Rehearsal Scheduling: Developing An Optimization Solution With Practitioner Input

Emily Hill · Mark Velednitsky

the date of receipt and acceptance should be inserted later

Abstract We consider the REHEARSAL SCHEDULING problem, typically stated in the context of theater, film, and performing arts. We are given a set of actors and scenes. Scenes may consist of several actors and actors may be in several scenes. We are also given a set of timeslots when scenes can be scheduled. The objective is to arrange the scenes into timeslots to produce a "good" schedule.

In theoretical work, "good" means minimizing the total *hold time*: the time between an actor's first scheduled scene and last scheduled scene during which they are not working. The theoretical setting is idealized to ignore actor conflicts, precedence between scenes, uneven scene lengths, optional actors, and other practical considerations. In practice, REHEARSAL SCHEDULING is as much an art as a science, as there is no concise, agreed-upon definition of what constitutes a good schedule.

In this work, we conduct a survey of practitioners who regularly solve the REHEARSAL SCHEDULING problem. Almost all of them currently solve the optimization problem by hand. We ask them to prioritize features of a good schedule. As a control, we also ask them to prioritize various utility features: features which do not affect the optimization, but affect how the user interacts with the system that generates the schedule. Based on the survey, we formulate the REHEARSAL SCHEDULING problem as a series of integer programs. We then build a tool for practitioners to schedule their shows.

We test the tool in 14 real-world use cases. We find that practitioners respond very positively to the results of the optimization solver. However, we learn that there are many other utility features which would be necessary for the optimization tool to gain widespread use.

E. Hill
University of California, Berkeley
E-mail: emilyhill@berkeley.edu

M. Velednitsky
University of California, Berkeley
E-mail: marvel@berkeley.edu

1 Introduction

In the REHEARSAL SCHEDULING problem, we are given a set of n actors and m scenes. Scenes require a certain set of actors. Actors may be required by several scenes. We are also given a set of timeslots into which scenes can be scheduled. The objective is to arrange the scenes into timeslots to produce a "good" schedule.

The TALENT SCHEDULING problem is the study of the REHEARSAL SCHEDULING problem in the idealized setting, with a single, clear objective function, namely to minimize *hold time*. We assume that the actor arrives for the first scene they are in and leaves after the last scene. Any time between their arrival and departure when they are not rehearsing is hold time. The TALENT SCHEDULING problem is sometimes called the HOLD COST MINIMIZATION problem.

The input to the TALENT SCHEDULING problem is typically presented as a binary matrix, where there is one row for each actor and one column for each scene. The output is an ordering of the columns of the matrix. For each row, its "hold time" is the number of zeros between the first and last one in the new column ordering. The objective is to minimize total hold time. See figure 1.

There is a dynamic programming algorithm for solving TALENT SCHEDULING optimally, though it has an exponential running time [1]. For solving the TALENT SCHEDULING problem on instances of realistic sizes, custom branch-and-bound algorithms have been effective. The key idea in most of these algorithms is to first specify the first scene, then specify the last scene, and gradually work "inwards" towards the middle scenes. The first such algorithm was proposed by Cheng, Diamond, and Lin [5] and later refined by Garcia de la Banda et al. [7], Qin et al. [10], and Cheng et al. [4]. Others have applied meta-heuristics to the problem, such as genetic algorithms [9].

Other authors have considered special cases of the *talent scheduling* problem. Determining if there is a solution with no hold time is equivalent to the CONSECUTIVE 1S problem, which can be solved in polynomial time [3]. The case where each actor appears in exactly two scenes is the LINEAR ARRANGEMENT PROBLEM, which is known to be NP-hard [5]. There exists a polylog-

Fig. 1: Example of an input to and output from the TALENT SCHEDULING problem, an idealized version of REHEARSAL SCHEDULING. Each row corresponds to an actor and each column corresponds to a scene.

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

(a) An example input to the TALENT SCHEDULING problem. In this ordering, the total hold time is 2.

(b) An example output from the TALENT SCHEDULING problem. The scenes are reordered to minimize hold time.

arithmic approximation algorithm for the LINEAR ARRANGEMENT PROBLEM [6].

The TALENT SCHEDULING problem is widely understood to be an idealized simplification of the more amorphous REHEARSAL SCHEDULING problem. TALENT SCHEDULING ignores actor conflicts, precedence between scenes, uneven scene lengths, optional actors, and other practical considerations. The custom branch-and-bound algorithms do not accommodate these complexities. As such, they are not adequate for practitioners. In practice, REHEARSAL SCHEDULING is as much an art as a science, as there is no concise, agreed-upon definition of what constitutes a "good" schedule.

Several authors have approached REHEARSAL SCHEDULING by extending the models and solution ideas used for TALENT SCHEDULING. Examples include Sakulsom et al., who consider the problem the problem of scheduling rehearsals of unequal length [11]. Another generalization comes from Wang et al., who recognized that daily and hourly scheduling should be considered separately: with separate costs for intra-day holding and inter-day holding [12].

Even after determining a suitable model that accounts for practitioners needs, there is considerable engineering work. A comprehensive system was developed by Bomsdorf and Derigs [2]. In their paper, they acknowledge "to be accepted in practice, any planning methodology has to be highly interactive, allowing fast and flexible rescheduling, and has to respect the planners problem solving style, allowing them to bring in their experience."

To develop a scheduling tool with the potential to be accepted for practical use, we took the insight of Bomsdorf et al. to heart. In our work, we take a user-first approach to the REHEARSAL SCHEDULING problem. Our primary contributions are:

- We conduct a survey of 44 practitioners of REHEARSAL SCHEDULING to determine what constitutes a "good" schedule. In addition to asking about the optimization features of a hypothetical scheduling system, we also ask about utility features: features which affect users' interactions with the tool, but not the optimization solution.
- Based on the survey results, we describe the REHEARSAL SCHEDULING problem as multi-objective optimization problem with three objectives. We treat the objective functions hierarchically and solve the problem as a series of three integer programs.
- We test our solution in 14 real-world use cases. We find that the optimization is adequate, but additional utility features would be required in order for the tool to have a possibility of wider adoption.

Our paper is organized as follows: In section 2, we describe our survey and its results. In section 3, we describe the integer programming formulation. In section 4, we describe our experimental set-up and results. In section 5, we conclude with a few final remarks.

Remark 1 (Terminology) There is some disagreement between the terminology used in the academic literature and the terminology used by practitioners.

In the academic literature, it is assumed that each scene is rehearsed once. In practice, a scene can be rehearsed several times. Switching between these contexts is not a problem, mathematically: if a scene needs to be rehearsed several times, then we can create several copies of it in the input.

For the sake of succinctly presenting our mathematical formulation, we will assume the convention that each scene is rehearsed once. However, in our survey and scheduling tool developed for practitioners, we will use the industry-standard terminology: a scene is defined once and potentially rehearsed several times.

2 Survey of Practitioners

Surveys are effective, low-cost, and non-intrusive tools to collect primary data about users' pain points, needs, and preferences surrounding an existing workflow. Human-centered design practitioners regularly implement insights derived from survey results into software development cycles [8].

We surveyed 44 active practitioners to understand their current scheduling workflow and factors that matter the most when they schedule. We collected the type of productions the practitioners do, the sizes of their respective production, and their current methods of creating schedules. The responses were collected through Google Form.

2.1 Survey Design

To formulate the REHEARSAL SCHEDULING problem in a way that appeals to practitioners, we needed to understand features of good schedules and their relative importance. The first section of our survey was dedicated to understanding the considerations practitioners take into account when they create a schedule. These features are described in section 2.1.1. As a control, we included a second section, in which we asked practitioners to rank certain usability features unrelated to the scheduling optimization. These control questions served as a useful insight into the relative importance of optimization features. They are described in section 2.1.2.

2.1.1 Optimization Features

Our survey asked participants how often they considered each of the following features when they designed a "good" schedule. Participants rated the frequency on a Likert Scale. 1 being "never considered" the feature, and 5 being "always considered" the feature.

Hard Conflicts Times when an actor cannot make it to rehearsal. It may be possible to rehearse a scene without a certain actor present, but, if possible, it is best to rehearse the scene when all the actors are present.

Soft Conflicts Times when an actor would prefer not to make it to rehearsal. It is nice to honor these if possible to keep the actors happy, but the actor will attend if called.

Space Constraints The availability of certain rooms in which the rehearsal can be held. For example, it may be desirable to only rehearse a certain scene when a music room is available or when a large room is available.

Time Efficiency Using the actors' time as efficiently as possible.[1]

Spacing In some cases, it may be important to space certain rehearsals *apart* from each other. For example, in one rehearsal the actors might learn choreography and then several days later they review it in another rehearsal. Even though these rehearsals use the same actors, it will be illogical to rehearse them consecutively. See remark 1.

Precedence Assuring that certain rehearsals happen before others. This may be necessary when there is a musical number that needs separate vocal and dance rehearsals before they can be combined in a joint rehearsal.

Parallelization Allow several rehearsals to be scheduled in the same timeslot. This is particularly useful in musicals, where there may be a separate vocal and dance rehearsals. No actor can participate in two rehearsals simultaneously.

Breaks Create a schedule in which there are intentional temporal gaps between rehearsals. These gaps serve a dual purpose. The first is to allow actors in several consecutive scenes some time to rest. The second is to serve as a buffer in case certain scene rehearsals take longer than expected.

To test the comprehensiveness of our list of features, our survey also asked practitioners if there were any features they regularly considered which we had not asked about: "What else do you think about when designing a good rehearsal schedule?"

2.1.2 Utility Features

Our survey asked participants, for them to want to use a scheduling tool, how important each of the following features is to them. Participants rated the importance on a Likert Scale. 1 being "not important," and 5 being "extremely important."

Manager Enters Availability Allow a central manager to enter the availability of each actor.

Actors Enter Availability Allow the individual actors to enter their availability themselves, so that a central manager does not have to enter it on the actors' behalf.

Schedule Chunks Rather than scheduling all the scenes into all the timeslots at once, allow the practitioner to select a subset of the scenes and a subset of the timeslots to schedule.

[1] For the purposes of the survey, this features was intentionally left vague, since "hold time" is not an industry-standard term. Indeed, since our timeslots are not all contiguous, we require a more nuanced notion of efficient time use. For more details, see section 3.

Feature	Average	**5**	**4**	**3**	**2**	**1**
Hard Conflicts	4.9	33	9	1	0	1
Manager Enters Availability	4.6	30	9	5	0	0
Prescheduling	4.5	30	10	1	2	1
Rescheduling	4.4	27	11	4	2	0
Space Availability	4.4	24	8	4	3	0
Time Efficiency	4.2	19	18	5	2	0
Schedule Chunks	4.2	25	8	4	4	2
Mobile	4.2	23	12	4	3	2
Choose Several Schedules	3.9	15	17	5	5	2
Integrations	3.9	18	14	5	3	4
Parallel Rehearsals	3.7	9	13	11	1	2
Scene Spacing	3.5	9	12	14	7	1
Actors Enter Availability	3.4	13	8	11	7	5
Soft Conflicts	3.1	1	14	18	11	0
Breaks	3.2	5	15	13	6	5
Revise Conflicts	2.8	6	7	11	10	10
Scene Ordering	2.8	1	8	13	11	7

Table 1: For each feature, we report the number of respondents who gave it a particular score on the Likert scale. The optimization features are highlighted blue. The utility features are white. The features are sorted by their average score in descending order.

Prescheduling Manually fix certain scenes to certain timeslots before calculating an optimal schedule.

Choose Several Schedules Choose from several possible schedules suggested by the algorithm.

Revise Conflicts Allow conflicts to be edited after the schedule has been created, possibly leading to rescheduling.

Rescheduling Allow the manager to reschedule all or a subset of the scheduled rehearsals.

Integrations With popular calendar products, for example.

Mobile Provide a mobile-friendly interface where the manager and actors can access the scheduling optimization input and output.

2.2 Survey Results

Practitioners had, on average, 3 productions in the last year that required them to spend more than 1 hour per week on scheduling rehearsals alone. Each production, on average, rehearsed for 6 weeks, with, on average, 35 actors and crew members in each production. In total, practitioners were spending, on average, 9.6 hours per production doing the scheduling by hand.

In table 1, for each feature we report the number of respondents who gave it a particular score on the Likert scale. The optimization features are highlighted blue. The utility features are white. The features are sorted by their average score in descending order.

To determine the highest-priority features, we used a t-test to determine which features were statistically distinguishable from each other. Among the top five features, for each pair we found $p > 0.05$. Among the top eight features, for each pair we found $p > 0.01$. On the other hand, **Hard Conflicts** and **Choose Several Schedules** were statistically significantly different from each other at the $p < 0.01$ level. Thus, we decided to distinguish the top eight features as the *high priority* features. The top optimization features were **Hard Conflicts**, **Space Availability**, and **Time Efficiency**. The top utility features were **Manager Enters Availability**, **Prescheduling**, **Rescheduling**, **Schedule Chunks**, and **Mobile**.

When asked if there where any features not covered in our survey, eight of the practitioners mentioned accounting for union contracts and labor laws, especially for shows involving child actors. These rules limit the number of consecutive hours an actor can work or the number of hours they can work in a day. While this feature was not applicable in our 14 real-world use cases, the fact that we omitted it may be a barrier to wider adoption. Aside from accounting for contracts and labor laws, no other missing features were reported by the practitioners, which suggests that our list of features was reasonably comprehensive.

From our survey, two results stand out:

- Among the optimization features, we discovered that accounting for actors' hard conflicts was even more important to the practitioners than making the schedule temporally efficient for the actors. In the theoretical study of TALENT SCHEDULING, conflicts are typically ignored. This result highlights the need to reformulate the optimization problem for practitioners.
- Many of the utility features we asked about as controls were *more* important to the practitioners than optimization features. For example, more than half of the practitioners responded that having a mobile version of the scheduling tool is an "extremely important" determinant to whether they would use the tool. More than half also considered the ability to reschedule to be "extremely important."

These survey results indicate that, for practitioners to adopt a scheduling tool in practice, they expect a highly flexible scheduling tool that addresses a few core optimization concerns. Out of 44 practitioners, only 14 reported that they were currently using one or more online tools to help them schedule rehearsals. Among the 14 who are using one or more online tools, only 1 uses professional artistic management and planning software specialized for theatre scheduling. The remaining 13 use Google Forms, when2meet, and Doodle to collect availability, Excel to organize data, and Google Calendar to provide visualizations. Given the proliferation of scheduling algorithms and specialized scheduling software, we found this number to be surprisingly low. Our survey results suggest that usability plays a large role in this phenomenon and may be the largest obstacle preventing practitioners from taking advantage of scheduling algorithms. We will revisit this idea in later sections.

3 Integer Programming Formulation

In addition to the aforementioned high-priority optimization features, there were two departures from the idealized setting we considered "obvious" and therefore did not ask about in the survey. The first was a distinction between days and hours. In the idealized setting, it is typical to assume that the timeslots are contiguous: either consecutive hours on the same day or consecutive days. In the practical setting, hours and days must be distinguished. The second was handling scenes of unequal length. For the sake of formulating the problem, we assume that the timeslots have equal length and that scenes require a positive integer number of timeslots. Both of these features have been recognized as important in previous works. For example, Wang, Chuang, and Lin [12] make a distinction between days and hours. Sakulsom and Tharmmaphornphilas [11] make a distinction between days and hours and also consider rehearsals of unequal length. In our case, we also consider actor conflicts, which is not addressed in either of the aforementioned works.

3.1 Variables and Objectives

We will use A to denote the set of actors, S to denote the set of scenes, and T to denote the set of timeslots. We assume that the timeslots have equal length and are numbered in temporal order. Let D be the set of days. The timeslots can be partitioned into days: $T_0, \ldots, T_{|D|}$. We use the notation ℓ_i to denote the *length* of scene i: the number of consecutive timeslots which scene i requires. For most practical instance of the REHEARSAL SCHEDULING problem, $|A| \lessapprox 100$, $|S| \lessapprox 100$, and $|T| \lessapprox 1000$.

We define the following binary variables. The variable y_{ij} will be 1 if scene i starts in timeslot j, 0 otherwise. The variable x_{ij} will be 1 if scene i is happening in timeslot j, 0 otherwise. The variable z_{jk} will be 1 if actor k is present in timeslot j, 0 otherwise. We intentionally avoided introducing any variable indexed over $|S| \times |T| \times |A|$, which would create a prohibitively large number of variables.

In our formulation, we group **hard conflicts** and **space constraints** into a single utility score for scene i in timeslot j: $R_{ij} \in \mathbb{R}$. The distinguishing property of these features is that the utility of scheduling a certain scene in a certain timeslot can be calculated independently of the rest of the scenes. On the other hand, **time efficiency** depends on the relative positions of *all* the scenes.

If a scene must be scheduled at a time when one of the actors in it has hard conflict, the effect on utility depends on the actor and the scene. For example, it is better to schedule a 16-person scene with 1 actor missing than it is to schedule a 2-person scene with 1 actor missing. For the sake of our experiments, the reward function we use is the fraction of actors who could attend the whole rehearsal multiplied by the length of the rehearsal. If the only space available at time j was undesirable for rehearsing scene i, then R_{ij} was

further multiplied by $\frac{1}{2}$. Formally, let s_i be the set of actors needed for scene i and let t_j be the set of actors available at time j. Then, for our experiments,

$$R_{ij} = \begin{cases} \ell_i \frac{|s_i \cap t_j|}{|s_i|} & \text{if desirable space available} \\ \frac{1}{2}\ell_i \frac{|s_i \cap t_j|}{|s_i|} & \text{if desirable space unavailable} \\ -1 & \text{if scene } i \text{ impossible at time } j. \end{cases}$$

Though our experiments use a relatively simple formula for the reward, one could imagine expressing a much richer set of features. For example, each scene could be assigned a scalar representing its importance. Then, the reward of a certain scene in a certain timeslot could be multiplied by the importance of the scene. Similarly, each actor could be assigned an importance in each scene and the reward could reflected a weighted average of their availability, weighted by their importance. We intentionally write our formulation in terms of "reward" to leave open the possibility of adding these features in the future.

The final feature we would like to capture is **time efficiency**. Recall that the *hold time* is typically defined in settings where all the timeslots can be thought of as contiguous. In our setting, we consider separate days. In order to account for this distinction, we need to update our understanding of hold time.

It is tempting, as a first pass, to simply optimize the total hold time for all the actors across all the days. However, this results in peculiar behavior. For example, consider an actor which is in exactly three scenes. Assume that there are three days worth of timeslots. To minimize their total hold time across all the days, one solution would be to schedule all three of their scenes consecutively on the same day. Since the three scenes are consecutive, the actor experiences no hold time between them. However, an equally optimal solution would schedule each of the actor's scenes on a different day. In this solution, the actor also experiences no hold time on any of the days because their first and last scene each day is the one and only scene they are called for.

For the above example, it is clear that we need to consider more than just hold time when deciding if we have used actors' time efficiently. We resort to considering two quantities: the total number of days for which each actor is called as well as the total hold time across all days.

In total, we consider three objectives:

1. maximize reward (accounting for hard conflicts and for space constraints)
2. minimize the number of days called, summed across actors (an aspect of time efficiency)
3. minimize the hold time, summed across actors (another aspect of time efficiency)

3.2 Optimizing Multiple Objectives

We considered several options for handling our multi-objective optimization problem. Ultimately, we opted to treat the objectives hierarchically, leading

to a three-stage solution process. The first stage finds a solution with the maximum possible reward. The second stage finds, among all solutions with maximum reward, the solution which minimizes the sum across actors of the number of days called. The third stage, calculated separately for each day, minimizes the total hold time across actors that day.

Our decision to treat the objectives hierarchically was based on our survey results and our preliminary experiments. In our survey, we found that the **hard conflicts** feature was significantly more important to the practitioners than the **time efficiency** feature ($p < 0.05$). In a hierarchical ordering of objectives, the former feature is prioritized over the latter, consistent with the survey. When we did try combining the objectives by introducing weights, our preliminary results suggested that the integer program took a prohibitively long time to solve. Treating the objectives hierarchically limits the search space, but makes the integer programs tractable. A reasonable solve time is crucial for practical usage.

Other researchers have also chosen to solve REHEARSAL SCHEDULING in phases. Sakulsom and Tharmmaphornphilas [11] and Wang, Chuang, and Lin [12] treated REHEARSAL SCHEDULING as a multi-objective optimization problem with two objectives, corresponding to our second and third objectives (days and hold time, respectively). In both cases, they ultimately solved the problem in two phases: one for each objective. Wang, Chuang, and Lin used heuristics to produce good feasible solutions in both phases. Sakulsom and Tharmmaphornphilas used a heuristic in the first phase and then solved an integer program in the second phase. In this work, we have three phases and solve them all with integer programming.

3.3 Reward Optimization

The aforementioned reward of scheduling scene i to start in timeslot j is $R_{ij} \in \mathbb{R}$. In this formulation, we find an assignment of scenes to timeslots that maximizes total reward.

$$\max \sum_{i \in S, j \in T} R_{ij} y_{ij} \tag{IP1}$$

$$\sum_{j \in T} y_{ij} \leq 1 \quad \forall i \in S \qquad \% \text{ Each scene rehearsed at most once.}$$

$$\sum_{j'=j-\ell_i+1}^{j} y_{ij'} = x_{ij} \quad \forall i \in S, j \in T \qquad \% \text{ Define the variable x.}$$

$$\sum_{i \in S} x_{ij} \leq 1 \quad \forall j \in T \qquad \% \text{ At most one rehearsal at a time.}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in S, j \in T$$

$$y_{ij} \in \{0,1\} \quad \forall i \in S, j \in T$$

Notice that if all the scenes have length $\ell_i = 1$, then $x_{ij} = y_{ij}$, and this problem becomes a maximum bipartite matching problem between scenes and timeslots.

On the other hand, if R_{ij} depends only on the scene i, not on the timeslot j, then $\sum_{j \in T} y_{ij}$ becomes a binary indicator of whether or not scene i gets assigned to any timeslot. Finding the set of rehearsals which maximize reward while not exceeding the number of timeslots is equivalent to the Knapsack problem.

3.4 Time Efficiency Optimization

Daily For the next phase in our optimization formulation, we introduce the variable δ_{dk}. This variable will be 1 if actor k is called on day $d \in D$, 0 otherwise. Let R be optimal objective value found in formulation IP1.

The idea of this formulation is to find, among the solutions with maximum reward, the one that minimizes the sum over all the δ_{dk}.

$$\min \sum_{d \in D, k \in A} \delta_{dk} \qquad \text{(IP2)}$$

$$\sum_{j \in T} y_{ij} \leq 1 \quad \forall i \in S \qquad \% \text{ Each scene rehearsed at most once.}$$

$$\sum_{j'=j-\ell_i+1}^{j} y_{ij'} = x_{ij} \quad \forall i \in S, j \in T \qquad \% \text{ Define the variable x.}$$

$$\sum_{i \in S} x_{ij} \leq 1 \quad \forall j \in T \qquad \% \text{ At most one rehearsal at a time.}$$

$$z_{jk} = \sum_{\{i | k \in s_i\}} x_{ij} \quad \forall k \in A, j \in T \qquad \% \text{ Define the variable z.}$$

$$\delta_{dk} \geq z_{jk} \quad \forall d \in D, j \in T_d, k \in A \qquad \% \text{ Define the variable } \delta.$$

$$\sum_{i \in S, j \in T} R_{ij} y_{ij} = R \qquad \% \text{ Maintain reward.}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in S, j \in T$$
$$y_{ij} \in \{0,1\} \quad \forall i \in S, j \in T$$
$$z_{jk} \in \{0,1\} \quad \forall j \in T, k \in A$$
$$\delta_{dk} \in \{0,1\} \quad \forall d \in D, k \in A$$

Hourly In the previous formulation, we assigned scenes to days. Let S_d be the set of scenes which were assigned to day $d \in D$ and let R_d be the reward achieved that day. Our final formulation assumes the scenes are fixed to certain days and refines the hold time within the days. Because the formulations for each day do not affect each other, they can be solved in parallel.

For the sake of this program, we will only consider the timeslots which fall on a particular day d. Recall that T_d denotes these timeslots. We introduce two new variables, $h^+_{jk} \in \{0,1\}$ and $h^-_{jk} \in \{0,1\}$. The variable h^+_{jk} will be 1 if and only if actor k is called for a scene at *or after* timeslot j. The variable h^-_{jk} will be 1 if and only if actor k is called for a scene at *or before* timeslot j. Thus, the sum $h^+_{jk} + h^-_{jk}$ is 2 for actor k if they are rehearsing or held at timeslot j. Otherwise, the sum is 1.

$$\min \sum_{j \in T_d, k \in A} h^+_{jk} + h^-_{jk} \tag{IP3}$$

$$\sum_{j \in T_d} y_{ij} \leq 1 \quad \forall i \in S_d \qquad \% \text{ Each scene rehearsed at most once.}$$

$$\sum_{j'=j-\ell_i+1} y_{ij'} = x_{ij} \quad \forall i \in S_d, j \in T_d \qquad \% \text{ Define the variable x.}$$

$$\sum_{i \in S_d} x_{ij} \leq 1 \quad \forall j \in T_d \qquad \% \text{ At most one rehearsal at a time.}$$

$$z_{jk} = \sum_{\{i | k \in s_i\}} x_{ij} \quad \forall k \in A, j \in T_d \qquad \% \text{ Define the variable z.}$$

$$\sum_{i \in S_d, j \in T_d} R_{ij} y_{ij} = R_d \qquad \% \text{ Maintain reward.}$$

$$h^+_{jk} \geq z_{jk} \quad \forall j \in T_d, k \in A \qquad \% \ h^+\text{: if an actor } \textit{is} \text{ called.}$$

$$h^-_{jk} \geq z_{jk} \quad \forall j \in T_d, k \in A \qquad \% \ h^-\text{: if an actor } \textit{is} \text{ called.}$$

$$h^+_{jk} \geq h^+_{j+1,k} \quad \forall j \in T_d, k \in A \qquad \% \ h^+\text{: if an actor } \textit{will be} \text{ called.}$$

$$h^-_{jk} \geq h^-_{j-1,k} \quad \forall j \in T_d, k \in A \qquad \% \ h^-\text{: if an actor } \textit{was} \text{ called.}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in S_d, j \in T_d$$

$$y_{ij} \in \{0,1\} \quad \forall i \in S_d, j \in T_d$$

$$z_{jk} \in \{0,1\} \quad \forall j \in T_d, k \in A$$

$$h^+_{jk} \in \{0,1\} \quad \forall j \in T_d, k \in A$$

$$h^-_{jk} \in \{0,1\} \quad \forall j \in T_d, k \in A$$

3.5 Solutions

The relation among the integer programs is illustrated in figure 2. We solved the integer programs using the open-source solver COIN-OR Branch and Cut (CBC) with a one-minute time limit. In practice, on instances of practical size and structure, IP1 was typically solved to optimality within the time limit, IP2 typically terminated at the time limit with an optimality gap less than 20%, and IP3 was typically solved to optimality within the time limit.

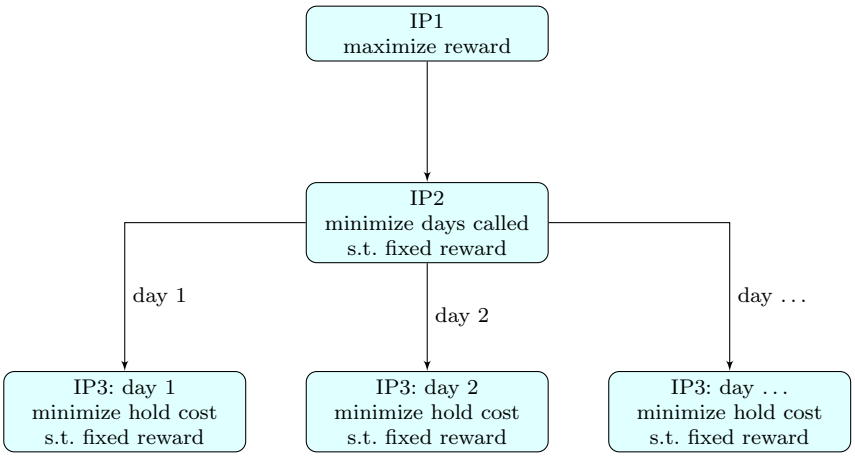


Fig. 2: A flowchart illustrating the series of integer programs we use to find a "good" feasible solution to the REHEARSAL SCHEDULING problem. Arrows indicate sending a feasible solution from one integer program to the next.

4 Experiments

4.1 Setup

We conducted user studies with a total of 14 practitioners, in two phases. We helped practitioners generate optimized rehearsal schedules for their shows using our scheduling system.

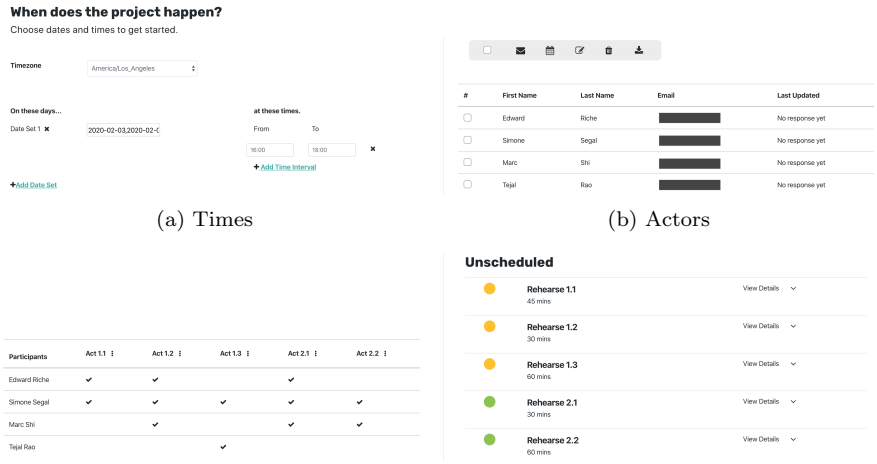
A diverse set of shows were tested, including a mix of plays and musicals, classical and new. Some of the shows used in this phase of testing included *Hamlet*, *The 25th Annual Putnam County Spelling Bee*, *Twelfth Night*, *Salomé*, *As You Like It*, *The Wizard of Oz*, *A Midsummer Night's Dream*, *Next to Normal*, and several independent works. The theater companies were predominantly located in the San Francisco Bay Area and the Greater Boston Area.

Phase I: Experimenter-in-the-Loop First, we tested our scheduling optimization algorithm by manually entering the scheduling data and constraints for two practitioners scheduling their respective shows. The practitioners were asked to evaluate the resulting schedules. Practitioners were not presented with an interface.

Next, we built a software tool for the practitioners to enter inputs to the algorithm, themselves. We designed and developed the software with the high-priority usability features in mind (see section 2.2). In particular, we implemented **Manager Enters Availability**, **Prescheduling**, **Rescheduling**, and **Schedule Chunks**. Images of the interface can be seen in figure 3.

In this phase, we used "experimenter-in-the-loop" testing to assist practitioners navigating the interface. The practitioners were prompted to use the

Fig. 3: Images of the interface in which users entered data for the scheduling algorithm.



(a) Times (b) Actors

(c) Scenes (d) Rehearsals

interface to generate a rehearsal schedule for a show that they had managed in the past. Six practitioners participated. In the software interface, the practitioners were prompted to input the following data:

Times General dates and times when rehearsals could happen.
Actors Names and emails of actors.
Scenes The set of actors required for rehearsing a section of the play. See remark 1.
Rehearsals The duration and actors needed for each rehearsal. See remark 1.

The practitioners were asked to evaluate the quality of the schedule generated by the algorithm. They were also asked to evaluate their experience of using the scheduling tool.

Phase II: Experimenter-out-of-the-Loop After iterating our interactive software prototype, incorporating the feedback we collected from *Phase I*, we sent the updated software prototype to six participants to complete the task of scheduling rehearsals for their respective shows. Participants were asked to perform the scheduling independently, and on their own time. They were asked to evaluate the resulting schedule generated by the algorithm, as well as the experience of using the scheduling software.

4.2 Results

Participants responded very positively to the schedules produced by the algorithm, and were relieved by the ease of automating scheduling. Notably, in

Phase I of the experiment, all participants were able to complete scheduling rehearsals for the entire production in 2 hours, in contrast to the average 9.6 hours spent scheduling each production using their current methods.

Phase I: Experimenter-in-the-Loop All the participants reported that they were satisfied with the optimized schedules that the algorithm generated. Participants said that the automatically optimized rehearsal schedules were similar to schedules they manually created. Participants further stated that deviations from their manually produced schedules were either insignificant (for example, swapping similar rehearsals) or reasonable (for example, scheduling some rehearsals at better times at the expense of other rehearsals). All participants finished creating rehearsal schedules for the entirety of their shows under 2 hours, a significant reduction from 9.6 hours on average spent scheduling per production as reported in the survey.

The participants were allowed to give free-form feedback. Several commented on the time and energy saved. Others commented on the quality of schedules generated. Responses included:

- "I cannot tell you how exciting it is to have all the scheduling done for me."
- "It personally would have saved me a lot of work."
- "The schedules are clearly better than last year."
- "Your algorithm really saved us time and stress."
- "This is really cool. If people can put their conflicts in, it's, like, no work."
- "The workflow can be a little simpler, but how the tool automatically created this perfect schedule was cool. I see a lot of potential in this tool."

One limitation of the user study was not including the time of rescheduling rehearsals due to changes in actor availability.

Phase II: Experimenter-out-of-the-Loop This turned out to be the more challenging testing session for users. One user confessed they in fact did not finish scheduling, saying that "[The software] is too complicated." While the practitioners who did finish scheduling were satisfied with the resulting schedule (one practitioner called the results "delightful"), they were significantly less satisfied with the process than the users who received experimenter assistance in Phase I. The practitioners expressed concern that the data entry was cumbersome and lacked flexibility. One practitioner said they would "probably not use it again." Another said they would "rather manually schedule by themselves," because of familiarity and flexibility.

The only difference between *Phase I* and *Phase II* was the guidance of researchers in navigating the interface, in addition to minor improvements to the interface. Experimenters' guidance turned out to be fundamental to the practitioners' success and positive experience using the scheduling tool.

Additionally, our user studies revealed that users have varying levels of trust in computer-generated results. Participants were curious about yet skeptical of the schedule generated by the computer throughout the data entry

process. All eight users from *Phase I* expressed the desire to tweak the outcome of the initial computer-generated schedule. Giving users the flexibility to make free-form changes is important. The ability to freely manipulate and potentially re-purpose the outcome returns agency back to the users, which positively contributes to users' experience of interacting with the scheduling software.

5 Final Remarks

In designing a scheduling system, our goal was to capture human intuition for what makes a good schedule in an integer programming formulation. Our user studies suggest that we achieved that, generating schedules which felt "good" to practitioners.

Although our tool was adequate from an optimization standpoint, a lot more work would need to be done outside of optimization in order to make the tool appealing to non-technical practitioners. In our survey, we discovered that certain utility features were as popular as, and sometimes more popular than, the optimization features.

A practical scheduling tool needs to support scheduling demands of varying intensity, to improve significantly or at least be compatible with practitioners' existing workflows, and be considerate of users' comfort level with foreign software interfaces.

Through our experiment, we found that the best practice for building a usable scheduling tool is to involve users in every cycle of the development, collecting their feedback and iterating. Rather than defer this to be handled by engineers and designers, we believe this process should start as early as developing an optimization formulation that incorporates practitioner input and is aware of their non-optimization needs.

References

[1] RM Adelson, JM Norman, and G Laporte. "A dynamic programming formulation with diverse applications". In: *Journal of the Operational Research Society* 27.1 (1976), pp. 119–121.

[2] Felix Bomsdorf and Ulrich Derigs. "A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem". In: *Or Spectrum* 30.4 (2008), pp. 751–772.

[3] Kellogg S Booth and George S Lueker. "Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms". In: *Journal of computer and system sciences* 13.3 (1976), pp. 335–379.

[4] Tai Chiu Edwin Cheng, Bertrand Miao-Tsong Lin, Hsiao-Lan Huang, et al. "Talent hold cost minimization in film production". In: (2017).

[5] TCE Cheng, JE Diamond, and BMT Lin. "Optimal scheduling in film production to minimize talent hold cost". In: *Journal of Optimization Theory and Applications* 79.3 (1993), pp. 479–492.

[6] Uriel Feige and James R Lee. "An improved approximation ratio for the minimum linear arrangement problem". In: *Information Processing Letters* 101.1 (2007), pp. 26–29.

[7] Maria Garcia de la Banda, Peter J Stuckey, and Geoffrey Chu. "Solving talent scheduling with dynamic programming". In: *INFORMS Journal on Computing* 23.1 (2011), pp. 120–137.

[8] Andreas Holzinger. "Usability engineering methods for software developers". In: *Communications of the ACM* 48.1 (2005), pp. 71–74.

[9] Anna-Lena Nordstrom and Suleyman Tufekci. "A genetic algorithm for the talent scheduling problem". In: *Computers & Operations Research* 21.8 (1994), pp. 927–940.

[10] Hu Qin et al. "An enhanced branch-and-bound algorithm for the talent scheduling problem". In: *European Journal of Operational Research* 250.2 (2016), pp. 412–426.

[11] Noppadon Sakulsom and Wipawee Tharmmaphornphilas. "Scheduling a music rehearsal problem with unequal music piece length". In: *Computers & Industrial Engineering* 70 (2014), pp. 20–30.

[12] Sin-Yi Wang, Ying-Ting Chuang, and Bertrand MT Lin. "Minimizing talent cost and operating cost in film production". In: *Journal of Industrial and Production Engineering* 33.1 (2016), pp. 17–31.