CIRRELT
Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport
Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Stochastic Online Bipartite Resource Allocation Problems

Antoine Legrain
Patrick Jaillet

June 2013

CIRRELT-2013-38

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Stochastic Online Bipartite Resource Allocation Problems

Antoine Legrain[1,*], Patrick Jaillet[2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

[2] Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Room 32-D624, Cambridge, MA 0213, U.S.A.

Abstract. This paper deals with online resource allocation problems (online auctions) whereby buyers with a limited budget want to purchase items which arrive one at a time and which consume some limited amount of resources upon allocation. There have been two main recent algorithmic approaches to address such online problems. One seeking algorithms with performance guarantee against worst case input; another with performance guarantee against specific probabilistic assumptions on the input. We propose in this paper an innovative practical method that combines the strengths of these two approaches, and that requires only a limited amount of information about how the future can unfold. We provide extensive numerical comparisons about our proposed computational scheme.

Keywords: Resource allocation, online optimization, primal-dual algorithm, stochastic optimization, L-Shaped method, adwords problem, bayesian inference.

Acknowledgements. Financial support for this work was provided in part by National Science Foundation (NSF) and Office of Naval Research (ONR) grant. This support is gratefully acknowledged.

---

* Corresponding author: Antoine.Legrain@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2013

1. Introduction

As mentioned in 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 advertisement generated more than 20 billions of dollars in 2010. Online optimization techniques are essential for Google and each improvement can lead to significant benefits. For such optimization problems, information (e.g., about clients, advertisements, etc.) is typically revealed step by step, and irrevocable decisions must be taken along the way. The so-called AdWords problem, introduced in [22, 23] is a typical example: search engines must choose which advertisement is the best to display for each new keyword search request in order to maximize their revenue.

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

- a set of N buyers, each interested in purchasing one or more items in a set $\{1 \dots M\}$;
- buyer $i \in \{1 \dots N\}$ is willing to pay c_{ik} for each item $k \in \{1 \dots M\}$ and has a limited total budget B_i;
- quantities $F_k \in \mathbb{R}_+^L$ of L distinct resources are available for all requests of type $k \in \{1 \dots M\}$;
- each item $k \in \{1 \dots M\}$ consumes amounts $(d_{ik}^l)_{l=1}^L$ of resources F_k and provides a revenue c_{ik} to the operator when allocated to buyer $i \in \{1 \dots N\}$.

The objective is to maximize the revenue of the operator under all the previous problem features. In the online case, the rate at which requests arrive and the type k_j of the jth request are unknown and, in some cases, can be described by a stochastic process. The challenge is to match each request to a buyer without knowing the overall sequence.

1.1. Applications

A wide range of applications can be modeled by this bipartite resource allocation problem. In addition to the ad display example introduced above, the following examples are of interests in routing, revenue management or scheduling.

Recently, Google has published two anonymous sets of real-data of their compute clusters [25]. Tasks from different services are arriving continuously; the resources allocated to each service has to be precisely estimated in order to avoid under or over-provisioning. New tasks must be placed in the workload while respecting priorities and constraints. There are two types of constraints; first, hardware constraints dealing with disk space, memory space and the number of cpu. Second, software constraints ensures the right configuration of virtual machines. Ali-Eldin et al. [1] present the problem and its different challenges. The algorithm which manages the cloud has to be fast, robust and adaptive. Balancing resources among services and inferring future resource needs are difficult under uncertainty.

Seat allocation problems are also very interesting, and appear in hospitality, rail and airline industries. Clients purchase a ticket or a room and the company must choose a fare for this service according to the current availability. The capacity of an hotel, a train or a plane must be respected as companies try to maximize their revenue. The uncertainty of the demand make decisions difficult.

A last application is the patient booking problem. Patients with different priorities arrive one by one and ask for an appointment in hospitals or clinics. On one hand, resources should be kept available for future high priority patients and, on the other hand, low priority patients has to also access to an appointment. Legrain et al. [20] introduce a practical problem in a radiotherapy center. The resources management is challenging because of the random arrival of patients.

All those examples have motivated our research. We are confident that the use of stochastic optimization techniques combined to an analysis of historical data can lead to great improvements of current techniques in online optimization.

1.2. Our Techniques and Results

We present different improvements on current approaches for solving online bipartite resource allocation problems, making use of available stochastic information. Our work builds upon the primal-dual algorithm presented by Buchbinder et al. [7]. 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 in order to infer the expected revenue of an allocation. We make the best decision for the current request by maximizing this revenue. This procedure, although providing high quality solutions, remains however computationally very demanding.

A Re-optimized Primal-dual Algorithm: The previous algorithm is modified to estimate the dual variables in the primal-dual procedure. Deterministic algorithms can take poor decisions which lead to a significant deterioration of the solution. We aim to correct these mistakes by performing updates of the dual variables during the process.

An Estimation of the Future: We assume here that the stochastic process of the demand is unknown. We first use machine learning tools to infer the probability distribution of the M items based on historical data. Then, upon each arrival of a request, we 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. Results shows that our procedure performs very well for most scenarios: its competitive ratio is always above 0.9.

Outline: In the next paragraph, we provide a literature review of relevant work, placing these contributions in the context of on online bipartite allocation problems. In Section 2, we introduce our procedure built on a stochastic optimization problem. In Section 3, we show different modifications on our algorithm to solve more realistic problems. In Section 4, we provide extensive numerical comparisons about our computational scheme. Finally, we

conclude with some final remarks and ideas of future developments.

1.3. Related Work

It is difficult to design algorithms operating under uncertainty which perform well in all situations. There are different ways to handle uncertainty. On one hand, we have stochastic optimization techniques, which are based on probabilistic models of the future: dynamic programming [5], Markov decision processes [24], robust optimization [4] are some examples. These approaches can provide good results if the probability distribution governing the data is well known.

On the other hand, online optimization approaches e.g., see [15], can insure a certain quality without knowledge about the future. The competitive ratio is a frequently used measure of quality. The competitive ratio of an online algorithm on an instance I is defined as follow:

$$c(I) = \frac{\text{Obj}_{online}(\text{I})}{\text{Obj}_{optimal}(\text{I})}$$

Obj_{online} is the value of the objective for the solution given by the online algorithm and $\text{Obj}_{optimal}$ is the value of the objective for the offline solution. For a maximization problem, the competitive ratio c is defined as the infimum of $c(I)$ over all instances. We then say that the online algorithm is c-competitive.

The offline version of the bipartite resource allocation problems of interest in our paper can be formulated as the following integer linear program (1):

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

$$\text{subject to:} \tag{1b}$$

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

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

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

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

The variables x_{ij} is equal to 1 if the buyer i purchases the jth request, 0 otherwise. The objective (1a) represents the revenue of the operator. Constraint (1c) ensures that a request can be bought only once. Constraint (1d) limits the total budget of the buyer i. Finally, constraint (1e) verifies that the items k do not consume more than the available resources.

Different versions of the bipartite resource allocation problems have been studied in an online fashion. Karp et al. [19] deals with a simple form of the bipartite matching (i.e. model (1) without constraints (1d) and (1e) and with $c_{ik} = 1$). The goal is thus to maximize the number of requests matched. The authors proposed a best possible $(1 - \frac{1}{e})$-competitive randomized algorithm (RANKING). For the b-matching problem where each buyer cannot be matched more than b times (i.e. model (1) without constraint (1e) and with $B_i = b$, $c_{ik} = 1$), Kalyanasundaram and Pruhs [16] provide a $(1 - \frac{1}{e})$-competitive algorithm (BALANCE). They also prove that this competitive ratio is the best.

For the more complex Adwords problem (i.e. model (1) without constraint (1e)), the most intuitive algorithm is the greedy algorithm (Algorithm 1). For the jth request, the algorithm chooses the buyer i with the higher price c_{ik_j}, as long as there is enough budget B_i left. This algorithm is $(\frac{1}{2})$-competitive [23].

Algorithm 1 Greedy Algorithm

$x_{ij} = 0$

for all jth request **do**

FIND a buyer i **such that**

MAXIMIZE c_{ik_j} **AND** there is enough budget B_i left

SET $x_{ij} = 1$

end for

Mehta et al. [23] introduce an algorithm with a better competitive ratio of $1 - \frac{1}{e}$. Buchbinder et al. [7] shows that a primal-dual algorithm (see Algorithm 2 below) can be designed for this problem with the same competitive ratio. The mathematical formulation is presented in Table 1. The primal problem is just the linear programming relaxation of the general model (1) without the resource constraints (1e).

Please note that the primal-dual algorithm (Algorithm 2) has a simple interpretation using reduced costs associated with the simplex algorithm as applied to the primal prob-

Packing Problem P	Covering Problem D
$\max \sum_{i=1}^{N}\sum_{j=1}^{T} c_{ik_j}x_{ij}$	$\min \sum_{i=1}^{N} B_i r_i + \sum_{j=1}^{T} z_j$
subject to:	subject to:
$\sum_{i=1}^{N} x_{ij} \leq 1 \qquad \forall j = 1 \dots T$	$c_{ik_j} r_i + z_j \geq c_{ik_j} \qquad \forall i = 1 \dots N, \forall j = 1 \dots T$
$\sum_{j=1}^{T} c_{ik_j} x_{ij} \leq B_i \qquad \forall i = 1 \dots N$	
$x_{ij} \geq 0 \qquad \forall i = 1 \dots N, \forall j = 1 \dots T$	$r_i, z_j \geq 0 \qquad \forall i = 1 \dots N, \forall j = 1 \dots T$

Table 1: Formulations primal-dual of the LP relaxation of the AdWords Problem

Algorithm 2 Primal-dual Algorithm

$x_{ij} = 0, r_i = 0, z_j = 0$

for all jth request **do**

FIND a buyer i who **MAXIMIZES** $c_{ik_j}(1 - r_i)$

REQUIRE $r_i < 1$ AND there is enough budget B_i left

SET $x_{ij} = 1, z_j = c_{ik_j}(1 - r_i)$ and $r_i = r_i(1 + \frac{c_{ik_j}}{B_i}) + \frac{c_{ik_j}}{(c-1)B_i}$

end for

lem. Indeed, the current solution before the arrival of the jth request (i.e. $\forall l \geq j,\ \forall i \in \{1 \dots N\},\ x_{ij} = 0$) is a basis of the primal problem. To apply the next step of the simplex algorithm from this basis, the variable with the highest reduced cost has to enter the basis. The reduced cost is $c_{ik_j}(1 - r_i) - z_j$. Algorithm 2 just applies the same idea than the simplex algorithm by choosing a variable (x_{ij}) which maximizes the reduced costs for the jth request. Jaillet and Lu [12] use those ideas for a more general case, including resource constraints of type (1e), but assuming a special homogeneous case ($c_{ik} = d_{ik}$). They obtain a $\frac{1}{2}$-competitive algorithm.

Other authors such as Feldman et al. [10], Manshadi et al. [21], Jaillet and Lu [13], Karande et al. [17] mix online and stochastic ideas. They study the online matching problem and improve the bounds on the competitive ratios up to 0.706 [13] by using offline strategies. They assume that the requests are drawn independent and identically distributed from a known probability distribution.

Bent and Van Hentenryck [3] show different kinds of architecture to use stochastic information during the online procedure. They present three different architectures: Expectation, Consensus and Regret. All these architectures use the same kind of ideas: the future has to be sampled and an offline algorithm is used to solve the problem with the sample events. Recent papers present algorithms for solving the resource allocation problems with stochastic information. They use an offline LP problem to build their future strategy. Ciocan and Farias [8] have proved a worst case guarantee on average (i.e. the expected competitive ratio is 0.342 under mild assumptions). Their algorithm computes a strategy based on statistics at the beginning which gives a percentage of item k allocated to buyer i. Then, primal problems are solved during the allocation process to update this strategy with the new information available. Contrary, Feldman et al. [11], Jaillet and Lu [14] use the dual solution as a base of their strategy. Furthermore, Jaillet and Lu [14] do not require the total number of requests at the beginning, this is the first paper of this kind.

We present an hybrid algorithm which mixes online and stochastic optimization techniques and solves bipartite resource allocation problems. Our stochastic procedure and the primal-dual algorithm are related, the main difference being the dual variables r_i's updates. Our algorithm is using available stochastic information during the online procedure to compute these dual variables with a linear program. It also deals with an unknown total number of requests. We will compare our procedure with the algorithm of Ciocan and Farias [8].

2. Stochastic Algorithm

In this section, we present a stochastic optimization formulation and an algorithm to solve the bipartite resource allocation problem (1). The proposed algorithm 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 events, the set of the future scenarios. Each event ω has the same number of items T_j;
- p^ω is the probability of the event $\omega \in \Omega_j$;
- T_{jk}^ω is the number of items k in the event ω. So $T_j = \sum_{k=1}^{M} T_{jk}^\omega$.

2.1. Stochastic optimization formulation

We present here a classical stochastic optimization formulation 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 over all the events $\omega \in \Omega_j$. Define the following new variables:

- B_i^{left} is the budget left of 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 item k allocated to the buyer i for the event 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) insures that the jth request is allocated to a maximum of one buyer. The constraints (2c) bound by T_{jk}^{ω} the number of items of type k matched with a buyer for the event ω. Finally, the constraints (2d) prevent exceeding the budget for each buyer and each event. The formulation (2) is the most simple that we can consider to describe the local best decisions to be made upon the arrival of a new request, but this is a very large and difficult problem to solve. The relaxation of the integer constraint (2e) on y_{ik}^{ω} is a first good idea to simplify the model. The L-Shaped method, presented in the next Section 2.2, is a second way to improve the computational time.

2.2. L-Shaped Method

In their book, Birge and Louveaux [5] study different stochastic problems and show how to use the L-Shaped method in order to solve them. This technique is based on a Benders decomposition [2]. 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 in (2a) the part of this objective dealing with scenarios by a recourse function Q which becomes the objective function in the slave problems. Then, we obtain the problems presented in Table 2.

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

Table 2: Decomposition in sub-problems

The recourse function Q has to be computed for each value of the variable x and for each event 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. Table 3 presents a new master problem where the cuts approximate the recourse function Q.

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:
$\sum_{i=1}^{N} x_i \leq 1$	$\alpha_k^\omega + c_{ik} \beta_i^\omega \geq c_{ik} \qquad \forall i = 1 \dots N, \forall k = 1 \dots M$
$\theta^\omega \leq \sum_{k=1}^{M} T_{jk}^\omega \alpha_k^\omega + \sum_{i=1}^{N} (B_i^{left} - c_{ik_j} x_i) \beta_i^\omega \qquad \forall \omega \in \Omega_j$	
$x_i \in \{0,1\} \qquad \forall i = 1 \dots N$	$\alpha_k^\omega,\ \beta_i^\omega \geq 0 \qquad \forall i = 1 \dots N, \forall k = 1 \dots M$

Table 3: Benders decomposition

It can be noted that the weak duality theorem justifies those approximations:

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

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 Figure 1, stops as soon as the optimum is reached, otherwise some cuts continue to be generated. We make a simplification: we solve the master problem only once. Indeed, that allows to decrease the computational time and to make an easy link with the primal-dual Algorithm 2, as explained in Section 2.3.

Algorithm 3 chooses the best buyer at the beginning, and, if the master problem gives the same solution, this is the optimum, otherwise the solution of the master problem should be

Algorithm 3 Stochastic primal-dual Algorithm

$x_{ij} = 0$

for all jth request **do**

USE the greedy Algorithm 1 to set x

Solve all the dual slave problems and **add** every cuts to the master problem

Solve the master problem and keep this solution x

end for

better for the future. This algorithm seems to have promising results. Our computational tests will confirm this later.

2.3. Links with the Primal-dual

The constraints (3) become equalities in the stochastic primal-dual algorithm ; indeed, as this is a maximization problem, each variable θ^ω reaches, for every events, the minimum of the constraint (3) associated to this variable. With this new formulation, we are able to write:

$$\forall x \in [0,1], \omega \in \Omega_j \ \theta^\omega = \sum_k T^\omega_{jk} \alpha^\omega_k + \sum_i (B^{left}_i - c_{ik_j} x_i) \beta^\omega_i$$

With all those equalities, the variables θ^ω are no longer needed, the updated cuts can be integrated in the objective:

$$\sum_{i=1}^{N} c_{ik_j} x_i + \sum_{\omega \in \Omega_j} p^\omega [\sum_{k=1}^{M} T^\omega_{jk} \alpha^\omega_k + \sum_{i=1}^{N} (B^{left}_i - c_{ik_j} x_i) \beta^\omega_i]$$

All the constants are taken off the objective to obtain $\sum_{i=1}^{N} c_{ik_j} x_i (1 - [\sum_{\omega \in \Omega_j} p^\omega \beta^\omega_i])$. The cost of the variable x_i in this objective is exactly the same than its reduced 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. In our case, we use stochastic information to build the dual variables $r_i = \sum_{\omega \in \Omega_j} p^\omega \beta^\omega_i$. 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 matched to the buyer i;

- $c_{ik_j} r_i$ is the expected loss in the future if such an allocation is chosen (the future budget B_i^{left} will be reduced of c_{ik_j}).

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

2.4. Generalized Problems

We apply those techniques to the problem (1) seen in the Section 1.2. This leads to the following stochastic optimization where F_k^{left} is the resources left of the item k. This is the new master problem:

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

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:

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

subject to:

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

$$\alpha_k^\omega,\ \beta_i^\omega,\ \gamma_k^\omega \geq 0 \qquad \forall i = 1 \dots N, \forall k = 1 \dots M$$

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

2.5. Re-optimized Primal-dual

Algorithm 3 should be better than the primal-dual algorithm, because 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, the linear problem will be solve only each Δ requests.

Algorithm 4 Re-optimized primal-dual Algorithm

$x_{ij} = 0, r_i = 0$

for all jth request **do**

  if $j \equiv 0$ (**mod** Δ) **then**

    USE stochastic primal-dual Algorithm 3

    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

  USE primal-dual Algorithm 2 without update, if a re-optimization has been made

end for

Algorithm 4 will use, most of the time, the same updates than the primal-dual algorithm, because this algorithm is $(1 - \frac{1}{e})$-competitive. Each Δ requests, Algorithm 4 uses stochastic information to fix errors which may have been made during the latter steps by updating the dual variables with Algorithm 3. Furthermore, the parameter ρ allows us to smooth the value of the dual variables at each re-optimization. Results are shown in the last Section 4. In the next Section 3, we make some modifications to use this algorithm in a more realistic world.

3. Practical Modifications

In this section, we show some ideas to transform Algorithm 4 to a realistic procedure. To compare the quality of an improvement, we use the competitive ratio.

3.1. Improvement of the Solving Time

The set of the sample events Ω_j is huge, it is impossible to compute the slave problems for all events ω. The Table 4 presents the competitive ratio 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 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 4: 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 Ω_j, we will solve only one random event ω_0 for the dual slave problems (Table 3). First, as the constraints of the dual slave problems are independent of the events, the dual variables $\alpha_k^{\omega_0}$, $\beta_i^{\omega_0}$ and $\gamma_k^{\omega_0}$ remains feasible for every events. Then, the cut (3) for the event ω_0 becomes a good approximation of $Q(x,\omega)$ for all events ω. Second, the use of one random scenario instead of a determinist scenario leads to a randomized algorithm. Randomization is usually a way to eliminate worst-case behavior within a deterministic algorithm [18]. The new objective is: $\sum_{i=1}^{N}\{c_{ik_j} - [\sum_{\omega\in\Omega_j} p^{\omega}(c_{ik_j}\beta_i^{\omega_0} + d_{ik_j}\gamma_{k_j}^{\omega_0})]\}x_i = \sum_{i=1}^{N}\{c_{ik_j} - (c_{ik_j}\beta_i^{\omega_0} + d_{ik_j}\gamma_{k_j}^{\omega_0})\}x_i$. This idea allows to decrease dramatically the computational time.

3.2. 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 are going to use Bayesian Statistics (Bolstad [6]) 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 consider that $(X_j)_{j=1}^T$ is a Bernoulli process: the stochastic process $(X_j)_{j=1}^T$ is i.i.d. and each X_j follows a Bernoulli distribution of mean μ which is in $[0,1]$.

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

This stochastic variable μ corresponds to the probability that we want infer. μ 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 $\beta(a_t, b_t)$, 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

$\hat{\mu}_{j+1} = \mathbb{P}[X_{j+1} = 1 | (X_l)_{l=1}^{j}, a, b]$:

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

We obtain the probability p_k that an item of type k has to arrive on the $(j+1)$th request. We must infer all the probabilities p_k at the same time. Let us now consider that $(X_j^k)_{j=1}^{T}$ follows a Bernoulli process for each item k:

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

We can 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 not any piece of information before the first request, we suppose that all requests are equiprobable ($\mathbb{E}[\beta(a^k, b^k)] = \frac{1}{C}$) and i.d. ($a^k = a$, $b^k = b$). So $a^k + b^k = Ca^k = Ca$. We also remark that:

$$\sum_{k=1}^{M} \hat{\mu_j^k} = \sum_{k=1}^{M} \frac{a + \sum_{l=1}^{j} X_l^k}{Ca + j} = \frac{Ca + \sum_{l=1}^{j} \sum_{k=1}^{M} X_l^k}{Ca + j} = \frac{Ca + j}{Ca + j} = 1 \qquad \forall j = 1 \dots T$$

Consequently, $\hat{\mu_j^k}$ can be interpreted as the probability that the $(j+1)$th request is an item of type k. We use this method to infer every p_k. Even if the stochastic process $(X_j)_{j=1}^{T}$ is not independent or does not follow a Bernoulli distribution, the procedure remains generally efficient.

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 [14]). In our case, we will present to ways to deal with this issue. First, we will infer the number of requests left T_j with a liner program. Second, we will assume as Jaillet and Lu [14] that the distribution governing the arrival times of requests is known.

The Table 5 shows different competitive ratios. Those tests have been made with 300 requests which were i.i.d and the competitive ratios are an average computed over 500 draws.

	greedy	primal-dual	stochastic primal-dual		
			real T_j	$T_j = 50$	
mean of the competitive ratio	0.9011	0.9569	0.9901	0.9332	real probability
			0.9877	0.9294	inferred probability

Table 5: Comparison of competitive ratios

This Table 5 proves that the inference of the probabilities is good comparing to the choice of the value T_j. If the number of requests left T_j is not estimated correctly, the stochastic primal-dual algorithm is worse than the primal-dual procedure. That is why it is very important to estimate T_j accurately. The optimization problem (4) seems to work well in order to infer T_j, but the estimation of T_j can be changed according to the reality of the environment.

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

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 take the minimum T_j such that there are enough requests to fill the budget left. Constraints (4b) force the optimization problem to fill the whole budget. At the same time, constraints (4c) insure that the expected number $p_k T_j$ of items k is just enough to cover every items k needed by constraint (4b). Then, we solve the optimization problem (4) in order to evaluate the number T_j of requests left. It remains one practical issue, instead of optimizing one linear problem, we have two solve two of them. Consequently, the computational time of the re-optimized primal-dual algorithm is double (tests have been made). This is necessary to obtain good results. Furthermore, if we make few re-optimizations, doubling the computational time should not affect a lot the algorithm. Results and a sensitivity analysis will be presented in Section 4.1.4.

The second method to estimate the number of requests left T_j is to change our way to model the problem. Instead of defining a problem by its number of requests T, we can define a time window in which requests arrive. In this case, the number of requests left depends of the event; the arrival of request is modeling by a stochastic process $(X_t^k)_{t\in[0,T]}$. T represents now the end of the process, is known and, consequently, does not need to be inferred. For the Google Adwords problem, if $T = 24$ hours, the algorithm maximizes Google's revenue over one day and B_i is the daily budget of buyer i. We will present some results later where the stochastic process follows an exponential distribution. We call this problem the daily Adwords problem.

4. Numerical Results

All these next tests have been done 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 problems (Table 3) 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 spend an important part of the whole budget. If this number is small, the greedy Algorithm 1 is the best, because it is always possible to chose the best bid without breaking any constraint.

4.1. Sensitivity analysis

The same instance is used for the following tests. It has 3 buyers (N =3), 8 items (M =8) and 350 requests (T =300). As the sequence of requests follows a Multinomial law, the number of each items is different from one instance to another one. At the same time, we keep the same probabilities for the arrivals of items. We have run 500 times this instance in order to have a good average for each output. We have also supposed that the probabilities (p_k) and the number of request T was known.

4.1.1. Analysis of parameters

We will study the comportment of the competitive ratios for different values of the couple (Δ, ρ). The tests have been done for the re-optimized primal-dual Algorithm 4.

		Δ							
		1	*3*	*6*	*15*	*30*	*60*	*150*	*300*
ρ	*0*	0.991	0.989	0.989	0.983	0.978	0.972	**0.971**	0.965
	0.01	**0.996**	0.994	0.99	0.982	0.977	0.972	0.969	**0.966**
	0.1	**0.996**	0.994	0.99	0.984	0.98	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**	0.98	0.972	0.966	0.961

Table 6: Tuning of the parameters Δ and ρ

The Table 6 shows that the competitive ratio results is very sensible to the number of re-optimization made. The parameter ρ is less important for the quality of the competitive ratio: $\rho = 0.2$ seems to be a good value according those results. We will keep this value for ρ in the next sections and we will continue to study the behavior of the re-optimized primal-dual algorithm as a function of the number of re-optimization.

4.1.2. Trade-off between computational time and competitive ratio

The computational time is very important in online optimization. The greedy and primal-dual algorithms need about one millisecond (ms) to solve 300 requests. The stochastic primal-dual procedure (Algorithm 3) need in average 1400 ms for the 300 requests. We can note that this last algorithm is a special case of the re-optimized primal-dual where $\Delta = 1$ and $\rho = 0$.

The Figure 2 shows first the evolution of the computational time as a function of the number of re-optimizations ($= \lfloor \frac{T}{\Delta} \rfloor$). The results are intuitive, the computational time depends linearly on the number of re-optimizations. As the same time, the second figure shows that the competitive ratio increases fast with the number of re-optimizations. It proves that the re-optimized primal-dual algorithm performs well with a small number of re-optimizations. We will use $\Delta = 30$ (10 re-optimizations), because 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 Δ has to be chosen according to the time and the

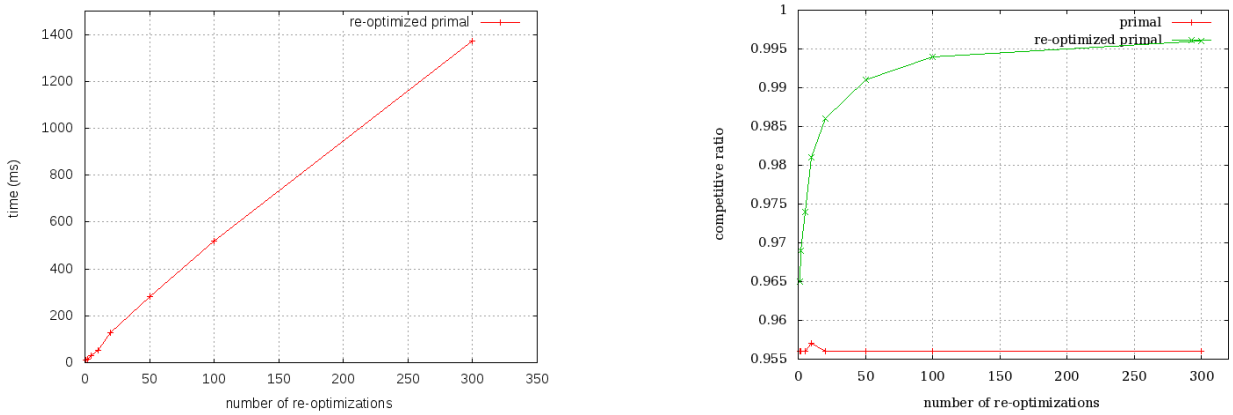

Figure 2: Trade-off between computational time and competitive ratio

computing resources available in reality.

4.1.3. Comparison beetwen the re-optimized primal-dual and Ciocan's algorithms

We compare our algorithm to Ciocan's algorithm [8]. We can note, before any comparison between those two algorithms, that they approximately have the same computational time as a function of the number of re-optimizations. We evaluate the competitive ratio under two different policies.

Number of re-optimizations		*1*	*3*	*6*	*15*	*30*	*60*	*150*	*300*	Policy	
Competitive ratio	Re-optimized primal	0.964	0.968	0.974	0.981	0.986	0.991	**0.994**	**0.996**	T_j	p_k
	Ciocan's method	**0.977**	**0.984**	**0.99**	**0.993**	**0.994**	**0.995**	**0.994**	**0.996**	known	known
	Re-optimized primal	**0.971**	**0.974**	**0.978**	**0.978**	**0.979**	**0.981**	**0.982**	**0.983**	T_j	p_k
	Ciocan's method	0.962	0.948	0.957	0.966	0.972	0.976	0.977	0.977	inferred	inferred

Table 7: Comparison between two algorithms

The Table 7 compares two policies: one ideal when parameters are known and one more realistic where parameters are inferred. These two algorithms react approximately in the same when the number of re-optimizations change: both decrease around the same rate. Furthermore, it is clear that Ciocan's algorithm is better for the first policy and that the re-optimized primal-dual algorithm is better for the second policy. It proves that Ciocan's procedure needs to know the probabilities p_k as well as the number of request T. The strength of the re-optimized primal-dual algorithm is to work well without these parameters. That is

why the estimation of T_j is a key problem. We will study the sensitivity of the optimization problem (4) in the next Section. Finally, $\Delta = 30$ seems to still be a good parameter for further tests.

4.1.4. Competitive ratios as a function of the number T_j of requests left

We study the influence of having a good inference for T_j. Let us consider that the probabilities p_k and the number of requests are unknown now. Bayesian inference, as shown in Section 3.2, is used to estimate p_k and T_j is inferred by the optimization problem (4). We change a little the constraint (4b): $\sum_{i=1}^{N}[\sum_{l=1}^{j-1} c_{ik_l}x_{il} + \sum_{k=1}^{M} c_{ik}y_{ik}] \geq \sum_{i=1}^{N} B_i$ in order to analyze the consequences of the inferred value T_j. This constraint is 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.

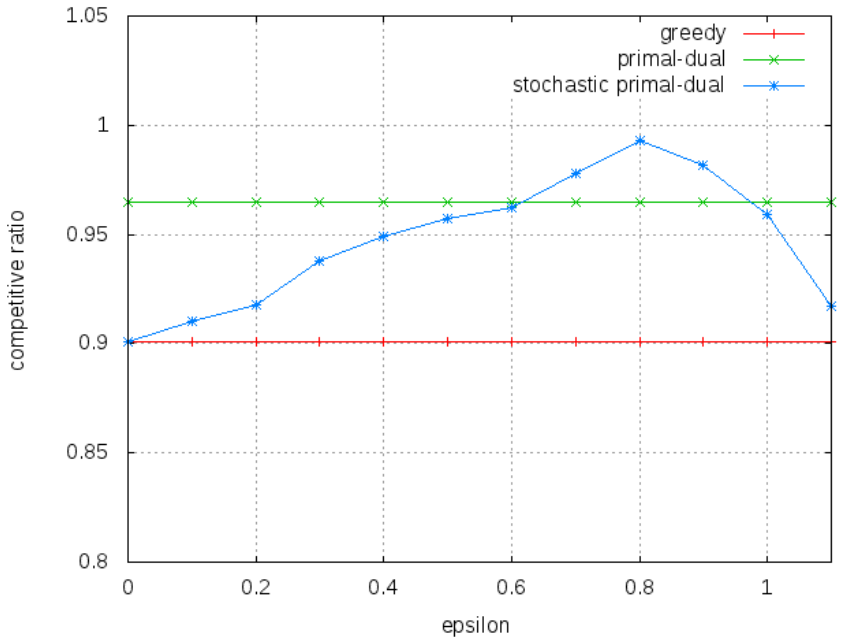

Figure 3: Evolution of the competitive ratio

The Figure 3 represents the evolution of the competitive ratio as a function of ϵ for the same instance as before. This graphic seems intuitive. First, for ϵ 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 ; so the dual variables r_i are null and our algorithm is the same than the greedy. Second, if ϵ is too big, the algorithm gives a weight that is too important to the future. The

procedure is always hoping to obtain some better requests in the future. The algorithm is waiting to maximize the revenue with the future requests, that is why its performance is decreasing. In this case, the dual variables r_i are close to 1. We can remark that it is better to underestimate the number of request left on the beginning and overestimate it at the end. On the beginning, we can choose important bids (underestimate T_j), but later, we have to be more careful (overestimate T_j), because a bad decision can cost a lot. The optimization problem (4) follows this evolution. Finally, we set the parameter ϵ on 0.8. The parameter ϵ is linked to the instance, it has to be chosen depending the environment.

4.2. Average competitive ratio

We keep using a Multinomial distribution to draw the sequences of items. The following tests shows the expected competitive ratios ; they still have been built on a sample of 500 draws. The instances have been sorted according to different criteria: the number of buyers, items and requests, the variance of the distribution, the gap between the bids c_{ik}. We generate six instances thanks to different criteria. They all have a reasonable gap for the capacity amounts d_{ik}. The following Table 8 describes them. Furthermore, we define the parameters δ in order to have the same behavior of the re-optimized primal-dual over the difference instance: $\delta = \frac{\Delta}{T}$. Then, $\rho = 0.2$, $\epsilon = 0.8$. The probabilities and the number of requests left are inferred. Finally, we compare the greedy Algorithm 1, the primal-dual Algorithm 2, the re-optimized primal-dual Algorithm 4 and Ciocan's algorithm [8]. For all these next tests, we suppose that the probabilities p_k and the number of request T are unknown.

4.2.1. In the AdWords Problem case

The Table 9 shows the average competitive ratio of our procedure and compares it with the other algorithms. The "Gain against greedy" is computed according to this formula:

$$\frac{c_{re-opt} - c_{greedy}}{c_{greedy}}$$

where c_{re-opt} and c_{greedy} are the competitive ratios of the re-optimized primal-dual and greedy algorithms. "Gain against primal-dual" and "Gain against Ciocan" are defined as the same way.

number of the instance	N	M	T	variance of distribution	gap of 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 8: Description of the instances

Instance	1	2	3	4	5	6
Average 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

Our procedure is most of the time the best. As we said before 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. This can be explained by the fact that our procedure is an hybrid method using the primal-dual algorithm and stochastic information. The primal-dual algorithm is $(1 - \frac{1}{e})$-competitive, it insures that the competitive ratio will be in any cases better than $(1 - \frac{1}{e})$. At the same time, the stochastic information allows to improve this ratio and can sometimes worsen it.

4.2.2. In the General Problem Case

The gains are really important when the resources constraints are added. The results remain good in this general case, even if our algorithm is not always the best. When the gains are negative, the percentage is very close to zero. It can be noted that the method of Jaillet and Lu [12] is used for the primal-dual and for the re-optimized primal-dual algorithms.

Instance	1	2	3	4	5	6
Average 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

In this case, there are big gains against the greedy and the primal-dual algorithms. On the other side, Ciocan's algorithm and our procedure have on average the same results. It can be noted that the instance 2 is difficult to solve.

4.2.3. The daily AdWords Problem case

We are also using the re-optimized primal-dual procedure on this problem. Instead of re-optimizing every Δ requests, we are re-optimizing every two hours (12 re-optimizations). The number of request left T_j is inferred by an exponential distribution. In order to stabilize this inferred number, we have computed the mean of the result of the exponential law and the estimation of T_j by the optimization model (4).

Instance	1	2	3	4	5	6
Average number of requests	359	480	394	313	456	311
Average 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

The Table 11 shows that the re-optimized primal-dual have the best results when probabilities p_k and the number of requests left T_j are unknown. The average competitive ratios are very high.

4.2.4. The daily General Problem case

In this section, we are computing the same kind of results than the previous Section, but we are adding the resource constraints (1e) as in the general optimization problem (1).

Instance	1	2	3	4	5	6
Average number of requests	360	480	396	312	456	312
Average 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

Gains remains big for most of the instances. The behavior of the re-optimized primal-dual algorithm is not as good as the greedy algorithm for the instance 5 and 6, bur the average competitive ratio remains high. Another time, our procedure is a little better than Ciocan's algorithm.

5. Conclusions

In this paper, we consider a general bipartite allocation problem with budget and resource constraints in an online fashion. The main goal is to improve current online algorithms with the information available during the procedure. We propose a modification of the primal-dual algorithm to take into account this information. The new formulation of the problem uses stochastic programming and especially the L-Shaped method. We build stochastic sub-problems to update the dual variables. This new algorithm gives very good results. It outperforms the greedy and the primal-dual algorithms on the Adwords and the general bipartite resource allocation problems. Results show that our procedure is also generally better than Ciocan's algorithm. Furthermore, the practical modifications allow us to be very applicable: the computational time is reasonable, the learning process from information is useful, the inference of the number of requests left is efficient and can be easily changed.

One may raise the issue of computing the competitive ratio. As it is a stochastic algorithm, this bound is very difficult to compute. Furthermore, we aim to developed practical algorithms. Finally, it will be really good to test this algorithm on real sets of data as those that Google should have.

References

[1] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth. Efcient provisioning of bursty scientic workloads on the cloud using adaptive elasticity control. *ScienceCloud '12 Proceedings of the 3rd workshop on Scientific Cloud Computing Date*, pages 31–40, 2012.

[2] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, pages 238–252, 1962.

[3] R. Bent and P. Van Hentenryck. Online stochastic optimization without distributions. *Proceedings of the 15th international conference on automated planning and scheduling (ICAPS 2005)*, 2005.

[4] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, pages 49–71, 2003.

[5] J-R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer, 1997.

[6] W.M. Bolstad. *Introduction to Bayesian statistics*. John Wiley, 2007.

[7] N. Buchbinder, K. Jain, and J. Noar. Online primal-dual algorithms for maximizing ad-auctions revenue. *ESA'07 Proceedings of the 15th Annual European Conference on Algorithms*, pages 253–264, 2007.

[8] D. F. Ciocan and V. F. Farias. Dynamic allocation problems with volatile demand. *Mathematics of Operations Research*, 2011.

[9] Peter Coy. The secret to google's success. *Bloomberg Businessweek*, March 05 2006.

[10] J. Feldman, A. Mehta, V. S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. *FOCS'09: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126, 2009.

[11] J. Feldman, M. Henzinger, N. Korula, V. Mirrokni, and C. Stein. Online stochastic packing applied to display ad allocation. *ESA'10 Proceedings of the 18th Annual European Conference on Algorithms*, pages 182–194, 2010.

[12] P. Jaillet and X. Lu. Online resource allocation problems. *Working paper*, 2011.

[13] P. Jaillet and X. Lu. Online stochastic matching: New algorithms with better bounds. *Working paper*, 2012.

[14] P. Jaillet and X. Lu. Near-optimal online algorithms for dynamic resource allocations. *Working paper*, 2012.

[15] P. Jaillet and M. Wagner. *Online Optimization - An Introduction*. INFORMS, 2010.

[16] B. Kalyanasundaram and K. R. Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, pages 319–325, 2000.

[17] C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. *STOC '11: Proceedings of the 43nd Annual ACM Symposium on Theory of computing*, 2011.

[18] R. M. Karp. An introduction to randomized algorithms. *Discrete Applied Mathematics*, pages 165–201, 1991.

[19] R. M. Karp, U. Vazirani, and V. Vazirani. An optimal algorithm for on-line bipartite matching. *STOC'90 Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 352–358, 1990.

[20] A. Legrain, N. Lahrichi, L.M. Rousseau, and M.A. Fortin. Online stochastic optimization of radiotherapy patient booking. *Working paper*, 2013.

[21] V. H. Manshadi, S. O. Gharan, and A. Saberi. Online stochastic matching: Online actions based on offline statistics. *SODA'11 Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1285–1294, 2010.

[22] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. *FOCS'05: Proceedings of the 4th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–273, 2005.

[23] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. *Journal of the ACM*, 54(5), 2007.

[24] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[25] C. Reiss, J. Wilkes, and J. L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA, November 2011. Posted at URL `http://code.google.com/p/googleclusterdata/wiki/TraceVersion2`.