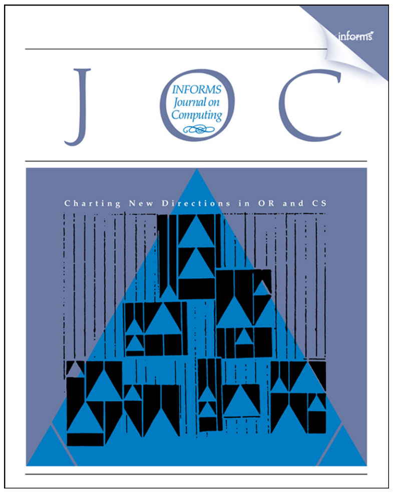

INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Scheduling Banner Advertisements on the Web

Syam Menon, Ali Amiri,

To cite this article:
Syam Menon, Ali Amiri, (2004) Scheduling Banner Advertisements on the Web. INFORMS Journal on Computing 16(1):95-105.
https://doi.org/10.1287/ijoc.1020.0003

INFORMS Journal on Computing
Vol. 16, No. 1, Winter 2004, pp. 95–105
ISSN 0899-1499 | EISSN 1526-5528 | 04 | 1601 | 0095

DOI 10.1287/ijoc.1020.0003
© 2004 INFORMS

Scheduling Banner Advertisements on the Web

Syam Menon
School of Management, University of Texas at Dallas, Richardson, Texas 75083, USA, syam@utdallas.edu

Ali Amiri
College of Business Administration, Oklahoma State University, Stillwater, Oklahoma 74078, USA, amiri@okstate.edu

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% of the total revenue. There are considerable gains to be realized through the efficient scheduling of banner advertisements. This problem has received only limited attention in the literature. This paper introduces formulations for an important version of the banner advertisement scheduling problem. Two solution approaches, one based on Lagrangean decomposition and the other based on column generation, are presented, along with extensive results based on 1,500 randomly generated data sets. These results suggest that, while both approaches do very well in general, column generation consistently performs better than Lagrangean decomposition, giving gaps of 0.0004% or better in 4.4 seconds or less, even on relatively large problem instances.

Key words: Internet; banner advertising; scheduling; Lagrangean decomposition; column generation
History: Accepted by Ramayya Krishnan; received June 2001; revised June 2002, September 2002; accepted October 2002.

1. Introduction

According to the Interactive Advertising Bureau (2001), Internet advertising revenue for the United States was $8.2 billion in the year 2000. It also notes that approximately 47% of the advertisements viewed were in banner form. Buchwalter et al. (2001) mention that 99% 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 version of the banner advertisement placement problem was introduced in Adler et al. (2002). They had a patent application approved in 1999, which underscores the practical relevance of the problem at hand. It was explored further in Kumar et al. (2000) and Amiri and Menon (2001). In each of these papers, the customers were allowed to specify the number of copies of an advertisement to be displayed, and all selected advertisements had to be run exactly that number of times. This version of the problem is valid when the length of the scheduling horizon for the advertisement (the period during which the advertising space provider has to run the requested number of copies of an advertisement) is the same as the length of the planning horizon (the period for which a particular schedule is generated). In a large number of realistic scheduling instances however, planning horizons tend to be much shorter than scheduling horizons and this requirement becomes far more restrictive than necessary.

1.1. Scheduling Horizons and Planning Horizons

Advertisers usually require that a specified number of copies of an advertisement be displayed over a specified period of time. That is, the start and end dates for a particular advertisement need to be specified as part of the *insertion order* (the order requesting a particular ad to be scheduled, along with the associated details of the request). Often, the period between the start date and end date (i.e., the scheduling horizon) is of the order of a few months. For example, *E-Commerce Times* (`www.ecommercetimes.com`), a free on-line publication with daily news and feature articles on electronic commerce, charges between $55 and $150 as the cost per thousand exposures (CPM) and requires a minimum purchase of 50,000 impressions per month (`www.ecommercetimes.com/ad_info`). The scheduling horizon becomes even longer in the presence of volume/time discounts that many providers of advertising space offer: Several Web sites allow for the selection of scheduling horizons that are 12 months long. For instance, SourceCode.com (`sourcecode2.bitnetworks.com`), a free service-based site offering utilities, training, etc., charges a CPM of $100 for 100,000 impressions in one month. The CPM drops to $90 for a 6-month scheduling horizon, and it drops further to \$85 for a 12-month horizon (`sourcecode2.bitnetworks.com/newdesign/advertisingsales.cfm`). Indeed, Buchwalter et al. (2001) note that the number of weeks an advertisement ran in the fourth quarter of 2000 ranged from 1 to 13 weeks, with the average being 4.3 weeks. This average is an underestimate of the average scheduling horizon, as the measure reported considers only the fourth quarter of 2000: It is very likely that many advertisements scheduled prior to this quarter would have terminated in this quarter while many advertisements scheduled in this quarter (or prior to it) were scheduled to terminate in a future quarter (only the weeks during which the advertisement ran in the fourth quarter of 2000 are considered in calculating the reported average).

Schedules based on extremely long planning horizons will be very inflexible and it is not in the interest of the provider of advertisement space to use long planning horizons. Flexibility is often indispensable in a Web-based environment and given this need for flexibility, the planning horizon cannot be as long as these scheduling horizons: Scheduling advertisements months in advance essentially makes the setup unnecessarily rigid. For instance, it might be worth accommodating a particularly lucrative customer who shows up at short notice, but scheduling too far ahead could prevent one from doing so. Shorter planning horizons are essential to retain flexibility in scheduling, with the space provider keeping track of the number of impressions of an advertisement displayed to date. The maximum number of copies of an advertisement available for display in any planning horizon is just the number of copies of the advertisement remaining to be displayed. For example, consider planning for April 1 (a planning horizon of 1 day) for a customer who requires 10,000 copies of an advertisement to be displayed between March 1 and June 1. Suppose 2,500 copies of this ad have been displayed already (between March 1 and March 31). On April 1, there are 7,500 copies remaining to be displayed, but not all 7,500 need to be displayed on that day: There are two more months remaining in this advertisement's scheduling horizon. Therefore, the constraint for this ad, for the April 1 planning period should say that *at most* 7,500 copies should be scheduled (rather than exactly 7,500 copies). If 500 copies are scheduled for April 1, the number remaining to be scheduled drops to 7,000 (which is the number used in the next planning period).

This paper develops formulations and solution approaches for the banner advertisement scheduling problem when planning horizons are smaller than scheduling horizons. One of the formulations is seen to be amenable for solution via Lagrangean decomposition; the other is tractable via column generation. The column generation approach is observed to be extremely efficient, providing optimal or near-optimal solutions in 4.4 seconds or less on a PC with 512 MB of RAM running Windows NT on an Intel Pentium II 450 MHz processor. While variants of the problem presented in this paper have been known to the research community, the exact problem being studied here has not been dealt with previously, to the best of our knowledge.

A description of the problem is given in §2 along with information on previous attempts at solving related problems. The Lagrangean decomposition based procedure is presented in §3 while the column generation based approach is described in §4. The results from both approaches are furnished in §5, and conclusions are in §6.

2. Problem Description and Related Previous Work

The off-line advertisement placement problem was introduced in Adler et al. (2002), who characterized Web 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*. We consider one-dimensional geometry, which is the most common among banner advertisements. The set of advertisements to be scheduled is known in advance, and the objective is to find the subset of advertisements that will maximize revenue for the owner of the advertising space.

The ad placement problem from Adler et al. (2002) 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. As already mentioned, long planning horizons are often impractical and the requirement that the advertisements in A' have to be displayed in exactly w_i slots could be unnecessarily restrictive. When shorter planning horizons are used, it is more appropriate to alter this requirement to say that each advertisement $i \in A'$ can be displayed once in *at most* w_i time slots. As explained in §1, w_i can then be considered to be the number of copies of an advertisement remaining to be displayed.

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 Web 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 time slots are assumed to be of equal duration. This can be done without loss of generality (Adler et al. 2002). We also assume 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. In addition, the pricing scheme is assumed proportional to the space needed for a particular advertisement. This is consistent with the CPM model, which is a common model used in practice.

The banner advertisement scheduling problem can be formulated as the integer program (*BAS*) below, after defining binary variables x_{ij} to be 1 if advertisement i is assigned to slot j and 0 otherwise. Parameter s_i is the size of advertisement i; w_i is the maximum number of copies of the advertisement to run (or equivalently, the maximum number of slots the advertisement should feature in); and S is the size of the advertising slot.

$$(BAS)\qquad \begin{aligned} \max\quad & \sum_{i\in A}\sum_{j\in T} s_i x_{ij} \\ \text{s.t.}\quad & \sum_{i\in A} s_i x_{ij} \le S \quad \forall j \in T && (1)\\ & \sum_{j\in T} x_{ij} \le w_i \quad \forall i \in A && (2)\\ & x_{ij} \in \{0,1\} \quad \forall i \in A;\ \forall j \in T && (3) \end{aligned}$$

The objective function maximizes the revenue from scheduling the advertisements. If the prices are not proportional to the size of the advertisement, or if price discounts have been applied to some advertisements, the appropriate objective function would be to maximize $\sum_{i\in A}\sum_{j\in T} p_i x_{ij}$, where p_i is the price per copy of advertisement i. This does not affect the solution approaches presented in this paper in any way. Constraints 1 ensure that the space used in each slot is within that available, while constraints 2 guarantee that advertisement i is assigned to at most w_i slots. The binary requirement on the x_{ij} variables (constraints 3) ensure that at most one copy of each advertisement is allocated to each slot. We note that this formulation is a variant of the model solved in Adler et al. (2002), Kumar et al. (2000), and Amiri and Menon (2001): In the model presented in those papers, constraint 2 is altered to $\sum_{j\in T} x_{ij} = w_i y_i\ \forall i \in A$, where y_i is a binary variable indicating whether or not advertisement i is selected for display.

The banner advertisement scheduling problem can be interpreted as a variant of the bin-packing problem. While the bin-packing problem has been studied extensively (Garey and Graham 1975, Pinedo 1995), the variant presented here has not received much attention. Adler et al. (2002) have shown that the original version of the ad placement problem is NP-hard. When the w_i values are all equal to 1, problem (*BAS*) reduces to the multiple subset-sum problem, which is known to be NP-hard (Caprara et al. 2000). The objective of the multiple subset-sum problem is to find the subset of items (from a given set of N items, each with weight w_i) of maximum total weight that can be packed into M bins of identical capacity c. If actual prices p_i are used instead of the sizes s_i in the objective function, this is a generalization of the multiple knapsack problem, which is well known to be NP-hard. The objective of the multiple knapsack problem is to find the subset of items (from a given set of N items, each with profit p_i and weight w_i) of maximum total weight that can be packed into M bins of various capacities c_j. Given that problem (*BAS*) is a generalization of the problems above, it is also NP-hard.

Adler et al. (2002) presented heuristic approaches with guaranteed worst-case bounds to solve the original version of the ad-placement problem. Kumar et al. (2000) solved it using hybrid genetic algorithms, while Amiri and Menon (2001) solved it via Lagrangean decomposition. As far as we are aware, the modification presented in this paper has not been tackled in the literature.

3. The Lagrangean Decomposition-Based Solution Procedure

A brief description of a solution approach based on Lagrangean decomposition is presented in this section. Held and Karp (1970, 1971) introduced the application of Lagrangean relaxation to integer programming, while Geoffrion (1974) and Fisher (1981) helped establish it as a powerful tool for solving integer programs. A special case of Lagrangean relaxation, called *Lagrangean decomposition*, was presented in Guignard and Kim (1987). Generic details of the methodology are not provided here: Readers interested in the details are referred to Geoffrion (1974), Fisher (1981), Guignard and Kim (1987), and to books describing related material (Nemhauser and Wolsey 1988, Martin 1999, Schrijver 1986).

3.1. The Lagrangean Decomposition

A new set of variables v_{ij} are introduced to delink constraint sets 1 and 2 in (*BAS*). We can re-write (*BAS*) after incorporating these variables as

$$\max \quad \sum_{i\in A}\sum_{j\in T} s_i x_{ij}$$

$$
(BASa)\quad \begin{array}{llr}
\text{s.t.} & x_{ij} - v_{ij} = 0 \quad \forall i \in A;\ \forall j \in T & (4)\\
& \sum_{i\in A} s_i x_{ij} \le S \quad \forall j \in T & (5)\\
& \sum_{j\in T} v_{ij} \le w_i \quad \forall i \in A & (6)\\
& x_{ij} \in \{0,1\} \quad \forall i \in A;\ \forall j \in T & (7)\\
& v_{ij} \in \{0,1\} \quad \forall i \in A;\ \forall j \in T & (8)
\end{array}
$$

Formulations (*BAS*) and (*BASa*) are clearly equivalent. Note that removing constraint set 4 decomposes (*BASa*) into independent subproblems.

Consider the Lagrangean dual of (*BASa*) obtained by dualizing constraint set 4 using dual multipliers $\lambda_{ij} \in \Re$, given below.

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

(*BASaL*) s.t. Constraints 5, 6, 7, and 8

The best bound is then obtained by solving

$$
Z_L = \min_{\lambda\in\Re} Z_L(\lambda).
$$

3.2. The Subproblems

(*BASaL*) decomposes into $|T| \times |A|$ independent subproblems, represented by ($SUB1_j$) and ($SUB2_i$) below. For each time slot $j \in T$, there is a subproblem of the form

$$
(SUB1_j)\quad \begin{array}{ll}
\max & \sum_{i\in A} (s_i + \lambda_{ij}) x_{ij}\\
\text{s.t.} & \text{Constraints 5 and 7}
\end{array}
$$

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

$$
(SUB2_i)\quad \begin{array}{ll}
\max & -\sum_{j\in T} \lambda_{ij} v_{ij}\\
\text{s.t.} & \text{Constraints 6 and 8}
\end{array}
$$

($SUB1_j$) is a binary knapsack problem. Specialized solution procedures exist to solve the binary knapsack problem (Balas and Zemel 1980, Pisinger 1997, Martello et al. 2000). We solve the continuous relaxations of the binary knapsack problems in order to expedite the solution process.

Solving ($SUB2_i$) is much easier: Sort the variables v_{ij} in descending order of $-\lambda_{ij}$ and go down the list setting variables v_{ij} to 1 until either the value of $-\lambda_{ij}$ becomes nonpositive, or $\sum_{j\in T} v_{ij} = w_i$. The cumulative value of $-\lambda_{ij}$ at this point gives us the optimal objective function value for ($SUB2_i$).

Subgradient optimization (Bazarra et al. 1993) is used to optimize the Lagrangean dual. Feasible solutions to problem (*BAS*) 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 is described below.

1. Sort ads in decreasing order based on 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 time slots recommended by ($SUB2_i$) is feasible (i.e., check whether assigning this ad will keep the space used in each of the suggested 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, 3, and 4 until the sorted list is empty.

4. The Column Generation-Based Solution Procedure

Column generation-based formulations of integer programs date back to the early 1960s (Dantzig and Wolfe 1960). Perhaps the best-known example of column generation is that of the cutting-stock problem (Gilmore and Gomory 1961, 1963), where columns are obtained as feasible solutions to an integer program. Before describing the solution procedure, we introduce the column generation based reformulation for the banner-advertisement problem below.

$$
(CBAS)\quad \begin{array}{llr}
\max & \sum_{k\in K} r_k y_k & \\
\text{s.t.} & \sum_{k\in K} a_{ik} y_k \le w_i \quad \forall i \in A & (9)\\
& \sum_{k\in K} y_k \le |T| & (10)\\
& y_k \in \mathscr{Z}^+ \quad \forall k \in K & (11)
\end{array}
$$

Each column represents a feasible assignment of advertisements to a time slot. Parameter a_{ik} is 1 if advertisement i is present in column k and 0 otherwise, while $r_k = \sum_{i\in A} s_i a_{ik}$ is the revenue from the feasible assignment represented by column k. The index set K represents the set of all feasible columns. The objective maximizes the total revenue. Constraints 9 state that across all the columns, at most w_i copies of advertisement i can be scheduled. Constraint 10 states that we cannot exceed the number of time slots available. Variable y_k is the number of copies of column k used. Note that all feasible columns k in formulation (*CBAS*) have to satisfy the knapsack inequality

$$
\sum_{i\in A} s_i a_{ik} \le S.
$$

4.1. Linear Programming Column Generation

In general, formulation (*CBAS*) involves a large number of columns, and it is impractical to include every feasible column into the formulation. Most column generation procedures use the following standard approach.

1. Solve a *Restricted Linear Programming Master Problem*. This restricted master problem is identical to the linear programming relaxation of (*CBAS*), except that it has only a very small subset of the columns.

2. Solve a *Pricing Subproblem* to identify columns that could improve the objective function value of the master problem. In the case of problem (*CBAS*), the dual variables $\mu_i \geq 0$ on constraints 9 and $\nu \geq 0$ on constraint 10 obtained from Step 1 can be used to price out columns that have a positive reduced cost. A column with a positive reduced cost exists if and only if the column generation subproblem below has a positive objective function value.

$$
\begin{array}{llll}
 & \max & \sum_{i\in A}(s_i-\mu_i)z_i-\nu & \\
(CSUB) & \text{s.t.} & \sum_{i\in A} s_i z_i \leq S & (12)\\
 & & z_i \in \{0,1\} \quad \forall i \in A & (13)
\end{array}
$$

If columns price out with positive reduced costs, they are added to the restricted master problem that is then re-solved to obtain new dual prices. We iterate back and forth between the restricted master problem and the subproblem until no columns price out with a positive reduced cost. At this point, we conclude with the LP optimal solution.

The master problem is solved using the LINDO Systems (2001) Callable Library. The column generation subproblem is solved via a simple sorting heuristic while possible. When the heuristic fails to generate useful columns, we resort to an exact solution approach. (*CSUB*) has been implemented using the LINDO Systems (2001) Callable Library, but we note that a quicker approach like that of Pisinger (1997) could expedite the solution process. However, given the sizes of the subproblems, and in light of the solution times presented in §5, these gains are bound to be minimal.

4.2. Other Relevant Issues in Column Generation

It is a well-known fact that the standard trick of switching on the integrality requirements after arriving at the LP column generation optimum may not provide us the optimal integral solution. Various approaches based on column generation within branch and bound exist (Vance 1998, Vanderbeck 1994). Both of these approaches, as well as that of Degraeve and Schrage (1999), can be adapted to solve formulation (*CBAS*) to optimality. The results indicate that this is unwarranted, however, as the standard approach works extremely well, especially if a few judiciously selected columns are added a priori. The gaps between the column generation LP bound and feasible solutions obtained are extremely small: Often, the column generation LP objective function value ends up being equal to the objective function value associated with the feasible integral solution obtained.

One adjustment is made to the formulation (*CBAS*) in order to enable implicit consideration of subsets of the columns generated while solving the LP relaxation. After column generation has concluded, we add $|A|$ variables to the formulation before switching on the integrality requirements. The formulation after incorporating the new variables is (*CBAS'*) below.

$$
\begin{array}{llll}
 & \max & \sum_{k\in K} r_k y_k - \sum_{i\in A} s_i u_i & \\
(CBAS') & \text{s.t.} & \sum_{k\in K} a_{ik} y_k - u_i \leq w_i \quad \forall i \in A & (14)\\
 & & \sum_{k\in K} y_k \leq |T| & (15)\\
 & & y_k \in \mathscr{Z}^+ \quad \forall k \in K & (16)\\
 & & u_i \in \mathscr{Z}^+ \quad \forall i \in A & (17)
\end{array}
$$

This alteration enables the option of setting a variable k involving an advertisement i to a value greater than w_i if necessary: For each copy of the column in excess of w_i, the excess copies of advertisement i are removed from the list, and the benefit from these widths is removed from the objective function.

4.3. An Initialization Heuristic

Initializing the master problem with a few select columns could help reduce the gaps and solution times considerably. A heuristic approach is used to generate good columns to incorporate into the master problem before column generation is initiated. The heuristic can be described as follows:

1. Initialize $w'_i = w_i \; \forall i \in A$.

2. Solve a binary knapsack problem of the form (*CSUB*), maximizing the objective function $\sum_{i\in A} s_i z_i$. This gives a feasible column to add into the master problem. Let A^g be the set of advertisements included in the generated column. Define $w_g = \min_{i\in A^g}\{w_i\}$ and $g = \operatorname{argmin}_{i\in A^g}\{w_i\}$. Given the constraints, the maximum number of copies of this column that can be used is w_g.

3. For all advertisements $i \in A^g$, update $w'_i = w'_i - w_g$.

4. Set the simple upper bound in the knapsack problem to be 0 for all advertisements $i \in A^g$ for which $w'_i = 0$.

5. If any $w'_i > 0$, go to Step 2; otherwise, conclude the heuristic and add the columns obtained into the master problem.

Results with and without the heuristic are provided in the next section: In general, the heuristic is observed to be of benefit. There is a small amount of preprocessing time associated with the heuristic. This preprocessing time increases as the number of advertisements $|A|$ and the desired frequency of ads w_i increase. As noted in §5, problems where the demand is significantly greater than the space available are relatively easy to solve (this is further elaborated in §5), and as a result, these problems are often solved to optimality, even when the heuristic is not used. Therefore, applying the heuristic *selectively* could improve solution times. Results based on a selective application of the heuristic are also presented.

5. Results

Results from applying Lagrangean decomposition and column generation (with and without the initialization heuristic) are provided in this section, along with information on how the data sets were generated.

5.1. Data Generation

The key parameters of the twenty-five categories are given in the first three columns of Tables 1–6. $|A|$ is the number of advertisements; $|T|$ is the number of time slots available; and S is the size of each slot. Ten instances were generated for each category. Values of s_i were generated from a uniform distribution $U[10, 40]$. Six variations of the twenty-five categories were generated, based on the values of w_i. In the first three, the values of w_i were generated from uniform distributions $U[1, 30]$, $U[1, 20]$, and $U[1, 10]$. It was observed that the column generation approach solved instances with high values of w_i almost instantaneously. This is intuitive: When the w_i are very high relative to the number of time slots, all that needs to be done is to identify a few columns that use up all the space in a time slot and to run as many copies of these columns as possible until we run out of time slots. Therefore, three more variations of the categories were generated from uniform distributions $U[1, 2.2\overline{w}]$, $U[1, 2.3\overline{w}]$, and $U[1, 2.4\overline{w}]$, where $\overline{w} = (|T| \times S)/(\bar{s} \times |A|)$. Here, $|A|$, S, and $|T|$ are as defined earlier, while $\bar{s}$ is the average value of the s_i. Therefore, $\overline{w}$ can be thought of as the average value of w_i that will exactly utilize the slots available. These data sets ensure that relatively easy solutions of the type mentioned above are avoided.

Table 1 **Results for w_i from $U[1, 30]$**

					Column Generation								
			Lagrangean		Without Heuristic			With Heuristic				Selective	
$\|A\|$	$\|T\|$	S	Gap (%)	Time (sec)	Gap (%)	Time (sec)	IPs	Gap (%)	Preproc. Time (sec)	Total Time (sec)	IPs	Gap (%)	Time (sec)
20	40	50	1.451	2.20	0.000	0.00	0	0.000	0.00	0.10	0	0.0000	0.10
40	40	50	1.184	5.56	0.000	0.00	0	0.000	0.10	0.10	0	0.0000	0.00
60	40	50	1.235	7.75	0.000	0.10	0	0.000	0.30	0.30	0	0.0000	0.10
80	40	50	1.518	10.53	0.000	0.10	1	0.000	0.50	0.50	0	0.0000	0.10
100	40	50	1.327	14.70	0.000	0.10	1	0.000	0.80	0.80	0	0.0000	0.10
60	20	50	1.031	3.59	0.000	0.00	0	0.000	0.30	0.30	0	0.0000	0.10
60	40	50	1.046	8.22	0.000	0.10	0	0.000	0.20	0.20	0	0.0000	0.00
60	60	50	1.324	12.38	0.000	0.00	0	0.000	0.30	0.30	0	0.0000	0.10
60	80	50	2.268	16.66	0.000	0.10	1	0.000	0.20	0.30	0	0.0000	0.10
60	100	50	2.726	20.83	0.000	0.20	0	0.000	0.20	0.20	0	0.0000	0.20
60	40	40	3.352	8.22	0.000	0.00	0	0.000	0.20	0.30	0	0.0000	0.00
60	40	60	1.006	8.45	0.000	0.00	0	0.000	0.30	0.30	1	0.0000	0.10
60	40	80	0.629	8.56	0.000	0.00	1	0.000	0.20	0.30	0	0.0000	0.00
60	40	100	0.371	8.91	0.000	0.00	2	0.000	0.10	0.20	0	0.0000	0.10
60	40	120	0.410	9.14	0.000	0.10	1	0.000	0.20	0.20	0	0.0000	0.10
100	100	100	1.234	38.54	0.000	0.40	6	0.000	0.60	0.60	0	0.0000	0.30
100	150	100	1.355	58.80	0.000	0.70	6	0.000	0.70	0.70	4	0.0000	0.80
150	150	100	1.325	93.40	0.001	1.20	10	0.000	1.70	1.80	0	0.0000	0.90
150	200	100	1.540	125.46	0.000	1.50	8	0.000	1.60	1.90	6	0.0000	1.50
200	200	100	1.464	169.56	0.000	2.00	6	0.000	4.10	4.30	1	0.0000	1.70
200	200	50	2.642	166.32	0.000	1.20	3	0.000	4.30	4.30	0	0.0000	1.10
200	20	50	0.170	3.01	0.000	0.00	0	0.000	4.30	4.30	0	0.0000	0.20
200	40	50	1.322	26.62	0.000	0.20	0	0.000	4.20	4.20	0	0.0000	0.20
200	20	100	0.060	5.56	0.000	0.30	0	0.000	4.10	4.10	0	0.0000	0.20
200	60	100	0.702	49.19	0.000	0.40	5	0.000	4.10	4.20	0	0.0000	0.60

Table 2 **Results for w_i from $U[1, 20]$**

					Column Generation								
			Lagrangean		Without Heuristic			With Heuristic				Selective	
$\|A\|$	$\|T\|$	S	Gap (%)	Time (sec)	Gap (%)	Time (sec)	IPs	Gap (%)	Preproc. Time (sec)	Total Time (sec)	IPs	Gap (%)	Time (sec)
20	40	50	2.717	2.89	0.003	0.10	1	0.003	0.00	0.00	1	0.0000	0.10
40	40	50	2.124	5.67	0.000	0.00	0	0.000	0.10	0.10	0	0.0000	0.00
60	40	50	1.792	8.33	0.000	0.10	0	0.000	0.20	0.20	0	0.0000	0.00
80	40	50	2.213	11.69	0.000	0.10	1	0.000	0.50	0.50	0	0.0000	0.10
100	40	50	1.854	14.70	0.000	0.20	1	0.000	0.80	0.80	0	0.0000	0.10
60	20	50	1.215	3.82	0.000	0.00	1	0.000	0.20	0.20	0	0.0000	0.10
60	40	50	1.153	8.33	0.000	0.10	0	0.000	0.20	0.20	0	0.0000	0.10
60	60	50	1.654	12.50	0.000	0.10	1	0.000	0.30	0.30	0	0.0000	0.10
60	80	50	2.372	16.78	0.000	0.10	1	0.000	0.30	0.30	1	0.0000	0.10
60	100	50	3.146	21.06	0.000	0.20	3	0.000	0.20	0.20	0	0.0000	0.20
60	40	40	4.058	8.22	0.000	0.10	0	0.000	0.20	0.20	0	0.0000	0.00
60	40	60	1.300	8.45	0.000	0.10	3	0.000	0.30	0.30	1	0.0000	0.20
60	40	80	0.959	8.68	0.000	0.10	2	0.000	0.30	0.30	0	0.0000	0.10
60	40	100	0.672	9.03	0.000	0.20	4	0.000	0.20	0.20	0	0.0000	0.10
60	40	120	0.720	9.14	0.000	0.10	5	0.000	0.30	0.30	1	0.0000	0.20
100	100	100	1.498	38.77	0.000	0.60	8	0.000	0.60	0.70	3	0.0000	0.70
100	150	100	1.676	59.03	0.000	1.40	10	0.000	0.60	0.70	7	0.0000	0.70
150	150	100	1.502	93.63	0.000	1.80	8	0.000	1.70	2.00	6	0.0000	1.90
150	200	100	1.925	125.70	0.001	3.40	10	0.000	2.00	2.00	7	0.0000	2.00
200	200	100	1.717	170.02	0.000	3.70	10	0.000	4.20	4.20	7	0.0000	3.80
200	200	50	2.543	166.55	0.000	1.70	4	0.000	4.00	4.10	0	0.0000	1.50
200	20	50	0.462	10.18	0.000	0.20	0	0.000	4.40	4.40	0	0.0000	0.10
200	40	50	1.802	32.18	0.000	0.30	0	0.000	4.00	4.20	0	0.0000	0.30
200	20	100	0.241	13.77	0.000	0.20	0	0.000	4.00	4.10	0	0.0000	0.30
200	60	100	0.940	49.31	0.000	0.70	5	0.000	4.10	4.10	0	0.0000	0.70

Table 3 **Results for w_i from $U[1, 10]$**

					Column Generation								
			Lagrangean		Without Heuristic			With Heuristic				Selective	
$\|A\|$	$\|T\|$	S	Gap (%)	Time (sec)	Gap (%)	Time (sec)	IPs	Gap (%)	Preproc. Time (sec)	Total Time (sec)	IPs	Gap (%)	Time (sec)
20	40	50	6.564	3.01	0.033	7.20	3	0.033	0.10	0.10	4	0.0003	0.10
40	40	50	2.648	5.67	0.000	0.00	0	0.000	0.00	0.10	0	0.0000	0.10
60	40	50	2.317	8.45	0.000	0.20	2	0.000	0.10	0.20	0	0.0000	0.20
80	40	50	2.533	11.80	0.000	0.20	0	0.000	0.40	0.40	0	0.0000	0.10
100	40	50	1.932	14.70	0.000	0.20	1	0.000	0.80	0.80	0	0.0000	0.30
60	20	50	2.281	4.28	0.000	0.10	1	0.000	0.30	0.30	0	0.0000	0.10
60	40	50	1.968	8.45	0.000	0.10	2	0.000	0.20	0.20	0	0.0000	0.10
60	60	50	2.368	12.73	0.000	0.20	1	0.000	0.20	0.20	1	0.0000	0.20
60	80	50	3.316	16.90	0.000	0.30	1	0.000	0.20	0.20	1	0.0000	0.30
60	100	50	3.687	21.29	0.007	0.50	7	0.009	0.20	0.50	6	0.0000	0.50
60	40	40	6.157	8.33	0.000	0.10	1	0.000	0.30	0.30	0	0.0000	0.10
60	40	60	2.010	8.68	0.000	0.10	2	0.000	0.20	0.20	1	0.0000	0.20
60	40	80	1.385	8.80	0.003	0.20	4	0.000	0.20	0.30	0	0.0000	0.10
60	40	100	1.350	9.03	0.003	0.40	8	0.000	0.10	0.20	1	0.0000	0.10
60	40	120	1.334	9.38	0.002	0.70	10	0.000	0.10	0.20	3	0.0000	0.20

continue

Table 3 **Continued**

			Lagrangean		Column Generation									
					Without Heuristic			With Heuristic				Selective	
$\lvert A\rvert$	$\lvert T\rvert$	S	Gap (%)	Time (sec)	Gap (%)	Time (sec)	IPs	Gap (%)	Preproc. Time (sec)	Total Time (sec)	IPs	Gap (%)	Time (sec)
100	100	100	2.003	38.89	0.002	3.30	10	0.000	0.60	0.70	5	0.0000	0.70
100	150	100	22.635	60.19	0.137	87.80	10	0.000	0.60	1.00	10	0.0000	1.10
150	150	100	2.171	93.98	0.003	14.20	10	0.000	1.60	1.80	9	0.0000	1.80
150	200	100	11.882	110.18	0.475	97.52	9	0.000	1.40	2.50	10	0.0000	3.50
200	200	100	2.345	171.53	0.005	57.80	10	0.000	3.60	3.80	10	0.0000	3.60
200	200	50	2.777	167.13	0.000	3.10	9	0.000	4.10	4.10	1	0.0000	3.80
200	20	50	1.348	14.58	0.000	0.30	0	0.000	4.00	4.10	0	0.0000	0.30
200	40	50	2.093	32.18	0.000	0.60	1	0.000	4.00	4.00	0	0.0000	0.70
200	20	100	0.639	16.55	0.000	0.50	2	0.000	3.50	3.50	0	0.0000	0.50
200	60	100	1.513	49.54	0.000	1.70	7	0.000	3.30	3.40	0	0.0000	1.50

All the procedures were tested on the 1,500 data sets (25 categories × 6 variations × 10 instances) that were generated.

5.2. Results from Lagrangean Decomposition

The results from applying the Lagrangean decomposition procedure are given in Tables 1–6, under the columns titled *Lagrangean*. The gaps reported are the averages over the ten instances in each category; they are calculated as $((UB - LB)/LB) \times 100\%$, where UB is the upper bound from the Lagrangean decomposition and LB is the objective function value of the best feasible solution found. The times reported are in seconds. The problems were

Table 4 **Results for w_i from $U[1, 2.2\overline{w}]$**

			Lagrangean		Column Generation									
					Without Heuristic			With Heuristic				Selective	
$\lvert A\rvert$	$\lvert T\rvert$	S	Gap (%)	Time (sec)	Gap (%)	Time (sec)	IPs	Gap (%)	Preproc. Time (sec)	Total Time (sec)	IPs	Gap (%)	Time (sec)
20	40	50	6.746	3.01	0.034	0.10	4	0.034	0.10	0.10	4	0.0003	0.10
40	40	50	4.772	5.67	0.006	0.20	5	0.011	0.20	0.30	5	0.0000	0.00
60	40	50	3.977	8.68	0.010	0.40	3	0.006	0.20	0.20	3	0.0001	0.40
80	40	50	3.327	11.92	0.000	0.60	6	0.000	0.20	0.50	2	0.0001	0.50
100	40	50	2.722	15.05	0.000	1.10	6	0.000	0.50	0.50	0	0.0000	0.60
60	20	50	2.965	4.40	0.000	0.30	4	0.000	0.10	0.20	0	0.0000	0.10
60	40	50	3.493	8.56	0.008	0.40	9	0.008	0.40	0.80	6	0.0000	0.30
60	60	50	4.015	12.85	0.012	0.40	8	0.012	0.20	0.40	5	0.0000	0.50
60	80	50	4.764	16.90	0.015	0.50	4	0.015	0.30	0.50	4	0.0004	0.50
60	100	50	4.673	21.41	0.010	0.60	7	0.025	0.40	0.60	8	0.0002	0.60
60	40	40	8.320	8.56	0.029	0.20	6	0.029	0.30	0.40	6	0.0003	0.50
60	40	60	3.124	8.68	0.004	0.60	9	0.004	0.20	0.20	1	0.0000	0.40
60	40	80	2.132	8.91	0.003	0.50	7	0.000	0.20	0.20	1	0.0000	0.20
60	40	100	1.776	9.14	0.015	1.20	10	0.000	0.20	0.30	5	0.0000	0.30
60	40	120	1.448	9.26	0.031	6.80	10	0.027	0.10	1.90	7	0.0003	1.90
100	100	100	2.140	39.12	0.005	3.90	10	0.001	0.70	0.70	8	0.0000	0.80
100	150	100	2.058	59.38	0.004	5.70	10	0.000	0.70	0.80	10	0.0000	0.80
150	150	100	2.156	93.98	0.004	12.70	10	0.000	1.70	1.90	10	0.0000	1.90
150	200	100	2.292	126.27	0.007	35.00	10	0.000	1.50	2.00	9	0.0000	2.00
200	200	100	2.416	171.64	0.005	67.40	10	0.000	3.20	3.50	9	0.0000	3.50
200	200	50	3.425	168.28	0.001	7.10	8	0.001	3.70	4.40	3	0.0000	4.40
200	20	50	2.114	16.32	0.000	1.30	5	0.000	2.30	2.30	0	0.0000	1.30
200	40	50	2.401	32.29	0.005	2.90	6	0.000	2.20	2.20	0	0.0000	2.40
200	20	100	1.404	16.78	0.055	4.60	10	0.000	1.20	1.30	0	0.0002	2.80
200	60	100	1.970	50.00	0.015	11.40	10	0.000	1.90	2.00	0	0.0000	2.00

Table 5 Results for w_i from $U[1, 2.3\overline{w}]$

			Lagrangean		Column Generation									
					Without Heuristic			With Heuristic				Selective	
$\lvert A\rvert$	$\lvert T\rvert$	S	Gap (%)	Time (sec)	Gap (%)	Time (sec)	IPs	Gap (%)	Preproc. Time (sec)	Total Time (sec)	IPs	Gap (%)	Time (sec)
20	40	50	6.576	2.53	0.045	0.10	6	0.040	0.00	0.10	5	0.0004	0.10
40	40	50	4.308	4.96	0.005	0.20	3	0.005	0.20	0.20	2	0.0000	0.20
60	40	50	3.745	7.42	0.010	0.40	6	0.005	0.20	0.20	4	0.0001	0.30
80	40	50	3.386	10.32	0.005	0.60	5	0.000	0.30	0.40	0	0.0000	0.40
100	40	50	2.775	12.94	0.000	0.80	3	0.000	0.50	0.50	0	0.0000	0.60
60	20	50	2.881	3.77	0.010	0.30	5	0.000	0.20	0.20	0	0.0000	0.20
60	40	50	3.316	7.38	0.005	0.40	7	0.005	0.20	2.40	6	0.0000	0.30
60	60	50	3.921	11.07	0.003	0.40	5	0.003	0.20	0.40	4	0.0002	0.50
60	80	50	4.600	14.71	0.026	0.50	8	0.001	0.30	0.40	3	0.0000	0.40
60	100	50	4.297	18.40	0.015	0.50	5	0.017	0.20	0.50	5	0.0002	0.50
60	40	40	8.013	7.33	0.026	0.40	4	0.026	0.20	0.40	5	0.0003	0.40
60	40	60	3.137	7.49	0.004	0.40	6	0.004	0.30	0.30	3	0.0000	0.30
60	40	80	2.086	7.72	0.000	0.50	10	0.000	0.20	0.20	2	0.0000	0.20
60	40	100	1.766	7.83	0.008	0.80	9	0.000	0.10	0.20	1	0.0000	0.20
60	40	120	1.407	8.01	0.013	1.68	7	0.000	0.30	0.30	6	0.0000	0.20
100	100	100	2.047	33.81	0.002	3.00	10	0.000	0.60	0.70	8	0.0000	0.70
100	150	100	2.082	51.29	0.001	3.70	9	0.000	0.80	0.80	8	0.0000	0.80
150	150	100	2.149	81.15	0.002	8.17	8	0.000	1.70	1.80	10	0.0000	1.80
150	200	100	2.218	108.83	0.003	15.22	7	0.000	1.70	1.80	9	0.0000	1.90
200	200	100	2.427	148.02	0.004	31.02	6	0.000	3.30	3.60	9	0.0000	3.30
200	200	50	3.315	145.29	0.005	7.00	10	0.001	3.80	4.20	4	0.0000	4.20
200	20	50	2.114	14.08	0.000	1.30	5	0.000	2.10	2.20	0	0.0000	1.30
200	40	50	2.401	27.91	0.005	2.90	6	0.000	2.30	2.30	0	0.0000	2.30
200	20	100	1.404	14.43	0.055	4.80	10	0.000	1.10	1.10	0	0.0002	2.80
200	60	100	2.062	43.13	0.003	9.90	10	0.000	2.10	2.20	1	0.0000	2.20

solved on a PC with an Intel Pentium II 450 MHz processor.

The gaps varied from a minimum of 0.06% to a maximum of 22.64%. The average gap across the 1,500 instances is 2.67% while the median gap is 2.13%. The time taken for solution varied from a minimum of 2.2 seconds to a maximum of 171.64 seconds. The average time was 34.82 seconds while the median was 12.9 seconds. So with a few exceptions, Lagrangean decomposition does reasonably well.

5.3. Results from Column Generation

As mentioned earlier, the column generation approach was tested with and without the initialization heuristic. A selective implementation of the heuristic was also tried, with the heuristic being used when $\sum_{i\in A} s_i w_i \le 2.5 \times |T| \times S$; that is, when the space needed is less than 2.5 times the space available. The value of the multiplier (2.5) was arrived at through experimentation. The results from the approach without the initialization heuristic are in Tables 1–6 under the columns titled *Without Heuristic*, while the results when the heuristic is used are in the columns titled *With Heuristic*. The times needed for the preprocessor heuristic are given under *Preproc. Time*. Results from the selective implementation of the heuristic are given in the column titled *Selective*. As before, the gaps reported are the averages over the ten instances in each category. The gaps are calculated as $((UB - LB)/LB) \times 100\%$; UB is the LP solution value, and LB is the objective function value of the best feasible solution found. A limit of 100 seconds was set for branch and bound. Eleven instances (out of the 1,500) were terminated as a result of this limit when the initialization heuristic was not used. The time limit was not a factor when the heuristic was incorporated (either selectively or in every run). All versions were implemented using the LINDO Systems (2001) Callable Library on an Intel Pentium II 450 MHz processor.

When no initialization heuristic was used, the gaps ranged from 0.00% to 0.47%, while the average and median gaps were 0.01% and 0.001%, respectively. This is a substantial improvement over the Lagrangean-decomposition results. The solution times ranged from 0.00 second to 97.52 seconds, with an average time of 4.38 seconds and a median time of

Table 6 **Results for w_i from $U[1,\ 2.4\overline{w}]$**

			Lagrangean		Column Generation									
					Without Heuristic			With Heuristic					Selective	
$\|A\|$	$\|T\|$	S	Gap (%)	Time (sec)	Gap (%)	Time (sec)	IPs	Gap (%)	Preproc. Time (sec)	Total Time (sec)	IPs		Gap (%)	Time (sec)
20	40	50	6.135	2.52	0.045	0.10	6	0.036	0.00	0.10	5		0.0004	0.10
40	40	50	4.015	4.96	0.005	0.20	3	0.015	0.20	0.20	4		0.0001	0.10
60	40	50	3.885	7.41	0.010	0.40	6	0.005	0.20	0.30	2		0.0000	0.40
80	40	50	3.343	10.32	0.005	0.60	5	0.000	0.20	0.20	0		0.0000	0.30
100	40	50	2.722	12.95	0.000	0.80	3	0.000	0.60	0.60	0		0.0000	0.60
60	20	50	2.870	3.78	0.010	0.30	5	0.000	0.10	0.10	0		0.0000	0.20
60	40	50	3.066	7.38	0.005	0.40	7	0.018	0.20	0.30	4		0.0002	0.30
60	60	50	3.731	11.03	0.003	0.40	5	0.000	0.30	0.30	2		0.0000	0.40
60	80	50	3.848	14.69	0.026	0.50	8	0.005	0.20	0.40	6		0.0000	0.30
60	100	50	4.225	18.39	0.015	0.50	5	0.008	0.10	0.50	7		0.0001	0.40
60	40	40	7.882	7.34	0.026	0.40	4	0.014	0.30	0.30	4		0.0001	0.30
60	40	60	3.142	7.49	0.004	0.40	6	0.000	0.10	0.30	1		0.0000	0.50
60	40	80	1.979	7.67	0.000	0.50	10	0.000	0.30	0.30	2		0.0000	0.20
60	40	100	1.608	7.82	0.080	0.80	9	0.000	0.20	0.20	3		0.0000	0.20
60	40	120	1.426	8.02	0.013	1.68	7	0.000	0.30	0.30	5		0.0000	0.20
100	100	100	2.113	33.83	0.002	3.00	10	0.000	0.70	0.70	7		0.0000	0.60
100	150	100	2.017	51.24	0.001	3.70	9	0.000	0.70	0.80	7		0.0000	0.80
150	150	100	2.047	81.11	0.002	8.17	1	0.000	1.70	1.80	9		0.0000	1.80
150	200	100	2.075	108.78	0.003	15.22	2	0.000	1.70	1.80	7		0.0000	1.80
200	200	100	2.372	147.88	0.004	31.02	3	0.000	3.40	3.70	9		0.0000	3.70
200	200	50	3.288	145.25	0.005	7.00	10	0.000	3.50	4.00	2		0.0000	3.80
200	20	50	2.114	14.07	0.000	1.30	5	0.000	2.20	2.20	0		0.0000	1.20
200	40	50	2.401	27.90	0.005	2.90	6	0.000	2.20	2.20	0		0.0000	2.40
200	20	100	1.404	14.44	0.055	4.80	10	0.000	1.20	1.20	0		0.0002	2.80
200	60	100	2.048	43.08	0.003	9.90	10	0.000	2.00	2.20	0		0.0000	2.20

0.5 second. This too is a notable improvement over the Lagrangean-decomposition results.

Once the initialization heuristic was incorporated, the gaps ranged from 0.00% to 0.04%, with the average and median gaps being 0.002% and 0.00%, respectively. The time taken for solution ranged from 0.00 second to 4.40 seconds, with an average time of 1.22 seconds and a median time of 0.5 second. These numbers represent significant improvements over the corresponding values without the heuristic. When the heuristic is used, most of the time needed for solution is spent on the heuristic: Very little additional time goes into solving the IP. The selective implementation of the heuristic did even better. The gaps ranged from 0.00% to 0.0004% while the average and median gaps were 0.00% and 0.00%, respectively. Solution times ranged from 0.00 second to 4.40 seconds, with an average time of 0.84 second and a median time of 0.00 second.

The columns with subheading *IP*s provide the number of instances in each category where branch and bound had to be resorted to, i.e., instances where the LP solution was not integral. Branch and bound was needed in 917 of 1,500 instances when the heuristic was not used; this number fell to 395 when the heuristic was used. When the heuristic was selectively used, the number of instances that needed branch and bound was 532.

6. Conclusions

This paper presents a problem encountered in the scheduling of Web-banner advertisements when the planning horizon is shorter than the scheduling horizon—a common situation observed in practice. Two formulations are introduced for the problem and solution methods involving Lagrangean decomposition and column generation are presented. Extensive computational tests are conducted over a variety of parameter values. The Lagrangean-decomposition approach gives reasonably good solutions, but the column generation procedure is observed to do substantially better, providing near-optimal solutions in an average time of less than a second. Given that banner advertisements account for almost half of all Web advertisements, and considering that Web advertising revenues add up to a few billion dollars per year, even a small improvement in the quality of schedules could result in a considerable increase in revenue.

References

Adler, M., P. Gibbons, Y. Matias. 2002. Scheduling space sharing for internet advertising. *J. Scheduling* **5** 103–119.

Amiri, A., S. Menon. 2001. Efficient scheduling of internet banner advertisements. Working paper, College of Business, Oklahoma State University, Stillwater, OK.

Balas, E., E. Zemel. 1980. An algorithm for large zero–one knapsack problems. *Oper. Res.* **28** 1130–1154.

Bazarra, M., H. Sherali, C. Shetty. 1993. *Nonlinear Programming Theory and Algorithms*. John Wiley & Sons, New York.

Buchwalter, C., M. Ryan, D. Martin. 2001. The state of online advertising: Data covering fourth quarter 2000. Technical Report, AdRelevance (a Jupiter Media Metrix company), Seattle, WA.

Caprara, A., H. Kellerer, U. Pferschy. 2000. The multiple subset sum problem. *SIAM J. Optim.* **11** 308–319.

Dantzig G., P. Wolfe. 1960. Decomposition principle for linear programs. *Oper. Res.* **8** 101–111.

Degraeve, Z., L. Schrage. 1999. Optimal integer solutions to cutting stock problems. *INFORMS J. Comput.* **11** 406–419.

Fisher, M. 1981. The Lagrangian relaxation method for solving integer programming problems. *Management Sci.* **27** 1–18.

Garey, M., R. Graham. 1975. Bounds for multiprocessor scheduling with resource constraints. *SIAM J. Comput.* **4** 187–200.

Geoffrion, A. 1974. Lagrangean relaxation for integer programming. *Math. Programming Stud.* **2** 82–114.

Gilmore, P., R. Gomory. 1961. A linear programming approach to the cutting stock problem. *Oper. Res.* **9** 849–859.

Gilmore P., R. Gomory. 1963. A linear programming approach to the cutting stock problem—Part II. *Oper. Res.* **11** 863–888.

Guignard M., S. Kim. 1987. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Math. Programming* **39** 215–228.

Held, M., R. Karp. 1970. The traveling salesman problem and minimum spanning trees. *Oper. Res.* **18** 1138–1162.

Held, M., R. Karp. 1971. The traveling salesman problem and minimum spanning trees—Part II. *Math. Programming* **1** 6–25.

Interactive Advertising Bureau. 2001. Internet ad revenue report, fourth quarter of 2000. http://www.iab.net/forms/qreport.html.

Kumar, S., V. Jacob, C. Sriskandarajah. 2000. Scheduling advertisements on a web page to maximize space utilization. Working paper, School of Management, The University of Texas at Dallas, Richardson, TX.

LINDO Systems, Inc. 2001. *LINDO Callable Library User's Manual*. Chicago, Illinois.

Martello, S., D. Pisinger, P. Toth. 2000. New trends in exact algorithms for the 0–1 knapsack problem. *Eur. J. Oper. Res.* **123** 325–332.

Martin. R. 1999. *Large Scale Linear and Integer Programming: A Unified Approach*. Kluwer Academic Press, Boston, MA.

Nemhauser, G., L. Wolsey. 1988. *Integer and Combinatorial Optimization*. Wiley Interscience Series in Discrete Mathematics and Optimization, New York.

Pinedo, M. 1995. *Scheduling Theory, Algorithms and Systems*. Prentice Hall, Englewood Cliffs, NJ.

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.

Vance, P. 1998. Branch-and-price algorithms for the one-dimensional cutting stock problem. *Comput. Optim. Appl.* **9** 211–228.

Vanderbeck, F. 1994. *Decomposition and Column Generation for Integer Programs*. Ph.D. thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium.