JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 79, No. 3, DECEMBER 1993

Optimal Scheduling in Film Production to Minimize Talent Hold Cost[1]

T. C. E. CHENG,[2] J. E. DIAMOND,[3] AND B. M. T. LIN[4]

Communicated by W. Stadler

Abstract. The problem of minimizing the cost due to talent hold days in the production of a feature film is considered. A combinatorial model is developed for the sequencing of shooting days in a film shoot. The problem is shown to be strongly NP-hard. A branch-and-bound solution algorithm and a heuristic solution method for large instances of the problem (15 shooting days or more) are developed and implemented on a computer. A number of randomly generated problem instances are solved to study the power and speed of the algorithms. The computational results reveal that the heuristic solution method is effective and efficient in obtaining near-optimal solutions.

Key Words. Scheduling, branch-and-bound algorithms, NP-hardness.

1. Introduction

In the production of a feature film, the various components or scenes of the film are not generally filmed in the same sequence in which they appear in the final version. The sequence in which the scenes are shot is determined by the first assistant director, who considers various economic, logistic, and artistic factors in deciding upon a particular sequence.

[1]This research was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant OPG-0036424. The authors are thankful to two anonymous referees for their helpful comments on an earlier version of this paper.

[2]Professor and Head, Department of Management, Hong Kong Polytechnic, Kowloon, Hong Kong.

[3]Doctoral Student, Department of Mechanical and Industrial Engineering, University of Manitoba, Winnipeg, Manitoba, Canada.

[4]Lecturer, Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.

In shooting a film, especially when it is shot on location, many of the actors or talent required for the film must travel great distances to the filming location. Any person who travels from his residence to the filming location must be paid for each day in which he is present at the location regardless of whether he is required in the scenes which are being shot that day. Days in which an actor is present on location but not required for filming are called "hold days" for the actor concerned. In some instances, a talent may travel back and forth from his residence to the filming location, in which case the producers of the film pay only for those days the talent is present on location. However, the producers must then cover the cost associated with this travel. It is often more economical to employ the talent continuously and pay for the hold days incurred (Ref. 1).

We will present a model in which the minimization of the cost due to hold days is the sole criterion for the optimal scheduling of a film. In general, a variety of other criteria will also be involved in the decision process. However, the cost of hold days does, in real situations, feature prominently and so it will be of use to isolate this factor for modelling. Other considerations such as restricted availability of some talent, setup costs for scenes, artistic constraints, and so on are ignored in this paper.

The model which we will consider is a combinatorial one. We will attempt to sequence the shooting days of the film so that the actors are employed as continuously as possible and the cost of paying actors for hold days is minimized.

2. Literature Review

The problem which we consider in this paper could be regarded as a project scheduling problem. There is a large body of literature available on this class of problems. However, our problem possesses two aspects, which together make it very different from the types of project scheduling problems considered to date. One fundamental difference is that we are considering scheduling the work of particular employees. While a typical project scheduling problem considers the scheduling of the different tasks of a project where each task may require a certain number of employees of a given class to perform (e.g., two physicists and three chemists are required for the R&D task of some project), it does not require particular employees (or other resources such as machines and tools) at given times (e.g., machine 1 and machine 2 must work simultaneously on processing a job; Tom Jackson and John Smith are required for the last stage of the project).

The second aspect is the concept of incurring a penalty for hold days. There are many models of manufacturing scheduling in which penalty is incurred for allowing machines idle. But these models are fundamentally different from our model in that generally only one machine works on a given job at a time, whereas in our problem many actors may be required to work simultaneously on one scene.

So far, we have not been able to locate any literature (other than some qualitative studies from the film industry) which addresses problems similar enough in structure to ours to provide guidance or reference for model formulation and method of solution.

3. Model Formulation

We assume that the filming is divided into a number of shooting days, each of which requires a different subset of the talent contracted for the film. Scenes with similar talent or setup requirements may be grouped together into one shooting day; or an entire shooting day may be devoted to a single scene of a particularly complex nature. Normally, once it is decided which scenes will be shot on which days, the talent requirement for each day is recorded on a calendar cross plot called the "day out of days" (Ref. 2). In the day out of days, each column corresponds to one shooting day and each row corresponds to one actor. If an actor is required on a given shooting day, a mark (usually a number assigned to the actor's character) will be made in the cell where the row corresponding to that actor intersects with the column corresponding to the day. Assuming that the grouping of the components of the film into shooting days is given, we consider the problem of sequencing the shooting days in such a way that the cost due to talent hold days is minimized. We will use the generic term "actor" to refer to any person, animal, equipment, location, and so on which must be contracted on a continuous basis so that all off-days (hold days) must be paid.

Consider a film shoot composed of n shooting days and involving a total of m actors. We represent the requirements for the various shooting days by the day out of days matrix (DODM) $T^0 \in \{0, 1\}_{m \times n}$, with the (i, j) entry given by

$$t_{ij}^0 = \begin{cases} 1, & \text{if actor } i \text{ is required in shooting day } j, \\ 0, & \text{otherwise.} \end{cases}$$

We also define the pay vector $C \in \Re^m$, with the ith element given by

$$c_i = \text{rate of pay per day of the } i\text{th actor.}$$

Together, T^0 and C determine all the data required for the problem. Let $\sigma \in \sigma_n$ be any permutation of the n columns of T^0, i.e.,

$$\sigma: \{1, \ldots, n\} \to \{1, \ldots, n\},$$

where σ_n is the permutation set of the n shooting days. Define $T(\sigma)$ to be the matrix T^0 with its columns permuted according to σ. Then,

$$t_{ij}(\sigma) = t^0_{i\sigma(j)}, \qquad \text{for } i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}.$$

Since each column of T^0 is associated with a particular shooting day, σ determines a filming sequence or schedule S for the filming. Let $e_i(\sigma)$ and $l_i(\sigma)$ denote respectively the earliest and latest days in the schedule S determined by σ which require actor i. Then, actor i must be present on location for exactly $l_i(\sigma) - e_i(\sigma) + 1$ number of days. But he is actually required for only $r_i = \sum_{j=1}^n t^0_{ij}$ of those days, where r_i is the ith element of $R \in \Re^m$, the vector of number of days each actor is required to be present. Let $h_i(S)$ be the ith element of $H \in \Re^m$, the vector of number of days each actor is on hold under the schedule S. Then, the number of hold days for actor i in the schedule S is given by

$$\begin{aligned} h_i(S) = h_i(\sigma) &= l_i(\sigma) - e_i(\sigma) + 1 - r_i \\ &= l_i(\sigma) - e_i(\sigma) + 1 - \sum_{j=1}^n t^0_{ij}. \end{aligned}$$

It follows that the total cost of hold days for the film is

$$K(\sigma) = \sum_{i=1}^m c_i h_i(\sigma) = \sum_{i=1}^m c_i \left[l_i(\sigma) - e_i(\sigma) + 1 - \sum_{j=1}^n t^0_{ij} \right],$$

and our problem, referred to as the film scheduling (FS) problem, becomes

$$\text{(FS)} \quad \underset{\sigma \in \sigma_n}{\text{minimize}}\, K(\sigma) = \sum_{i=1}^m c_i \left[l_i(\sigma) - e_i(\sigma) + 1 - \sum_{j=1}^n t^0_{ij} \right].$$

4. Strong NP-Hardness of the Film Scheduling Problem

In this section, we will show that the FS problem is NP-hard by a reduction from the optimal linear arrangement (OLA) problem, which has been shown to be NP-hard by Garey et al. (Ref. 3). In fact, the FS problem is strongly NP-hard, because we show that, even the restricted version, in which each actor is needed for just two days and pay vector is 1, is polynomially reducible to the OLA problem. Thus, even the existence of a pseudopolynomial algorithm is unlikely (Ref. 4).

Optimal Linear Arrangement Problem. Given an undirected graph $G = (V, E)$ and a positive integer B, does there exist a one-to-one function $\sigma: \{1, 2, \ldots, |V|\} \to \{1, 2, \ldots, |V|\}$ such that $\sum_{[V_i, V_j] \in E} |\sigma(i) - \sigma(j)| \leq B$?

Theorem 4.1. The FS problem is strongly NP-hard.

Proof. Given an instance of OLA with $G = (V, E)$ and B, we construct an instance of the recognition version of the FS problem with the DODM T^0 as follows. Each edge E_i in E corresponds to row i in T^0, while each V_j in V corresponds to column j of T^0 such that $t_{ij}^0 = 1$ if and only if V_j is a terminal of edge E_i; otherwise, $t_{ij}^0 = 0$. And, we define the pay vector by letting $c_i = 1$ for all i. We claim that the answer to the OLA problem is "yes" if and only if the instance of the FS problem has a hold days cost

$$K(\sigma) \leq B - |E|.$$

It is easy to see that this construction takes polynomial time and the guessing and checking of an answer can be nondeterministically performed in polynomial time.

It is clear that

$$\sum_{j=1}^{|V|} t_{ij}^0 = 2, \qquad \text{for all } i.$$

Thus, any permutation or linear arrangement σ of vertices in V such that

$$\sum_{[V_i, V_j] \in E} |\sigma(i) - \sigma(j)| \leq B$$

will imply that the hold days cost corresponding to σ is

$$\begin{aligned} K(\sigma) &= \sum_{i=1}^{|E|} 1 \times [l_i(\sigma) - e_i(\sigma) + 1 - 2] \\ &= \sum_{i=1}^{|E|} [l_i(\sigma) - e_i(\sigma)] - |E| \\ &= \sum_{[V_i, V_j] \in E} |\sigma(i) - \sigma(j)| - |E| \\ &\leq B - |E|. \end{aligned}$$

The converse is also true. Therefore, the proof is completed. □

5. Branch-and-Bound Solution

We propose solution of this problem by a branch-and-bound algorithm where each node in the branching tree represents a partially determined schedule. Denote by J_i the shooting day which is scheduled to be filmed in the ith day of the shoot. We determine the schedule from outside in by determining J_1 first, then J_n, then J_2, J_{n-1}, then J_3, J_{n-2}, and so on in that order. Thus, the nodes in the first level below the root in the branching tree represents partial schedules for which only J_1 has been determined. The nodes in the second level represent partial schedules with J_1 and J_n determined. In the third level, J_1, J_n and J_2 are determined, and so forth. In the next two sections, we will discuss the construction of bounds and the search strategy used in the branch-and-bound algorithm.

5.1. Bounds. For each node or partial schedule, we determine a lower bound on the cost of hold days as follows. First, note that the shooting days in a partial schedule P can be divided into two subsets:

$$E(P) = \{k \in \{1, \ldots, n\} \mid J_k \in P \text{ and } k \leq \lceil n/2 \rceil\};$$

$$L(P) = \{k \in \{1, \ldots, n\} \mid J_k \in P \text{ and } k > \lceil n/2 \rceil\}$$

where $\lceil j \rceil$ denotes the smallest integer greater than or equal to j. We call the shooting days in $E(P)$ early and those in $L(P)$ late. We then note that, in any partial schedule, if a given actor is required in both an early shooting day and a late shooting day (which have already been scheduled), then the number of hold days for that actor is determined.

Let P be any partial schedule and define

$$\epsilon_i(P) = \begin{cases} 1, & \text{if actor } i \text{ is required in a shooting day scheduled early in } P, \\ 0, & \text{otherwise,} \end{cases}$$

$$\lambda_i(P) = \begin{cases} 1, & \text{if actor } i \text{ is required in a shooting day scheduled late in } P, \\ 0, & \text{otherwise.} \end{cases}$$

Also, define

$$e_i(P) = \begin{cases} \text{first day in } P \text{ where actor } i \text{ is required}, & \text{if } \epsilon_i(P) = 1, \\ 0, & \text{otherwise,} \end{cases}$$

$$l_i(P) = \begin{cases} \text{last day in } P \text{ where actor } i \text{ is required}, & \text{if } \lambda_i(P) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

We now define a schedule S to be a completion of P if and only if S is a complete schedule which agrees with P on each day which is scheduled in P. Then, if $\epsilon_i(P)\lambda_i(P) = 1$, the cost of hold days for actor i in any completion S of P, $K_i(S)$, is given by •

$$K_i(S) = c_i\left[l_i(P) - e_i(P) + 1 - \sum_{j=1}^{n} t_{ij}^0\right]. \tag{1}$$

Let $T(P)$ be the partial day out of days matrix determined by P. The columns of $T(P)$ are the columns of T which correspond to shooting days scheduled in P. If n' shooting days are scheduled in P, then $T(P) \in \{0, 1\}_{m \times n'}$. Now suppose that, for some $i \in \{1, \dots, m\}$, there exists $j \in E(P)$ such that the following condition holds:

$$0 < \sum_{k=1}^{j-1} t_{ik}(P) = \sum_{k=1}^{j} t_{ik}(P) < \sum_{k=1}^{n} t_{ik}(P). \tag{2}$$

The first inequality guarantees that actor i is required for some day scheduled earlier than j while the last inequality guarantees that he is also required for some day scheduled later than j. Then, actor i must be present on location on day j. But the equality in the middle implies that $T_{ij}(P) = 0$, which means that actor i is not required for day j of the partial schedule P. Thus, j is a hold day for actor i in any completion S of P.

Similarity, if $j' \in L(P)$ satisfies the condition

$$0 < \sum_{k=j'+1}^{n} t_{ik}(P) = \sum_{k=j'}^{n} t_{ik}(P) < \sum_{k=1}^{n} t_{ik}(P), \tag{3}$$

then j' is a hold day for actor i in any completion of P. Now, let

$D_i^e(P) =$ number of days $j \in E(P)$ which satisfy condition (2),

$D_i^l(P) =$ number of days $j' \in L(P)$ which satisfy condition (3).

Then, $D_i^e(P) + D_i^l(P)$ is a lower bound on the number of hold days for actor i in a completion S of P. If $\epsilon_i(P)\lambda_i(P) = 0$, the cost $K_i(S)$ due to hold days for actor i cannot be determined by Eq. (1). However, a lower bound on this cost is given by

$$K_i(S) \geq c_i\{D_i^e(P) + D_i^l(P)\}.$$

Thus, we have the following lower bound for the cost due to hold days of a completion S of the partial schedule P:

$$K(S) \geq \sum_{i=1}^{m} c_i \left\{ \epsilon_i(P)\lambda_i(P)\left[l_i(P) - e_i(P) + 1 - \sum_{j=1}^{n} t_{ij}^0 \right] + [1 - \epsilon_i(P)\lambda_i(P)][D_i^e(P) + D_i^l(P)] \right\}. \tag{4}$$

This determines the lower bound for the node in the branching tree which corresponds to P.

5.2. Search Strategy. Now that we have the bounds for the nodes in the branching tree, we must develop a search strategy—a set of rules which will determine the order in which the nodes of the branching tree are examined. Various such search strategies have been proposed, but the one which we are primarily interested in is the depth first search. One of the main advantages of this search technique is that it requires very little storage of information during the process and thus is also one of the easiest strategies to implement on computers. We will use a depth first search with the following two modifications.

Most Attractive Route. In order to find a good starting solution before we begin the depth first search, we take the most attractive route to the bottom of the tree. This means that we construct a complete schedule from outside to in (see discussion in the previous section) by selecting, at each stage, the shooting day which would increase the current lower bound by the least amount. Thus, upon arriving at some node in the tree, we will always branch to the adjacent node on the next level which has the lowest bound. If there is a tie, we branch to the first node accessed which has the minimum bound.

Since all bounds on the first level of the branching tree (one shooting day scheduled) and zero in our case, we will begin the above process at the second level of the branching tree. Thus, in the first stage, we consider all $n(n-1)/2$ nodes on the second level of the tree and branch to the one with the lowest bound. We then continue the process, scheduling one day at a time, until we reach the bottom of the branching tree (schedule completed), obtaining a starting feasible solution for the depth first search.

This procedure is expected to give a good starting solution since, in general, we see that the cost incurred by scheduling outside days (close to the start or end of the shoot) is higher than the cost of scheduling inside days (close to the middle of the shoot). For example, if an actor is required for only two shooting days, then the highest cost for that actor will clearly be incurred by scheduling those days on the first and last days of the shoot. The number of hold days (and thus the cost of hold days) decreases as we schedule the two days progressively toward the middle of the shoot. Thus, heuristically speaking, the scheduling of the outside days is most crucial and so it makes good sense to allow ourselves the most freedom (most shooting days to choose from) and to choose shooting days which incur the least immediate cost when scheduling the outside days.

Pairwise Interchanges. French (Ref. 5) provides a discussion of this technique, which consists of calculating the change in value of the objective function as a result of interchanging the positions of two jobs (shooting days) in a completed schedule. If the objective function value improves, then the interchange is carried out. This process continues until we arrive at a schedule for which none of the interchange of the $n(n-1)/2$ pairs of

Table 1. Elements of the day-out-of-days matrix and pay vector.

i	Actor	t_{ij}^0	c_i
1	Luce	1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0	1000
2	Tom	1 1 1 0 0 0 0 1 1 0 0 1 1 1 0 0 1 1 1 1 1 0 1 0 0 0 0 1	400
3	Mindy	0 1 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0	500
4	Maria	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0	500
5	Gianni	0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 1 1	500
6	Dolores	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0	4000
7	Lance	0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0	400
8	Sam	0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	2000

jobs (shooting days) will result in an improvement of the objective. We will call such a schedule a locally optimal schedule or more briefly, a local optimum.

In our search strategy, we will implement such a search for a locally optimal schedule each time we come to the bottom of the branching tree in the course of our search. Note that this includes the first time we arrive at the bottom of the tree, after exploring the most attractive route as discussed above.

Example 5.1. We consider the example of a feature film called *Mob Story*, which was filmed in Winnipeg, Canada between February and March of 1989. The DODM T^0 is given in Table 1 along with a disguised pay vector C (the actual rates of pay are confidential).

The cost of hold days is $K(T^0) = \$36{,}400$. We applied to heuristic algorithm discussed above to get the locally optimal schedule

$$S = \{25, 4, 1, 12, 13, 15, 14, 3, 17, 18, 16, 2, 9, 7, \\ 10, 6, 8, 11, 5, 20, 21, 19, 23, 27, 24, 22, 28, 26\}.$$

The cost of hold days for this sequence is $K(S) = \$17{,}900$ and the CPU time required was 1.05 seconds. Thus, it seems that the heuristic algorithm has improved significantly upon the schedule T^0, which was constructed manually by the trial-and-error method.

6. Computational Complexity of the Heuristic Algorithm

In general, the complexity of branch-and-bound algorithms grows exponentially with the size of the instance of the problem considered. The heuristic algorithm (Steps 1 and 2 of the branch-and-bound algorithm), however, is a pseudopolynomial time algorithm. This can be seen as follows.

Most Attractive Path. In this part of the algorithm, we schedule one day at a time, each time updating the lower bound. Since this updating is independent of the number of days, it is easy to see that the time required to perform this part of the algorithm is bounded by a linear function of n, the number of shooting days.

Pairwise Interchanges. Let c_M be the maximum entry in the pay vector C. Then mnc_M is an upper bound on the cost of hold days for an instance of the problem with m actors and n shooting days. Since all the pay rates are given as integers ($'s per day), the minimum decrease in the objective function due to any interchange which the algorithm performs is $1. Thus, it takes at most mnc_M interchanges to arrive at a local optimum. Each time (except for the last iteration), we check the $n(n-1)/2$ pairs of shooting days to see if their interchange would improve the objective function. If we do not find such a pair, then the algorithm terminates. Thus, we must perform at most $[mn^2(n-1)/2]c_M$ such checks. In order to perform each check, we need to known the positions of the first, second, last and second last 1's in each row of the DODM; to get this information, we need to check each entry of the DODM once. Thus, the time required for each check t_c is bounded by a linear function of the number of entries mn, with $a, b \in \Re$, as follows:

$$t_c \leq a + bmn.$$

Thus, the total time t_p for the pairwise interchange part of the heuristic algorithm is pseudopolynomially bounded in both m and n, given by

$$t_p \leq (a + bmn)[mn^2(n-1)/2]c_M.$$

7. Computational Experience

We applied the techniques discussed above to nine randomly generated instances of the problem at hand. The instances were generated in the following manner. First, we chose m and n, the number of actors and shooting days, respectively. For each of the m actors, we generated a random number n_i, $i = 1, \dots, m$, between 1 and n, which represented the number of shooting days which required actor i. Then, for actor i, we generated n_i random numbers, $n_{ij}, j = 1, \dots, n_i$, between 1 and n, which represented the n_i shooting days in T^0 which required actor i. Finally, we

Table 2. Computational results of nine randomly generated problem instances.

								Percentage error	
Instance	m	n	Starting objective value	Local objective value	CPU time (sec)	Global objective value	CPU time (sec)	Starting solution	Local optimal solution
1	6	6	178	134	0.05	134	0.11	32.8	0
2	7	7	174	167	0.07	134	0.32	4.2	0
3	8	8	163	145	0.11	145	0.39	12.4	0
4	9	9	553	494	0.13	457	2.65	21.0	8.1
5	10	10	648	435	0.17	435	10.67	49.0	0
6	10	10	607	507	0.16	424	7.86	43.2	19.6
7	11	11	1005	842	0.24	842	9.57	22.0	0
8	13	13	2099	1639	0.34	1639	104.36	28.1	0
9	14	14	1877	1149	0.45	1120	552.37	67.6	2.6

generated a random number c_i, $i = 1, \ldots, m$, between 1 and 100 for each actor to represent the actor's daily rate of pay. This defined the DODM and pay vector for each instance of the problem.

We first coded the branch-and-bound algorithm in the WATFIV programming language and applied it to solve the problems on the University of Manitoba AMDAHL mainframe computer. The computational results are summarized in Table 2. We record the value of the objective function for the starting solution (from most attractive route) and local optimum (from applying the pairwise interchange algorithm to the starting solution), the CPU time used to arrive at the local optimum, the objective function value at the global optimum, and the CPU time used for the complete branch-and-bound search.

It is evident from Table 2 that, for the problems considered, the first two steps of the algorithm (most attractive route and pairwise interchange) do, on average, as well or very nearly as well as the complete branch-and-bound search. The time required to implement these two steps, however, grows very slowly with the size of the instance n, whereas the time required to complete the entire branch-and-bound search seems to grow exponentially. Thus, the first two steps together will provide an excellent heuristic algorithm for solving large instances of the problem which could not otherwise be solved exactly in a reasonable amount of time.

Next, we solved one hundred instances of the problem (generated randomly as described above) with $m = 10$; ten instances were generated for each integer value of n in the interval [5, 14]. The average CPU times for execution of the heuristic algorithm as well as for the branch-and-bound search algorithm are recorded for each value of n considered. The average of the percentage error $E = ((Z_h - Z_b)/Z_b) \times 100$, where Z_h and Z_b are the objective function values from the heuristic and branch-and-bound solutions, respectively, is also recorded for each value of n. A graph of the average heuristic algorithm CPU time and branch-and-bound algorithm CPU time is plotted as a function of n in Fig. 1. The CPU time for the branch-and-bound algorithm grows rapidly (essentially exponentially) with n, but the CPU time for the heuristic algorithm grows much more slowly (apparently constant). Figure 2 displays a plot of the average percentage

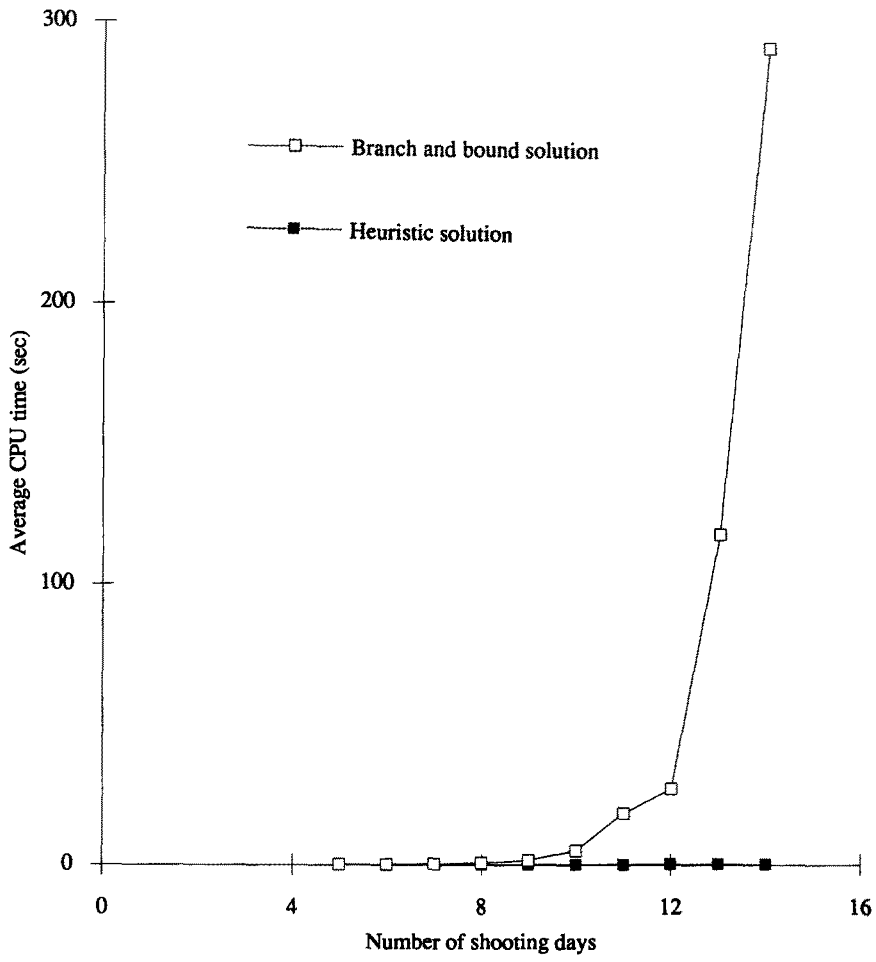

Fig. 1. Average CPU time versus number of shooting days.

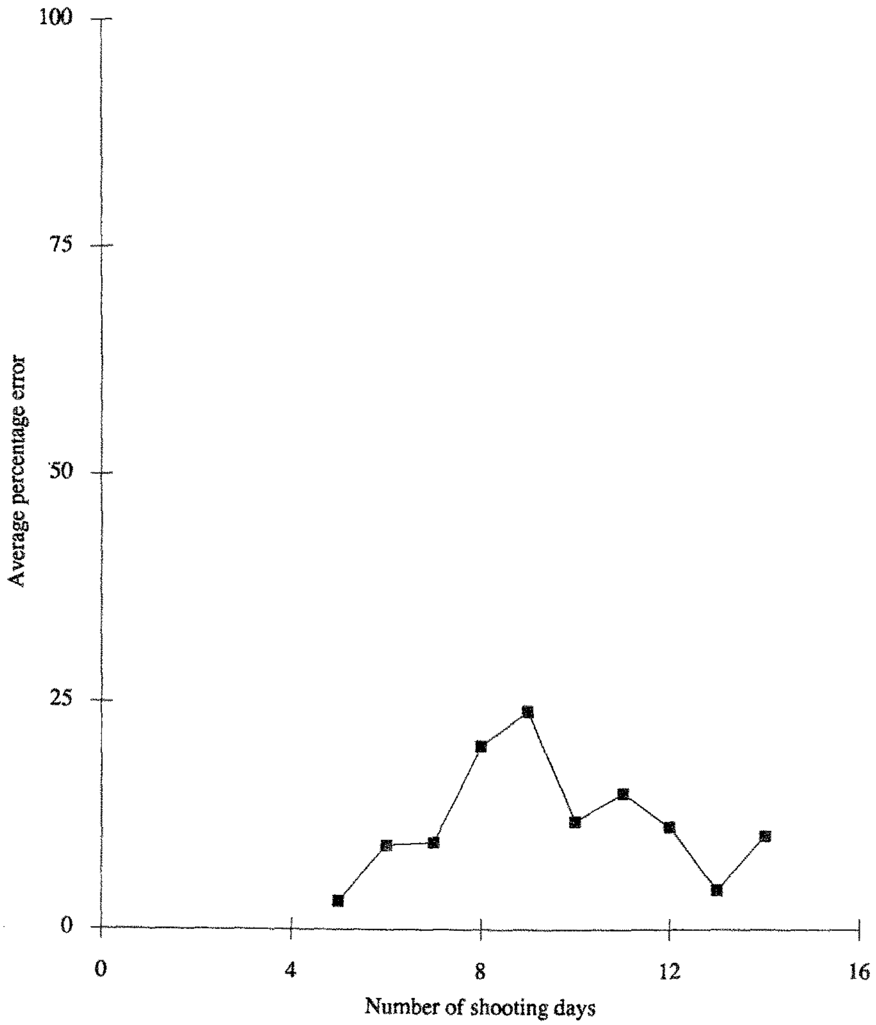

Fig. 2. Average percentage error versus number of shooting days.

error versus n. The percentage error seems to fluctuate randomly. This indicates that the heuristic algorithm is efficient in the range of n we have considered. Since the percentage error does not increase steadily with n, we can speculate that the percentage error for much larger problems will be comparable. Thus, it will clearly be of use to apply this heuristic algorithm to problems which are too large ($n > 15$) to be solved by the branch-and-bound algorithm in a reasonable amount of time.

8. Conclusions

It seems clear that we now have an effective heuristic algorithm for solving the FS problem. We believe that the program of the algorithms (both the branch-and-bound and heuristic methods) developed could be of use to the film industry in its present form. However, there are other factors (e.g., constraints due to actor availability, the possibility of sending actors home for long idle periods, and so on), which our model has not considered. So, although the schedule obtained from our model is optimal with respect to the cost of hold days, it may not be an optimal schedule for the film when all factors are taken into consideration. Nevertheless, the solution obtained by our model could serve as a starting point for the first assistant director who could alter the schedule to take into account all other relevant factors. The current model is being further developed in consultation with the film production practitioners to incorporate all factors considered crucial for the scheduling decision. Our next step is to give some practitioners of the film industry access to the program in order to evaluate its utility and to find areas for improvement. Other areas of application of this scheduling model, such as (perhaps) tool management in flexible manufacturing system designs, will also be explored.

References

1. LOHTKA, P., Private Communication, 1989.
2. HEHNER, B., *Making It: The Business of Film and Television in Canada*, Academy of Canadian Cinema and Television, Ottawa, Canada, 1987.
3. GAREY, M. R., JOHNSON, D. S., and STOCKMEYER, L., *Some Simplified NP-Complete Graph Problems*, Theoretical Computer Science, Vol. 1, pp. 237–267, 1976.
4. GAREY, M. R., and JOHNSON, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, New York, 1979.
5. FRENCH, S., *Sequencing and Scheduling: An Introduction to the Mathematics of Job Shop*, John Wiley and Sons, New York, New York, 1982.