A Stochastic Algorithm for Online Bipartite Resource Allocation Problems

Antoine Legrain\textsuperscript{a}, Patrick Jaillet\textsuperscript{b}

\textsuperscript{a}CIRRELT and Polytechnique Montréal, Department of Mathematical and Industrial Engineering, C.P. 6079, Succursale Centre-ville, Montréal, QC, Canada, H3C 3A7
\textsuperscript{b}Laboratory for Information and Decision Systems, Department of Electrical Engineering and Computer Science, and Operations Research Center, MIT, Cambridge, MA, 02139, USA

Abstract

This paper deals with online resource allocation problems whereby buyers with a limited total budget want to purchase items that become available one at a time and which consume some amount of various limited resources upon allocation. Sponsored search advertising is a typical example: in order to maximize revenue, search engines try to choose the best available advertisement to display for each new arriving keyword search.

Two main approaches have been proposed to address such online problems, depending on the assumptions made about the input sequence: one is trying to guarantee a performance against a worst case scenario (sometimes called the adversarial mode); the other one, based on specific probabilistic assumptions about the input, is concerned about expected performance guarantee. However, combining the strengths of these two approaches could potentially outperform current methods. In this paper we propose a practical method that combines these strengths and that requires a limited amount of information about the future. We provide extensive computational results which demonstrate the superior performance of the proposed algorithm.

Keywords: resource allocation, online optimization, primal-dual algorithm, stochastic optimization, L-Shaped method, adwords problem

1. Introduction

According to Bloomberg Businessweek (2006), “Google didn’t make money until it started auctioning ads that appear alongside the search results. Advertising today accounts for 99% of the revenue”. Google’s advertisements generated more than 59 billion dollars in 2014 (Google, 2014). Online optimization techniques are useful in such settings and combined with modern statistical tools can lead to significant benefits. For such online optimization problems, detailed information (about users, clients, and/or advertisements) is

\begin{flushleft}
\textit{Email addresses:} antoine.legrain@polymtl.ca (Antoine Legrain), jaillet@mit.edu (Patrick Jaillet)
\end{flushleft}
typically revealed step by step, and irrevocable decisions must be taken along the way. The AdWords problem (introduced in (Mehta et al., 2007)) is a typical example: search engines aim at maximizing their revenue by choosing the best possible advertisement to display for each new keyword search.

Here we consider a general class of resource-constrained allocation problems where items arrive one by one and must be allocated upon arrival among a set of buyers. More formally, this problem, hereafter called the online bipartite resource allocation problem, can be described as follows:

- each buyer in a set \{1 \ldots N\} is interested in purchasing one or more items from a set \{1 \ldots M\} of distinct object types;

- a total of \( T \) items come one at a time (\( T \) requests);

- buyer \( i \in \{1 \ldots N\} \) is willing to pay \( c_{ik} \) for each item of type \( k \in \{1 \ldots M\} \) and has a limited total budget \( B_i \);

- an allocation of an item to a buyer consumes some specific amount of \( L \) distinct and limited resources; more specifically:
  - a total of \( F_k \in \mathbb{R}_{+}^L \) amounts of these \( L \) distinct resources are available for the overall allocations of items of type \( k \in \{1 \ldots M\} \);
  - each item of type \( k \in \{1 \ldots M\} \) consumes amounts \((d_{ik})_l \) of resources \( F_k \) and provides a revenue \( c_{ik} \) to the operator when allocated to buyer \( i \in \{1 \ldots N\} \);

- a central resource allocation platform (the "operator") is in charge of allocating items to potential buyers, with the goal of maximizing total revenue subject to budget and resource constraints.

A mathematical programming formulation corresponding to this problem can be written as follows:

\[
\max \sum_{i=1}^{N} \sum_{j=1}^{T} c_{ik} x_{ij} \quad \text{(1a)}
\]

subject to:

\[
\sum_{i=1}^{N} x_{ij} \leq 1 \quad \forall j = 1 \ldots T \quad \text{(1b)}
\]

\[
\sum_{j=1}^{T} c_{ik} x_{ij} \leq B_i \quad \forall i = 1 \ldots N \quad \text{(1c)}
\]

\[
\sum_{i=1}^{N} \sum_{j=1 \mid k_j = k}^{T} d_{ik} x_{ij} \leq F_k \quad \forall k = 1 \ldots M \quad \text{(1d)}
\]

\[
x_{ij} \in \{0, 1\} \quad \forall i = 1 \ldots N, \forall j = 1 \ldots T \quad \text{(1e)}
\]
The objective (1a) is to maximize the revenue of the operator for $T$ requests. Constraints (1b) ensure that a request is allocated no more than once. Constraints (1c) are budget constraints limiting the total expense for each buyer $i$. Finally, constraints (1d) ensure that all allocated items of type $k$ do not consume more than the available resources. In the online case, the type $k_j$ of the $j$th request is unknown ahead of time and the total number of items of type $k$ out of the $T$ requests (say $n_k$) is also a priori unknown. In some cases ($\frac{n_k}{T}$) can be assumed to follow some (known or estimated) stochastic processes. The challenge for an online strategy is to decide the allocation of each request without knowing the exact sequence of future requests.

A wide range of applications can be modeled by this online bipartite resource allocation problem. In addition to the search ad display example introduced above, there are applications in routing, revenue management, and scheduling. For example, Google has published two anonymous sets of real-data about their compute clusters (Reiss et al., 2011). Tasks with different priorities are continuously arriving from several services and must be placed in the clusters’ workload while respecting a number of constraints: first, hardware constraints dealing with disk space, memory space and the number of CPUs, and, second, software constraints ensuring the right configuration of virtual machines. Space allocation problems also deal with online allocation and appear in the hospitality, rail, and airline industries, to name a few. Clients purchase a service from a company which must price it beforehand according to the current availability of their hotel, their train, and/or their plane. Other applications worth mentioning correspond to patient scheduling problems. Patients arriving in hospitals or clinics must be able to obtain an appointment while resources should also be kept available for future possible high priority patients (Legrain et al., 2014).

Given the uncertainty about the arrival of future requests, it is challenging to both balance resources among different categories of items and infer future resource needs to avoid under- or over-provisioning. It is difficult to design general algorithms which perform well under uncertainty. One way of handling uncertainty is through stochastic optimization techniques (e.g., multistage stochastic programs (Birge and Louveaux, 2011), Markov decision processes (Puterman, 2014)), or via robust optimization (Bertsimas and Sim, 2003). These approaches provide good results when uncertainty can be reasonably well approximated by probability distributions. Uncertainty can also be handled through online optimization approaches (Jaillet and Wagner, 2010) which try to ensure a certain quality of solutions given by online algorithms, designed without knowledge about the future. The competitive ratio $c$ measures the quality of an online algorithm and for a maximization problem is defined as $\min_I \{ \frac{\text{Obj}_{\text{online}}(I)}{\text{Obj}_{\text{optimal}}(I)} \}$. $\text{Obj}_{\text{online}}(I)$ is the value of the objective for the solution given by the online algorithm and $\text{Obj}_{\text{optimal}}(I)$ is the value of the objective for the offline solution for an instance $I$. We then say that the online algorithm is $c$-competitive. Different versions of the bipartite resource allocation problems have been studied in this online fashion. Karp et al. (1990) deals with a simple form of this problem maximizing the number of requests matched to buyers (i.e., model (1) without constraints (1d) and with $B_i = 1$, $c_{ik} = 1$). The authors
proposed a best possible $1 - \frac{1}{e}$-competitive randomized algorithm. Kalyanasundaram and Pruhs (2000) provide a $1 - \frac{1}{e}$-competitive algorithm for the b-matching problem which is defined as a bipartite resource allocation problem where each buyer can be matched at most b times (i.e., model (1) without constraints (1d) and with $B_i = b$, $c_{ik} = 1$). They also prove that this competitive ratio is the best possible. Mehta et al. (2007) introduce the Adwords problem (i.e., model (1) without constraints (1d)) and propose a $1 - \frac{1}{e}$-competitive algorithm. Buchbinder et al. (2007) show that a primal-dual algorithm can be designed for this problem with the same competitive ratio. Jaillet and Lu (2011) propose a $\frac{1}{2}$-competitive primal-dual algorithm for the bipartite resource allocation problem in a special homogeneous case (i.e., model (1) with $c_{ik} = d_{ik}$).

Other authors (Feldman et al. (2009); Karande et al. (2011); Manshadi et al. (2012); Jaillet and Lu (2013)) combine online and stochastic ideas. They study the online bipartite matching problem and improve the bounds on the competitive ratio from $1 - \frac{1}{e}$ up to 0.706 (Jaillet and Lu, 2013) by using statistics about offline strategies. These papers assume that the requests are independent and identically distributed from a known probability distribution.

Bent and Van Hentenryck (2005) propose three architectures to build an online procedure with stochastic information. These architectures use the same ideas: first, the future is sampled and then an offline algorithm is used to solve the problem with the sampled events.

Recent papers present algorithms solving the resource allocation problems with dynamic stochastic information: an offline linear programming problem is used to update the future strategy. Ciocan and Farias (2012) have proved a worst case guarantee on average (i.e., an expected competitive ratio of 0.342). Based on statistics, the algorithm computes an initial strategy allocating a percentage of item $k$ to buyer $i$. Primal problems are then solved during the allocation process to update this strategy given the new information. In contrast, Feldman et al. (2010) and Jaillet and Lu (2012) use the dual solution to improve their current strategy. Jaillet and Lu (2012) also infer the total number of requests $T$.

Most of the aforementioned articles analyze the algorithms from a theoretical perspective. However, many assumed parameters in these approaches remain unknown in reality. In this paper, we propose several improvements to current approaches so as to solve practical online bipartite resource allocation problems, making use of available stochastic information. Our work builds upon the primal-dual algorithm presented in Buchbinder et al. (2007). Our main contributions are as follows:

**A Stochastic Online Algorithm:** We assume that an underlying stochastic process describes the arrival of requests. Our algorithm takes into account future requests to infer the expected revenue from an allocation. We make the best decision for the current request by maximizing this revenue. This procedure provides high quality solutions, but it remains computationally demanding.

**A Re-optimized Primal-dual Algorithm:** The previous algorithm is modified to
estimate the dual variables in the primal-dual procedure. As deterministic algorithms may make poor decisions, leading to a significant deterioration of the solution, we aim at correcting these mistakes by performing updates of the dual variables during the process.

**An Estimation of the Future:** We assume that the stochastic process of the demand is initially unknown. We first use machine learning tools to infer the probability distribution of the arrival rates of $M$ types based on historical data. We then use an optimization problem to estimate upon each arrival of a request the number of remaining future requests. The quality of this last inference is crucial to obtain an overall good solution.

**Computational Experiments:** We conduct numerical tests over different scenarios to compare four algorithms. We also analyze the sensitivity of our scheme to different parameters. The results show that our procedure performs very well for most scenarios leading to a competitive ratio above 0.9.

**Outline:** The remainder of this paper is organized as follows. Section 2 introduces the stochastic online algorithm. Section 3 presents different modifications on the algorithm to solve more realistic problems. Section 4 provides extensive numerical results. Finally, conclusions are drawn in Section 5.

### 2. Stochastic Algorithm

In this section, we present a stochastic optimization formulation and an algorithm to solve the online bipartite resource allocation model (1) in an online fashion. The proposed procedure tries to infer the future and use the information in order to improve the current primal-dual algorithm. We assume here that the number of requests is known and the demand (i.e., the type of each request) is described by a stochastic process $(X^k_j)$: $X^k_j = 1$ if the $j$th request is an item of type $k$, 0 otherwise. So we suppose that we have the following information:

- $T$ the total number of requests;
- the distribution of the stochastic process $(X^k_j)_{j=1}^T$.

From this information, those next parameters can be computed:

- $T_j$ the number of requests left after the $j$th request. So $T_j = T - j$;
- $\Omega_j$ the set of the future sample events i.e., the future scenarios. Each event $\omega \in \Omega_j$ has the same number of elements $T_j$;
- $p^\omega$ is the probability of the event $\omega \in \Omega_j$;
- $T^\omega_{jk}$ is the number of requests for items of type $k$ in the event $\omega$. So $T_j = \sum_{k=1}^M T^\omega_{jk}$.

It can be noted that the specific order in the sequence of the future requests associated with a given event does not matter, as decisions on how to allocate them under this particular event would be done with the full knowledge of the event, and thus all at once.
2.1. Stochastic optimization formulation

We present here a two-stage stochastic integer program with fixed recourse (Birge and Louveaux, 2011) for solving online decisions for our problem. We suppose that the $j$th request has just arrived and that we have to make a decision. The objective is to maximize the expected revenue considering all possible future requests as given by all the events $\omega \in \Omega_j$. Let us define the following new variables:

- $B_i^{left}$ is the budget left for the buyer $i$;
- $x_i$ is equal to 1 if the $j$th request is allocated to the buyer $i$, 0 otherwise;
- $y_{ik}^\omega$ is the number of item $k$ allocated to the buyer $i$ for the event $\omega$.

At the time of the $j$th request, we obtain the following formulation:

\[
\max \sum_{i=1}^{N} c_{ikj} x_i + \sum_{\omega \in \Omega_j} \rho^\omega \sum_{i=1}^{N} M \sum_{k=1}^{M} c_{ik} y_{ik}^\omega \tag{2a}
\]

subject to:

\[
\sum_{i=1}^{N} x_i \leq 1 \tag{2b}
\]

\[
\sum_{i=1}^{N} y_{ik}^\omega \leq T_{jk}^\omega \quad \forall \omega \in \Omega_j, \forall k = 1 \ldots M \tag{2c}
\]

\[
c_{ikj} x_i + \sum_{k=1}^{M} c_{ik} y_{ik}^\omega \leq B_i^{left} \quad \forall \omega \in \Omega_j, \forall i = 1 \ldots N \tag{2d}
\]

\[
x_i \in \{0, 1\}, y_{ik}^\omega \in \mathbb{N} \quad \forall \omega \in \Omega_j, \forall i = 1 \ldots N, \forall k = 1 \ldots M \tag{2e}
\]

The objective (2a) maximizes the revenue for the $j$th request and the expected revenue of remaining future requests. The constraint (2b) ensures that the $j$th request is allocated to a maximum of one buyer. The constraints (2c) bound by $T_{jk}^\omega$ the number of items of type $k$ allocated to all buyers for the event $\omega$. Finally, the constraints (2d) prevent exceeding the budget for each buyer and each event. The formulation (2) is the simplest that we can consider to describe locally the best decisions to be made upon the arrival of a new request. At the same time this is a very large and difficult problem to solve. The relaxation of the integer constraint (2e) on $y_{ik}^\omega$ is a first good idea to simplify the model. The L-Shaped method, presented in the next section, is a second way to improve the computational time.
2.2. L-Shaped Method

In their book, Birge and Louveaux (2011) study different stochastic problems and show how to use the L-Shaped method to solve them. This technique is based on a Benders decomposition (Benders, 1962). Concentrating on our specific problem, the idea is to decompose the optimization problem (2) in a master problem and slave problems and then approximate the objective of each slave problems using some cuts.

For the master problem, we replace the part of the objective dealing with scenarios in (2a) by a recourse function $Q$, which becomes the objective function in the slave problems. Then, we obtain the problems presented in Table 1.

<table>
<thead>
<tr>
<th>Master Problem</th>
<th>Slave Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\max \sum_{i=1}^{N} c_{ik}x_i + \sum_{\omega \in \Omega_j} p^\omega Q(x, \omega)$</td>
<td>$Q(x, \omega) = \max \sum_{i=1}^{N} \sum_{k=1}^{M} c_{ik}y_{ik}^\omega$</td>
</tr>
<tr>
<td>subject to: $\sum_{i=1}^{N} x_i \leq 1$</td>
<td>subject to: $\sum_{i=1}^{N} y_{ik}^\omega \leq T_{jk}^\omega \ \forall k = 1 \ldots M$</td>
</tr>
<tr>
<td>$x_i \in {0, 1} \ \forall i = 1 \ldots N$</td>
<td>$\sum_{k=1}^{M} c_{ik}y_{ik}^\omega \leq B_{i}^{left} - c_{ik}x_i \ \forall i = 1 \ldots N$</td>
</tr>
<tr>
<td>$y_{ik}^\omega \in \mathbb{N} \ \forall i = 1 \ldots N, \forall k = 1 \ldots M$</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Decomposition in sub-problems

The recourse function $Q$ has to be computed for each value of the variable $x$ and for each event $\omega$. In order to have an approximation of $Q$, we use the dual of the slave problems. In our case, the solution of this problem gives a cut for the master problem. Table 2 presents a new master problem where the cuts approximate the recourse function $Q$.

<table>
<thead>
<tr>
<th>Master Problem</th>
<th>Dual Slave Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\max \sum_{i=1}^{N} c_{ik}x_i + \sum_{\omega \in \Omega_j} p^\omega \theta^\omega$</td>
<td>$\min \sum_{k=1}^{M} T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^{N} (B_{i}^{left} - c_{ik}x_i) \beta_i^\omega$</td>
</tr>
<tr>
<td>subject to: $\sum_{i=1}^{N} x_i \leq 1$</td>
<td>subject to: $\alpha_k^\omega + c_{ik} \beta_i^\omega \geq c_{ik} \ \forall i = 1 \ldots N, \forall k = 1 \ldots M$</td>
</tr>
<tr>
<td>$\theta^\omega \leq \sum_{k=1}^{M} T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^{N} (B_{i}^{left} - c_{ik}x_i) \beta_i^\omega \ \forall \omega \in \Omega_j$</td>
<td>$\alpha_k^\omega, \beta_i^\omega \geq 0 \ \forall i = 1 \ldots N, \forall k = 1 \ldots M$</td>
</tr>
</tbody>
</table>
| $x_i \in \{0, 1\} \ \forall i = 1 \ldots N$ | $|}$

Table 2: Benders decomposition
It can be noted that the weak duality theorem justifies these approximations:

\[
\forall x \in [0, 1], \omega \in \Omega_j, Q(x, \omega) \leq \sum_{k=1}^{M} T_{jk}^{\omega} \alpha_{k}^{\omega} + \sum_{i=1}^{N} (B_{i}^{left} - c_{ik} x_{i}) \beta_{i}^{\omega}
\]  

(3)

1. Set \( x = 0 \)
2. Solve all the dual slave problems and add every cuts to the master problem
3. Solve the master problem:
   - if the solution \( x \) remains the same, STOP.
   - otherwise GO TO 2.

Figure 1: The L-Shaped procedure

The L-Shaped algorithm, presented in Figure 1, stops as soon as the optimum is reached, otherwise some cuts continue to be generated. We make an additional simplification, and assume that the master problem is solved only once. In addition to decreasing the computational time, this also allows us to make an easy link with the primal-dual algorithm as explained in Section 2.3.

Algorithm 1 Stochastic primal-dual Algorithm

\[
\begin{align*}
x_{ij} &= 0 \\
\text{for all } j\text{th request do} \\
&\quad \text{Use a greedy algorithm to set } x \\
&\quad \text{Solve all the dual slave problems and add every cuts to the master problem} \\
&\quad \text{Solve the master problem and keep this solution } x \\
\end{align*}
\]

end for

For each request, algorithm 1 chooses first the buyer which offers the highest bid, then, if the master problem gives the same solution, this is the global optimum; otherwise the solution of the master problem is only a local optimum depending on the current cuts. This algorithm does provide promising results as confirmed by our computational tests in Section 4.

2.3. Links with the Primal-dual

The inequalities (3) become equalities in the master problem; indeed, as this is a maximization problem, each variable \( \theta^\omega \) will take on a value at optimality which will be the corresponding upper bound defined by the constraint (3). So with this new formulation, we are able to write:

\[
\forall x \in [0, 1], \omega \in \Omega_j, \quad \theta^\omega = \sum_{k=1}^{M} T_{jk}^{\omega} \alpha_{k}^{\omega} + \sum_{i=1}^{N} (B_{i}^{left} - c_{ik} x_{i}) \beta_{i}^{\omega}
\]
With all those equalities, the variables $\theta^\omega$ can be eliminated from the formulation. The updated cuts can thus be integrated in the objective:

$$\sum_{i=1}^{N} c_{ikj} x_i + \sum_{\omega \in \Omega_j} \rho^\omega \left[ \sum_{k=1}^{M} T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^{N} (B_{i \leftarrow ft} - c_{ikj} x_i) \beta_i^\omega \right]$$

All the constants are taken off the objective to obtain $\sum_{i=1}^{N} c_{ikj} x_i \left( 1 - \left[ \sum_{\omega \in \Omega_j} \rho^\omega \beta_i^\omega \right] \right)$. The cost of the variable $x_i$ in this objective is exactly the same as its cost in the primal-dual algorithm ($c_{ikj} (1 - r_i)$); the only difference is the way to compute the dual variables $r_i$. The primal-dual algorithm updates $r_i$ with $r_i = r_i (1 + \frac{c_{ikj}}{B_i}) + \frac{c_{ikj}}{(c-1)B_i}$. In our case, we use stochastic information to build the dual variables $r_i = \sum_{\omega \in \Omega_j} \rho^\omega \beta_i^\omega$. The cost $c_{ikj} - c_{ikj} r_i$ can be interpreted as the difference between two revenues:

- $c_{ikj}$ is the benefit that the operator earns immediately if the $j$th request is matched to the buyer $i$;
- $c_{ikj} r_i$ is the expected loss if such an allocation is chosen (the future budget $B_{i \leftarrow ft}$ will be reduced by $c_{ikj}$).

The stochastic primal-dual algorithm is just seeking an equilibrium between the instant revenue $c_{ikj}$ and the expected loss.

2.4. Generalized Problems

We apply those techniques to the general model (1). This leads to the following stochastic optimization where $F_{k \leftarrow ft}$ is the remaining amount of resources left for allocating future items of type $k$. This is the new master problem:

$$\max \sum_{i=1}^{N} \left\{ c_{ikj} - \left[ \sum_{\omega \in \Omega_j} \rho^\omega (c_{ikj} \beta_i^\omega + d_{ikj} \gamma_k^\omega) \right] \right\} x_i$$

subject to:

$$\sum_{i=1}^{N} x_i \leq 1$$

$$x_i \in \{0, 1\} \quad \forall i = 1 \ldots N$$

with the dual slave problems:

$$\min \sum_{k=1}^{M} T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^{N} \left[ (B_{i \leftarrow ft} - c_{ikj} x_i) \beta_i^\omega - d_{ikj} x_i \gamma_k^\omega \right] + \sum_{k=1}^{M} F_{k \leftarrow ft} \gamma_k^\omega$$
subject to:

\[ \begin{align*}
\alpha_k^\omega + c_{ik}\beta_i^\omega + d_{ik}\gamma_k^\omega & \geq c_{ik}, \quad \forall i = 1 \ldots N, \forall k = 1 \ldots M \\
\alpha_k^\omega, \beta_i^\omega, \gamma_k^\omega & \geq 0, \quad \forall i = 1 \ldots N, \forall k = 1 \ldots M
\end{align*} \]

The slave problems have more variables and the dual variables \( r_i \) are now equal to \( \sum_{\omega \in \Omega_j} p^\omega (\beta_i^\omega + \frac{d_{ikj}}{c_{ikj}} \gamma_k^\omega) \). This new algorithm follows the same procedure as before.

2.5. Re-optimized Primal-dual

Algorithm 1 should be better than the primal-dual algorithm, as the updates of the dual variables \( r_i \) use stochastic information. However, this algorithm is too slow in a real time environment: solving a linear problem at each arrival of a request is much more demanding in computational time. Consequently, we are proposing a re-optimized algorithm, for which the linear problem will be solve only each \( \Delta \) requests.

Algorithm 2 Re-optimized primal-dual Algorithm

\begin{align*}
   x_{ij} &= 0, r_i = 0 \\
   \text{for all jth request do} & \\
   & \text{if } j \equiv 0 \text{ (mod } \Delta) \text{ then} \\
   & \quad \text{MAKE one step of the stochastic primal-dual Algorithm 1} \\
   & \quad \text{UPDATE } r_i = \rho r_i + (1 - \rho)\left[\sum_{\omega \in \Omega_j} p^\omega \beta_i^\omega\right], \quad \forall i = 1 \ldots N \\
   & \text{end if} \\
   & \text{FIND a buyer } i \text{ who MAXIMIZES } c_{ikj}(1 - r_i) \\
   & \text{REQUIRE } r_i < 1 \text{ AND there is enough budget } B_i \text{ left} \\
   & \text{SET } x_{ij} = 1 \\
   & \quad \text{UPDATE } r_i = r_i(1 + \frac{c_{ikj}}{B_i}) + \frac{c_{ikj}}{(c-1)B_i} \text{ if a re-optimization has not been made} \\
   \text{end for}
\end{align*}

Algorithm 2 will use most of the time the same updates as used by the primal-dual algorithm, taking advantage of the competitive ratio of \( 1 - \frac{1}{e} \) provided by the primal-dual algorithm. Each \( \Delta \) requests, Algorithm 2 will use corrective updates to fix mistakes, which may have been made during the previous steps, by updating the dual variables with a step of Algorithm 1. Furthermore, the parameter \( \rho \) allows us to smooth the value of the dual variables at each re-optimization. Results are shown in Section 4. In the next section, we propose additional modifications to use this algorithm in practical settings of interest.

3. Practical Modifications

In this section, we propose some ideas to transform Algorithm 2 and adapt it to make it practical to use in realistic situations. The competitive ratio is used to compare the quality of an improvement.
3.1. Improvement of the Computational Time

The set of the sample events $\Omega_j$ is huge, it is impossible to compute the slave problems for all events $\omega$. Table 3 presents the competitive ratio for different size of $\Omega_j$. Those tests have been made with 300 requests, which were independently and identically distributed (i.i.d.) and each competitive ratio is an average over 500 draws.

| $|\Omega_j|$ | 1   | 2   | 3   | 5   | 10  |
|-----------|-----|-----|-----|-----|-----|
| Competitive ratio | 0.995 | 0.995 | 0.996 | 0.996 | 0.996 |

Table 3: Competitive ratio as a function of $|\Omega_j|$ for the stochastic primal-dual algorithm

As the competitive ratio does not really increase with the size of the sample set $\Omega_j$, we propose to solve the dual slave problems (Table 2) for only one random event $\omega_0$. Indeed, the cuts (3) $\theta^\omega \leq \sum_{k=1}^M T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^N \left[ \left( B_i^{left} - c_{ikj}^\omega x_i \right) \beta_i^\omega - d_{ikj}^\omega x_i \gamma_k^\omega \right] + \sum_{k=1}^M F_k^{left} \gamma_k^\omega$ remain valid inequalities for each event $\omega$. As the constraints of the dual slave problems are independent of the events, the dual solution for the event $\omega_0$ remains feasible for every events, and thus the cuts always holds.

Furthermore, the generation of one random event $\omega_0$ instead of a deterministic one leads to a randomized algorithm. According to Karp (1991), a randomized algorithm may avoid worst-case behaviors associated with a deterministic algorithm.

With only one event, the objective function now becomes $\sum_{i=1}^N \left\{ c_{ikj}^\omega - (c_{ikj}^\omega \beta_i^\omega + d_{ikj}^\omega \gamma_k^\omega) \right\} x_i$ and the computational time decreases dramatically.

3.2. Bayesian Inference

In practice, it is rare to know the distribution of the stochastic process $(X^k_j)^T_{j=1}$. That is why we propose to use Bayesian Statistics (Bolstad (2004)) to infer this distribution from the current historical data. We first focus on one type of item; we forget the index $k$ of the process. Let us assume that $(X^k_j)^T_{j=1}$ is a Bernoulli process: that is, the stochastic process $(X^k_j)^T_{j=1}$ is i.i.d. and each $X^k_j$ follows a Bernoulli distribution of mean $\mu$ in $[0,1]$.

$$X^k_j = \begin{cases} 1 & \text{with probability } \mu \\ 0 & \text{otherwise} \end{cases}$$

Assume that $\mu$ is unknown, but that it follows a prior distribution $\mathbb{P}[\mu|\alpha]$. Then, it is well-known that $\forall j \in \{1 \ldots T\}$, $\mathbb{P}[\mu|X^k_j]^T_{j=1}, \alpha]$ is a beta distribution $\beta(a_j, b_j)$, if the prior distribution $\mathbb{P}[\mu|\alpha]$ is also a beta distribution $\beta(a, b)$. Furthermore, $a_j = a + \sum_{i=1}^j X^k_i$ and $b_j = b + \sum_{i=1}^j (1 - X^k_i)$. We can now estimate $\hat{\mu}_{j+1} = \mathbb{P}[X^k_{j+1} = 1|X^k_j]^T_{j=1}, a, b]$: $\hat{\mu}_{j+1} = \mathbb{E}[\mu|X^k_j]^T_{j=1}, a, b] = \frac{a_j}{a_j + b_j} = \frac{a + \sum_{i=1}^j X^k_i}{a + b + j}$

We obtain an estimation of the probability $p_k$ that an item of type $k$ arrives on the $(j+1)$th request. As all the probabilities must be inferred at the same time, let us now consider that
$(X^k_j)^T_{j=1}$ follows a Bernoulli process for each item $k$:

$$X^k_j = \begin{cases} 1 & \text{if } k = k_j \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in 1 \ldots T, \forall k = 1 \ldots M$$

It can be noted that $\sum_{k=1}^{M} X^k_j = 1, \forall j = 1 \ldots T$. With the same notations as before, we obtain that $\forall j = 1 \ldots T, \forall k = 1 \ldots M$, $X^k_j$ follows a beta distribution $\beta(a^k_j, b^k_j)$ and that $\hat{\mu}^k_j = \frac{a^k_j + \sum_{l=1}^{j} X^k_l}{a^k + b^k + j}$. As we have no information before the first request, we suppose that all requests are equiprobable ($\mathbb{E}[\beta(a^k, b^k)] = \frac{1}{M}$) and i.d. $(a^k = a, b^k = b)$. So $a^k + b^k = Ma^k = Ma$. We also remark that:

$$\sum_{k=1}^{M} \hat{\mu}^k_j = \sum_{k=1}^{M} \frac{a + \sum_{l=1}^{j} X^k_l}{Ma + j} = \frac{Ma + \sum_{l=1}^{j} \sum_{k=1}^{M} X^k_l}{Ma + j} = \frac{Ma + j}{Ma + j} = 1 \quad \forall j = 1 \ldots T$$

Consequently, $\hat{\mu}^k_j$ can be interpreted as an estimation of the probability that the $(j + 1)$th request is an item of type $k$. We use this method to infer every $p_k$.

### 3.3. Adaptive Horizon

Most papers suppose that the total number of requests is known. However, in practice, this number is unknown (see also a discussion and theoretical treatment of this issue in (Jaillet and Lu, 2012)). In our case, we will present two ways to deal with this issue. First, we will infer the number of requests left $T_j$ using a linear program. Second, we will consider a different model of request arrivals, and assume as in Jaillet and Lu (2012), that the distribution governing the arrival times of the requests is known.

Table 4 shows different competitive ratios obtained using tests with 300 requests, the competitive ratios being an average computed over 500 i.i.d. draws.

<table>
<thead>
<tr>
<th>greedy</th>
<th>primal-dual</th>
<th>stochastic primal-dual</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean of the competitive ratio</td>
<td>0.901</td>
<td>0.957</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Comparison of competitive ratios

Table 4 shows that the choice of the value $T_j$ has a stronger impact on the quality of the solution than the inference of the probabilities. If the number of requests left $T_j$ is not estimated correctly, the stochastic primal-dual algorithm is worse than the primal-dual procedure. It is thus important to estimate $T_j$ accurately. The optimization model (4) seems to work well to infer $T_j$, but other estimation of $T_j$ could be used depending on the specific particularity of a given application.

$$\min T_j$$
subject to:

\[ \sum_{i=1}^{N} \left( \sum_{l=1}^{j-1} c_{ikl} x_{il} + \sum_{k=1}^{M} c_{ik} y_{ik} \right) \geq \sum_{i=1}^{N} B_{i} \quad (4b) \]

\[ \sum_{i=1}^{N} y_{ik} \leq p_{k} T_{j} \quad \forall k = 1 \ldots M \quad (4c) \]

\[ y_{ik} \geq 0 \quad \forall i = 1 \ldots N, \forall k = 1 \ldots M \quad (4d) \]

The idea is to calculate the minimum \( T_{j} \) such that there are enough potential future requests to fill the remaining budget. Constraints (4b) force the optimization problem to fill the whole budget, while constraints (4c) ensure that the number of allocated items of type \( k \) does not exceed the expected number \( p_{k} T_{j} \) of items of type \( k \). As two linear problems have now to be solved, the computational time of the re-optimized primal-dual algorithm is double (tests have been made). However, if we make few re-optimizations, doubling the computational time should not affect significantly the algorithm. Results and a sensitivity analysis will be presented in Section 4.1.4.

The second method to estimate the number of requests \( T_{j} \) requires that we change our way to model the problem. Instead of defining a problem by its number of requests \( T \), we propose to solve the online bipartite resource allocation problem over a planing horizon as Jaillet and Lu (2012) did. Requests arrive now randomly over this horizon: each request is modeled by a stochastic process \( (X_{t}^{k})_{t \in [0,T]} \). \( T \) now represents the end of the process, is assumed known and, consequently, does not need to be inferred. We introduce a daily Adwords problem where the algorithm maximizes the search engine revenue over one day (\( T = 24 \) hours) and \( B_{i} \) is the daily budget of buyer \( i \). In the next section, we present some numerical results integrating these practical modifications.

4. Numerical Results

All the tests have been computed on the following computer: Intel(R) Xeon(TM) CPU 2.66GHz with 1 Gb of Memory. The software CPLEX 12.4 is used to solve the linear programs (Table 2) and (4). The sequences of requests \( (k_{j})_{j=1,...,L} \) follow a multinomial distribution. They are non-trivial, i.e., the number of requests is big enough to allow the spending of an important part of the entire budgets. Otherwise, a greedy algorithm is the best, as it can always choose the best bid without violating any constraints.

4.1. Sensitivity analysis

The same problem instance is used for the tests: it has 3 buyers (\( N = 3 \)), 8 items (\( M = 8 \)), and 300 requests (\( T = 300 \)). As the sequence of requests follows a multinomial law, the number of items of each type is different from one draw to another one. We generate and solve this problem instance 500 times to obtain good averages for each output of interest. The probabilities \( p_{k} \) and the number of requests \( T \) are supposed to be known.
4.1.1. Analysis of parameters

Table 5 illustrates the evolution of the competitive ratios for different values of the pair \((\Delta, \rho)\). The tests have been performed for the re-optimized primal-dual Algorithm 2.

<table>
<thead>
<tr>
<th>(\Delta)</th>
<th>1</th>
<th>3</th>
<th>6</th>
<th>15</th>
<th>30</th>
<th>60</th>
<th>150</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.995</td>
<td>0.989</td>
<td>0.989</td>
<td>0.978</td>
<td>0.972</td>
<td>0.971</td>
<td>0.965</td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.996</td>
<td>0.994</td>
<td>0.990</td>
<td>0.982</td>
<td>0.977</td>
<td>0.972</td>
<td>0.969</td>
<td>0.966</td>
</tr>
<tr>
<td>0.1</td>
<td>0.996</td>
<td>0.994</td>
<td>0.990</td>
<td>0.984</td>
<td>0.980</td>
<td>0.972</td>
<td>0.969</td>
<td>0.966</td>
</tr>
<tr>
<td>0.2</td>
<td>0.996</td>
<td>0.994</td>
<td>0.991</td>
<td>0.986</td>
<td>0.981</td>
<td>0.974</td>
<td>0.968</td>
<td>0.964</td>
</tr>
<tr>
<td>0.413</td>
<td>0.996</td>
<td>0.994</td>
<td>0.991</td>
<td>0.985</td>
<td>0.981</td>
<td>0.973</td>
<td>0.968</td>
<td>0.964</td>
</tr>
<tr>
<td>0.6</td>
<td>0.996</td>
<td>0.995</td>
<td>0.992</td>
<td>0.986</td>
<td>0.980</td>
<td>0.972</td>
<td>0.966</td>
<td>0.961</td>
</tr>
</tbody>
</table>

Table 5: Tuning of the parameters \(\Delta\) and \(\rho\)

Table 5 shows that the competitive ratios tend to be very sensible to the number of re-optimizations made. The absence of re-optimization deteriorates the solutions up to 3%. The parameter \(\rho\) is less important for the quality of the competitive ratio: \(\rho = 0.2\) appears to be a good value according to those results. Indeed, for each value of the parameter \(\rho\), the gap between the best (bold face) and the worst values remains less than 0.5%. This value (0.2) is kept for \(\rho\) and the behavior of the re-optimized primal-dual algorithm as a function of the number of re-optimizations is presented in the next paragraphs.

4.1.2. Trade-off between computational time and competitive ratio

Computational time is an important criteria for online optimization. The greedy and primal-dual algorithms need about one millisecond (ms) to solve 300 requests. The stochastic primal-dual procedure (Algorithm 1) need on average 1400 ms for the 300 requests, implementing a special case of the re-optimized primal-dual with \(\Delta = 1\) and \(\rho = 0\).
Figure 2 illustrates the evolution of the computational time as a function of the number of re-optimizations \( (\Delta = \lceil \frac{T}{\Delta} \rceil) \). The results are intuitive as the computational time depends linearly on the number of re-optimizations. At the same time, the second figure shows a fast increase of the re-optimized primal-dual competitive ratio before leveling off. It proves that this algorithm performs well with a small number of re-optimizations. \( \Delta = 30 \) (10 re-optimizations) is used in the next comparisons, as the re-optimized primal-dual procedures keeps a good competitive ratio (0.981) while the computational time stays at 50 ms on average. The parameter \( \Delta \) has to be chosen according to the computational time and computing resources available in reality.

### 4.1.3. Comparison between the re-optimized primal-dual and Ciocan’s algorithms

The re-optimized primal-dual algorithm is compared to Ciocan’s algorithm (Ciocan and Farias, 2012). Both algorithms approximately have the same computational time as a function of the number of re-optimizations.

<table>
<thead>
<tr>
<th>Number of re-optimizations</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>300</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Re-optimized primal</td>
<td>0.964</td>
<td>0.968</td>
<td>0.974</td>
<td>0.981</td>
<td>0.986</td>
<td>0.991</td>
<td>0.994</td>
<td>0.996</td>
<td>( T_j ) and ( p_k )</td>
</tr>
<tr>
<td>Ciocan’s method</td>
<td>0.977</td>
<td>0.984</td>
<td>0.99</td>
<td>0.993</td>
<td>0.994</td>
<td>0.995</td>
<td>0.994</td>
<td>0.996</td>
<td>known</td>
</tr>
<tr>
<td>Re-optimized primal</td>
<td>0.971</td>
<td>0.974</td>
<td>0.978</td>
<td>0.978</td>
<td>0.979</td>
<td>0.981</td>
<td>0.982</td>
<td>0.983</td>
<td>( T_j ) and ( p_k )</td>
</tr>
<tr>
<td>Ciocan’s method</td>
<td>0.962</td>
<td>0.948</td>
<td>0.957</td>
<td>0.966</td>
<td>0.972</td>
<td>0.976</td>
<td>0.977</td>
<td>0.977</td>
<td>inferred</td>
</tr>
</tbody>
</table>

**Table 6: Comparison between two algorithms**

Table 6 evaluates two policies: one ideal when parameters are known and one more realistic where parameters are inferred. These two algorithms tend to react in the same way when the number of re-optimizations rises as they both increase. Furthermore, it is clear that Ciocan’s algorithm, which relies on a primal formulation, is more efficient for the first policy while the re-optimized primal-dual algorithm, which relies on a dual formulation, perform better for the second policy. It shows that Ciocan’s procedure needs to know the probabilities \( p_k \) as well as the number of requests left \( T_j \). The strength of the re-optimized primal-dual algorithm is to work well without these parameters. The probabilities \( p_k \) are easily inferred by machine learning tools. However the estimation of \( T_j \) remains a key problem. The sensitivity of the optimization formulation (4) is studied in the next paragraph. Finally, we note that 10 re-optimizations (\( \Delta = 30 \)) remains a good parameter for this last tests.

### 4.1.4. Competitive ratios as a function of the number \( T_j \) of requests left

Let consider that the probabilities \( p_k \) and the number of requests \( T_j \) are now unknown. The probabilities \( p_k \) are estimated by Bayesian inference, as shown in Section 3.2, and \( T_j \) is inferred using the optimization formulation (4). Constraints (4b) \( (\sum_{i=1}^{N} [\sum_{l=1}^{j-1} c_{ik} x_{il} + \sum_{k=1}^{M} c_{ik} y_{ik}] \geq \sum_{i=1}^{N} B_i) \) are slightly modified in order to analyze the impact of the inferred value \( T_j \). They are replaced by \( (\sum_{i=1}^{N} [\sum_{l=1}^{j-1} c_{ik} x_{il} + \sum_{k=1}^{M} c_{ik} y_{ik}] \geq \epsilon \sum_{i=1}^{N} B_i) \), where \( \epsilon \) is a parameter.

Figure 3 presents the evolution of the competitive ratio as a function of \( \epsilon \) (for the same instance as before) and the graph seems to be intuitive. First, for \( \epsilon \) close to 0, the stochastic primal-dual algorithm has the same behavior as the greedy. Indeed, with \( \epsilon = 0 \), \( T_j \) is also
equal to 0 and then the dual variables \( r_i \) are null. Second, if \( \epsilon \) is too big, the algorithm gives too much weight to the future. It waits to maximize revenue with future requests, thus decreasing its performances. In this case, the dual variables \( r_i \) are close to 1. Also it seems better to underestimate the number of requests left at the beginning and overestimate it at the end. At the beginning, important bids are chosen (underestimate \( T_j \)), but later, the algorithm should be more careful (overestimate \( T_j \)), as bad decisions could be costly and not easy to compensate. The optimization model (4) follows this evolution, and the parameter \( \epsilon \) is then set on 0.8. The parameter \( \epsilon \) is linked to the instance, it should be chosen carefully depending on the environment.

4.2. Average competitive ratio

The following tests present some results for the greedy algorithm, the primal-dual algorithm, the re-optimized primal-dual algorithm (Algorithm 2) and Ciocan’s algorithm (Ciocan and Farias, 2012). We compare these algorithms on different instances with their average competitive ratios computed over 500 draws. Table 7 describes six instances generated following different criteria: the number of buyers, items, and requests, the variance of the distribution, and the gap between the bids \( c_{ik} \). These instances also have a reasonable gap between the capacity amounts \( d_{ik} \).

Let define the parameter \( \delta = \frac{\Delta}{T} \) as the proportion of reoptimizations in Algorithm 2: as this ratio is independent of the instance, it will be used to fix the number of reoptimizations for any instance. For all the next tests, \( \rho = 0.2 \), \( \epsilon = 0.8 \), and the probabilities \( p_k \) and the number of requests left are inferred. Tables 8-11 present the results of the comparison in 4 different situations. The “Gain against
greedy" is defined as $\frac{c_{\text{re-} \text{opt}} - c_{\text{greedy}}}{c_{\text{greedy}}}$, where $c_{\text{re-} \text{opt}}$ and $c_{\text{greedy}}$ are the expected competitive ratios of the re-optimized primal-dual and greedy algorithms. “Gain against primal-dual” and “Gain against Ciocan” are similarly defined.

4.2.1. The AdWords problem case

Table 8 presents the average competitive ratios of the re-optimized primal-dual procedure for the AdWords problem: these ratios are very good and always over 0.96. The gains are also always positive except for the instance 5 against Ciocan’s algorithm. As shown in Section 4.1.3, the re-optimized primal-dual algorithm has a better behavior than Ciocan’s algorithm when parameters $p_k$ and $T$ are unknown. Furthermore, the re-optimized primal-dual procedure is an hybrid method taking advantage of the strengths of the primal-dual algorithm and of the stochastic optimization. The primal-dual algorithm is $1 - \frac{1}{e}$-competitive, while the re-optimizations improve this ratio with stochastic optimization but might sometimes worsen it.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Average competitive ratio</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average competitive ratio</td>
<td>0.978</td>
<td>0.986</td>
<td>0.977</td>
<td>0.999</td>
<td>0.962</td>
<td>0.994</td>
<td></td>
</tr>
<tr>
<td>Gain against greedy (%)</td>
<td>8.3</td>
<td>2.8</td>
<td>2.6</td>
<td>0.7</td>
<td>0.5</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>Gain against primal-dual (%)</td>
<td>2.3</td>
<td>0.0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Gain against Ciocan (%)</td>
<td>1.3</td>
<td>3.5</td>
<td>1.3</td>
<td>1.3</td>
<td>-0.2</td>
<td>1.8</td>
<td></td>
</tr>
</tbody>
</table>

Table 8: Results for the Adwords problem

4.2.2. The general problem case

In the general problem case, the dual variables updates of Jaillet and Lu (2011) are used for the primal-dual and for the re-optimized primal-dual algorithms. The results presented in Table 9 are also good in this case. However, the re-optimized primal-dual algorithm is not always the best (especially for the instance 2), but the gains remain close to 0 when negative. The re-optimized primal-dual algorithm outperforms the greedy and the primal-dual algorithms, and obtain approximately the same results as those from Ciocan’s algorithm.
### 4.2.3. The daily AdWords problem case

The re-optimized primal-dual procedure is also tested on the daily AdWords problem. Instead of re-optimizing every $\Delta$ requests, re-optimizations are performed every two hours (12 re-optimizations). The number of requests $T$ is now drawn from an exponential distribution as in (Jaillet and Lu, 2012). However, the number of request left $T_j$ need to be stabilized as the variance of an exponential law is large. Let $T_j^{\text{exp}}$ be the number drawn from the exponential distribution and $T_j^{\text{opt}}$ the value of the solution obtained with the optimization model (4). As $T_j^{\text{opt}}$ is more stable, we define $T_j$ as the mean of $T_j^{\text{exp}}$ and $T_j^{\text{opt}}$.

<table>
<thead>
<tr>
<th>Instance</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average competitive ratio</td>
<td>0.985</td>
<td>0.889</td>
<td>0.968</td>
<td>0.977</td>
<td>0.992</td>
<td>0.994</td>
</tr>
<tr>
<td>Gain against greedy (%)</td>
<td>10.2</td>
<td>11.3</td>
<td>5.9</td>
<td>9.5</td>
<td>-0.4</td>
<td>-0.5</td>
</tr>
<tr>
<td>Gain against primal-dual (%)</td>
<td>3.4</td>
<td>6.0</td>
<td>4.4</td>
<td>8.7</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>Gain against Ciocan (%)</td>
<td>0.2</td>
<td>-1.7</td>
<td>-0.8</td>
<td>1.9</td>
<td>1.4</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Table 9: Results for the general problem

Table 10 shows that the re-optimized primal-dual algorithm has the best results with high average competitive ratios. It proves that this algorithm performs even better when the total number of requests $T$ varies. Indeed, the re-optimizations always compute good updates, as the dual slave problems (presented in Table 2) will remain feasible whenever $T$ changes.

### 4.2.4. The daily general problem case

In this paragraph, we propose to solve the general problem over a day with a daily budget for each buyer.

Table 11 shows that the gains remain big for most of the instances with the exception of instances 5 and 6. Although the greedy algorithm beats the re-optimized primal-dual algorithm by a small percentage, the average competitive ratio remains high. Finally, the re-optimized primal-dual procedure still outperforms Ciocan’s algorithm.

<table>
<thead>
<tr>
<th>Instance</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average number of requests</td>
<td>359</td>
<td>480</td>
<td>394</td>
<td>313</td>
<td>456</td>
<td>311</td>
</tr>
<tr>
<td>Average competitive ratio</td>
<td>0.984</td>
<td>0.994</td>
<td>0.989</td>
<td>0.999</td>
<td>0.969</td>
<td>0.994</td>
</tr>
<tr>
<td>Gain against greedy (%)</td>
<td>8.7</td>
<td>3.4</td>
<td>3.9</td>
<td>0.8</td>
<td>1.4</td>
<td>1.2</td>
</tr>
<tr>
<td>Gain against primal-dual (%)</td>
<td>0.8</td>
<td>1.1</td>
<td>1.4</td>
<td>0.2</td>
<td>1.3</td>
<td>0.3</td>
</tr>
<tr>
<td>Gain against Ciocan (%)</td>
<td>1.0</td>
<td>0.6</td>
<td>0.4</td>
<td>1.2</td>
<td>2.8</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Table 10: Results for the daily Adwords problem

Table 10 shows that the re-optimized primal-dual algorithm has the best results with high average competitive ratios. It proves that this algorithm performs even better when the total number of requests $T$ varies. Indeed, the re-optimizations always compute good updates, as the dual slave problems (presented in Table 2) will remain feasible whenever $T$ changes.

### 4.2.4. The daily general problem case

In this paragraph, we propose to solve the general problem over a day with a daily budget for each buyer.

Table 11 shows that the gains remain big for most of the instances with the exception of instances 5 and 6. Although the greedy algorithm beats the re-optimized primal-dual algorithm by a small percentage, the average competitive ratio remains high. Finally, the re-optimized primal-dual procedure still outperforms Ciocan’s algorithm.

### 5. Conclusions

In this paper, we consider an online bipartite allocation problem with budget and resource constraints. The main goal is to improve current online algorithms with additional
### Table 11: Results for the daily general problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average number of requests</td>
<td>360</td>
<td>480</td>
<td>396</td>
<td>312</td>
<td>456</td>
<td>312</td>
</tr>
<tr>
<td>Average competitive ratio</td>
<td>0.982</td>
<td>0.921</td>
<td>0.977</td>
<td>0.974</td>
<td>0.987</td>
<td>0.991</td>
</tr>
<tr>
<td>Gain against greedy (%)</td>
<td>9.7</td>
<td>15.0</td>
<td>6.5</td>
<td>9.9</td>
<td>-0.9</td>
<td>-0.8</td>
</tr>
<tr>
<td>Gain against primal-dual (%)</td>
<td>3.3</td>
<td>8.2</td>
<td>5.2</td>
<td>9.1</td>
<td>0.4</td>
<td>-0.1</td>
</tr>
<tr>
<td>Gain against Ciocan (%)</td>
<td>0.2</td>
<td>0.3</td>
<td>-0.1</td>
<td>1.7</td>
<td>0.9</td>
<td>0.7</td>
</tr>
</tbody>
</table>

information which can become available during the procedure. We propose to re-optimize the primal-dual algorithm in order to make use of this information. At each re-optimization, we generate, based on the current historical data, a random scenario that represents all the future requests which will arrive until the end of the process, and then we seek the optimal solution for such a scenario. This stochastic problem is solved with the L-Shaped method: the dual solution of the subproblems of the Benders decomposition updates the dual variables of the primal-dual algorithm.

This new procedure gives very good results and outperforms the greedy and the primal-dual algorithms on the Adwords and the general bipartite resource allocation problems. The results also show that the re-optimized primal-dual procedure is generally better than Ciocan’s algorithm. Furthermore, the practical modifications allow us to solve some instances in a realistic framework: the computational time remains reasonable, the learning process estimates the distribution of the items type from the current historical data, and the inference of the number of requests left is efficient and can be easily changed according to the framework.

**Acknowledgments**

Research funded in part by NSF (grant 1029603).


