International Journal of Operations Research

International Journal of Operations Research Vol. 13, No. 3, 102–120 (2016)

Banner Advertisements Scheduling to Maximize Space Utilization

Yen-Ting Lu, Bertrand M.T. Lin*, and Kuei-Tang Fang

Institute of Information Management, Department of Information Management and Finance

National Chiao Tung University, Hsinchu 300

Received February 2015; Revised February 2015; Accepted April 2015

Abstract — One of the commercial Internet applications is banner advertisement, which is also the major source of income for portal websites. Limited research works address the issues of efficiently scheduling advertisement requests into available advertisement banners in view of profitably. This paper proposes a new model for scheduling banner advertisements that more closely resembles the Internet business world. In the proposed model, we apply a feasible time window to each candidate order, and schedule advertisements into various predefined banner spaces on the webpage. This approach allows different pricing strategies according to banner types and is contrary to previous research models that schedule and price by banner space sharing. Each order for ad space will demand a certain frequency for each type of banner. The goal is to select and schedule requests to achieve the maximum ad capacity utilization. We first give a mathematical formulation to formally describe the problem. Due to the computational complexity of the studied problem, we seek to produce approximate solutions in a reasonable time. A three-phase heuristic is developed to cope with this problem. In addition, an upper bound on profits is developed. Computational experiments are designed to examine the performance of the heuristic. Statistics from the experiments reveal that the heuristic can successfully fill up to 97% of ad spaces.

Keywords — Banner advertisements, Operations scheduling, Time window, Capacity utilization

1. INTRODUCTION

Ever since advertisements emerged, they are seldom absent from any influential media, including fliers, newspapers, radio, and television. In today's world, the Internet is no exception. In this Internet age, one can hardly find a website or portal with business power without advertisement banners on the site. As of 2010, the growth rate of Internet ads market is still booming. The IAB Internet Advertising Revenue Report (2011) shows that Internet ad revenues for 2010 increase 15% to $26 billion, a new record. Moreover, Internet ad revenues hit $7.3 billion in Q1'11 which is the highest first-quarter revenue level on record according to IAB. Although the cost of Internet advertisements has not reached as high as television advertisements, the latter can be charged millions of dollars to be shown on television for only a couple of seconds. This investment is often effective and profitable because the television viewing population is large. Many properties on the Internet are worth utilizing, however, including lower cost and more conveniently connecting to other countries and other continents. In 2006, England and Sweden have the online advertisements market beyond 10% of their total domestic market of advertisements Payne (2006). These two countries are the first in the world with a double-digit percent market share. The report of Szalai Payne (2006) also predicted that this may also happen in Australia, Israel, Japan, Norway, South Korea, and Taiwan before 2008. Zenith's global ad spending forecast for major media for 2011 is now $470.8 billion Payne (2006). Moreover, Zenith sees the Internet overtaking newspapers to become the world's second-largest advertising medium behind television in 2013

The value of web ads is clearly illustrated by decisions made by famous IT companies; for example, Google acquiring Doubleclick for $3.1 billion; Yahoo buying RightMedia for $680 million in April 2007; and Microsoft buying aQuantive for $6 billion in May 2007 by Abramovich (2007), Freund A and Naor (2004), Johnson (2007).

As information technologies progress rapidly, the popularity of Internet advertising rises and includes many unique advantages that are not present with traditional media. The four main attractive characteristics of Internet advertising are ADPAGENET:

1. ***Interaction***: Advertisements on the Internet deliver information to browsers as conventional ads in newspapers or on television. Internet ads can interact with viewers step by step, much like an extension of direct marketing. This feature effectively catches the viewer's attention and communicates in-depth with the targeted

* Corresponding author's e-mail: bmtlin@mail.nctu.edu.tw

audience.

2. ***Traceability***: One of the most convenient functions of the Internet is hyperlinks. Links allow Internet users to reach ad host websites easily and rapidly from the referring page, and even finish the entire transaction process on the net. This makes the feedback and effectiveness of advertising traceable and measurable. Tracing is a very useful tool for ads hosts to make decisions and improve advertisement quality.
3. ***Segmentation***: Website hosts can collect visitors' information using online tables or recording their behavior on the Internet at a low cost. This feature is very useful for targeted promotion, which not only improves advertising effectiveness, but also reduces the possibility of disturbing people who are not interesting in receiving the ads.
4. ***Flexibility***: Websites on the Internet can operate around the clock. Hosts can modify and update the ads easily at anytime. This makes promotion activities more flexible and powerful. One of the remarkable examples of flexibility is online auction capabilities.

As the importance of web advertisements increase, many research studies have focused on this subject in recent years. One part of such research is studying how to improve the effectiveness of web advertising with each impression. These studies are based on many different social science methodologies. Some research studies developed models and algorithms to select the most attractive advertisements for specified web guests by observing and recording user behavior and hobbies based on Internet activity. Other studies worked to classify types of advertisements and types of web context (or content), and then place relative advertisements on matching web pages. Google, Yahoo!, AOL, and MSN, the four biggest portals in American, made $19.5 billion in 2007, which accounted for 66% of Internet ad spending (EADP, 2007) by Szalai (2011). The phenomenon that web ad revenues center on these extremely big portals is caused not only by their large numbers of browsers, but also the various services they provide. These services can hold visitors for a longer time; be useful to website managers to record browsers' behavior; target browsers' interests; and raise the effectiveness of ads impressions. United States Patent No. 5948061, "Method of Delivery, Targeting, and Measuring Advertising over Networks," which belongs to Double Click, Inc. is one famous instance that reveals the commercial value and wide-spread use of strategic ads IAB (2011). Another group of research studies has explored how to select and allocate advertisement orders to generate the best profits for advertisement agents or website owners. Internet advertising faces the same challenges that ads on other media do, including how to fill the ad with content that the audience wants to receive; not causing excessive disturbance to the audience; and generating the maximum revenue of media owners with limited display space/time slots. This is the topic we study in this paper, and we begin by reviewing existing literature.

In this paper, we present a new model that schedules off-line order contents with a static pricing scheme, but limits the feasible time window for each ad order (i.e. the release date and due date), with the objective of maximizing revenues for ad agents. This paper is organized as follows. We present formal statements of the problem definition in Section 2. A binary integer programming formulation is also given to mathematically describe the studied problem. In Section 3, we introduce our 3-phase heuristic to solve this problem and implement the heuristic with a small data set. In Section 4, we design one upper bound for our objective, and we implement the upper bound with the same data set. Section 5 reports our computational experiments and analysis of the results. Section 6 includes conclusions and remarks.

2. Problem Formulation and Literature Review

In this section, we present a formal formulation of the banner scheduling problem. A review on previous research related subjects follows.

2.1 Problem definition and mathematical model

Consider a website that offers m different types of advertisement space $S = \{s_1, s_2, \ldots, s_m\}$, in which s_j denotes space type j. From clients, n orders $O = \{o_1, o_2, \ldots, o_n\}$ are placed. An order may contain ads of one or more space types. Each order o_i, has a time window $[\alpha_i, \beta_i]$ that specifies the duration of ad activities. Denote by w_{ij} the number of days on which ads of space type s_i are required by order o_i. The days scheduled for an order may not necessarily be consecutive; for example, all ads within an order may scatter across non-consecutive days within a given time window. An order is fulfilled and the web site gets paid only if: (1) all ads requested in the order are successfully scheduled within the window; (2) on each day of the schedule, at most only one ad of any type can be scheduled due to the effectiveness consideration. Subsequently, we need the assumption $W_i = \sum_{j=1}^{m} w_{ij} \le \beta_i - \alpha_i$ to maintain the feasibility of each individual order o_i. For collecting O of n orders, a planning horizon of T days is given. The goal of planning is to select a subset of orders and schedule the demands of the selected orders so as to maximize the utilization of ad banners.

To describe the studied problem (Banner Advertisement Scheduling, BAS) in a mathematical way, we provide a binary

integer programming formulation. Binary decision variable x_{ijt} is 1 if ad type s_j of order o_i is scheduled on day t, $1 \le i \le n$, $1 \le j \le m$, $1 \le t \le T$. Binary auxiliary variable y_i is defined to dictate if order o_i is selected and scheduled successfully for $1 \le i \le n$.

Program BAS

Maximize $\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{t=1}^{T} x_{ijk}$

subject to

$$\sum_{j=1}^{m} x_{ijt} \le 1, \qquad 1 \le i \le n,\ 1 \le t \le T; \tag{1}$$

$$\sum_{i=1}^{n} x_{ijt} \le 1, \qquad 1 \le j \le m, 1 \le t \le T; \tag{2}$$

$$\sum_{t=1}^{\alpha_i - 1} x_{ijt} = 0, \qquad 1 \le i \le n, 1 \le j \le m; \tag{3}$$

$$\sum_{t=\beta_i+1}^{T} x_{ijt} = 0, \qquad 1 \le i \le n,\ 1 \le j \le m; \tag{4}$$

$$\sum_{t=\alpha_i}^{\beta_i} x_{ijt} = w_{ij} y_i, \qquad 1 \le i \le n, 1 \le j \le m; \tag{5}$$

$$x_{ijt} \in \{0.\ 1\}, \qquad 1 \le i \le n, 1 \le j \le m, 1 \le t \le T; \tag{6}$$

$$y_i \in \{0.\ 1\}, \qquad 1 \le i \le n. \tag{7}$$

Constraints (1) dictate that any order o_i can have at most one ad type on day t. The restriction that each ad space can be occupied by at most one order is given by constraints (2). Constraints (3) and (4) confine the forbidden days for each order. Constraints (5) ensure that scheduling of an order is successful only if the requirement of any ad type in this order is fully allocated in the allowed time window. Constraints (6) and (7) define the binary characteristics of the decision and auxiliary variables. The mathematical program uses $O(n^2T)$ binary variables and consists of $O(nT + n^2)$ functional constraints.

2.2 Literature review

Scheduling of Internet advertising was probably first formulated by Adler et al. (2002), who presented a space-sharing model. In this model, a fixed area on the web page is available for advertisements and the demands of each ad are specified by geometric size and display frequency. One constraint of the Internet ads scheduling problem is that a web site cannot place ads of the same company in the same time slot due to the effectiveness considerations. The first type of problem is to schedule the activities, satisfying the frequency of each ad and not violating the space constraints, so as to use the minimal number of time slots to finish all tasks. Another type of problem studied is to select the optimal subset of ads to maximize ad agent revenues, subject to a fixed number of time slots (limited planning time horizon). They designed efficient algorithms to find optimal solutions to the former problem with specified restrictions, and 2-approximation solutions for the latter problem. Amiri et al. (2003) used Lagrangean decomposition to solve the problem in a relatively short computing time to get good results. Dawande et al. (2003) provided the first known algorithms with constant factor approximations to solve this problem. The simple one guarantees 1/4 and the complex one does 3/10 ratio of their performance to the upper bound value presented in this paper. Freund et al. (2004) designed a algorithm guarantee $1/(3+\varepsilon)$-approximate for general cases and $1/(2+\varepsilon)$ for two special cases.

For the same objective function, to obtain maximum revenues for ad agents, Menon et al. (2004) introduced another model to address the Internet ads scheduling problem. They relaxed the constraint of reaching all demands of accepted ads, which allowed the ad agents to realize better revenues, but just satisfy the ad's partial demands in the schedule. Menon et al. (2004) used Lagrangean decomposition and column generation to solve this problem and compare performances. The column generation achieved better results and took a shorter time. Amiri et al. (2006) also presented a more flexible and practical model, using the same relaxation of ad demands previously mentioned, but suggest a more precise pricing scheme for real-world trading. Ad customers can list several levels of demands to be implemented, and each ad display pays to ad agents according to its levels. In other words, the level with more counts of display, the higher the individual price of each display. They deployed a Lagrangean decomposition-based solution procedure that can perform very well for reasonably

large problems.

Except the maximum revenues objective, some research studies have attempted to solve the problem called "MINSPACE," defined by Dawande, et al. (2003). They set a fixed number of time slots to be used; schedule all the ads with their demands exactly; and try to minimize the maximum slot fullness. The solution to this problem can help ad agents decide what size banner is efficient and suitable to meet their customers' demands. Dawande et al. (2003) provided a 2-approximation algorithm for this problem. They presented an online version of the MINSPACE problem, and provided an algorithm with the performance bound of (2-1/N), where N stands for the maximum capacity of one slot. They also provided two off-line algorithms with performance ratios $(1+1/\sqrt{2})$ and 3/2, respectively. In a recent study of Liu and Zhang (2010), a space optimization problem was studied by a subgradient optimization approach incorporating Langragean decomposition. A heuristic algorithm was also designed to produce schedules in a timely manner. Boskamp et al. (2011) formulated the revenue optimization of web banners as a two-dimensional bin packing problem. They proposed several heuristic algorithms, including left justified algorithm, the orthogonal algorithm, the GRASP constructive algorithm, and the greedy stripping algorithm. To the best of our knowledge, the model proposed in this paper to take into account multiple orders with frequency variations and time windows were never studied before.

3. Placement Algorithm

In this section, we will present a heuristic algorithm to deal with the banner ads scheduling problem of maximizing ad capacity utilization. The exposition of the designed algorithm will be applied to a test data set to provide an illustration.

The heuristic consists of three phases. Phase 1 of the algorithm seeks to derive an initial solution. It first fills slots in a forward manner, then changes the direction to allocate unscheduled orders in a backward manner. In phase 2, the algorithm manages to arrange the remaining orders by shifting scheduled ads without sacrificing the feasibility of scheduled orders. In phase 3, we cancel scheduled orders of less commercial value in order to arrange orders that provide better gains. To better describe the algorithm, we use the following data set for an illustration.

Consider $n = 10$ orders $\{A, B, \ldots, I, J\}$; $m = 4$ types of ads space with a scheduling horizon of $T = 16$ days. The specifications of orders are given in 6-tuple vectors. For example, order A consists of two days of type 1, three days of type 2, four days of type 3, and does not require type 4. Its allowed time window is from day 2 to day 12.

10 orders:

A (2, 3, 4, 0; 2, 12)
B (3, 2, 1, 2; 1, 10)
C (4, 0, 0, 0; 0, 8)
D (2, 0, 0, 0; 0, 15)
E (1, 6, 0, 2; 0, 12)
F (2, 5, 0, 0; 7, 15)
G (2, 1, 1, 2; 4, 14)
H (0, 0, 0, 3; 6, 12)
I (0, 1, 2, 3; 5, 13)
J (0, 5, 0, 0; 10, 15)

Phase 1: Initial Placement

This phase consists of two approaches for arranging the given orders. The first approach, called **putS**, selects orders based on release dates and space type. With a partial schedule, we locate the last day on which the scheduling of the last order was finished. The unscheduled order with the earliest release date from the located finishing day is selected to schedule. If the last scheduled order was finished with ads of type s_j, we start arranging the selected order by sequencing its type s_j ads first, followed by type s_{j+1} ads, …, s_m, s_0, s_1,…, and then type s_{j-1} ads in a round-robin way. This method aims to keep the slots of the target types as full as possible.

The other approach, called **rev_putS**, runs in the reverse direction of **putS**, and schedules orders backward from the rear end of the planning horizon. We select the unscheduled order with the latest due date to the earliest scheduled slots on the target type frame, and place the ads in order with the sequence from the target slot type s_j to $s_{j-1},\ldots,s_0, s_m, s_{m-1},\ldots, s_{j+1}$.

Deploying the two approaches depends on the current time point under consideration. **PutS** is applied first. If the order just finished here exceeds 1/2 of the time horizon on its target type, and we then use **rev_putS** to schedule the same target frame until no order can be scheduled with **rev_putS**, we can then set the next slot type as the target type to start the next round. After finishing m rounds, we will check if any unscheduled order can be further completed by **putS** or **rev_putS**. The main steps of Phase 1 are outlined in the following:

Start with frame of type 1 as the target frame for round 1.

1. Schedule the order with the earliest release date as first using **putS**, and schedule the next order with the earliest release

date to the completion date of the previous scheduled order on the target frame by **putS**. Repeat the process until the completion time of the current order exceeds the middle point of the scheduling horizon. Schedule the order with the latest due date by **rev_putS**, and schedule the next order which has the latest due date to the earliest scheduled date of the previous order on the target frame type by **rev_putS** until all orders have been examined once.

2. Move to the next target frame until all frames are considered as the target frame once.
3. Apply **putS** and **rev_putS** to dispatch all the unscheduled orders.

The detailed implementation is:

1. Start with target frame type $s_j = 1$
2. Set time pointer $p = 0$ and tag all un-scheduled orders un-checked.
3. If an un-checked order exists, go to step 4; otherwise set $s_j = s_j + 1$, and go to step 2.
4. Check if pointer $p < T/2$. If yes, go to step 5; otherwise go to step 6.
5. Find the un-checked order o_i with the nearest release date β_i to p. Try to place the order in the schedule by method **putS** and tag o_i checked. If this fails, go to step 3; if this succeeds, set p = the latest date when ad of o_i has been put in space j and go to step 3.
6. Set time pointer $q = T$
7. If an un-checked order exists, go to step 8; otherwise set $s_j = s_j+1$, and go to step 2.
8. Find the order o_i which has not been checked with the nearest due date β_i to q. Try to place the order in the schedule by method **rev_putS** and tag o_i checked. If this fails, go to step 7; if this succeeds, set q = the earliest date when ad of o_i has been put in space m and go to step 7.
9. Finish when $s_j = m$ is done.
10. Examine if any orders can be scheduled by method **putS** or **rev_putS**.

Applying Phase 1 to a 10-order example will produce the plan shown in Table 1.

Table 1: Output of Phase 1.

Time Frame-type	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	*C*	*C*	*C*	*C*	*G*	*G*	*B*	*F*	*F*	*B*	*D*	*D*				
2				*B*	*B*	*B*	*G*		*I*	*F*	*F*	*F*	*F*	*F*		
3								*G*	*B*	*I*	*I*					
4								*B*	*G*	*G*	*B*	*I*	*I*	*I*		

The detail process is given below. The success of applying **putS** to some orders is denoted by "s" and failure by "f'. The same applies to **rev_putS**.

Round 1:

putS(*C*) = s ; putS(*G*) = s; *H* is skipped because there is no demand of type 1;
putS(*F*) = s ; Reverse the direction because $p = 8 > 7$.
D is skipped because there is no demand of type 4;
rev_putS(*J*) = f ; rev_putS(*I*) = s;
A is skipped because there is no demand of type 4;
rev_putS(*B*) = s ; rev_putS(*H*) = f; rev_putS(*E*) = f;

Round 2:

D is skipped because there is no demand of type 2;
putS(*E*) = f ; putS(*A*) = f ;
H is skipped because there is no demand of type 2;
putS(*J*)= f ;

Round 3:

D is skipped because there is no demand of type 3;
E is skipped because there is no demand of type 3;
putS(*A*) = f ; *H* is skipped because there is no demand of type 3;
J is skipped because there is no demand of type 3;

Round 4:

D is skipped because there is no demand of type 4 ;
putS(*E*) = f ; *A* is skipped because there is no demand of type 4;
putS(*H*) = f ; *J* is skipped because there is no demand of type 4;

Final Check:

putS(*A*) = f ; rev_putS(*A*) = f ;
putS(*D*) = s ;
putS(*E*) = f ; rev_putS(*E*) = f ;
putS(*H*) = f ; rev_putS(*H*) = f ;
putS(*J*) = f ; rev_putS(*J*) = f ;

Phase 2: Shift the filled slots to accommodate remaining unscheduled orders

In this phase, we reorder the filled slots to arrange the unscheduled orders one by one. Method **shiftA** is applied first to a selected order. If it fails to schedule the order, then we will use a method called **shiftB**. If **shiftB** also fails, then we skip the considered order and move to the next unscheduled order.

shiftA:

1. For each unscheduled order o_i, we consider its ads in the ordering of type $s_1, s_2, \ldots, s_m$.
2. Start from date a_i. If a slot of a certain type is not occupied, but an ad of some other type in order o_i is scheduled on the same date, then move the scheduled ad to another feasible date and put in the current ad. If the slot has been occupied by an ad of another order, move the scheduled ad to another feasible date and put in the current ad. Whether we can arrange the current ad successfully or not, keep examining the next time slot until the ads required for the slot type are successfully placed or the time β_i is reached. If scheduling an order of this ad type is successful, then proceed to the next space type and repeat Step 2. Otherwise, give up this order and select the next one for consideration until all the unscheduled orders are inspected exactly once.

shiftB:

1. Start from time α_i. Put the ads in in each frame by the non-increasing order of space type. The remaining operations of **shiftB** are the same as those of **shiftA**.

From Table 1, we know that schedules *A*, *B*, *H*, *J* were not successfully scheduled in Phase 1. Therefore, we consider the four orders one-by-one and achieve the following results, in which the plan is augmented by inserting order *A*. The new plan is shown in Table 2.

shiftA(*A*) = s ;
shiftA(*E*) = f ; shiftB(*E*) = f ;
shiftA(*H*) = s ;
shiftA(*J*) = f ; shiftB(*J*) = f ;

Table 2: Output of Phase 2.

Time Frame-type	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	*C*	*C*	*C*	*C*	*A*	*A*	*B*	*F*	*F*	*B*	*D*	*D*	*G*	*G*		
2		*B*		*A*	*B*	*B*	*A*	*A*	*I*	*F*	*F*	*F*	*F*	*F*	*G*	
3			*A*	*B*		*I*	*I*	*G*	*A*	*A*	*A*					
4			*B*		*G*		*H*	*H*	*H*	*G*	*B*	*I*	*I*	*I*		

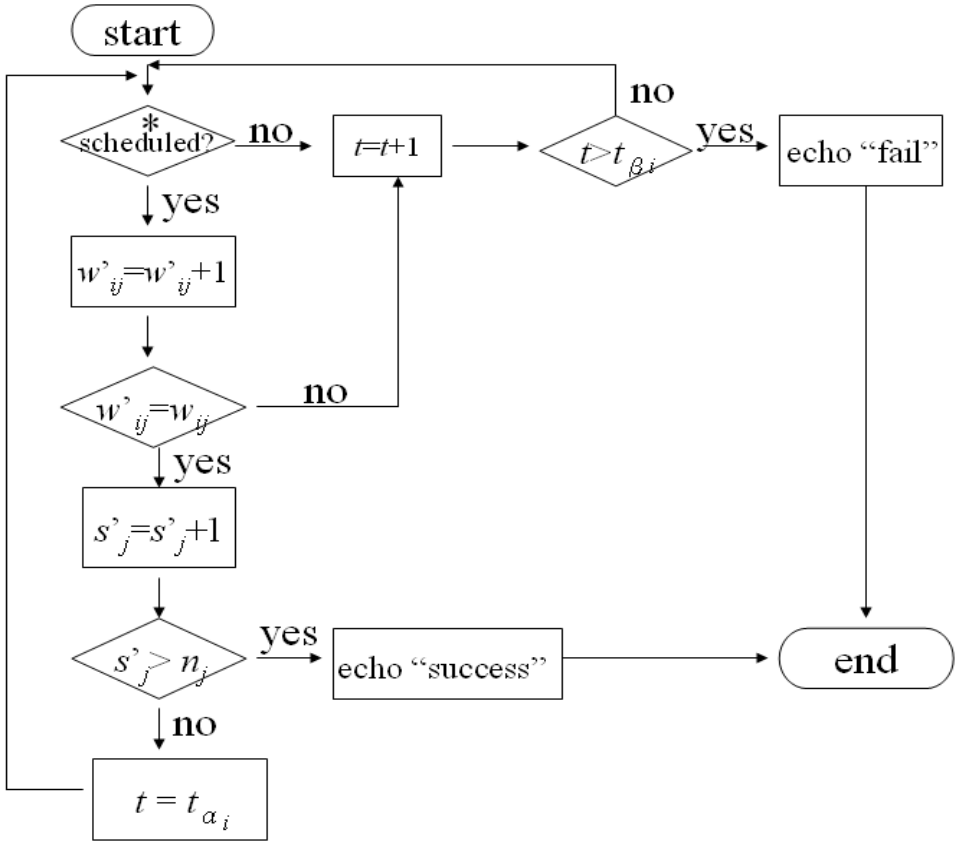


Figure 1: Flowchart of Phase 2.

Figure 1 shows part of the scheduling procedure in Phase 2 for one order. The process with an asterisk ("*") uses method **shiftA** to schedule the ads of any single order. If the trial of scheduling the order fails, we will clear all ads belonging to the order on the schedule, and use method **shiftB** to schedule the order again.

Phase 3: Exchange scheduled and unscheduled orders

In the previous two phases, we arrange as many orders as possible. In this phase, we improve the plan by canceling some orders in order to accommodate unscheduled orders such that the total revenue can be increased. The exchange process is explained below.

1. Maintain two sets of orders, where P = the set of scheduled orders and Q = the set of unscheduled orders. Sort the orders of Q in non-increasing order of commercial values.
2. Examine orders o_i by their indices from $i = 0$ to $i = n$. If order o_i belongs to Q, skip it and move on to the next order. If order o_i belongs to P and the commercial value of o_i is less than that of the first order, say o_j, of Q, then remove order o_i from the plan and find the first order belonging to Q that can successfully replace the removed order o_i. If such a replacement is successful, then put o_j into P and o_i into Q. Proceed to examine the next scheduled order.

Applying the operations of Phase 3 to the tentative plan shown in Table 2, we have the following detailed steps and a new plan included in Tables 3(a) and 3(b):

Total demands of unscheduled orders: $E(9) \geqq J(5)$
Check order A: $A(9) \geqq E(9)$
Check order C: $C(4) < E(9)$, cancel C , putS(E) = f , rev_putS(E) = f, shiftA(E) = f, shiftB(E) = f; putS(J) = f , rev_putS(J) = f, shiftA(J) = f, shiftB(J) = f; ; recover C ;
Check order D: $D(2) < E(9)$, cancel D, putS(E) = f, rev_putS(E) = f, shiftA(E) = f, shiftB(E) = f, putS(J) = f, rev_putS(J) = f,shiftA(J)=f, shiftB(J)=f; recover D;
Check order F: $F(7) < E(9)$,cancel F, putS(E) = s ,
Total demands of unscheduled orders: $F(7) \geqq J(5)$

Check order *I*: *I*(6) < *F*(7) , cancel *I* , putS(*F*) = f, rev_putS(*F*) = f , shiftA(*F*)= f, shiftB(*F*) = f, skip *J* because *I*(6) > *J*(5)

Table 3(a): Intermediate plan resulted from insertion of order *E*

Time Frame-type	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	*C*	*C*	*C*	*C*	*A*	*A*	*B*	*E*		*B*	*D*	*D*	*G*	*G*		
2	*E*	*B*	*E*	*A*	*B*	*B*	*A*	*A*	*I*	*E*	*E*	*E*	*E*		*G*	
3			*A*	*B*		*I*	*I*	*G*	*A*	*A*	*A*					
4		*E*	*B*	*E*	*G*		*H*	*H*	*H*	*G*	*B*	*I*	*I*	*I*		

Table 3(b): Final plan resulted from Phase 3.

Time Frame-type	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	*C*	*C*	*C*	*C*	*A*	*A*	*B*	*E*		*B*	*D*	*D*	*G*	*G*		
2	*E*	*B*	*E*	*A*	*B*	*B*	*A*	*A*	*I*	*E*	*E*	*E*	*E*		*G*	
3			*A*	*B*		*I*	*I*	*G*	*A*	*A*	*A*					
4		*E*	*B*	*E*	*G*		*H*	*H*	*H*	*G*	*B*	*I*	*I*	*I*		

The flow of operations of Phase 3 is illustrated in Figure 2. The process with the asterisk ("*") uses the four scheduling methods used previously, including **putS**, **rev_putS**, **shiftA**, and **shiftB**. If all methods cannot successfully schedule the order, give up on this order and proceed to another unscheduled order.

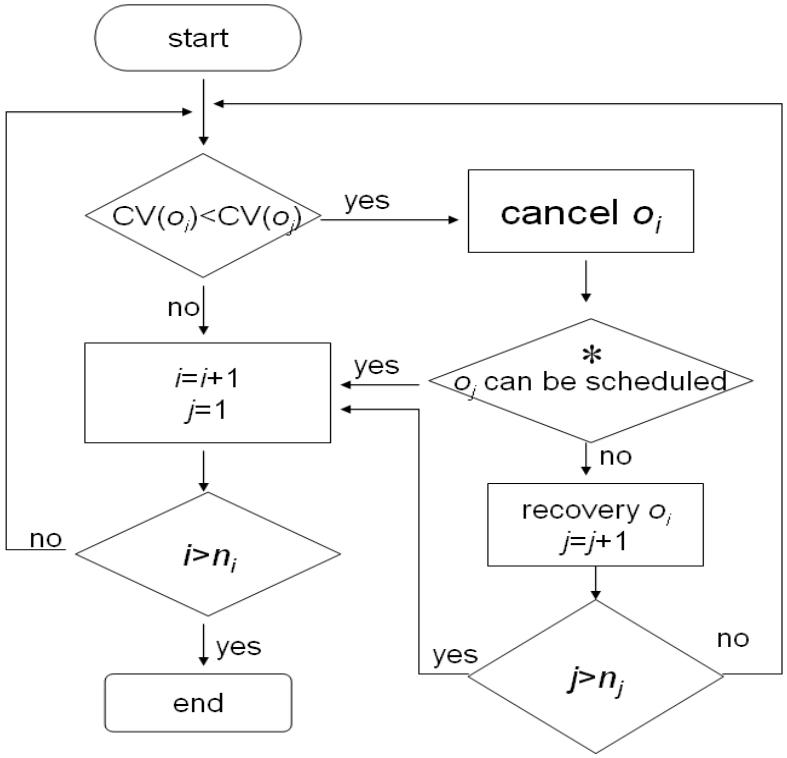


Figure 2: Flowchart of Phase 3.

4. Upper Bound

While it is very unlikely to derive the optimal value of the studied scheduling problem, it is reasonable to derive an upper bound on the optimal number of slots that can be filled. The upper bound can be used to estimate the optimal value and to measure the quality of the plan produced by the proposed placement algorithm. In this section, we present a theoretical analysis of an upper bound by scrutinizing the profiles of the given orders. The upper bound is developed by considering the other side of the question: deriving a lower bound on the number of slots that are impossible to use. In the development, the lower bound, denoted by *LB*, is expressed as

$LB = X + \max\{Y_1, Y_2\} + Z$

In the following section, the derivation of each term is explained in detail. We will use a numerical example to illustrate each term.

Initialization

For day $t = 1$ to T, define variable c_t as the number of orders eligible for time t. For example, an order o_i is eligible on day t if t is covered by the interval $[\alpha_i,\beta_i]$. The c_t values of the 10-order instance are given in Table 4:

Table 4: The c_t values.

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
c_t	3	4	5	5	6	7	8	9	9	8	9	8	8	5	4	3

The value of X

For any $c_t < m$, we can readily have $m - c_t$ slots that will not be used on day t. Therefore, for the whole planning horizon, at least $X = \sum_{t=1}^{m} \chi_t$ slots can never be used, where

$$\chi_t = \begin{cases} m - c_t, & \text{if } c_t < m; \\ 0, & \text{otherwise.} \end{cases}$$

In the example data, we have $X = 1+0+0+0+0+0+0+0+0+0+0+0+0+0+0+1 = 2$.

The value of Y_1

When deriving χ_t for some specific day t, we assumed c_t slots will be used. Further inspection suggests that some of the assumed-to-be-filled slots will not be filled. Denote variable f_i for order o_i the number of days on which order o_i is eligible and $c_t > 0$. In developing all χ_t's, order o_i occupies f_i slots. The actual total demand of order o_i is w_i. If $w_i < f_i$, then among the assumed-to-be-filled slots will never be used. Therefore, we can obtain the following value, which is denoted by Y_1:

$$Y_1 = \sum_{i=1}^{n} f_i - w_i \quad \text{for all orders } o_i \text{ satisfying } w_i < f_i.$$

In the example, the orders satisfying the condition on each day are shown in Table 6.

Table 6: Plan of admissible orders.

T	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
c_t	3	4	5	5	6	7	8	9	9	8	9	8	8	5	4	3
Order o_i	*C* *D* *E*	*B* *C* *D* *E*													*D* *F* *G* *J*	*D* *F* *J*

Therefore, we examine the orders {*B*, *C*, *D*, *E*, *F*, *G*, *J*} as in the following table.

Table 7: Details of admissible orders.

o_i	*B*	*C*	*D*	*E*	*F*	*G*	*J*
f_i	1	2	4	2	2	1	2
w_i	8	4	2	9	7	6	5
$f_i - w_i$	0	0	2	0	0	0	0

The value of Y_1 is 2, which resulted from order *D*.

The value of Y_2 and Z

In this section, we examine potential conflicts between each two orders. Two orders are said to conflict with one another if it is impossible to arrange both of them in a plan. Therefore, one of the two orders will not be fulfilled and thus needs to be discarded. A definition of conflicts will be given later. For example, when $m = 4$ and there is some day t with $c_t = 3$, if we find two orders eligible on this day and thus conflicting with each other. Then we can have $X = 1$ and $Y_2 = 1$ because on day t we can actually assign at most two orders. The conflict between two orders is determined as follows. We find the length of overlap between their time windows and the sum of demands for each frame type. If the sum of demands for *any* frame type exceeds the length of time window overlap, then the two orders conflict. The conflict occurs when two orders compete for the same type of ad space and their feasible time windows overlap in a large proportion. Consider the following two orders as an example, $o_i = (8, 0, 0, 0; 0, 10)$ and $o_j = (6, 2, 0, 0; 1, 11)$. The length of unified time window is 12,

from time t = 0 to 11. The sum of demands for type 1 is 8 + 6 = 14. It is impossible to arrange orders o_i and o_j both in a single plan because 12<14.

To determine the value of Y_2, we investigate the potential conflicts that may arise on each individual day. Assume there are k_t orders whose durations include day t. An undirected graph is constructed. Each order is represented by a node. If a conflict exists between two orders, then an edge is added to connect the two corresponding nodes. We select the largest number, say k_t', of orders such that they are mutually free from conflicts. Any of the remaining $k_t - k_t'$ orders has conflicts with at least one of the k_t' orders. In other words, at least $k_t - k_t'$ (lower bound) among the k orders cannot be scheduled on day t. Therefore, Y_2 can be calculated by summing $k_t - k_t'$ over all days if the value k_t' is less than m. There is one difficulty, however, in finding $k_t - k_t'$ for each t because it is equivalent to the maximum independent set problem, already known to be NP-hard. When considering the effectiveness and efficiency in the design of our upper bound scheme, we only examine the conflicts among eligible orders on the days when c_t is smaller than m+2. The computational challenge is avoided by enumerating all patterns that will help determine the lower bound value, $k_t - k_t'$. This simplicity stems from our previous assumption (or limitation) that we investigate only the cases where c_t is smaller than m+2. We separate the impossible lower bound by this conflict detection method into two parts. The Y_2 value covers days with c_t from 2 to m and Z value covers days with c_t from m+1 to m+2. In the Appendix, all patterns are enumerated with their associated lower bounds included.

We can see that Y_1 could be implemented if c_t ranges from 1 to m, and Y_2 could be counted if c_t ranges from 2 to any larger number. We will compare the effectiveness of these two methods when c_t ranges from 1 to m, and sum up the lower bound and the lower bound value from X. On the other hand, the lower bound value of Z is used when c_t is larger than m. The sum is our final impossible lower bound in the case.

In the numerical example, we first detect conflicts between orders F and J. Conflicts occur on day 14 and day 15. Examining the patterns given in the Appendix, we see that at most three of the four eligible orders can be scheduled on day 14, and at most two of the three orders can be scheduled on day 15. Therefore, Y_2 = 2.

Table 8: The c_t and Y_2 values in the plan.

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
c_t	3	4	5	5	6	7	8	9	9	8	9	8	8	5	4	3
Y_2	0	0	0	0	0									0	1	1

As aforementioned, although we can examine all of the conflict situations at time t with c_t larger than 2, for efficiency consideration, we only calculate the Y_2 value when the c_t value is between 2 and 6.

5. Computational Experiments

In this section, we explain the experiments of the banner scheduling problem to study the performance of the proposed placement algorithm. The upper bound will be also examined. The platform used for the experiments is a personal computer with a Pentium 4 CPU of 3.4 GHz PC and 512 MB memory, running Microsoft Windows XP. The programs are coded in Java.

In the experiments, the planning horizon is 365 days. The length of allowable duration, i.e., W_i specified by order o_i, is determined by probability: 10% for $W_i \in [1, 10]$; 35% for $W_i \in [11, 30]$; 40% for $W_i \in [41, 60]$; and 15% for $W_i \in [61, 90]$. A roulette wheel is used for the random selection. When each W_i is determined, the demands for different types of space are sampled with the uniform distribution. We randomly generate release date α_i and due date β_i of order o_i subject to the feasibility constraints $\sum_{j=1}^{m} w_{ij} \le \beta_i - \alpha_i$. For simplicity in comparing the effectiveness of schedules, we assumed the prices of all types of ad frames are the same. We use our heuristic to solve test instances with four, six, and eight types of ad space, with the number of orders being 50, 100, 150, 200, 300, and 400. The computational results shown in the tables below are average values across the 50 instances for each combination of number of space types and number of orders.

Information presented in the tables includes:

- n: Number of orders.
- #_*Orders_Scheduled*: Average number of orders successfully scheduled
- #_*Slots_Filled*: Average number of slots successfully filled.
- *UB*: Average upper bound.
- *Utilization* (%): Average percentage of filled slots out of all slots; that is, $\frac{\#\_Slots\_Filled}{\#\_Slots\_All} \times 100\%$.

- *Adjusted_Utilization* (%): Average percentage of slots filled based on *UB*; that is, $\frac{\#\_Slots\_Filled}{UB} \times 100\%$.
- Time (sec.): Average run time.

From the statistics presented in Tables 8, 9, and 10, we can make our first observation on the trend of utilization. When the number of orders increases, the placement algorithm achieves better utilizations of slots. When four types of ad spaces are provided for 50 orders, the utilization is 89.99%. When the number of orders received rises to 400, the utilization climbs to 97.95%. The trend occurs for different numbers of ad types.

Our second observation is about the lower bounds. Recall that we estimate the number of slots that are impossible to use in any plan. In other words, we give an upper bound that is an estimate of the slots that can be used. The numerical results reveal that upper bounds become tighter as the number of orders increases. This trend can be evinced from the three tables for different numbers of ad types. Another implicit relationship between upper bounds and numbers of ad types can be established. Consider the rows of $n = 200$ orders in the tables. The ratios between upper bounds and total number of slots are

$$\frac{1430}{1460} = 0.9794, \ \frac{2131}{2190} = 0.9733, \ and \ \frac{2817}{2920} = 0.9647.$$

The ratio appears to decrease (in other words, the upper bounds become tight) when the number of ad types increases, although the slop is slight.

This discussion has addressed the effectiveness of the placement algorithm. The utilization we cited above is calculated using the total number of slots. Developing *UB* has indicated that some fraction of the slots will never be used. Considering the "effective" slots, we have better utilization as shown in the columns titled *Adjusted_Utilization*. The results further demonstrate the impressive effectiveness of our algorithm.

We integrated several columns from Tables 8, 9, and 10 to create Table 11. This table shows us the average demand amount of orders we scheduled. We find the average demand of orders scheduled decreases when the *n* values increase; that is, our heuristic will successfully select and schedule the orders with fewer demands more easily, thus creating a better solution for this kind of scheduling problem.

The final concern of our discussion relates to efficiency, that is, the run time required to produce the final plan. Phase 2 and Phase 3 of the proposed placement algorithm are designed to improve the initial plan. Their execution depends on how many scheduled slots are moved around and how many unscheduled orders are examined and eligible for insertions. Therefore, no strict formula is given to describe the overall time complexity. Still, we circumvent rigorous theoretical analysis to look into the elapsed run time in the experiments. First, the actual run time does not exhibit an exponential growth of the number of orders. Consider the rows of $n = 100$, 200, and 400 in Table 9. The run times are 2.00, 7.39, and 22.48. The ratios between each two successive run times for the values of *n* doubled are 3.7 and 3.0. Then we turn to examine the role of *m*, the number of ad types. Consider the scenarios of $n = 200$ orders in the three tables. The run times for $m = 4$, 6, and 8 are 5.23, 7.39, 10.8. The ratios between 6 and 4 and between 8 and 6 are 1.50 and 1.33, respectively. The ratios between 7.39 and 5.23 and between 10.8 and 7.39 are 1.41 and 1.46, respectively. It appears that the ratio between two successive run times coincides with the ratio between two successive values of *m*. From the above discussion, we can see that the proposed algorithm's run time will not drastically deteriorate when the number of orders and the number of ad types increase. From a practical point of view, resolving 400 orders in 20+ seconds is acceptable.

6. Conclusions

This paper presents a new model, Scheduling of Banner Advertisement with Time Windows. In addition to adding a time window to each order, we also introduce heterogeneous types of ad spaces. A binary integer program is presented to mathematically describe the proposed model and potentially facilitate further developments of solution approaches and bounding strategies. We design a placement heuristic and an upper bound to this maximum utilization problem and implement them with four, six, and eight ad space types. The scheduling results show the performance of our heuristic is good and the upper bound we designed is tight.

When connecting to any of the most popular portals, our eyes may catch some interesting messages or funny games. We may seldom recognize that we are receiving advertisement information through a promotion process. Attracting more people to surf the Internet and access messages that advertisers want to distribute is the core value of web promotions. The scheduling of banner advertisements discussed in this paper is both interesting and crucial to portal websites. For further research, incorporating flexibility into on-line scheduling will convey an even more practical significance, although the difficulty in theoretical analysis will rise.

Table 8: m = Four types of ad space. Number of slots is 4×365=1,460.

n	#_*Orders*_ *Scheduled*	#_*Slots*_ *Filled*	*Utilization* (%)	*UB*	*Adjusted*_ *Utilization* (%)	*Time* (sec.)
50	38.20	1,313.9	89.99	1,390.57	94.49	0.37
100	48.93	1,400.3	95.91	1,420.50	98.58	1.61
150	53.63	1,423.5	97.50	1,435.67	99.15	3.34
200	57.23	1,430.0	97.95	1,437.93	99.45	5.23

Table 9: *m* = Six types of ad space. Number of slots is 6×365=2,190.

n	#_*Orders*_ *Scheduled*	#_*Slots*_ *Filled*	*Utilization* (%)	*UB*	*Adjusted*_ *Utilization* (%)	*Time* (sec.)
100	64.04	2,069.32	94.49	2,102.28	98.43	2.00
200	75.52	2,131.68	97.34	2,146.10	99.33	7.39
300	81.92	2,151.58	98.25	2,163.12	99.47	15.08
400	85.66	2,161.28	98.69	2,169.84	99.61	22.48

Table 10: *m* = Eight types of ad space. Number of slots is 8×365=2,920.

n	#_*Orders*_ *Scheduled*	#_*Slots*_ *Filled*	*Utilization* (%)	*UB*	*Adjusted*_ *Utilization* (%)	*Time* (sec.)
100	77.80	2,671.28	91.48	2,775.40	96.25	3.52
200	93.78	2,817.94	96.50	2,845.46	99.03	10.80
300	99.28	2,854.56	97.76	2,874.16	99.32	21.05
400	105.74	2,870.96	98.32	2,887.50	99.43	34.16

Table 11: Average demand amount of orders scheduled.

m	*n*	# of orders scheduled	# of slots allocated	average demand of orders scheduled
4	50	38.2	1313.9	34.40
4	100	48.93	1400.3	28.62
4	150	53.63	1423.5	26.54
4	200	57.23	1430	24.99
6	100	64.04	2069.32	32.31
6	200	75.52	2131.68	28.23
6	300	81.92	2151.58	26.26
6	400	85.66	2161.28	25.23
8	100	77.8	2671.28	34.34
8	200	93.78	2817.94	30.05
8	300	99.28	2854.56	28.75
8	400	105.74	2870.96	27.15

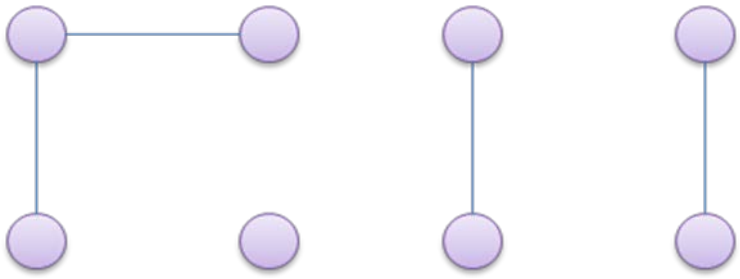
Appendix:

The following graphs depict the procedure for calculating the values of Y_2 and Z on the day with c_t value, and the conflicting situation between the eligible orders on the day. Each node means an eligible order and the line linking two nodes means conflict occurring between the two orders linked. We use *max* meaning the maximum number of orders not conflicting to each other. The $Y_2 = c_t - max$ in the conflicts situation between the orders on the same day showing in the following graphs if *max* is smaller than *m*. If the conflicts situation on some day is actually the models we draw below, we will get the *max* to evaluate the Y_2.

C_t = 4, 1 conflict lines

max = 2 for all other situations

C_t = 4, 2 conflict lines



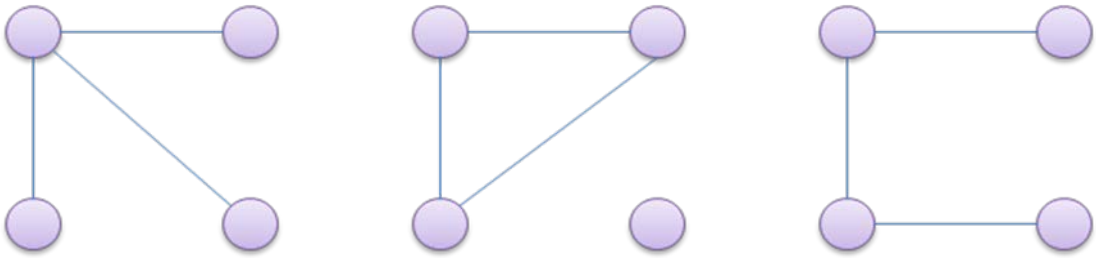
(2, 1, 1, 0)

max = 3

(1, 1, 1, 1)

max = 2

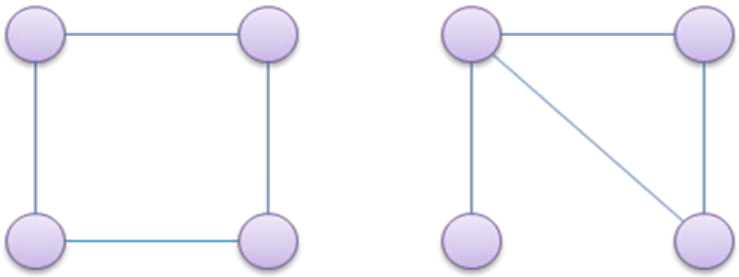
C_t = 4, 3 conflict lines



(3, 1, 1, 1) (2, 2, 2, 0) (2, 2, 1, 1)

max = 3 max = 2 for all other situations

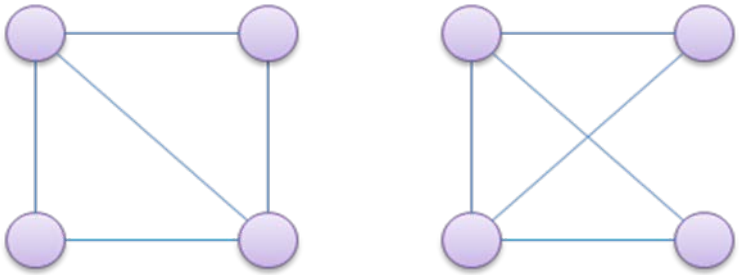
C_t = 4, 4 conflict lines



(2, 1, 1, 0) (1, 1, 1, 1)

max = 2 for all other situations

C_t = 4, 5 conflict lines



(3, 3, 2, 2) (3, 3, 2, 2)

max = 2 for all other situations

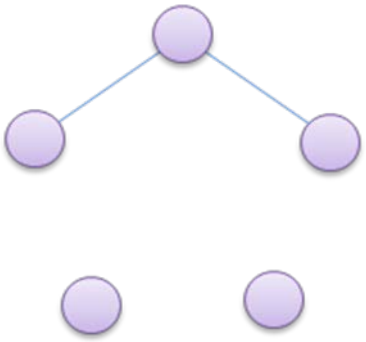
C_t = 4, 6 conflict lines

max = 1 for all other situations

C_t = 5, 1 conflict lines

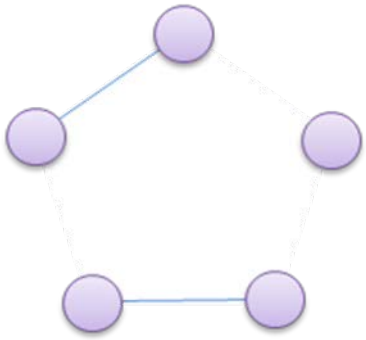
max = 4 for all other situations

C_t = 5, 2 conflict lines



(2, 1, 1, 0, 0)

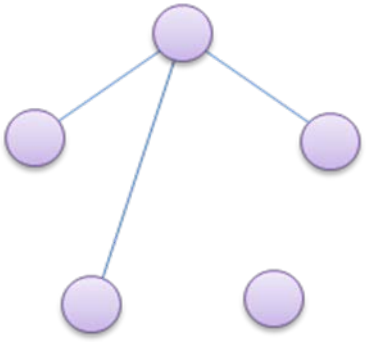
max = 4



(1, 1, 1, 1, 0)

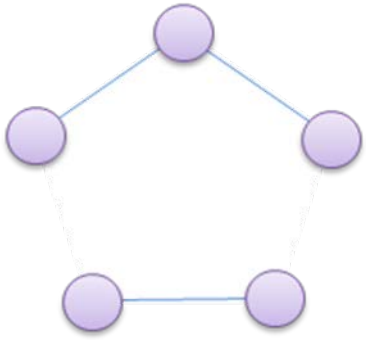
max = 3 for all other situations

C_t = 5, 3 conflict lines



(3, 1, 1, 0, 0)

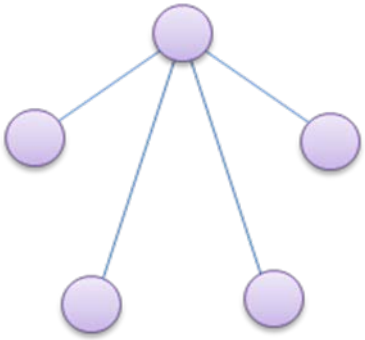
max = 4



(2, 1, 1, 1, 1)

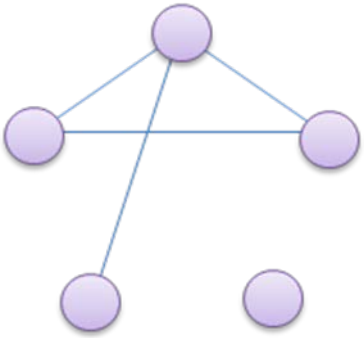
max = 3 for all other situations

C_t = 5, 4 conflict lines



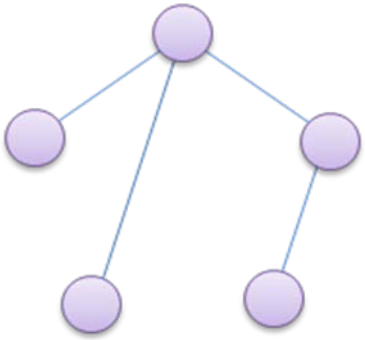
(4, 1, 1, 1, 1)

max = 4



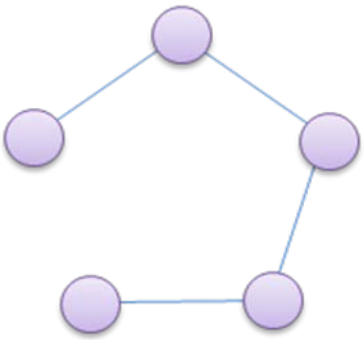
(3, 2, 2, 1, 0)

max = 3



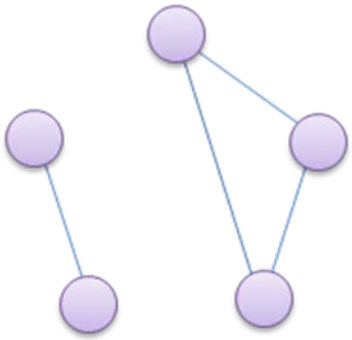
(4, 1, 1, 1, 1)

max = 4



(3, 2, 2, 1, 0)

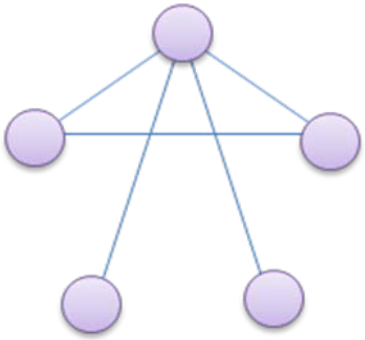
max = 3



(2, 2, 2, 1, 1)

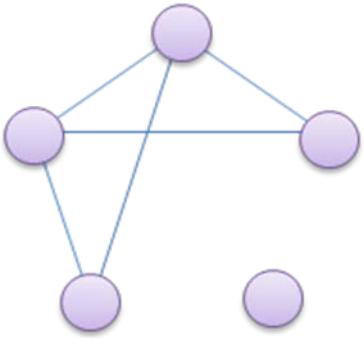
max = 2

C_t = 5, 5 conflict lines



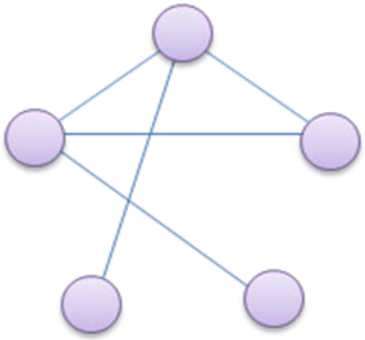
(4, 2, 2, 1, 1)

max = 3



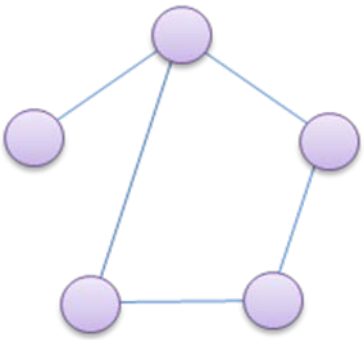
(3, 3, 2, 2, 0)

max = 3



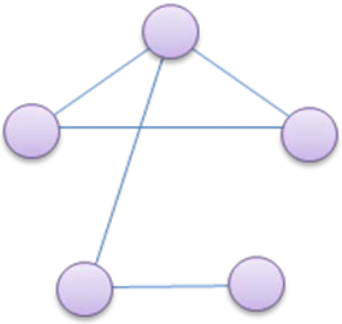
(3, 3, 2, 1, 1)

max = 3



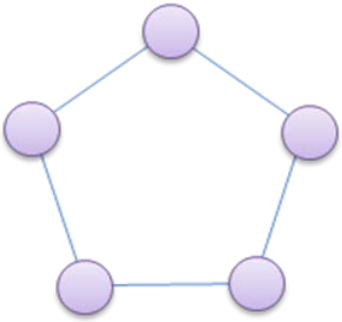
(3, 2, 2, 2, 1)

max = 3



(3, 2, 2, 2, 1)

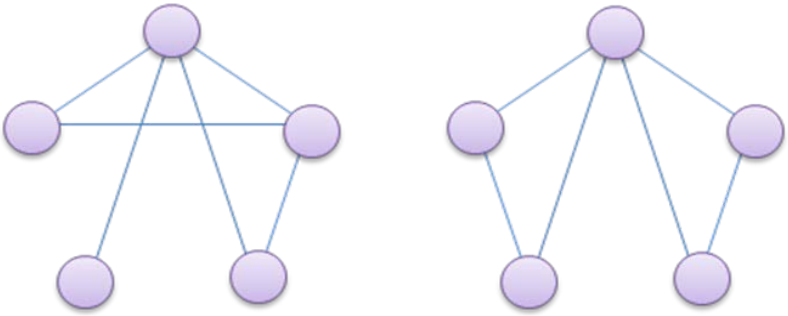
max = 2



(2, 2, 2, 2, 2)

max = 2

C_t = 5, 6 conflict lines

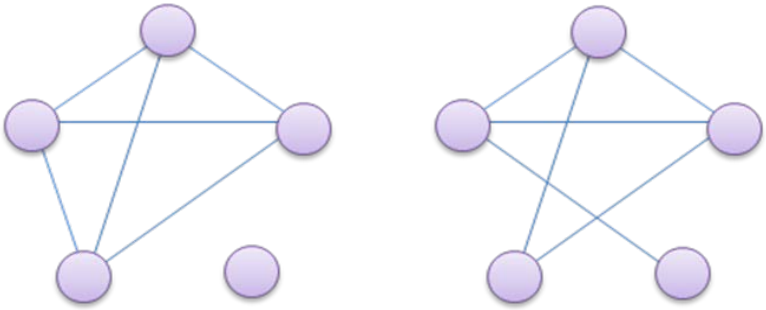


(4, 3, 2, 2, 1)

max = 3

(4, 2, 2, 2, 2)

max = 2

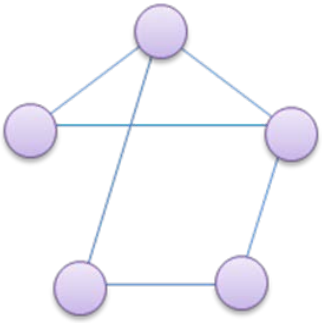


(3, 3, 3, 3, 0)

max = 2

(3, 3, 3, 2, 1)

max = 2



(3, 3, 2, 2, 2)

max = 2

References

1. Abramovich, G (2007) Microsoft to acquire aQuantive. *DMNews*. http://www.dmnews.com/Microsoft-to-acquire-aQuantive/article/95613/
2. *ADPAGENET*. http://www.adpagenet.com/advantage.html
3. Adler, M, Gibbons, PB, and Matias, Y (2002) Scheduling space-sharing for Internet advertising. *Journal of Scheduling* **5:** 103-119.
4. Amiri, A and Menon, S (2003) Efficient scheduling of Internet banner advertisements. *ACM Transactions on Internet Technology* **3**(4): 334-346.
5. Amiri, A and Menon, S (2006) Scheduling web banner advertisements with multiple display frequencies. *IEEE Transactions on Systems, Man, and Cybernetics- Part A* **36**(2): 245-251.
6. Boskamp, V, Knoops, A, Frasincar, F and Gabor, A (2011) Maximizing revenue with allocation of multiple advertisements on a web banner. *Computers & Operations Research* 38(10): 1412-1424.
7. Bollapragada, S., Cheng, H., Phillips, M., Garbiras, M., Scholes, M., Gibbs, T. and Humphreville, M. (2002). NBC's optimization systems increase revenues and productivity. *Interfaces* **32**(1): 47-60.

8. Bollapragada, S. and Garbiras, M. (2004). Scheduling commercials on broadcast television. *Operations Research* **52**(3): 337-345.
9. Bollapragada, S., Bussieck, M.R. and Mallik, S. (2004). Scheduling commercial videotapes in broadcast television. *Operations Research* **52**(5): 679-689.
10. Chickering, D.M. and Heckerman, D. (2003). Targeted advertising on the web with inventory management. *Interfaces* **33**(5): 71-77.
11. Dawande, M., Kumar, S. and Sriskandarajah, C. (2003). Performance bounds of algorithms for scheduling advertisements on a web page. *Journal of Scheduling* **6:** 373-393.
12. Dawande, M., Kumar, S. and Sriskandarajah, C. (2005). Scheduling web advertisements: a note on the minspace problem. *Journal of Scheduling* **8:** 97-106.
13. EADP (2007), Google, Yahoo, AOL and MSN get lion's share of online advertising, European Association of Directory and Database Publishers, March 2007. http://www.eadp.org/index.php?q=node/15530.
14. Freund, A. and Naor, J. (2004). Approximating the advertisement placement problem. *Journal of Scheduling* **7**: 365-374.
15. Johnson, L.D. (2007). Microsoft's acquisition of aQuantive Jabs at Google-DoubleClick. *Fast Company*. http://www.fastcompany.com/blog/lynne-d-johnson/digital-media-diva/microsofts-acquisition-aquantive-jabs-google-doubleclick
16. Kaye, K. (2007). Right Media buy pits Yahoo against Google's DoubleClick exchange. *The ClickZ Network*. http://www.clickz.com/showPage.html?page=3625712.
17. Liu, L. and Zhang, L. (2010). Study on an extended web advertising placement problem. *Proceedings of the International Conference on Information Science and Management Engineering* 373-377.
18. Menon, S. and Amiri, A. (2004). Scheduling banner advertisements on the web. *INFORMS Journal on Computing* **16**(1): 95-105.
19. Merriman, D.A. and O'Connor, K.J. Method of delivery, targeting, and measuring advertising over networks, United States Patent 5948061.
20. Payne, T., David, E., Jennings, N.R. and Sharifi, M. (2006). Auction mechanisms for efficient advertisement selection on public displays. *The 17th European Conference on Artificial Intelligence*: 259
21. Reyck, B.D. and Degraeve, Z. (2003). Broadcast scheduling for mobile advertising. *Operations Research* **51**(4): 509-517.
22. Szalai, G. (2011). Zenith Optimedia cuts global ad growth forecast due to middle East uprising, Japan earthquake. *Hollywood Reporter*. http://www.hollywoodreporter.com/news/zenithoptimedia-cuts-global-ad-growth-176713
23. IAB Internet Advertising Revenues Report (2011.04) http://www.iab.net/media/file/IAB_Full_year_2010_0413_Final.pdf.