Les Cahiers du GERAD ISSN: 0711–2440

Optimizing keyword positions for search engine marketing

K. Azeuli, M. Gamache,
A. Hertz, S. Paroz

G–2017–26

April 2017

Optimizing keyword positions for search engine marketing

Koukla Azeuli [a,b]
Michel Gamache [a,b]
Alain Hertz [a,b]
Sandrine Paroz [c]

[a] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7*

[b] *GERAD, HEC Montréal, Montréal (Québec), Canada, H3T 2A7*

[c] *Acquisio, Brossard (Québec) Canada, J4Z 3P2*

koukla.azeuli@polymtl.ca
michel.gamache@polymtl.ca
alain.hertz@polymtl.ca
sparoz@acquisio.com

April 2017

Les Cahiers du GERAD
G–2017–26

Abstract: Most papers on digital advertising focus on the point of view of Internet companies such as Google and Microsoft, and were written by people working for those companies. Even though that might help understand better how such companies manage the ad auctions, advisers do not have access to the required data and cannot use those works directly to improve the effectiveness of their digital campaigns. In this paper, we describe a model for determining which keyword positions to aim for on search engines using users' navigation histories accessible to advertisers. The navigation history of users who interacted with an element of a campaign is represented by a flow in a graph. We use formulas that link changes in keyword positions with the number of clicks on the ads which appearance was triggered by those keywords, and show how these changes modify the flow. Taking into account budget constraints, we then describe algorithms that reorganize the flow by changing keyword positions in order to maximize the expected profit from conversions tracked online.

Keywords: Digital advertising, search engine marketing, keyword positioning, network flow, heuristic algorithms

1 Introduction

According to eMarketer [7], over $180 billions were spent in 2015 in media advertising in the United States alone, and nearly a third of that amount was spent on digital ads. There are several digital advertising channels and the search medium accounts for over 40% of the digital ad spending in the US, that represents over $25 billions as stated by eMarketer [8]. That expense is the result of ad placement auctions. On search engines, after a user types a request in the search field, the auction takes place during the loading of the page. Each search engine has a set of criteria to rank the applying ads. According to AdWords [1, 2] and Bing Ads [4], the most common criteria are the maximum cost-per-click (i.e., the maximal amount the advertiser is ready to pay for each click on his ad), the ad relevance, and its past performance. The higher an ad is ranked (i.e., the closest it is to position 1), the more people click on it, and the more expensive each click will be. Quinn [21] developed two formulas: the first links an ad position and the average cost-per-click billed to the advertiser; the second links the ad position and the number of clicks on the said ad. In this work, we introduce a model that uses those formulas to determine the best keyword positions to maximize the expected profit under budget constraints. We solve this problem using graph theory, and more specifically network flow models. We thus offer a tool to help advertisers identify which elements of their campaigns contributed the most to their goals using navigation histories of users who interacted with at least one element of their campaigns. Those elements are:

- Search queries: texts typed by users in the search field of the search engine;
- Keyword/ad pairs: any given keyword can trigger the impression of multiple ads and the impression of any given ad can be triggered by more than one keyword. In this work, we aggregate a keyword to each ad which impression it triggered in a pair. For simplification, in the rest of the article, the term keyword will refer to such a pair;
- Banners: display ads published on a third party website i.e. that is neither the advertiser's website nor a search engine;
- Conversions: a conversion happens when someone clicks on an ad and then takes an action that the advertiser has defined as valuable to his business, such as an online purchase or a call to his business from a mobile phone;
- Pages of the advertiser's website: only those that are tagged.

The effectiveness of a marketing campaign depends on many factors. In this paper, we focus on the keyword positions with the aim of maximizing profitability under budget constraints. We propose a model and algorithms that determine which position to assign to each keyword in a search campaign of an advertiser.

In order to establish links between the elements of the campaigns of a given advertiser, the users' navigation histories are represented by a flow in a graph where each vertex represents one element, and an arc links two elements if and only if the first preceded the second in at least one of the users' histories. The model takes into account the change of the average costs-per-click as well as changes in users' behavior related to the changes of keyword positions. To evaluate the consequences of those changes on the rest of the campaigns, once the position is set for each keyword, the expected flow changes on the vertices representing the keywords are propagated through the network.

The paper is organized as follows. The next section contains a literature review of a few works in the field. We then indicate the data used to build the considered graph and describe the proposed model and the considered constraints. An example is given to illustrate the model. Five algorithms are then presented, and experimental results are reported and analyzed. We conclude with a final discussion and remarks.

2 Related works

When it comes to research on online advertising, several subjects come to mind. Here are a few works done on some of them.

Bid optimization. The goal here is to determine how much to bid on keywords in order to have an ad displayed at the most profitable position. Borgs et al. [5] developed a method to find the bids for each advertiser and each keyword that maximize the utility of the keywords. But this is done for multiple advertisers competing for the same ad placements. Advertisers cannot use this method since they do not have access to the data of their competitors. Even Dar et al. [9] presented a work on bid optimization with a focus on the particular case of "broad match" keywords. However, they implicitly assumed that the position of one ad does not impact the expected number of clicks on it as the position is not modeled in their problem. Zhou et al. [24] formulated the bid optimization problem as a knapsack problem where the advertiser must choose one placement to aim for at a given time in order to maximize his revenues or profits while being oblivious of the other advertisers' bids. They do not consider that there might be a relation between the auctions whereas in this work the users' histories are taken into account to simulate the potential impact of each decision on the rest of the campaigns. Zhang et al. [23] proposed a model to solve both the bid optimization and the budget optimization problems to maximize the advertisers' revenue.

Budget optimization In this case, advertisers want to maximize the number of clicks on the ads while respecting the campaign budget. Archak et al. [3] addressed the budget optimization problem using a Markov decision process. They assumed that the probability that a user goes from one state to another is fixed and constant whereas in our work, we assume that this probability depends on the position of the ad. DasGupta and Muthukrishnan [6] addressed the budget optimization problem with stochastic optimization methods. They defined a scenario with an expected number of clicks on the ads triggered by each keyword and evaluated the probability that a scenario comes true. Their goal was to find the set of positions of the ads that maximizes the sum of the expected number of clicks of all the scenarios. We apply a similar logic except that we work with only one scenario. Feldman et al. [10] addressed the budget optimization problem by defining the bids as the variables of their model whereas, in the present article, they are keyword positions.

Conversion attribution Assigning weights to the elements of a campaign that represent how much each of them contributed to the conversions is the purpose of conversion attribution. Jayawardane et al. [15] made a literature review on conversion attribution and they gave a set of criteria for an algorithm to be easily adopted in the industry. Jerath et al. [16] developed a model to determine how much to pay each publisher in a context where they are paid per conversion.

Click prediction Click prediction, as its name entails, is about predicting the number of clicks an ad will receive. McMahan et al. [18], Graepel et al. [14] presented learning algorithms to predict the click-through-rate (CTR) which is the probability that a user clicks on an ad. Rutz et al. [22] proposed a model to predict the CTR using ad positions and keywords characteristics as variables.

User behavior analysis The idea of using graph theory to represent users interactions with online marketing campaigns has been explored more than once. Nasraoui and Krishnapuram [19], Nasraoui et al. [20] extracted users' profiles out of their navigation histories, while Li et al. [17] presented an algorithm to find the most frequent elements in users' histories.

3 History graph

The history graph gathers the users' navigation histories called users' paths. We create one vertex for every element of the campaign (i.e., for every keyword, banner, conversion, search query, and page of the advertiser's website), and an arc links two vertices if and only if the first one is immediately succeeded by the second one in at least one of the users' paths. A source s, a sink t and two other vertices c and l are added to the graph: vertex c, is the only successor to every vertex representing a conversion, and vertex l, is a successor to every last vertex of a user's path, with the exception of conversions. The sink t is the one and only successor to c and l, and those two vertices are the only predecessors of t. An example is given in Figure 1.

The main purpose of the graph is to represent all the paths the users have taken within a fixed period of time, and also all the possible paths deducted from the previous. This allows to evaluate the repercussions of any change of position of a keyword on other elements in the graph. More specifically, a change of position will trigger a change in the number of clicks on the corresponding keyword and its successors, and also a change in its average costs-per-click. Both those changes will either lead to freeing up some or consuming more budget. A change of traffic will be propagated through the arcs all the way to the sink and whenever it changes the number of clicks on a banner that also impacts the available budget.

The flow at the source vertex is equal to the number of paths. Any given path can contain more than one occurrence of a vertex other than a conversion. That is why the flow on any vertex representing an element of the campaign does not necessarily equals the number of paths containing that element, but rather the total number of occurrences of that element in all the paths or the number of times the users clicked on that element.

The following statistical and monetary information about the advertiser's campaigns, and the following elements of the users' histories of their interactions with the advertiser's campaign are used to build the graph:

- the daily number of clicks;
- the daily average position of each keyword;
- the daily average cost-per-click of each keyword and banner;
- the set of users that interacted with the advertiser's campaign;
- the date and time of each interaction;
- the type of each interaction: it can be a simple click (that is a click on a page of the client's website containing a tracker), an organic click (the user clicked on a result generated by his request that is not an ad), a paid click (the user clicked on an ad that was among the results generated by his request), or a conversion;
- a set of search queries: if a user clicks on an organic result, his query is represented in the graph by a vertex;
- a set of keywords: if a user clicks on a text ad, the keyword that triggered the display of the ad and the ad itself are paired in a single vertex;
- a set of referrers: a referrer is the URL of the previous webpage from which a link was followed;
- a set of banners: if a user clicks on a banner on a third-party website, the banner is a vertex of the graph.

4 Modeling the problem

The flow in the graph is reorganized by changing the positions of the keywords represented in it. To serve that purpose, the following assumptions were made:

A1 Additional flow cannot be created: the flow at the source must remain constant. This assumption was made because it is not possible to predict where all the new flow will be coming from. This work is thus limited to reorganizing the existing flow. While this hypothesis may seem very restrictive, it is possible to regularly update the graph and the flow that circulates in it when new users' navigation histories are made available (typically once per day, week, or month).

A2 If the position of one keyword is improved, it will generate more clicks on itself by withdrawing them from the other options the users have: this is a result of assumption A1. A better position for a keyword leads to more clicks on it but since new flow cannot be generated, the extra clicks must be withdrawn from the other results. In this work, it is considered that the other advertisers are represented through the loss sink. So, the extra flow will be taken from all the other vertices that share the same parent and the loss sink is considered to always be one of those vertices.

A3 If the position of a keyword is deteriorated, it will lose some of its clicks to the other options: this is also a result of assumption A1. A worse position for a keyword leads to less clicks on it. Those clicks are lost to the other options, and the extra flow will be distributed to the other vertices that share the same parent.

4.1 Notations

In what follows, we use the following notations.

Sets	
$\mathcal{A}$	set of arcs
$\mathcal{V}$	set of vertices
$\mathcal{B}$	set of vertices representing banners
$\mathcal{C}$	set of vertices representing conversions
$\mathcal{K}$	set of vertices representing keywords
$\mathcal{Q}$	set of vertices representing search queries
$\mathcal{W}$	set of vertices representing pages of the client's website
Γ_v^+	set of vertices immediately succeeding vertex v
Γ_v^-	set of vertices immediately preceding vertex v
$\mathcal{P}$	set $\{1,\ldots,\bar{p}\}$ of positions that can be assigned to a keyword. Position 1 attracts the largest number of user clicks. A keyword at position $\bar{p}$ is considered as not being displayed since the variation in the number of conversions or user clicks is very small beyond that position. In what follows, we have fixed $\bar{p} = 11$.
Parameters	
β_{click}	adjustment coefficient for the number of clicks when a keyword is displayed one position higher
β_{cpc}	adjustment coefficient for the average cost-per-click when a keyword is displayed one position higher
B	budget of the campaign
B_{da}	budget for the display advertising campaign ($B_{da} < B$)
cpc_b	average cost-per-click of banner $b \in \mathcal{B}$
cpc_k	initial average cost-per-click of keyword $k \in \mathcal{K}$
δ	proportion of all flow variations attributed to loss
r_v	value of conversion $v \in \mathcal{C}$
Variables	
$y_{k,p}$	$= \begin{cases} 1 & \text{if keyword } k \text{ is assigned position } p \\ 0 & \text{otherwise} \end{cases}$
x_v	new flow going through vertex $v \in \mathcal{V}$ when new positions are assigned to keywords
$x_{(v,w)}$	new flow on arc $(v,w) \in \mathcal{A}$
Additional notations	
s, t	vertices representing the source and the sink of the graph, respectively
l	vertex representing losses
p_k^{init}	initial position of keyword $k \in \mathcal{K}$
n_v	initial flow going through vertex $v \in \mathcal{V}$
$n_{(v,w)}$	initial flow circulating on arc $(v,w) \in \mathcal{A}$
$c_{k,p}$	estimated average cost-per-click of keyword $k \in \mathcal{K}$ when it is displayed at position $p \in \mathcal{P}$
$\Delta_{(v,k)}^p$	estimated variation of flow circulating on (v,k) when keyword $k \in \mathcal{K}$ is displayed at position $p \in \mathcal{P}$
d_v	estimated variation of flow circulating on the arc linking vertex v to the loss vertex l, when new positions are assigned to keywords
μ_v	estimated new flow going through vertex $v \in \mathcal{V}$
$\mu_{(v,w)}$	estimated new flow circulating on arc $(v,w) \in \mathcal{A}$
$\alpha_{(v,w)}$	new proportion of flow going through v to a successor $w \in \Gamma_v^+$ of v.

4.2 Flow conservation

Assume every keywork $k \in \mathcal{K}$ changes its position from p_k^{init} to p_k. The flow (number of clicks) in the graph has to be modified according to assumptions A1, A2 and A3. So consider an arc $(v,w) \in \mathcal{A}$ and let $\mu_{(v,w)}$ be the estimated flow on (v,w) resulting from all these changes:

- if $w \notin \mathcal{K} \cup \{l\}$: the estimated flow $\mu_{(v,w)}$ on (v,w) is equal to the original one $n_{(v,w)}$;
- if $w \in \mathcal{K}$: we set $\mu_{(v,w)} = 0$ if $p_w = \bar{p}$, and we use the estimation formula developed in Quinn [21] if $p_w \neq \bar{p}$. More precisely we set

$$\mu_{(v,w)} = n_{(v,w)} \left(\beta_{click}\right)^{p_w - p_w^{init}} \qquad \text{for all } w \in \mathcal{K}, p_w \neq \bar{p}.$$

These estimations raise the issue of flow conservation. Indeed, if keyword k changes its positions and does not have any keyword among its ancestors in the graph, then the flow entering every vertex $v \in \Gamma_k^-$ cannot vary, while the flow on (v,k) must change. To make sure that flow conservation constraints are respected, the above estimations are treated as requests: every arc $(v,k) \in \mathcal{A}$ with $k \in \mathcal{K}$ requests its current flow (amount of clicks) $n_{(v,k)}$ to be raised/lowered to $\mu_{(v,k)}$.

For a vertex $k \in \mathcal{K}$, let $y_{k,p}$ be a binary variable that equals 1 if the new position p_k of keyword k is p, and 0 otherwise. It follows from the above considerations that the estimated variation $\mu_{(v,k)} - n_{(v,k)}$ of flow circulating on arc (v,k) is equal to $\sum\limits_{p \in \mathcal{P}} y_{k,p} \Delta^p_{(v,k)}$ with

$$\Delta^p_{(v,k)} = \begin{cases} n_{(v,k)} \left[(\beta_{click})^{p - p_k^{init}} - 1 \right] & \text{if } p \neq \bar{p} \\ -n_{(v,k)} & \text{if } p = \bar{p} \end{cases}$$

For an arc (v,k) with $k \in \mathcal{K}$, part of the difference $\mu_{(v,k)} - n_{(v,k)}$ has to be covered by the loss vertex l. If the flow increases on (v,k), it decreases on (v,l), and vice versa. We estimate the variation of flow on (v,l), due to the assignment of position p_k to keyword k, to be $-\delta \sum\limits_{p \in \mathcal{P}} y_{k,p} \Delta^p_{(v,k)}$.

The total variation of flow on (v,l) possibly leads to a negative flow. To avoid such a situation, we limit it to $-n_{(v,l)}$. In summary, the estimated variation of flow on (v,l) is equal to $-d_v$ with $d_v = \min\{n_{(v,l)}, \delta \sum\limits_{u \in \Gamma_v^+ \cap \mathcal{K}} \sum\limits_{p \in \mathcal{P}} y_{u,p} \Delta^p_{(v,u)}\}$. So, consider any vertex $v \in \mathcal{V}$, and let $w \in \Gamma_v^+$ be one of its immediate successors. We set:

$$\begin{aligned} \mu_{(v,w)} &= n_{(v,w)} + \sum_{p \in \mathcal{P}} y_{w,p} \Delta^p_{(v,w)} && \text{if } w \in \mathcal{K} \\ \mu_{(v,w)} &= n_{(v,w)} - d_v && \text{if } w = l \\ \mu_{(v,w)} &= n_{(v,w)} && \text{if } w \notin \mathcal{K} \cup \{l\} \end{aligned}$$

The total estimated flow μ_v going through vertex v is then the sum of the flows on the arcs outgoing v:

$$\mu_v = \sum_{w \in \Gamma_v^+} \mu_{(v,w)} = \sum_{w \in \Gamma_v^+} n_{(v,w)} - d_v + \sum_{u \in \Gamma_v^+ \cap \mathcal{K}} \sum_{p \in \mathcal{P}} y_{u,p} \Delta^p_{(v,u)} = n_v - d_v + \sum_{u \in \Gamma_v^+ \cap \mathcal{K}} \sum_{p \in \mathcal{P}} y_{u,p} \Delta^p_{(v,u)}$$

We finally impose that the proportion $\alpha_{(v,w)}$ of flow going through v to a successor $w \in \Gamma_v^+$ of v is equal to the estimated one. We thus get:

$$x_{(v,w)} = \alpha_{(v,w)} x_v = \frac{\mu_{(v,w)}}{\mu_v} x_v \qquad \text{for all } w \in \Gamma_v^+$$

4.3 Budget constraints

Another constraint to consider in the model is the budget constraint. That is the one that will prevent us from placing all keywords at position 1. Indeed, changing a keyword position will lead to a change in its average cost-per-click. Let $c_{k,p}$ be the estimated average cost-per-click of keyword $k \in \mathcal{K}$ when it is displayed at position $p \in \mathcal{P}$. We set $c_{k,p} = 0$ when $p = \bar{p}$, and we use the formula developed by Quinn [21] when k is displayed at position $p \in \mathcal{P} \setminus \{\bar{p}\}$. More precisely, we set:

$$c_{k,p} = cpc_k \left(\beta_{cpc}\right)^{p - p_k^{init}}.$$

For a solution to be feasible, the combination of the changes in the average costs-per-click and in flow has to respect the budget:

$$\sum_{k \in \mathcal{K}} x_k \sum_{p \in \mathcal{P}} c_{k,p} y_{k,p} + \sum_{b \in \mathcal{B}} cpc_b x_b \leq B$$

Some campaigns also have a budget dedicated to display advertising, another constraint must then be added:

$$\sum_{b\in\mathcal{B}} cpc_b x_b \leq B_{da}.$$

4.4 The proposed model

The goal is to maximize the efficiency of the campaign. Depending on the advertiser's need, the measure of efficiency might differ. To illustrate our model, we have decided to maximize the profit of the advertiser but we could have chosen any other measure. The resulting model reads as follows:

$$\max \quad \sum_{v\in\mathcal{C}} r_v x_v - \left(\sum_{k\in\mathcal{K}} x_k \sum_{p\in\mathcal{P}} y_{k,p} c_{k,p} + \sum_{b\in\mathcal{B}} cpc_b x_b\right) \tag{1}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}} x_k \sum_{p\in\mathcal{P}} y_{k,p} c_{k,p} + \sum_{b\in\mathcal{B}} cpc_b x_b \leq B \tag{2}$$

$$\sum_{b\in\mathcal{B}} cpc_b x_b \leq B_{da} \tag{3}$$

$$d_v = \min\{n_{(v,l)}, \delta \sum_{u\in\Gamma_v^+\cap\mathcal{K}} \sum_{p\in\mathcal{P}} y_{u,p}\Delta^p_{(v,u)}\} \qquad \forall v\in\mathcal{V} \tag{4}$$

$$\alpha_{(v,w)} = \frac{n_{(v,w)} + \sum_{p\in\mathcal{P}} y_{w,p}\Delta^p_{(v,w)}}{n_v - d_v + \sum_{u\in\Gamma_v^+\cap\mathcal{K}} \sum_{p\in\mathcal{P}} y_{u,p}\Delta^p_{(v,u)}} \qquad \forall v\in\mathcal{V}, \forall w\in\Gamma_v^+\cap\mathcal{K} \tag{5}$$

$$\alpha_{(v,l)} = \frac{n_{(v,l)} - d_v}{n_v - d_v + \sum_{u\in\Gamma_v^+\cap\mathcal{K}} \sum_{p\in\mathcal{P}} y_{u,p}\Delta^p_{(v,u)}} \qquad \forall v\in\mathcal{V} \tag{6}$$

$$\alpha_{(v,w)} = \frac{n_{(v,w)}}{n_v - d_v + \sum_{u\in\Gamma_v^+\cap\mathcal{K}} \sum_{p\in\mathcal{P}} y_{u,p}\Delta^p_{(v,u)}} \qquad \forall v\in\mathcal{V}, \forall w\in\Gamma_v^+ \setminus (\mathcal{K}\cup\{l\}) \tag{7}$$

$$\sum_{p\in\mathcal{P}} y_{k,p} = 1 \qquad \forall k\in\mathcal{K} \tag{8}$$

$$x_s = n_s \tag{9}$$

$$x_v = \sum_{u\in\Gamma_v^-} x_{(u,v)} \qquad \forall v\in\mathcal{V}\setminus\{s\} \tag{10}$$

$$x_{(v,w)} = \alpha_{(v,w)} x_v \qquad \forall (v,w)\in\mathcal{A} \tag{11}$$

$$x_v \geq 0 \qquad \forall v\in\mathcal{V} \tag{12}$$

$$x_{(v,w)} \geq 0 \qquad \forall (v,w)\in\mathcal{A} \tag{13}$$

$$y_{k,p} \in \{0,1\} \qquad \forall k\in\mathcal{K}, \forall p\in\mathcal{P} \tag{14}$$

Constraints (10) and (11) are flow conservation constraints. Constraint (9) was added to comply with assumption A1. The particularity of this model is that it becomes linear once the positions p_k of every keyword k are known. Indeed, knowing p_k implies that $y_{k,p}$ can be set equal to 1 if and only if $p = p_k$. This means that the values of d_v and $\alpha_{(v,w)}$ can be fixed. The value of the remaining variables x_v and $x_{(v,w)}$ can then be obtained by determining a solution satisfying equations (9)–(13). We can then determine whether this solution respects budget constraints (2) and (3), and calculate its value with objective (1).

An example

We illustrate in Figure 1 the impact of changing the position of one keyword on the flow of the graph. The source vertex s, the conversion vertex c, the loss vertex l, and the sink t are represented in light gray. The two keyword vertices $k1$ and $k2$ are shown in dark gray. The other vertices are a search query q, a banner b, three vertices $w1, w2, w3$ representing pages with trackers on the client website, and two conversions $c1, c2$. The numbers in the boxes under the vertex names indicate the flow going through each vertex. The numbers on the arcs correspond to their flow. Arcs with no flow are not represented. The graph on the top corresponds to the initial flow while the graph at the bottom shows the impact of assigning position $\bar{p} = 11$ to keyword $k2$, and not changing the position of keyword $k1$.

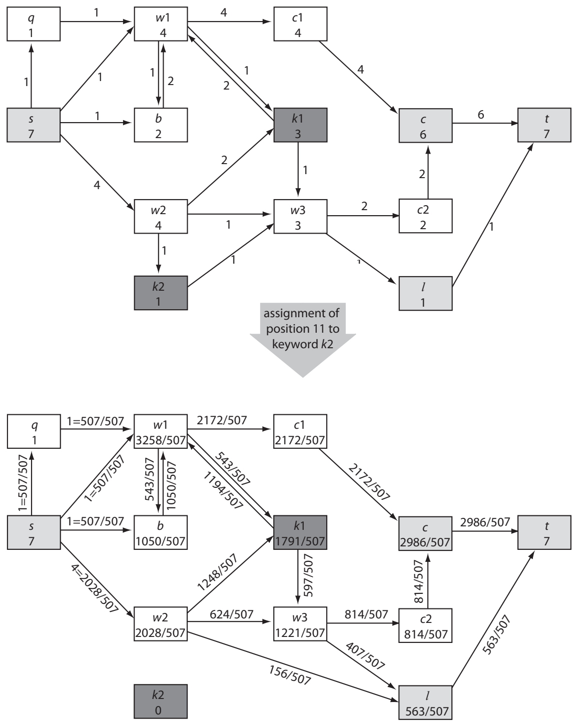

Figure 1: Illustration of the assignment of position $\bar{p}$ to a keyword

The new flow was obtained as follows, assuming $\delta = \frac{1}{4}$. First of all, we have set $y_{k1,p^{init}_{k1}} = y_{k2,\bar{p}} = 1$, while all other $y_{k,p}$ values equal 0. All propotions $\alpha_{(v,w)}$ are kept unchanged, with the exception of those with $v = w2$. We thus have:

$$\Delta^{\bar{p}}_{(w2,k2)} = -n_{(w2,k2)} = -1$$
$$\Delta^{p^{init}_{k1}}_{(w2,k1)} = 0$$

$$d_{w2} = \min\{n_{(w2,l)}, \delta(\Delta^{p^{init}_{k1}}_{(w2,k1)} + \Delta^{\bar{p}}_{(w2,k2)})\} = \min\{0, \frac{1}{4}(-1)\} = -\frac{1}{4}$$

$$\alpha_{(w2,k2)} = \frac{n_{(w2,k2)} + \Delta^{\bar{p}}_{(w2,k2)}}{n_{w2} - d_{w2} + (\Delta^{p^{init}_{k1}}_{(w2,k1)} + \Delta^{\bar{p}}_{(w2,k2)})} = \frac{1-1}{4+\frac{1}{4}+(-1)} = 0$$

$$\alpha_{(w2,k1)} = \frac{n_{(w2,k1)} + \Delta^{p^{init}_{k1}}_{(w2,k1)}}{n_{w2} - d_{w2} + (\Delta^{p^{init}_{k1}}_{(w2,k1)} + \Delta^{\bar{p}}_{(w2,k2)})} = \frac{2+0}{4+\frac{1}{4}+(-1)} = \frac{8}{13}$$

$$\alpha_{(w2,w3)} = \frac{n_{(w2,w3)}}{n_{w2} - d_{w2} + (\Delta^{p_{k1}^{init}}_{(w2,k1)} + \Delta^{\bar{p}}_{(w2,k2)})} = \frac{1}{4 + \frac{1}{4} + (-1)} = \frac{4}{13}$$

$$\alpha_{(w2,l)} = \frac{n_{(w2,l)} - d_{w2}}{n_{w2} - d_{w2} + (\Delta^{p_{k1}^{init}}_{(w2,k1)} + \Delta^{\bar{p}}_{(w2,k2)})} = \frac{0 + \frac{1}{4}}{4 + \frac{1}{4} + (-1)} = \frac{1}{13}.$$

The new flow is then obtained by determining a solution satisfying (9)–(13), with the new values $\alpha_{w,x}$ for $x = k1, k2, w3, l$. In order to facilitate the verification that there is flow conservation at each vertex, all numbers at the bottom of Figure 1 have the common denominator 507. For example, the only arc entering $w2$ is $(s, w2)$, which means that the flow going through $w2$ is equal to $\frac{2028}{507} = 4$. It is distributed to its successors as follows:

- the amount sent to l is $\frac{156}{507}$, which corresponds to a proportion of $\frac{156}{2028} = \frac{1}{13} = \alpha_{(w2,l)}$;
- the amount sent to $w3$ is $\frac{624}{507}$, which corresponds to a proportion of $\frac{624}{2028} = \frac{4}{13} = \alpha_{(w2,w3)}$;
- the amount sent to $k1$ is $\frac{2028-156-624}{507} = \frac{1248}{507}$ which corresponds to a proportion of $\frac{1248}{2028} = \frac{8}{13} = \alpha_{(w2,k1)}$.

Algorithms

Tabu search is a local search technique that visits a search space S by moving step by step from a current solution $s \in S$ to a neighbor solution $s' \in N(s)$, where $N(s)$ is a subset of S called the *neighborhood* of s. A *tabu list* T forbids some moves which would bring the search back to a recently visited solution. Tabu search was introduced in [12]. For more details the reader may refer to [13]. We just give a general scheme of the algorithm for a minimization problem:

```
Generate an initial solution $s \in S$, set $T \leftarrow \emptyset$ and $s^* \leftarrow s$;
**while** *no stopping criterion is met* **do**
    Determine a solution $s' \in N(s)$ with minimum value $f(s')$ such that the move from $s$ to $s'$ does not belong to $T$ or
      $f(s') < f(s^*)$;
    **if** $f(s') < f(s^*)$ **then** set $s^* \leftarrow s'$;
    Set $s \leftarrow s'$ and update $T$;
**end**
```

The following adaptation of tabu search was developed to determine a good solution to the proposed model. A solution is defined as an assignment of a position to every keyword $k \in \mathcal{K}$. In other words, a solution fixes the value of every variable $y_{k,p}$. As mentioned earlier, we can determine the flow F_s associated with a solution s by determining the values of the flow variables that satisfy (9)–(13). The search space S contains all solutions s such that F_s satisfies budget constraints (2) and (3). We use objective (1) to evaluate every solution $s \in S$. The neighborhood $N(s)$ of s contains all solutions $s' \in S$ which can be obtained from s by adding one unit (if $p_k < \bar{p}$) or subtracting one unit (if $p_k > 1$) to the position p_k of exactly one keyword $k \in \mathcal{K}$. When moving from a solution s to a neighbor $s' \in N(s)$, the keyword k to which a new position is assigned enters the tabu list T, and it is forbidden for $2\sqrt{|\mathcal{K}|}$ iterations to modify the position of keyword k, unless such a modification improves the best solution s^* found so far. This basic tabu search algorithm will be called Tabu$_1$. Three variations have been implemented:

- Tabu$_2$ is similar to Tabu$_1$, except that the modification of position of a keyword k is not limited to one unit when moving from s to $s' \in N(s)$.
- Tabu$_3$ is similar to Tabu$_1$, except that we accept to visit solutions that do not respect budget constraints (2) and (3). This means that the search space S contains all possible assignment of positions to the keywords. A component is added to the objective function so that solutions that violate budgets constraints are penalized. More precisely, the value of a solution $s \in S$ is set equal to

$$\sum_{v\in\mathcal{C}} r_v x_v - \left(\sum_{k\in\mathcal{K}} x_k \sum_{p\in\mathcal{P}} y_{k,p}c_{k,p} + \sum_{b\in\mathcal{B}} cpc_b x_b\right)$$
$$-\lambda_1\left(\max\{0, \sum_{k\in\mathcal{K}} x_k \sum_{p\in\mathcal{P}} y_{k,p}c_{k,p} + \sum_{b\in\mathcal{B}} cpc_b x_b - B\}\right) - \lambda_2\left(\max\{0, \sum_{b\in\mathcal{B}} cpc_b x_b - B_{da}\}\right)$$

where λ_1 and λ_2 are penalty factors initialized to 1 and updated every 10 iterations as in [11]: if the last 10 solutions were all feasible with respect to budget constaint (2) (respectively to (3)), then the value of λ_1 (respectively λ_2) is halved, while if they were all infeasible, the penalty factor is doubled.

- Tabu$_4$ is the agorithm obtained from Tabu$_2$ exactly as Tabu$_3$ was obtained from Tabu$_1$: non-feasible solutions are possibly visited, and a keywork can modify its position by more than one unit when moving from a solution s to a neighbor $s' \in N(s)$.

In addition to these four tabu search algorithms, we have implemented a greedy one. It works as follows. Given a solution s, we determine for each keyword k the best value that can be obtained by increasing, decreasing, or maintaining its current position, these modifications being not limited to one unit. The value of a modification is measured by using the same objective function as in Tabu$_2$ and Tabu$_4$, which means that violations of budget constraints are permitted. If a modification of a keyword position leads to a feasible solution (i.e., that satisfies the budget constraints) of better value than the best found solution s^*, then s^* is updated accordingly. We then perform all these modifications simultaneously and consider the resulting solution as the new current solution s for the next iteration. Penalty factor λ is updated every 10 iteration as in Tabu$_2$ and Tabu$_4$. The process is stopped when s is not changed from one iteration to the next one, and the output of the algorithm is then s^*.

The four tabu search algorithms and the greedy one use a procedure that determines the flow F_s associated with a solution s by determining the variable values that satisfy (9)–(13). We have compared four techniques able to solve such a linear system. One possibility is to use the CPLEX solver. Other options are to use the famous Jacobi or Gauss-Seidel methods. Since the order of the variables can have a big impact on the performance of the Gauss-Seidel method, we have implemented a version that uses the following order. Let $s' \in N(s)$ be a neighbor of s, and let k be the keyword having different positions in s and s': we first consider all variables related to k (i.e, variables x_k, $x_{(v,k)}$ with $v \in \mathit{\Gamma}_k^-$ and $x_{(k,v)}$ with $v \in \mathit{\Gamma}_k^+$); we then consider those related to its parents and children, then its grandparents and grandchildren and so on. One iteration of the method updates the value of each variable in the above order. This process is repeated until a desired precision is reached or hundred iterations have been performed, whichever comes first.

Computational experiments

To test the model and the algorithms, we have generated 12 instances, denoted $G_1, \ldots, G_{12}$, with given proportions of each kind of vertex (keyword, banner, etc.). Six of them have 50 keywords, and the other six have 500 keywords. Random paths were generated according to the following rules:

- a path can start with any kind of vertex except conversions;
- a page of the advertiser's website can be followed by any vertex;
- a conversion always marks the end of a path;
- all vertices, except a conversion and a page of the advertiser's website, are always followed by a page of the advertiser's website;
- a path can only end with a conversion or with a page of the advertiser's website.

The average cost-per-click cpc_b of every banner $b \in \mathcal{B}$, the initial average cost-per-click cpc_k of every keyword $k \in \mathcal{K}$, and the revenue r_c of every conversion were chosen randomly in $[0.5, 10]$, $[0.01, 5]$, and

$[10, 100]$, respectively. We have set $\beta_{click} = 0.843$ and $\beta_{cpc} = 0.613$, as suggested in [21]. Also, we have chosen $\delta = \frac{1}{20}$. For every keyword, we have chosen a random position in $\{1, \ldots, 10\}$. Recalling that n_v is the initial flow on vertex v, we thus have a solution s_i for every graph G_i with a cost $(\sum_{k\in\mathcal{K}} n_k \sum_{p\in\mathcal{P}} y_{k,p}c_{k,p} + \sum_{b\in\mathcal{B}} cpc_b n_b)$, a revenue $(\sum_{v\in\mathcal{C}} r_v n_v)$, and a profit (revenue−cost), which are given in Table 1 for comparison with the results produced by the proposed algorithms. The budget B of the campaign was fixed equal to the cost of s_i, while the budget B_{da} for the display advertising campaign was set equal to $\sum_{b\in\mathcal{B}} cpc_b n_b$. These are reasonable assumptions since advertisers tend to use their full budget. The characteristics of the twelve instances are summarized in Table 1, where we indicate :

- the graph identifiers;
- the total number of vertices, the number of vertices of each type, and the total number of arcs;
- the total number of paths, the number of paths that led to a conversion $(\sum_{v\in\mathcal{C}} n_v)$ or to a loss (n_l), and the total number of visits of a page of the client's website $(\sum_{v\in\mathcal{W}} n_v)$;
- the cost, revenue and profit of every s_i.

Table 1: Characteristics of the instances

Graph	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{K}\rvert$	$\lvert\mathcal{B}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{W}\rvert$	$\lvert\mathcal{Q}\rvert$	$\lvert\mathcal{A}\rvert$	paths	con.	losses	visits	cost	revenue	profit
G_1	545	50	104	8	95	284	2190	1000	8	992	1437	1729.91	447.20	-1282.71
G_2	292	50	25	23	77	113	1138	500	23	477	755	516.28	1271.30	755.02
G_3	146	50	27	1	32	32	456	200	1	199	308	571.95	58.60	-513.35
G_4	505	50	129	57	91	174	2026	1000	57	943	1352	2532.21	3076.50	544.29
G_5	279	50	23	15	88	99	936	500	15	485	671	539.74	660.30	120.56
G_6	144	50	26	14	45	5	376	200	14	186	280	577.00	827.60	250.60
G_7	5413	500	1689	663	1879	678	18449	10000	663	9337	12865	30791.61	35799.90	5008.29
G_8	2921	500	663	253	854	647	9859	5000	253	4747	6809	12280.91	13954.60	1673.69
G_9	1416	500	354	46	289	223	4631	2000	46	1954	3076	7613.96	2303.20	-5310.76
G_{10}	5005	500	479	506	1188	2328	19347	10000	506	9494	13306	9643.65	29245.70	19602.05
G_{11}	2754	500	752	142	271	1085	11452	5000	142	4858	7336	14532.41	8248.30	-6284.11
G_{12}	1404	500	171	150	91	488	5090	2000	150	1850	3158	4554.96	8665.80	4110.84

We have described five algorithms (four tabu search algorithms, and a greedy one) and four techniques to solve the linear system (9)–(13), which gives a total of 20 possible methods to solve the considered problem.

For all graphs, we have run every algorithm with 15 different initial solutions: one with all keywords at position 1, one with all keywords at position $\bar{p} = 11$, and 13 with all keyword positions randomly chosen in $\{1, \ldots, 10\}$. The best, worse, and average computing times (in seconds) of each algorithm over the 15 runs are denoted $\underline{t}^x_y$, $\bar{t}^x_y$, and t^x_y, respectively, with $x \in \{T1, T2, T3, T4, Gr\}$ and $y \in \{C, J, GS, oGS\}$, where $T1, T2, T3, T4$ stand for Tabu$_1$, Tabu$_2$, Tabu$_3$, Tabu$_4$, Gr for Greedy, C for CPLEX (version 12.2), J for Jacobi, GS for Gauss-Seidel, and oGS for optimized Gauss-Seidel (with the special order of vertices mentioned in the previous section). Also, the average profit obtained with algorithm x coupled with technique y for the solution of the linear system is denoted z^x_y. All tests were run on a 2.67 GHz Intel Core i7 machine with 8 GB of RAM.

We indicate in Table 2 the best, worse, and average computing times $\underline{t}^{Gr}_{oGS}$, $\bar{t}^{Gr}_{oGS}$, and t^{Gr}_{oGS} obtained with the greedy algorithm coupled with the optimized Gauss-Seidel technique. These values are compared to those obtained with the greedy algorithm coupled with the three other ways of solving the linear system. More precisely, for $y \in \{C, J, GS\}$, we indicate the ratios $\frac{\underline{t}^{Gr}_y}{\underline{t}^{Gr}_{oGS}}$, $\frac{\bar{t}^{Gr}_y}{\bar{t}^{Gr}_{oGS}}$, and $\frac{t^{Gr}_y}{t^{Gr}_{oGS}}$.

We observe that the Gauss-Seidel and optimized Gauss-Seidel methods are faster than the Jacobi and CPLEX methods. For graphs G_3 and G_6, CPLEX is up to 200 times slower. The Jacobi method is typically 3 to 4 times slower than the two versions of the Gauss-Seidel method. For small instances (G_i with $1 \le i \le 6$), the non-optimized version of the Gauss-Seidel technique is a little bit faster than the optimized one, while for larger ones (G_i with $7 \le i \le 12$), it is the opposite.

We have observed in our experiments that the second scenario, which starts with all keywords at position $\bar{p}$, is typically the fastest one and gives the lower bound $\underline{t}^{Gr}_y$ for all $y \in \{C, J, GS, oGS\}$. For this reason, the next

Table 2: Computing times of the greedy algorithm with four different linear system solvers

Graph	optimized Gauss-Seidel			CPLEX			Jacobi			Gauss-Seidel		
	$\underline{t}^{Gr}_{oGS}$	$\bar{t}^{Gr}_{oGS}$	t^{Gr}_{oGS}	$\frac{\underline{t}^{Gr}_{C}}{\underline{t}^{Gr}_{oGS}}$	$\frac{\bar{t}^{Gr}_{C}}{\bar{t}^{Gr}_{oGS}}$	$\frac{t^{Gr}_{C}}{t^{Gr}_{oGS}}$	$\frac{\underline{t}^{Gr}_{J}}{\underline{t}^{Gr}_{oGS}}$	$\frac{\bar{t}^{Gr}_{J}}{\bar{t}^{Gr}_{oGS}}$	$\frac{t^{Gr}_{J}}{t^{Gr}_{oGS}}$	$\frac{\underline{t}^{Gr}_{GS}}{\underline{t}^{Gr}_{oGS}}$	$\frac{\bar{t}^{Gr}_{GS}}{\bar{t}^{Gr}_{oGS}}$	$\frac{t^{Gr}_{GS}}{t^{Gr}_{oGS}}$
G_1	0.70	2.71	1.65	19.03	20.99	21.23	3.80	3.34	3.74	1.00	0.90	0.98
G_2	0.31	0.76	0.53	80.78	76.67	78.26	3.88	3.92	3.95	0.96	1.00	0.99
G_3	0.08	0.23	0.16	253.29	237.32	239.22	3.29	3.53	3.52	0.96	0.99	0.99
G_4	0.85	2.30	1.39	22.02	24.90	23.22	3.34	2.90	3.35	1.00	0.94	0.99
G_5	0.27	0.61	0.41	88.85	103.11	91.08	3.10	3.27	3.29	0.95	0.99	1.00
G_6	0.08	0.17	0.12	244.99	302.01	294.40	2.91	3.43	3.29	0.89	1.02	0.98
G_7	1093.04	1997.93	1495.97	1.19	0.83	1.01	3.48	3.49	3.48	1.01	1.01	1.01
G_8	346.22	746.75	545.49	1.36	1.18	1.31	4.74	3.59	3.60	1.03	1.03	1.04
G_9	75.36	226.69	148.55	3.49	2.67	2.98	3.68	3.70	3.69	1.07	1.06	1.07
G_{10}	1027.09	2112.46	1522.72	1.86	1.88	2.03	3.28	3.32	3.31	1.02	1.04	1.02
G_{11}	317.45	737.52	525.71	1.69	1.43	1.65	3.55	3.50	3.55	1.04	1.03	1.03
G_{12}	88.60	217.78	154.81	3.59	3.21	3.30	3.62	3.55	3.59	1.07	1.04	1.07

reported results are all obtained with this second scenario, and with the optimized Gauss-Seidel technique to solve the linear system.

In Table 3, the computing times and the profits obtained by the greedy algorithm are compared to those produced by the tabu search algorithms. The stopping criteria used for the tabu search algorithms was an absolute difference $|f(s') - f(s)|$ smaller than 10^{-2} between two consecutive visited solutions. We indicate in Table 3 the computing time of the greedy algorthim, and the ratios $\frac{t^{x}_{oGS}}{t^{Gr}_{oGS}}$ for $x \in \{T1, T2, T3, T4\}$. Table 3 contains the profit z^{Gr}_{oGS} obtained with the greedy algorithm, as well as the profit differences $z^{Gr}_{oGS} - z^{x}_{oGS}$ with $x \in \{T1, T2, T3, T4\}$. We observe that the tabu search algorithms are much slower than the greedy one. For instances with 500 keywords, the computing time ratios for Tabu$_1$ and Tabu$_3$ reaches one hundred. Also, the greedy algorithm gets better profits than the tabu search algorithms. This is due to the stopping criteria. If we stop the tabu search algorithms when the absolute difference $|f(s') - f(s)|$ is smaller than 10^{-3} (instead of 10^{-2}) between two consecutive visited solutions, then better profits are obtained. For example, this modified stopping criterion allows Tabu$_2$ to determine a solution with $z^{Gr}_{oGS} - z^{T2}_{oGS} = -3,59\ 10^{-6}$, but the computing time is multiplied by 4. We have run all tabu search algorithms for longer times. While they all can reach the profits produced by the greedy algorithm, they require a very large computing time to gain a few dollars.

Table 3: Comparisons of the greedy algorithm with the four tabu search algorithms coupled with the optimized Gauss-Seidel technique

Graph	Computing times					profits				
	Greedy	Tabu$_1$	Tabu$_2$	Tabu$_3$	Tabu$_4$	Greedy	Tabu$_1$	Tabu$_2$	Tabu$_3$	Tabu$_4$
	t^{Gr}_{oGS}	$\frac{t^{T1}_{oGS}}{t^{Gr}_{oGS}}$	$\frac{t^{T2}_{oGS}}{t^{Gr}_{oGS}}$	$\frac{t^{T3}_{oGS}}{t^{Gr}_{oGS}}$	$\frac{t^{T4}_{oGS}}{t^{Gr}_{oGS}}$	z^{Gr}_{oGS}	z^{Gr}_{oGS}-z^{T1}_{oGS}	z^{Gr}_{oGS}-z^{T2}_{oGS}	z^{Gr}_{oGS}-z^{T3}_{oGS}	z^{Gr}_{oGS}-z^{T4}_{oGS}
G_1	0,70	7,72	75,26	7,78	73,71	-1137,56	0,01	0,00	0,01	0,00
G_2	0,31	45,08	44,23	44,48	44,58	916,68	0,06	0,00	0,06	0,00
G_3	0,08	9,40	5,52	9,32	5,39	-370,29	0,73	$7\ 10^{-8}$	0,73	$7\ 10^{-8}$
G_4	0,85	28,09	13,20	28,05	13,09	713,23	0,08	0,00	0,08	0,00
G_5	0,27	14,11	12,37	14,35	12,62	305,86	0,18	0,01	0,18	0,01
G_6	0,08	27,55	6,00	30,39	6,61	507,53	1,12	1,09	1,12	1,09
G_7	1093,04	102,13	27,05	101,85	27,04	7243,88	14,30	0,96	14,30	0,96
G_8	346,22	81,42	26,63	81,27	26,69	3742,09	20,44	0,95	20,44	0,95
G_9	75,36	54,42	24,97	53,56	25,25	-3719,84	25,22	1,24	25,22	1,24
G_{10}	1027,09	90,95	25,76	91,17	25,91	21462,73	16,78	0,98	16,78	0,98
G_{11}	317,45	57,98	22,88	58,38	23,26	-4843,32	28,54	1,97	28,54	1,97
G_{12}	88,60	59,51	29,66	59,03	29,08	5752,34	41,43	0,80	41,43	0,80

Discussion and conclusions

We have described a model and five algorithms for maximizing the profit of search advertising campaigns for one advertiser. The greedy algorithm turned out to be the fastest one and always found solutions similar in value to those produced by the tabu search algorithms. A few seconds or minutes are sufficient to determine the best positions of a few hundred of keywords. If more time is available (such as hours), the tabu algorithms can produce solutions with slightly higher profit.

While our model assumes that the total flow circulating in the graph is constant (assumption A1), it is in fact possible to update the flow and the graph in it on a regular basis, when new users' historical data are made available.

In our model, parameter δ, which indicates the proportion of flow variations attributed to loss, is the same for every keyword and is fixed. It could be relevant to do a statistical analysis to determine its actual value and if it is acceptable to use the same value for all keywords. That would not translate to much change to the model presented here and none to the optimization algorithms.

Also, in this work, the positions of banner ads are considered as fixed, while in reality, they are also subject to online bidding. This feature could be introduced in our model and make it more realistic.

Finally, advertisers often measure the success of their campaigns not only with the profit, but also with other indicators such as the revenue, the number of visits on their website, the number of conversions and many more. It would be interesting to analyze the impact of the objective of a campaign on its budget management. But this is a subject for future work.

References

[1] AdWords. Bid on keywords in the adwords auction, 2016. https://support.google.com/adwords/answer/6366577.

[2] AdWords. Check and understand quality score, 2016. https://support.google.com/adwords/answer/2454010.

[3] Nikolay Archak, Vahab Mirrokni, and S Muthukrishnan. Budget optimization for online advertising campaigns with carryover effects. In Sixth Ad Auctions Workshop. Citeseer, 2010.

[4] Bing Ads. Learn how bing ads works, 2016. https://help.bingads.microsoft.com/#apex/3/en/53102/0-500.

[5] Christian Borgs, Jennifer Chayes, Nicole Immorlica, Kamal Jain, Omid Etesami, and Mohammad Mahdian. Bid optimization in online advertisement auctions. In Proceedings of the 16th international conference on World Wide Web, pages 531–540. ACM, 2007.

[6] Bhaskar DasGupta and S Muthukrishnan. Stochastic budget optimization in internet advertising. Algorithmica, 65(3):634–661, 2013.

[7] eMarketer. Digital ad spending to surpass tv next year, 2016. http://www.emarketer.com/Article/Digital-Ad-Spending-Surpass-TV-Next-Year/1013671.

[8] eMarketer. Us digital display ad spending to surpass search ad spending in 2016, 2016. http://www.emarketer.com/Article/US-Digital-Display-Ad-Spending-Surpass-Search-Ad-Spending-2016/1013442.

[9] Eyal Even Dar, Vahab S Mirrokni, S Muthukrishnan, Yishay Mansour, and Uri Nadav. Bid optimization for broad match ad auctions. In Proceedings of the 18th international conference on World wide web, pages 231–240. ACM, 2009.

[10] Jon Feldman, S Muthukrishnan, Martin Pal, and Cliff Stein. Budget optimization in search-based advertising auctions. In Proceedings of the 8th ACM conference on Electronic commerce, pages 40–49. ACM, 2007.

[11] Michel Gendreau, Alain Hertz, and Gilbert Laporte. A tabu search heuristic for the vehicle routing problem. Management Science, 40:1276–1290, 1994.

[12] Fred Glover. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 5:533–549, 1986.

[13] Fred W. Glover and Manuel Laguna. Tabu Search. Springer, 1997.

[14] Thore Graepel, Joaquin Q Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), pages 13–20, 2010.

[15] CHW Jayawardane, SK Halgamuge, and U Kayande. Attributing conversion credit in an online environment: An analysis and classification. In Computational and Business Intelligence (ISCBI), 2015 3rd International Symposium on, pages 68–73. IEEE, 2015.

[16] Kinshuk Jerath, Liye Ma, Young-Hoon Park, and Kannan Srinivasan. A ?position paradox? in sponsored search auctions. Marketing Science, 30(4):612–627, 2011.

[17] Hua-Fu Li, Suh-Yin Lee, et al. Mining top-k path traversal patterns over streaming web click-sequences. Journal of Information Science and Engineering, 25(4):1121–1133, 2009.

[18] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1222–1230. ACM, 2013.

[19] Olfa Nasraoui and Raghu Krishnapuram. One step evolutionary mining of context sensitive associations and web navigation patterns. In Proceedings of the 2002 SIAM International Conference on Data Mining, pages 531–547. SIAM, 2002.

[20] Olfa Nasraoui, Cesar Cardona, Carlos Rojas, and Fabio Gonzalez. Mining evolving user profiles in noisy web clickstream data with a scalable immune system clustering algorithm. In Proc. of WebKDD, pages 71–81, 2003.

[21] Patrick Quinn. Modélisation et prédiction du comportement de mots-clés dans des campagnes publicitaires sur les moteurs de recherche. Master's thesis, École Polytechnique de Montréal, 2011.

[22] Oliver J Rutz, Randolph E Bucklin, and Garrett P Sonnier. A latent instrumental variables approach to modeling keyword conversion in paid search advertising. Journal of Marketing Research, 49(3):306–319, 2012.

[23] Weinan Zhang, Ying Zhang, Bin Gao, Yong Yu, Xiaojie Yuan, and Tie-Yan Liu. Joint optimization of bid and budget allocation in sponsored search. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1177–1185. ACM, 2012.

[24] Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In Internet and Network Economics, pages 566–576. Springer, 2008.