Creating and monitoring a competitive and cost-effective pay-per-click advertisement campaign through the web-search channel is a resource demanding task in terms of expertise and effort. Assisting or even automating the work of an advertising specialist will have an unrivaled commercial value. In this paper we propose a methodology, an architecture, and a fully functional prototype for semi- and fully-automated creation, monitoring, and management of cost-efficient pay-per-click campaigns with budget constraints. The fully functional prototype is experimentally evaluated on real world Google AdWords campaigns and shows a promising behavior with regards to campaign success statistics outperforming systematically the competitive manual campaigns.

Categories and Subject Descriptors: H.4.m [Information Systems Applications]: Miscellaneous

General Terms: Algorithms, Experimentation, Measurement

Additional Key Words and Phrases: online advertising, pay-per-click advertising, automated campaign management, genetic algorithms, Google AdWords

1. INTRODUCTION

1.1. Online Advertising Campaigns

Online advertising is gaining acceptance and market share while it has evolved into a $26 billion industry for advertisers [iab]. One form of online advertising is the promotion of products and services through search-based advertising. This follows an ad auction process with pay-per-click (PPC) model [Douzet Douzet] for the advertisers. The selected search engine has the role of the auctioneer for the ad slots and the bids that the advertisers set for keywords and their ads. The dominant strategy for ad selection is the hybrid second-price auction [Edelman et al. 2007] system. The three most prevalent options in the search-based advertising market are Google AdWords, Yahoo Search Marketing, and Microsoft AdCenter (the two latter have merged) [yah]. Today’s most popular search-based advertising platform is Google AdWords having the largest share of revenues amongst its competitors. Google 1 2010 annual report showed that company’s advertising revenues made up 97% of its revenues in 2008 and 2009, and

1 Google Search is the most popular search engine with more than 300 million searches a daily basis. This ranks Google Search as the top search engine and also the top site in terms of web-traffic [ale]
96% of its revenues in 2010. In 2010, Google advertising revenues (as an auctioneer) were $29 billion \cite{go}. 

1.2. Contributions

This paper offers an in-depth analysis of advertising campaign management, monitoring, and budget optimization. We especially focus on budget optimization - a very challenging issue - presenting a methodology for selecting the most effective keywords and their bids, aiming at maximizing either the profits for the advertiser based on a specific budget or the traffic on their website. In this effort, we focus only on the advertisers and not on the other bidders or the self-interested auctioneer. We also assume that there are no public data on the competitors bidding behavior. Nevertheless, the auctioneer, which in our case is Google AdWords, provides us with two important variables: Global Monthly Searches and Competition for each campaign keyword. We use these parameters for observing and predicting the campaign behavior in favor of the advertiser.

Our main contributions are the following:

— An automated method for budget optimization based on a MCKP (multiple-choice knapsack problem) modeling and capitalizing on genetic algorithms to maximize profit or traffic, the two usual objectives for websites.
— A fully implemented and functional prototype system, developed for Google AdWords platform, which currently occupies a vast share of web-search advertising volume.
— A comprehensive experimental evaluation on real world data. Experimental results show that automated campaigns overall outperform the manual competitive ones. In this way, the proposed system contributes substantially in the campaign management process which is involving otherwise continuous monitoring, a task that is time consuming and exceptionally tedious for humans.

2. RELATED WORK

Given the ad auction mechanism of the search engine, the main question that the advertisers are facing is to decide their bidding strategy and how they are going to split their limited budget among several keywords of their campaign. There have been various attempts to solve the budget optimization problem, some of which are based on heuristics, some calculate approximations using linear programming variations, and others take a more statistical and stochastic approach. Even Dar et al. \cite{Even Dar et al. 2009} present their approach of maximizing profit, using a linear programming (LP) based polynomial-time algorithm. To deal with the NP-hardness of the problem, they propose a constant-factor approximation when the optimal profit significantly exceeds the cost. It is based on rounding a natural LP formulation of the problem. Szymanski and Lee \cite{Szymanski 2006} discuss how advertisers, by considering minimum return on investment (ROI), change their bidding and, consequently the auctioneer’s revenue in sponsored search advertisement auctions.

Borgs et al. \cite{Borgs et al. 2007} propose a bidding heuristic that is based on equalizing the marginal ROI across all keywords, so they change each keyword bid based on the ROI performance of the previous day. Their system converges to its market equilibrium in the case of the first price mechanism with a single slot when everybody adopts the proposed perturbed bid solution.

Rusmevichientong and Williamson \cite{Rusmevichientong and Williamson 2006} develop an adaptive algorithm that learns the proportions of clicks for different keywords by bidding on different prefix solutions, and eventually converges to near-optimal profits, assuming that various parameters are concentrated around their means. Muthukrishnan et al. \cite{Muthukrishnan et al. 2010} consider stochastic algorithms that attempt
to solve the problem in advance, and not by adaptive learning as in [Rusmevichientong and Williamson 2006], and work for pre-specified probability distributions of keyword clicks.

Zhou et al. [Zhou et al. 2008] model the problem of advertisers winning an ad slot for one keyword they bid upon as an online multiple-choice knapsack problem. Zhou and Naroditskiy [Zhou and Naroditskiy 2008] continue the work of [Zhou et al. 2008] modeling budget-constrained keyword bidding as a stochastic multiple-choice knapsack problem. Their algorithm selects keywords based on a threshold function which can be built and updated using historical data. It employs distributional information about prices and tries to solve the bidding problem with multiple ad-position, keywords, and time periods.

The problem of finding an near-optimal bidding strategy has been also approached by using autonomous agents. The TAC Ad Auctions (TAC/AA) game investigates complex strategic issues found in real sponsored search auctions through a simulation of the general auction process [Jordan and Wellman 2010].

3. BUDGET OPTIMIZATION PROBLEM

The most challenging issue in managing an advertising campaign is the Budget Optimization for the multiple keywords of the campaign. We consider the problem as follows: Assuming a limited budget $B$, we aim to find the combination of keywords with bids that maximizes the campaign profit. In particular, we are looking for a set of keywords $k \in K$ ($K$ is the set of all possible relevant keywords), and their bids $b \in \mathbb{R}_{\geq 0}$ with

\[ \sum_{k \in K} w_k(k, b) \leq B \quad (1) \]

where $w_k$ is the bidding cost $b$ on keyword $k$ (otherwise called weight) that produce:

\[ \max \sum_{k \in K} v_k(k, b) \quad (2) \]

where $v_k$ is the function that computes the expected profit of keyword $k$ (value) assuming of bidding price $b$. We also consider that for any given $k$ with $b = 0$, $b_k = 0 \Rightarrow w_k(k, b) = 0$ and $v_k(k, b) = 0$. A zero bid actually means that we choose not to bid on the particular keyword, so there is no cost or profit produced. In the following sections, we present our approach of finding the best combination of keywords and bids that produce maximum profit. Profit can be either monetary profit from product sales or generated traffic (clicks on ads) for the advertiser’s website. We define the above concepts as follows:

**Definition 1. (Weight and Cost)** The cost of a keyword $k$ for a given bid $b$ is the product of expected number of clicks and the average cost per click.

\[ w(k, b) = CPC(k, b) \ast Clicks(k, b) \quad (3) \]

In Definition 1 $Clicks(k, b) = CTR(k, b) \ast Impr(k, b)$, $CTR = \text{Click-through-rate}$, $Impr = \text{Impressions}$, $CPC = \text{Average-cost-per-click}$

**Definition 2. (Value for maximum monetary profit)** The profit from each keyword-bid combination comes from subtracting the cost of clicks, which is the cost of advertisement, from the revenue of sales.

\[ v(k, b) = Revenue(k) \ast CR(k, b) \ast Clicks(k, b) - w(k, b) \quad (4) \]
In Definition 2 $CR(k, b) * Clicks(k, b)$ is the total conversions (sales) that we expect to have and $Revenue(k) * CR(k, b) * Clicks(k, b)$ is the revenue expected for $(k, b)$. $CR = Conversion-rate$, $Revenue = Revenue-per-conversion$.

**Definition 3.** (Value for maximum traffic) **When we are interested in maximizing the traffic led to a website, the only valuable measure is the amount of clicks that are generated from keywords.**

$$v(k, b) = Clicks(k, b)$$

### 3.1. Multiple-choice knapsack problem formulation

In the online advertising campaign, the advertiser plays the role of an investor. The capital is the total budget $B$ for the period that the campaign is active. The profit from the conversions or clicks for each investment is represented as $v$. The cost that the advertiser is finally charged for a specific investment is $w$. Each investment is represented by a candidate item $x$ which is a pair $(k, b)$ where $k$ is the keyword and $b$ the bid that the advertiser initially sets as maximum $CPC$ for the specific keyword. The advertiser has $j$ options of $(k, b)$ candidate pairs for each investment, but he must select only one pair per investment for his final proposal, because for a particular keyword $k$ in the auction process, he can set only one bid. The total number $N$ of the final chosen investments must be equal to the $r$ available keywords of the campaign. This is a Multiple-Choice Knapsack Problem (MCKP). MCKP is a 0-1 knapsack problem in which a partition $N_1 \cdots N_r$ of the item set $N$ is given and it is required that exactly one item per subset is selected. Formally for our problem, the objective is to

$$\text{maximize } \sum_{i=1}^{r} \sum_{j \in N_i} v_{ij} x_{ij}$$

subject to

$$\sum_{i=1}^{r} \sum_{j \in N_i} w_{ij} x_{ij} \leq B$$

with $\sum_{j \in N_i} x_{ij} = 1$, for all $1 \leq i \leq r$

and $x_{ij} \in \{0, 1\}$, for all $1 \leq i \leq r$ and all $j \in N_i$.

The above imply that only one bid option is going to be selected for each keyword.

The optimal solution of the MCKP will indicate the best possible choice of keyword-bid options. Our approach was to model this combinatorial optimization problem in a certain way where we can also formulate it as a genetic algorithm (GA) process. In MCKP, the goal is to find for each keyword the option that maximizes the achieved profit. In GA, different chromosomes represent different instances of candidate items and the goal is to find the fittest chromosomes. As we will describe later, a GA finds approximately the proper options of MCKP for profit maximization. This process aims to collect proper statistics from previous time periods and keep only the most profitable options for the next time period. This problem formulation, as we can see in Figure 1, is different from the approach that we have seen in [Zhou and Naroditskiy 2008] and [Zhou et al. 2008], as in our method we focus on the clicks each keyword gains, rather than use MCKP to model the ad auction policy, where each advertiser can select to win at most one ad slot for each keyword.

3.2. Genetic Algorithm Advantages

Multiple-choice knapsack [Martello and Toth 1990] is a known NP-Complete problem, although some solutions for approximate optima in (pseudo-)polynomial time have been found. The approach we take is to capitalize on genetic algorithms [Djannaty and Doostdar Djannaty and Doostdar] that have also polynomial complexity and are used in a variety of global optimization [Mitchell 1996], [Weise 2008] problems. GAs find optima in certain search spaces and are able to combine exploration, the process of discovering possible solutions in search spaces, and exploitation, the process of using the knowledge of past solutions (past generations) to the benefit of a new more advanced solution. GA finds optimal solutions or near-optimum, since it is an approximation method, like any other polynomial time method that exists. Deterministic methods result in the same approximate solution in each run, thus making it difficult to collect data for many keywords. This is because this method will use those keywords that were chosen repeatedly in the past. On the other hand, the solution of a genetic algorithm may vary, resulting in a different near-optimum solution in each run. This trait is an advantage, as we do not want our method to have obsessions with certain solutions, thus choosing persistently certain keywords. This kind of flexibility, allows our system to discover faster whether keywords are performing better or worse than they did in the past. Therefore, as ad campaign parameters change, something frequent in the case of ad auctions, a deterministic method will adapt much slower than a genetic algorithm.

3.3. Bidding Strategy

The objective is to create a population of candidate solutions (called chromosomes). In each successive generation, a new population of chromosomes is produced by combining (with a procedure called crossover) pairs of chromosomes of the last generation to create new chromosomes (reproduction). The best chromosomes have a better chance to reproduce in the next generation (survival of the fittest), ensuring that each genera-
tion is improving. Selection is the process of finding the fittest chromosomes to become the parents of the next generation. For this purpose, there are fitness-proportionate techniques such as Weighted Roulette Wheel Selection (Weighted RWS) and Stochastic Universal Sampling (SUS). These methods make sure that, if a chromosome has a strong fitness, it will have proportionately high probability of reproducing. Moreover, we make sure that a (small) proportion of the fittest chromosomes pass directly to the next generation. This action is called elitism and its purpose is to prevent loosing the few best found solutions, increasing the performance of the genetic algorithm. The process of combining two chromosomes is called crossover. Every time, two offsprings are produced by two parents and the parents are replaced. The first offspring takes a part of each parent while the other obtains the remaining part of the parents. We want our genetic algorithm to avoid falling in local optima. Thus, the concept of mutation is applied on the chromosomes after the crossover process. Mutation changes the new offspring by altering, with a small probability, the value of their genes increasing the chance for reaching to the global optimum. The process of generating new populations terminates, usually, when ~ 90% of the chromosomes have the same fitness value or the highest ranking solution's fitness has reached a plateau, i.e. successive iterations no longer produce better results. Alternatively termination occurs in the case number of generations is greater than a certain limit.

3.3.1. Strategy Steps.
In this part, we present the main process of our methodology implementation. First, we describe the primary steps for initializing system parameters. We must define a default initial bid for all keywords that are going to be tested, so given a specific variable information from Google AdWords, we set $b_{initial} \leftarrow \max_{\text{EstimatedFirstPageBid}}$. Next, we define time for task periods (e.g. 2 days) and adgroups for each landing page along with their keywords and text ads. In Algorithm 1, we describe the general form of testing periods to test campaign and adgroup settings in order to collect proper statistics. The genetic algorithm step is the implementation of the optimization process. Finally, in each testing phase after optimization, we follow the same process of the first testing periods but we also pause previous keywords that are not selected by the optimization module.

The reason for different testing periods with only a small amount of testing keywords maintained is that, due to a limited daily budget for our experiments, we do not want to exhaust the budget without having tested many keyword options.

Optimization step
In the genetic algorithm, the bids for each keyword that are available to choose from are ones that have been tested and we have kept statistics on. It is not possible to have full information about the performance of a keyword that has not been tested at some point. Important performance criteria for our method are click-through rate, impressions, average cost-per-click and conversion-rate. Once we have collected the proper statistics, we are ready to apply our genetic algorithm for optimization.

Genetic Algorithm Representation

1. Start. Generate random population of $m$ ($m = 40$) chromosomes

Chromosome Representation
— For the budget optimization problem, each chromosome consists of N genes, N being the number of available keywords

Each gene has a value of the bid index that is selected for the specific keyword
\section*{Algorithm 1: Testing Period}

\textbf{Input:} Settings of Adgroups  \\
\textbf{Output:} Collected statistics  \\
Let $t$ be the number of task periods  \\
Let $S_G \subset N_G$, where $N_G$ are all the candidate keywords of AdGroup $G$  \\
$\triangleright$ Make a subset of $|S|$ keywords for testing for each AdGroup $G$  \\
$\triangleright$ $|G|$ is the total number of AdGroups  \\
$|S_G| \leftarrow |N_G|/t$  \\
forall the $g \in |G|$ do  \\
\hspace{1em} add(AdGroup[g].getMostRelevantKeywords($|S_G|$), keyword)  \\
end  \\
$|M| \leftarrow |G| \times |S|$  \\
forall the $\mu \in |M|$ do  \\
\hspace{1em} setBid($b_{initstat}$, keyword[$\mu$])  \\
\hspace{1em} activate(keyword[$\mu$])  \\
end  \\
if not firstPeriod then  \\
\hspace{1em} chooseRandom(keyword)  \\
forall the $\mu \in |M|$ do  \\
\hspace{1em} if choseed(keyword[$\mu$]) then  \\
\hspace{2em} bidNew[$\mu$] = bidPrevious[$\mu$] $\pm$ bidPrevious[$\mu$] $\times$ 50%  \\
\hspace{1em} end  \\
forall the $\mu \in |M|$ do  \\
$\triangleright$ Do not test again other keywords that received clicks  \\
\hspace{1em} if notChoosed(keyword[$\mu$]) and receivedClicks(keyword[$\mu$]) then  \\
\hspace{2em} pause(keyword[$\mu$])  \\
\hspace{1em} end  \\
while taskPeriod $> 0$ do  \\
forall the $\mu \in |M|$ do  \\
\hspace{1em} stat[$\mu$] = collect(impressions[$\mu$] $\cap$ clicks[$\mu$] $\cap$ conversions[$\mu$] $\cap$ averageCPC[$\mu$])  \\
\hspace{1em} add(stat[$\mu$], statistics)  \\
end  \\
taskPeriod $\leftarrow$ taskPeriod $- 1$  \\
end  \\
return statistics

— Table I shows a chromosome that has selected the second bid for keyword $k_1$ and zero bid (value 0) for keyword $k_2$  \\
— Table II shows that the second bid ($bidIndex = 2$) for $k_1$ is the actual bid value of $0.60$  \\
— Table III shows that this bid has a cost of $16.2$ and a positive profit of $1.40$. If a keyword is not selected ($bidIndex = 0$), like $k_2$ in Table I, it produces zero cost and profit  

\subsection*{2. Fitness}  

Fitness Function Evaluation  

— The fitness function is the expected total profit for the bids selected in the chromosome genes.

\begin{equation}
\text{Chromosome Fitness} = \sum v(k, b)
\end{equation}

Fitness function resembles the objective function of the knapsack problem. It can be easily computed since we have pre-computed all the costs and profits of the bids for every keyword, as shown in Table III  

Evaluate the fitness function of each chromosome in the population. Take into consideration actual or predicted values
— When a chromosome is generated, it has to pass the $\sum w(k, b) \leq B$ condition, otherwise randomly selected genes of the chromosome will be set to 0 until the condition is met.

3. New Population. Create a new population by repeating the following steps until the new population is complete:
   
   a. Selection. Select two parent chromosomes from a population according to their fitness (Weighted RWS). The best chromosomes are the ones with the highest values of the fitness function.
   
   b. Crossover. With a crossover probability, cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
   
   c. Mutation. With a mutation probability ($\sim 0.1\%$) mutate new offspring at each locus (position in chromosome).
   
   d. Accepting. Place new offspring in a new population.

4. Replace. Use new generated population for a further run of algorithm.

5. Test. End Condition

— Since we don’t know what the best answer is going to be, we just evolve the max number of times ($\text{MaxAllowedEvolutions} = 3000$).

If the end condition is satisfied, stop, and return the best solution in current population.

— Loop. Go to step 2.

After a period of testing and collecting statistics, budget optimization task is ready to run again. This process continues executing until the last day of our campaign.

3.3.2. Impressions Prediction.

We wanted also to examine if a certain type of campaign behavior prediction could be helpful to our system. Clicks, click-through rate, and conversion rate are parameters that are more dependent to inner factors of the advertiser’s choices such as the quality and relevance of the selected keywords and ad-texts for the product promotion. However, this is not exactly the case for the impressions that a user query generates. The impressions fluctuate primarily and because of other factors external to the keyword-bid combination. Consequently, we need a means to predict or at least make a good estimation of how many impressions a keyword will receive matched with a specific bid, knowing:

1. Past received clicks from a specific selected combination $(k, b)$
2. Current average user searches for a query similar to this keyword
3. Current will of competition of all the other bidders upon this specific keyword

The idea is to use past results of keyword behavior in a model that can capture externalities of the ad auctions and predict current or future behavior. Google AdWords provides information such as Global Monthly Searches (GMS) and Competition of a keyword which are factors that affect the number of impressions of a keyword and, at the same time, are independent of a particular AdWords Account.

Past data of all keywords with known Impressions has the following form:

\[
[\text{Clicks}(k_1, b_1), \text{GMS}(k_1), \text{Competition}(k_1)] \rightarrow \text{Impressions}(k_1, b_1)
\]

\[
[\text{Clicks}(k_2, b_2), \text{GMS}(k_2), \text{Competition}(k_2)] \rightarrow \text{Impressions}(k_2, b_2)
\]

\[
[\text{Clicks}(k_n, b_n), \text{GMS}(k_n), \text{Competition}(k_n)] \rightarrow \text{Impressions}(k_n, b_n)
\]

thus, we aim to predict the impressions of another keyword – bid $(k_i, b_i)$:

\[
[\text{Clicks}(k_i, b_i), \text{GMS}(k_i), \text{Competition}(k_i)] \rightarrow \text{Impressions}(k_i, b_i)
\]
Table I. Example of chromosome representation and the values of its genes

<table>
<thead>
<tr>
<th>bidIndex</th>
<th>k_1</th>
<th>k_2</th>
<th>k_3</th>
<th>...</th>
<th>k_N</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>3</td>
<td></td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>

| valueRange | [0-4] | [0-3] | [0-3] | [0-2] |

Table II. Bid matrix example. For \( k_1 \), we have a value range of [0-4] of the bidIndex

<table>
<thead>
<tr>
<th>bidIndex</th>
<th>( k_1 )</th>
<th>( k_2 )</th>
<th>( k_3 )</th>
<th>...</th>
<th>( k_N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$0.50</td>
<td>$0.90</td>
<td>$0.45</td>
<td>...</td>
<td>$0.55</td>
</tr>
<tr>
<td>2</td>
<td>$0.60</td>
<td>$1.10</td>
<td>$0.55</td>
<td></td>
<td>$0.70</td>
</tr>
<tr>
<td>3</td>
<td>$0.70</td>
<td>$1.30</td>
<td>$0.65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$0.80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table III. Example of expected costs and profits for each different \((k, b)\) pair

<table>
<thead>
<tr>
<th>( k_1 )</th>
<th>...</th>
<th>( k_N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>bidIndex</td>
<td>( w(k,b) )</td>
<td>( v(k,b) )</td>
</tr>
<tr>
<td>1</td>
<td>$14.5</td>
<td>$1.5</td>
</tr>
<tr>
<td>2</td>
<td>$16.2</td>
<td>$1.4</td>
</tr>
<tr>
<td>3</td>
<td>$18.1</td>
<td>$0.3</td>
</tr>
<tr>
<td>4</td>
<td>$19.8</td>
<td>$0.5</td>
</tr>
</tbody>
</table>

After prediction of impressions of all keyword-bid combinations is carried out, new values for clicks and conversions can be computed. A good estimation of impressions may result in a good cost and profit estimation, and can possibly lead to an improved budget optimization. To perform impressions prediction, we choose multiple linear regression [Montgomery et al. 2007]. This method assumes the existence of linear correlation between the dependent variable \( y \) (Impressions) and the independent variables (in our case \( x_1 = \text{Clicks}, x_2 = \text{GMS}, x_3 = \text{Competition} \)). So, we need to find the best coefficients that show the relationship between \( y \) and \( x_i \). The goal is to be able to calculate a new value of \( y \) out of the independent variables and the coefficients.

\[ y' = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_k x_k \] (9)

In our case, we have 3 independent variables:

\[ y' = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \] (10)

The regression model is fitted with the least squares [Lawson and Hanson 1995] approach. The sum of square residuals is considered to be the error \( (e) \) when comparing the \( y \) with \( y' \):

\[ e = \sum_{i=1}^{N} (y_i - y'_i)^2 \] (11)

\( N \) is the amount of all available records. In the end, the chosen coefficients \( \theta_i \) must minimize the error produced by prediction to have the best fit of our model.

The result from this process was an alternative input to the genetic algorithm with different calculated statistics in order to study if using impressions prediction would achieve better campaign performance.
4. AUTOMATED MONITORING OF ONLINE ADVERTISING CAMPAIGNS

4.1. Architecture

The AD-MAD System that we have developed is separated in two parts:

(1) The Keywords part (GrammAds Component), which is responsible for retrieving the most relevant keywords and generating ad creatives based on information taken from the landing pages. (described in [Thomaidou and Vazirgiannis 2011])

(2) The Campaign part (Adomaton Component), which is responsible for initializing, monitoring, and managing the advertising campaigns in the course of time, based on keyword statistics maintained by the system, in the view of optimizing available budget. (described in this paper)

In Figure 2, we present the two parts with the associations of their components. The Adomaton Component, that we describe in this paper, is responsible not only for the proper monitoring and optimization of the campaign but also for the overall integration of the system. Once the right keywords are selected for each product and the ad texts are written, everything is ready to create and manage the advertising campaign, with the ultimate goal of increasing the web traffic or sales of the advertiser’s website, according to the assigned preferences that the system user has given. In the next section, we will see the major tasks that the system must perform to reach the desired goal.

4.2. Software Description

The software that has been developed serves three basic purposes:

— To showcase that the proposed architecture and methodology is plausible and feasible and provide proof of concept.

— To become a good basis for a larger system that has more features and can provide more functionality useful for the field of web-search advertising.

— To find and implement tangible solutions to the issues arising at the very core of a Campaign creation and Budget optimization software.
The code of this software was completely written in Java using JDK SE 6 on the Eclipse IDE. For the database engine, MySQL 5.5 Community Server was used. We used AdWords API Java Library to communicate with the programming interface of AdWords, JGAP Library which is an open source framework for implementing and configuring genetic algorithms and genetic programming, and Flanagan's Java Scientific Library to implement multiple linear regression.

4.2.1. Campaign Creation.

AdWords API Wrapper

The Google AdWords API is fairly tricky. Even for trivial tasks, such as creating and setting a keyword, it requires a series of functions to be called, making it a long procedure to code. For the sake of programming sanity, many tasks must be wrapped to simpler functions, saving effort in typing, code readability and making the software less error prone. Furthermore, a lot of attention must be paid to the fact that using the AdWords API is not for free. In particular, Google charges API units per thousand and there is a rate-sheet [rat] of every API-call. So, it makes sense to build a wrapper that balances usability with low-costliness.

Create the campaigns

The campaigns must be organized into adgroups which in turn contain keywords and ad creatives. Through the graphical user interface (GUI), it is requested from the advertiser to give, as input, the landing pages of the website that they want to be advertised. Using the extraction tools from the Keyword part of the system, those landing pages are analyzed and information is extracted, such as product information, languages, and type of advertisement needed. This information is verified and corrected by the advertiser if the system works in semi-automatic mode, otherwise the system uses this information as is provided, working fully-automatically. Then, campaigns are created that contain adgroups and the rule of thumb is that for each landing page there must be a separate adgroup that contains its keywords and ad creatives. At the creation stage, all keywords work in trial mode, which means that no statistical data exist for them yet and the system waits to see how they will perform. After the first statistics are collected, the system can then use budget optimization.

4.2.2. Statistics and Database.

Task Scheduler

Since ad campaigns run for weeks, months, or even years, a scheduler is important for the system in order to ensure a task start and end. If a scheduler does not exist, the system cannot store proper statistics and allocate resources properly. The system also has to respect the budget constraint for a certain time period. For each new period, a new budget must be allocated and new optimal keywords must be selected.

Statistical Information

A SQL database must store all the statistical information collected for the keywords to be able to track their performance. All information about keywords, such as impressions, clicks, click-through rate, conversion rate, average position, average cost-per-click are necessary for the budget optimization module function. We also store temporal information, in the sense of distinguishing time periods. Also, there is the need for some data access objects (DAOs) that make retrieval and insertion of data to the database simpler. Both the Genetic Algorithm and the Prediction modules make extensive use of the statistical information of the keywords.

4.2.3. Budget Optimization.

Genetic Algorithm

The statistics module must provide a list of all possible keywords. Each keyword of the list must have all the possible bids for which we have statistics. For each keyword-
bid combination \((k, b)\), an evaluation of the cost and profit must be computed based on the keyword performance information kept in the database. If a bid gives negative expected profit, it must be erased from the data because it cannot contribute to a maximum profit solution. Optionally, instead of directly using the value of keyword impressions to compute cost and profit, prediction can be used. The predicted impressions will have an effect on expected clicks that in turn affect cost and profit. The problem is then modeled into chromosomes and the fittest chromosome is finally selected by the genetic algorithm after several generations of breeding using the methods of crossover, mutation, and elitism. A final list of keyword-bid pairs \((k, b)\) is produced from the fittest chromosome. These keyword-bid pairs will form the new bidding strategy, which must be set in the AdWords account of the advertiser using the AdWords API. Evolving the population the max number of times (we set the number of maximum allowed evolutions to 3000) and setting the initial population size of chromosomes to 40, help us avoid premature convergence, which was considered as the main problem of GA theory.

**Prediction**

This is the module that is responsible for performing impressions prediction using the past statistics of keywords and targeted information taken from AdWords, such as the Global Monthly Searches and the level of Competition that exists for a given keyword. Before budget optimization, optionally, we can use this module in order to refine the statistics used by applying impressions prediction. A regression model finds the relationship between the impressions and other variables, such as clicks, global monthly searches, and competition. It then re-computes the value of impressions for each keyword-bid combination using the independent variables. The values of Global Monthly Searches and Competition must be computed from the AdWords API.

5. **SYSTEM EVALUATION AND EXPERIMENTS**

We present here several experiments and data analysis that we have conducted in order to study the performance of our proposed methodology and system. In the first three experiments, we evaluate only the optimization step, while in the fourth experiment we test the campaign creation, management, and optimization modules as an integrated process to evaluate the system as a whole.

5.1. **Evaluation Data**

We use the historical data of a large scale AdWords Campaign of a web site in the area of car rental. We selected from the collected data all the campaigns and adgroups that promote “car rental in Crete”. The data collected are relevant to derive the period May 2009 to November 2010, during which the campaign was very active the majority of the time, generating traffic and sales for the car rental website. With the retrieved data of Google AdWords keyword statistics and sales for the car renting business, we get sufficient data for 39 weeks of this large scale campaign to perform tests on the impressions prediction and budget optimization modules. The final form of the integrated statistics table contains the following features: \{Campaign, Adgroup, Week, Keyword, MaxCPC, Impressions, Clicks, Conversions, CTR, Avg CPC, Cost, Profit, Quality Score, FirstPageCPC, Avg Position, Avg CPM\}. For the impressions prediction module of the system, we need to have for every keyword the “Global Monthly Searches” and the “Competition” values, retrieved using the AdWords API. Our budget optimization system provides two options; to optimize budget for maximum traffic or for maximum profit. Additionally, we can use original or predicted impressions.

Our budget optimization system provides two options; to optimize budget for maximum traffic or for maximum profit. Additionally, we can use original or predicted impressions. These options give us four basic testing scenarios:
5.2. Experiments

5.2.1. Scenario Comparison. The data used for the first experiment on the budget optimization process are the keyword statistics we collected from the car rental website for 39 weeks and the budget to be allocated for the next (hypothetical) week. Since budget optimization is performed with a genetic algorithm and the stochastic method - the result will slightly vary every time it is executed, even with the same input data. So, each scenario (NoPredProfit, NoPredTraffic, PredProfit, PredTraffic) is executed 30 times and the result reported is the average value of 30 executions.

The result of every execution of the budget optimization module is an optimal keyword-bid combination that ensures either maximum traffic or maximum profit for a limited budget. In particular, every result of the genetic algorithm application produces the following data:

— **Clicks**: How many clicks is the optimal solution (keyword - bid combination) expected to produce in the following week? This is an estimation, so it is represented with a double instead of an integer value.
— **Cost**: How much is it expected to cost in the following week? This value must always be lower or equal to the budget.
— **Profit**: How much profit are we expected to make in the following week? The profit is calculated after excluding the advertisement cost, meaning: \(\text{Revenue} = \text{Cost} + \text{Profit}\).
— **#Keywords Used**: This value counts the number of keywords which were selected in the optimal solution.
— **Average Bid**: The average value of the bid (or MaxCPC) of every selected keyword of the optimal solution.

The above output is the average result solution of the applied budget optimization for a future 40th week of the advertising campaign. This experiment is using a simulation and we make here the assumption that the metrics are computed as if CTR, clicks, costs, and impressions were maintained the same for each \((k, b)\) choice in the future. We first run budget optimization for different values of the available budget. In Table IV, we present the average results of 30 executions for the four scenarios with budgets of 50, 100, 200, 400, and 600 units (euros).

We notice the following on all tests: The methods that were using prediction outperform the simple GA ones. Optimization for profit always produces more profit than optimization for traffic, as expected. Optimization for traffic always produces more clicks than optimization for profit, as expected. We notice that the Average Bid increases along with the available budget. This is because the cheaper (cost-efficient) keywords are running out, so we have to use more costly ones. All solutions deplete their budget unless there are no more keywords left or the keywords left are not profitable. In the case of budget=600, when optimizing for traffic without prediction, we reach the limit of how many clicks can be made, and therefore our solution produces the maximum cost (405.16), which is less than the budget (600). This solution also uses all available keyword options (107 in size). In the case of budget=400 and budget=600, when optimizing for profit without prediction, we reach an upper limit of the profit, so the budget is not depleted. Not all keywords are used in this case because not all keywords are profitable. In the cases of small budgets, we notice that optimizing for profit generates...
Table IV. Budget optimization evaluation results

<table>
<thead>
<tr>
<th>Budget = 50</th>
<th>Clicks</th>
<th>Cost</th>
<th>Profit</th>
<th>#Keywords Used</th>
<th>AverageBid</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoPredProfit</td>
<td>60</td>
<td>49.94</td>
<td>219.51</td>
<td>24</td>
<td>1.49</td>
</tr>
<tr>
<td>NoPredTraffic</td>
<td>61</td>
<td>49.93</td>
<td>206.22</td>
<td>23</td>
<td>1.43</td>
</tr>
<tr>
<td>PredProfit</td>
<td>82.36</td>
<td>49.90</td>
<td>317.1</td>
<td>18</td>
<td>1.37</td>
</tr>
<tr>
<td>PredTraffic</td>
<td>86.51</td>
<td>49.88</td>
<td>274.81</td>
<td>18</td>
<td>1.42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Budget = 100</th>
<th>Clicks</th>
<th>Cost</th>
<th>Profit</th>
<th>#Keywords Used</th>
<th>AverageBid</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoPredProfit</td>
<td>108</td>
<td>99.93</td>
<td>374.98</td>
<td>25</td>
<td>1.48</td>
</tr>
<tr>
<td>NoPredTraffic</td>
<td>109</td>
<td>99.92</td>
<td>356.44</td>
<td>26</td>
<td>1.44</td>
</tr>
<tr>
<td>PredProfit</td>
<td>130.80</td>
<td>99.87</td>
<td>467.86</td>
<td>20</td>
<td>1.41</td>
</tr>
<tr>
<td>PredTraffic</td>
<td>134.21</td>
<td>99.92</td>
<td>364.53</td>
<td>19</td>
<td>1.46</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Budget = 200</th>
<th>Clicks</th>
<th>Cost</th>
<th>Profit</th>
<th>#Keywords Used</th>
<th>AverageBid</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoPredProfit</td>
<td>197</td>
<td>199.87</td>
<td>621.32</td>
<td>56</td>
<td>1.55</td>
</tr>
<tr>
<td>NoPredTraffic</td>
<td>200</td>
<td>199.90</td>
<td>582.21</td>
<td>54</td>
<td>1.50</td>
</tr>
<tr>
<td>PredProfit</td>
<td>236.94</td>
<td>199.86</td>
<td>787.63</td>
<td>31</td>
<td>1.42</td>
</tr>
<tr>
<td>PredTraffic</td>
<td>248.60</td>
<td>199.85</td>
<td>638.13</td>
<td>32</td>
<td>1.43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Budget = 400</th>
<th>Clicks</th>
<th>Cost</th>
<th>Profit</th>
<th>#Keywords Used</th>
<th>AverageBid</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoPredProfit</td>
<td>333</td>
<td>389.61</td>
<td>798.90</td>
<td>98</td>
<td>1.61</td>
</tr>
<tr>
<td>NoPredTraffic</td>
<td>340</td>
<td>399.92</td>
<td>791.93</td>
<td>102</td>
<td>1.63</td>
</tr>
<tr>
<td>PredProfit</td>
<td>425.74</td>
<td>399.82</td>
<td>1313.99</td>
<td>54</td>
<td>1.51</td>
</tr>
<tr>
<td>PredTraffic</td>
<td>441.42</td>
<td>399.90</td>
<td>1191.51</td>
<td>45</td>
<td>1.45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Budget = 600</th>
<th>Clicks</th>
<th>Cost</th>
<th>Profit</th>
<th>#Keywords Used</th>
<th>AverageBid</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoPredProfit</td>
<td>393</td>
<td>599.80</td>
<td>798.90</td>
<td>97</td>
<td>1.61</td>
</tr>
<tr>
<td>NoPredTraffic</td>
<td>343</td>
<td>405.16</td>
<td>795.28</td>
<td>107</td>
<td>1.85</td>
</tr>
<tr>
<td>PredProfit</td>
<td>607.74</td>
<td>599.84</td>
<td>1645.60</td>
<td>70</td>
<td>1.56</td>
</tr>
<tr>
<td>PredTraffic</td>
<td>622.69</td>
<td>599.82</td>
<td>1569.21</td>
<td>68</td>
<td>1.52</td>
</tr>
</tbody>
</table>

Fig. 3. Best solutions

almost as much traffic as optimizing for traffic. This could mean that keywords that generate more profit are more relevant, hence they are clicked more often.

5.2.2. Genetic algorithm performance on finding the best solutions for MCKP.

In the second experiment, we apply the genetic algorithm to evaluate the hypothesis of choosing the optimal keyword-bid combination of each week. The input weekly budget for our scenarios is the corresponding actual weekly cost of the campaign. For each week, the input keyword options for the genetic algorithm are the actual tested keywords and bids for the specific week. Each scenario output is the average result of five executions of the genetic algorithm. In Figure 3 we present the results for total traffic and profit comparison, where we notice that our method finds in total the most profitable keywords for both traffic and profit maximization cases.
5.2.3. Genetic algorithm performance on optimizing next week’s performance.

In the third experiment, we test the expected weekly performance of each of our methodology scenarios towards the actual campaign weekly performance. For estimating performance of week \( i \), the genetic algorithm takes into consideration the statistics from weeks 1 to \( i-1 \), resulting in a “leave-one-out” cross-validation-like process. The training set is the actual statistic set from week 1 to \( i-1 \) and the testing set is the actual statistic set of week \( i \). For example, the input features for the optimal keywords and bids of the 20th week are the collected statistics from weeks 1 to 19. The purpose of this evaluation is to find solutions that achieve higher weekly performance than the actual one. Each scenario output from the budget optimization process is the average result of 10 executions of the genetic algorithm. The input weekly budget for our scenarios is a bit higher (1-2 euros) than the corresponding actual weekly cost of the campaign, assuming without having the actual information, that on average the budget is not completely depleted. As we present in Figure 4, in the case of traffic maximization as the advertising goal, our two methods which use prediction, surpass the real results. In this experiment, the optimization process had started after the 4th week, because the advertiser until the 3rd week had been testing very few keyword options (3-4) and the GA needs more testing data to perform a valid optimization. The important observation here compared with the stronger performance of the previous experiment was the use of much older and thus outdated data that did not correspond to valid receiving impressions and clicks in the \( i \)th week. The Impressions Prediction module had a major contributed role in the calculation of more up-to-date data because it achieved to capture current external factors and conditions of the ad auction. Thus, the methods that were using prediction outperformed the other ones.

5.2.4. Comparison of parallel competing campaigns (Automated compared to Manual).

In the fourth experiment, we create Google AdWords campaigns for two companies; Client1 is a company that offers web developing solutions (a highly competitive field for online advertising) and Client2 is a company that offers aluminum railing and fencing products. For each company we create one manual and one automated campaign. Each automated campaign is created semi-automatically by our system (the only intervention is the parameter input of daily budget, account credentials, period of active campaign, and keywords). We set our automated campaigns for traffic maximization as the advertising goal. We use for each manual and automated campaign the same keywords and the same budget in order to test only the monitoring and optimization process. In this experiment, we do not use impressions prediction, only the real values case scenario (due to limited budget for further experiments at that time). In Figures

![Weekly Traffic Comparison](image1.png)

![Total Clicks Comparison](image2.png)

Fig. 4. Optimization for next week
5 and 6, we present the final results after a period of 17 days. In the case of Client2, the automated campaign achieved higher performance in total traffic than the manual one. In the case of Client1, the automated achieved a slightly lower performance than the manual one. In both cases, the automated campaigns achieved better placement in the advertising slots than the manual ones, as well as lower prices for average cost-per-click.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed an architecture and a prototype system for automated advertising campaign development, monitoring, and optimization, putting emphasis on the campaign creation, management, and budget optimization modules. We articulated the optimization problem as a multiple-choice knapsack for which we find the most profitable combination of keywords and their bids. We approximated the solution capitalizing on a genetic algorithm for budget optimization with multiple keyword options. We also proposed the use of keyword statistics to predict keyword behavior using multiple linear regression. Both a. the use of a genetic algorithm and b. impressions prediction for this type of problem form innovative solutions with respect to existing literature. Proof of concept was given with the implementation of the proposed architecture. The implemented system was extensively evaluated with data from a real
website providing insight on how budget optimization works with and without keyword impressions prediction. We have also tested the integrated system for two real world web sites and compared our automated campaigns towards the manually created ones. The budget optimization problem, even though it is an NP-hard problem, has been practically solved by modeling it as a multiple-choice knapsack problem. In future work we plan to:

— Test more methods and apply, for example, Expectation Maximization [Zacharouli et al. 2009] techniques to cluster the keyword data
— Additionally, Hidden Markov Model [Bilmes 1998] could be tested to see if there are any transitions in keyword state that could be predicted
— Experiment further with real world campaigns

Currently, we are already working towards prediction of clicks with some preliminary but promising results.

REFERENCES

Douvzet, A. An Investigation of Pay Per Click Search Engine Advertising: Modeling the PPC Paradigm to lower Cost per Action. cdn.theladders.net, 1–7.


