

Efficient Scheduling of Internet Banner Advertisements

ALI AMIRI

Oklahoma State University

and

SYAM MENON

University of Texas at Dallas

Despite the slowdown in the economy, advertisement revenue remains a significant source of income for many Internet-based organizations. Banner advertisements form a critical component of this income, accounting for 40 to 50 percent of the total revenue. There are considerable gains to be realized through the efficient scheduling of banner advertisements. This problem has been observed to be intractable via traditional optimization techniques, and has received only limited attention in the literature. This paper presents a procedure to generate advertisement schedules under the most commonly used advertisement pricing scheme—the CPM model. The solution approach is based on Lagrangean decomposition and is seen to provide extremely good advertisement schedules in a relatively short period of time, taking only a few hundred seconds of elapsed time on a 450 MHz PC compared to a few thousand seconds of CPU time on a workstation that other approaches need. Additionally, this approach can be incorporated into an actual implementation with minimal alterations and hence is of particular interest.

Categories and Subject Descriptors: K.4.4 [**Computers and Society**]: Electronic Commerce; G.1.6 [**Numerical Analysis**]: Optimization—*Integer programming*; G.2.3 [**Discrete Mathematics**]: Applications

General Terms: Algorithms, Management

Additional Key Words and Phrases: WWW, scheduling, banner advertising

1. INTRODUCTION

According to the Internet Ad Revenue Report from the Interactive Advertising Bureau (IAB) [2001], Internet advertising revenue for the United States was \$8.2 billion in the year 2000. While this value decreased to \$7.2 billion in 2001, it remained a significant part of total revenue. Indeed, for many web-based

Author's addresses: A. Amiri, College of Business Administration, Oklahoma State University, Stillwater, OK 74078; email: amiri@okstate.edu; S. Menon, School of Management, University of Texas at Dallas, Richardson, TX 75083; email: syam@utdallas.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax: +1 (212) 869-0481, or permissions@acm.org.

© 2003 ACM 1533-5399/03/1100-0334 \$5.00

organizations, revenue from advertisements is often the only source of income (e.g., Yahoo.com, AltaVista.com). The report also notes that approximately 47% of the advertisements viewed were in banner form. Buchwalter et al. [2001] mention that 99 percent of all Web sites offer standard banner advertisements, underlining the importance of this form of on-line advertising. Under these conditions, it is reasonable to expect that there are potentially significant gains to be made by scheduling banner advertisements efficiently.

One notable (and to some extent, unique) feature of Internet advertising is space sharing. Advertising space is shared in various ways—for example, advertisements can target specific users or share the total area available for display. This paper will address the latter problem in the context of banner advertisements. The importance of this problem is underlined by the fact that approximately 50% of the advertisements viewed tend to be in banner form [Haney 2001].

A banner ad is essentially a hypertext link that is associated with a box containing graphics that instructs a Web server to bring up a particular Web page when a user clicks on the banner. Unlike a traditional print advertisement, it has the ability to bring a potential customer directly to the advertiser's Web site. Another critical difference involves the ability to dynamically alter the advertisements presented. Banner ads come in a variety of shapes and sizes. The IAB specifies fourteen different banner sizes, according to pixel dimensions, with the full banner (468×60) being the most popular. While these are not the only sizes being used, they are a good representation of the range of common banner ads available.

An advertiser interested in posting banner ads can (i) arrange to display other Web sites' banner ads in exchange for them displaying its ad, (ii) pay publisher sites to post their banner, or (iii) pay a banner ad network like DoubleClick.com to post the banner on a variety of publisher sites. As already mentioned, selling banner advertising space is a common approach to generate revenue. The seller can either try to market the space on their own, or join a banner ad network, which will recruit advertisers, keep track of earnings, and control banner ad placement on the seller's site.

The most commonly used pricing scheme is the cost-per-thousand-impressions (CPM) model where the cost is associated with the number of exposures of the advertisement. Variants include the cost-per-click (CPC) model where the advertiser pays the publisher each time the banner is clicked on, the cost-per-event (CPE) model, where the advertiser pays the publisher each time a desired event (like a sale) occurs, and various hybrid pricing schemes. The procedure developed in this model is applicable primarily to the CPM pricing model, which is the most widely used model in practice. According to the IAB, the prevalent pricing model shifted from hybrid pricing (39% in the 3rd quarter of 2001 vs. 48% in the 3rd quarter of 2000) to straight CPM (48% in the 3rd quarter of 2001 vs. 41% in the 3rd quarter of 2000). Part of this increase could be a result of the fact that a CPM based model is relatively easier to set up, while having the advantage of not sharing any of the privacy concerns of personalization. Although it is not the focus of this article, it might be possible to use the model presented in this article in a hybrid, CPC or CPE based

| Ad Server | Browser | Publisher |
|---|--|-----------------|
| 5. Calls script to select banner 6. Logs Ad display 7. Returns banner graphic | 1. Request publisher page 3. Decodes HTML; finds link to embedded image 4. Requests image from Ad Server 8. Renders and displays page | 2. Returns page |

Fig. 1. Control/data flow diagram.

pricing scheme by incorporating the probability of a clickthrough into the objective function, thereby maximizing expected revenue. There may not be any advantage to doing this however, as schedulers using different pricing models can use a scheduling model that is best suited to their pricing scheme.

Many advertisements compete for space on a given Web page in any given scheduling horizon. Ads are updated at regular intervals of time and a rectangular slot with advertisements is displayed in each interval. As a result, each of these time intervals represents decisions to be made regarding which advertisements to present to the viewer.

A review of the mechanics of serving up a banner advertisement is provided in Section 2. A description of the specific problem being considered in this paper is given in Section 3 along with related integer programming formulations, while Section 4 provides some information on previous attempts at solving the problem. A description of the solution procedure is presented in Section 5; results from applying the procedure are given in Section 6. Section 7 concludes.

2. BANNER ADVERTISING—TECHNICAL BACKGROUND

A variety of advertisement servers are available in the market. This section provides an overview of the process through which an advertisement is displayed when a web page is requested by the browser. A detailed description can be obtained from Langheinrich et al. [1999].

Together, HTML (Hypertext Markup Language) and HTTP (Hypertext Transfer Protocol) allow the separation of advertisements and page content, thereby enabling dynamic advertisement selection every time a Web page is requested. Langheinrich et al. [1999] summarize the process via Figure 1.

The browser initiates the process (step 1) by requesting a page from the publisher (the provider of advertisement space). The publisher returns the requested page (step 2), which contains a link requesting the Ad Server for an ad. The browser decodes the HTML and identifies this link (step 3). This

initiates a request to the Ad Server for an image (step 4). The Ad Server calls a script that selects the appropriate banner ad to run (step 5). It keeps count of the number of copies of this ad displayed to date by logging the current display (step 6) and returns the selected graphic image (step 7), at which point the browser displays it (step 8).

Note that the contribution of this article will affect only details of the script used in step 5 of Figure 1. The script used in step 5 would take as input the results obtained from the solution of the problem solved in this article and select the appropriate advertisement (or set of advertisements) to display when a particular page is requested. As a consequence, the approach presented here does not require drastic changes to the control or data flow and can be integrated relatively easily into a generic ad serving framework. This is a very convenient feature that could play a key role in the eventual applicability and acceptability of this approach.

3. PROBLEM DESCRIPTION

The formal description of the problem was introduced by Adler et al. [2002] who referred to it as the off-line version of the *ad placement problem*. The set of advertisements to be scheduled is known in advance and the objective is to find the subset of advertisements that will maximize revenues for the party who owns the advertising space.

Adler et al. [2002] characterize advertisements in terms of three parameters—the *access fraction*, a_i of advertisement i , which is the fraction of accesses that actually see the advertisement; the *time fraction*, t_i of advertisement i , which is the fraction of time the ad is displayed to any access that sees the ad, and the *ad geometry*. As done by Adler et al. [2002] and Kumar et al. [2000, 2001a, 2001b], we consider only one-dimensional geometry—this is the most common geometry among banner advertisements. The space available for display is assumed to be fixed.

The time fraction represents the fraction of time that an ad is displayed (when an ad is displayed) for a given user access. When accesses are of short duration, the time fractions tend to be 1, as the advantage to be gained by occasionally updating the ads is negligible. However, when accesses tend to be long, there may be significant advantages to updating the advertisement space over time.

The ad placement problem is formally described as follows. Given a set of time slots T , a slot size S and a set of advertisements A , with each advertisement $i \in A$ characterized by a *size* $s_i \leq S$ and a *weight* $w_i \leq |T|$, find a subset $A' \subseteq A$ such that each advertisement $i \in A'$ is displayed *exactly once* in each of w_i time slots, and such that $\sum_{i \in A'} s_i w_i$ is maximized. A critical characteristic of the feasible subset A' is that *at most one copy of an advertisement can be assigned to a time slot*. This is a necessary feature in Internet advertising—advertisers who have paid for an advertisement to be shown to 100 users would usually find it unacceptable to have two copies of the ad displayed to 50 people. This requirement differentiates it from related problems in scheduling and bin packing.

The pricing scheme is assumed to be proportional to the space needed for a particular advertisement. The CPM is usually proportional to the size of the advertisement, in which case maximizing $\sum_{i \in A} s_i w_i$ is equivalent to maximizing revenue. The time slots are assumed to be of equal duration. This can be done without loss of generality by setting the number of slots $|T|$ to the least common denominator of the access fractions a_i . For example, given $A = \{1, 2, 3\}$ with $a_1 = \frac{1}{2}$, $a_2 = \frac{1}{3}$, $a_3 = \frac{1}{4}$, we can set $|T|$ to be 12, the least common multiple of the denominators. The weights $w_i = |T| \times a_i \Rightarrow w_1 = 6, w_2 = 4, w_3 = 3$. We also assume (as in Adler et al. [2002] and Kumar et al. [2000, 2001a, 2001b]) that the width of the available space is fixed, with all advertisements having the same width, which is usually true in the case of banner advertisements.

The ad placement problem can be formulated as an integer program after defining binary variables x_{ij} to be 1 if advertisement i is assigned to time slot j and 0 otherwise, and y_i to be 1 if advertisement i is assigned to any slot and 0 otherwise. Recall that parameter s_i is the size of advertisement i ; w_i is the number of copies of the advertisement to run (or equivalently, the number of slots the advertisement should feature in) and S is the size of the advertising slot. The ad placement problem can be formulated as (*APP*) below.

$$\max \sum_{i \in A} \sum_{j \in T} s_i x_{ij} \quad \text{s.t.} \quad \sum_{i \in A} s_i x_{ij} \leq S \quad \forall j \in T \quad (1)$$

$$(APP) \quad \sum_{j \in T} x_{ij} = w_i y_i \quad \forall i \in A \quad (2)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in A; \quad \forall j \in T \quad (3)$$

$$y_i \in \{0, 1\} \quad \forall i \in A \quad (4)$$

The objective function maximizes the revenue from scheduling the advertisements. Constraints 1 ensure that the space used in each slot is within that available, while constraints 2 ensure that if advertisement i is assigned to any slot, it is assigned to exactly w_i slots. Note that as a result of constraints 2, the objective of (*APP*), $\sum_{i \in A} \sum_{j \in T} s_i x_{ij} = \sum_{i \in A} s_i (\sum_{j \in T} x_{ij}) = \sum_{i \in A} s_i w_i y_i$. We try both versions of the objective function in our solution approach.

4. RELATED PREVIOUS WORK

(*APP*) can be interpreted as a variant of the standard bin packing problem, with the additional requirement that if a copy of advertisement i is to be placed, then a copy needs to be placed in exactly w_i time slots (constraint 2). While the bin packing problem has been extensively studied [Garey and Graham 1975; Pinedo 1995], the variant represented by (*APP*) has not received much attention. As mentioned earlier, the ad placement problem was introduced in Adler et al. [2002], which showed the problem to be NP-Hard. It also presented heuristic approaches and bounds to solve the problem. While the approaches presented there are efficient and have guaranteed worst-case bounds, Kumar et al. [2000, 2001a, 2001b] note that they are not effective in generating good solutions.

The ad placement problem was tackled using hybrid genetic algorithms in Kumar et al. [2000, 2001a, 2001b]. While their results indicate better solutions relative to the approaches presented in Adler et al. [2002], the improvement often comes at significant computational cost—the times needed for solution often being a few thousand seconds of CPU time on a system running SunOS. The approach presented in this paper obtains solutions that are better than those obtained via either of the procedures above. In addition, it does so with much less computational effort.

5. THE SOLUTION PROCEDURE

This article presents a solution approach based on Lagrangean decomposition. The application of Lagrangean relaxation to integer programming was introduced by Held and Karp [1970, 1971]. Based on these articles an earlier article by Everett [1963] and their own work, Geoffrion [1974] and Fisher [1981] helped establish Lagrangean relaxation as a powerful tool for solving integer programs. A special case of Lagrangean relaxation, often referred to as Lagrangean decomposition, was presented in Guignard and Kim [1987].

In the case of the ad placement problem (*APP*), a direct implementation of Lagrangean decomposition provides results that are very close to the optimal solution. This is a nice feature to have, as the basic Lagrangean decomposition approach is relatively easy to implement. In this section, we provide a brief description of the solution approach. Details of the methodology are not provided here—readers interested in the details are referred to Geoffrion [1974], Fisher [1981] or to books describing this material [Nemhauser and Wolsey 1988; Martin 1999; Schrijver 1986].

5.1 The Lagrangean Decomposition

A new set of variables v_{ij} are introduced to de-link constraint sets 1 and 2. We can re-write the formulation incorporating these variables as

$$\max \sum_{i \in A} \sum_{j \in T} s_i x_{ij} \quad \text{s.t.} \quad x_{ij} - v_{ij} = 0 \quad \forall j \in T; \forall i \in A \quad (5)$$

$$\sum_{i \in A} s_i x_{ij} \leq S \quad \forall j \in T \quad (6)$$

$$(APP1) \quad \sum_{j \in T} v_{ij} = w_i y_i \quad \forall i \in A \quad (7)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in A; \forall j \in T \quad (8)$$

$$v_{ij} \in \{0, 1\} \quad i \in A; \forall j \in T \quad (9)$$

$$y_i \in \{0, 1\} \quad \forall i \in A \quad (10)$$

Formulations (*APP*) and (*APP1*) are clearly equivalent. Note that removing constraint set 5 decomposes (*APP1*) into independent subproblems. Let the equivalent formulation with the alternative objective function ($\max \sum_{i \in A} s_i w_i y_i$) be (*APP2*).

Consider the Lagrangean dual of $(APP1)$ obtained by dualizing constraint set 5 using dual multipliers $\lambda_{ij} \in \mathfrak{R}$, given by $(APP1L)$ below.

$$Z_L(\lambda) = \max \sum_{i \in A} \sum_{j \in T} s_i x_{ij} + \sum_{i \in A} \sum_{j \in T} \lambda_{ij} x_{ij} - \sum_{i \in A} \sum_{j \in T} \lambda_{ij} v_{ij}$$

$$\text{s.t. } \sum_{i \in A} s_i x_{ij} \leq S \quad \forall j \in T \quad (11)$$

$$(APP1L) \quad \sum_{j \in T} v_{ij} = w_i y_i \quad \forall i \in A \quad (12)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in A; \forall j \in T \quad (13)$$

$$v_{ij} \in \{0, 1\} \quad \forall i \in A; \forall j \in T \quad (14)$$

$$y_i \in \{0, 1\} \quad \forall i \in A \quad (15)$$

If the alternative objective function ($\max \sum_{i \in A} s_i w_i y_i$) is used for $(APP1)$, the objective function of $(APP1L)$ should in turn be to maximize $\sum_{i \in A} s_i w_i y_i + \sum_{i \in A} \sum_{j \in T} \lambda_{ij} x_{ij} - \sum_{i \in A} \sum_{j \in T} \lambda_{ij} v_{ij}$. The best upper bound is then obtained by solving

$$Z_L = \min_{\lambda \in \mathfrak{R}} Z_L(\lambda)$$

5.2 The Subproblems

Formulation $(APP1L)$ decomposes into two independent subproblems, represented by $(SUB1)$ and $(SUB2)$ below.

$$\max \sum_{i \in A} \sum_{j \in T} (s_i + \lambda_{ij}) x_{ij}$$

$$(SUB1) \quad \text{s.t. } \sum_{i \in A} s_i x_{ij} \leq S \quad \forall j \in T \quad (16)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in A; \forall j \in T \quad \max - \sum_{i \in A} \sum_{j \in T} \lambda_{ij} v_{ij} \quad (17)$$

$$(SUB2) \quad \text{s.t. } \sum_{j \in T} v_{ij} = w_i y_i \quad \forall i \in A \quad (18)$$

$$v_{ij} \in \{0, 1\} \quad \forall i \in A; \forall j \in T \quad (19)$$

$$y_i \in \{0, 1\} \quad \forall i \in A \quad (20)$$

Both $(SUB1)$ and $(SUB2)$ decompose further—for each time slot $j \in T$, there is a subproblem of the form

$$\max \sum_{i \in A} (s_i + \lambda_{ij}) x_{ij} \quad (SUB1_j) \quad \text{s.t. } \sum_{i \in A} s_i x_{ij} \leq S \quad (21)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in A \quad (22)$$

while for each advertisement $i \in A$, we get a subproblem of the form

$$(SUB2_i) \quad \max - \sum_{j \in T} \lambda_{ij} v_{ij} \quad \text{s.t.} \quad \sum_{j \in T} v_{ij} = w_i y_i \quad (23)$$

$$v_{ij} \in \{0, 1\}, \quad \forall j \in T \quad (24)$$

$$y_i \in \{0, 1\} \quad (25)$$

$(SUB1_j)$ is a binary knapsack problem. Various specialized solution procedures exist to solve the binary knapsack problem [Balas and Zemel 1980; Pisinger 1997; Martello et al. 2000]. The continuous relaxation of $(SUB1_j)$ is solved in order to expedite the solution process. If $\sum_{i \in A} s_i w_i y_i$ is used as the objective function, the objective function of $(SUB1_j)$ should in turn be to maximize $\sum_{i \in A} \lambda_{ij} x_{ij}$.

Solving $(SUB2_i)$ is much easier—sort the variables v_{ij} in descending order of $(-\lambda_{ij})$ and compute $f_i = -\sum_{j \in A_i} \lambda_{ij}$ (if the objective used is to maximize $\sum_{i \in A} s_i w_i y_i$, $f_i = s_i w_i - \sum_{j \in T_i} \lambda_{ij}$), where T_i is the set consisting of the first w_i variables in the sorted list. If $f_i > 0$, the optimal solution is $v_{ij} = 1 \forall j \in T_i$, $v_{ij} = 0 \forall j \notin T_i$ and $y_i = 1$. If $f_i \leq 0$, the optimal solution is $v_{ij} = 0 \forall j \in T$ and $y_i = 0$.

Subgradient optimization [Bazaraa et al. 1993] is used to optimize the Lagrangean dual. Feasible solutions to (APP) are obtained at each iteration through a heuristic embedded in the subgradient optimization procedure. The feasible solution is updated whenever a better solution is obtained and the best feasible solution obtained across all the iterations is reported when the algorithm terminates.

The heuristic used to generate feasible solutions is greedy in nature, and can be described as below.

- (1) Sort the ads in decreasing order based on f_i , the solutions to $(SUB2_i)$ for each ad $i \in A$.
- (2) Start with the first ad in the sorted list.
- (3) Check whether assigning this advertisement to the w_i time slots recommended by $(SUB2_i)$ is feasible (i.e., check whether assigning this ad will keep the space used in each of the w_i slots within the available space S). If yes, assign this ad to the appropriate time slots.
- (4) Remove this ad from the sorted list.
- (5) Repeat Steps 2 and 3 until the sorted list is empty.

5.3 A Note on the Performance of Commercial Optimization Packages

Commercial integer programming packages rely on branch and bound based techniques for solution. It is well established that symmetry in integer programs makes them significantly harder to solve [Barnhart et al. 1998]. Symmetry arises when there are alternative solutions to the problem that are effectively equivalent. Identical time slots induce symmetry in the formulations presented for the ad placement problem—while the effective solution does

not change as a result of relabeling the time slots, a branch and bound based solution technique will have to go through a large number of the permutations of a solution before termination. This can mire branch and bound based solvers by burdening them to explore and eliminate such alternative symmetric solutions. As noted in Menon and Schrage [2002], avoiding (or exploiting) symmetry can lead to significant gains through the elimination of unnecessary enumeration. For instance, it might be possible to avoid considering a solution (x_b, x_a) for two identical blocks if we have already considered the (effectively) identical solution (x_a, x_b) .

This is corroborated even when solving relatively small problems in a commercial IP solver. For instance, when one of the data sets from the case study in Kumar et al. [2000, 2001a, 2001b] is solved in CPLEX [ILOG, Inc. 2001] and LINDO [2001], branch and bound proceeds for hours without termination. Indeed, even after over two hours, the feasible solution obtained was still drastically worse than the one obtained through the procedure presented in this paper. This observation is not specific to this example—this general behavior was observed on all data sets tested. Kumar et al. [2000, 2001a, 2001b] have also pointed out this fact. Consequently, there is no additional benefit of reporting the individual solution times using a commercial integer programming solver. Indeed, it is worth noting that most of the data sets used in this paper are intractable via a direct application of branch and bound.

6. RESULTS

The Lagrangean decomposition based solution procedure was applied to the four data sets from the case study reported in Kumar et al. [2000, 2001a, 2001b] and to additional data sets that were generated randomly. Both versions of the objective function for problem (*APP*) were tested. Results are presented here for the case study and the randomly generated data sets under both objective functions. The largest data sets generated had $|A| = 200$, $|T| = 200$ and $S = 100$, which is larger than the largest set used in Kumar et al. [2000, 2001a, 2001b]. All the computational experiments were conducted on an Intel Pentium II running Windows NT at 450 MHz.

6.1 Results from Case Study Data

Kumar et al. [2000, 2001a, 2001b] report results from four case studies, the data for one of which is provided in their paper. Readers are directed to this article for details on the case studies. The case studies involved 48 advertisements ($|A| = 48$) and 180 time slots ($|T| = 180$). The advertisement slot had a length of 49 units. Table I compares the results from applying the Lagrangean decomposition approach to these four data sets with the results from the approaches of Adler et al. [2002] and Kumar et al. [2000, 2001a, 2001b]. The gaps are calculated based on the bound obtained from the Lagrangean decomposition, which in all four cases was equal to 8820 (essentially, 49×180).

The gaps were obtained as $(\frac{UB-LB}{UB}) \times 100$ where UB is 8820 and LB is the best feasible objective function value obtained. It is worth pointing out that

Table I. Results from Case Study Data

| Case Study | Adler et al. | | Kumar et al. | | Lag. Decomp. | | |
|------------|--------------|---------|--------------|---------|--------------|---------|------------|
| | Obj. | Gap (%) | Obj. | Gap (%) | Obj. | Gap (%) | Time (sec) |
| Set 1 | 7368 | 19.71 | 7813 | 12.89 | 8716 | 1.19 | 135 |
| Set 2 | 7260 | 21.49 | 8461 | 4.24 | 8640 | 2.08 | 164 |
| Set 3 | 6609 | 33.45 | 8154 | 8.17 | 8709 | 1.27 | 161 |
| Set 4 | 7688 | 14.72 | 7996 | 10.31 | 8802 | 0.20 | 129 |

these gaps are over-estimates of the actual gaps (relative to the best integral solution), and that the actual gaps could be much smaller. In all four cases, the formulation using $\max \sum_{i \in A} s_i w_i y_i$ as the objective function gave the better feasible solution, while the other formulation gave the better bound. This pattern was observed with the generated data as well. The best objective function value was selected from the two versions of the Lagrangean decomposition approach; the solution time reported for each set is the total time taken to solve both formulations (each formulation took approximately 73 seconds on average).

As can be seen, the Lagrangean decomposition based approach provided the best solutions in all four cases. In addition, it did so in a reasonable amount of time (solution times were not available for the other two approaches). The average gap was 1.185% for the Lagrangean decomposition approach. This is a significant improvement over the existing approaches presented in both Adler et al. [2002] and Kumar et al. [2000, 2001a, 2001b], where the average gaps were 22.34% and 8.90% respectively.

6.2 Results from Generated Data

Data was generated randomly to further analyze the performance of the approach presented in this paper. 20 categories were generated based on the values of $|A|$, $|T|$ and S . Given the number of advertisements $|A|$, the number of times slots $|T|$ and the size of the advertising space S , the size of each advertisement $s_i \forall i \in A$ is generated from a uniform distribution $U[10, 40]$, while the required number of slots $w_i \forall i \in A$ is generated from a uniform distribution $U[10, 30]$. Ten instances were generated of each of twenty combinations of $|A|$, $|T|$ and S . These are the broadest set of test data used to date, with the larger data sets being significantly larger than the largest sets used in any previous work.

Results from applying each of the formulations are given in Table II. In general, the first formulation seems to give a better bound while the second one seems to give a better feasible solution. The gaps for both formulations are reasonable. The gaps for each formulation are obtained as $(\frac{UB-LB}{UB}) \times 100$, where UB is the upper bound from the Lagrangean decomposition and LB is the objective function value of the best feasible solution. The gaps and solution times reported for each formulation are averages over the 10 data sets of each of the 20 types.

Given that the solution times are not very high, it is reasonable to combine the results from the two relaxations. Using the best bound and the best feasible solution from the two approaches, we get the results presented in the last column of Table II. Specifically, if f_1 and f_2 are the values of the best feasible

Table II. Results from Generated Data

| Parameters | | | Obj: $\max \sum_{i \in A} \sum_{j \in T} s_i x_{ij}$ | | Obj: $\max \sum_{i \in A} s_i w_i y_i$ | | Combined Gap % |
|------------|-----|-----|--|------------|--|------------|-------------------|
| A | T | S | Gap ($\frac{UB-LB}{UB}$) % | Time (sec) | Gap ($\frac{UB-LB}{UB}$) % | Time (sec) | |
| 20 | 40 | 50 | 4.88 | 5.5 | 4.04 | 7.2 | 4.02 |
| 40 | 40 | 50 | 3.82 | 10.1 | 2.26 | 15.6 | 2.12 |
| 60 | 40 | 50 | 3.99 | 14.8 | 2.25 | 23.6 | 2.05 |
| 80 | 40 | 50 | 2.60 | 19.4 | 2.03 | 31.9 | 2.02 |
| 100 | 40 | 50 | 2.90 | 24.4 | 1.74 | 40.5 | 1.63 |
| 60 | 20 | 50 | 4.44 | 5.6 | 7.98 | 10.4 | 4.44 |
| 60 | 40 | 50 | 3.99 | 14.8 | 2.25 | 23.6 | 2.05 |
| 60 | 60 | 50 | 4.24 | 22.4 | 3.09 | 37.2 | 2.77 |
| 60 | 80 | 50 | 4.40 | 32.0 | 3.34 | 50.6 | 2.23 |
| 60 | 100 | 50 | 4.40 | 40.9 | 3.73 | 64.6 | 1.98 |
| 60 | 40 | 40 | 4.93 | 14.0 | 3.78 | 23.6 | 3.78 |
| 60 | 40 | 60 | 3.81 | 15.2 | 2.40 | 23.7 | 2.34 |
| 60 | 40 | 80 | 3.67 | 15.6 | 4.23 | 23.9 | 3.67 |
| 60 | 40 | 100 | 3.36 | 16.1 | 2.32 | 24.0 | 2.27 |
| 60 | 40 | 120 | 3.06 | 16.5 | 3.01 | 24.0 | 2.99 |
| 100 | 100 | 100 | 2.76 | 72.6 | 3.81 | 113.3 | 1.09 |
| 100 | 150 | 100 | 2.67 | 116.3 | 4.90 | 177.3 | 1.46 |
| 150 | 150 | 100 | 2.69 | 171.0 | 4.95 | 271.9 | 0.80 |
| 150 | 200 | 100 | 2.66 | 231.7 | 6.31 | 373.8 | 1.12 |
| 200 | 200 | 100 | 2.54 | 303.9 | 7.08 | 497.3 | 0.73 |

solutions obtained when using (*APP1*) and (*APP2*) respectively, and if b_1 and b_2 are the values of the best upper bounds obtained when using (*APP1*) and (*APP2*) respectively, then the combined gap is obtained as

$$\frac{\min\{b_1, b_2\} - \max\{f_1, f_2\}}{\max\{f_1, f_2\}}.$$

These gaps are between 0.73% and 4.44% (with an average of 2.28%), which is very good. As pointed out earlier, these gaps overestimate the actual gaps which could be much smaller. Additionally, the gaps seem to improve with problem size.

7. CONCLUSION

This paper extends an important but lightly researched problem on scheduling banner advertisements on the Web. Commercial branch and bound based integer programming solvers perform very poorly on this problem, partly due to the level of symmetry present in the model. A Lagrangean decomposition based solution procedure is presented, along with results from both a set of case studies and randomly generated data. The results indicate that this approach performs significantly better than previous approaches used to solve this problem, both in terms of solution quality and solution time. In addition, the approach presented here would require minimal alterations to any ad server setup and therefore can easily be incorporated into the overall implementation scheme.

ACKNOWLEDGMENTS

We thank Drs. S. Kumar, V. Jacob and C. Sriskandarajah for providing us the additional case study data sets and the objective function values for these data sets when solved via the procedures of Adler et al. [2002] and Kumar et al. [2000, 2001a, 2001b].

REFERENCES

- ADLER, M., GIBBONS, P., AND MATIAS, Y. 2002. Scheduling Space Sharing for Internet Advertising. *J. Sched.* 5, 2, 103–119.
- BALAS, E. AND ZEMEL, E. 1980. An Algorithm for Large Zero–One Knapsack Problems. *Oper. Res.* 28, 1130–1154.
- BARNHART, C., JOHNSON, E., NEMHAUSER, G., SAVELSBERGH, M., AND VANCE, P. 1998. Branch and Price : Column Generation for Solving Huge Integer Programs. *Oper. Res.* 46, 316–329.
- BAZARAA, M., SHERALI, H., AND SHETTY, C. 1993. *Nonlinear Programming Theory and Algorithms*. John Wiley & Sons, New York, NY.
- BUCHWALTER, C., RYAN, M., AND MARTIN, D. 2001. The State of Online Advertising: Data Covering Fourth Quarter 2000. Tech. rep., AdRelevance (A Jupiter Media Metrix company). February.
- EVERETT III, H. 1963. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Oper. Res.* 11, 399–417.
- FISHER, M. 1981. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science* 27, 1–18.
- GAREY, M. AND GRAHAM, R. 1975. Bounds for Multiprocessor Scheduling with Resource Constraints. *SIAM J. Comput.* 4, 2, 187–200.
- GEOFFRION, A. 1974. Lagrangean Relaxation for Integer Programming. *Mathematical Programming Study* 2, 82–114.
- GUIGNARD, M. AND KIM, S. 1987. Lagrangean Decomposition: a Model Yielding Stronger Lagrangean Bounds. *Mathematical Programming* 39, 215–228.
- HANEY, K. 2001. Online Ad Spending Rakes in \$2 Billion. *eBiz Daily*, April 06, 2001. http://www.digitrends.net/ebna/index_11402.html.
- HELD, M. AND KARP, R. 1970. The Traveling Salesman Problem and Minimum Spanning Trees. *Oper. Res.* 18, 1138–1162.
- HELD, M. AND KARP, R. 1971. The Traveling Salesman Problem and Minimum Spanning Trees: Part II. *Mathematical Programming* 1, 6–25.
- ILOG, Inc. 2001. *ILOG CPLEX 7.5 User's Manual*. ILOG, Inc., Gentilly, France.
- Interactive Advertising Bureau 2001. Internet Ad Revenue Report. Fourth Quarter of 2000, April, 2001. <http://www.iab.net/forms/qreport.html>.
- KUMAR, S., JACOB, V., AND SRISKANDARAJAH, C. 2000. Scheduling Advertising at a Web Site. In *10th Annual Workshop On Information Technologies and Systems*. Sydney, Australia.
- KUMAR, S., JACOB, V., AND SRISKANDARAJAH, C. 2001a. Hybrid Genetic Algorithms for Scheduling Advertising on a Web Page. In *Proceedings of the Twenty-Second International Conference on Information Systems (ICIS)*. New Orleans, LA.
- KUMAR, S., JACOB, V., AND SRISKANDARAJAH, C. 2001b. Scheduling Advertisements on a Web Page to Maximize Space Utilization. Working Paper. School of Management, The University of Texas at Dallas.
- LANGHEINRICH, M., NAKAMURA, A., ABE, N., KAMBA, T., AND KOSEKI, Y. 1999. Unintrusive Customization Techniques for Web Advertising. In *8th International World Wide Web Conference*. Toronto, Canada, 181–194.
- LINDO Systems, Inc. 2001. *LINDO Callable Library User's Manual*. LINDO Systems, Inc.
- MARTELLO, S., PISINGER, D., AND TOTH, P. 2000. New Trends in Exact Algorithms for the 0–1 Knapsack Problem. *Europ. J. Oper. Res.* 123, 325–332.
- MARTIN, R. 1999. *Large Scale Linear and Integer Programming : A Unified Approach*. Kluwer Academic Press, Boston, MA.
- MENON, S. AND SCHRAGE, L. 2002. Order Allocation for Stock Cutting in the Paper Industry. *Oper. Res.* 50, 2, 324–332.

- NEMHAUSER, G. AND WOLSEY, L. 1988. *Integer and Combinatorial Optimization*. Wiley Interscience Series in Discrete Mathematics and Optimization, New York, New York.
- PINEDO, M. 1995. *Scheduling Theory, Algorithms and Systems*. Prentice Hall, New Jersey.
- PISINGER, D. 1997. A Minimal Algorithm for the 0–1 Knapsack Problem. *Oper. Res.* 45, 758–767.
- SCHRIJVER, A. 1986. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, NY.

Received August 2002; revised January 2003, May 2003; accepted May 2003