國 立 交 通 大 學

資訊管理研究所

碩 士 論 文

指導教授：林妙聰 博士

有時間窗限制的線上廣告排程
Scheduling of Banner Advertisements with Time Windows

研究生：盧彥廷

中華民國九十六年六月

有時間窗限制的線上廣告排程

Scheduling of Banner Advertisements with Time Windows

研　究　生：盧彥廷　　　　Student: Yen-Ting Lu
指導教授：林妙聰　　　　Advisor: B.M.T. Lin

國立交通大學
資訊管理研究所
碩士論文

A Thesis
Submitted to Institute of Information Management
College of Management
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Business Administration
in
Information Management
June 2006
Hsinchu, Taiwan, the Republic of China

中華民國九十六年六月

誌謝

碩班兩年，過著可說是很愜意的生活，擁有一間小套房，一個溫暖的研究室與窩在裡面的夥伴，在專業而又開明的教授指導下，一步一步完成自己研究的課題。新竹，成了我的第二個家鄉，有熟悉的朋友，熟悉的景物，有忘不了的深刻記憶，唯一不足夠的，也許是我能待在這的時間吧。

筱嵐學姊，可說是我進交大後第一個熟識的朋友，雖然已經是博班新生了，比我多懂很多東西，可是年紀相近，有很多話可以聊，兩年來受了你很多的關心與照顧，很感心。延聰是同期一起奮鬥的夥伴，待人和善好相處，尤其又比我細心很多，讓我很多事可以趕在到期之前完成，真的很不容易啊。亞梅學姊進來得比較晚，可是是個生活達人，研究室裡大大小小的事物都暗藏著她的巧思，讓這個小地方，更像一個家。昱劭、癸棠兩個學弟，在碩班的第二年陪著我趴趴走，讓我有機會去新竹各處走訪，睡研究室時不必自己關燈，可說是讓我”獲益匪淺”。我要感謝林教授的除了學業上的指導之外，開明而包容的引導是讓我能快樂的做研究最大的原因，也讓 meeting 成了一件開心的事情。牛義是研究室外最好的朋友，幾乎什麼事都能跟你討論，也因爲你讓我有多一間研究室可以去。小六是資管所外最好的朋友，也是交大裡我認識最久的朋友，在我低潮時總能給我理性的諮詢與建議。小摩姊雖然不在交大，雖然偶爾就會消失，你的鼓勵與分享總帶來久旱甘霖的驚喜感。我的家人們，雖然常不在台灣，但也讓我成了有國際電話關心的孩子，知道你們工作很辛苦，給了我讀碩班的機會真的很感謝，讓我可以留下很多人生的體驗，結交很多的朋友。奶奶的離去，是兩年內最悲傷的事情，感謝那段期間曾給我幫助的所有人。其他如羽球殺手怡君，沉默酷哥平祺與網路研究室的朋友們，曾來訪新竹的菸鬼、鬼魅跟印地安，在此祝大家事事如意，平平安安，有沒頭路都賺大錢，有沒伴侶都不孤單，謝謝大家。

Title of Thesis: Scheduling of Banner Advertisements with Time Windows

Name of Institutes: Institute of Information Management

Student Name: Yen-Ting Lu Advisor Name: Professor B.M.T. Lin

Abstract

As the population of the Internet users increases explosively, the media power of the Internet becomes concrete and remarkable. One of the leading commercial applications on the Internet is banner advertisements publishing, which also is the major source of income of many portal websites. However, there are few researches on how to schedule banners efficiently and profitably in the past decades. We thereby propose a new model for banner advertisements scheduling that further resembles the Internet business world. In the proposed model, we apply a feasible time window to each candidate order, and we schedule advertisements into various predefined banner spaces on the webpage. This approach allows various pricing strategies according to banner types, contrary to previous research models which schedule and price by banner space sharing. Each order will demand its frequency of each type of banner. The goal is to select orders and schedule their demands to achieve a maximum total profit. 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. An upper bound on the 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 the ad spaces.

Keyword: Banner advertisements, operations scheduling, time windows

摘要

隨著網際網路使用者數量的快速累積，網際網路其媒體力量隨之變得具體而引人注目。網頁版面上的橫幅廣告為其最重要的商業應用之一，同時也是許多入口網站最主要的收入來源。然而，對於如何有效率的排程這些網路廣告以增加獲利的研究卻顯得稀少，因此，我們在這篇論文中，提出了一個更加貼近今日網際網路商業情形的模式。在此模式中，我們為每筆廣告訂單設定一段有效的刊登期間，然後站在廣告排程者的立場，努力滿足每筆訂單中對不同廣告欄位的次數需求，完全滿足者方可依各廣告欄位的訂價賺取收入。此模式不同於以往的研究，以廣告欄位的種類計價，而不以所占用的分割廣告空間大小為計價方式。而排程的目標，則是選取訂單並滿足其訂單內容，以獲得最大的營收。在這篇論文裡，我們先提出一個正式的數學式描述這個題目，考量此題目的計算複雜度，我們試著在合理的運算時間下尋求近似解。我們設計了一個三段式解題的啟發式演算法來解題，也設計一個推算可得利潤上限的方法，最後實作了此啟發式演算法，實驗結果顯示此法可有效的填滿百分之九十七的總廣告欄位。

關鍵字: 網路, 廣告, 時間窗

Table of Contents

Chapter 1 Introduction

Since the emergence of advertisements, ads are seldom absent on any influential media, like flier, newspaper, radio, television, and there is no exception of the Internet. We can hardly find one website or portal with business power but without hanging advertisement banners on their board in this Internet times. There are 99 percent of all web sites offer standard banner advertisements (Buchwalter et al. 2001). The growth rate of the Internet ads market is still booming in 2006. The IAB Internet Advertising revenue report (2006 second-quarter and first six-month results) shows that the growth rate of the second quarter revenue based on its of first quarter in 2006 is 5.5% (4,061 versus 3,848 million) , and the growth rate of first 6-month revenue of 2006 based on its of 2005 is 36.7% (7,909 versus 5,787 millions). Although the unique price of Internet advertisement has not been yet as high as its of TV advertisement, the latter can be charged thousands of dollars to show on TV for couple seconds and this investment being effective and profitable cause the population of TV viewers is so large, but many properties on the Internet are worthy to utilize, for example, the lower cost and more convenience of connection to other countries and even other continents. In 2006, England and Sweden have the online advertisements market beyond 10% of their total domestic market of advertisements, these two countries are the first two in the world with two-digit percent market share, and it may happen on Australia, Israel, Japan, Norway, South Korea, and Taiwan before 2008 predicted by eMarketer. The value of web ads may be illustrated by some decisions of the famous IT companies, Google acquiring Doubleclick for $3.1 billion and Yahoo buying RightMedia for $680 million in April 2007, and Microsoft buying aQuantive for $6 billion in May 2007.

As the information technologies progress rapidly, the popularity of internet advertising rises with many unique advantages which had not ever appeared on other traditional media. Four main attractive characteristics of internet advertising:

1. *Interaction*: The advertisements on the Internet are not just showing and delivering information to browsers like ads on papers or TV. They can interact with their viewers step by step, like an extension of direct marketing. This feature can catch the viewer's attention effectively and communicate with the targeted audience in depth.

2. *Traceability*: One of the most convenient functions of the Internet is hyperlinks, and the users of the Internet can reach the websites of the ad hosts easily and rapidly from the page they are viewing, and even finish the whole transaction process just on the net. This function makes the feedback and effectiveness of advertising traceable and measurable. It is a very useful tool for ads hosts to make decisions and to improve advertisement quality.

3. *Segmentation*: Website hosts can collect visitors' information by online table directly or record their behavior on the Internet at a low cost. It is very useful for targeted promotion, which not only improves the effectiveness of advertising but also reduces the possibility of disturbing the people who are not happy to receive the ads.

4. *Flexibility*: Websites on the Internet can operate around the clock. Hosts can modify and update the ads easily anytime. This makes the promotion activities more flexible and powerful. One of the remarkable instances is the online auction.

As the importance of web advertisement increases, many researches focus on this subject in recent years. One part of researches is studying how to improve the effectiveness of web advertisement of each impression, and they are based on many different social science methodologies. Some researches founded the models and algorithms to select the attractive advertisements for the specified web guests by observing and recording their behavior and hobbies on the Internet. Some researches made effort to classify the types of advertisements and the types of web context (or content), and then put the relative advertisements on the matching web pages. Google, Yahoo!, AOL, and MSN the four big portals in American made

$12.5 billion in 2005, which is 53.7% of Internet ad spending in the year (EADP 2007). The phenomenon that revenues of web ads center on these extremely big portals is not only caused by their huge mass browsers but also the various services they provide which can hold the visitors longer time, be useful to website managers to record the browsers' behavior, target their interests, and raise the effect of ads impressions. United States Patent No. 5948061, "Method of Delivery, Targeting, and Measuring Advertising over Networks" belong to Double Click Inc. is one famous instance revealing the commercial value of spreading strategy of ads.

Another part of researches is studying how to select and allocate the advertisement orders to make best profits for advertisement agents or website owners. Internet advertising have to face the same challenge like ads on other media, how to fill in the ads to the contents which audience want to receive and not disturb them too much, and get the maximum revenues of media owners with the limited display space/time slots. This is the topic we study in this paper and we will review the literature below.

Before go back to our topic on the Internet, we may examine some researches which developed practical and applicable methods useful to schedule TV advertisements and were taken in practice by NBC. One special property of TV advertisements is that the first and last slots in a commercial break are more valuable because of their higher audience ratings than other slots, and the schedulers of TV ads have to guarantee the advertisers the proper percentage of their ads appearing in the hot period (i.e. the first or last slots in a break). Besides, each ad being to be displayed must be classified to some specific type and two ads of competing hosts belonging the same type are not allowed to appear in the same break. Bollapragada et al. (2004) presented a formal mathematical program formulation for this problem and developed a heuristic replacing the existing manual schedule process of NBC. Another property of TV advertisements is that advertisers hope their ads could be displayed as evenly as possible during the feasible periods, by each identical ad as a unit. Bollapragada et al. (2004) presented a formal formulation for this problem and developed heuristics for different size of problems to get the near-optimal solution within reasonable computing time.

The scheduling problem of Internet advertising was first formulated by Adler et al. (2002), who presented a space-sharing model for this scheduling problem. They defined a fixed area for advertising on the webpage, and each ad needed put on has its specified geometry size and displayed frequency. One special constraint for the Internet ads scheduling problem is, we cannot put the same ads at the same time slot due to the real effectiveness consideration, for all the types of problems we will mention below. For the first type of problem in this paper, our schedule has to satisfy the frequency of each ad and cannot violate the space constraint at any moment, and our objective is to use the minimal time slots to finish all the tasks, and we can view this problem as a variant of bin packing problem. Another type of problem is that we set a fixed number of time slots (limited planning time horizon), and select the optimal subset of ads to maximize the revenues of ad agents, based on the assumption that the price of each ad showed once is in direct ratio to its display space. The authors also provide efficient algorithms in this paper to find optimal solutions to the former problem with specified restrictions and 2-approximation solutions for the latter problem. They also discuss how to make a decision to take the orders or not in online condition, and one algorithm was designed for it in the paper.

The maximum revenues problem in offline mode, which means all demands of ads must be implemented, was broadly discussed and studied, and many algorithms and approximation results were published. Amiri et al. (2003) used Lagrangean decomposition to solve the problem in relatively short computing time to get good results. Dawande et al. (2003) provided the first known algorithms with constant factor approximations to this problem, the simple one guarantees 1/4 and the complex one does 3/10 ratio of their performance to upper bound value presented in this paper. Freund et al. (2004) designed a algorithm guarantee 1/(3+ ε)-approximate for general case and 1/(2+ ε) for two special cases.

For the same objective function, to get the maximum revenues of ad agents, Menon et al. (2004) introduced another model to the Internet ads scheduling problem. They relaxed the constraint of reaching all demands of accepted ads, and this model allowed the ad agents to make better revenues but just satisfy partial demands of the ads in schedule. They used Lagrangean decomposition and column generation to solve this problem and compare their performances, and the column generation got better results and took a shorter time. Amiri et al. (2006) present a more flexible and practical model, the same relaxation on demands of ads as mentioned before, but a more precise pricing scheme for the real world trading. The ad customers can list out several levels of demands been implemented, and each display of ad need to pay to ad agents by its levels, the level with more counts of display, the higher individual price of each display. Authors of this paper deploy a Lagrangean decomposition-based solution procedure to deal with problems of this new model, and this approach can perform very well for reasonably large problems.

Except the maximum revenues objective, some researches were trying to solve the problem call "MINSPACE", defined by Dawande et al. (2003), we set a fixed number of time slots for use, and we have to schedule all the ads with their demands exactly, and try to minimize the maximum slot fullness. The solution of this problem can help the ad agents decide what size the banner is being efficient and suitable for their customers' demands. Dawande et al. (2003) provide a 2-approximation algorithm for this problem. Dawande et al. (2005) present an online version of the MINSPACE problem, and provided an algorithm with the performance bound of (2-1/N), and they provided two off-line algorithms with performance ratios $(1+1/\sqrt{2})$ and 3/2 , respectively. The N stands for the maximum capacity of one slot.

In this thesis, we present a new model that schedules the off-line orders contents with static pricing scheme but limit the feasible time window for each ad order (i.e. release date and due date), with the same objective of maximizing revenues of ad agents. The rest of this thesis is organized as follows. We present a new model called Scheduling of Banner Advertisements with Time Windows in Chapter 2. In Chapter 3, we introduce our heuristic to solve this problem and implement the heuristic with a small size data set. In Chapter 4, we design one upper bound for our objective, and we will implement the upper bound with the same data set. Chapter 5 is our computational experiments and analysis of the results. Chapter 6 is the conclusion and remarks.

Chapter 2 Problem statement and preliminaries

Formal Formulation of the Problem

A web site offers *m* different types of advertisement space $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$, in which space type s_j charges p_j per day. There are *n* orders placed $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$. An order may contain ads of one or more types. Each order o_i has a time window $[\alpha_i, \beta_i]$ that specifies the duration of its ad activities. Denote by w_{ij} the number of days on which ads of space type s_j are required by order o_i. The days scheduled for an order are not necessarily consecutive, i.e. all ads of an order may scatter non-consecutively within the given window. An order is fulfilled and the web site gets paid only if (1) All ads described in the order are scheduled within the window; (2) On each day of the schedule, at most one ad of any type from the order can be scheduled due to the real effectiveness consideration. Subsequently, we need the assumption $W_i = \sum_{j=1}^{m} w_{ij} \leq \beta_i - \alpha_i$ to maintain the feasibility of each individual order. For the collection $\mathcal{O}$ of *n* orders, a scheduling horizon of *T* days is given. Our objective is to maximize the revenues by taking some orders and scheduling the demand of orders successfully.

To describe the studied problem better, in the following we give an integer programming (IP) formulation. Binary decision variables x_{ijt} to be 1 if ad type s_j in order o_i is scheduled on day *t* for $1 \leq i \leq n$, $1 \leq j \leq m$, $1 \leq t \leq T$, and y_i to be 1 if order o_i is selected and scheduled successfully for $1 \leq i \leq n$.

(Problem *P*)

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

subject to

$$\sum_{j=1}^{m} x_{ijt} \le 1, \text{ for any order } o_i \text{ and day } t; \qquad (1)$$

(No two ads from the same order can be scheduled on the same day.)

$$\sum_{t=\alpha_i}^{\beta_i} x_{ijt} = w_{ij} y_i, \text{ for any order } o_i \text{ and space type } s_j\,; \qquad (2)$$

$$\sum_{t=1}^{\alpha_i - 1} x_{ijt} = 0, \text{ for the ad of space type } s_j \text{ in order } o_i; \qquad (3)$$

$$\sum_{t=\beta_i + 1}^{T} x_{ijt} = 0, \text{ for the ad of space type } s_j \text{ in order } o_i; \qquad (4)$$

$$x_{ijt} = \begin{cases} 1 \text{ if ad type } s_j \text{ in order } o_i \text{ is scheduled on day } t; \\ 0 \qquad\qquad\qquad \text{otherwise.} \end{cases}$$

$$y_i = \begin{cases} 1 \text{ if order } o_i \text{ is scheduled;} \\ 0 \qquad \text{otherwise.} \end{cases}$$

Chapter 3 Placement Algorithm

In this chapter, we will present a heuristic algorithm to deal with the banner ads scheduling problem of revenue maximization. The designed algorithm will be applied to a test set for an illustration.

The heuristic consists of 3 phases so as to maximize revenues in web banner scheduling. Phase 1 of the algorithm seeks to derive an initial solution. It first fills up the slots in a forward manner and then changes the direction to allocated unscheduled orders in a backward manner. In phase 2, the algorithm manages to arrange the remaining orders by shifting the scheduled ads without sacrificing the feasibility of scheduled orders. In the phase 3, we cancel the scheduled orders of less commercial value so as to arrange orders that can provide better gains. Because we do not have the pricing data of the real industry, we define the sequence of ad types in the non-increasing order of its unique price in the operation of our heuristic and the heuristic can be easily transfer to an applied model by inserting the price scheme of ad types. We will temporarily set an identical price for every ad type to evaluate the performance of our heuristic in the following example and the experiment in chapter 5. To better describe the algorithm, we use the following data set for an illustration.

Consider n = 10 orders {A, B, …, I, J}, m = 4 types of ads space with scheduling horizon of T = 16 days. The specifications of orders are given in 6-tuple vectors. For example, order A consists of 2 days of type 1, 3 days of type 2, 4 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 one, called **putS**, selects orders based upon 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 method, called **rev_putS**, which runs in the reverse direction to **putS**, and schedule the orders backward from the rear end of the planning horizon. We selects the unscheduled order with the latest due date to the earliest scheduled slots on the target type frame, and put in the ads of the order with the sequence from the target slot type s_j to s_{j-1},…,s_0, s_m, s_{m-1},…, s_{j+1}.

The deployment of the two approaches depends on the current time point under consideration. **PutS** is applied first. If the completion of the order just finished 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**, and set the next slot type as the target type to start with for the next round. After finishing m rounds, we will check if any unscheduled order can be further completed by **putS** or **rev_putS**. The following outlines the main steps of Phase 1:

1. Start with frame of type 1 as the target frame for the round 1.
2. Schedule the order with the earliest release date as the first one by **putS**, and schedule the next order which has 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 the orders are examined once.
3. Move on to the next target frame until all frames are considered as the target frame once.
4. Apply **putS** and **rev_putS** to dispatch all the unscheduled orders.

Following is the detailed implementation of Phase 1:

1. Start with target frame type $s_j = 1$
2. Set time pointer $p = 0$, tag all un-scheduled orders un-checked.
3. If 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 datẹ α_i to p and try to put the order in schedule by method **putS** and tag o_i checked. If fails, go to step 3 ; if succeeds, set p = the latest date when ad of o_i been put in space j and go to step 3.
6. Set time pointer $q = T$
7. If 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 and try to put the order in schedule by method **rev_putS** and tag o_i checked. If fails, go to step 7; if succeeds, set q = the earliest date when ad of o_i 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 the 10-order instance will produce the plan shown in Table 3.1.

Table 3.1: 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 in the following. The success of applying **putS** to some order is denoted by "s" and failure by "f". Same 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 move the filled slots around so as 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 another method, called **shiftB**. If **shiftB** also fails, then we skip the considered order and move on 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 α_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 we move the scheduled ad to other feasible date and put it the current ad. If the slot has been occupied by some ad of other order, move the scheduled ad to some other feasible date and put in the current ad. No matter whether if we can arrange the current ad successfully or not, we keep examining the next time slot until the ads required on the slot type are successfully placed or the time β_i is reached. If the scheduling of the 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, we put the ads in by the sequence of space type with non-increasing of remaining jobs to each frame. The remaining operations of **shiftB** are the same as those of **shiftA**.

From Table 3.1, 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 have the following results in which the plan is augmented by inserting order *A*. The new plan is shown in Table 3.2.

shiftA(*A*) = s ;

shiftA(*E*) = f ; shiftB(*E*) = f ;

shiftA(*H*) = s ;

shiftA(*J*) = f ; shiftB(*J*) = f ;

Table 3.2: 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 3.1 shows part of the scheduling procedure in Phase 2 for one order. The process with "*" in Figure 3.2 uses method **shiftA** to schedule the ads of any single order. If the trial of scheduling the order fails, we will clear all the ads belonging the order on the schedule, and use method **shiftB** to schedule the order again.

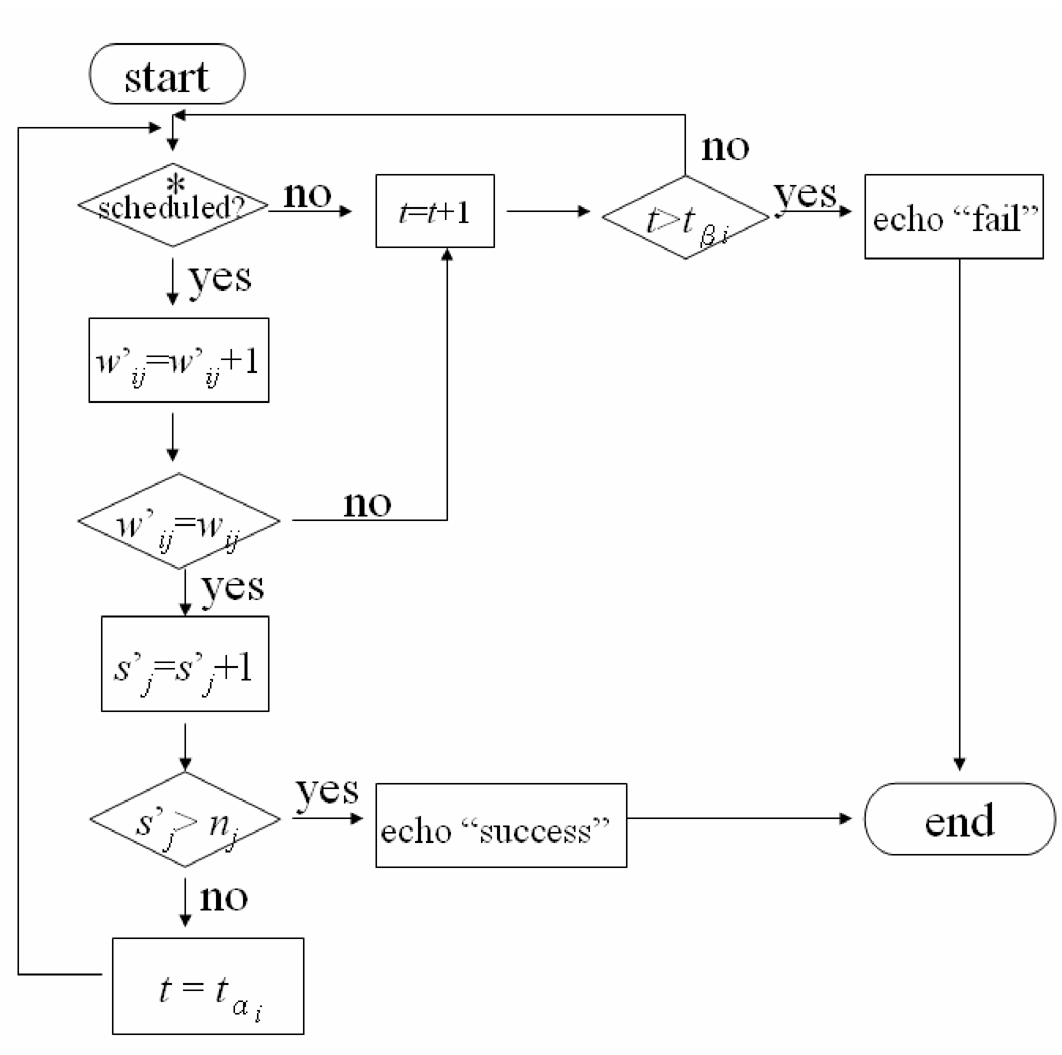


Figure 3.1: Operations of Phase 2.

Phase 3: Exchange scheduled and unscheduled orders

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

1. Maintain two sets of orders, P = the set of scheduled orders and Q = the set of unscheduled orders. Sort the orders of Q in non-increasing order of its commercial value.
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 is in 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 of Q that can successfully take the place of the removed order o_i. If such a replacement is successfully 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 3.2, we have the following detail steps and a new plan in Table 3.3

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 3.3: 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 3.4: 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*		

Figure 3.2 shows the operations of Phase 3. The process with "*" in Figure 3.3 uses the four scheduling methods used before, including **putS**, **rev_putS**, **shiftA**, and **shiftB**. If all methods cannot successfully schedule the order, we will give up this order and proceed to another unscheduled order.

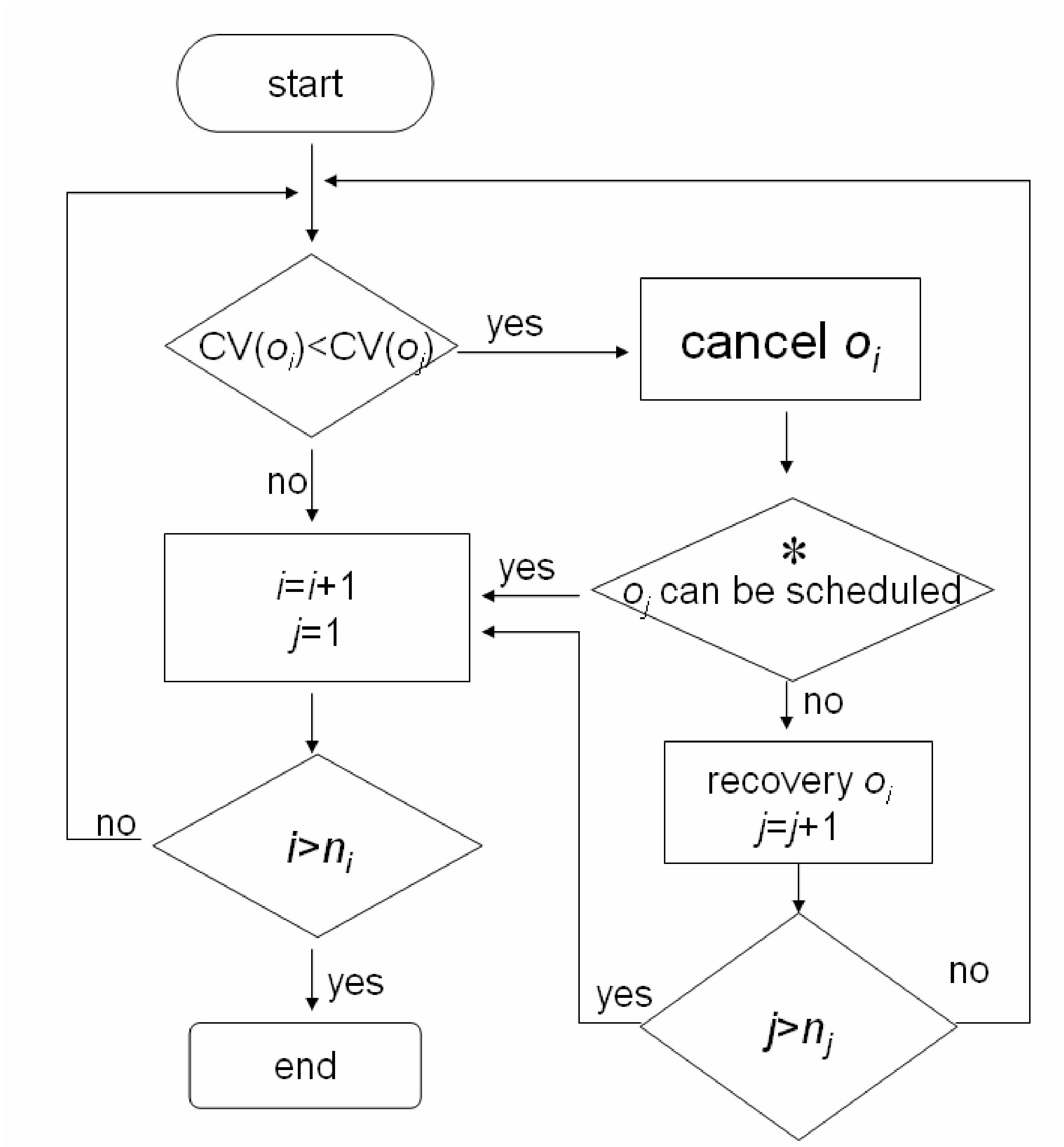


Figure 3.2: Operations of Phase 3.

Chapter 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 as well as to measure the quality of the plan produced by the proposed placement algorithm. In this section we present theoretical analysis of an upper bound by scrutinizing the profiles of the given orders. The upper bound is developed by consider the other side of the question: deriving a lower bound on the number of slots that is impossible to be used. In the development, the lower bound, denoted by *LB*, is expressed as

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

In the following, the derivation of each term will be explained in details. We will use a numerical example for an illustration for each term.

Initialization

For day $t = 1$ to T, define variable c_t as the number of orders eligible for time t, i.e 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 the following:

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 $\boldsymbol{X} = \sum_{t=1}^{m} \chi_t$ slots can never be used, where

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

In the example data, we have $\boldsymbol{X}$ = 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 the development of 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 and denoted it 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 the following table.

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*.

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, resulted from order *D*.

***The value of** Y_2* **and Z**

In this part, we examine the potential conflicts between each two orders. Two orders are said to have conflict with each other if it is impossible to arrange both of them in a plan. Therefore, one of the two orders will not be fulfilled and needs to be discarded. 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 2 orders eligible on this day and conflicting with each other, then we can have $X = 1$ and $Y_2 = 1$ because on day t we can actually assign at most 2 orders only. The conflict between two orders is determined in the following. We find the length of the union of 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 union of time windows, then the two orders have conflicts. 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, and the sum of demands for type 1 is 8 + 6 = 14. Because 12 < 14, it is impossible to arrange orders o_i and o_j both in a single plan.

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. A undirected graph is constructed. Each order is represented by a node. If a conflict does exist 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. However, there is one difficulty in finding $k_t - k_t^{'}$ for each t because it is equivalent to the maximum independent set problem, which is already known to be NP-hard. For considerations of 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, $k_t - k_t^{'}$. The simplicity stems from our previous assumption (or limitation) that we investigate 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 Appendix, all patters 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 plus the lower bound value from X and the lower bound value of Z when c_t is larger than m and the sum is our final impossible lower bound in the case.

In the numerical example, we will first detect conflicts between orders F and J. Conflicts occur on day 14 and day 15. Examining the patterns given in Appendix, we have that at most 3 of the 4 eligible orders can be scheduled on day 14, and at most 2 of the 3 orders can be scheduled on day 15. Therefore, Y_2 = 2.

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.

Chapter 5 Computational Experiments

In this chapter, we design 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 of 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 $W_i \in [61, 90]$, and. A roulette wheel is used for the random selection. When each W_i is determined, the demands for different types of space is 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 constraint $\sum_{j=1}^{m} w_{ij} \le \beta_i - \alpha_i$. For the simplicity in comparing the effectiveness of schedules, we set the prices of all types of ad frames are assumed to be the same. We use our heuristic to solve test instances with 4, 6, and 8 types of ads space, and the numbers of orders are 50, 100, 150, 200, 300, and 400. The computational results shown in the tables below are average values over the 50 instances for each combination of number of space types and number of orders.

Notations used in the tables include:

- *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, i.e. $\frac{\#\_Slots\_Filled}{\#\_Slots\_All} \times 100\%$.
- *Adjusted_Utilization* (%): average percentage of slots filled based on *UB*, i.e. $\frac{\#\_Slots\_Filled}{UB} \times 100\%$.
- time (sec.): average run time.

From the statistics are shown in Tables 5.1, 5.2, and 5.3, we make our first observation on the trend of utilization. When the number of orders increases, the placement algorithm achieves better utilizations of slots. When 4 types of ads spaces are provided for 50 orders, the utilization is 89.99%. If the number of order we receive is 400, then the utilization climbs to 97.95%. The trend occurs for different number of ads types.

Second observation is made on the lower bounds. Recall that we estimate the number of slots that are impossible to be used in any plan. Or, 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 order increase. This trend can be evinced in the three tables for different numbers of ads types. Another implicit relationship between upper bounds and numbers of ads types can be established. Consider the rows of n = 200 orders in Tables 5.1, 5.2 and 5.3. The ratios between upper bounds and total number of slots are

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

The ratio appears to decrease (or another interpretation is that the upper bounds become tight) when the number of ads types increases, although the decrease is slight.

The discussion has addressed the effectiveness of the placement algorithm. Still we need to further present remarks on this issue. The utilization we cited above is calculated using the total number of slots. The development of *UB* has indicated that some fraction of the slots will never be used. Considering the “effective” slots, we have better utilizations as shown in the columns entitled *Adjusted_Utilization*. The results further demonstrate the impressive effectiveness of our algorithm.

We integrate some columns in the table 5.1, table 5.2, and table 5.3 to get the Table 5.4, which shows us the average demand amount of orders we scheduled. We can find the average demand of orders scheduled decrease when the n values increases, i.e. our heuristic will select and schedule successfully the orders with small demand easier and make a better solution for this kind of scheduling problems.

The final concern of our discussion is about the efficiency, i.e. the run time required by producing the final plan. Phase 2 and Phase 3 of the proposed placement algorithm are designed for improving the initial plan and 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 of all, the actual run time does not exhibit an exponential growth of the number of orders. For example, consider the rows of n = 100, 200, and 400 in Table 5.2. 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. We turn to examine the role of m, the number of ads types. Consider the scenarios of *n* = 200 orders in 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 realize that the run time of the proposed algorithm will not drastically deteriorate when the number of orders and the number of ads types increase. From a practical point of view, solving 400 orders in twenty some seconds is acceptable.

Table 5.1: $m = 4$ types of ads 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 5.2: $m = 6$ types of ads 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 5.3: *m* = 8 types of ads 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 5.4: the average demand amount of orders scheduled

m	*n*	orders scheduled	slots sold out	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

Chapter 6 Conclusions

We present a new model called Scheduling of Banner Advertisement with Time Windows in this paper. Except the addition of time window to each order, we also replace the pricing scheme in the previous research of web banner scheduling, with price levels to different types of ad spaces. We design a heuristic and a upper bound to this maximum revenues problem and implement them with 4, 6, and 8 ad space types, respectively. The scheduling results show the performance of our heuristic is good and the upper bound we designed is tight.

Link to one of the most popular portals, your eyes may be caught by some interesting messages or funny games, and you may never sense that you are receiving the advertisement information after the whole promotion process. Attract people surfing on the Internet to access the messages you want to distribute actively, it is the core value of web advertisements. The scheduling of banner advertisements discussed in this thesis is a very important and valuable subject to the hosts of the powerful websites, and we think adding the on-line scheduling quality may make the research more practical and flexible. Furthermore, we think how to integrate different web advertising tools including banner ads, keyword searching, and classified ads, etc. to achieve the impression number required by ad hosts and make maximum revenues of websites managers being an interesting and useful subject to research.

References

[1] Adler, M., P.B. Gibbons, Y. Matias. 2002. Scheduling space-sharing for Internet advertising. *Journal of Scheduling*. **5** 103-119.

[2] Amiri, A., S. Menon. 2003. Efficient scheduling of Internet banner advertisements. *ACM Transactions on Internet Technology*. **3**(4) 334-346.

[3] Amiri, A., S. Menon. 2006. Scheduling web banner advertisements with multiple display frequencies. *IEEE Transactions on Systems, Man, and Cybernetics- Part A: Systems and Humans*. **36**(2) 245-251.

[4] Bollapragada, S., H. Cheng, M. Phillips, M. Garbiras, M. Scholes, T. Gibbs, M. Humphreville. 2002. NBC's optimization systems increase revenues and productivity. *Interfaces*. **32**(1) 47-60.

[5] Bollapragada, S., M. Garbiras. 2004. Scheduling commercials on broadcast television. *Operations Research*. **52**(3) 337-345.

[6] Bollapragada, S., M. R. Bussieck, S. Mallik. 2004. Scheduling commercial videotapes in broadcast television. *Operations Research*. **52**(5) 679-689.

[7] Chickering, D.M., D. Heckerman. 2003. Targeted advertising on the web with inventory management. *Interfaces*. **33**(5) 71-77.

[8] Dawande, M., S. Kumar, C. Sriskandarajah. 2003.Performance bounds of algorithms for scheduling advertisements on a web page. *Journal of Scheduling*. **6** 373-393.

[9] Dawande, M., S. Kumar, C. Sriskandarajah. 2005. Scheduling web advertisements: a note on the minspace problem. *Journal of Scheduling*. **8** 97-106.

[10] Freund, A., J. Naor. 2004. Approximating the advertisement placement problem. *Journal of Scheduling*. **7** 365-374.

[11] Menon, S., A. Amiri. 2004. Scheduling banner advertisements on the web. *INFORMS Journal on Computing*. **16**(1) 95-105.

[12] Payne, T., E. David, N.R. Jennings, M. Sharifi. Auction mechanisms for efficient advertisement selection on public displays. *The 17th European Conference on Artificial Intelligence*. 259

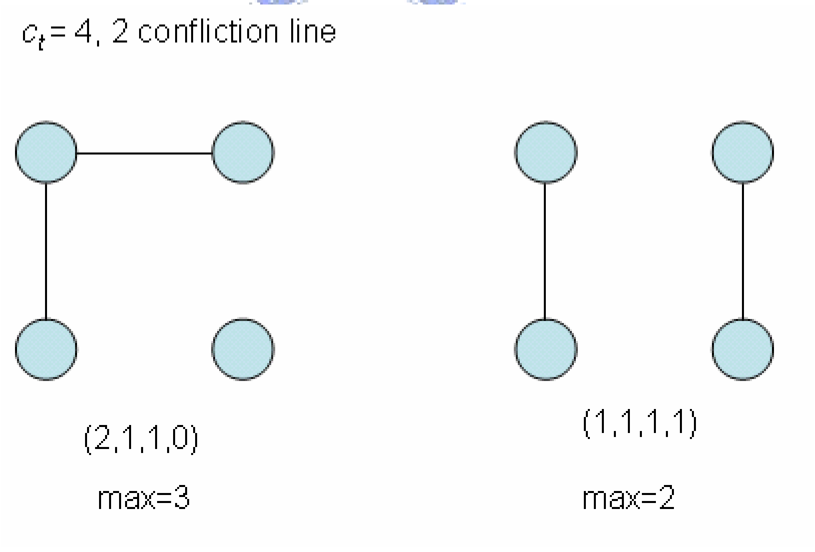
[13] Reyck, B.D., Z. Degraeve. 2003. Broadcast scheduling for mobile advertising. *Operations Research*. **51**(4) 509-517.

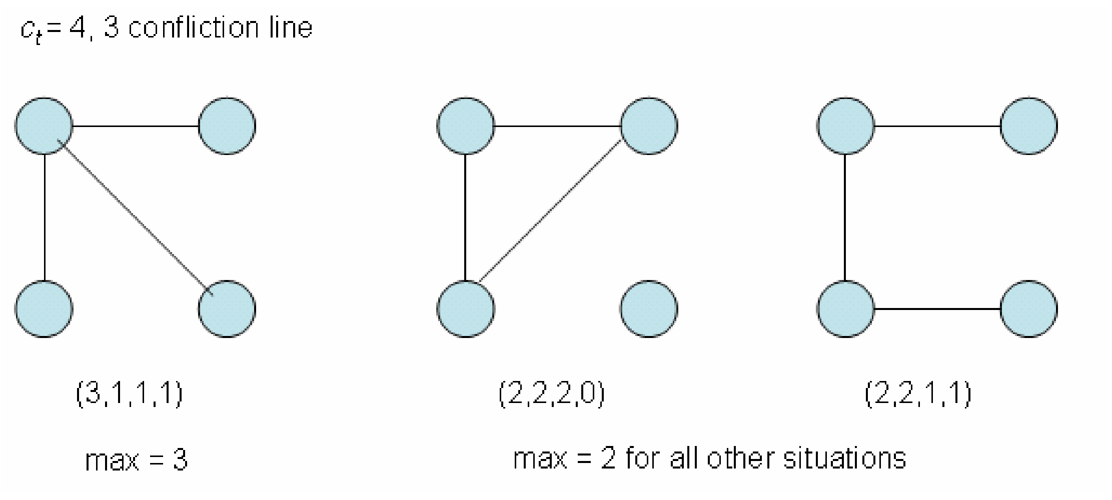
Appendix:

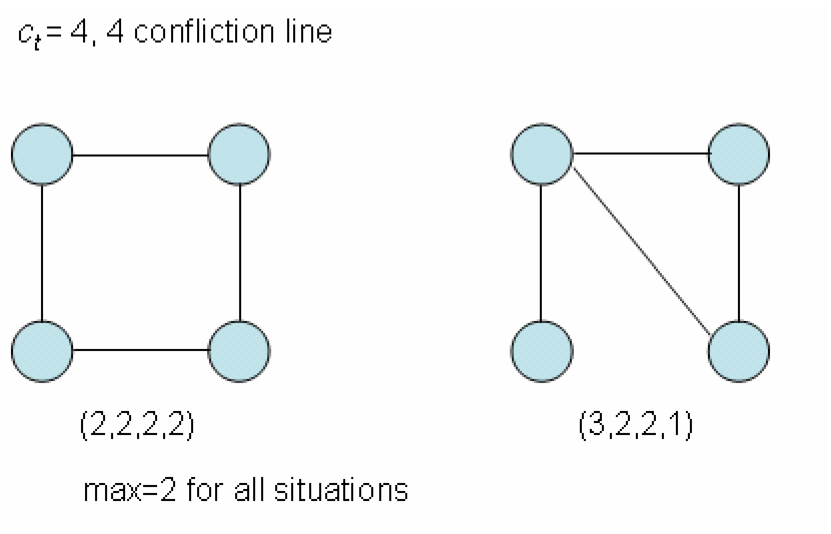
The following graphs show us how to count the Y_2 and Z on the day with c_t value, and the conflicting situation between the eligible orders on the day. Each node means an eligible order and the line linking two nodes means conflict occurring between the two orders linked. We use *max* meaning the maximum 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.

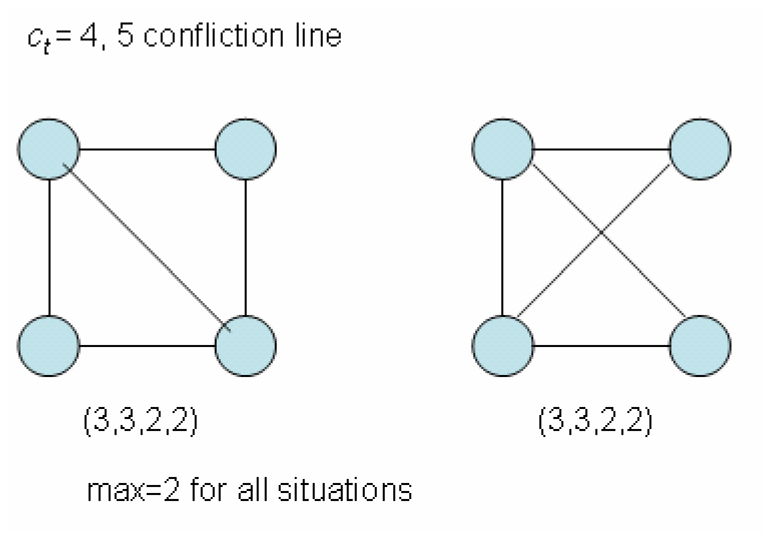
$c_t = 4$, 1 confliction line

max=2 for all other situations




c_t = 4, 3 confliction line
(3,1,1,1)
(2,2,2,0)
(2,2,1,1)
max = 3
max = 2 for all other situations


c_t = 4, 4 confliction line
(2,2,2,2)
(3,2,2,1)
max=2 for all situations


c_t = 4, 5 confliction line
(3,3,2,2)
(3,3,2,2)
max=2 for all situations

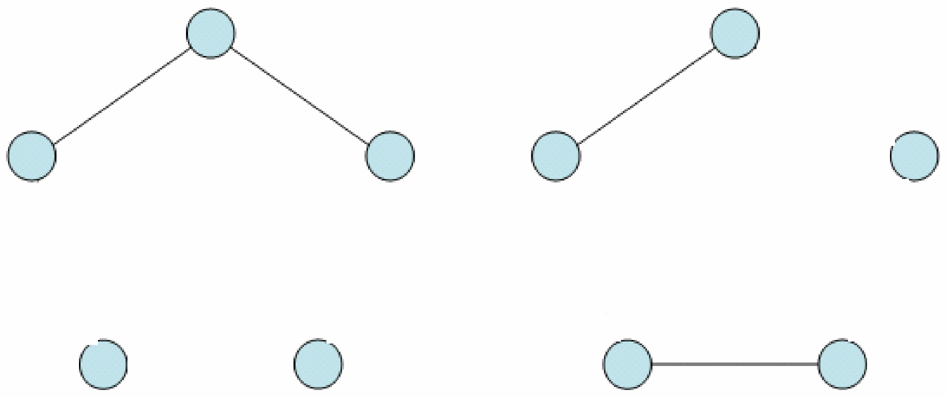
$c_t = 4$, 6 confliction line

max=1

$c_t = 5$, 1 confliction line

max=4

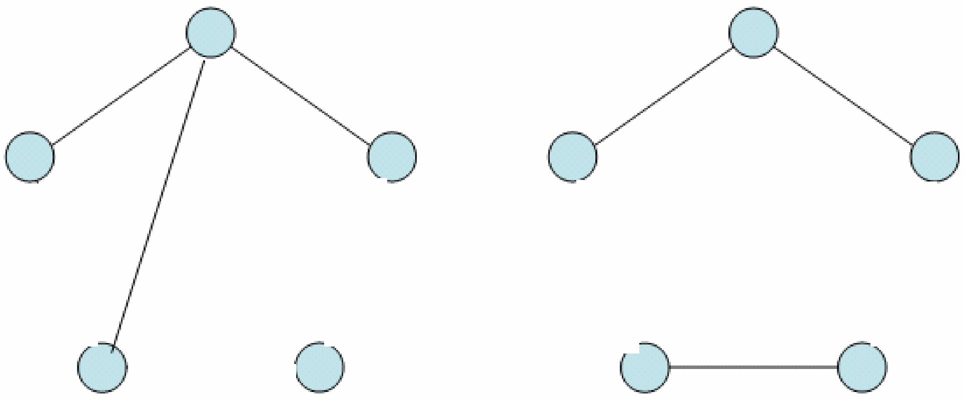
$c_t = 5$, 2 confliction line



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

max=4 max=3 for all other situations

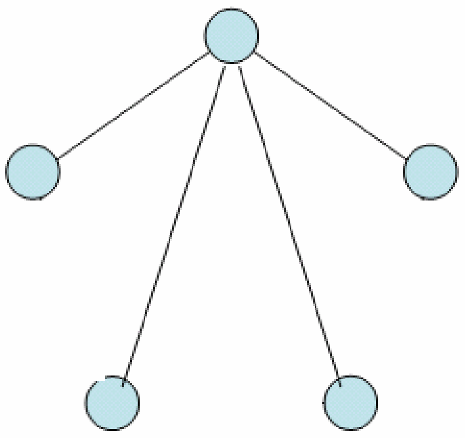
$c_t = 5$, 3 confliction line



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

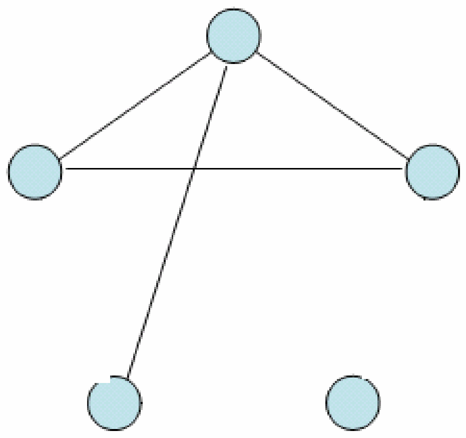
max=4 max=3 for all other situations

c_t= 5, 4 confliction line



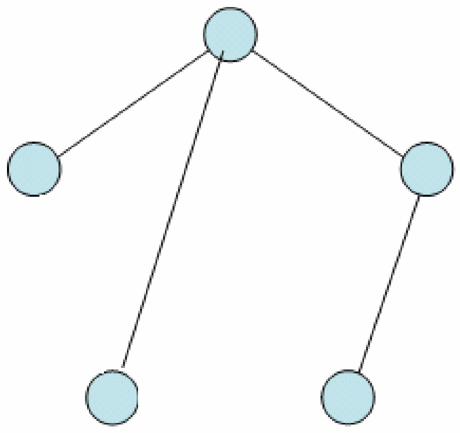
(4,1,1,1,1)

max=4



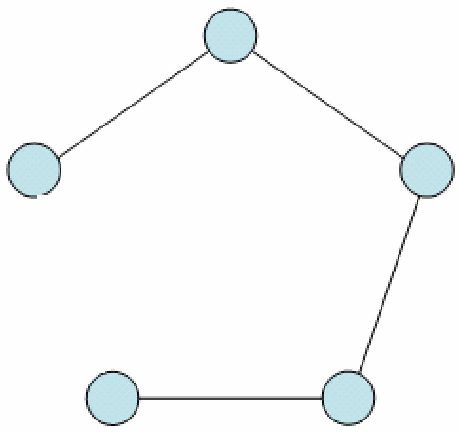
(3,2,2,1,0)

max=3



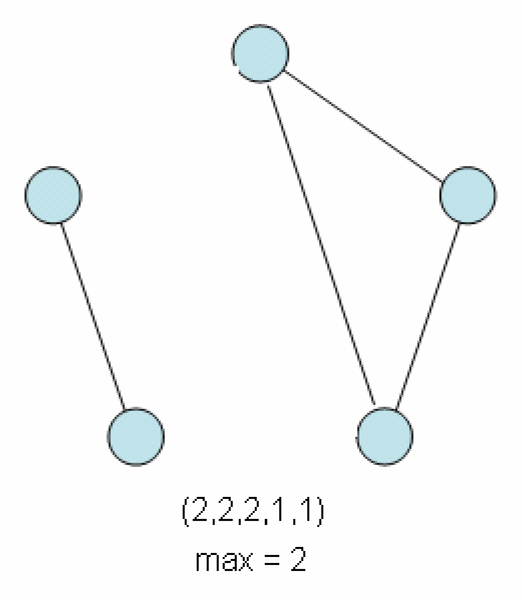
(3,2,1,1,1)

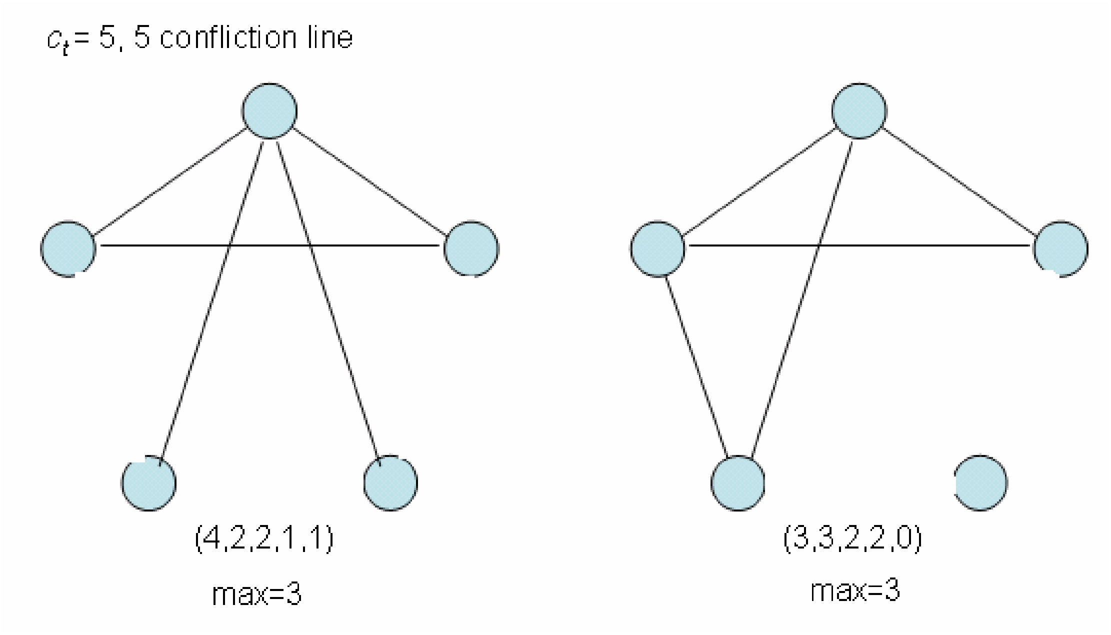
max=3

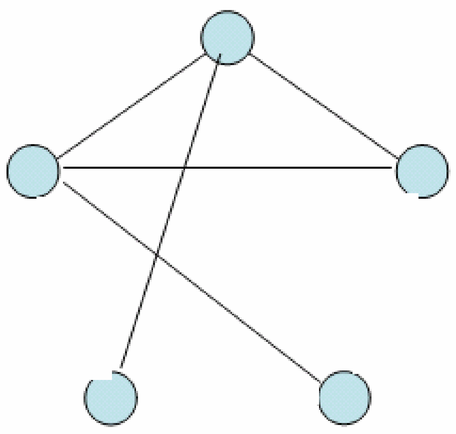


(2,2,2,1,1)

max=3

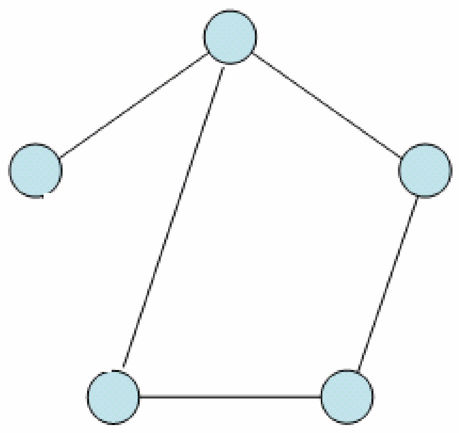

(2,2,2,1,1)
max = 2


c_t= 5, 5 confliction line
(4,2,2,1,1)
max=3
(3,3,2,2,0)
max=3



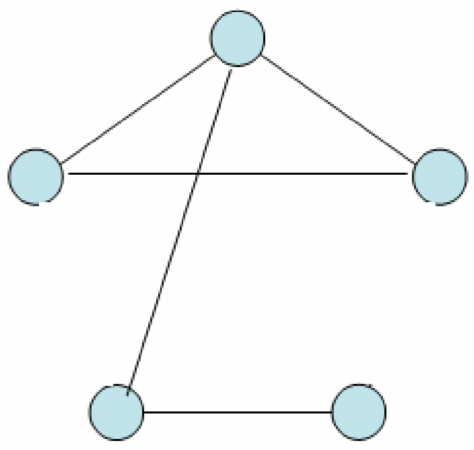
(3,3,2,1,1)

max=3



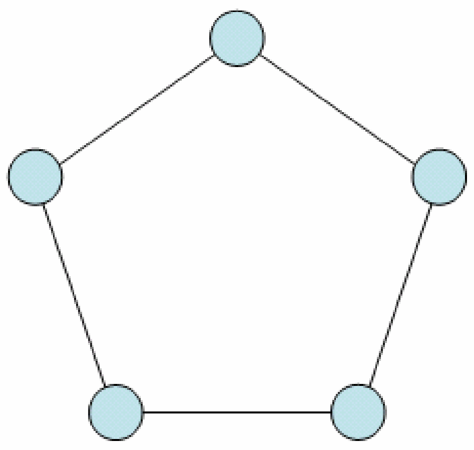
(3,2,2,2,1)

max=3



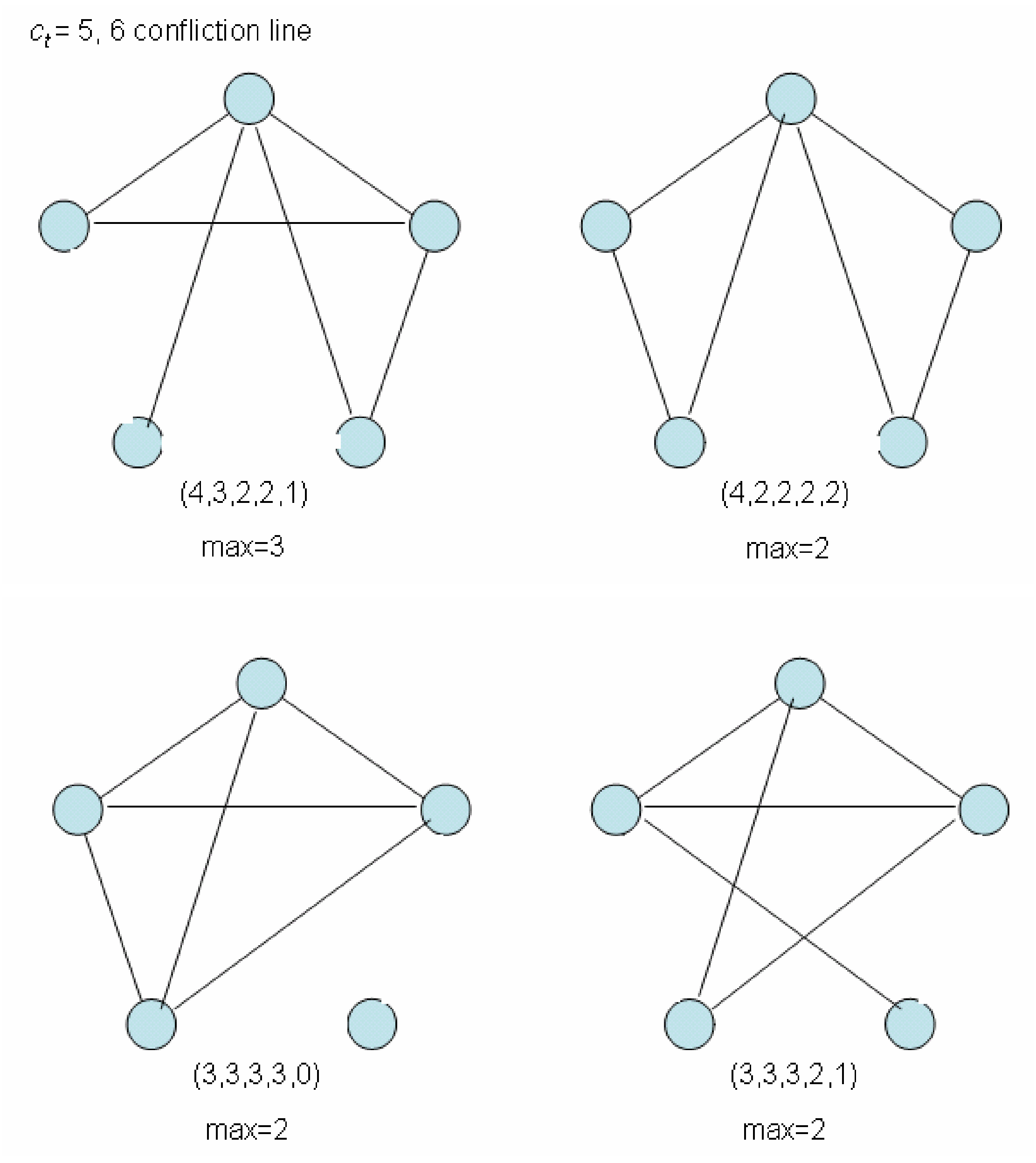
(3,2,2,2,1)

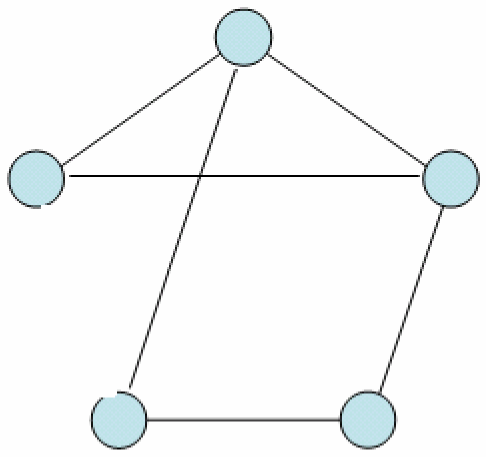
max=2



(2,2,2,2,2)

max=2


c_t= 5, 6 confliction line
(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