On Continuous-space Embedding of Discrete-parameter Queuing Systems

Neha V. Karanjkar Madhav P. Desai

Department of Electrical Engineering
Indian Institute of Technology Bombay
Email: nehak@ee.iitb.ac.in, madhav@ee.iitb.ac.in

Abstract

We consider the problem of embedding a discrete parameter queueing system into continuous space by using simulation-based interpolation. We first show that a discrete-time Geom/Geom/1 queue with service time $T$ can be exactly simulated by making the service time a Bernoulli random variable which switches between $T_1$ and $T_2$ with $T_1 < T < T_2$ with expected value $T$. This is a form of simulation-based interpolation which can, in principle, be applied to more complex queueing networks. We show that such an interpolation is possible for queueing networks whose parameters can include deterministic service times, queue capacities, and the number of servers, and empirically demonstrate that the interpolation is well behaved. Unlike spatial interpolation, the interpolated value can be computed using a single simulation run irrespective of the number of parameters in the system. To demonstrate the utility of the interpolation scheme, we solve a discrete parameter queuing network optimization problem by embedding the discrete parameters into continuous space, and then using a continuous space optimization algorithm to find optimal configurations.

Keywords— Queuing, Simulation, Discrete Optimization
1 Introduction

The use of simulation is often necessary in the optimization of complex real-life queueing networks. Such queueing networks typically have discrete valued parameters such as queue capacities, server delays, the number of servers etc. More concretely, consider a queuing system with parameter set \( X = (x_1, x_2, \ldots, x_n) \), where each \( x_i \) can take an integer value. The set of all possible values that \( X \) can take is the discrete parameter space \( \Omega_D \). A simulation model of the system allows us to measure some objective function \( f : \Omega_D \rightarrow \mathbb{R} \) (which may be composed of performance measures such as throughput, average waiting time per customer or blocking probabilities).

If the discrete parameter space can be embedded into a continuous parameter space by using some form of interpolation, the optimization problem can be solved by directly applying descent-based continuous space techniques [1, 2]. Such continuous space methods often converge fast and can scale well with the number of parameters. We introduce an efficient simulation-based interpolation scheme using which the discrete parameters in the system can be embedded into continuous space, so that optimization can be performed using descent-based continuous space methods. The basis of our proposed interpolation technique is to randomize the simulation model itself, so that the average value of a parameter over the simulation is the required continuous value. To establish this basis, we study a Geom/Geom/1 queue (which is a discrete-time counterpart of an M/M/1 queue), and prove that the behaviour of the Geom/Geom/1 queue with service time \( T \) can be exactly simulated by further randomizing the service time parameter \( T \). Choose two numbers \( T_1, T_2 \) such that \( 1 \leq T_1 < T < T_2 < \infty \) and let \( X \) be a Bernoulli random variable with the distribution

\[
X = \begin{cases} 
T_1 & \text{with probability } p \\
T_2 & \text{with probability } 1 - p
\end{cases}
\]

chosen so that \( E(X) = T \). Then we simulate the server as follows: every time a job is started by the server, it assigns a service time \( X \) to the job. We call this the interpolated Geom/Geom/1 queue model. We prove that the interpolated Geom/Geom/1 queue simulation model with parameters \( T_1, T_2, p \) produces the same equilibrium performance and queue occupancy metrics that would have been produced by a simulation of the original Geom/Geom/1 queue.

The basic concept of randomization of the simulation model itself can be applied to parameters such as queue capacities, service time parameters and the number of servers. During such a randomized simulation, if we wish to assign a real value \( v \in \mathbb{R} \) to a parameter, we would do so by randomizing the parameter itself to take values in the set \( \lfloor v \rfloor, \lceil v \rceil \). As each parameter in the system can be embedded independently, the interpolated value can be obtained in a sin-
gle simulation run irrespective of the number of parameters. This is in contrast to spatial interpolation methods such as Lagrange interpolation, where evaluating the interpolated value at a point in $n$ dimensions would require $2^n$ simulations.

We observe that the randomization of the simulation model produces good embeddings of a discrete parameter space (buffer capacities, number of servers, and integer service times) into continuous space. We present several examples in Section 4 and demonstrate using simulations that the interpolation scheme is well behaved. The embedding is not unique, and the actual randomization can be achieved in several ways. The choice of the randomization scheme does affect the smoothness of the generated interpolation function, and in general may need to be done carefully.

Using the randomized simulation-based interpolation scheme, continuous space optimization algorithms can be applied directly over the interpolated model for finding optimal configurations. To demonstrate this, we consider a queuing network optimization problem with 6 discrete valued parameters. We embed the parameters into a continuous space using the randomized simulation-based interpolation scheme, and apply COBYLA [3], a continuous space optimization algorithm over the embedding to find the solutions. We present the results of this optimization study in Section 5. The simulation-based interpolation method can thus be a useful tool in the optimization of complex queuing networks.

2 Related Work

A survey of simulation based optimization approaches is available in [4, 5]. The discrete parameter set is typically treated separately from the continuous parameter set, using approaches such as ranking and selection, heuristic-based local search or randomized search.

If the discrete parameters have a natural integer order (for example, the number of servers), gradient-based continuous space methods can be applied for optimization. This can be achieved in two ways. The first way is to perform optimization over a meta-model of the system [6]. A meta-model (in the form of a regression model or an artificial neural network etc.) is an approximation to the original system that is computationally cheaper to evaluate. It is constructed by systematically sampling a subset of the design space. The second way is to embed the discrete parameter space into a continuous space using some form of interpolation. Unlike a meta-model, each function evaluation of the embedded model can be as accurate as the original simulation model. In [1] and [2], the embedding is performed using spatial linear interpolation. However, computing the interpolated value at a point in $n$ dimensions requires $2^n$ simulations (using linear interpolation) or $n + 1$ simulations (using simplex interpolation).
Our work presents a simulation-based interpolation method that can produce the interpolated value in a single simulation run. It is based on a randomization of the simulation model and works by *averaging in time*, unlike spatial interpolation methods which perform an averaging in space. The technique was used in [7] for the optimization of multi-core systems. In this paper, we give a justification for the method by analyzing a discrete time Geom/Geom/1 queue, and extend the technique to discrete parameter queuing systems.

### 3 The Geom/Geom/1 Queue

A discrete-time Geom/Geom/1 queue is defined as follows [8, Sec. 3.2.2]. Time is divided into equal-sized slots. Jobs arrive into the system at the beginning of a slot and depart at the end of a slot. The arrivals are Bernoulli distributed and the probability of a job arriving in a slot is $p$. The server has a single parameter $T$. Whenever the queue is non-empty, the server finishes a job at the head of the queue in the current slot, with probability $q = 1/T$.

![State transitions in a Geom/Geom/1 queue](image)

The number of jobs in the system at the end of a slot can be modeled as a Markov chain in Figure 1. The equilibrium distribution of the number of jobs ($\pi$) can be obtained by equating the rate of transitions between states $K$ and $K+1$. 

![Figure 1: State transitions in a Geom/Geom/1 queue](image)
\[ p(1 - q)\pi_K = q(1 - p)\pi_{K+1} \]
\[ \therefore \pi_{K+1} = \frac{p(1 - q)}{(1 - p)q} \pi_K \quad (1) \]
\[ \therefore \pi_{K+1} = \rho \pi_K \quad \text{where } \rho = \frac{p(1 - q)}{(1 - p)q} \quad (2) \]
\[ \therefore \pi_K = \rho^K \pi_0 \quad (3) \]

Equating the sum of all state probabilities to 1, and assuming that the system is stable \((\rho < 1)\) we get:
\[ \sum_{K=0}^{\infty} \rho^K \pi_0 = 1 \]
\[ \therefore \pi_0 = (1 - \rho) \quad (4) \]
and \(\pi_K = \rho^K (1 - \rho)\) \( (5) \)

### 3.1 The Interpolated Geom/Geom/1 Queue

We introduce an additional randomization of the Geom/Geom/1 Queue, which we call the interpolated Geom/Geom/1 queue. The interpolated Geom/Geom/1 queue is identical to a Geom/Geom/1 queue, except that the service time parameter is itself a random variable. Choose two numbers \(T_1, T_2\) such that \(1 \leq T_1 \leq T < T_2 < \infty\) and let \(X\) be a Bernoulli random variable with the distribution
\[ X = \begin{cases} 
T_1 & \text{with probability } \alpha \\
T_2 & \text{with probability } 1 - \alpha
\end{cases} \]
chosen so that \(E(X) = T\). That is, \(T = \alpha T_1 + (1 - \alpha)T_2\).

Then the behavior of the server is as follows: every time a job is started by the server, it assigns a service time \(X\) to the job. Once the job is started, the probability of the server finishing the job in the current slot is the reciprocal of the job’s service time. We say that the average service time of this interpolated Geom/Geom/1 queue is \(T\). We prove the following.

**Theorem 1** The Geom/Geom/1 queue with service time \(T\) and an interpolated Geom/Geom/1 queue with average service time \(T\) have the same queue occupancy probabilities at equilibrium.

**Proof:**
The interpolated Geom/Geom/1 queue can be modeled by the Markov chain shown in Figure 2.
• $U_K$ represents the state of the system with $K$ jobs, when the ongoing job has service time $T_1$
• $V_K$ represents the state of the system with $K$ jobs, when the ongoing job has service time $T_2$
• $q_1 = 1/T_1$ is the probability that the server finishes an ongoing job with service time $T_1$ in the current slot
• $q_2 = 1/T_2$ is the probability that the server finishes an ongoing job with service time $T_2$ in the current slot
• $\pi_{U,K}$ and $\pi_{V,K}$ are the probabilities that the system is in state $U_K$ and $V_K$ respectively
• $\pi_K = \pi_{U,K} + \pi_{V,K}$ is the probability that there are $K$ jobs in the system.

Equating the probabilities of transitions between states $K$ and $K + 1$, we get

$$p(1 - q_1)\pi_{U,K} + p(1 - q_2)\pi_{V,K} = (1 - p)q_1\pi_{U,K+1} + (1 - p)q_2\pi_{V,K+1} \quad (6)$$

We assume that the ratio $\pi_{U,K}/\pi_{V,K}$ is constant for all $K$. Let:

$$\pi_{U,K} = \beta\pi_K$$
and $$\pi_{V,K} = (1 - \beta)\pi_K$$
Where $\beta$ is a constant between 0 and 1. Substituting these values in Equation 6, we get

$$\pi_{K+1} = \frac{p}{(1-p)} \times \frac{[1 - (\beta q_1 + (1 - \beta)q_2)]}{[\beta q_1 + (1 - \beta)q_2]} \pi_K$$

(7)

Let $\rho = \frac{p}{(1-p)} \times \frac{[1 - (\beta q_1 + (1 - \beta)q_2)]}{[\beta q_1 + (1 - \beta)q_2]}$

Then $\pi_{K+1} = \rho \pi_K$

(8)

Next, we equate the probabilities of all transitions going from $U$ to $V$ and those going from $V$ to $U$:

$$(1 - \alpha)q_1 \sum_{K=0}^{\infty} p \pi_{U,K} + (1 - p)\pi_{U,K+1} = \alpha q_2 \sum_{K=0}^{\infty} p \pi_{V,K} + (1 - p)\pi_{V,K+1}$$

(9)

Substituting $\pi_{U,K} = \beta \pi_K$, $\pi_{V,K} = (1 - \beta)\pi_K$, and $\pi_{K+1} = \rho \pi_K$, we get

$$\beta(1 - \alpha)q_1 = (1 - \beta)\alpha q_2$$

(10)

Since $T = \alpha T_1 + (1 - \alpha)T_2$, $\alpha$ can be expressed as

$$\alpha = \frac{T - T_2}{T_1 - T_2} = \frac{1/q - 1/q_2}{1/q_1 - 1/q_2}$$

(11)

Substituting this value of $\alpha$ in Equation 10, we get $\beta$:

$$\beta = \frac{q - q_2}{q_1 - q_2}$$

(12)

and $q = \beta q_1 + (1 - \beta)q_2$

(13)

Substituting this in equation 7, we get:

$$\pi_{K+1} = \frac{p(1-q)}{(1-p)q} \pi_K$$

(14)

Equations 2 and 14 are identical, and thus the equilibrium queue occupancy distribution of the interpolated Geom/Geom/1 queue with average service time $T$ is the same as that of the Geom/Geom/1 queue with service time $T$. This completes the proof.

From this result, it follows that all performance metrics, such as throughput, average sojourn times, etc. are identical for the two models. The analysis can be extended easily to a continuous time M/M/1 queue as the behavior of a
Geom/Geom/1 queue is identical to that of the continuous time M/M/1 queue in the limit (as the length of each slot approaches zero).

We observe that the interpolated Geom/Geom/1 queue gives us a simulation algorithm in which a parameter of the queueing network (in this case, the server) is randomized so that the effective value of the parameter is an average of two values. This approach can be used to embed a discrete valued parameter in a queueing network into continuous space.

4 Interpolation in More Complex Queuing Systems

We show that the interpolation strategy can be extended to parameters such as queue capacities, service times and number of servers. We consider several examples of discrete time queuing systems and show that the method produces good interpolations. The same embedding scheme can be applied to corresponding continuous time queues.

4.1 Randomization scheme

To embed a discrete parameter \( p \) into continuous space, we construct an interpolation model where the value of \( p \) is randomized. The interpolation model can be defined at \( p = v \in \mathbb{R} \) as follows:

Choose two integers \( v_1, v_2 \) such that \( 0 < v_1 \leq v \leq v_2 \). A natural choice is to make \( v_1 = \lfloor v \rfloor \) and \( v_2 = \lceil v \rceil \). Let \( \Omega_p \subset \mathbb{R} \) be the set of values that the parameter \( p \) can take in the interpolated model, and let \( h : \Omega_p \to \mathbb{R} \) be any continuous and strictly monotonic function over \( \Omega_p \). We define a number \( \alpha \) (where \( 0 \leq \alpha \leq 1 \)) as

\[
\alpha = \frac{h(v_2) - h(v)}{h(v_2) - h(v_1)}
\]

Let \( X \) be a Bernoulli random variable with the following distribution:

\[
X(v, v_1, v_2, h) = \begin{cases} v_1 & \text{with probability } \alpha \\ v_2 & \text{with probability } 1 - \alpha \end{cases}
\]  

(15)

Then in the interpolated model, the parameter \( p \) is assigned values of the random variable \( X \) at key points during simulation. The expected value of the parameter (that is, \( E(X) \)) varies continuously with \( v \) and this produces an embedding of the parameter into continuous space. The embedding is not unique and \( h \) can be chosen in several ways. In most cases, we observe that choosing \( h(x) = x \) or \( h(x) = \frac{1}{x} \) produces a smooth interpolation. We present specific examples in the following sections.
4.2 Interpolation-based embedding of the service time \((T)\) in a Geom/D/1 queue

Consider a Geom/D/1 queue (a discrete time counterpart of an M/D/1 queue). Arrivals are Bernoulli distributed and the probability of a job arriving in a slot is \(p\). The server is deterministic. Each job accepted by the server takes a constant time of \(T\) slots to finish, where \(T \in \mathbb{N}\). To embed the discrete parameter \(T\) into continuous space, we construct an interpolated model where \(T\) is randomized.

To evaluate the interpolation at \(v \in \mathbb{R}\), we choose a function \(h(x) = x\) and obtain the Bernoulli random variable \(X(v, \lfloor v \rfloor, \lceil v \rceil, h)\) (as introduced in Eq. 15). In the simulation, whenever a job is accepted by the server, we assign the job a service time of \(X\) slots.

In Figure 3, we show the average number of jobs in the system (denoted by \(N(T)\)) observed in a simulation of the interpolated Geom/D/1 queue (with job arrival probability \(p = 0.1\)). The service time parameter \(T\) is swept in steps of 0.1 and \(N(T)\) is computed at each point using a single simulation run of the interpolated model. We see that the interpolated function \(N(T)\) is continuous and smooth and agrees with the original Geom/D/1 queue at integer values of \(T\). The statistical error in the measured value of \(N(T)\) reduces as each simulation run is made longer. As a result, the smoothness of the interpolated function also improves with the length of the simulation. In Figure 4, we show the effect of the simulation length (in units of number of jobs simulated) on the smoothness of the interpolation in the Geom/D/1 example.

4.3 Interpolation-based embedding of the number of servers \((K)\) in Geom/Geom/K and Geom/D/K queues

Consider a Geom/Geom/K queue (a discrete time counterpart of an M/M/K queue), with \(K\) identical servers working in parallel. The probability of a job arriving in a slot is \(p\). The behavior of the system can be specified as if the system contains a single server with a parameter \(K\).

- The server pulls jobs from the head of the queue as long as the queue is not empty and the number of ongoing jobs in the server is less than \(K\).

- The server then iterates over the list of ongoing jobs. For each ongoing job, the server finishes the job in the current slot with probability \(q\).

To embed the discrete parameter \(K\) into continuous space, we construct an interpolated model by randomizing \(K\). To evaluate the interpolation at \(v \in \mathbb{R}\),
Figure 3: Mean number of jobs in the system \( (N) \) versus service time \( (T) \) in a discrete-time Geom/D/1 queue \( (p = 0.1) \)

Figure 4: Effect of simulation length of the smoothness of interpolation. The function being interpolated is the mean number of jobs in the system \( N(T) \) as a function of service time \( T \) in a Geom/D/1 queue
we choose \( h(x) = x \) and a Bernoulli random variable \( X(v, \lfloor v \rfloor, \lceil v \rceil, h) \) (as introduced in Eq. 15). In the interpolated model, whenever the server starts a new job, the parameter \( K \) is modified by setting it to the random variable \( X \).

In Figure 5, we show the simulation results obtained with this interpolation scheme for a Geom/Geom/K queue with arrival probability \( p = 1/10 \) and service probability \( q = 1/19 \). The interpolated function \( N(K) \) is the average number of jobs in the system as a function of \( K \).

![Figure 5: Mean number of jobs in the system (N) versus number of servers (K) in a discrete-time Geom/Geom/K queue (p = 1/10, q = 1/19)](image)

The interpolation scheme for a Geom/D/K queue is identical to that for a Geom/Geom/K queue, except that all jobs started by the server take \( T \) slots to finish, where \( T \) is a constant. In Figure 6, we show the simulation results obtained with this interpolation scheme for a Geom/D/K queue with arrival probability \( p = 1/10 \) and service time \( T = 19 \) slots. The interpolated function \( N(K) \) is the average number of jobs in the system as a function of \( K \).

### 4.4 Interpolation-based embedding of multiple parameters: the number of servers (K) and the service time (T) in a Geom/D/K queue

For a discrete-time Geom/D/K queue, with \( K \) identical servers working in parallel and a deterministic service time \( T \), the behavior of the server is as follows:
• The server pulls jobs from the head of the queue as long as the queue is not empty and the number of ongoing jobs in the server is less than $K$. Each job $j$ accepted by the server is assigned a service time $T_j = T$.

• The server then iterates over the list of ongoing jobs, and removes jobs that have expired. A job $j$ stays in the server for $T_j$ slots.

In the interpolated model the parameters $T$ and $K$ are randomized. To evaluate the interpolation at some point $(v_T, v_K) \in \mathbb{R}^2$ in the continuous parameter space, we choose $h(x) = x$, and two independent Bernoulli random variables $X(v_T, \lfloor v_T \rfloor, \lceil v_T \rceil, h)$ and $Y(v_K, \lfloor v_K \rfloor, \lceil v_K \rceil, h)$. Whenever the server accepts a new job from the queue, the parameter $T$ is modified by setting it to the random variable $X$, and the parameter $K$ is set to the random variable $Y$.

In Figure 7, we show the simulation results obtained with this interpolation scheme for a Geom/D/K queue with arrival probability $p = 1/10$. The interpolated function $N(T, K)$ is the average number of jobs in the system. The parameter $T$ is swept in steps of 0.1 and $K$ is swept in steps of 0.05. At each point $(T, K)$ a single simulation run of the interpolated model is used to obtain $N(T, K)$. We observe that the interpolation surface is continuous and smooth. This example illustrates that multiple parameters in the model can be embedded simultaneously.
4.5 Interpolation-based embedding of queue capacity ($C$) in a Geom/Geom/1/C+1 queue

Consider a discrete-time Geom/Geom/1/C+1 queue. The queue in front of the server has a finite capacity. The behaviour of the queue with respect to the parameter $C$ (where $C \in \mathbb{N}$) is as follows:

- Whenever a job arrives, if the number of jobs already present in the queue is less than $C$, the job enters the queue. Else, the job is lost.

To embed the parameter $C$ into continuous space, we construct an interpolated Geom/Geom/1/C+1 queue by randomizing $C$. To evaluate the interpolation at $v \in \mathbb{R}$, we choose $h(x) = x$ and a Bernoulli random variable $X(v, \lfloor v \rfloor, \lceil v \rceil, h)$. Whenever a job arrives and tries to enter the queue, the parameter $C$ is modified by setting it to the random variable $X$.

In Figure 8, we show the simulation results obtained with this interpolation scheme for a system with arrival probability $p = 1/10$ and service probability $q = 1/9$. The queue capacity is swept between $(1, 9)$ in steps of 0.1. The interpolated function $P_B(C)$ is the blocking probability (probability that an arriving job is lost).
4.6 Interpolation-based embedding of queue capacity (C) in a Geom/D/1/C+1 queue

The interpolation scheme for a Geom/D/1/C+1 queue is identical to that for a Geom/Geom/1/C+1 queue (described in Section 4.5), except that all jobs started by the server take $T$ slots to finish, where $T$ is a constant. The quantity being interpolated is the blocking probability ($P_B$) as a function of queue capacity $C$.

We observe that choosing $h(x) = x$ produces an interpolation with kinks (Figure 9a). A smoother interpolation can be obtained by choosing $h(x) = \frac{1}{x}$ (Figure 9b). Thus the choice of the randomization scheme can affect the smoothness of the interpolated function, and needs to be made with care.

5 An Optimization Example

Using the randomized simulation-based interpolation scheme, the discrete parameters in a queuing system can be embedded into a continuous space. Optimization can then be performed using continuous space techniques, applied directly over the interpolated model. To demonstrate this approach we consider the optimization of a queuing network shown in Figure 10:
The system consists of three servers $S_0, S_1$ and $S_2$ with deterministic service times and a finite capacity queue in front of each server. Each server $i$ has service time $T_i$ and a queue capacity $C_i$.

- The servers $S_1$ and $S_2$ internally consist of multiple identical servers working in parallel. The number of servers in $S_1$ and $S_2$ are $K_1$ and $K_2$, respectively.
Arrivals into the system are Bernoulli distributed. The probability of a job arriving in a slot is \( p = 0.5 \). Upon arrival of a job, if the queue in front of server \( S_0 \) is full, the job is lost.

Upon processing each job, the server \( S_0 \) forwards the job to either \( S_1 \) (with probability 0.9) or to \( S_2 \) (with probability 0.1). If the destination queue selected for a job is full, the server \( S_0 \) blocks and does not process any jobs until the current job can be placed into the destination queue.

Optimization is performed over the parameter set \( \{C_1, T_1, K_1, C_2, T_2, K_2\} \). The other parameters are kept fixed at \( p = 0.5, C_0 = 2 \) and \( T_0 = 1 \). The function to be minimized \( (f) \) is a weighted sum of cost and throughput components. Cost is defined as:

\[
\text{cost} = \frac{K_1}{T_1} + \frac{K_2}{T_2}
\]

The objective function is:

\[
f = W \times \text{cost} - \text{throughput}
\]

where \( W \) is a constant representing the weight assigned to cost during optimization. Table 1 lists the bounds and constraints on the parameter values. The parameters are embedded into a continuous space by constructing a randomized interpolation model. The randomization schemes used for embedding service times \( (T_1, T_2) \) and the number of servers \( (K_1, K_2) \) are identical to those described in Section 4. The queues in front of servers \( S_1 \) and \( S_2 \) are fed by a blocking source. For embedding the queue capacity parameters \( (C_1, C_2) \), we have used a randomization with \( h(x) = x \).

Over the interpolated model, we apply COBYLA [3], a continuous space optimization algorithm that does not require computation of gradients, and can accept
| Parameter | Range (min,max) | Parameter | Range (min,max) |
|-----------|----------------|-----------|----------------|
| $C_1$     | (1,4)          | $C_2$     | (1,4)          |
| $T_1$     | (10,∞)         | $T_2$     | (10,∞)         |
| $K_1$     | (1,10)         | $K_2$     | (1,10)         |

Constraints:
$$K_1 + K_2 \leq 10, \quad C_1 + C_2 \leq 4$$

linear constraints. We use an existing implementation of COBYLA from Python’s SciPy library [9]. For each evaluation of $f$, the simulation length was chosen to be 50,000 jobs. On an average, each optimization run makes about 60 function evaluations and takes 2-3 seconds to execute on a desktop PC. We perform several optimization runs by sweeping the weight $W$ assigned to cost, and using multiple randomly chosen initial points, to get a cost-performance trade-off curve (Figure 11). Each point in the plot represents the results of a single optimization run, with the $(x,y)$ coordinates of the point representing (cost, throughput) values at the optimum. The plot shows a clear knee. The solutions are well-clustered and all solutions near the knee region are observed to have the ratio $K_1/T_1$ close to 0.5 and $K_2/T_2$ close to 0.1.

![Throughput versus Cost](image)

Figure 11: Pareto front obtained by sweeping $W$. Each point represents the results of a single optimization run, with the $(x,y)$ coordinates representing (cost, throughput) at the optimum.
To assess the smoothness of the interpolation, we plot the surface of the objective function $f$ near an optimum $((C_1, T_1, K_1, C_2, T_2, K_2) = (3, 17, 8, 1, 20, 2))$ with $W = 0.2$. As the function is 6-dimensional, we plot $f$ with respect to two parameters at a time by keeping the other parameter values fixed at their optimum (Figure 12). The plots indicate that near the optimum, the interpolation is smooth.

![Surface plots](image)

Figure 12: Surface plots of the objective function $f$ near the optimum $((C_1, T_1, K_1, C_2, T_2, K_2) = (3, 17, 8, 1, 20, 2))$

# 6 Conclusions

We have presented a simple, efficient technique for embedding discrete parameters in a queuing system into continuous space so that continuous space methods...
can be used to optimize the queuing system. The technique is based on a randomization of the simulation model, and can produce the interpolated value in a single simulation run. We observe that the technique produces good embeddings for parameters such as queue capacities and the number of servers using several examples.

To demonstrate the utility of this technique, we have solved a queuing network optimization problem with 6 parameters. The technique produced a smooth embedding of the discrete parameter space, and a continuous optimization algorithm can be effectively used on the embedding. The randomized simulation-based interpolation method can thus be a useful tool in the optimization of complex discrete parameter queuing networks.

References

[1] H. Wang and B. W. Schmeiser, “Discrete Stochastic Optimization using Linear Interpolation,” in 2008 Winter Simulation Conference. IEEE, Dec. 2008.

[2] E. Lim, “Stochastic Approximation over Multidimensional Discrete Sets with Applications to Inventory Systems and Admission Control of Queueing Networks,” ACM Trans. Model. Comput. Simul., vol. 22, no. 4, pp. 19:1–19:23, Nov. 2012.

[3] M. Powell, “On Trust Region Methods for Unconstrained Minimization without Derivatives,” Mathematical Programming, vol. 97, no. 3, 2003.

[4] M. C. Fu, “Optimization via simulation: A review,” Annals of Operations Research, vol. 53, no. 1, pp. 199–247, 1994.

[5] J. R. Swisher, P. D. Hyden, S. H. Jacobson, and L. W. Schruben, “A Survey of Simulation Optimization Techniques and Procedures,” in Simulation Conference, 2000. Proceedings. Winter, vol. 1, 2000, pp. 119–128 vol.1.

[6] R. R. Barton and M. Meckesheimer, “Chapter 18: Metamodel-Based Simulation Optimization ,” in Simulation, ser. Handbooks in Operations Research and Management Science, S. G. Henderson and B. L. Nelson, Eds. Elsevier, 2006, vol. 13, pp. 535 – 574.

[7] N. V. Karanjkar and M. P. Desai, “An Approach to Discrete Parameter Design Space Exploration of Multi-core Systems Using a Novel Simulation Based Interpolation Technique,” in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2015 IEEE 23rd International Symposium on, Oct 2015, pp. 85–88.
[8] T. G. Robertazzi, *Networks and Grids: Technology and Theory*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[9] Implementation of COBYLA in Python’s scipy.optimize library. [Online]. Available: http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.optimize.fmin_cobyla.html