Probabilistic Matrix Inspection and Group Scheduling

Hooyeon Lee, Ashish Goel
Stanford University
haden.lee@cs.stanford.edu and ashishg@stanford.edu

Abstract

Consider an event organizer who is trying to schedule a group meeting. Availability of agents is unknown to the organizer a priori, but the organizer may have probability estimates on availability of each agent for each date/time option. The organizer can ask an agent to reveal her availability, but it causes inconvenience for the agent, and thus the organizer wishes to find an agreeable outcome at a minimum number of such queries. Motivated by this example, we study the Probabilistic Matrix Inspection problem in which we are given a matrix of Bernoulli random variables that are mutually independent, and the objective is to determine whether the matrix contains a column consisting only of 1's. We are allowed to inspect an arbitrary entry at unit cost, which reveals the realization of the entry, and we wish to find an inspection policy whose expected number of inspections is minimum. We first show that an intuitive greedy algorithm exists for 1-row and 1-column matrices, and we generalize this to design an algorithm that finds an optimal policy in polynomial time for the general case.

1 Introduction

Scheduling an event for a group of agents is a frustrating task; it tends to be tedious and time consuming. A typical scheduling process can be described as an iterative approval voting: An event organizer selects a candidate set of date/time options, and asks her invitees to respond with their availability. Given the responses, the organizer chooses an agreeable option and announces it, or she may repeat the process by proposing another set of date/time options if no feasible outcome is found. Naturally the organizer and her invitees wish to reach an agreement within a small number of iterations and proposed options – the more iterations and proposed options there are, the more laborious a scheduling process becomes.

There exist several software tools that are designed to help an event organizer handle a scheduling process more efficiently – one of the most well-known tools is Doodle [1]. In Doodle, an organizer can simply list as many date/time options as she likes, and each invitee is asked to respond with her availability. Essentially, invitees participate in an approval voting upon all proposed options. Hence if too many options are proposed, then invitees are given the burden of answering them all. This often leads to undesired behaviors of agents such as herding or procrastination, instead of honest, quick responses [Zou *et al.*, 2015]. On the other hand, if the organizer proposes too few options, there may not exist an agreeable outcome after all, which may result in another iteration of proposals and responses. In fact, surveys find that the most challenging part of group scheduling is due to "chasing people who do not answer" and "finding a suitable time." [2]

One way to re-design Doodle is to ask invitees availability questions in an adaptive manner. For instance, if a certain agent reports her unavailability for a specific date/time option, the system may skip asking other agents about the same option, but rather propose a different option that is more likely to work for everyone. Formally, we model a group scheduling process as the problem of inspecting a random matrix. Consider a matrix of mutually independent Bernoulli random variables where rows represent agents and columns represents the set of potential outcomes (such as date/time options). Given probability distribution of the matrix, the organizer can "inspect" an entry of the matrix at unit cost to know of the realization of it, and wishes to determine with certainty whether the matrix contains a column consisting only of 1's at minimum number of inspections possible. In our example, this corresponds to querying an agent for her availability, and seeking an agreeable outcome. By abstracting away the details of group scheduling, we are left with a mathematical model that represents the scheduling process, and in this work we study the properties of an optimal inspection policy and design an algorithm that finds it in polynomial time.

2 Related Work

Group scheduling is of tremendous practical importance, and much research has been devoted to it. Jennings et al. proposed the design of an agent-based meeting scheduling system, in which an autonomous agent negotiates with other agents on behalf of its human user [Jennings *et al.*, 1995].

[1] http://www.doodle.com

[2] http://en.blog.doodle.com/2012/07/26/new-findings-a-small-number-of-initiators-organize-most-of-the-meetings/

The emphasis was on the system description, rather than on theoretical or empirical results. Somewhat more formal work was done by Sen and Durfee, which focused on heuristic negotiation-strategies for group scheduling [Sen and Durfee, 1998]. Ephrati et al. tackled incentive issues, adopting a game-theoretic approach [Ephrati *et al.*, 1994]. They proposed three monetary-based meeting systems, in which invitees bid their preferences using monetary "points". They extended the Vickrey-Clarke-Groves mechanism, preventing manipulative behavior by the agents, while assuming that the host has an access to the calendars of invitees – the authors called this the "open calendar" system. In our work we assume that agents are non-strategic and that the event organizer has no access to calendars of invitees, but rather has probability estimates on availability of agents.

All of the above directions are relevant to the general problem of group scheduling, but not directly to this paper. The most closely related work of which we are aware is the "Batched Doodle" problem [Lee and Shoham, 2014]. Lee and Shoham studied the "Batched Doodle" problem which shares the same input (a matrix of mutually independent Bernoulli random variables) and the same objective (to determine whether it contains a column consisting only of 1's) as our work. However, in the Batched Doodle problem, an organizer is allowed to query agents for a subset of date/time options which corresponds to a submatrix consisting of a subset of columns; essentially each operation of inspection is one iteration of polling on a subset of outcomes. A solution to the Batched Doodle problem is an ordered partition of the columns, and its cost is measured by the (expected) number of iterations (called "Time") and queries incurred during the process (called "Inconvenience"). Lee and Shoham show that there exists an efficient algorithm for finding an optimal partition for a fairly large class of cost functions that aggregate Time and Inconvenience (for instance, a linear combination of the two). The main difference between this work and the work of Lee and Shoham lies in the solution domain (a unit operation on a group of columns versus entries) and the cost model (a function of Time and Inconvenience versus the number of inspections). The Batched Doodle generalizes Doodle by allowing columns-by-columns inspections, and we further generalize it by allowing entry-by-entry inspections. This generalization allows our model to be more applicable than that of Lee and Shoham.[3]

Lastly, in our work, we assume that the event organizer is given probability estimates on availability of agents, but it is not clear how one can obtain such probabilities. Probability estimation is an interesting and challenging research question on its own, and we do not attempt to solve the question in this work. However, we provide several plausible methods for estimating the probabilities in the context of group scheduling, which can enable our model and algorithm to be deployed as a real-world application in the future. In the psychology literature, Mann et al. found that cultural differences between the Western, individualistic countries (such as the United States) and the Eastern, collectivistic countries (such as China and Japan) lead to different behaviors of respondents when it comes to a group-decision making process [Mann *et al.*, 1998]. More recently, Reinecke et al. analyzed more than 1.5 million Doodle date/time polls from 211 countries, and confirmed similar findings regarding time perception and group's behavior [Reinecke *et al.*, 2013]. Among others, they found that "in comparison to predominantly individualist societies, poll participants from collectivist countries respond earlier, agree to fewer options but find more consensus," which agrees with the findings of Mann et al. Besides the cultural differences, Doodle's own surveys on event scheduling found that people tend to respond to the scheduling surveys on Mondays, while Monday is the least popular day for having a meeting. [4] We believe that these studies and findings can be used to design a reasonable estimator for availability of agents, by utilizing the features that are known to be crucial – such as demographics of the group and purpose of the event being scheduled. Recent work by Zou et al. analyzed over 340,000 Doodle polls data to study behavioral patterns of the users, and they were able to identify response functions that match the response patterns observed in the real data [Zou *et al.*, 2015]. We believe that a similar approach can be taken to tackle the problem of probability estimation in the context of group scheduling.

3 Formal Model

In this section we introduce mathematical notation we use in this paper, and formally define the Probabilistic Matrix Inspection Problem, followed by an example to help the reader understand the setup.

3.1 Notation and Definitions

Definition 1 (Feasibility)**.** Let A be a matrix whose entries are from $\{0, 1\}$, and refer to the entry of A at row r and column c as $a_{r,c}$. We say that a column c of A is *feasible* if it consists only of 1's (otherwise it is *infeasible*). We say that A is *feasible* if it contains at least one feasible column (otherwise it is *infeasible*).

In the Probabilistic Matrix Inspection Problem, we do not know of the values of the entries of A, as they are Bernoulli random variables, but we know of probability distribution of each entry. An input to the problem is this probability distribution.

Definition 2 (Input instance)**.** An input instance of the Probabilistic Matrix Inspection problem is a pair of matrices (A, P_A) of size n by m where A is a matrix of Bernoulli random variables and P_A is the probability matrix associated with it. We denote an entry of A as $a_{r,c}$ and of P_A as $p_{r,c}$. Each entry $a_{r,c}$ of A is a Bernoulli random variable, and $p_{r,c}$ is the probability of success for $a_{r,c}$ (i.e., $p_{r,c} = \mathbf{P}[a_{r,c} = 1]$). We define $s_c = \prod_{r=1}^{n} p_{r,c}$ to denote the probability that column c is feasible (probability of success for column c).

[3] It is worth noting that both this paper and the work of Lee and Shoham consider an optimization problem under probabilistic assumptions, but we are not aware of relevant work from the stochastic optimization literature.

[4] http://en.blog.doodle.com/2012/05/23/mondays-for-planning-busy-weekends/

Throughout this work we will assume that the set of random variables $\{a_{r,c}\}$ are mutually independent. This assumption is crucial to our technical results because it allows to compute (in polynomial time) the probability of a specific realization of A conditioning on the event in which some entries of A have already been realized. Without this assumption, it is unclear how one can compute such probabilities without having an access to the joint probability distribution over all realizations of A (whose size is exponential in the size of A).

We define an "inspection" as an operation that can be performed on A. One can inspect an arbitrary entry $a_{r,c}$ of A at unit cost, so as to know of the realization of the random variable. In group scheduling, an inspection corresponds to querying an agent about her availability for a certain outcome. The objective of the problem is to determine whether A is feasible or not, with minimum (expected) number of inspections possible. The expectation is with respect to the probability distribution specified by P_A.

Because we are interested in determining feasibility of A with minimum number of inspections, there are certain "unnecessary inspections" that an optimal strategy must avoid. For instance, if a certain entry $a_{r,c}$ is found to be unsuccessful (i.e., $a_{r,c} = 0$ is realized), then there is no need to inspect any other entry from the same column because the column is already known to be infeasible. Similarly, if a certain column is found to be feasible (which implies A is feasible) or if all columns are found to be infeasible (which implies A is infeasible), then there is no need to inspect any other entries of the matrix. Lastly, if $p_{r,c} = 0$ or $p_{r,c} = 1$, then there is no need to inspect the entry $a_{r,c}$ because we already know its realization with probability 1. Therefore, without loss of generality, we will assume that $p_{r,c} \in (0, 1)$ (i.e., $p_{r,c} \neq 0, 1$) in this work.

Let us define what constitutes a solution to the problem.

Definition 3 (Inspection policy)**.** Given an input instance (A, P_A), a solution is any permutation of the entries of A, and we call it an "inspection policy" (or simply, a "policy").

The interpretation of a permutation is as follows. The entries of A will be inspected in order specified by the permutation. After each inspection, if A is found to be feasible or infeasible, the inspection process ends. Otherwise, it continues inspecting the entries as specified, but it will not make any unnecessary inspections as mentioned earlier.

If π is a permutation of the entries of A, we write $C(\pi)$ to denote the number of inspections performed by π. $C(\pi)$ is a random variable whose probability distribution is determined by P_A. We are interested in finding an optimal policy which minimizes the expected number of inspections, $\mathbf{E}[C(\pi)]$. Note that there are $(nm)!$ permutations of the entries of A, and therefore exhaustive search for an optimal permutation will not produce an efficient algorithm.

3.2 Example

Consider a 2-by-2 matrix A of Bernoulli random variables whose probability of success is given by P_A as follows. In group scheduling, this corresponds to two agents and a set of two date/time options being considered.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}, P_A = \begin{bmatrix} 0.6 & 0.7 \\ 0.9 & 0.8 \end{bmatrix}$$

Let us consider an inspection policy π which inspects the entries of A column-by-column while inspecting them from top to bottom within a column:

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ a_{1,1} & a_{2,1} & a_{1,2} & a_{2,2} \end{pmatrix}.$$

Suppose that the realization of A happens to be the identity matrix of size 2 (i.e., $a_{1,1} = a_{2,2} = 1$ and $a_{2,1} = a_{1,2} = 0$). If we use π, it will first inspect $a_{1,1}$ and learn its realization. Since $a_{1,1} = 1$, it will inspect $a_{2,1}$ next only to find that column 1 is infeasible after all. It will then inspect $a_{1,2}$ and learn that column 2 is also infeasible, which implies that A is infeasible. At this point, the inspection process terminates without inspecting $a_{2,2}$. This specific realization of A happens with probability $p_{1,1}(1-p_{2,1})(1-p_{1,2})p_{2,2}$, and yields $C(\pi) = 3$ because 3 inspections would occur. If the realization of A happens to be the null matrix (i.e., all entries are 0's), then π would only inspect $a_{1,1}$ and $a_{1,2}$ but skip $a_{2,1}$ and $a_{2,2}$. In this manner one can consider all $2^{2\cdot 2} = 16$ possible realizations of A, and compute $\mathbf{E}[C(\pi)]$ in this example.

Another way to compute $\mathbf{E}[C(\pi)]$ is by de-coupling $C(\pi)$ into two random variables $N(\pi, c)$ with $c \in \{1, 2\}$ where $N(\pi, c)$ denotes the number of inspections (on column c) performed by π conditioning on the event that (at least one element of) column c is inspected. We can efficiently compute these: $\mathbf{E}[N(\pi, c)] = 1 \cdot \mathbf{P}[a_{1,c} = 0] + 2 \cdot \mathbf{P}[a_{1,c} = 1]$ for $c \in \{1, 2\}$. To express $\mathbf{E}[C(\pi)]$ in $N(\pi, c)$'s, we need to take conditional probability into account, as column 2 is inspected only if column 1 is infeasible: $\mathbf{E}[C(\pi)] = \mathbf{E}[N(\pi, 1)] + \mathbf{E}[N(\pi, 2)](1 - s_1) = 2.382$. Recall that s_c is the probability of success for column c.

4 Technical Results

We first consider two special cases (1-row or 1-column matrices) of the Probabilistic Matrix Inspection problem, which admit intuitive, greedy algorithms. We then discuss a couple of interesting properties of an optimal inspection policy, which leads to our main result and algorithm.

4.1 1-Row Matrix and 1-Column Matrix

Let us first consider the case where an input matrix A has only one row (i.e., $n = 1$). In this case it is natural to inspect entries with largest probability first because we can stop as soon as we find an entry whose value is 1 – which makes its column and A feasible. This intuition is exactly what an optimal policy should do in the single row case.

Lemma 1 (1-Row Matrix)**.** *When $n = 1$, an inspection policy π is optimal if and only if it inspects the entries in non-increasing order of their associated probabilities.*

Proof. Without loss of generality let us assume that a policy π inspects the entries in increasing order of their column index; that is, $\pi(i) = a_{1,i}$. Recall that $C(\pi)$ is a random variable that denotes the number of inspections π incurs. We can express the expectation of $C(\pi)$ in terms of $p_{1,c}$'s as follows:

$$\mathbf{E}\left[C(\pi)\right] = m\left(\prod_{k=1}^{m-1}(1-p_{1,k})\right) + \sum_{j=1}^{m-1} j\left(p_j \prod_{k=1}^{j-1}(1-p_{1,k})\right). \quad (1)$$

Suppose that there exists some c^* such that $p_{1,c^*} < p_{1,c^*+1}$ (if no such c^* exists, then π is an inspection policy that inspects the entries in non-increasing order of probabilities). Let π' be the same policy as π except we swap the order of a_{1,c^*} and a_{1,c^*+1}. That is, π' is defined as follows.

$$\pi'(j) = \begin{cases} \pi'(j) = \pi(c^*+1) & \text{if } j = c^* \\ \pi'(j) = \pi(c^*) & \text{if } j = c^*+1 \\ \pi'(j) = \pi(j) & \text{if } j \neq c^* \wedge j \neq c^*+1 \end{cases}$$

After expressing $\mathbf{E}[C(\pi)]$ and $\mathbf{E}[C(\pi')]$ as in Equation 1, one can re-arrange the terms to obtain the following:

$$\mathbf{E}\left[C(\pi)\right] - \mathbf{E}\left[C(\pi')\right] = \left(\prod_{j=1}^{c^*-1}(1-p_{1,j})\right)(p_{1,c^*+1} - p_{1,c^*}). \tag{2}$$

This quantity is positive if $p_{1,c^*+1} > p_{1,c^*}$ (recall that $p_{1,j} \in (0,1)$ for all j as mentioned in Section 3).

This proves the lemma because any policy that inspects an entry with smaller probability before another entry with higher probability is suboptimal, and therefore an optimal policy must inspect entries in non-increasing order of their associated probabilities. □

Although the proof of Lemma 1 is simple, it confirms correctness of our intuition. Equation 2 illustrates this intuition; conditioning on the event that the first $c^* - 1$ inspections fail (whose probability is the product term in Equation 2), the difference $\mathbf{E}[C(\pi)] - \mathbf{E}[C(\pi')]$ depends on the difference in the probabilities of success between the next-entry-to-be-inspected by π and π'.

We can also consider the case where an input matrix A has only one column. Intuitively, if we wish to minimize the expected number of inspections, we must inspect entries with smallest probability first because we can stop as soon as we determine that A is infeasible. Lemma 2 formally states this intuition about optimal policy, and we omit a proof of it as it can be easily done by following the proof of Lemma 1.

Lemma 2 (1-Column Matrix)**.** *When $m = 1$, an inspection policy π is optimal if and only if it inspects the entries in non-decreasing order of their associated probabilities.*

4.2 Inspection of Entire Column

Another interesting property of an optimal inspection policy is that once it inspects the first entry of a column, then it must commit to it and continue inspecting the remaining entries of the column until feasibility of the column is determined. Otherwise, if the policy switches to another column too soon, then it is not optimal.

Theorem 1 (Optimality of inspecting entire column)**.** *Consider any inspection policy π. Without loss of generality, let us assume that for each column c, π inspects $a_{n,c}$ the last among n entries of the column. Let b_c be the index of π such that $\pi(b_c) = a_{n,c}$. Without loss of generality, assume $b_1 < b_2 < \cdots < b_m$ (we can do this by re-labeling the columns of A). If there is some column c^* such that $b_{c^*} > n \cdot c^*$, then π is not optimal.*

Proof. First, note that $b_c \geq n \cdot c$ for all c because we assumed $b_1 < b_2 < \cdots < b_m$, and therefore the entries of previous columns must appear before the last entry of each column.

Let π be an inspection policy being considered in the theorem for which there exists some c with $b_c > n \cdot c$. Let us construct a different inspection policy π'. First, π' inspects all entries of column 1 in the same order π does. Then, π' inspects all entries of column 2 in the same order π does, and so on. In particular, π' inspects all entries of a column before inspecting another column, while preserving the original ordering of the entries within each column that is given by π. We will show that $\mathbf{E}[C(\pi')] < \mathbf{E}[C(\pi)]$.

Let us define a set of new random variables which can be used to express $C(\cdot)$, as we did in Section 3.2 when analyzing an example. Recall that $s_c = \prod_{r=1}^{n} a_{r,c}$ is the probability of success for column c. Let $N(\pi, c)$ ($N(\pi', c)$, respectively) be a random variable that denotes the number of entries of column c that is inspected by π (by π', respectively), conditioning on the event that column c is inspected (i.e., when the previous $c-1$ columns are infeasible). We can then express $\mathbf{E}[C(\pi)]$ and $\mathbf{E}[C(\pi')]$ as follows:

$$\mathbf{E}[C(\pi)] = \sum_{c=1}^{m} \mathbf{E}[N(\pi, c)] \left(\prod_{k=1}^{c-1}(1-s_c)\right) \tag{3}$$

and

$$\mathbf{E}[C(\pi')] = \sum_{c=1}^{m} \mathbf{E}[N(\pi', c)] \left(\prod_{k=1}^{c-1}(1-s_c)\right). \tag{4}$$

To prove the theorem we will first show that for any realization of A, $N(\pi, c) \geq N(\pi', c)$ holds for all c; this immediately implies $\mathbf{E}[C(\pi)] \geq \mathbf{E}[C(\pi')]$. We will then show that there exists at least one realization of A such that for some column c' the strict inequality $N(\pi, c') > N(\pi', c')$ holds. These two statements together imply that $\mathbf{E}[C(\pi)] > \mathbf{E}[C(\pi')]$.

Consider any realization of A with the condition that the first $m-1$ columns are infeasible (recall that m is the number of columns of A). Then $N(\pi, c) = N(\pi', c)$ for all c regardless of feasibility of column m. To see why, both π and π' would inspect the same set of entries in each of the first $m-1$ columns in the same order until the column is determined to be infeasible, and therefore $N(\pi, c) = N(\pi', c)$ if $c < m$. If column m is feasible, then both π and π' would inspect all n entries of it, and thus we have $N(\pi, m) = N(\pi', m) = n$. Otherwise, if column m is also infeasible (in which case A is infeasible), then π and π' would inspect the same set of entries of column m in the same order until the first infeasible entry of the column is found. Therefore if the first $m-1$ columns are infeasible we have $N(\pi, c) \geq N(\pi', c)$ for all c.

Now consider any realization of A with the condition that at least one of the first $m-1$ columns is feasible. Let c' be the smallest index of feasible columns of A. Because the columns from 1 to $c'-1$ are infeasible, $N(\pi, c) = N(\pi', c)$ for all $c < c'$ for the same reason we stated earlier for the other case. Since c' is feasible, $N(\pi, c') = N(\pi', c') = n$ as both policies would inspect all n entries of c'. By our construction of π' it is clear that $N(\pi', c) = 0$ for all $c > c'$;

therefore we have $N(\pi, c) \geq N(\pi', c)$ for all $c > c'$. In summary $N(\pi, c) \geq N(\pi', c)$ holds for all c in this case as well.

So far we proved the first claim we stated earlier: for all realizations of A, we have $N(\pi, c) \geq N(\pi', c)$ for all c. Let us now prove the second claim. Let c^* be the smallest index c of columns such that $b_c > nc$ (note that $c^* < m$ because $b_m = nm$ by definition). Consider any realization of A with the condition that the first $c^* - 1$ columns are infeasible and column c^* is feasible (feasibility of other columns do not matter). Using the same arguments we used earlier, we can show that $N(\pi, c) = N(\pi', c)$ for all $c < c^*$, that $N(\pi, c^*) = N(\pi', c^*) = n$, and that $N(\pi', c) = 0$ for all $c > c^*$. However, because $b_{c^*} > n \cdot c^*$, there is at least one entry $a_{r',c'}$ with $c' > c^*$ which appears before b_{n,c^*} in π (otherwise, if no such entry exists, then b_{c^*} would be equal to $n \cdot c^*$). This implies that there exists some c' with $c' > c^*$ such that $N(\pi, c') > 0$. This proves the second claim that for some realization of A, there is some column c' for which $N(\pi, c') > N(\pi', c')$, and together with the first claim we proved earlier, this implies that $\mathbf{E}[C(\pi)] > \mathbf{E}[C(\pi')]$.

This proves the theorem: Any policy that does not inspect all entries of a column consecutively is suboptimal. □

By Theorem 1, when seeking an optimal policy, it is sufficient to consider the set of policies that inspect an entire column before committing to another column. Lemma 2 hints that one should inspect the entries of each column in increasing order of probabilities, and this is what we prove next.

4.3 Optimal Ordering within Column

Lemma 2 states that an optimal policy must inspect the entries in increasing order of their probability of success, if A is a 1-column matrix. This argument can be generalized to the case where there is more than one column: If an optimal policy is to inspect an entry of some column c, it must inspect the entry with smallest probability of success first.

Theorem 2 (Optimal ordering within column)**.** *Consider any inspection policy π. If there exist two entries $a_{r_1,c}$ and $a_{r_2,c}$ from the same column such that $a_{r_1,c}$ appears before $a_{r_2,c}$ in π and $p_{r_1,c} > p_{r_2,c}$, then π is not optimal. In other words, when restricted to each column, an optimal policy must inspect the entries of the column in non-decreasing order of probabilities.*

Proof. Let π be an inspection policy being considered in the theorem. Because of Theorem 1 we can assume, without loss of generality, that π inspects all entries of column 1, followed by column 2, and so on. Further let us assume that π inspects the entries of each column in increasing order of their row index (we can do so by re-labeling the indices of entries). Precisely, $\pi(r + n(c-1)) = a_{r,c}$ defines π. Let p_{r,c^*} and p_{r+1,c^*} be the entries with $p_{r,c^*} > p_{r+1,c^*}$. Let us consider a different inspection policy π' that is the same as π except that π' inspects p_{r+1,c^*} before p_{r,c^*}, by swapping the ordering of them.

$$\pi'(j) = \begin{cases} \pi'(j) = a_{r+1,c^*} & \text{if } \pi(j) = a_{r,c^*} \\ \pi'(j) = a_{r,c^*} & \text{if } \pi(j) = a_{r+1,c^*} \\ \pi'(j) = \pi(j) & \text{otherwise} \end{cases}$$

We claim that $\mathbf{E}[C(\pi')] < \mathbf{E}[C(\pi)]$, which implies that π is not optimal.

Let us define new random variables $N(\pi, c)$ and $N(\pi', c)$ as we did in our proof of Theorem 1 (i.e., the number of inspections performed by the respective policy on column c, conditioning on the event that the column is inspected). Then we can express $\mathbf{E}[C(\pi)]$ and $\mathbf{E}[C(\pi')]$ in terms of the new random variables and s_c's as we did in Equations 3 and 4.

Observe that $N(\pi, c) = N(\pi', c)$ for any realization of A if $c \neq c^*$. To see this, first note that column c would not be inspected by π or by π' if any of the previous columns (that is, columns 1 through $c-1$) is found to be feasible, in which case $N(\pi, c) = N(\pi', c) = 0$. Otherwise, if column c is inspected, both policies would inspect the entries of c in the very same order, so $N(\pi, c) = N(\pi', c)$ must hold. Therefore we conclude that $\mathbf{E}[N(\pi, c)] = \mathbf{E}[N(\pi', c)]$ when $c \neq c^*$.

We will now show that $\mathbf{E}[N(\pi, c^*)] > \mathbf{E}[N(\pi', c^*)]$ holds. This immediately implies $\mathbf{E}[C(\pi)] > \mathbf{E}[C(\pi')]$ due to Equations 3 and 4. Let us express $\mathbf{E}[N(\pi, c^*)]$ in terms of p_{r,c^*}'s.

$$\mathbf{E}[N(\pi, c^*)] = n \left(\prod_{k=1}^{n-1} p_{k,c^*} \right) + \sum_{j=1}^{n-1} j(1 - p_{j,c^*}) \left(\prod_{k=1}^{j-1} p_{k,c^*} \right)$$

Note that the event $N(\pi, c^*) = j$ occurs if the first $j-1$ entries are feasible while the j-th entry is not feasible when $j < n$, and $N(\pi, c^*) = n$ occurs if the first $n-1$ entries are feasible (but the n-th entry's feasibility does not matter).

We can express $\mathbf{E}[N(\pi', c^*)]$ in a similar manner, and simplify $\mathbf{E}[N(\pi, c^*)] - \mathbf{E}[N(\pi', c^*)]$ as follows:

$$\mathbf{E}[N(\pi, c^*)] - \mathbf{E}[N(\pi', c^*)] = r \left(\prod_{k=1}^{r-1} p_{k,c^*} \right) (p_{r,c^*} - p_{r+1,c^*}).$$

The quantity above is positive if $p_{r,c^*} > p_{r+1,c^*}$, which is the assumption we began with. This proves the theorem. □

Theorems 1 and 2 together tell us that in order to find an optimal policy we only need to decide the ordering of the columns. There are still $m!$ orderings of columns, and an exhaustive search algorithm would not be efficient. As we were able to generalize Lemma 2 to Theorem 2 by generalizing the optimal solution for 1-column case, it would be natural to consider generalizing Lemma 1 in a similar manner.

This idea leads to the following greedy algorithm: First we sort columns by their probability of success ($s_c = \prod_{r=1}^{n} p_{r,c}$) in decreasing order, and inspect the entries of each column in increasing order of their associated probabilities. However, as the following example shows, this algorithm is suboptimal.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}, P_A = \begin{bmatrix} 0.4459 & 0.2262 \\ 0.4459 & 0.8114 \end{bmatrix}$$

Here we have $s_1 = 0.199$ and $s_2 = 0.184$, and the greedy algorithm would produce $\pi = (a_{1,1} \quad a_{2,1} \quad a_{1,2} \quad a_{2,2})$. Its expected cost, $\mathbf{E}[C(\pi)]$, is 2.428, but if we inspect the second column first, then the expected cost is 2.407 which is optimal in this example. One can consider another greedy algorithm which inspects the columns in increasing order of their expected number of inspections (within column), but this algorithm turns out to be suboptimal as well.

4.4 Main Result and Algorithm

Let us present the main result that leads to an efficient algorithm for finding an optimal inspection policy.

Theorem 3. *Let s_c be the probability of success for column c as before. Let μ_c be the expected number of inspections that column c incurs if its entries are inspected in increasing order of their probability of success, conditioning on the event that column c is inspected and infeasible. An optimal policy must be a column-by-column policy (due to Theorem 1), must inspect the entries of each column in non-decreasing order of probabilities (due to Theorem 2), and must inspect the columns in non-decreasing order of $\mu_c(1-s_c)/s_c$.*

Proof. Consider a column-by-column inspection policy π which inspects the column 1 through m in increasing order of their index (we can assume this without loss of generality by re-labeling columns).

As before, let $N(\pi, c)$ be a random variable that denotes the number of inspections performed by π on column c, conditioning on the event that column c is inspected. Then we can express $\mathbf{E}[N(\pi, c)]$ in terms of s_c and μ_c as follows.

$$\mathbf{E}[N(\pi, c)] = s_c \cdot n + (1 - s_c) \cdot \mu_c \tag{5}$$

This equation holds because if the column is feasible (with probability s_c), it would require n inspections, but if it is not (with probability $1 - s_c$), it would require μ_c inspections in expectation. The equation above simply considers these two events, and calculates the expected value of $N(\pi, c)$.

Suppose that there is some column c^* such that $\mu_{c^*}(1 - s_{c^*})/s_{c^*} > \mu_{c^*+1}(1 - s_{c^*+1})/s_{c^*+1}$. Because π inspects column c^* before column $c^* + 1$, it would not be inspecting the columns in increasing order of $\mu_c(1 - s_c)/s_c$. Consider a different inspection policy π' which inspects the columns in the same order as π except that π' inspects column $c^* + 1$ before c^* by swapping the inspection ordering of the two. We can relate $N(\pi, \cdot)$ to $N(\pi', \cdot)$ as follows.

$$N(\pi', c) = \begin{cases} N(\pi, c^* + 1) & \text{if } c = c^* \\ N(\pi, c^*) & \text{if } c = c^* + 1 \\ N(\pi, c) & \text{otherwise} \end{cases}$$

As we did in proofs of Theorems 1 and 2, we can use Equations 3 and 4, and simplify $\mathbf{E}[C(\pi)] - \mathbf{E}[C(\pi')]$ as follows.

$$\mathbf{E}[C(\pi)] - \mathbf{E}[C(\pi')] = \left(\frac{\mathbf{E}[N(\pi, c^*)]}{s_{c^*}} - \frac{\mathbf{E}[N(\pi, c^* + 1)]}{s_{c^*+1}} \right) \cdot \left(\prod_{j=1}^{c^*-1} (1 - s_j) \right) s_{c^*} s_{c^*+1} \tag{6}$$

The quantity in Equation 6 is positive if the difference of the weighted expected values (in the first parentheses) are positive. Using Equation 5 we obtain the following inequality.

$$\frac{\mathbf{E}[N(\pi, c^*)]}{s_{c^*}} > \frac{\mathbf{E}[N(\pi, c^* + 1)]}{s_{c^*+1}}$$
$$\Leftrightarrow \mu_{c^*}(1 - s_{c^*})/s_{c^*} > \mu_{c^*+1}(1 - s_{c^*+1})/s_{c^*+1}$$

By definition of c^*, the second inequality above holds, which implies $\mathbf{E}[C(\pi)] > \mathbf{E}[C(\pi')]$. Therefore, an optimal inspection policy must inspect the columns in non-decreasing order of $\mu_c(1 - s_c)/s_c$. □

Because there is a unique ordering of columns if we sort them by $\mu_c(1 - s_c)/s_c$ (up to ties), Theorem 3 leads to the following algorithm: We inspect the columns in increasing order of $\mu_c(1 - s_c)/s_c$, and in each column, we inspect the entries of it in increasing order of probabilities. This algorithm can easily be implemented to run in polynomial time.

5 Discussion and Future Work

In this work we defined the Probabilistic Matrix Inspection problem motivated by group scheduling and Doodle. We first considered two special cases, and discovered interesting properties of an optimal inspection policy which agree with our intuition. We then generalized our findings to design an efficient algorithm to solve the general case, and along the way we showed that two natural greedy algorithms fail to find an optimal solution. While we believe that our technical results make a great starting point for studying and optimizing a group scheduling process, there remain several open problems and future work to be done.

As we discussed in Section 2, our model and algorithm rely on the assumption that probability estimates on availability of agents are available. We suggested several ideas motivated by previous work in the literature, but it will be important to deploy such ideas into a system, and integrate it with our algorithm. From a theoretical perspective, there remain several open problems. While we assumed that an inspection can be performed on a single entry at unit cost, one can generalize the cost model by allowing an inspection of any subset of entries whose cost depends on, for example, the number of entries being inspected. In the context of group scheduling, an inspection on many entries means querying multiple agents at the same time for one or many outcomes (but an inspection is not limited to the entries from the same column or row). This generalization is particularly useful when scheduling takes place in a hierarchical setting such as corporates. For instance the event organizer may feel that the cost of querying a supervisor is significantly different from that of querying a colleague.

Lastly, although we focused on relating our model to group scheduling, the Probabilistic Matrix Inspection problem has other applications. Finding the right childcare facility, for example, involves extensive inquiries as parents wish to gather more information about how they would handle certain situations, what benefits and environments they provide, and so on. Through advertisements or brochures parents may even be able to gauge the likelihood of a certain facility satisfying their needs. Yet they still need to inquire facilities for precise information, which can be modeled by our probabilistic matrix model where columns correspond to facilities and rows correspond to the needs of parents.

Acknowledgements

This work was funded in part by the National Science Foundation (under grant IIS-1347214), AFOSR MURI, and the Kwanjeong Educational Foundation. The authors also thank anonymous reviewers, Albert No, Ernest Ryu, and Minyong Lee for providing feedback on this paper.

References

[Ephrati *et al.*, 1994] E. Ephrati, G. Zlotkin, and J.S. Rosenschein. A non-manipulable meeting scheduling system. In *Proceedings of the 13th international workshop on distributed artificial intelligence*, pages 105–125, 1994.

[Jennings *et al.*, 1995] N. R. Jennings, A. J. Jackson, and London E Ns. Agent-based meeting scheduling: A design and implementation, 1995.

[Lee and Shoham, 2014] Hooyeon Lee and Yoav Shoham. Optimizing time and convenience in group scheduling. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1345–1346. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[Mann *et al.*, 1998] Leon Mann, Mark Radford, Paul Burnett, Steve Ford, Michael Bond, Kwok Leung, Hiyoshi Nakamura, Graham Vaughan, and Kuo-Shu Yang. Cross-cultural differences in self-reported decision-making style and confidence. *International Journal of Psychology*, 33(5):325–335, 1998.

[Reinecke *et al.*, 2013] Katharina Reinecke, Minh Khoa Nguyen, Abraham Bernstein, Michael Näf, and Krzysztof Z Gajos. Doodle around the world: Online scheduling behavior reflects cultural differences in time perception and group decision-making. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 45–54. ACM, 2013.

[Sen and Durfee, 1998] S. Sen and E.H. Durfee. A formal study of distributed meeting scheduling. *Group Decision and Negotiation*, 7(3):265–289, 1998.

[Zou *et al.*, 2015] James Zou, Reshef Meir, and David Parkes. Strategic voting behavior in doodle polls. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 464–472. ACM, 2015.