

Scheduling Commercial Videotapes in Broadcast Television

Srinivas Bollapragada

GE Global Research Center, 1 Research Circle, Schenectady, New York 12309, bollapragada@research.ge.com

Michael R. Bussieck

GAMS Development Corporation, 1217 Potomac Street NW, Washington, DC 20007, mbussieck@gams.com

Suman Mallik

Department of Business Administration, University of Illinois, 350 Wohlers Hall, 1206 South Sixth Street, Champaign, Illinois 61820, mallik@uiuc.edu

This paper, motivated by the experiences of a major U.S.-based broadcast television network, presents algorithms and heuristics to schedule commercial videotapes. Major advertisers purchase several slots to air commercials during a given time period on a broadcast network. We study the problem of scheduling advertiser's commercials in the slots it purchased when the same commercial is to be aired multiple times. Under such a situation, the advertisers typically want the airings of a commercial to be as evenly spaced as possible. Thus, our objective is to schedule a set of commercials in a set of available slots such that multiple airings of the same commercial are as evenly spaced as possible. A natural formulation of this problem is a mixed-integer program that can be solved using third-party solvers. We also develop a branch-and-bound algorithm based on a problem-specific bounding scheme. Both approaches fail to solve larger problem instances within a reasonable time frame. We present an alternative mixed-integer program that lends itself to an efficient solution. For solving even larger problems, we present multiple heuristics.

Subject classifications: scheduling, mixed-integer programming, heuristics, broadcast television.

Area of review: Optimization.

History: Received November 2002; revision received May 2003; accepted July 2003.

1. Introduction

The objective of this paper is to develop heuristics and algorithms for scheduling commercial videotapes in broadcast television. Our work is motivated by the problem faced by the National Broadcasting Company (NBC), one of the leading firms in the television industry. Major advertisers such as Procter & Gamble and General Motors buy hundreds of time slots to air commercials on a network during any broadcast season. The actual commercials to be aired in these slots are decided at a later stage. During the broadcast season the clients ship videotapes of the commercials to be aired in the slots that they had purchased. Each tape has a single commercial, which has a code written on it for identification. These codes are called Industry Standard Commercial Identification (ISCI) codes. The commercials are scheduled by their ISCI codes by the TV network personnel according to the instructions given by the advertiser. The advertisers most often specify the following guideline. Whenever a commercial is to be aired multiple times within a specified period (for example: a month), the advertiser wants these airings to be *evenly spaced as much as possible* over that time period. Thus, a client has a certain number of advertising slots that it had purchased during a specified time period. It also has a set of commercials to be scheduled in its time slots. The question naturally

arises: how to schedule the commercials in the available advertising slots such that two airings of the same commercial are as evenly spaced as possible. We propose to study this problem.

Stated in more formal terms, we have N balls (commercials), out of which n_1 are of color 1 (ISCI code 1), n_2 are of color 2 (ISCI code 2), and so on. We want to put the N balls into N slots such that balls of any one color are as evenly spaced as possible; i.e., the distance between subsequent balls of color c is as close as possible to N/n_c . This paper presents algorithms and heuristics for accomplishing this. We will call this problem to be the basic ISCI rotator problem or, simply, the ISCI problem. To motivate the discussion, we provide the following data from the said firm. Table 1a shows the individual commercials of an advertiser, with their ISCI codes and the number of times each commercial is to be aired. In this example, $N = 17$, the total number of slots, while $n_1 = n_5 = 3$, $n_2 = 5$, and $n_3 = n_4 = n_6 = 2$. Table 1b shows the 17 advertisement slots purchased by the advertiser between May 11 and May 27, 2000 on the Pax Network, which is partially owned by NBC. The column labeled Pod refers to a commercial break within a program. Thus, the first entry in the table denotes that the client purchased an advertising slot in the 5th commercial break (Pod 5) for *Touched by an Angel*

Table 1a. The individual commercials to be aired.

ISCI code	Airings
TOPS9016	3
MABH7503	5
TOPS9004	2
MABT6903	2
MAIT0206	3
MAGM0205	2

on May 11. The last column, labeled Schedule, represents an “optimal” solution.

The literature on scheduling problems in the television industry mainly deals with scheduling *programs* (rather than commercials) for television to optimize some specified criteria (the viewership ratings or a network’s share of audiences). Typical examples are Goodhardt et al. (1975), Headen et al. (1979), Henry and Rinnie (1984), Webster (1985), and Rust and Echambadi (1989). Reddy et al. (1998) describe strategies for optimal prime-time TV program scheduling. Strategies for scheduling advertisements (though not necessarily the television commercials) have also been studied in marketing literature. This literature is concerned with whether the advertising should be steady or pulsed (i.e., turned on and off), so that the effectiveness of the advertising is maximized. Some examples are Simon (1982) and Mahajan and Muller (1986). Lilien et al. (1992) provides a review of these models.

There is a vast array of operations research-based literature on scheduling. Lawler et al. (1993) provides a comprehensive review of this literature. The work that comes closest to our current work is the problem of obtaining optimal-level schedules for mixed model assembly lines in just-in-time (JIT) systems studied by Miltenburg (1989) and Kubiak and Sethi (1991). They consider C products with demands n_1, n_2, \dots, n_C to be produced during a specified

time horizon. Each product takes a unit of time to be produced, so that the specified time horizon is $N = \sum_{c=1}^C n_c$. Define $r_c = n_c/N$. The objective is to keep the proportion of cumulative production of product c to the total production as close to r_c as possible. Miltenburg (1989) proposes a quadratic integer programming formulation of the problem and several approximate solutions to it. Kubiak and Sethi (1991) show that the same problem can be transformed into an assignment problem and, hence, can be solved efficiently. We differ from these studies in the following way. In the ISCI problem there is no fixed demand schedule determined by r_c . Rather than minimizing the deviation from this fixed demand schedule, the placement of a ball in the ISCI problem is based on the deviation of the distance of *subsequent* balls of the same color to the ideal distance. Nevertheless, to compare our work with the literature, we describe a heuristic solution based on the work by Kubiak and Sethi (1991) in §4.

It is evident from the previous discussion that while scheduling programs and the effectiveness of various advertising policies are well studied, actual scheduling of commercials in the time slots purchased by a client has received very little attention in the literature. Recently, Bollapragada et al. (2002b) have developed a mathematical programming-based algorithm to rapidly generate near-optimal sales plans that meet advertiser requirements. A sales plan consists of a complete schedule of commercials to be aired for an advertiser to meet its requirements. The requirements include budget goals, audience demographics, and the mix of shows, commercial lengths, and the weeks that the client is interested in during the broadcast year. They implemented the sales planning and demand prediction algorithms in a system that is currently being used by NBC, generating more than \$50 million in additional revenues annually. They also introduced the ISCI problem and presented a simple heuristic. The objective of our work is to analyze the ISCI problem by using a formal optimization framework.

Table 1b. Slots purchased by an advertiser plus “optimal” schedule.

Show	Day of week	Air date	Air time	Pod	Schedule
Touched by an Angel	THU	5/11/2000	9:00:00 PM	5	TOPS9004
Touched by an Angel	FRI	5/12/2000	9:00:00 PM	2	MAIT0206
Twenty-One	SAT	5/13/2000	9:00:00 PM	2	MAGM0205
Pax Three-Hanky Movie	SAT	5/13/2000	10:00:00 PM	2	MABH7503
Christy	SUN	5/14/2000	5:00:00 PM	3	TOPS9016
Shop 'til You Drop	MON	5/15/2000	6:30:00 PM	4	MABT6903
Scarecrow & Mrs. King	TUE	5/16/2000	4:00:00 PM	2	MABH7503
Treasures in Your Home	TUE	5/16/2000	11:35:00 PM	3	MAIT0206
Dr Quinn Medicine Woman	WED	5/17/2000	3:00:00 PM	3	TOPS9004
It's a Miracle	THU	5/18/2000	8:00:00 PM	3	MABH7503
It's a Miracle	FRI	5/19/2000	8:00:00 PM	2	TOPS9016
Diagnosis Murder	FRI	5/19/2000	10:00:00 PM	3	MAGM0205
Diagnosis Murder	MON	5/22/2000	10:00:00 PM	5	MABH7503
Jack Hanna	TUE	5/23/2000	5:30:00 PM	2	MAIT0206
It's a Miracle	WED	5/24/2000	11:05:00 PM	3	MABT6903
Eight is Enough	SAT	5/27/2000	6:00:00 PM	2	MABH7503
D Jack Hanna	SAT	5/27/2000	8:00:00 PM	2	TOPS9016

Note that the focus of this paper is to provide efficient solutions for the scheduling problem discussed. In line with our observations in the industry, we treat the objective (i.e., to have airings of the same commercial evenly spaced) as given. Thus, we are not concerned with whether this is the best advertising strategy for the advertiser.

The remainder of this paper is organized as follows. Section 2 describes an intuitive approach based on a mixed-integer programming formulation. In §2.1, we present a branch-and-bound algorithm based on a problem-specific bounding scheme. In §3, we develop an alternative mixed-integer programming model for the ISCI problem. We explore several heuristic solutions for large ISCI instances in §4. All computational results are presented in §5, while the summary and the conclusions are presented in §6.

2. Model Formulation

Consider a set of N balls out of which n_1 are of color 1, n_2 are of color 2, and so on. We want to place these N balls into N slots such that the balls of any one color are as evenly spaced over the slots as possible. We define the following notation to formulate our first “intuitive” model.

2.1. Notation

c = index on color, $c = 1, 2, \dots, C$.

n_c = number of balls of color c .

N = total number of balls = $\sum_c n_c$, also equals the total number of slots.

i_c = index on balls of color c , $i_c = 1, 2, \dots, n_c \forall c$.

j, k = index on slots, $j, k = 1, 2, \dots, N$.

q_c = ideal distance between any two balls of color $c = N/n_c$.

Z_{i_c} = slot number of ball i of color c (decision variable).

$Y_{i_c,k} = \begin{cases} 1, & \text{if ball } i \text{ of color } c \text{ is assigned to slot } k, \\ 0, & \text{otherwise.} \end{cases}$

We formulate the basic ISCI rotator problem as an integer program with nonlinear objective function.

Problem P1

$$\text{Minimize } \sum_c \sum_{i_c} |Z_{i_c} - Z_{i_c-1} - q_c| \quad (1)$$

subject to

$$Z_{i_c-1} \leq Z_{i_c} - 1 \quad \forall i_c, c, \quad (2)$$

$$\sum_{i_c, k} Y_{i_c,k} = n_c \quad \forall c, \quad (3)$$

$$Z_{i_c} = \sum_k k Y_{i_c,k} \quad \forall i_c, c, \quad (4)$$

$$\sum_{i_c, c} Y_{i_c,k} = 1 \quad \forall k, \quad (5)$$

$$\sum_k Y_{i_c,k} = 1 \quad \forall i_c, c, \quad (6)$$

$$1 \leq Z_{i_c} \leq N, \quad Y_{i_c,k} \text{ binary.} \quad (7)$$

Note that we defined q_c to be the *ideal* distance between any two balls of color c so that the balls of color c are evenly spaced over the slots. This distance need not be an integer. However, the slot numbers (indexed by k) are always integers. Our objective is to have the spacing between any two balls of color c be as close to q_c as possible. The quantity $|Z_{i_c} - Z_{i_c-1} - q_c|$ is the deviation of the distance between the $(i-1)$ th and i th ball of color c from its ideal spacing. The objective function (1) in the above formulation, thus, is the sum of deviations from the ideal spacing for each ball of each color. Constraint (2) ensures an ordered arrangement of the balls. Constraint (3) ensures that all balls are used. Constraint (5) ensures that a slot can hold only one ball, while (6) ensures that each ball can be placed in only one slot. Note that the integrality of the variables Z_{i_c} is guaranteed through constraint (4), which describes the relationship between the Y and Z variables. The nonlinear objective of **P1** can be transformed to a linear problem using a standard technique described below.

Problem P2

$$\text{Minimize } \sum_c \sum_{i_c} (\pi_{i_c} + \nu_{i_c}) \quad (8)$$

subject to

constraints (2), (3), (4), (5), (6)

$$\pi_{i_c} - \nu_{i_c} = Z_{i_c} - Z_{i_c-1} - q_c \quad \forall i_c, c, \quad (9)$$

$$\pi_{i_c} \geq 0, \nu_{i_c} \geq 0 \quad \forall i, c. \quad (10)$$

In the above formulation we have introduced two additional sets of variables π and ν to accommodate the positive and negative parts of the distance calculation. Together with the right direction of π and ν in the objective function, we transformed the nonlinear $|\cdot|$ (absolute value) function into a linear representation. Corollary 1 follows directly from our model.

COROLLARY 1. *Problem P2 can be solved trivially when $n_c = n \forall c$, i.e., when we have the same number of balls of each color.*

The proof follows trivially. Consider an arrangement of balls in which the sequence of colors $1, 2, \dots, C$ are repeated n times. This is an optimal arrangement, as it gives rise to an objective function value of zero. In other words, when $n_c = n \forall c$, we can obtain the optimal arrangement of balls for an N -slot problem by solving a problem with N/n slots (containing one ball of each color) and repeating the arrangement n times. Under such a situation, we say that solving a problem of size N/n is equivalent to solving a problem of size N . However, a generalization of Corollary 1 is not possible. We state this result as Corollary 2.

COROLLARY 2. *If there is a common factor, α (with $\alpha > 1$), among the n_c s, then solving a problem of size N/α is not necessarily equivalent to solving a problem of size N .*

Table 2. Test problems.

ID	N	C	Color details	ID	N	C	Color details
1	8	2	5R, 3B	21	50	3	21R, 16B, 13W
2	8	3	4R, 2B, 2W	22	50	4	26R, 12B, 10W, 2G
3	10	3	5R, 3B, 2W	23	60	3	25R, 23B, 12W
4	11	3	6R, 3B, 2W	24	75	4	27R, 25B, 14W, 9G
5	12	3	6R, 4B, 2W	25	100	5	33R, 25B, 21W, 15G, 6Y
6	12	4	4R, 3B, 3W, 2G	26	100	4	39R, 35B, 17W, 9G
7	14	3	6R, 4B, 4W	27	150	5	31R, 31B, 30W, 29G, 29Y
8	15	3	6R, 5B, 4W	28	200	4	77R, 67B, 53W, 3G
9	16	4	8R, 3B, 3W, 2G	29	200	5	61R, 47B, 39W, 30G, 23Y
10	17	3	7R, 6B, 4W	30	250	5	91R, 63B, 54W, 31G, 11Y
11	20	3	8R, 7B, 5W	31	300	4	191R, 83B, 17W, 9G
12	20	4	8R, 7B, 3W, 2G	32	300	5	102R, 95B, 77W, 19G, 7Y
13	20	5	5R, 5B, 5W, 4G, 1Y	33	325	5	72R, 68B, 65W, 62G, 58Y
14	25	4	8R, 6B, 6W, 5G	34	350	5	99R, 81B, 63W, 54G, 53Y
15	25	5	7R, 6B, 5W, 4G, 3Y	35	400	4	135R, 102B, 92W, 71G
16	30	4	10R, 8B, 7W, 5G	36	400	5	229R, 149B, 11W, 8G, 3Y
17	30	5	9R, 7B, 6W, 5G, 3Y	37	425	5	126R, 107B, 86W, 73G, 33Y
18	40	4	17R, 10B, 8W, 5G	38	450	5	138R, 95B, 82W, 79G, 56Y
19	45	3	16R, 15B, 14W	39	500	4	170R, 150B, 90W, 90G
20	50	3	25R, 13B, 12W	40	500	5	143R, 112B, 111W, 70G, 64Y

Note. Key: R = red, B = blue, W = white, G = green, Y = yellow.

PROOF. The proof is by construction. Consider the following example: $N = 14$, with $n_1 = 6, n_2 = n_3 = 4$. Here, $\alpha = 2, q_1 = 2.33, q_2 = q_3 = 3.5$. We want to show that the optimal arrangement of balls for this problem cannot be obtained by solving a problem with $N = 7$, and $n_1 = 3, n_2 = n_3 = 2$. It can be verified (using any commercial MIP solver) that the optimal objective value for the original problem with $N = 14$ is 4.67. Also note that in any feasible solution to this problem, the contribution to the objective function of Equation (7) from the balls of colors 2 and 3 will be at least 1.5 each (because $q_2 = q_3 = 3.5$, any arrangement of a ball of color 2 or color 3 will contribute at least $0.5 \times (4 - 1) = 1.5$ to the objective). Thus, in an optimal arrangement, the contribution to the objective function from color 1 balls will be, at most, 1.67. Note that $q_1 = 2.33$. This means that contribution to the objective function from color 1 balls will be minimum when the spacing between the balls is 2. Thus, we can have only four possible arrangements of color 1 balls in the slots: (1, 3, 5, 7, 9, 11), (2, 4, 6, 8, 10, 12), (3, 5, 7, 9, 11, 13), and (4, 6, 8, 10, 12, 14), each of which will give a contribution of 1.67 to the objective function. Therefore, in an optimal solution, we must have one of these four arrangements for color 1 balls. Note, however, that none of these arrangements can be obtained by juxtaposing optimal solutions for the problem with $N = 7$ twice (as none of the four arrangements of color 1 balls are symmetric at halfway). Therefore, it is not possible to find the optimal arrangement of balls for the original problem of size 14 by solving a problem of size 7. \square

We implemented the mixed-integer programming (MIP) Model **P2** in GAMS (see Brooke et al. 1988) and solved with MIP solver GAMS/CPLEX (2002). The implementation details are discussed in §5. We used 40 test problems

with sizes ranging from 8 slots to 500 slots to test our model. The test problems are presented in Table 2 and are discussed at length in §5. Problems with 20–50 slots are typical in the broadcast industry. Such problems need to be solved in a few minutes, as schedulers use the algorithm in an interactive mode to schedule videotapes for each of the several hundred clients sequentially. Using GAMS/CPLEX on Model **P2**, we were only able to solve 13 test problems within one minute. Twenty-four out of the 40 test cases did not solve to optimality in 10 minutes of CPU time. We therefore explored alternative solution methodologies to obtain optimal or close-to-optimal solutions. In the next subsection, we present an algorithm that takes advantage of the structure of the problem to obtain the optimal solution.

2.2. A Modified Branch-and-Bound Algorithm

The linear programming relaxation of **P2** at the nodes of the branch-and-bound tree does not result in tight lower bounds. As a result, we developed a customized branch-and-bound algorithm that solves a logic-based formulation of the problem (Bollapragada et al. 2001 have shown that logic-based methods that branch directly on logical disjunctions can solve substantially larger problems than mixed-integer programming for some problems). We refer to this algorithm as the modified branch-and-bound algorithm.

This algorithm constructs the solution by placing one ball at a time, starting with the first slot. At the root node, we decide the color of the ball to be assigned to slot 1. Because there are C colors, there are at most C branches emanating from this node, one for each color ball. At level 2, we decide the ball to be placed in slot 2, given that slot 1 is already filled. Thus, the depth of the complete tree to be evaluated is N . Let $\mathbf{S} = (s_1, s_2, \dots, s_N)$ be the vector of

assignments of balls in the slots. Thus, $s_k = c$ implies that a ball of color c is placed in slot k ($s_k = 0$ implies that slot k is empty, with no balls assigned). Also, let $\mathbf{M} = (m_1, m_2, \dots, m_c)$ denote the vector of yet-unassigned balls. Thus, $m_c = \lambda$ implies that there are λ number of color c balls yet to be assigned to slots. At level i of the search tree, all s_k for which $k < i$ are fixed and a decision on s_i has to be made. Visiting all nodes in the tree is equivalent to evaluating all the feasible solutions. However, if we can compute a good lower bound on the solution at each node, the search space could be pruned. We use the following scheme to obtain the lower bounds.

2.2.1. Computing the Lower Bound. At level i of the search tree:

- Set current lower bound (LB) equal to the objective function contribution from the first $i - 1$ slots.

- For each color c :

Step 1. Place the remaining m_c balls in the remaining $N - i$ open slots to achieve spacing as close to q_c as possible between these balls (i.e., assume that all of the remaining $N - i$ slots are available to each color). Let OB_c be the contribution to the objective function from these m_c balls.

Step 2. Update lower bound by setting $LB = LB + OB_c$.

2.2.2. The Algorithm. Let $Obj(\mathbf{S})$ and $LB(\mathbf{S})$ denote, respectively, the objective function value and the lower bound (calculated using the procedure described in §2.1.1) for a vector of assignments \mathbf{S} . The modified branch-and-bound algorithm involves the following steps.

Step 1. Start with any feasible solution. This initial feasible solution could be obtained using any of the heuristics described in §4. Let B denote the current best objective value.

Step 2. Set $s_k = 0$, $k = 1, 2, \dots, N$.

Step 3. At any branch level k , $k = 1, 2, \dots, N - 1$:

(a) For any color c with $m_c > 0$, set $s_k = c$.

(b) If $LB(\mathbf{S}) < B$, then set $B = Obj(\mathbf{S})$. Set $k = k + 1$ (i.e., go to the next level of the tree). Continue. Otherwise, go to Step 3(a), repeat for the next color c with $m_c > 0$.

The modified branch-and-bound algorithm runs significantly faster than GAMS/CPLEX can solve Model **P2** at the cost of exchanging an “off-the-self” product with a custom-built branch-and-bound code. The number of instances solved to optimality increases from 16 to 20 when the greedy heuristic described in §4.1 is used to obtain an initial solution. (Refer to §5 for the full computational results.) However, the algorithm failed to solve the last 20 test problems within 10 minutes and does not produce near-optimal solutions. The run time of the algorithm may be improved by using one of the heuristics in §4 to obtain better upper bounds at some of the intermediate nodes. However, we developed an alternative and less intuitive integer programming model for the ISCI problem that lends itself to a more efficient solution. We call this model the flow formulation of the ISCI problem. We describe this model in the next section.

3. Flow Formulation of the ISCI Problem

We define the following notation in addition to our notation of §2:

$$p_{cj} = \begin{cases} 1 & \text{if ball of color } c \text{ is assigned to slot } j, \\ 0 & \text{otherwise,} \end{cases}$$

$$f_{cjk} = \begin{cases} 1 & \text{if ball of color } c \text{ is shipped} \\ & \text{from slot } j \text{ to slot } k, \\ 0 & \text{otherwise.} \end{cases}$$

Problem P3

$$\text{Minimize} \quad \sum_c \sum_{0 < j < k \leq N} |k - j - q_c| f_{cjk} \quad (11)$$

subject to

$$\sum_c p_{cj} = 1 \quad \forall j, \quad (12)$$

$$\sum_{1 \leq j \leq N} p_{cj} = n_c \quad \forall c, \quad (13)$$

$$\sum_{0 \leq k < j} f_{ckj} = \sum_{j < k \leq N} f_{cjk} + f_{c,j,0} \quad \forall c, j, \quad (14)$$

$$\sum_{0 < k \leq N} f_{c,0,k} = 1 \quad \forall c, \quad (15)$$

$$\sum_{j < k} f_{cjk} + f_{c,j,0} = p_{cj} \quad \forall c, j, \quad (16)$$

$$f, p \text{ binary.} \quad (17)$$

In the above formulation, for each color c we have a set of arcs as shown in Figure 1. Slot 0 represents an artificial source and sink in the network. Arcs go from slot j (including 0) to all other slots k , $j < k \leq N$, as well as back from j to 0. The network arcs $f(c, j, k)$ “ship” one ball of color c from slot 0 through a number of slots back to 0. At each slot, and for each color, the flow conservation constraint holds (Equation (14)). In addition, a flow-carrying arc $f(c, j, k)$ can leave position j if and only if position j is

Figure 1. The underlying graph of flow formulation **P3**.

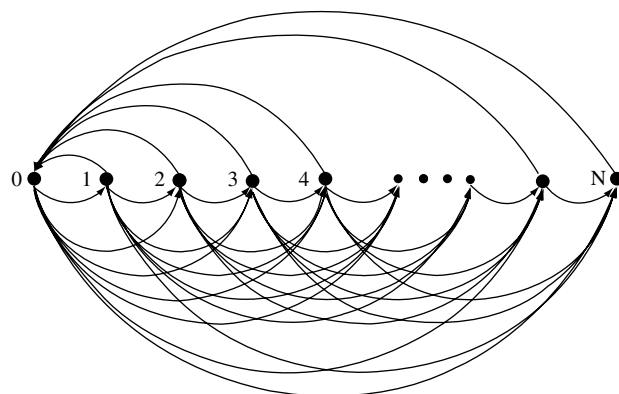


Table 3. Evolution and comparison of optimal solution strategies.

Solution method	Optimal solution	Integer solution	Comment
GAMS/CPLEX P2	Up to 25 slots	Up to 100 slots	Typical size of a broadcast television industry problem 20–50 slots
Modified B&B	Up to 50 slots	N/A	
GAMS/CPLEX P3	Up to 60 slots	Up to 200 slots	Solves larger problems meeting most real-world needs, forms basis for heuristic solutions

colored c (Equation (16)). This, coupled with (13), ensures that the chain of arcs carrying a ball of color c goes through exactly n_c slots. Equation (12) ensures that a slot is occupied by only one ball, while (15) ensures that only one ball of each color leaves the source 0. Traversing an arc $f(c, j, k)$ represents the placement of consecutive balls of color c in positions j and k ; hence, we have a cost of $|k - j - q_c|$ for using that arc. The objective in (11) is to find flows of minimum cost that obey the constraints of the problem.

The number of variables in Formulation **P3** is much larger than the number of variables in Formulation **P2**. The main advantage of this formulation, as the reader will see in §5, is that the relaxation provides a decent bound on the problem.

Note that the underlying graph of this formulation (Figure 1) is very dense, as we have arcs going out from every slot j to every other node k (with $k > j$) and back to slot 0, resulting in a model with a large number of variables. Arcs that are unlikely to carry flow may be removed (at the risk of losing the optimal solution for **P3**) to reduce the size of the model. Good candidates for removal are arcs jk with cost $|k - j - q_c| \geq \delta_c$. Choosing appropriate small values of δ that make the graph sparse but retain the optimal solution is the main challenge in this approach.

Using Formulation **P3**, we were able to solve more than half of the test problems to optimality within one minute using GAMS/CPLEX. We were also able to solve three more problems to optimality within 10 minutes of CPU time and found feasible solutions for five more problems. Table 3 describes the evolution of our optimal solution strategies for the ISCI problem and compares them. While our best optimal solution technique (solving **P3** using GAMS/CPLEX) was able to solve the problems we usually encountered at NBC, the gap between quickly solvable and unsolvable is extremely small (e.g., compare instances 23 and 24 in Table 4). Therefore, we explored several heuristic solutions described in the next section.

4. Heuristic Solutions for the ISCI Problem

We describe four heuristics for the ISCI problem in this section. The first heuristic is based on a simple greedy

search algorithm, while the second and third utilize the efficiency of the Flow Formulation **P3** described earlier. The fourth heuristic is motivated by the work of Kubiak and Sethi (1991). All computation results will be presented in §5.

4.1. Greedy Heuristic

Under a greedy heuristic we fill the slots sequentially. For each slot, we choose the ball color (from the pool of available balls of different colors) that gives the least contribution to the objective function defined by Equation (1). The specific steps are described below.

Step 1. Place a ball of color 1 (or of any other color) in slot 1.

Step 2. Update the number of balls left for the current color.

Step 3. Proceed to the next slot. Among the available balls, select the color that gives the minimum contribution to the objective function based on the arrangement thus far.

Step 4. Stop if this is the last slot, else go to Step 2.

4.2. P3 Delta Heuristic

The **P3** Delta Heuristic is directly derived from the Flow Model **P3** with a sparse underlying network. Arcs with high costs (i.e., $|k - j - q_c| \geq \delta_c = \max(20, 0.05 \cdot q_c)$) have been removed from the network.

4.3. P3 Batching Heuristic

We developed this heuristic with the large problems in mind. We divide a large problem into several smaller subproblems (or batches) so that each batch can be solved efficiently by using Formulation **P3**. The specific steps are described below.

Step 1. Divide a problem into several batches whenever $N > 60$. Therefore, a problem with $N \leq 60$ is solved in one batch, while a problem with $60 < N \leq 120$ is solved in two batches, and so on. For example, our test problem 36 consists of 400 balls with 229 red, 149 blue, 11 white, 8 green, and 3 yellow balls. This problem will be divided into seven batches.

Step 2. Distribute the balls of each color evenly over the batches. For the example discussed above, the first

Table 4. Computational results.

Id	GAMS/CPLEX P2						Modified B&B algorithm		GAMS/CPLEX P3		Greedy heuristic		P3/Delta heuristic		P3/Batching heuristic		Assignment heuristic		Best choice	
	Obj		Bound		CPU		Obj	CPU	Obj	Bound	Obj	CPU	Obj	CPU	Obj	CPU	Obj	CPU	Obj	Bound
	Obj	Bound	Obj	Bound	CPU	CPU														
1	2.667	2.667	0	0	0	2	2.667	0	2.667	2.667	2.667	0	2.667	1	2.667	0	2.800	1	2.667	2.667
2	0.000	0.000	0	0	0	1	0.000	0	0.000	0.000	0.000	1	0.000	1	0.000	1	0.000	0	0.000	0.000
3	2.333	2.333	0	2.333	2.333	1	2.333	0	2.333	2.333	2.333	0	2.333	1	2.333	1	2.333	1	2.330	2.333
4	3.000	3.000	0	3.000	3.000	0	3.000	0	3.000	3.000	3.000	0	3.000	1	3.000	1	4.000	1	3.000	3.000
5	3.000	3.000	0	3.000	3.000	0	3.000	0	3.000	3.000	3.000	0	3.000	1	3.000	1	4.000	1	3.000	3.000
6	2.000	2.000	0	2.000	2.000	1	2.000	0	2.000	2.000	2.000	0	2.000	0	2.000	1	3.000	1	2.000	2.000
7	4.667	4.667	3	4.667	4.667	1	4.667	0	4.667	4.667	4.667	0	4.667	0	4.667	0	5.000	1	4.667	4.667
8	4.250	4.250	3	4.250	4.250	1	4.250	0	4.250	4.250	4.250	0	4.250	0	4.250	0	5.750	1	4.250	4.250
9	3.333	3.333	9	3.333	3.333	1	3.333	0	3.333	3.333	3.333	0	3.333	1	3.333	0	4.333	1	3.330	3.333
10	5.393	5.393	6	5.393	5.393	1	5.393	0	5.393	5.393	5.393	0	5.393	1	5.393	1	7.607	1	5.393	5.393
11	5.786	5.786	15	5.786	5.786	1	5.786	0	5.786	5.786	5.786	0	5.786	1	5.786	1	9.071	1	5.786	5.786
12	7.167	7.167	147	7.167	7.167	1	7.167	0	7.167	7.167	7.167	0	7.167	2	7.167	2	9.500	1	7.167	7.167
13	3.000	3.000	81	3.000	3.000	2	3.000	0	3.000	3.000	3.000	0	3.000	2	3.000	2	5.000	1	3.000	3.000
14	6.792	6.792	576	6.792	6.792	3	6.792	0	6.792	6.792	6.792	0	6.792	3	6.792	2	11.458	1	6.792	6.792
15	8.464	8.464	600	8.464	8.464	7	8.464	0	8.464	8.464	8.464	0	8.464	9	8.464	9	11.250	1	8.464	8.464
16	9.250	9.250	600	9.250	9.250	3	9.250	0	9.250	9.250	9.250	0	9.250	6	9.250	6	14.250	1	9.250	9.250
17	11.238	5.017	600	10.095	10.095	37	10.095	0	10.095	10.095	10.095	0	10.095	29	10.095	30	16.571	1	10.095	10.095
18	24.706	5.647	600	13.706	13.706	56	13.706	0	13.706	13.706	13.706	0	13.706	54	13.706	54	21.000	1	13.706	13.706
19	6.223	6.223	6	6.223	6.223	1	6.223	0	6.223	6.223	6.223	0	6.223	1	6.223	1	11.616	2	6.223	6.223
20	3.680	3.680	0	3.680	3.680	1	3.680	0	3.680	3.680	3.680	0	3.680	1	3.680	2	29.039	1	3.680	3.680
21	23.036	12.032	600	19.495	19.495	47	19.495	0	19.495	19.495	19.495	0	19.495	30	19.495	28	24.502	1	19.495	19.495
22	22.962	5.064	600	14.449	14.449	104	14.449	0	14.449	14.449	14.449	0	14.449	78	14.449	88	19.680	1	14.449	14.449
23	27.583	19.730	600	22.800	22.800	8	22.800	0	22.800	22.800	22.800	0	22.800	4	22.800	5	36.365	1	22.800	22.800
24	60.135	15.310	600	34.643	23.117	600	34.643	0	34.643	23.117	23.117	0	34.643	600	33.818	35	44.246	1	33.818	23.117
25	136.043	15.089	600	54.900	15.434	600	54.900	0	54.900	15.434	15.434	0	54.900	600	46.108	542	65.459	2	46.108	15.434
26	—	28.643	600	38.808	32.444	600	38.808	0	38.808	32.444	32.444	0	38.808	600	39.574	17	65.500	3	38.459	32.444
27	—	22.042	600	22.042	22.042	14	22.042	0	22.042	22.042	22.042	0	22.042	6	24.687	9	50.681	3	22.042	22.042
28	—	55.923	600	96.549	55.923	600	96.549	0	96.549	55.923	55.923	0	96.549	600	115.067	343	100.591	4	81.698	55.923
29	—	51.367	600	188.977	52.934	600	188.977	0	188.977	52.934	52.934	0	188.977	600	121.258	600	161.400	6	121.258	52.934
30	—	65.072	600	216.205	65.263	600	216.205	0	216.205	65.263	65.263	0	216.205	600	130.014	600	168.819	7	130.014	65.263
31	—	139.870	600	182.297	139.870	600	182.297	0	182.297	139.870	139.870	0	182.297	600	173.262	31	177.078	5	145.730	139.870
32	—	42.401	600	441.575	43.026	600	441.575	0	441.575	43.026	43.026	0	441.575	600	133.196	600	164.845	7	133.196	43.026
33	—	97.968	600	518.118	98.023	600	518.118	0	518.118	98.023	98.023	0	518.118	600	134.621	527	167.570	11	134.621	98.023
34	—	112.270	600	—	154.226	600	—	0	—	154.226	154.226	0	—	600	177.701	600	298.494	11	177.701	154.226
35	—	—	600	572.650	85.653	600	572.650	0	572.650	85.653	85.653	0	572.650	600	174.723	600	270.560	10	174.723	85.653
36	—	100.488	600	227.274	154.210	600	227.274	0	227.274	154.210	154.210	0	227.274	600	216.641	132	211.637	7	207.017	154.210
37	—	—	600	934.476	83.321	600	934.476	0	934.476	83.321	83.321	0	934.476	600	218.371	600	306.397	12	218.371	83.321
38	—	—	600	1042.130	143.889	600	1042.130	0	1042.130	143.889	143.889	0	1042.130	600	252.888	600	341.895	15	252.888	143.889
39	—	—	600	997.758	154.440	600	997.758	0	997.758	154.440	154.440	0	997.758	600	192.706	468	242.523	11	192.706	154.440
40	—	—	600	—	202.233	600	—	0	—	202.233	202.233	0	—	600	280.748	600	393.945	20	280.748	202.233

five batches will have 33 red balls each and the remaining batches will have 32 red balls each. Similarly, the first two batches will have 22 blue balls each and the remaining batches will have 21 blue balls each. The first batch will have two green balls, while the remaining batches will have only one. Only batches 2, 4, and 7 will have one yellow ball each. Finally, batches 1, 2, 4, and 6 will have two white balls each, while batches 3, 5, and 7 will have one white ball each. A simple GAMS program accomplishes this distribution.

Step 3. Solve Model **P3** using a sparse network with

$\delta_c = \max(20, 0.2 \cdot q_c)$ in the following steps:

(a) Solve **P3** for the first 60 slots using the first batch of balls only.

(b) Solve Model **P3** for the first 120 slots using balls from batch 2 and the solution for the first 60 slots obtained from Step 3a. Treat the solution for the first 60 slots as fixed (unchangeable).

(c) Next, solve Model **P3** for the first 180 slots and treat the solution for the first 120 slots as fixed. Repeat until we have covered all slots.

The philosophy of this heuristic is similar to that of divide and conquer. The choice of the batch size 60 is driven by the trade-off of avoiding many batches and being able to solve each batch efficiently.

4.4. Assignment Heuristic

Miltenberg (1989) and Kubiak and Sethi (1991) consider the problem of obtaining optimal-level schedules for mixed model assembly lines in JIT systems. We have defined their problem in §1. Letting x_{ck} denote the total cumulative production of product c in periods 1 through k , their problem can be formulated as the following integer program:

$$\left\{ \max \sum_{k=1}^N \sum_{c=1}^C (x_{ck} - kr_c)^2 \mid \sum_{c=1}^C x_{ck} = k \forall k; \right. \\ \left. 0 \leq x_{ck} - x_{c,k-1} \leq 1 \forall c; x_{ck} \geq 0, \text{ integer} \right\}. \quad (18)$$

Kubiak and Sethi (1991) show that the above problem can be transformed into an assignment problem with decision variable x_{jk}^c (binary variable; equals 1 when the j th unit of product c is produced in the k th period, and is zero otherwise). The cost γ_{jk}^c (for assigning the j th unit of product c to the k th period) depends on the deviation of position k from the *perfect position* $Z_j^c = \lceil (2j-1)/2r_c \rceil$ of the j th unit of product c . The solution of the assignment problem

$$\left\{ \min \sum_{j,k,c} \gamma_{jk}^c x_{jk}^c \mid \text{st. } \sum_{j,c} x_{jk}^c = 1 \forall k; \right. \\ \left. \sum_k x_{jk}^c = 1 \forall j, c, x_{jk}^c \geq 0 \right\} \quad (19)$$

can be easily transformed into a solution of the original problem (18). For details, see Kubiak and Sethi (1991).

The key for using this assignment approach for the ISCI problem is to find a strong relationship between the

perfect positions and the *ideal distance*. Consider the following trivial example, with nine slots and three colors $n_1 = n_2 = n_3 = 3$. The perfect positions Z_j^c are 2, 5, and 8 for $j = 1$ to 3 and all colors. Because Z_j^c are the same for all colors c , the costs γ_{jk}^c are the same, and therefore the set of optimal solutions of (19) includes solutions that are not optimal with respect to the ISCI objective. For example, the sequence $c_1 c_2 c_3 c_3 c_2 c_1 c_1 c_3 c_2$ has the same cost in (19) as $c_1 c_2 c_3 c_1 c_2 c_3 c_1 c_2 c_3$, which is clearly favored under the ISCI objective. The problem is that there are multiple perfect positions for an ideal distance. In our example, the positions (1, 4, 7) and (2, 5, 8), as well as (3, 6, 9), are *perfect* with respect to the ideal distance $9/3 = 3$. If we would change the perfect positions to $Z^{c_1} = (1, 4, 7)$, $Z^{c_2} = (2, 5, 8)$, and $Z^{c_3} = (3, 6, 9)$, together with the cost γ_{jk}^c , Model (19) would produce the optimum sequence $c_1 c_2 c_3 c_1 c_2 c_3 c_1 c_2 c_3$. In general, we try to find for each color one of the shifted perfect positions $P^c(\varepsilon_c) = \{Z_j^c + \varepsilon_c \mid j = 1 \dots n_c\}$, with $\varepsilon_c \in \{-\lceil N/2n_c \rceil + 1, \dots, \lceil (2n_c - 1)N/2n_c \rceil\} = I_c$, that make the optimal solution of (19) *likely* to be a good solution for the ISCI problem. Our suggestion is to find positions $P^c(\varepsilon_c)$ such that the number of multiple perfect positions is minimized. More formally, we try to find

$$\varepsilon^* = \arg \min_{\varepsilon} \left| \{1, \dots, N\} \setminus \left\{ \bigcup_c P^c(\varepsilon_c) \right\} \right|.$$

Finding such an ε^* vector is in general difficult, but can be “easily” found for relevant ISCI instances by the following integer program, which is closely related to a minimum cover model. The binary variable $u_{\varepsilon_c}^c$ takes value 1 if the shifted perfect positions $P^c(\varepsilon_c)$ are used, 0 otherwise. Hence, for each slot k we can aggregate the contributing positions for all colors. Because we want to count (and minimize) the multiple used slots, we need a variable o_k that collects the overflow (22). Furthermore, we need to select one set of perfect positions for each color (21). The following set of equations gives the details:

$$\text{Minimize } \sum_k o_k \quad (20)$$

subject to

$$\sum_{\varepsilon_c \in I_c} u_{\varepsilon_c}^c = 1 \quad \forall c, \quad (21)$$

$$\sum_c \sum_{Z_j^c + \varepsilon_c = k, \varepsilon_c \in I_c} u_{\varepsilon_c}^c = 1 + o_k \quad \forall k, \quad (22)$$

$$o_k \geq 0, \quad u_{\varepsilon_c}^c \text{ binary.} \quad (23)$$

After finding ε^* , we define the cost

$$\gamma_{jk}^c = (P^c(\varepsilon_c^*) - k)^2 \quad (24)$$

to be the quadratic deviation from the perfect position, and solve (19). The resulting solution can be evaluated in terms of the ISCI objective.

5. Computational Results

We used 40 test problems to test the effectiveness of the heuristics and the optimal algorithms. A test problem with 20–50 slots is indicative of the usual size of the problem faced by the broadcast television industry. Test problems 26–40 are considered “large.” We constructed these test problems to check the range of effectiveness of our algorithm and heuristics. All computations were run on a personal computer with an Intel Pentium IV processor at 1.6 GHz. Table 2 describes our test problems.

We have specified the total number of slots (N) and the number of colors (C) against each problem. The column labeled Color Details in Table 2 describes the numbers of balls of each color. Thus, Problem 1 has a total of eight balls of two colors: five red balls and three blue balls. Computations for a test problem were terminated after 10 minutes (600 seconds) of CPU time usage. Table 4 has, for each test instance, the computational results for all models and algorithms described so far. Each section of the table includes the column Obj for objective value and CPU for CPU time usage in seconds. For GAMS/CPLEX **P2** and GAMS/CPLEX **P3**, we also list the best-known linear-programming-based lower bound from the CPLEX branch-and-cut algorithm. If a model/algorithm produced the best solution among all algorithms, numbers in the “Obj” column of the corresponding section are printed in bold; i.e., the more sets of bold numbers, the better the method. The last section, titled Best Choice, takes the minimum objective value over all sections and the maximum bound of GAMS/CPLEX **P2** and GAMS/CPLEX **P3** and also lists the relative optimality gap, i.e., (best objective—best bound)/best bound.

The development of efficient optimal and near-optimal solution methods for the ISCI problem has been the focus of this paper. Computational results are the essential tools for providing evidence for the relevance of our analysis. Most important is the reproducibility of our experiments. Therefore, we made the GAMS source code for all models available at www.gams.com/apps/isci.

The remainder of this section is organized into two subsections. Section 5.1 describes the computational results for the optimal solution approaches, i.e., Model **P2**, the modified branch-and-bound algorithm, and Flow Model **P3**. Section 5.2 describes the computations for the four heuristics.

5.1. Computations for the Optimal Solution Approaches

Formulation **P2** was modeled using GAMS, and the test problems were solved using GAMS/CPLEX solver (CPLEX version 8.0). We were able to obtain optimal solutions for Problems 1–14, 19, and 20 within 600 seconds of CPU time. In addition, a feasible integer solution was obtained for Problems 15–18 and 21–25, while no feasible solution was found for the rest of the problems. Along

with the size, the computation time is also influenced by the structure of the problem. For example, we were able to get optimal solutions for Problem 19 (45 slots), while we were unable to get optimal solutions for Problems 15–18, which are smaller. This is due to the special structure of Problem 19, where we have an almost equal number of balls of each color.

The modified branch-and-bound algorithm has a “feast-or-famine” behavior. We were able to obtain optimal solutions for the first 20 test problems using the algorithm within the allowable time. We could solve up to a maximum of a 50-slot problem within the said time. The solution times are significantly improved compared to Model **P2**. Unfortunately, it did not improve upon the starting feasible solution within the allowable time for instances higher than 20.

With Formulation **P3** we were able to obtain optimal solutions for Problems 1–23 (up to 60 slots), and Problem 27, within 600 seconds of CPU time. Feasible integer solutions were obtained for all instances except 34 and 40. Although the gap between objective value and best bound is significantly large, this represents a substantial improvement over Model **P2** and the modified branch-and-bound algorithm. This formulation is able to solve any typical broadcast television problem.

5.2. Computations for the Heuristics

The greedy heuristic performs reasonably well for smaller test problems (up to Problem 11). The elegance of this heuristic lies in its simplicity and easy computability. However, the deviation from the best-known solution increases, in general, as the size of the problem increases. Nevertheless, the solutions produced by the greedy algorithm can help to “hot-start” the CPLEX branch-and-cut algorithm for **P2** and **P3** as well as the modified branch-and-bound algorithm.

The removal of arcs unlikely to carry flow resulted in the **P3/Delta** Heuristic. δ_c has been selected in a way that the underlying graph had significantly fewer arcs but maintained the paths representing the optimal solution. For all but two instances the **P3/Delta** Heuristic found a feasible solution and provided in 28 of the 40 cases the best-known solution.

Recall that in the **P3** Batching Heuristic we are solving large problems in batches of 60 slots. Therefore, we are solving problems with less than 60 slots in one batch. This explains why we are getting the optimal solutions for Problems 1–23. By comparing GAMS/CPLEX **P3** and **P3/Batching** Heuristic, the reader will observe that the latter runs somewhat faster for these test problems. This is because we are using a sparse network to **P3/Batching** Heuristic. Overall, the **P3** Batching Heuristics perform very well, which is also expressed in the largest number of shaded table entries. Furthermore, this heuristic can be modified to speed up the overall process by reducing the batch size from 60 to a smaller number, which results in

faster solution times for the **P3** submodels, but also potentially worse solutions.

We next take a look at the Assignment Heuristic. The running times for the Assignment Heuristic are almost as good as for the Greedy Heuristic. However, the Greedy Heuristic outperforms the Assignment Heuristic based on speed and quality (always faster, better in 26, worse in 10 of the 40 cases). We also experimented with an absolute-value form of the cost function in (24). However, that results in worse performance of the heuristic compared to the squared form of cost function. Nevertheless, the Assignment Heuristic represents a first step in casting the ISCI problem into an assignment framework, which may result in the development of better optimal and near-optimal algorithms.

The last section, Best Choice in Table 4 summarizes our efforts to solve solving the ISCI problem to optimality. For 24 out of the 40 models, the solution of Model **P3** provided the optimum solution. For the remaining cases, the relative gap between best-known solution and best bound varies between 4% and 210%. It appears that the structure of the problem, in addition to its size, plays a key role in determining the accuracy of the heuristics as well as the quality of the lower bound.

6. Summary and Conclusion

In this paper, we developed efficient solution methods to schedule commercial videotapes in broadcast television. Our objective was to make the airings of the same commercial as evenly spaced as possible over a specified time period. We first modeled this problem as a mixed-integer program that minimizes the sum of deviations from the ideal spacing of the commercials. We also developed a branch-and-bound-like algorithm that uses the structure of the problem to obtain the optimal solution. Both models/algorithms did not solve problems of practical size in the set time limit. We then presented an innovative formulation that was able to improve upon the performance. We also described four heuristics for quickly solving the ISCI problem. Our choice of the objective function in Equation (1) is consistent with our observation in the industry, where the deviation from the ideal schedule is measured in absolute value. The flow formulation algorithm and all of our heuristics will remain valid for an alternative objective function involving the minimization of squared deviation from the ideal schedule. A variation of one of the heuristic algorithms described in §5 was implemented in a scheduling system for placing commercial videotapes at the Pax Network. The algorithm resulted in significant improvement in scheduling productivity.

Two extensions of the original ISCI problem are of practical importance. An advertiser often wants a specific slot for a specific commercial. For example, the client described in §1 might have wanted the commercial with ISCI code TOPS9016 to be aired at 9:00 pm on May 12. Television

networks typically accommodate these requests, while still maintaining the overall objective of placing the commercials of an ISCI code as evenly spaced as possible. Given that the exact identity of the desired slot is known, this situation can readily be accommodated in our modeling framework by assuming that some slots out of N possible slots contain preplaced balls. All of our models and algorithms handled this extension rather well by simply accounting for the preplaced balls. The second extension that we considered is called the ISCI problem with equal time intervals, where an advertiser wanted the commercials with the same ISCI code to be as *evenly spaced in time* as possible during the time period under consideration. This problem is similar in structure to that of the original ISCI problem. All algorithms described in this paper can be used for the ISCI problem with equal time intervals with minor modifications and yielding similar results. A detailed discussion including computational results for these extensions would exceed the scope of this paper. We refer the reader to the ISCI model web page for details.

It is interesting to note that while we developed the models and the algorithms presented here with one specific problem in mind, similar situations arise in other business scenarios. One such example is the problem of designing sales catalogs. While designing these catalogs, the catalog merchants often want the similar products to be spread out as much as possible through the catalog. Considering the sales catalog to be a collection of slots to hold descriptions for different types of products, this problem can be transformed into the models described in this paper.

Our future research will address some extensions to this problem. One such extension is to have different classes of balls. In our current work, we are minimizing the sum of deviations from the ideal spacing between any two balls of the same color. In a multiple-class scenario, it is more important to have even spacing for balls of one color over balls of another color. Thus, minimizing the weighted sum of deviations from the ideal spacing will be an appropriate objective.

Acknowledgments

The authors thank Hans Kellerer from Graz University, who pointed out the similarity of the ISCI problem and the problem studied by Miltenburg (1989) and Kubiak and Sethi (1991). The first author thanks the National Broadcasting Company for introducing the problem and funding the initial work. The last author acknowledges the support of the Campus Research Board, University of Illinois at Urbana-Champaign, through grant # 1-2-68042.

References

- Bollapragada, S., O. Ghattas, J. Hooker. 2001. Optimal design of truss structures by logic-based branch and cut. *Oper. Res.* **49**(1) 42–51.
- Bollapragada, S., M. R. Bussieck, S. Mallik. 2002a. The ISCI model web page: www.gams.com/apps/isci.

- Bollapragada, S., H. Cheng, M. Phillips, M. Scholes, T. Gibbs, M. Humphreville. 2002b. NBC's optimization systems increase its revenues and productivity. *Interfaces* **32**(1) 47–60.
- Brooke, A., D. Kendrick, A. Meeraus. 1988. *GAMS: A Users Guide*. The Scientific Press, Redwood City, CA.
- GAMS/CPLEX. 2002. *GAMS—The Solver Manuals*. GAMS Development Corporation, Washington, DC.
- Goodhardt, G. J., A. S. C. Ehrenberg, M. A. Collins. 1975. *The Television Audience: Patterns of Viewing*. Saxon House, Westmead, U.K.
- Headen, R. S., J. E. Klompmaker, R. T. Rust. 1979. The duplication of viewing law and television media scheduling evaluation. *J. Marketing Res.* **16** 333–340.
- Henry, M. D., H. J. Rinne. 1984. Predicting program shares in new time slots. *J. Advertising Res.* **24**(2) 9–17.
- Kubiak, W., S. Sethi. 1991. A note on “level schedules for mixed-model assembly lines in just-in-time systems.” *Management Sci.* **37**(1) 121–122.
- Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys. 1993. Sequencing and scheduling: algorithms and complexity. S. C. Graves et al., eds. *Handbooks in OR & MS*, Vol. 4. North-Holland, Amsterdam, The Netherlands.
- Lilien, G. L., P. Kotler, K. S. Moorthy. 1992. *Marketing Models*. Prentice Hall, Englewood Cliffs, NJ.
- Mahajan, V., E. Muller. 1986. Advertising pulsing policies for generating awareness for new products. *Marketing Sci.* **5**(2) 86–106.
- Miltenberg, J. 1989. Level schedules for mixed-model assembly lines in just-in-time systems. *Management Sci.* **35**(2) 192–207.
- Reddy, S. K., J. E. Aronson, A. Stam. 1998. SPOT: Scheduling programs optimally for television. *Management Sci.* **44**(1) 83–102.
- Rust, R. T., N. V. Echambadi. 1989. Scheduling network television programs: A heuristic audience flow approach to maximizing audience share. *J. Advertising* **18**(2) 11–18.
- Simon, H. 1982. ADPLUS: An advertising model with wear out and pulsation. *J. Marketing Res.* **19** 352–363.
- Webster, J. G. 1985. Program audience duplication: A study of television inheritance effects. *J. Broadcasting Electronic Media* **29**(2) 121–133.