A Stochastic Algorithm for Online Bipartite Resource Allocation Problems

Antoine Legrain[a], Patrick Jaillet[b]

[a] *CIRRELT and Polytechnique Montréal, Department of Mathematical and Industrial Engineering, C.P. 6079, Succursale Centre-ville, Montréal, QC, Canada, H3C 3A7*
[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 which become available one at a time and which consume some amount of various limited resources upon allocation. A central resource allocation platform is in charge of allocating the items to the potential buyers, with the goal of maximizing the total revenue subject to budget and resource constraints. Sponsored search advertising is a typical example: in order to maximize revenue, search engines try to choose the best available advertisement to display on a web page resulting from a search query.

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 model); the other one, based on specific probabilistic assumptions about the input, is concerned with expected performance guarantee. However, combining the strengths of these two approaches could potentially outperform both in some settings. In this paper we propose such a practical method which goes beyond the adversarial model but requires only a limited amount of information about the future. We provide extensive computational results which demonstrate settings under which the performance of the proposed algorithm becomes attractive.

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.

Email addresses: antoine.legrain@polymtl.ca (Antoine Legrain), jaillet@mit.edu (Patrick Jaillet)

For such online optimization problems, detailed information (about users, clients, and/or advertisements) is 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 online advertising example: many advertisers compete to display their advertisements on a given web page which would result from a search query within a search engine, or which would simply be known to attract traffic from internet users. An online advertising service aims at maximizing its revenue by choosing the best possible advertisement to display on a web page (i.e. an impression). For each impression, only one among many advertisements is displayed, thus generating a profit for the advertising service according to the advertisement chosen. The goal is to maximize the total revenue of all allocations without knowing the whole sequence of requests for displaying an advertisement. There are also variations of this problem which consider the possibilities of bounding the number of allocations to a client or a web page, limiting clients budget, displaying several advertisements on a web page, and/or managing different advertisements dimensions. In the remainder of this paper, we will refer to this class of problems as online bipartite resource allocation problems.

A wide range of other applications can be modeled by this class of problems, such as in routing, revenue management, and scheduling. For example, Google has published two anonymous sets of real-data about their computer 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, bandwidth and the number of CPUs, and, second, software constraints ensuring the right configuration of virtual machines. Online bipartite resource allocation problems also appear in the hotel, rail, and airline industries, to name a few. In these settings, clients typically purchase a service from a company which must price it beforehand according to the current availability of its key resource (hotel rooms, train seats, or plane seats, respectively). Other applications of interest 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 balance resources among different categories of items and also infer future resource needs to avoid under- or over-provisioning. In general, it is difficult to design general-purpose algorithms which perform well under all types of 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 competitive analysis approaches (Borodin and El-Yaniv, 1998; Jaillet and Wagner, 2010) which correspond to designing online algorithms with worst-case guarantees against any future requests. In this approach, the competitive ratio of an online algorithm for a maximization problem is defined as $c = \min_{I \in \mathcal{X}} \{\frac{\text{Obj}_{online}(\text{I})}{\text{Obj}_{optimal}(\text{I})}\}$, where $\text{Obj}_{online}(\text{I})$ is the value of the objective for the solution given by the online algorithm, $\text{Obj}_{optimal}(\text{I})$ is the value of the objective for the offline solution for an instance I, and $\mathcal{X}$ is the set of all possible instances. We then say that the online algorithm is c-competitive. Such an approach about handling uncertainty is often referred to as an adversarial model: the online algorithm designer is facing an all powerful adversary who can choose any

specific instance and can solve the offline problem on that instance.

Different versions of the online bipartite resource allocation problems have been studied under the adversarial model when the total number of requests in the instance is known to the online algorithm designer. Karp et al. (1990) deal with a simple form of this problem maximizing the number of requests matched to buyers. The author 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 an online bipartite resource allocation problem where each buyer can be matched at most b times. They also prove that this competitive ratio is the best possible. Mehta et al. (2007) introduce the Adwords problem 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 online bipartite resource allocation problem in a special homogeneous case. Aggarwal et al. (2011) give a $1 - \frac{1}{e}$-competitive algorithm for the vertex-weighted bipartite matching problem.

As the adversarial model may be too conservative, other models have been proposed. The random permutation model allows some mild assumptions on the request arrival process. The set of requests remains unknown, but the order of the sequence of requests is random instead of being chosen by an adversary. Goel and Mehta (2008) prove that a greedy algorithm is a $1 - \frac{1}{e}$-competitive algorithm for the AdWords problem. Devanur et al. (2011); Agrawal et al. (2014); Eghbali et al. (2014); Molinaro and Ravi (2014); Kesselheim et al. (2014)) all propose near-optimal algorithms for various settings of the online bipartite resource allocation and online packing problems under the random permutation model.

Other authors (Feldman et al. (2009); Karande et al. (2011); Manshadi et al. (2012); Jaillet and Lu (2014)) combine online and stochastic ideas for applications where statistics on the probability distribution of the future requests is either known or learnable. They study the online bipartite matching problem under such a probabilistic model and improve the bounds on the competitive ratio from $1 - \frac{1}{e}$ for the adversarial model up to 0.706 for the probabilistic model (Jaillet and Lu, 2014), using statistics about offline strategies. These papers assume that the requests are independent and identically distributed from a known probability distribution.

Other papers present algorithms for solving the online bipartite resource allocation problems with adaptive updates of stochastic information, when an offline linear programming problem is used to update the future strategy. For example, Ciocan and Farias (2012) have obtained an expected worst case guarantee of 0.342 with the following procedure. Based on known statistics of the distribution, 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 dual solutions to improve current strategy. Jaillet and Lu (2012) also try to infer the total number of requests T (instead of assuming to be known ahead of time).

Finally, Van Hentenryck and Bent (2009) develop practical algorithms for online stochastic combinatorial optimization. They 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 scenarios. There is no assumption made on the distribution which can dynamically evolve during the resolution: these algorithms just need to retrieve a sampled scenarios set upon each request arrival. Although these architectures propose efficient ways to solve general online allocation problems, they do not provide any competitive ratio against any model.

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 paper is not about deriving new competitive ratios under any classical (adversarial, random permutation, or probabilistic) models as described above. Our work builds upon the primal-dual algorithm presented in Buchbinder et al. (2007) and the online stochastic algorithms proposed by Van Hentenryck and Bent (2009). We design a new data-driven algorithm taking advantages of any probabilistic underlying structures, which are more common with the rise of the big data era. 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 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 of items based on historical data. Upon each arrival of a request, we then use an optimization problem to estimate 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 an empirical competitive ratio above 0.9.

The rest of this paper is organized as follows. Section 2 mathematically defines the online bipartite resource allocation problem. Section 3 introduces the stochastic online algorithm. Section 4 presents different modifications on the algorithm to solve more realistic problems. Section 5 provides extensive numerical results. Finally, conclusions are drawn in Section 6.

2. Online bipartite resource allocation problem

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, the online bipartite resource allocation problem can be described as follows:

- each buyer in a set $\{1 \dots N\}$ is interested in purchasing one or more items from a set $\{1 \dots M\}$ of distinct object types;
- a total of T items come one at a time (T requests);
- buyer $i \in \{1 \dots N\}$ is willing to pay c_{ik} for each item of type $k \in \{1 \dots M\}$ 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 $\boldsymbol{F_k} \in \mathbb{R}_+^L$ amounts of these L distinct resources are available for the overall allocations of items of type $k \in \{1 \dots M\}$;
  - each item of type $k \in \{1 \dots M\}$ consumes amounts $\boldsymbol{d_{ik}} \in \mathbb{R}_+^L$ of resources $\boldsymbol{F_k}$ and provides a revenue c_{ik} to the operator when allocated to buyer $i \in \{1 \dots N\}$;
- 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.

All mathematical notations are summarized in Appendix A. A mathematical programming formulation corresponding to this problem can be written as follows:

$$\max \sum_{i=1}^{N} \sum_{j=1}^{T} c_{ik_j} x_{ij} \tag{1a}$$

subject to:

$$\sum_{i=1}^{N} x_{ij} \leq 1 \qquad \forall j = 1 \dots T \tag{1b}$$

$$\sum_{j=1}^{T} c_{ik_j} x_{ij} \leq B_i \qquad \forall i = 1 \dots N \tag{1c}$$

$$\sum_{i=1}^{N} \sum_{j=1|k_j=k}^{T} \boldsymbol{d_{ik}} x_{ij} \leq \boldsymbol{F_k} \qquad \forall k = 1 \dots M \tag{1d}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i = 1 \dots N, \forall j = 1 \dots T \tag{1e}$$

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, for example the available space on a web page or the maximum number of allocations to a web page. In the online case, the type k_j of the jth 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 the fraction of requests of type k, $(\frac{n_k}{T})_k$, 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.

The different resource allocation problems presented in the introduction are special cases of the online bipartite resource allocation problem defined above. The matching problem corresponds to the case without constraints (1d) and with $B_i = 1$, $c_{ik} = 1$. The b-matching problem corresponds to the case without constraints (1d) and with $B_i = b$, $c_{ik} = 1$. The AdWords problem corresponds to the case without constraints (1d).

3. Stochastic algorithm

In this section, we present a stochastic optimization formulation and an algorithm to solve the online bipartite resource allocation problem (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_j^k): $X_j^k = 1$ if the jth 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 $(X_j^k)_{j=1}^T$.

From this information, those next parameters can be computed:

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

An important point to note is that the specific order in the sequence of the future requests associated with a given scenario does not matter when solving a whole scenario with an offline algorithm, as decisions on how to allocate them under this particular scenario would be done with the full knowledge of the scenario, and thus all at once.

3.1. 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 jth 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 scenarios $\omega \in \Omega_j$. 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 jth request is allocated to the buyer i, 0 otherwise
- y_{ik}^ω is the number of items of type k allocated to the buyer i for the scenario ω.

At the time of the jth request, we obtain the following formulation:

$$\max \sum_{i=1}^{N} c_{ik_j} x_i + \sum_{\omega \in \Omega_j} p^\omega \sum_{i=1}^{N} \sum_{k=1}^{M} c_{ik} y_{ik}^\omega \tag{2a}$$

subject to:

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

$$\sum_{i=1}^{N} y_{ik}^\omega \leq T_{jk}^\omega \qquad \forall \omega \in \Omega_j, \forall k = 1 \dots M \tag{2c}$$

$$c_{ik_j} x_i + \sum_{k=1}^{M} c_{ik} y_{ik}^\omega \leq B_i^{left} \qquad \forall \omega \in \Omega_j, \forall i = 1 \dots N \tag{2d}$$

$$x_i \in \{0, 1\}, y_{ik}^\omega \in \mathbb{N} \qquad \forall \omega \in \Omega_j, \forall i = 1 \dots N, \forall k = 1 \dots M \tag{2e}$$

The objective (2a) maximizes the revenue for the jth request and the expected revenue of remaining future requests. The constraint (2b) ensures that the jth request is allocated to a maximum of one buyer. The number of items of type k allocated to all buyers for the scenario ω is bounded by T_{jk}^ω in constraints (2c). Finally, the constraints (2d) prevent exceeding the budget for each buyer and each scenario. The formulation (2) is the simplest to consider in order 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}^ω is a first way to simplify the model. The L-Shaped method, presented in the next section, is a second way to improve the computational time.

3.2. 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 problem 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.

The recourse function Q has to be computed for each value of the variable $\boldsymbol{x}$ and for each scenario ω. 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.

Note that the weak duality theorem justifies these approximations:

$$\forall \boldsymbol{x} \in [0, 1]^N, \omega \in \Omega_j,\ Q(\boldsymbol{x}, \omega) \leq \sum_{k=1}^{M} T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^{N} (B_i^{left} - c_{ik_j} x_i) \beta_i^\omega \tag{3}$$

Master Problem	Slave Problems
$\max \sum_{i=1}^{N} c_{ik_j} x_i + \sum_{\omega \in \Omega_j} p^\omega Q(\boldsymbol{x}, \omega)$	$Q(\boldsymbol{x}, \omega) = \max \sum_{i=1}^{N} \sum_{k=1}^{M} c_{ik} y_{ik}^\omega$
subject to:	subject to:
$\sum_{i=1}^{N} x_i \leq 1$	$\sum_{i=1}^{N} y_{ik}^\omega \leq T_{jk}^\omega \quad \forall k = 1 \dots M$
	$\sum_{k=1}^{M} c_{ik} y_{ik}^\omega \leq B_i^{left} - c_{ik_j} x_i \quad \forall i = 1 \dots N$
$x_i \in \{0, 1\} \quad \forall i = 1 \dots N$	$y_{ik}^\omega \in \mathbb{N} \quad \forall i = 1 \dots N, \forall k = 1 \dots M$

Table 1: Decomposition in sub-problems

Master Problem	Dual Slave Problems
$\max \sum_{i=1}^{N} c_{ik_j} x_i + \sum_{\omega \in \Omega_j} p^\omega \theta^\omega$	$\min \sum_{k=1}^{M} T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^{N} (B_i^{left} - c_{ik_j} x_i) \beta_i^\omega$
subject to:	subject to:
$\sum_{i=1}^{N} x_i \leq 1$	$\alpha_k^\omega + c_{ik} \beta_i^\omega \geq c_{ik} \quad \forall i = 1 \dots N, \forall k = 1 \dots M$
$\theta^\omega \leq \sum_{k=1}^{M} T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^{N} (B_i^{left} - c_{ik_j} x_i) \beta_i^\omega \quad \forall \omega \in \Omega_j$	
$x_i \in \{0, 1\} \quad \forall i = 1 \dots N$	$\alpha_k^\omega,\ \beta_i^\omega \geq 0 \quad \forall i = 1 \dots N, \forall k = 1 \dots M$

Table 2: Benders decomposition

1. Set $\boldsymbol{x} = 0$
2. Solve all the dual slave problems and add every cut to the master problem
3. Solve the master problem:
   - if the solution $\boldsymbol{x}$ 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 are 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 3.3.

For each request, Algorithm 1 chooses first the buyer that 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 5.

Algorithm 1 Re-optimized primal-dual Algorithm with continuous reviews

$x_{ij} = 0$
for each jth request **do**
  Use a greedy algorithm to set $\boldsymbol{x}$
  Solve all the dual slave problems and **add** every cut to the master problem
  Solve the master problem and keep this solution $\boldsymbol{x}$
end for

3.3. Links with the primal-dual

The inequalities (3) become equalities in the master problem; indeed, as this is a maximization problem, each variable θ^ω 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:

$$\forall \boldsymbol{x} \in [0,1]^N, \omega \in \Omega_j,\ \theta^\omega = \sum_{k=1}^{M} T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^{N} (B_i^{left} - c_{ik_j} x_i)\beta_i^\omega$$

With all those equalities, the variables θ^ω can be eliminated from the formulation. The updated cuts can thus be integrated in the objective:

$$\sum_{i=1}^{N} c_{ik_j} x_i + \sum_{\omega \in \Omega_j} p^\omega [\sum_{k=1}^{M} T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^{N} (B_i^{left} - c_{ik_j} x_i)\beta_i^\omega]$$

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

- c_{ik_j} is the benefit that the operator earns immediately if the jth request is allocated to the buyer i;
- $c_{ik_j} r_i$ is the expected loss if such an allocation is chosen (the future budget B_i^{left} will be reduced by c_{ik_j}).

The re-optimized primal-dual algorithm with continuous reviews is just seeking an equilibrium between the instant revenue c_{ik_j} and the expected loss.

3.4. Generalized problems

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

$$\max \ \sum_{i=1}^{N} \{c_{ik_j} - [\sum_{\omega \in \Omega_j} p^\omega (c_{ik_j}\beta_i^\omega + (\boldsymbol{d_{ik_j}})^T \boldsymbol{\gamma_{k_j}^\omega})]\} x_i$$

subject to:

$$\sum_{i=1}^{N} x_i \leq 1$$
$$x_i \in \{0,1\} \qquad \forall i = 1 \dots N$$

with the dual slave problems:

$$\min \sum_{k=1}^{M} T_{jk}^{\omega} \alpha_k^{\omega} + \sum_{i=1}^{N} [(B_i^{left} - c_{ik_j} x_i)\beta_i^{\omega} - (\boldsymbol{d_{ik_j}})^T \boldsymbol{\gamma_{k_j}^{\omega}}] x_i + \sum_{k=1}^{M} (\boldsymbol{F_k^{left}})^T \boldsymbol{\gamma_k^{\omega}}$$

subject to:

$$\alpha_k^{\omega} + c_{ik}\beta_i^{\omega} + (\boldsymbol{d_{ik}})^T \boldsymbol{\gamma_k^{\omega}} \geq c_{ik}, \qquad \forall i = 1 \dots N, \forall k = 1 \dots M$$
$$\alpha_k^{\omega},\ \beta_i^{\omega},\ \gamma_k^{\omega} \geq 0, \qquad \forall i = 1 \dots N, \forall k = 1 \dots M, \forall k = 1 \dots L$$

The slave problems have more variables and the dual variables r_i are now equal to $\sum_{\omega \in \Omega_j} p^{\omega}(\beta_i^{\omega} + \frac{1}{c_{ik_j}}(\boldsymbol{d_{ik_j}})^T \boldsymbol{\gamma_{k_j}^{\omega}})$. This new algorithm follows the same procedure as before.

3.5. Re-optimized primal-dual

On average, Algorithm 1 should improve on 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 Δ-re-optimized algorithm, for which the linear problem will be solve only each Δ requests.

Algorithm 2 Δ-re-optimized primal-dual Algorithm

$x_{ij} = 0, r_i = 0$
for each jth request **do**
  if $j \equiv 0$ (**mod** Δ) **then**
    MAKE one step of the re-optimized primal-dual Algorithm with continuous reviews 1
    UPDATE $r_i = \rho r_i + (1-\rho)[\sum_{\omega \in \Omega_j} p^{\omega} \beta_i^{\omega}], \qquad \forall i = 1 \dots N$
  end if
  FIND a buyer i who **MAXIMIZES** $c_{ik_j}(1 - r_i)$ **such that:** $r_i < 1$ AND there is enough budget B_i left
  SET $x_{ij} = 1$
  UPDATE $r_i = r_i(1 + \frac{c_{ik_j}}{B_i}) + \frac{c_{ik_j}}{(c-1)B_i}$ if a re-optimization has not been made
end for

Most of the time, Algorithm 2 use the same updates as the primal-dual algorithm. However, each Δ 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 similar to Algorithm 1. Furthermore, a parameter ρ will allow us to smooth the value of the dual variables at each re-optimization. Results are shown in Section 5. In the next section, we propose additional modifications to use this algorithm in specific practical settings of interest.

4. Practical modifications

In this section, we propose some ideas to transform Algorithm 2 and adapt it to make it more practical to use in realistic situations.

4.1. Improvement of the computational time

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

$\lvert\Omega_j\rvert$ =	1	2	3	5	10
Competitive ratio	0.995	0.995	0.996	0.996	0.996

Table 3: Empirical competitive ratio as a function of $|\Omega_j|$ for the re-optimized primal-dual algorithm with continuous reviews

As the empirical competitive ratio does not really increase with the size of the sample set Ω_j, we propose to solve the dual slave problems (Table 2) for only one random scenario ω_0. Indeed, the cuts (3) $\theta^\omega \leq \sum_{k=1}^{M} T_{jk}^{\omega} \alpha_k^{\omega_0} + \sum_{i=1}^{N} [(B_i^{left} - c_{ik_j} x_i)\beta_i^{\omega_0} - (\boldsymbol{d_{ik_j}})^T \boldsymbol{\gamma}_{\boldsymbol{k_j}}^{\boldsymbol{\omega_0}} x_i] + \sum_{k=1}^{M} (\boldsymbol{F_k^{left}})^T \boldsymbol{\gamma}_{\boldsymbol{k}}^{\boldsymbol{\omega_0}}$ remain valid inequalities for each scenario ω. As the constraints of the dual slave problems are independent of the scenarios, the dual solution for the scenario ω_0 remains feasible for every scenario, and thus the cuts always hold.

Furthermore, the generation of one random scenario ω_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 scenario, the objective function now becomes $\sum_{i=1}^{N} \{c_{ik_j} - (c_{ik_j}\beta_i^{\omega_0} + (\boldsymbol{d_{ik_j}})^T \boldsymbol{\gamma}_{\boldsymbol{k_j}}^{\boldsymbol{\omega_0}})\} x_i$ and the computational time decreases dramatically.

4.2. Bayesian inference

In practice, it is rare to know the distribution of the stochastic process $(X_j^k)_{j=1}^T$. 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 $(X_j)_{j=1}^T$ is a Bernoulli process: that is, the stochastic process $(X_j)_{j=1}^T$ is i.i.d. and each X_j follows a Bernoulli distribution of mean μ in $[0, 1]$.

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

Assume that μ is unknown, but that it follows a prior distribution $\mathbb{P}[\mu|\alpha]$. Then, it is well-known that $\forall j \in \{1 \dots T\}$, $\mathbb{P}[\mu|(X_l)_{l=1}^j, \alpha]$ 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, $a_j = a + \sum_{l=1}^{j} X_l$ and $b_j = b + \sum_{l=1}^{j} (1 - X_l)$. We can now estimate $\hat{\mu}_{j+1} = \mathbb{P}[X_{j+1} = 1|(X_l)_{l=1}^j, a, b]$:

$$\hat{\mu}_{j+1} = \mathbb{E}[\mu|(X_l)_{l=1}^j, a, b] = \frac{a_j}{a_j + b_j} = \frac{a + \sum_{l=1}^{j} X_l}{a + b + j}$$

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 $(X_j^k)_{j=1}^T$ follows a Bernoulli process for each item k:

$$X_j^k = \begin{cases} 1 & \text{if } k = k_j \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in 1 \dots T,\ \forall k = 1 \dots M$$

Note that $\sum_{k=1}^M X_j^k = 1,\ \forall j = 1 \dots T$. With the same notations as before, we obtain that $\forall j = 1 \dots T, \forall k = 1 \dots M,\ X_j^k$ follows a beta distribution $\beta(a_j^k, b_j^k)$ and that $\hat{\mu_j^k} = \frac{a^k + \sum_{l=1}^j X_l^k}{a^k + b^k + j}$. 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:

$$\begin{aligned} \sum_{k=1}^M \hat{\mu_j^k} &= \sum_{k=1}^M \frac{a + \sum_{l=1}^j X_l^k}{Ma + j} = \frac{Ma + \sum_{l=1}^j \sum_{k=1}^M X_l^k}{Ma + j} \\ &= \frac{Ma + j}{Ma + j} = 1 \quad \forall j = 1 \dots T \end{aligned}$$

Consequently, $\hat{\mu_j^k}$ 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 and sample the scenarios ω.

4.3. 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 empirical competitive ratios obtained using tests with 300 requests, averaged over 500 i.i.d. draws.

	greedy	primal-dual	re-optimized primal-dual		
			real T_j	$T_j = 50$	
mean of the competitive ratio	0.901	0.957	0.995	0.933	real probability
			0.988	0.929	inferred probability

Table 4: Comparison of empirical 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 re-optimized primal-dual algorithm with continuous reviews 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.

$$\min\ T_j \tag{4a}$$

subject to:

$$\sum_{i=1}^{N}[\sum_{l=1}^{j-1} c_{ik_l} x_{il} + \sum_{k=1}^{M} c_{ik} y_{ik}] \geq \sum_{i=1}^{N} B_i \tag{4b}$$

$$\sum_{i=1}^{N} y_{ik} \leq p_k T_j \qquad \forall k = 1 \dots M \tag{4c}$$

$$y_{ik} \geq 0 \qquad \forall i = 1 \dots N,\ \forall k = 1 \dots M \tag{4d}$$

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 Δ-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 5.1.4.

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.

5. 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 $(k_j)_{j=1\dots L}$ 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.

5.1. 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.

		Δ							
		1	*3*	*6*	*15*	*30*	*60*	*150*	*300*
ρ	*0*	0.995	0.989	0.989	0.983	0.978	0.972	**0.971**	0.965
	0.01	**0.996**	0.994	0.990	0.982	0.977	0.972	0.969	**0.966**
	0.1	**0.996**	0.994	0.990	0.984	0.980	0.972	0.969	**0.966**
	0.2	**0.996**	0.994	0.991	**0.986**	**0.981**	**0.974**	0.968	0.964
	0.413	**0.996**	0.994	0.991	0.985	**0.981**	0.973	0.968	0.964
	0.6	**0.996**	**0.995**	**0.992**	**0.986**	0.980	0.972	0.966	0.961

Table 5: Tuning of the parameters Δ and ρ

5.1.1. Analysis of parameters

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

Table 5 shows that the empirical 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 ρ is less important for the quality of the empirical competitive ratio: $\rho = 0.2$ appears to be a good value according to those results. Indeed, for each value of the parameter ρ, the gap between the best (**bold face**) and the worst values remains less than 0.5%. This value (0.2) is kept for ρ and the behavior of the Δ-re-optimized primal-dual algorithm as a function of the number of re-optimizations is presented in the next paragraphs.

5.1.2. Trade-off between computational time and empirical 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 re-optimized primal-dual procedure with continuous reviews (Algorithm 1) needs on average 1400 ms for the 300 requests, implementing a special case of the Δ-re-optimized primal-dual with $\Delta = 1$ and $\rho = 0$.

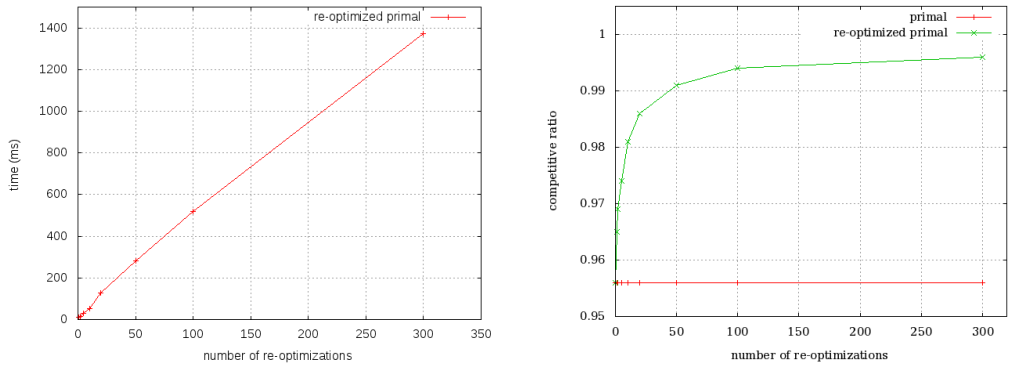

Figure 2: Trade-off between computational time and empirical competitive ratio

Figure 2 illustrates the evolution of the computational time as a function of the number of re-optimizations ($\lfloor \frac{T}{\Delta} \rfloor$). 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 Δ-re-optimized primal-dual empirical 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 Δ-re-optimized primal-dual procedures keeps a good empirical competitive ratio (0.981) while the computational time stays at 50 ms on average. The parameter Δ has to be chosen according to the computational time and computing resources available in reality.

5.1.3. Comparison between the re-optimized primal-dual and Ciocan's algorithms

The Δ-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.

Number of re-optimizations		*1*	*2*	*5*	*10*	*20*	*50*	*100*	*300*	Policy
Competitive ratio	Δ-re-optimized primal	0.964	0.968	0.974	0.981	0.986	0.991	**0.994**	**0.996**	T_j and p_k
	Ciocan's method	**0.977**	**0.984**	**0.99**	**0.993**	**0.994**	**0.995**	**0.994**	**0.996**	known
	Δ-re-optimized primal	**0.971**	**0.974**	**0.978**	**0.978**	**0.979**	**0.981**	**0.982**	**0.983**	T_j and p_k
	Ciocan's method	0.962	0.948	0.957	0.966	0.972	0.976	0.977	0.977	inferred

Table 6: Comparison between two algorithms

Table 6 evaluates two policies: one ideal when the number of requests left T_j and the probabilities p_k are known and one more realistic where these 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 Δ-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 Δ-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 test.

5.1.4. 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 4.2, and T_j is inferred using the optimization formulation (4). Constraints (4b) ($\sum_{i=1}^{N}[\sum_{l=1}^{j-1} c_{ik_l}x_{il} + \sum_{k=1}^{M} c_{ik}y_{ik}] \geq \sum_{i=1}^{N} B_i$) are slighty modified in order to analyze the impact of the inferred value T_j. They are replaced by $\sum_{i=1}^{N}[\sum_{l=1}^{j-1} c_{ik_l}x_{il} + \sum_{k=1}^{M} c_{ik}y_{ik}] \geq \epsilon \sum_{i=1}^{N} B_i$, where ϵ is a parameter.

Figure 3 presents the evolution of the empirical competitive ratio as a function of ϵ (for the same instance as before) and the graph seems to be intuitive. First, for ϵ close to 0, the Δ-re-optimized 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 ϵ 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 ϵ is then set on 0.8. The parameter ϵ is linked to the instance, it should be chosen carefully depending on the environment.

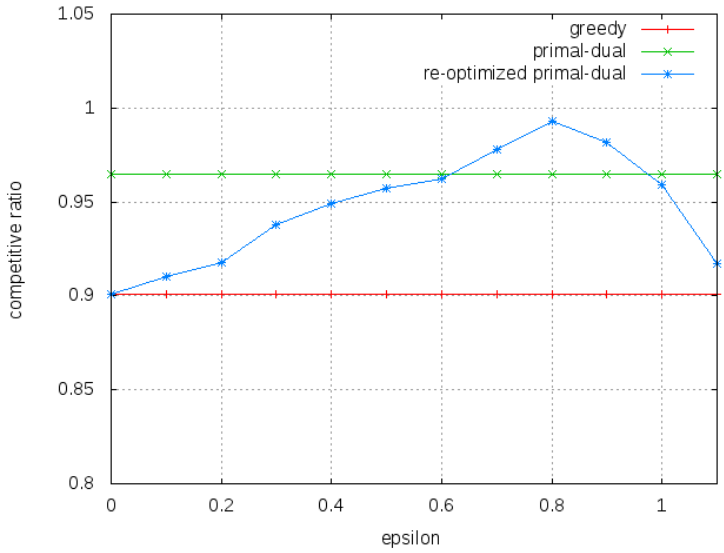

Figure 3: Evolution of the empirical competitive ratio

5.2. *Empirical competitive ratios of various algorithms*

The following tests present some results for the greedy algorithm, the primal-dual algorithm, the Δ-re-optimized primal-dual algorithm (Algorithm 2) and Ciocan's algorithm (Ciocan and Farias, 2012). We compare these algorithms on different instances with their empirical competitive ratios computed over 500 draws. Although the primal-dual algorithm has been designed under the adversarial model, our goal is to use it as a benchmark in order to measure the improvements one can get by doing stochastic updates in the Δ-re-optimized primal-dual algorithm.

Table 7 describes six instances generated following different criteria: the number of buyers, items, and requests, then the coefficient of variations of the distribution and the bids c_{ik}. These instances also have a reasonable coefficient of variation of the capacity amounts $\boldsymbol{d_{ik}}$. The coefficient of variation is considered to be small for the interval $[0.3, 0.5]$, reasonable for $[0.5, 1]$ and huge for $[1, 2]$.

number of the instance	N	M	T	coefficient of variation of	
				the distribution	the bids
1	3	8	300	small	reasonable
2	3	8	500	small	reasonable
3	4	10	400	small	reasonable
4	3	8	300	small	small
5	3	8	450	huge	reasonable
6	3	8	300	huge	reasonable

Table 7: Description of the instances

Let define the parameter $\delta = \frac{\Delta}{T}$ as the proportion of re-optimizations in Algorithm 2: as this ratio is independent of the instance, it will be used to fix the number of re-optimizations 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 9-12 present the results of the comparison in 4 different situations. The "Gain against greedy" is defined as $\frac{c_{re-opt}-c_{greedy}}{c_{greedy}}$, where c_{re-opt} and c_{greedy} are the empirical competitive ratios of the Δ-re-optimized primal-dual and greedy algorithms. "Gain against primal-dual" and "Gain against Ciocan" are similarly defined.

We compare the four previous algorithms on four different cases: the AdWords problem, the general problem, the daily AdWords problem, and finally the daily general problem. These cases represent four different problems which are summarized in Table 8.

Problem case	Horizon		Constraints		
	in requests	in hours	allocation	budget	resource
AdWords	yes	no	yes	yes	no
General	yes	no	yes	yes	yes
Daily AdWords	no	yes	yes	yes	no
Daily general	no	yes	yes	yes	yes

Table 8: Description of the cases

The columns horizon describe for a given instance whether the horizon is counted with a total number of requests T or in hours. The columns constraints list the active constraints in a given problem: the allocation constraints ensure that each request is allocated no more than once, the budget constraints limit the total expense for each buyer, and the resource constraints ensure that the resource consumption does not exceed the available resources.

5.2.1. The AdWords problem case

Table 9 presents the empirical competitive ratios of the Δ-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 5.1.3, the Δ-re-optimized primal-dual algorithm has a better behavior than Ciocan's algorithm when parameters p_k and T are unknown. Furthermore, the Δ-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.

Instance	1	2	3	4	5	6
Empirical competitive ratio	0.978	0.986	0.977	0.999	0.962	0.994
Gain against greedy (%)	**8.3**	**2.8**	**2.6**	**0.7**	**0.5**	**1.2**
Gain against primal-dual (%)	**2.3**	**0.0**	**0.1**	**0.2**	**0.5**	**0.3**
Gain against Ciocan (%)	**1.3**	**3.5**	**1.3**	**1.3**	-0.2	**1.8**

Table 9: Results for the Adwords problem

5.2.2. 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 Δ-re-optimized primal-dual algorithms. Furthermore, only one

resource ($L = 1$) is used in the following instances. The results presented in Table 10 are also good in this case. However, the Δ-re-optimized primal-dual algorithm is not always the best (especially for the instance 2), but the gains remain close to 0 when negative. It also outperforms the greedy and the primal-dual algorithms, and obtain approximately the same results as those from Ciocan's algorithm.

Instance	1	2	3	4	5	6
Empirical competitive ratio	0.985	0.889	0.968	0.977	0.992	0.994
Gain against greedy (%)	**10.2**	**11.3**	**5.9**	**9.5**	-0.4	-0.5
Gain against primal-dual (%)	**3.4**	**6.0**	**4.4**	**8.7**	**0.8**	**0.2**
Gain against Ciocan (%)	**0.2**	-1.7	-0.8	**1.9**	**1.4**	**0.7**

Table 10: Results for the general problem

5.2.3. The daily AdWords problem case

The Δ-re-optimized primal-dual procedure is also tested on the daily AdWords problem. Instead of re-optimizing every Δ 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^{exp} be the number drawn from the exponential distribution and T_j^{opt} the value of the solution obtained with the optimization model (4). As T_j^{opt} is more stable, we define T_j as the mean of T_j^{exp} and T_j^{opt}.

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

Table 11: Results for the daily Adwords problem

Table 11 shows that the Δ-re-optimized primal-dual algorithm has the best results with high empirical 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.

5.2.4. 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 12 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 Δ-re-optimized primal-dual algorithm by a small percentage, the empirical competitive ratio remains high. Finally, the Δ-re-optimized primal-dual procedure still outperforms Ciocan's algorithm.

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

Table 12: Results for the daily general problem

6. Conclusions

In this paper, we consider a class of online bipartite resource allocation problems with budget and resource constraints. The main goal has been to extend current online algorithms towards more practical settings where additional information about future requests can be available. In particular, we propose to re-optimize the primal-dual algorithm in order to make use of such information. At each re-optimization, based on the current historical data, we generate a random scenario that represents one sequence of future requests, and we seek an optimal solution for such a scenario, using 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 improves on the greedy and the basic primal-dual algorithms for the Adwords and the general online bipartite resource allocation problems. The results also show that the re-optimized primal-dual procedure is generally better than the algorithm of Ciocan and Farias (2012). Furthermore, some added practical modifications allow us to solve some problem instances within a more realistic framework: computational time is reduced to a more reasonable level, a learning process estimates the distribution (when unknown) of the items type from the current historical data, and an efficient inference about the number of remaining requests removes the need to know the total number of requests ahead of time.

Appendix A. Summary of all mathematical notations

Global parameters

Notation	Description
N	Total number of buyers
M	Total number of items
L	Total number of resources
T	Total number of requests

Global data

Notation	Description
n_k	Total number of items of type k out of the T requests
B_i	Total budget of buyer i
B_i^{left}	Total budget left of buyer i for the jth request
c_{ik}	Price that buyer i is willing to pay for item of type k
$\boldsymbol{F_k}$	Vector of size L representing the resources available for items of type k
$\boldsymbol{d_{ik}}$	Vector of size L representing the resources consumption of items of type k allocated to buyer i

Stochastic data

Notation	Description
Ω_j	Set of future scenarios for the jth request
p^ω	Probability of the scenario $\omega \in \Omega_j$
T_j	Number of requests left after the jth request. $T_j = T - j$
T^ω_{jk}	Number of requests for items of type k in the scenario ω

Primal variables

Notation	Description
X^k_j	=1 if the jth request is an item of type k, =0 otherwise
$Q(x, \omega)$	Recourse function
x_{ij} or x_i	=1 if the jth request is allocated to buyer i, =0 otherwise
y^ω_{ik}	Number of items of type k allocated to buyer i for scenario ω

Dual variables

Notation	Description (for a scenario ω)
α^ω_k	Dual variable associated to the allocation constraint of items of type k
β^ω_i	Dual variable associated to the budget constraint of buyer i
$\boldsymbol{\gamma}^\omega_k$	Vector of dual variables associated to the resource constraints of items of type k

Algorithms parameters

Notation	Description
Δ	Number of requests to wait for a re-optimization
δ	Proportion of re-optimizations ($= \frac{\Delta}{T}$)
ϵ	Minimum proportion of the expected total budget consumption
ρ	Smooth the updates of the re-optimization

Table A.13: Summary of all notations

Acknowledgments

Research funded in part by NSF (grant 1029603).

References

Aggarwal, G., Goel, G., Karande, C., Mehta, A., 2011. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In: Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms. SIAM, pp. 1253–1264.

Agrawal, S., Wang, Z., Ye, Y., 2014. A dynamic near-optimal algorithm for online linear programming. Operations Research 62 (4), 876–890.

Benders, J. F., 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische mathematik 4 (1), 238–252.

Bertsimas, D., Sim, M., 2003. Robust discrete optimization and network flows. Mathematical programming 98 (1-3), 49–71.

Birge, J. R., Louveaux, F., 2011. Introduction to stochastic programming. Springer Science & Business Media.

Bolstad, W. M., 2004. Introduction to Bayesian statistics. John Wiley & Sons.

Borodin, A., El-Yaniv, R., 1998. Online computation and competitive analysis. Cambridge University.

Buchbinder, N., Jain, K., Naor, J. S., 2007. Online primal-dual algorithms for maximizing ad-auctions revenue. In: Proceedings of the 15th annual European conference on Algorithms (ESA 2007). Springer, pp. 253–264.

Ciocan, D. F., Farias, V. F., 2012. Dynamic allocation problems with volatile demand. Mathematics of Operations Research 37 (3), 501–525.

Coy, P., March 05 2006. The secret to google's success. `http://www.bloomberg.com/bw/stories/2006-03-05/the-secret-to-googles-success`, Accessed: 2015-02-16.

Devanur, N. R., Jain, K., Sivan, B., Wilkens, C. A., 2011. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In: Proceedings of the 12th ACM conference on Electronic commerce. ACM, pp. 29–38.

Eghbali, R., Swenson, J., Fazel, M., 2014. Exponentiated subgradient algorithm for online optimization under the random permutation model. arXiv preprint arXiv:1410.7171.

Feldman, J., Henzinger, M., Korula, N., Mirrokni, V. S., Stein, C., 2010. Online stochastic packing applied to display ad allocation. In: Algorithms ESA 2010. Springer, pp. 182–194.

Feldman, J., Mehta, A., Mirrokni, V., Muthukrishnan, S., 2009. Online stochastic matching: Beating 1-1/e. In: 50th Annual IEEE Symposium on Foundations of Computer Science, 2009 (FOCS'09). IEEE, pp. 117–126.

Goel, G., Mehta, A., 2008. Online budgeted matching in random input models with applications to adwords. In: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, pp. 982–991.

Google, 2014. 2014 financial tables. `https://investor.google.com/financial/tables.html`, Accessed: 2015-02-16.

Jaillet, P., Lu, X., 2011. Online resource allocation problems. Tech. rep., Working paper.

Jaillet, P., Lu, X., 2012. Near-optimal online algorithms for dynamic resource allocation problems. arXiv preprint arXiv:1208.2596.

Jaillet, P., Lu, X., 2014. Online stochastic matching: New algorithms with better bounds. Mathematics of Operations Research 39 (3), 624–646.

Jaillet, P., Wagner, M. R., 2010. Online optimization-an introduction. TutORials in Operations Research: Risk and Optimization in an Uncertain World, INFORMS, 142–152.

Kalyanasundaram, B., Pruhs, K. R., 2000. An optimal deterministic algorithm for online b-matching. Theoretical Computer Science 233 (1), 319–325.

Karande, C., Mehta, A., Tripathi, P., 2011. Online bipartite matching with unknown distributions. In: Proceedings of the forty-third annual ACM symposium on Theory of computing. ACM, pp. 587–596.

Karp, R. M., 1991. An introduction to randomized algorithms. Discrete Applied Mathematics 34 (1), 165–201.

Karp, R. M., Vazirani, U. V., Vazirani, V. V., 1990. An optimal algorithm for on-line bipartite matching. In: Proceedings of the twenty-second annual ACM symposium on Theory of computing. ACM, pp. 352–358.

Kesselheim, T., Tönnis, A., Radke, K., Vöcking, B., 2014. Primal beats dual on online packing lps in the random-order model. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing. ACM, pp. 303–312.

Legrain, A., Fortin, M.-A., Lahrichi, N., Rousseau, L.-M., 2014. Online stochastic optimization of radiotherapy patient scheduling. Health care management science, 1–14.

Manshadi, V. H., Gharan, S. O., Saberi, A., 2012. Online stochastic matching: Online actions based on offline statistics. Mathematics of Operations Research 37 (4), 559–573.

Mehta, A., Saberi, A., Vazirani, U., Vazirani, V., 2007. Adwords and generalized online matching. Journal of the ACM (JACM) 54 (5), 22.

Molinaro, M., Ravi, R., 2014. The geometry of online packing linear programs. Mathematics of Operations Research 39 (1), 46–59.

Puterman, M. L., 2014. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons.

Reiss, C., Wilkes, J., Hellerstein, J. L., 2011. Google cluster-usage traces: format+ schema. Google Inc., White Paper.

Van Hentenryck, P., Bent, R., 2009. Online Stochastic Combinatorial Optimization. The MIT Press.