Static & Dynamic Appointment Scheduling with Stochastic Gradient Descent

Gary Cheng[1] and Kabir Chandrasekher[2] and Jean Walrand[1]

***Abstract*— This paper considers the optimization of static and dynamic appointments for a sequence of customers with random processing times and are served by a single agent. In the static case, the problem is to find the appointment times for the customers that minimize the expected weighted sum of idle time of the agent, waiting time of the customers, and overtime of the agent. The dynamic formulation is original. It limits cascading delays by using warning times, which are defined relative to the start of a previous job, to tell customers when to arrive for their job. We outline an algorithmic framework based on the notions of infinitesimal perturbation analysis (IPA) and stochastic gradient descent (SGD) that can be used to solve both problems. For the static case, we show that the SGD method converges to a global minimizer with no assumptions on the joint distribution of the processing durations. For the dynamic case, we prove that under certain conditions it outperforms the static scheme. We give empirical evidence of the efficacy of both approaches using both synthetic data as well as data collected from a year of elective surgeries at a local hospital. Our results suggest the validity of the dynamic formulation as a more cost-efficient solution than the static formulation to the appointment scheduling problem.**

I. Introduction

In a vast variety of situations, customers make appointments with an agent performing a specific service. The inherent tension in scheduling appointments arises from the following incentives: the agent would like to pack the schedule in order to minimize the time it spends idle, whereas the customer would like to spend as little time as possible waiting for the agent. In the ideal solution for the agent, customers may face significant delays if previous jobs run longer than expected. By contrast, in the ideal solution for the customer, the appointments will be too spaced out and the agent will face large amounts of idle time. Here, we consider the case with 1 agent and n jobs in a pre-specified order and aim to find a reasonable trade-off between these two incentives.

A. Our Contribution

We develop a sample-based stochastic gradient descent (SGD) algorithm that adjusts the appointment times based on observed task durations. We distinguish two situations:

1) **Static Appointments:** All appointment times are fixed prior to the start of the first jobs, and the times remain fixed throughout the day.
2) **Dynamic Appointments:** Appointment times are *not* fixed before-hand and can vary as the jobs progress.

Our main contribution is the development of an algorithmic framework that can be used as a solution to both of these situations. In the case of static appointments, we show that our algorithm converges to the optimal solution. Our dynamic formulation is original and is an attempt to mitigate the issue of cascading delays that occur when the long processing of a single job delays a series of subsequent jobs. We note that in our formulation the constraints of a single agent and jobs of pre-specified order is essential and relaxing these requirements requires non-trivial extension. We believe, however, based on the simplicity of our approach and the empirical evidence we provide that our framework could prove useful in the most general form of this problem.

B. Related Work

The static appointment scheduling problem is classic and has been extensively studied for decades. In its most general form, the problem is unwieldy and difficult to work with. To deal with this, many previous works add constraints in order to add more exploitable structure to the problem. One common way this is done is through the restriction of the distribution of service times. For example, Pegden and Rosenshine [9] considered the problem of minimizing customer waiting time and overall usage time in a continuous time, exponential service time setting. Wang [14] worked on the same problem with the assumption that service times were from a phase-type distribution i.e. exponential, coxian; with these assumptions, he was able to show that an optimal appointment can be found by solving a set of nonlinear equations. Bosch, Dietz, and Simeoni [12] consider customer waiting time and overtime in the problem formulation and demonstrated an optimal appointment schedule with i.i.d Erlang service times. However, constraining the service time distribution to be i.i.d and of a certain form is quite limiting and unrealistic in certain scenarios.

Constraining service times to be discrete was explored by Begen and Queyranne[2]. They proved that if the durations only take on integer values and the objective is L-convex, then the optimal solution can be found in polynomial time. A natural step to solving the continuous analog is to use discrete approximation of the continuous distribution. However, the runtime of this algorithm is polynomial in the number of discrete time intervals that the service times can take on, implying that the degree of precision for a discrete approximation of continuous time directly impacts

[1] Gary Cheng is at the Department of EECS, UC Berkeley, Berkeley, CA 94709, USA garycheng@berkeley.edu

[2]Kabir Chandrasekher is at the Department of Electrical Engineering, Stanford University, Palo Alto, CA, USA kabirc@stanford.edu

[1]Jean Walrand is at the Department of EECS, UC Berkeley, Berkeley, CA 94709, USA walrand@berkeley.edu

the runtime of the algorithm. We avoid this problem by working in the continuous time setting directly. Many other papers outline approaches that use general purpose stochastic optimization methods. Denton and Gupta [5] approached this problem using a stochastic linear program and developed distribution-independent upper bounds. They then use a sequential bounding algorithm to find ϵ-optimal solutions for a given static appointment problem. Zhang and Xie [17] used a stochastic gradient descent method to look at the multi-agent, multi-customer problem. They proved many nice features of their objective function, but were unable to show optimal convergence of their algorithm. Bai, Storer, and Tonkay [1] also used a SGD approach to empirically analyze cost savings in a more specific surgery operating room setting where added constraints were placed on hospital room capacity.

As mentioned before, it is essential to our framework that the jobs to be scheduled are pre-ordered. A large effort has been devoted in many works to find the optimal ordering of jobs. Denton, Viapiano, and Vogl [6] worked on the problem of job sequencing in the surgery operating room context. They introduce simple heuristics that can be used to determine a surgery ordering. Wang [15] showed that the optimal order of the exponentially distributed jobs can be found by solving a set of nonlinear equations. Other variations of the static appointment scheduling problem, with differing practical settings in mind, have also been studied. Kaandorp and Koole [8] considered the static appointment problem in a discrete time setting with exponential service times, where no-shows are allowed to occur; they showed under these conditions, a local search method finds the optimal solution. Zacharias and Pinedo [16] considered finding the optimal static appointment schedule with the assumption that patients had some probability of not showing. Similarly, Sabria and Daganzo [10] considered the problem where customers arrived early/late to their appointment according to some known distribution (e.g. Gumbel).

The work mentioned focuses on determining a fixed schedule at the beginning of the day. There has been some work on dynamic scheduling of customers which is related to our notification time scheduling problem. Wang [13] studies the problem of scheduling the $(n+1)$th customer after the schedule of the first n has been set and cannot be changed, where all service times are exponential. The scheduler tries to find which jobs the $(n+1)$th customer should be scheduled in between. One of the problems Zacharias and Pinedo [16] considers is the online problem of scheduling appointments while a set number of jobs scheduled statically beforehand are being executed. The scheduler optimizes to find the best appointment time of this new customer given that some part of the schedule has already been completed. These dynamic scheduling methods still have a similar structure to the static appointment counterpart; the appointment times are defined with respect to the beginning of the day. Our notification time formulation attempts to decouple this dependency by instead defining the appointment of one job to be relate the start time of a previous job.

C. Paper Organization

The rest of the paper is organized in the following manner. In section II, we formally define the static appointment time problem, describe our SGD algorithm, and prove that the aforementioned algorithm converges to the optimal solution. In section III, we carefully describe the dynamic extension of the appointment scheduling problem and outline our SGD algorithm for this case. Finally, we provide extensive empirical evidence corroborating our theoretical results as well as demonstrating the utility of these algorithms in section IV.

II. Static Appointment Scheduling

We begin by addressing the static appointment problem, formally defined as follows. An agent has an ordered set of n jobs that need to be completed in the order given; one job must be finished before the next job can begin. For $m = 1, \ldots, n$, the duration of job m is a continuous random variable X_m. We make no assumption on the dependence of these random variables. Customers are told their appointment time $\mathbf{a} = (a_1, \ldots, a_n)$ before any of the jobs begin; appointment times cannot be changed once they have been given out. Customers will arrive exactly at their appointment time, and the first customer will always start being be served at time 0. Thus, we can assume that $a_1 = 0$. Every customer must wait till all previous jobs are completed before being served. If a job completes and the next customer has not arrived yet, the agent must remain idle until the next customer arrives. The agent begins to incur overtime past a given time d. Customer waiting times, agent idle time, and agent overtime are penalized at a cost rate of α, β, and γ, respectively, to give us the wait cost, idle cost, and overtime cost; these parameters are all required to be nonnegative. The problem then is to find the appointment times $\mathbf{a} = (a_1, \ldots, a_m)$ that minimize the expected sum of idle cost, wait cost, and overtime cost.

For our problem, we assume there is an oracle that provides i.i.d samples from the joint distribution of job durations $\mathbf{X} = (X_1, \ldots, X_n)$. In practice these samples can be actual distributions observed during repeated instances of the sequence of processing of the jobs.

Let S_m be the start time and Y_m the end time of job m for $m = 1, \ldots, n$. Note that we do not write explicitly the dependency of S_m and Y_m on $\mathbf{a}$ and $\mathbf{X}$ for ease of notation. Then,

$$S_1 := 0; \tag{1}$$

$$S_m := \max(Y_{m-1}, a_m);\ m = 2, \ldots, n; \tag{2}$$

$$Y_m := S_m + X_m;\ m = 1 \ldots, n. \tag{3}$$

The wait cost C_W, idle cost C_I, overtime cost C_O, and total cost C are formally defined as follows. We omit the

dependency of C_W, C_I, C_O, and C on $\mathbf{a}$ and $\mathbf{X}$ for ease of notation. We define $(z)^+$ to mean $\max(z, 0)$.

$$C_W := \alpha \sum_{m=2}^{n} (Y_{m-1} - a_m)^+; \tag{4}$$

$$C_I := \beta \sum_{m=2}^{n} (a_m - Y_{m-1})^+; \tag{5}$$

$$C_O := \gamma (Y_n - d)^+; \tag{6}$$

$$C := C_W + C_I + C_O. \tag{7}$$

Our objective function is the expected value of the total cost, $\mathbb{E}_{\mathbf{X}}[C]$.

A. SGD for Static Appointment Scheduling

The static appointment SGD algorithm (**SA-SGD**) is as follows. Assume that the vector of appointment times is $\mathbf{a}^t$ at step t of the algorithm. One draws a new sample $\mathbf{X}^{t+1}$ of the random vector $\mathbf{X}$. One then calculates the sample gradient $\nabla_{\mathbf{a}} C(\mathbf{a}^t, \mathbf{X}^{t+1})$ with respect to $\mathbf{a}$ and takes a step in the opposite direction of the gradient. That is,

$$\mathbf{a}^{t+1} = \mathbf{a}^t - \eta_t \nabla_{\mathbf{a}} C(\mathbf{a}^t, \mathbf{X}^{t+1}).$$

In this expression, $\eta_t > 0$ is the step size, and we set it to be $\eta_t = \mathcal{O}(1/\sqrt{t})$. This choice is justified in the next subsection.

The sample gradient $\nabla_{\mathbf{a}} C(\mathbf{a}^t, \mathbf{X}^{t+1})$ is calculated using Infinitesimal Perturbation Analysis (IPA) [7]. This is an analysis of how much cost would be added if a_m was slightly increased. The procedure is illustrated in Figure 1. This figure shows the processing of six jobs with their appointment times $\{a_1 = 0, a_2, \ldots, a_6\}$. We detail some cases that outline how the sample gradient is calculated under different circumstances. Case 1: increasing a_2 by $\epsilon \ll 1$ increases the idle time before job 2 but decreases the idle time after job 2 and does not affect waiting times nor the overtime. Thus, $\partial C/\partial a_2 = 0$. Case 2: increasing a_3 by $\epsilon \ll 1$ increases the idle time before job 3 and decreases the idle time after job 4. In addition, it increases the waiting time of job 4. Thus, $\partial C/\partial a_3 = \alpha$. Case 3: increasing a_6 and a_4 by $\epsilon \ll 1$ only decrease the waiting time for job 6, so $\partial C/\partial a_4 = \partial C/\partial a_6 = -\alpha$. Using similar observations and the fact one finds that $\partial C/\partial a_5 = \alpha + \beta + \gamma$. It is clear from these examples that it is straightforward to compute the sample gradient $\nabla_{\mathbf{a}} C(\mathbf{a}, \mathbf{X})$.

We will now define some notation which will be useful in finding the gradient in the general case. Jobs m that incur no wait cost are called heads of chains; from the examples outlined above, its clear that only heads of chains have the potential to affect the wait times of jobs before them. The jobs (not including job m) that are affected by perturbing a_m are said to be apart of job m's chain. Heads of chains will have an indicator $H_m = 1$ and P_m will count the number of jobs affected by a perturbation of a_m i.e. the number of other jobs in job m's chain. Only perturbing the last chain affects

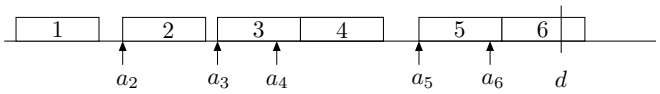

Fig. 1. A realization of the static appointment service process

the idle cost, so the indicator L_m will equal 1 for jobs in the last chain. The indicator O will equal 1 if the end time of the last job is greater than d. Using these indicators, we can observe that the gradient in general is:

$$\frac{\partial C}{\partial a_m} = H_m(\alpha P_m + \beta L_m + \gamma L_m O) - (1 - H_m)\alpha. \tag{8}$$

Because the sample cost function is differentiable almost surely and the sample cost function is lipshitz continuous, by the bounded convergence theorem [3], we know that the gradient of the expected cost function $\nabla_{\mathbf{a}} \mathbb{E}[C(\mathbf{a}, \mathbf{X})]$ is equal to the expectation of the gradient of the cost function $\mathbb{E}[\nabla_{\mathbf{a}} C(\mathbf{a}, \mathbf{X})]$. For this reason, it is justified that we use of the sample gradient in minimizing the expected cost in our problem.

B. Convergence to the Optimum for Static Scheduling

To prove the convergence of SGD, we begin by proving the cost function is convex.

Lemma 1. *C_O, C_W, C_I, C, and $\mathbb{E}[C]$ are all convex in $\mathbf{a}$*

Proof. By first showing Y_m is convex in $\mathbf{a}$ forall m, the others follow directly. $Y_1 = X_1$ is a constant function, which is trivially convex. Assume that Y_m is convex. Since Y_m is convex, so is $Y_{m+1} = \max(Y_m, a_{m+1}) + X_{m+1}$.

$C_O = \gamma \max(Y_n - d, 0)$ is convex because Y_n is convex in $\mathbf{a}$ and the function $\max(x - d, 0)$ is convex in x. $C_W = \alpha \sum_{m=2}^{n} \max(Y_{m-1} - a_m, 0)$ is a sum of convex function in $\mathbf{a}$ and is therefore convex. $C_I = \beta\,(Y_n - \sum_{i=1}^{n} X_m)$; this is an equivalent way of writing C_I because the amount of idle time is equal to the last completion time minus the total time spent processing jobs. From this expression, it is clear that C_I is convex. C is convex because it is the sum of convex functions. We also know that $\mathbb{E}[C]$, which is a convex combination over sample cost functions, is convex. □

The next theorem proves that static SGD algorithm converges to the optimal appointment time.

Theorem 1. *For $F := \mathbb{E}_X[C]$ convex and for constants D, $G^2 = (n\alpha + \beta + \gamma)\sqrt{n}$, it holds that $\mathbb{E}[\|\nabla_{\mathbf{a}} C(\mathbf{a}^t, \mathbf{X}^{t+1})\|_2] \leq G^2$ for all $\mathbf{a}$, and $\sup_{a,a'} \|a - a'\|_2 \leq D$. Consider SGD with step sizes $\eta_t = c/\sqrt{t}$ where $c > 0$ is a constant. Then for any $t > 1$, it holds that*

$$\mathbb{E}[F(a_t) - F(a^*)] \leq \left(\frac{D^2}{c} + cG^2\right) \frac{2 + \log(t)}{\sqrt{t}}$$

Proof. The proof of this is can be found in theorem 2 in Shamir and Zhang[11]. Convexity was proved in theorem

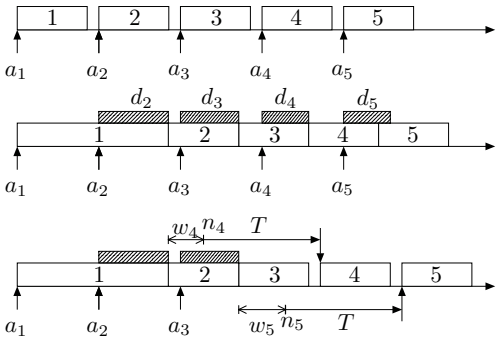

Fig. 2. Expected realization (top); Cascade of delays with static appointment times (middle); Limitation of cascade with notification times (bottom).

1. G^2 comes from the fact that the partial derivative with respect to a_m is bounded by the value $n\alpha + \beta + \gamma$. Each workday can be assumed to be bounded (e.g. 24 hours), so D can be chosen accordingly. □

This theorem requires that the sample gradient chosen at each step is in the sub-differential of the sample cost function. Our sample cost function is piecewise linear and therefore not differentiable at a finite number of positions; at those positions, we choose the sample gradient to be apart of the sub-differential.

While our analysis focuses on the case where there is one α wait cost penalty for all customers, there is no reason why we could not have different wait time penalties α_m for each individual customer. The optimality and convergence properties of the SGD hold even when we have a $\alpha = (\alpha_1, \ldots, \alpha_n)$ vector.

III. Dynamic Appointment Scheduling

A glaring issue that the optimum static appointments cannot fix is cascading delays. Figure 2 illustrates that issue: the expected schedule (top) and the appointment times of the customers. The middle part of the figure shows what happens if the first job takes longer than expected: it pushes back the execution of the subsequent jobs and causes the delays $d_2, \ldots, d_5$ for those customers. The bottom part of the figure illustrates the notification times scheme. Instead of giving a fixed appointment time a_4 to customer 4, one warns him at time n_4 that he should come at time $n_4 + T$. Here, T is the lead time that customers need to arrive at the agent. We assume that it is unrealistic to ask customers to arrive in a variable amount of time, so we fix T as a constant. The notification time n_4 is anchored at some time during the execution of job 2. Specifically, $n_4 = \min\{S_2 + w_4, Y_2\}$ where S_2 is the start time of job 2 and Y_2 is its completion time. w_4 is a parameter to be optimized, which we will call customer 4's warning time. As the figure indicates, if the execution of job 2 is postponed because of prior delays, as in this example, the appointment time of customer 4 slides automatically so that his waiting time does not increase. The situation is similar for customer 5.

In our discussion below, the job during which the notification of job j is anchored is defined to be $K(j)$. For instance, here, $K(4) = 2$. Note that some customers may get static appointment times. For instance, in the figure, this is the case for customers $1, 2, 3$. This may happen if the lead time T is so large that it is too late to warn those customers during the execution of previous jobs. In the extreme case where T is very small, one can warn a customer when the previous job completes and it is clear that the cost is then significantly smaller than with the best static appointments. Hence, one expects that if T is not very large compared to the processing times, the dynamic appointments will reduce the cost.

We formalize the notification time problem as follows. An agent is given a set of n jobs that need to be completed in the order given. Like before, $\mathbf{X} = (X_1, \ldots, X_n)$ is a random vector of the durations of each job. The SGD algorithm makes no assumptions on the joint distribution of these random variables; however, to prove some results later, we will require uncorrelated durations. Each job j will be warned during the execution of a previous job $1 \leq K(j) < j$. The first job will start at time 0 and will be represented by the mapping: $K(1) = -1$ and $a_1 = 0$. All other jobs after must have a mapping of $K(j) = i \geq 1$, and the notification time of those jobs are $n_j = \min(S_i + w_j, Y_i)$. Thus, the policy is parameterized by the constants $K(j)$ and w_j for $j = 1, \ldots, n$. Jobs j with a K mapping of 0 can have a $w_j \geq -T$; a negative warning time in this context simulates a static appointment at time $w_j + T$. For other K mappings, $w_j \geq 0$; since $S_{K(j)}$ is a random variable, this ensures that knowledge of unrealized, future events is not needed for a customer to be notified. The arrival time of customer j is then $R_j := n_j + T = \min(S_i + w_j, Y_i) + T$. If $K(j) = -1$, then $R_j = a_j$. Observe that if $j < i$, then it must be that $K(j) \leq K(i)$. Formally:

$$S_1 := R_1 = 0; \tag{9}$$

$$R_j := S_{K(j)} + \min(X_{K(j)}, w_j) + T; \tag{10}$$

$$S_j := \max(R_j, Y_{j-1}); \tag{11}$$

$$Y_j := S_j + X_j. \tag{12}$$

The wait cost, idle cost, overtime cost, and total cost are the following. For the sake of brevity, we will omit the dependency of the costs on $\mathbf{w}$ and $\mathbf{X}$ for the remainder of the analysis for the sake of brevity.

$$C_W := \alpha \sum_{j=2}^{n} (S_j - R_j)^+; \tag{13}$$

$$C_I := \beta \sum_{j=2}^{n} (S_j - Y_{j-1})^+; \tag{14}$$

$$C_O := \gamma (Y_n - d)^+; \tag{15}$$

$$C := C_W + C_I + C_O. \tag{16}$$

Unfortunately, in contrast with the static case, the cost is generally not convex in $\mathbf{w}$ even when the K vector is fixed. Consider two jobs, where $\Pr(X_1 = 2) = 1$, $T = 1$, $\alpha =$

$\beta > \varepsilon$, and $\gamma = 0$. A quick analysis shows that the slope of the cost function increase from α to β and then subsequently decreases to ε; a clear indicator of non-convexity.

A. Properties

In order to address the non-convexity of the problem, we simplify the problem a bit. We consider the simple K mapping defined as follows:

Definition 1 (Simple K mapping)**.** $K(1) = -1$ *and for all* $j > 1$, $K(j) = j - 1$.

This K mapping is of particular interest because it is empirically shown to perform very well. To prove some performance guarantees, we make the additional assumption that the overtime cost is not penalized $\gamma = 0$ and that the job durations are uncorrelated. These assumptions allow us to break the main objective into smaller ones. This property also highlights how cascades are avoided: the optimal notification time for the jth job only depends on the duration of the $j-1$ th job. If the $j \pm c$ th job went longer/shorter than expected, it wouldn't affect the cost incurred by the jth job. We summarize this with the following lemma, whose proof can be found in [FILL HERE].

Lemma 2 (Decoupling Principle)**.** *Assume the K-mapping used is simple and that there is no penalty for overtime. Then, within this restricted class, the optimal warning time can be found for each case separately.*

With lemma 2 and some algebra, we can obtain the second partial derivatives of the expected cost function. This is useful because we can show that for some distributions the cost function is convex. We now define the following, which will help us prove some more properties of our cost function:

Definition 2 (Optimal 2 case static appt.)**.** *Consider a schedule which only contains the* $j-1$*th and* j*th jobs.* $\hat{a}_j$*, which satisfies the condition* $\Pr(X_{j-1} \leq \hat{a}_j) = \alpha/(\alpha+\beta)$*, is the optimal static appointment time.*

Using lemma 2 in tandem with 2, we quickly form an estimate of the optimal warning time $w_j = (\hat{a}_j - T)^+$. If the lead time is shorter $\hat{a}_j$ for all j, then we can show that the simple K mapping notification time scheme outperforms the static scheme. Moreover, we can also show that no other K mapping can outperform the simple one. We summarize this result in the following theorem:

Theorem 2. *Let* $OPT_{dynamic}$*,* OPT_{static}*, and* OPT_{simple} *be the average cost incurred by the optimal dynamic appointment scheme, optimal static scheme, and optimal dynamic scheme restricted to the simple* K *mapping respectively. Assume no penalty for overtime, uncorrelated job durations, and a small enough lead time* $T < \hat{a}$*, then:*

$$OPT_{simple} = OPT_{dynamic} \leq OPT_{static}$$

The proof of theorem 2 can be found in [FILL HERE].

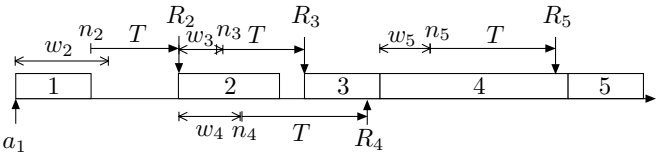

Fig. 3. A realization of the notification time service process. Customer 2 is notified after the end of job 1; therefore, moving back w_2 does not do anything. Moving back w_3 would increase the wait time of customer 4, but customer 5's waiting time is unaffected.

B. SGD for Dynamic Appointment Scheduling

Like the static variant, the dynamic algorithm uses SGD for optimizing the warning time parameters $\mathbf{w}$ that determine the notification times. The IPA calculation of the gradient is slightly different from the static case. Although the non-convexity of the problem means that the SGD scheme with any fixed K vector is no longer guaranteed to converge to a global minimum, SGD does converge to a critical point [4].

Let us first look at Figure 3 and examine some cases that outline how the sample gradient is calculated. Case 1: observe that w_2 takes place after the end of job 1. Consequently, increasing w_2 by $\epsilon \ll 1$ does not affect any outcome. Therefore, $\partial C/\partial w_2 = 0$. Case 2: if we increase w_3 by $\epsilon \ll 1$, then the idle time between jobs 2 and 3 increases. Customer 4 also has to wait longer than before, but customer 5 is unaffected because his notification time occurs during job 4, which has not happened yet. Therefore, $\partial C/\partial w_3 = \alpha + \beta$. Because customer 3 does not incur any wait cost, perturbing w_3 does change the start time of job 3 and thereby increases the waiting costs of the customers after it. Case 3: Increasing w_4 and w_5 decreases the wait time of the 4th and 5th customer respectively, so that $\partial C/\partial w_4 = \partial C/\partial w_5 = -\alpha$. Because customers 4 and 5 incur some waiting cost, increasing w_4 and w_5 can only decrease the waiting cost of those customers.

Now, looking at Figure 4, Case 4: we see that perturbing w_3 does not affect the idle costs or overtime costs; $\partial C/\partial w_3 = 0$. Case 5: perturbing w_4 does affect both idle and overtime cost; $\partial C/\partial w_4 = \beta + \gamma$. We can observe that the idle cost changes only if perturbing w_j changes the end time of the last job of the day.

To define the gradient in general, we define a new indicator random variable $Z_j = 1\{w_j \geq X_{K(j)}\}$; this indicator is useful for handling situations outlined in Case 1. As before, we use $H_j = 1$ if job j incurs no wait cost (the head of a chain) and P_j as the number of jobs (not including job j) affected by perturbing w_j (number of other jobs in the job j's chain). In particular, w_j affects the start time of job i if $j+1 \ldots i$ all incur a waiting cost and the ith customer has been warned before the start of the jth job, $K(i) < j$. Let L_j be the indicator that changing w_j affects the start time of the last job. Notice that perturbing w_j affects the overtime cost only if $L_j = 1$ and the last job ends after d. We will again use O as the indicator that the last job goes over d.

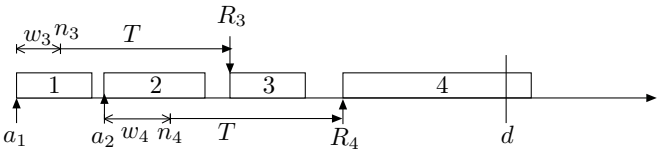

Fig. 4. Another realization of the notification time service process. Moving w_3 does not affect the total idle time because job 3 is not apart of the last chain, but moving back w_4 would increase total idle time. Overtime works similarly provided that the end time of the last job ends after d. Perturbing the last chain (e.g. job 4) is the only way that over time can increase.

The sample gradient is:

$$\frac{\partial C}{\partial w_j} = \begin{cases} H_j(\alpha P_j + \beta L_j + \gamma L_j O) - (1 - H_j)\alpha & Z_j = 0 \\ 0 & Z_j = 1 \end{cases}$$

As a heuristic, we generally set $\partial C / \partial w_j = \epsilon$ when $Z_j = 1$ for some $\varepsilon > 0$ to prevent the SGD from getting stuck in flat spots. A positive ε can also be thought of as a penalty for overestimating the duration of the $K(j)$th job.

There are three variants of the SGD algorithm: brute force K mapping optimization (**BK-NT-SGD**), fixed K mapping optimization (**FK-NT-SGD**), or an adjustable K mapping optimization (**AK-NT-SGD**). As the name suggests, the brute force K mapping optimization performs SGD on all valid K mappings. It calculates the empirical cost of the converged warning times for each K mapping and returns the lowest cost K mapping and warning time pair. While this takes care of the problem of K mapping, this search is over an exponential number of K mappings, so we also present two approximate versions of the brute force algorithm. The fixed K mapping version optimizes $\mathbf{w}$ for a fixed K mapping; the default choice of K mapping that we will use is the simple K mapping. The adjustable K mapping algorithm decrements $K(j)$ if w_j becomes smaller than δ and increments it if w_j becomes larger than the $100 - \zeta$th percentile of the random variable $X_{K(j)}$ for hyperparameters $\zeta, \delta \geq 0$. We will use the simple K mapping to initialize the adjustable K mapping algorithm.

IV. Empirical Results

In subsection IV-A we will begin with a toy example of 3 uniformly distributed jobs to demonstrate the importance of K mapping. Next, we evaluate the performance of our scheduling algorithms on a single queue of constructed discrete bimodal distribution that mimics what a true job duration may look like for various α, β, γ, and T. In subsection IV-B, we repeat the discrete bimodal experiment on a real hospital operating room schedule. We further demonstrate cost savings of the warning time algorithm against the static algorithm on over a years worth of real operating room schedules.

Recall that **SA-SGD**, **FK-NT-SGD**, **BK-NT-SGD**, and **AK-NT-SGD**, stand for the static, fixed K, brute K, and adjust K schemes respectively.

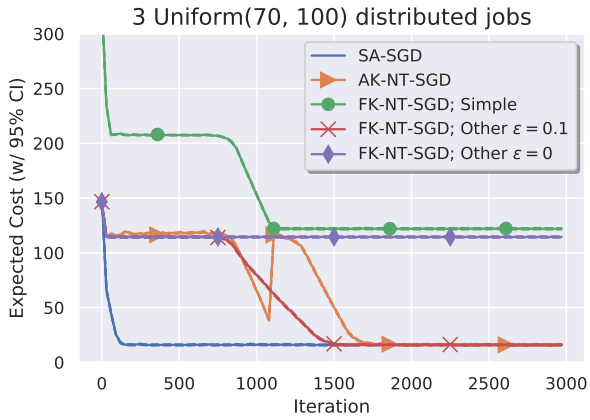

Fig. 5. The cost of the best warning time K mapping matches the static appointment scheme. The simple K mapping does poorly; the "Other" K mapping ($[-1, 1, 1]$) matches the static scheme. Choosing $\epsilon = 0.1$ leads to convergence. $T = 200$

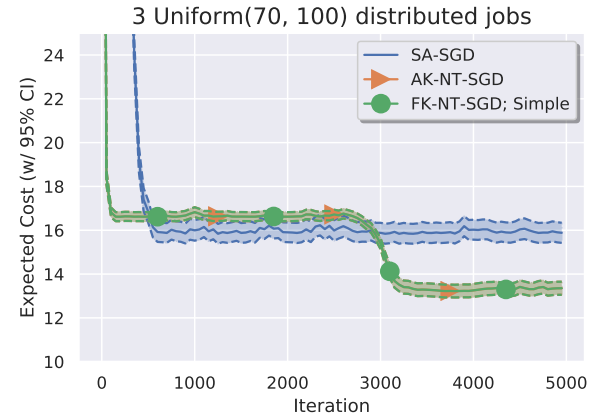

Fig. 6. The cost of the best warning time K mapping outperforms the static appointment scheme. The simple K mapping does the best. $T = 10$

A. Results on synthetic data

We begin with a simple example of 3 uniformly distributed jobs. We fix $\alpha = 1$, $\beta = 3$ and $\gamma = 1.5$ because we prioritize the hospital's time more than customer waiting time. We compare the static appointment cost against the warning time cost in figure 5 with lead time $T = 200$ and in figure 6 with a shorter lead time $T = 10$. We plot the average cost incurred by the schedules on 1000 sampled job duration vectors, $\mathbf{X}$, from the joint distribution. We also form a 95% confidence interval on the sample average cost.

In figure 5 because the lead time T is much larger than the maximum duration of the jobs, the simple K mapping is outperformed by the "Other" $[-1, 1, 1]$ K mapping. When the lead time is much shorter in figure 6 than the durations of the job, the simple K mapping outperforms the static scheme. This behavior matches the findings described in theorem 2; only when T is small enough is the simple K mapping guranteed to outperform the static scheme and other K mappings.

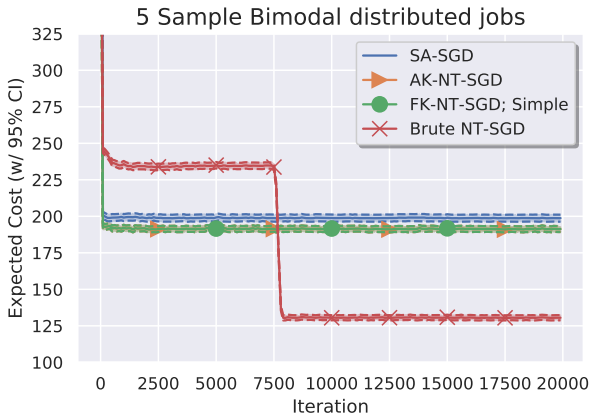

Fig. 7. The cost of the best warning time K mapping outperforms the static appointment scheme. The simple K mapping does the best. $T = 10$

α	β	γ	T	SA	FK-NT	BK-NT	AK-NT
1.0	3.0	1.5	40	196.7	**113.8**	**113.8**	**113.8**
1.0	3.0	1.5	80	196.7	225.8	**165.7**	**165.7**
1.0	3.0	1.5	120	196.7	555.7	**165.7**	196.0
3.0	1.0	1.5	40	269.0	**230.9**	**230.9**	**230.9**
3.0	1.0	1.5	80	269.0	246.9	**245.8**	248.2
3.0	1.0	1.5	120	269.0	339.7	**260.6**	**260.6**
1.0	1.5	3.0	40	187.5	**128.9**	**128.9**	**128.9**
1.0	1.5	3.0	80	187.5	202.4	**177.6**	**177.6**
1.0	1.5	3.0	120	187.5	511.1	**177.6**	211.0
1.0	1.0	1.0	40	140.5	**97.0**	**97.0**	**97.0**
1.0	1.0	1.0	80	140.5	**126.2**	**126.2**	139.9
1.0	1.0	1.0	120	140.5	249.5	**139.2**	**139.2**

Fig. 8. Best cost on 12 different α, β, γ, T combinations on 5 sample bimodal random variables after choosing the lowest cost solution for each of the 4 algorithms from 5 trials ($d = 420$).

α	β	γ	T	SA	FK-NT	BK-NT	AK-NT
1.0	3.0	1.5	30	119.9	122.7	**91.4**	100.7
1.0	3.0	1.5	60	119.9	299.6	**106.7**	106.8
1.0	3.0	1.5	90	119.9	700.5	**111.3**	130.2
3.0	1.0	1.5	30	132.5	**105.6**	**105.6**	125.2
3.0	1.0	1.5	60	132.5	150.7	**114.9**	164.3
3.0	1.0	1.5	90	132.5	327.7	**130.6**	149.3
1.0	1.5	3.0	30	92.3	82.8	**74.4**	**74.4**
1.0	1.5	3.0	60	92.3	167.1	**78.8**	85.1
1.0	1.5	3.0	90	92.3	498.3	**87.3**	87.7
1.0	1.0	1.0	30	76.9	65.1	**63.7**	64.1
1.0	1.0	1.0	60	76.9	116.9	**65.3**	74.5
1.0	1.0	1.0	90	76.9	275.1	**71.6**	74.3

Fig. 9. Best cost from 5 trials on 12 different α, β, γ, T combinations on 6 real operating room surgery duration random variables after choosing the lowest cost solution for each of the 4 algorithms from 5 trials ($d = 420$ minutes). T is realistic, ranging from 30 to 90 minutes.

Next we look at a schedule of 5 sample based bimodal distributed jobs. Each job distribution is created out of 100 samples from a bimodal distribution: 70% of the data points are generated from a $\mathcal{N}(60, 25)$ distribution and the other 30% of data is generated from a $\mathcal{N}(120, 25)$; negative samples are thrown out. We again choose the cost parameters $\alpha = 1$, $\beta = 3$ and $\gamma = 1.5$. The cost function is then evaluated on 1000 held out samples from the true bimodal distribution and a 95% confidence interval is also formed. This setup is more realistic because the bimodal distribution is similar to how jobs, such as surgeries, are distributed, and in practice, we would most likely only have access to a limited history of realizations. We plot the cost as a function of iteration of the SGD algorithm in figure 7.

It is clear at this point that the choices of α, β, γ, T have a large impact on how well the notification time scheme works. So for a variety of choices of α, β, γ, T, we calculate the optimal static appoint and run all three variants of notification time SGD on the 5 sample based bimodal distributed jobs. These results can be found in table 8. We run each of the algorithms 5 times from random initializations. We display the best performance for the algorithms. In each of these examples, even though the adjusting K map may not always find the best K mapping and warning time pair, it still generally performs better than the static appointment scheme.

B. Results on hospital operating room data

We look at one year's of operating room (OR) data from a real hospital to see how our algorithm leads to cost savings. This dataset was used to form an oracle. When we query the oracle for a duration of a surgery of a certain type, the oracle randomly selects a surgery of the same type in the data and returns the duration of that particular surgery.

We again test our algorithms for various choices of realistic α, β, γ, and T with 5 randomly initialized trials on one real operating room schedule from our data set; the best performing result from each algorithm can be found in table 9. In this experiment, the optimal K mapping outperforms the static cost. Even though many times AK-NT-SGD is unable to arrive at the brute force K mapping, it still is able to outperform static cost the simple K mapping.

Next, we use the dataset to form 345 real operating room schedules. These chosen schedules chosen only include types of surgeries that we have at least 20 previous duration observations and for schedules that were at least 3 surgeries long. We use the realistic parameters: $\alpha = 1$, $\beta = 3$, $\gamma = 1.5$, $T = 60$ min, and $d = 7$ hours $\times$ 60 min. For every schedule, we run the SA-SGD, AK-NT-SGD, and FK-NT-SGD algorithms each 3 times. For each of these OR schedules, we calculate the percent cost saved by using the best performing notification schedule (i.e. best of the returned AK-NT-SGD and FK-NT-SGD schedules) in place of the optimal static schedule. These results are shown in figure 10. The average cost savings is 9.8%; the median cost savings is 9.2%. In only 16 of the 345 schedules did the notification scheme perform worse than static scheme by more than 1%.

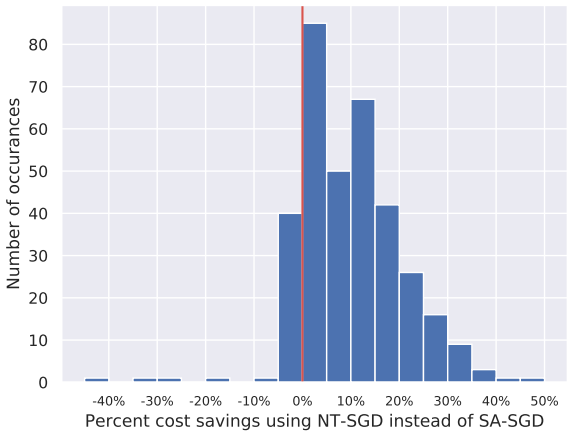

Fig. 10. For 345 realizations of operating rooms in our validation set, we calculate how much cost is saved by using the AK-NT-SGD scheme as opposed to the SA-SGD scheme. We then compile a histogram of the cost savings. $\alpha = 1$, $\beta = 3$, $\gamma = 1.5$, $T = 60$; $d = 7$ hours $\times 60$ min $= 420$

V. Conclusion

In this paper, we modeled the static appointment problem as a minimization of a linear combination of expected wait cost, idle cost, and overtime cost, and we demonstrate that this function is convex with bounded gradients. While gradient descent was too unwieldy due to the expectation in the objective function, we instead approached the appointment problem with a stochastic gradient descent algorithm. We show that this algorithm will converge to the optimum at a rate of $\mathcal{O}(\log(t)/\sqrt{t})$. Next, we introduced the concept of notification times as a new way of dynamically approaching the appointment scheduling problem. We modified the SGD algorithm to suite this problem and proved under certain conditions that it outperforms the static scheme. Moreover, we empirically demonstrated on synthetic jobs and real hospital OR data that the dynamic scheme generally create lower cost schedules than the optimal static scheme.

There is still much to explore in these two problem variants. Our work addresses the objective of scheduling jobs in a fixed order; the next task to address to ask is how we could integrate finding the optimal ordering of the jobs into the SGD algorithm. Furthermore, it remains unclear if the dynamic scheme provably outperforms the static scheme when the lead time is large relative to job durations.

References

[1] Miao Bai, Robert H. Storer, and Gregory L. Tonkay. "A sample gradient-based algorithm for a multiple-OR and PACU surgery scheduling problem." In: *IISE Transactions* 49.4 (2017), pp. 367–380.

[2] Mehmet A. Begen and Maurice Queyranne. "Appointment Scheduling with Discrete Random Durations." In: *Mathematics of Operations Research* 36.2 (2011), pp. 240–257.

[3] Patrick Billingsley. *Probability and Measure*. third. Wiley, 1995.

[4] L. Bottou, F. Curtis, and J. Nocedal. "Optimization Methods for Large-Scale Machine Learning." In: *SIAM Review* 60.2 (2018), pp. 223–311.

[5] Brian Denton and Diwakar Gupta. "A Sequential Bounding Approach for Optimal Appointment Scheduling." In: *IIE Transactions* 35.11 (2003), pp. 1003–1016.

[6] Brian Denton, James Viapiano, and Andrea Vogl. "Optimization of surgery sequencing and scheduling decisions under uncertainty." In: *Health Care Management Science* 10.1 (Feb. 2007), pp. 13–24.

[7] M. Eric Johnson and John Jackman. "Infinitesimal Perturbation Analysis: A Tool for Simulation." In: *The Journal of the Operational Research Society* 40.3 (1989), pp. 243–254.

[8] Guido C. Kaandorp and Ger Koole. "Optimal outpatient appointment scheduling." In: *Health Care Management Science* 10.3 (Sept. 2007), pp. 217–229.

[9] Claude Dennis Pegden and Matthew Rosenshine. "Scheduling arrivals to queues." In: *Computers and Operations Research* 17.4 (1990), pp. 343–348.

[10] Federico Sabria and Carlos F. Daganzo. "Approximate Expressions for Queueing Systems with Scheduled Arrivals and Established Service Order." In: *Transportation Science* 23.3 (1989), pp. 159–165.

[11] Ohad Shamir and Tong Zhang. "Stochastic Gradient Descent for Non-smooth Optimization: Convergence Results and Optimal Averaging Schemes." In: *Proceedings of the 30th International Conference on Machine Learning - Volume 28*. ICML'13. Atlanta, GA, USA: JMLR.org, 2013, pp. I-71–I-79.

[12] Peter Vanden Bosch, Dennis C. Dietz, and John R. Simeoni. "Scheduling Customer Arrivals to a Stochastic Service System." In: 46 (Aug. 1999), pp. 549–559.

[13] P. Patrick Wang. "Static and dynamic scheduling of customer arrivals to a single-server system." In: *Naval Research Logistics (NRL)* 40.3 (1993), pp. 345–360.

[14] P.Patrick Wang. "Optimally scheduling N customer arrival times for a single-server system." In: *Computers and Operations Research* 24.8 (1997), pp. 703–716.

[15] P.Patrick Wang. "Sequencing and scheduling N customers for a stochastic server." In: *European Journal of Operational Research* 119.3 (1999), pp. 729–738.

[16] Christos Zacharias and Michael Pinedo. "Appointment Scheduling with No-Shows and Overbooking." In: *Production and Operations Management* 23.5 (2014), pp. 788–801.

[17] Zheng Zhang and Xiaolan Xie. "Simulation-based optimization for surgery appointment scheduling of multiple operating rooms." In: *IIE Transactions* 47.9 (2015), pp. 998–1012.