clear from Figure

Position	Agent	Average Score
1	TacTex2010	67 345
2	Mertacor2010	54 802
3	**EDAAgent**	**47 326**
4	AstonTAC2009	44 619
5	Schlemazl2010	41 926
6	DNAgents2009	40 145
7	QuakTAC2009	7 735
8	epflagent2010	796

Table 6.1: Scores of agents over 40 TAC/AA games

6.3, the revenue of the EDAAgent is only comparable to that of TacTex, and even lower than that of Mertacor.

As Table 6.2 illustrates, most of the EDAAgent's revenue stems from queries that are perfect or good matches with the advertiser's specialization. The fact that no revenue was made on neutral queries can simply be explained by the fact that the ads of our EDAAgent were not displayed for such queries (cfr. the average position of 0 for F0 queries reported in Table 6.3). It is reasonable for a bidding strategy to not aim high for these kind of queries, as the users issuing them are the least likely to buy in the game environment. Note however that we did not impose a heuristic of bidding low for F0 queries ourselves, but that this reasonable bidding behavior emerged automatically through the evolutionary learning process.

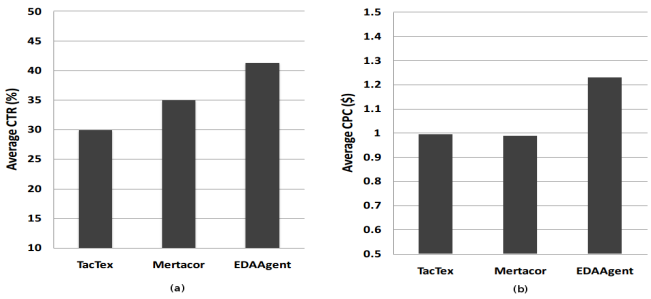

Figure 6.2: (a) Average clickthrough rate and (b) average cost per click of top 3 agents in 40 games

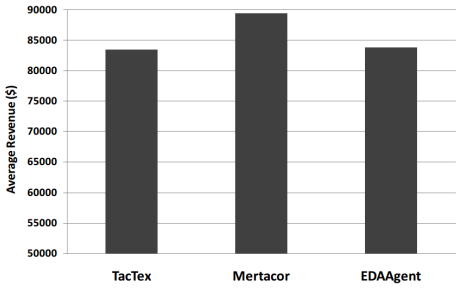

Figure 6.3: Average revenue of top 3 agents in 40 games

Specialty Match values	% Revenue made
very bad	0
bad	5.4
weak	3.6
neutral	0
good	40.4
perfect	50.6

Table 6.2: Revenue made by EDAAgent for various specialty matching values as percentage over total revenue, averaged over 40 consecutive games

Query Focus level	Average Position
F0	0
F1	2.56
F2	3

Table 6.3: Average position of EDAAgent for various query types in 40 consecutive games

Position	Agent	Avg Score	Avg CTR (%)	Avg CPC ($)	Avg Conversion Rate (%)	Avg ROI (%)
1	EDAAgent1	40 467	34	1.04	24.2	142.64
2	EDAAgent2	38 993	35	1.47	23.4	123.48.92
3	EDA_DNA_1	12 820	19	1.07	9.9	19.61
4	EDA_DNA_2	322	21	1.00	8.7	−1.33

Table 6.4: Ranking and average game statistics of agents developed using MARLEDA and differing in chromosome encoding over 40 TAC/AA games. The other 4 participants in these games were dummy agents provided by the game environment.

6.0.2 Influence of ROI feature

The approach proposed in this paper differs from the work from Munsey et al. ([17], DNAgents2009 in Table 6.1) in 2 aspects: (1) we used MARLEDA instead of a GA, and (2) we have swapped Munsey et al.'s original ad position feature for a new ROI feature. To investigate to what extent the improvement of our EDAAgent over Munsey et al.'s DNAgents can be attributed to the use of this different feature, we ran an additional experiment.

Munsey et al.'s ad position feature simply encodes the average position obtained by the ad for query type q on the previous day. This feature is discretized into 6 possible values: 1, 2, 3, 4, 5, and > 5. Value > 5 means that the ad did not get a slot position in the top 5, which might mean that it got a lower ranked slot, or no slot at all. The other features used by Munsey et al. are identical to the backorder feature and the specialty match feature described in chapter 5. Hence each chromosome corresponds to an array of $7 \times 6 \times 6 = 252$ real valued genes.

We evolved 100 generations of bidding strategies using MARLEDA and with a chromosome encoding identical to that of Munsey et al. as described above. In other words, the chromosomes were constructed with the ad position feature as proposed by Munsey et al. in place of the ROI feature that we introduced in chapter 5. We mapped the best 2 chromosomes from the 100th generation into the 2 agents called *EDA_DNA_1* and *EDA_DNA_2* in Table 6.4. We had them compete against the top 2 agents from the 100th generation from Figure 6.1, called *EDAAgent1* and *EDAAgent2* in Table 6.4, as well as 4 so-called dummy agents (bidding strategies provided by the game environment). To allow for a fair comparison, bidding tactics apart from the automatically learned ones, including spend limit and bidding tactics for the first 2 days, were set equal for all EDA evolved agents (as described in chapter 5, which differs from the way these issues are addressed in [17]).

Table 6.4 shows the result of a competition among these agents, i.e., the average score over 12 consecutive games. We observed that the strategies with the chromosome encoding proposed by Munsey et al. bid higher values to obtain higher average ad positions. However they were unable to make significant profit from the top positions consistently. This

indicates that, apart from the choice of a GA or an EDA to evolve bidding strategies, the use of a return on investment feature instead of (or perhaps, in addition to) an ad position feature has a beneficial effect.

Chapter 7

CONCLUSION AND FUTURE WORK

We have created a framework to evolve intelligent bidding strategies for ad auction mechanisms through a very powerful estimation of distribution algorithm. Our aim was to evolve bidding strategies with less randomness than a genetic algorithm approach, and our results prove that statistical modeling of successful strategies are indeed useful in creating better solutions. Even though the results are convincing, we believe that there is still additional room for improvement. Given the limitations of the TAC/AA game, we were able to define a chromosome structure that would be efficiently used throughout a game instance. However, inclusion of more parameters could theoretically improve the performance of agents, if the learning mechanism can effeciently handle the larger solution space.

We believe that the full power of MARLEDA algorithm can be exploited using a bigger population size. As the population size increases, the time required to train good strategies would increase as well. Each iteration of evolution took at least 18 minutes, and hence evolving about 400 generations was a very long process. The vast majority of this time was spent running TAC/AA games, which was further complicated by issues with the TAC/AA server such as connection time-outs.

Even though we have encoded a good set of parameters in our chromosome structure, none of the features are directly related to user distribution in the TAC/AA game. During our testing phase, we observed that user distribution seems to be influencing the profit values to a significant extent. User distribution modeling, however useful it may be, was out of scope for our work. The return on investment feature utilized in our chromosome structure indirectly captures the effects of user transactions within a game instance. We hope that if we let the evolution run for more iterations, then we might be able to obtain better strategies. Such strategies might take into account user distribution states and how

they would affect profit of a bidding agent. Our future work includes testing this hypothesis through longer evolution processes.

BIBLIOGRAPHY

[1] Aggarwal, G., Goel, A., and Motwani, R. Truthful auctions for pricing search keywords. In *Proceedings of the 7th ACM Conference on Electronic Commerce* (2006), ACM, pp. 1–7.

[2] Alden, M. *Marleda: Effective Distribution Estimation through Markov Random Fields*. PhD thesis, University of Texas at Austin, 2007.

[3] Archer, J., Devaraj, D. D., and Makarytska, N. Extending a Genetically Evolved Keyword Bidder. http://technicalreadings.wordpress.com/2011/03/23/12/, March 2010.

[4] Battelle, J. *The search: How Google and its rivals rewrote the rules of business and transformed our culture*. Portfolio Trade, 2006.

[5] Berg, J., Greenwald, A., Naroditskiy, V., and Sodomka, E. A First Approach to Autonomous Bidding in Ad Auctions.

[6] Borgs, C., Chayes, J., Immorlica, N., Jain, K., Etesami, O., and Mahdian, M. Dynamics of bid optimization in online advertisement auctions. In *Proceedings of the 16th International Conference on World Wide Web* (2007), ACM, pp. 531–540.

[7] Cary, M., Das, A., Edelman, B., Giotis, I., Heimerl, K., Karlin, A., Mathieu, C., and Schwarz, M. Greedy bidding strategies for keyword auctions. In *Proceedings of the 8th ACM Conference on Electronic Commerce* (2007), ACM, pp. 262–271.

[8] Chang, M., He, M., and Luo, X. Designing a Successful Adaptive Agent for TAC Ad Auction. In *Proceeding of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence* (2010), IOS Press, pp. 587–592.

[9] Goldberg, D. *Genetic Algorithms in Search and Optimization*. Addison-Wesley, 1989.

[10] Heinz, M., and Schlierkamp-Voosen, D. Predictive models for the breeder genetic algorithm. *Evolutionary Computation 1*, 1 (1993), 25–49.

[11] Hoffman, D., and NOVAK, D. How to acquire customers on the web. *Harvard Business Review 78*, 3 (2000), 179–188.

[12] Jordan, P., and Wellman, M. Designing an ad auctions game for the Trading Agent Competition. In *IJCAI-09 Workshop on Trading Agent Design and Analysis* (2009).

[13] Kitts, B., and Leblanc, B. Optimal bidding on keyword auctions. *Electronic Markets 14*, 3 (2004), 186–201.

[14] Larranaga, P., and Lozano, J. *Estimation of distribution algorithms: A new tool for evolutionary computation*. Springer, 2002.

[15] Liang, L., and Qi, Q. Cooperative or vindictive: Bidding strategies in sponsored search auction. In *Proceedings of the 3rd International Conference on Internet and Network Economics* (2007), Springer, pp. 167–178.

[16] Mühlenbein, H., and Paass, G. From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from NaturePPSN IV* (1996), 178–187.

[17] Munsey, M., Veilleux, J., Bikkani, S., Teredesai, A., and De Cock, M. Born To Trade: a Genetically Evolved Keyword Bidder. In *Proceedings of IEEE CEC at WCCI2010 (2010 IEEE World Congress on Computational Intelligence)* (2010), WCCI.

[18] Pardoe, D., Chakraborty, D., and Stone, P. TacTex09: A champion bidding agent for ad auctions. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)* (2010).

[19] Pelikan, M. Analysis of estimation of distribution algorithms and genetic algorithms on nk landscapes.

[20] Pelikan, M., and Goldberg, D. Hierarchical BOA solves Ising spin glasses and MAXSAT. In *Genetic and Evolutionary ComputationGECCO 2003*, Springer, pp. 213–213.

[21] Pelikan, M., Goldberg, D., and Lobo, F. A survey of optimization by building and using probabilistic models. *Computational optimization and applications 21*, 1 (2002), 5–20.

[22] Phelps, S., Cai, K., McBurney, P., Niu, J., Parsons, S., and Sklar, E. Auctions, evolution, and multi-agent learning. In *Proceedings of the 5th , 6th and 7th European conference on Adaptive Agents and Multi-Agent Systems* (2008), Springer, pp. 188–210.

[23] Rusmevichien tong, P., and Williamson, D. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *Proceedings of the 7th ACM Conference on Electronic Commerce* (2006), Citeseer, pp. 260–269.

[24] Sáez, Y., Quintana, D., Isasi, P., and Mochon, A. Effects of a rationing rule on the ausubel auction: a genetic algorithm implementation. *Computational Intelligence 23*, 2 (2007), 221–235.

[25] Santana, R. A Markov network based factorized distribution algorithm for optimization. *Machine Learning: ECML 2003*, 337–348.

[26] Shakya, S., McCall, J., and Brown, D. Using a Markov network model in a univariate EDA: an empirical cost-benefit analysis. In *Proceedings of the 2005 conference on Genetic and evolutionary computation* (2005), ACM, p. 734.

[27] Shakya, S., and Santana, R. A markovianity based optimisation algorithm. Tech. rep., Technical report, Department of Computer Science and Artificial Intelligence, University of the Basque Country, September.

[28] Vorobeychik, Y. A Game Theoretic Bidding Agent for the Ad Auction Game.

[29] Walter, I., and Gomide, F. Coevolutionary Genetic Fuzzy System to Assess Multiagent Bidding Strategies in Electricity Markets. In *Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference.* (2009), pp. 1114–1119.