

# On Continuous-space Embedding of Discrete-parameter Queueing Systems

Neha Karanjkar<sup>\*</sup>   Madhav P. Desai<sup>†</sup>   Shalabh Bhatnagar<sup>‡</sup>

## Abstract

Motivated by the problem of discrete-parameter simulation optimization (DPSO) of queueing systems, we consider the problem of embedding the discrete parameter space into a continuous one so that descent-based continuous-space methods could be directly applied for efficient optimization. We show that a randomization of the simulation model itself can be used to achieve such an embedding when the objective function is a long-run average measure. Unlike spatial interpolation, the computational cost of this embedding is independent of the number of parameters in the system, making the approach ideally suited to high-dimensional problems. We describe in detail the application of this technique to discrete-time queues for embedding queue capacities, number of servers and server-delay parameters into continuous space and empirically show that the technique can produce smooth interpolations of the objective function. Through an optimization case-study of a queueing network with  $10^7$  design points, we demonstrate that existing continuous optimizers can be effectively applied over such an embedding to find good solutions.

**Keywords**—*Queueing Systems, Discrete Parameter Simulation Optimization, Interpolation*

---

<sup>\*</sup>Post-doctoral Fellow, Robert Bosch Centre for Cyber-physical Systems, IISc Bangalore

<sup>†</sup>Professor, Department of Electrical Engineering, IIT Bombay

<sup>‡</sup>Professor and Chair, Department of Computer Science and Automation, IISc Bangalore

# 1 Introduction

## 1.1 Motivation

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, the number of servers and server-delays. More concretely, consider a queueing system with a parameter set  $X = (x_1, x_2, \dots, x_n)$  where each  $x_i$  can take integer values between some fixed bounds. The set of all possible values that  $X$  can take is the  $n$ -dimensional discrete parameter space  $\Omega_D$ . Let  $f : \Omega_D \rightarrow \mathbb{R}$  be a cost/performance measure of the system that we wish to optimize. In many applications,  $f$  may be composed of long-run average measures such as average throughput, average waiting time per customer or blocking probabilities. An analytical expression for  $f$  is rarely available and given  $X$ ,  $f(X)$  can only be measured using a simulation of the system. The measurements are noisy as each simulation run has finite length. We are motivated by the problem of finding an  $X^* \in \Omega_D$  that minimizes  $f(X^*)$ . This is a discrete-parameter simulation optimization (DPSO) problem. The problem is often difficult as the number of parameters can be very large and each function evaluation is computationally expensive.

For small parameter sets, techniques such as Optimal Computing Budget Allocation (OCBA) [1] have been effectively applied. When the number of parameters is large, an exhaustive evaluation of all design points is infeasible. The goal then is to find the best possible solution within a finite computational budget, rather than the global optimum. In such a case, randomized search techniques such as simulated annealing [2, 3] and genetic algorithms [4] or heuristic-based local search methods such as Tabu search [5] have been employed. Discrete-space variants of continuous optimizers such as Simultaneous Perturbation Stochastic Approximation (SPSA) [6] have also been proposed wherein the parameter estimate is projected back to the discrete space at each iteration [7, 8]. In all of the above methods, the objective function evaluation is limited strictly to points in the original discrete domain.

If the discrete parameter space can be embedded into a larger continuous space by using some form of interpolation, the optimization problem can be solved by directly applying descent-based continuous-space methods. Let  $\Omega_C \subset \mathbb{R}^n$  be the convex hull of  $\Omega_D$ . An embedding of the discrete parameter space into a continuous one is essentially an interpolation (say  $\hat{f}$ ) of  $f$  defined over  $\Omega_C$ . If  $\hat{f}$  can be constructed in a computationally efficient manner and has a suitable *structure* (that is,  $\hat{f}$  is continuous, piece-wise smooth, has few local minima) then continuous optimizers applied directly over  $\hat{f}$  can be expected to perform well. This is because gradient information can be utilized at each step to converge rapidly to local minima. Random multi-starts can be used in the case of non-convex functions. Most importantly, such methods often scale well with the number of parameters in comparison to enumerative approaches. However, the solution needs to be projected back to the original discrete space carefully. It is to be noted that an embedding-based approach is essentially different from methods where the parameter vector is rounded/truncated to integer points at each step. In the former case, a continuous-space offers more pathways to reach the solution aggressively.

The continuous-space embedding approach is suited to simulation optimization of queueing, inventory or manufacturing systems where most discrete parameters have an integer ordering and the objective function has some structure (unlike combinatorial problems). Such an approach has been recently reported in [9] and [10] for the optimization of queueing and inventory systems where spatial interpolation (piecewise-local interpolation over a simplex) was used to obtain a continuous-space embedding of the objective function.

## 1.2 Problem Statement

Motivated by the embedding-based approach to the DPSO problem, we consider the problem of obtaining a continuous-space embedding in a *computationally efficient* manner. Consider a system with the parameter vector  $X = (x_1, x_2, \dots, x_n)$  where each  $x_i$  is discrete valued and belongs to the set  $\Omega_{D_i} = \{x \mid x \in \mathbb{N}, x_i^{min} \leq x \leq x_i^{max}\}$ . Let  $\Omega_{C_i}$  denote the range  $[x_i^{min}, x_i^{max}]$ . The set of all possible values that  $X$  can take is  $\Omega_D = \prod_{i=1}^n \Omega_{D_i}$  and  $\Omega_C$  is the convex hull of  $\Omega_D$ .

We consider the problem of finding  $\hat{f} : \Omega_C \rightarrow \mathbb{R}$  such that  $\hat{f}(Y) = f(Y)$  when  $Y \in \Omega_D$  and  $\hat{f}$  is continuous over  $\Omega_C$ . Further, it is desirable that  $\hat{f}$  is continuously differentiable over  $\Omega_C$  so that gradient-based methods could be applied. Spatial interpolation as a means of obtaining  $\hat{f}$  is computationally expensive. Given  $Y \in \Omega_C$  and a set of points  $X^1, X^2, \dots, X^p \in \Omega_D$  in the neighbourhood of  $Y$ , the interpolation  $\hat{f}(Y)$  can be found as:

$$\hat{f}(Y) = \sum_{k=1}^p a_k f(X^k)$$

where  $a_1, \dots, a_p$  are interpolation coefficients. Here  $p$  simulations are required to estimate the interpolated value. For linear interpolation,  $p = 2^n$  and for piece-wise simplex interpolation,  $p \geq n + 1$ . Thus the computational effort required for the embedding grows rapidly with the dimensionality of the parameter space.

Instead, we propose an embedding technique which is based on a randomization of the simulation model itself. The interpolated value can be computed using a *single* simulation of the randomized model, irrespective of the dimensionality of the parameter space. The basic idea behind the embedding technique is described next.

## 1.3 Embedding via Randomization

Let  $Y = (y_1, y_2, \dots, y_n)$  be a point in  $\Omega_C$  at which we wish to compute the interpolated value  $\hat{f}(Y)$ . Thus the  $i^{th}$  parameter needs to be assigned a value  $y_i \in \mathbb{R}$ . To compute the interpolation, we construct a randomized version of the model where each parameter  $i$  is perturbed periodically (for example, at the beginning of each time-slot) and assigned values of a discrete random variable which we denote as  $\gamma_i(y_i)$ . The random variable  $\gamma_i$  is chosen in such a way that its moments can be made to vary continuously with respect to the parameter  $y_i$ , and  $\gamma_i(y_i) = y_i$  with probability 1 whenever  $y_i \in \Omega_{D_i}$ . The simplest example of such a random variable is:

$$\gamma(v) = \begin{cases} \lfloor v \rfloor & \text{with probability } \alpha(v) \\ \lceil v \rceil & \text{with probability } 1 - \alpha(v) \end{cases} \quad (1)$$

$$\text{where } \alpha(v) = \lceil v \rceil - v$$

Let  $\hat{f}$  be the long-run average measure obtained by simulating such a randomized model.  $\hat{f}$  is now a function of  $Y$ . Further,  $\hat{f}(Y) = f(Y)$  when  $Y \in \Omega_D$  by definition. Thus  $\hat{f}$  is an interpolation of  $f$ . As each parameter in the model can be embedded independently, the interpolated value can be computed using a single simulation of the randomized model, irrespective of the number of parameters. In essence, the interpolation technique relies on averaging in *time* in contrast to spatial interpolation methods which perform an averaging in space.

Such a technique can be applied in principle, to several types of discrete-event systems where the objective is a long-run average. A randomization-based approach was first reported in [11] for sensitivity measurement in VLSI systems (for producing small real-valued perturbations to discrete-valued parameters) and in [12] for the design optimization of multi-core system architectures. Both these works

demonstrated a specific application of the technique, however, without a theoretical justification. A theoretical basis for the embedding technique was introduced in [13] in the context of two specific algorithms for solving the DPSO problem. This work proposed variants of two continuous optimizers (SPSA and Smoothed Functional (SF) algorithm) wherein the parameter estimate at each iteration is projected back to the discrete space probabilistically (rather than in a deterministic manner) and showed that this effectively produces a continuous embedding of the underlying discrete-parameter process. The work focused on the optimization algorithms and the embedding technique itself was not explored in depth.

The focus of the current work is on the randomization-based embedding technique itself and its application to queueing systems (rather than a specific optimization method). In fact, once a suitable embedding has been obtained, a rich set of existing continuous optimizers become directly applicable to the original DPSO problem. The main contributions of this paper are listed below.

## 1.4 Our Contributions

We present a general scheme for randomization of discrete-valued parameters in a simulation model and show that such a randomization can indeed produce continuous interpolations of the long-run average objective. The proof (presented in Section 2) uses a result from Markov chain perturbation theory [14] and builds on a proof in [13] by relaxing some of its assumptions. The interpolation is not unique and can be tuned by varying the shape of the weights (probabilities) used during the randomization. We refer to these weights as *Stochastic Interpolation Coefficients*. In Section 3 we present a parameterized template for generating these stochastic interpolation coefficients. We then discuss the desirable properties of a good interpolation and demonstrate how the template parameters can be varied to improve the interpolation.

We explore in detail the application of the embedding technique to discrete-time queues. Such queues are of importance in many applications such as communication networks, manufacturing lines and transportation systems [15, 16]. Through several concrete examples, we demonstrate the continuous-space embedding of queue-capacity, number of servers and server-delay parameters and observe that the technique can produce smooth interpolations of the objective function. We present detailed simulation results for these discrete-time queues in Section 4. The simulation models and scripts used for all results reported in this paper are available online (see [17]).

As a demonstration of the utility of this embedding technique, we present an optimization case-study of a queueing network with 7 integer parameters and  $10^7$  design points. We observe that a randomization of the simulation model produces smooth embeddings of the objective function with a low computational overhead. Two well-established continuous-space optimizers (COBYLA [18] and SPSA [6]) applied directly over the embedding perform well and find good solutions in comparison to a direct discrete-space search. We present the observations from this optimization case-study in Section 5. In summary, our work establishes a computationally efficient technique for embedding discrete-parameter queueing systems into continuous-space, enabling direct application of continuous optimizers to solve the DPSO problem.

## 2 A General Randomization Scheme

We describe the randomization scheme with respect to a single parameter. All parameters in the model can be embedded in a similar manner simultaneously and independently of each other. Consider the  $i^{th}$  parameter in the model which can take integer values from the finite set  $\Omega_{D_i} = \{x_i^1, x_i^2, \dots, x_i^p\}$ . Let  $\Omega_{C_i} \subset \mathbb{R}$  denote the range  $[x_i^1, x_i^p]$ .

To embed the  $i^{\text{th}}$  parameter into a continuous space and assign to it the value  $y_i \in \Omega_{C_i}$ , we construct a randomized version of the model where the parameter is perturbed at multiple time-instants within a single simulation trajectory and assigned values of the discrete random variable  $\gamma_i(y_i)$ . The random variable  $\gamma_i$  can have a multi-point distribution taking values from the set  $\Omega_{D_i}$ . We choose a set of functions  $\alpha_i^1, \alpha_i^2, \dots, \alpha_i^p$  which map values from the domain  $\Omega_{C_i}$  to the range  $[0, 1]$  such that:

- $\sum_{k=1}^p \alpha_i^k(y) = 1 \quad \forall y \in \Omega_{C_i}$ , (2)

- $\alpha_i^k(y) = 1$  when  $y = x_i^k$  for  $k \in \{1, 2, \dots, p\}$ , (3)

- $\alpha_i^1, \dots, \alpha_i^p$  are continuous at all points in  $\Omega_{C_i}$ . (4)

The random variable  $\gamma_i(y)$  then has the distribution:

$$\gamma_i(y) = x_i^k \text{ with probability } \alpha_i^k(y) \quad \text{for } k \in \{1, 2, \dots, p\}. \quad (5)$$

At a given point  $y = y_i$  the values  $\alpha_i^1(y_i), \alpha_i^2(y_i), \dots, \alpha_i^p(y_i)$  serve as stochastic interpolation coefficients. There are an infinite number of choices for the set of functions  $\{\alpha_i^1, \alpha_i^2, \dots, \alpha_i^p\}$  that satisfy

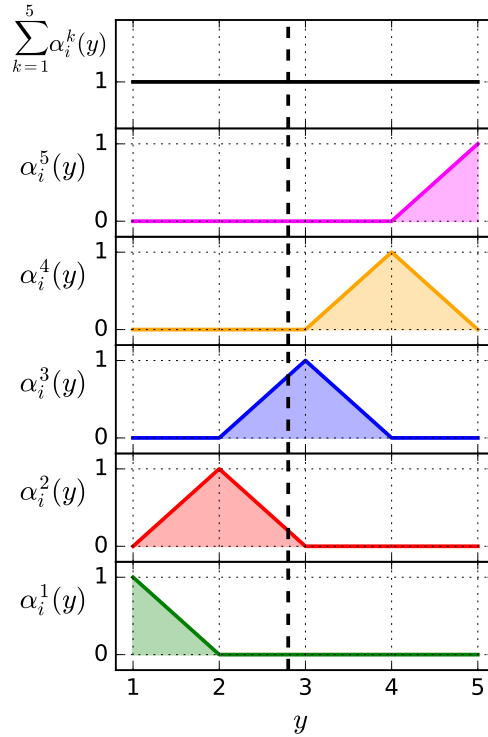


Figure 1: An example for the set of stochastic interpolation coefficient functions over the domain  $\Omega_{D_i} = \{1, 2, \dots, 5\}$ . At a given point  $y = y_i$ ,  $\alpha_i^k(y_i)$  represents the probability with which the random variable  $\gamma_i$  takes the value  $x_i^k$ . For example, at  $y = 2.8$  indicated by the dashed line,  $\alpha_i^2 = 0.2$  and  $\alpha_i^3 = 0.8$  whereas  $\alpha_i^k = 0$  for  $k \notin \{2, 3\}$ .

conditions (2) to (4). We illustrate one example of such a set in Figure 1. In general, the shape of these functions will affect the resulting interpolation  $\hat{f}$ . In the following paragraphs we show that such a randomization can indeed produce continuous interpolations of the long-run average measure. The result is similar to [13, Lemma 3], except that the assumption about the ergodicity of the constituent Markov

chains has been relaxed. This makes the analysis applicable to a wider set of parameters and systems (including chains with transient or periodic states such as those described in Section 4).

## 2.1 Analysis

Let  $X^1, X^2, \dots, X^m$  be points in the  $n$ -dimensional discrete parameter space  $\Omega_D$ . At each parameter point  $X^j$  we assume that the behavior of the system can be described as a stationary Markov chain with a finite state-space  $\mathcal{S}$  and a transition probability matrix  $P^j$ . Such a chain will have a unique stationary distribution (that is, a unique value of the distribution  $\pi$  which satisfies  $\pi P^j = \pi$ ) unless it contains two or more closed communicating classes. We assume that at each point  $X^j \in \Omega_D$  the corresponding chain contains exactly one closed communicating class and therefore has a unique stationary distribution  $\pi^j$ . The chain is not required to be ergodic and may contain periodic states and/or some transient states. Further, we assume that the chains at  $X^1, \dots, X^m$  all share a common state-space  $\mathcal{S}$ . This assumption holds when the model behavior is well-defined for dynamically changing parameter-values, as demonstrated in Section 4. Note that it is permissible for the subset of states forming a closed communicating class at points  $X^i$  and  $X^j$  to be different or altogether non-overlapping for  $i \neq j$ .

Let  $c : \mathcal{S} \rightarrow \mathbb{R}$  be a cost function that assigns a fixed cost to every occurrence of a state in the Markov chain. Let  $\pi_s^j$  denote the probability of occurrence of a state  $s \in \mathcal{S}$  under the distribution  $\pi^j$ . The long-run average cost  $f$  at the point  $X^j$  can then be defined in terms of the stationary distribution as follows:

$$f(X^j) = \sum_{s \in \mathcal{S}} \pi_s^j c(s). \quad (6)$$

Now consider the randomized model at  $Y = (y_1, y_2, \dots, y_n) \in \Omega_C$  where the  $i^{\text{th}}$  parameter in the model is assigned values of the random variable  $\gamma_i(y_i)$ . Here  $\gamma_1, \gamma_2, \dots, \gamma_n$  are independent random variables with the distribution given by Equation (5). We assume that all parameters in the model are perturbed at identical time instants. Let all of the stochastic interpolation coefficient functions  $\{\alpha_i^k : \Omega_{C_i} \rightarrow [0, 1] \mid i \in \{1, 2, \dots, n\}, k \in \{1, 2, \dots, |\Omega_{D_i}|\}\}$  be of differentiability class  $C^K$  (that is, have  $K$  continuous derivatives, where  $K$  is some non-negative integer). Let  $\hat{f} : \Omega_C \rightarrow \mathbb{R}$  be the corresponding long-run average measure of the randomized model.

**Theorem 1.**  $\hat{f}$  is a  $C^K$  interpolation of  $f$ .

*Proof.* The  $n$ -dimensional vector of parameter values at any instant of time is itself a random variable  $\Gamma$  whose distribution is a function of  $Y$  as follows:

$$\Gamma(Y) = X^j \text{ with probability } \beta^j(Y) \quad \text{for } j \in \{1, 2, \dots, m\}. \quad (7)$$

Let  $X^j = (x_1^j, x_2^j, \dots, x_n^j)$  be a point in  $\Omega_D$  and let  $I_i(x)$  denote the index of element  $x$  in the set  $\Omega_{D_i}$ . The coefficients  $\beta^j$  can then be obtained as:

$$\beta^j(Y) = \prod_{i=1}^n \mathbb{P}(\gamma_i(y_i) = x_i^j) = \prod_{i=1}^n \alpha_i^{I_i(x_i^j)}(y_i). \quad (8)$$

From equations (8) and (4) it follows that the functions  $\beta^1, \beta^2, \dots, \beta^m$  which map values from the domain  $\Omega_C$  to the range  $[0, 1]$  also satisfy the following conditions:

- $\sum_{j=1}^m \beta^j(Y) = 1 \quad \forall Y \in \Omega_C,$  (9)

- $\beta^j(Y) = 1$  when  $Y = X^j$  for  $j \in \{1, 2, \dots, m\},$  (10)

- $\beta^1, \dots, \beta^m$  are continuous at all points in  $\Omega_C.$  (11)

Let  $P(Y)$  denote the transition probability matrix of the randomized model. We choose the time instants at which to perturb the parameter values in such a way that  $P(Y)$  is given by

$$P(Y) = \sum_{j=1}^m \beta^j(Y) P^j. \quad (12)$$

In a discrete-time system, this can be achieved in a straightforward manner by perturbing the parameter values at the beginning of each time-slot. From (12) and (11) it follows that  $P(Y)$  is continuous with respect to  $Y$ . We now refer to a result from [14, Section 6] which states that: *If  $P^A$  is the transition probability matrix of a finite Markov chain containing a single irreducible subset of states (a single closed communicating class), then for an arbitrary stochastic matrix  $P^B$  with the same state-space as  $P^A$ , the randomized stationary Markov chain with transition probability matrix*

$$P(\lambda) = (1 - \lambda)P^A + \lambda P^B \quad 0 \leq \lambda < 1$$

*will also possess a single irreducible subset of states. Further,  $P(\lambda)$  has a unique stationary distribution  $\pi(\lambda)$  which is infinitely differentiable with respect to  $\lambda$  for  $\lambda \in [0, 1)$ .*

In Equation (12)  $P^1, P^2, \dots, P^m$  each have a single irreducible set of states. Therefore the stationary distribution  $\pi(Y)$  corresponding to  $P(Y)$  exists and is infinitely differentiable with respect to the coefficients  $\beta^1(Y), \dots, \beta^m(Y)$  and  $K$ -times continuously differentiable ( $C^K$ ) with respect to  $Y$ . Let  $\pi_s(Y)$  denote the probability of occurrence of a state  $s$  under the distribution  $\pi(Y)$ . The long-run average cost  $\hat{f}$  in the randomized model is given by:

$$\hat{f}(Y) = \sum_{s \in \mathcal{S}} \pi_s(Y) c(s)$$

The function  $\hat{f}$  is also  $C^K$  with respect to  $Y$ . From equations (7) and (10) we have  $\Gamma(Y) = X^j$  with probability 1 when  $Y = X^j$  for  $j \in \{1, 2, \dots, m\}$ . Thus  $\hat{f}(Y) = f(Y)$  whenever  $Y \in \Omega_D$ . Therefore  $\hat{f}$  is a  $C^K$  interpolation of  $f$ .  $\square$

Thus we have shown that a randomization of the simulation model can be used as a means of producing continuous interpolations of the long-run average measure  $f$  under the listed assumptions. Note that for  $\hat{f}$  to be of class  $C^K$ , it is sufficient but not necessary for the stochastic interpolation coefficients  $\alpha_i^1, \alpha_i^2, \dots$  to be  $C^K$  functions. For instance, it may be possible to obtain a continuously differentiable interpolation  $\hat{f}$  using coefficients that are not differentiable at integer points.

### 3 Generating the Stochastic Interpolation Coefficients

Consider the set of functions  $\{\alpha_i^k : \Omega_{C_i} \rightarrow [0, 1] \mid k \in \{1, 2, \dots, p\}\}$  that satisfy conditions (2) to (4). While an infinite number of choices exist for such a set of functions, we present one possible template that can be used to generate a family of such sets. The template is useful as the final interpolation curves can be tuned by varying the parameters of this template.

For a given point  $y \in \Omega_{C_i}$  we refer to the *stencil* around point  $y$ , denoted  $\mathbb{S}(y)$  as the set of points in  $\Omega_{D_i}$  that are chosen as basis to represent  $y$ . In other words  $\mathbb{S}(y)$  is the set of values that the random variable  $\gamma_i(y)$  would take with non-zero probabilities. For example, at  $y = 6.8$  a symmetric stencil of size 2 would be the set  $\{6, 7\}$ . As a concrete example for the case where  $\Omega_{D_i}$  consists of consecutive integers, a symmetric stencil of size  $\leq 2N$  (where  $N$  is a natural number) around any point  $y \in \Omega_{C_i}$  can be defined as follows:

$$\mathbb{S}(y) = \begin{cases} \{y\} & \text{if } y \in \Omega_{D_i} \\ \{(\lfloor y \rfloor - N + 1), \dots, \lfloor y \rfloor, \lceil y \rceil, \dots, (\lceil y \rceil + N - 1)\} \cap \Omega_{D_i} & \text{otherwise.} \end{cases} \quad (13)$$

Note that the stencil size can be lower than  $2N$  for points near the boundary of  $\Omega_{D_i}$ . Now corresponding to each point  $x^k \in \Omega_{D_i}$  we define a function  $L^k(y)$  such that  $L^k(y)$  is non-negative, continuous over  $\Omega_{C_i}$  and its value approaches 0 as  $y$  approaches points in its stencil  $\mathbb{S}(y)$  other than  $x^k$ . For example:

$$L^k(y) = \prod_{\substack{j \in \mathbb{S}(y) \\ j \neq x^k}} |y - j|. \quad (14)$$

The stochastic interpolation coefficients  $\alpha_i^k(y)$  for  $k \in \{1, 2, \dots, p\}$  can now be defined as:

$$\alpha_i^k(y) = \begin{cases} 0, & \text{if } x^k \notin \mathbb{S}(y) \\ \frac{L^k(y)}{\sum_{j \in \mathbb{S}(y)} L^j(y)} & \text{if } x^k \in \mathbb{S}(y). \end{cases} \quad (15)$$

Since the values of  $L^k(y)$  are non-negative and a normalization is performed in Equation (15) it can be seen that the coefficients  $\alpha_i^k(y)$  will always lie in the range  $[0, 1]$  and satisfy conditions (2) to (4). To obtain a more general template, let  $s$  and  $r$  be fixed constants such that  $s \in \mathbb{R} \setminus \{0\}$  and  $r \in \mathbb{R}_{>0}$ . Let  $m$  be the smallest element in  $\mathbb{S}(y)$ . Then,

$$L^k(y) = \prod_{\substack{j \in \mathbb{S}(y) \\ j \neq x^k}} |(y - m + 1)^s - (j - m + 1)^s|^r. \quad (16)$$

The coefficients  $\alpha_i^k(y)$  can be obtained as given by Equation (15) using the values of  $L^k(y)$  computed using Equation (16).

The stencil size ( $2N$ ), and the constants  $s$  and  $r$  are parameters of this template. Figure 2 shows sets of coefficients  $\alpha_i^1, \dots, \alpha_i^p$  generated using this template for different values of  $s$ ,  $r$  and the stencil size. The constant  $r$  controls the spread of  $\alpha_i^k(y)$  around the point  $y = x^k$ . The spread is linear for  $r = 1$ , and reduces with increasing values of  $r$ . It is interesting to note that for very large values of  $r$  (when the stencil size is 2 and  $s = 1$ ) the value of  $\alpha_i^k(y)$  is nearly 1 in the range  $(x^k - 0.5, x^k + 0.5)$  and the interpolation approaches a simple nearest-neighbour rounding. The constant  $s$  controls the skew. The functions are symmetric for  $s = 1$ , show a right-skew for  $s > 1$  and a left skew for  $s < 1$ . The template thus provides a simple means of varying the shape of the stochastic interpolation coefficients. In the next section, we demonstrate how this can be used to tune the final interpolation curves for a specific example.

## 4 Application to Discrete-time Queues

We now present concrete examples for the application of the embedding technique to discrete-time queues. We use the following notations and assumptions throughout this section: Time is divided into unit-sized slots. Jobs arrive into the system near the beginning of a slot and depart towards the end of the slot. At most one job can arrive within a single slot. The queue state is measured at the end of a slot, as illustrated in Figure 3.  $S_0$  denotes the initial state and  $S_t$  denotes the state measured at the end of slot  $t$ . For job arrivals with geometrically distributed inter-arrival times (denoted Geo) the probability of a job arriving in a slot is denoted as  $p$ . For geometrically distributed (Geo) service times, the probability of the server finishing a job in a slot is denoted as  $q$ . This probability is independent of the number of slots for which a job has already received service. For a deterministic server (denoted D), the number of slots taken to process a job is  $T$ .



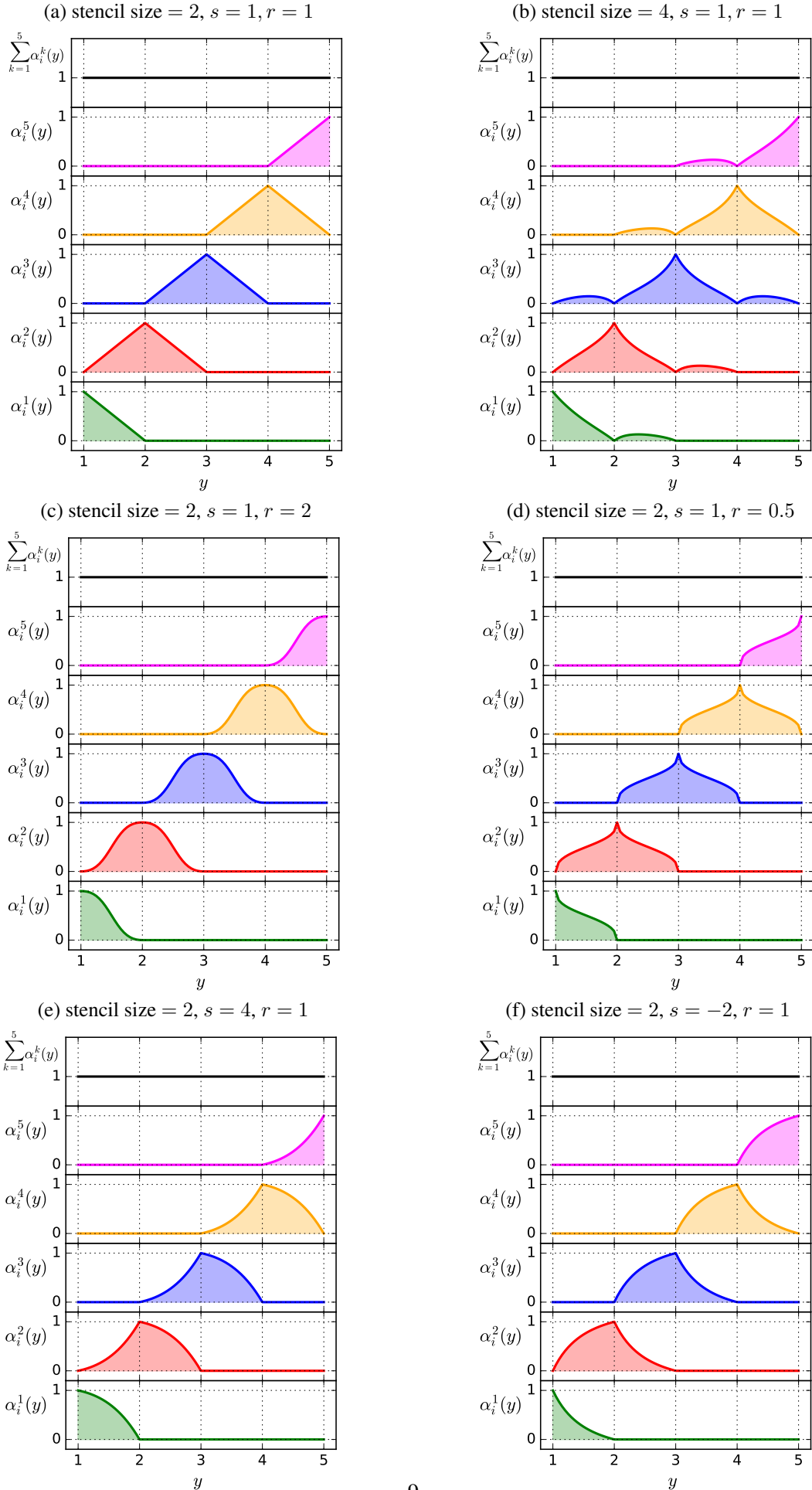


Figure 2: Stochastic interpolation coefficients over the domain  $\Omega_{C_i} = [1, 5]$

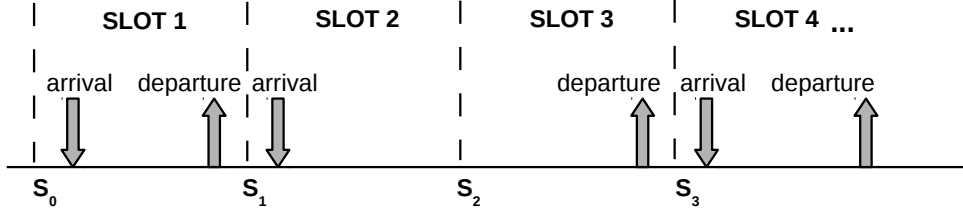


Figure 3: A Discrete-time Queue

#### 4.1 Embedding Queue-Capacity

Consider a Geo/Geo/1 queue with finite buffering. The queue state  $S_t$  is the number of jobs in the queue at the end of slot  $t$ . The queue has a capacity parameter  $C \in \mathbb{N}$  that we wish to embed into a continuous space. We first define the behavior of the queue with respect to  $C$  as follows:

**Definition 1.** *A job arriving in slot  $t$  is allowed to enter the queue if  $S_{t-1} < C$ , else the job is lost.*

By defining the capacity parameter in this way, we ensure that the Markov chains corresponding to every possible value of  $C$  share the same state-space. The states  $\{S_t \mid S_t > C\}$  are transient and unreachable from states  $\{S_t \mid S_t \leq C\}$ . As an example, Figure 4 shows the Markov chains for  $C = 1$  and  $C = 2$ . Let  $f : \mathbb{N} \rightarrow \mathbb{R}$  be some long-run average measure of this system expressed as a function of

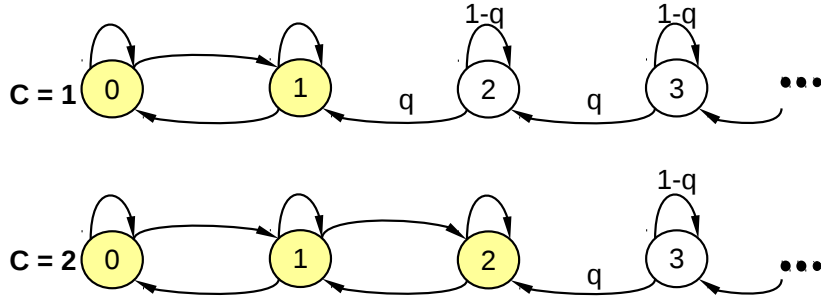


Figure 4: Markov chains for the finite capacity Geo/Geo/1 queue. (The non-shaded states are unreachable from the shaded states)

the capacity parameter  $C$ . We wish to embed the capacity parameter into continuous space, and evaluate the interpolation  $\hat{f}(y)$  of  $f$  at some given point  $y \in \mathbb{R}_{\geq 1}$ . To do so, we construct a randomized model where the capacity parameter is perturbed at the beginning of each slot and assigned the value of the random variable  $\gamma_c(y)$  with the distribution:

$$\gamma_c(y) = k \text{ with probability } \alpha^k(y) \quad \text{for } k \in \{1, 2, 3, \dots\},$$

where the coefficients  $\alpha^k$  satisfy the conditions described in Section 2. Let  $C_t$  denote the instantaneous value of the capacity parameter for the duration of slot  $t$ . By the definition of the capacity parameter above, whenever the parameter is updated the jobs already present in the queue are not disturbed. The updated value of the parameter is used solely to decide if a new job should be accepted into the queue. Thus it is possible that  $S_t > C_t$  for some values of  $t$ .

In Figure 5, we show the simulation results obtained with this randomization scheme. The interpolation coefficients  $\alpha^k$  are generated using the template described in Section 3 with the spread and skew attributes set to  $r = 1$  and  $s = -1$ . For all simulation results presented in this section we have used a 2-point stencil. We fix the arrival and service probabilities  $p, q$  and sweep the queue capacity parameter  $y$  in steps of 0.05. The interpolated function  $\hat{f}(y)$  is the blocking probability (probability of an arriving job being denied entry into the system). Each point in the plot is the mean value of  $\hat{f}(y)$  measured using 100 simulation samples with distinct randomization seeds. The shaded area around the plot represents the  $\pm 3$  standard-deviation interval.

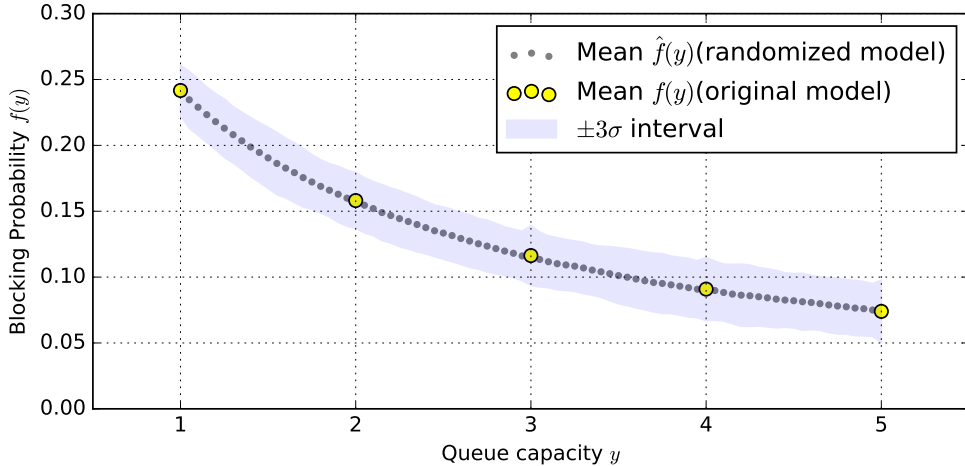


Figure 5: Blocking probability  $\hat{f}(y)$  versus the queue capacity  $y$  in a finite capacity Geo/Geo/1 queue ( $p = 0.5, q = 0.51$ , simulation length =  $10^4$  slots)

We observe that the randomization scheme produces a smooth interpolation and the standard deviation values (indicating simulation error) are similar at the discrete and the interpolated points. The computational overhead of the embedding contributed primarily by the additional calls to a random number generator, was between 5% to 10%. The overhead was computed as the relative difference between the time per simulation for the original discrete-parameter model (at some  $y = y_0 \in \mathbb{N}$ ) and the randomized model (at  $y_0 + 0.5$ ). Thus a smooth embedding of the queue capacity parameter could be obtained through a randomization of the simulation model.

## 4.2 Tuning the Interpolation

The interpolation is sensitive to the shape of the coefficient function  $\alpha^k(y)$  used during randomization. To demonstrate this, we consider the problem of embedding the queue capacity parameter in a finite capacity Geo/D/1 queue. This queue is similar to the Geo/Geo/1 queue except that the server is deterministic with a fixed service time of  $T$  slots. The queue capacity parameter is embedded using a randomization scheme identical to that of the Geo/Geo/1 case using the template described in Section 3. We show the simulation results and the resulting interpolation obtained using different values of the skew ( $s$ ) and spread ( $r$ ) parameters in Figure 6.

We observe that for  $r = 1, s = 1$  the interpolation has kinks at the integer points. If the original objective function  $f$  is integer-convex over some convex subset  $\Omega_{D_s}$  of the domain  $\Omega_D$ , it is desirable that the interpolation  $\hat{f}$  also be convex over the region that forms the convex-hull of  $\Omega_{D_s}$ . Otherwise a continuous optimizer applied directly over  $\hat{f}$  can get stuck in a local minimum that does not correspond to any minimizer of  $f$  in  $\Omega_{D_s}$ . Further, it is also desirable that the interpolation  $\hat{f}$  be smooth so that gradient-based continuous optimizers can converge fast. Considering these criteria, a reasonably good

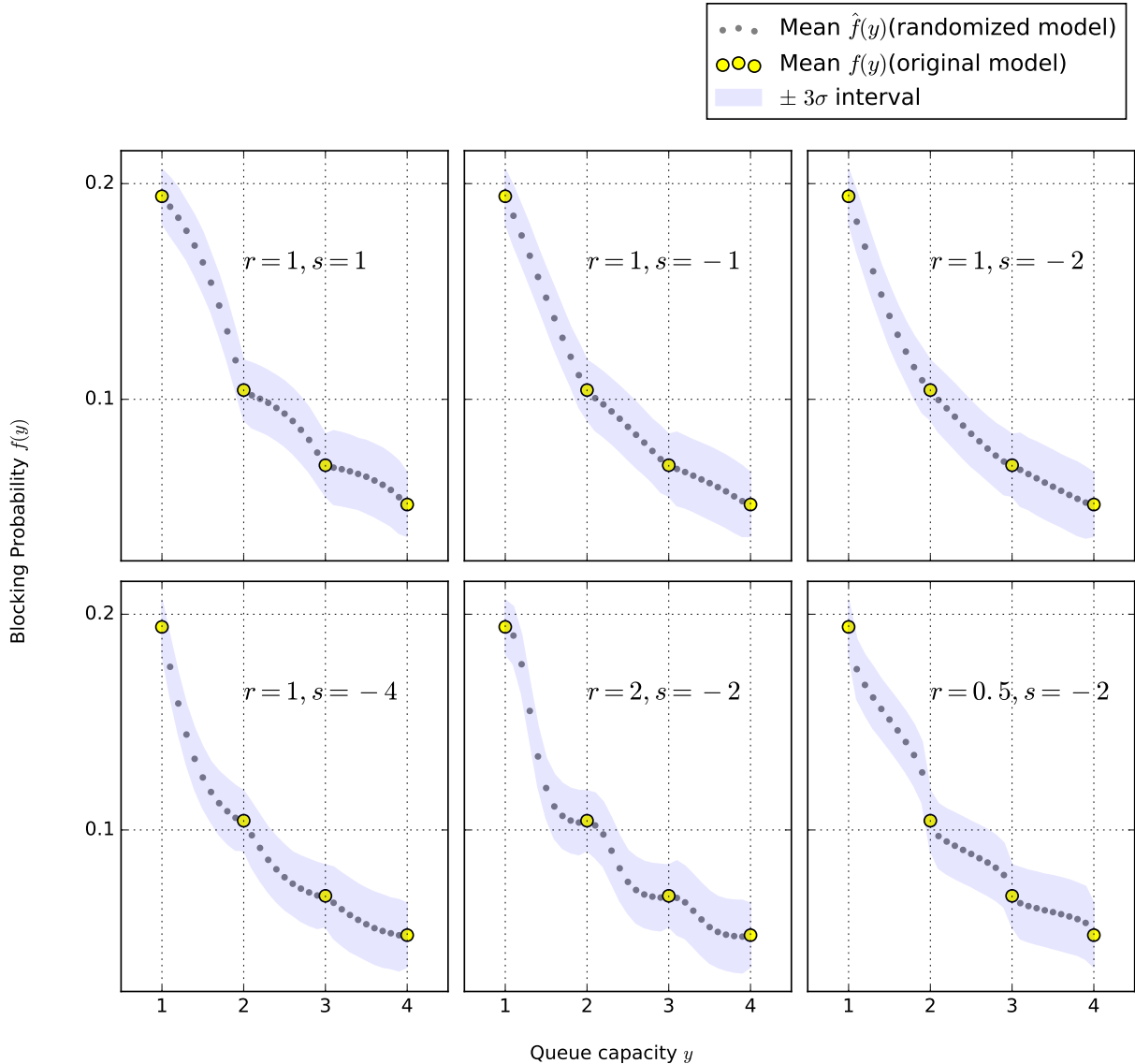


Figure 6: Interpolation results obtained for different values of the template parameters  $s$  and  $r$ . The objective function  $\hat{f}(y)$  is the blocking probability for the queue capacity  $y$  in a finite capacity Geo/D/1 queue ( $p = 0.49, T = 2$ , simulation length =  $10^4$  slots)

interpolation was obtained at  $r = 1, s = -2$ . For some systems, it may be possible to arrive at the best values for the randomization parameters ( $s, r$ ) analytically. This is an interesting problem, however beyond the scope of the current work. For examples presented henceforth, we have tuned the interpolation curves along each parameter axis by varying the template parameters.

### 4.3 Embedding the Number of Servers

Consider a Geo/Geo/ $K$  queue. The queue has infinite buffering and  $K$  identical, independent servers working in parallel. We wish to embed the parameter  $K \in \mathbb{N}$  into continuous space. To do so, we first re-define the system behavior as follows:

**Definition 2.** *The system consists of a single, infinitely fast controller with a parameter  $K$ , and a fixed large number ( $> K$ ) of identical, independent servers. In each time-slot, the controller pulls jobs from the head of the queue and assigns each job to a free server, as long as the queue is not empty and the*

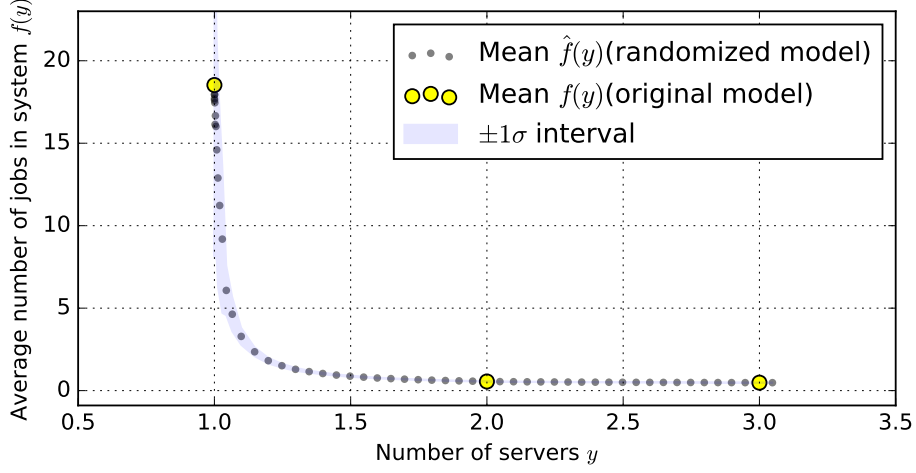


Figure 7: Average number of jobs in the system  $\hat{f}(y)$  versus the number of servers ( $K = y$ ) in a Geo/Geo/K queue ( $p = 0.5, q = 0.51$ , simulation length=  $10^4$  slots)

*number of jobs currently receiving service is less than  $K$ .*

Note that for a fixed (integer) value of  $K$ , this description is identical to a queue with  $K$  independent servers. However, the new definition of the queue behavior makes it intuitive for the controller parameter  $K$  to be updated dynamically. To embed  $K$  into continuous space and evaluate the interpolation at some point  $y \in \mathbb{R}_{\geq 1}$ , we construct a randomized model where  $K$  is perturbed at the beginning of each slot and assigned the value of the random variable  $\gamma_k(y)$ . By the definition above, whenever the parameter  $K$  is updated, the jobs already receiving service are not disturbed and the updated parameter value is used solely for deciding if service can commence for new jobs.

In Figure 7, we show the interpolation obtained using this scheme with  $s = 1$  and  $r = 1$ . We fix the arrival and service probabilities  $p, q$  and sweep the parameter  $y$  in small steps (the step-size is chosen to be smaller near the knee region). The interpolated function  $\hat{f}(y)$  is the average number of jobs in the system. Each point in the plot is the mean value of  $\hat{f}(y)$  measured using 100 simulation samples.

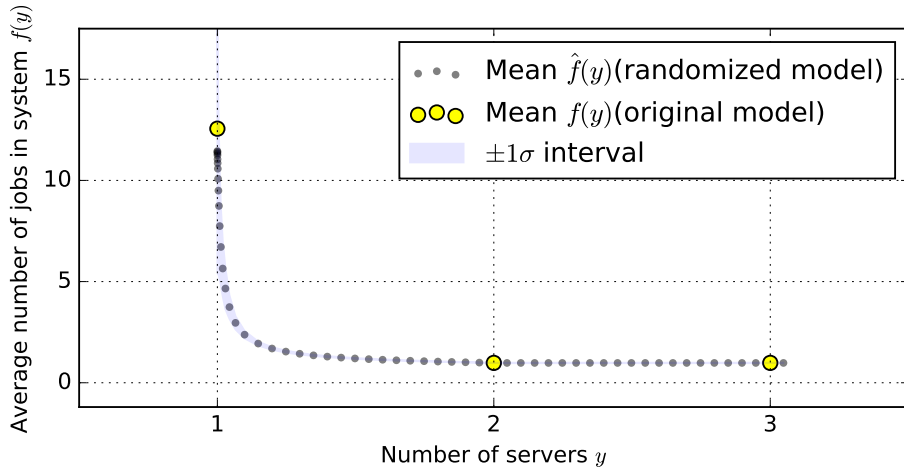


Figure 8: Average number of jobs in the system  $\hat{f}(y)$  versus the number of servers ( $K = y$ ) in a Geo/D/K queue ( $p = 0.49, T = 2$ , simulation length=  $10^4$  slots)

In Figure 8 we show the interpolation results for the same randomization scheme for a Geo/D/K queue (with a deterministic service time of  $T$  slots).

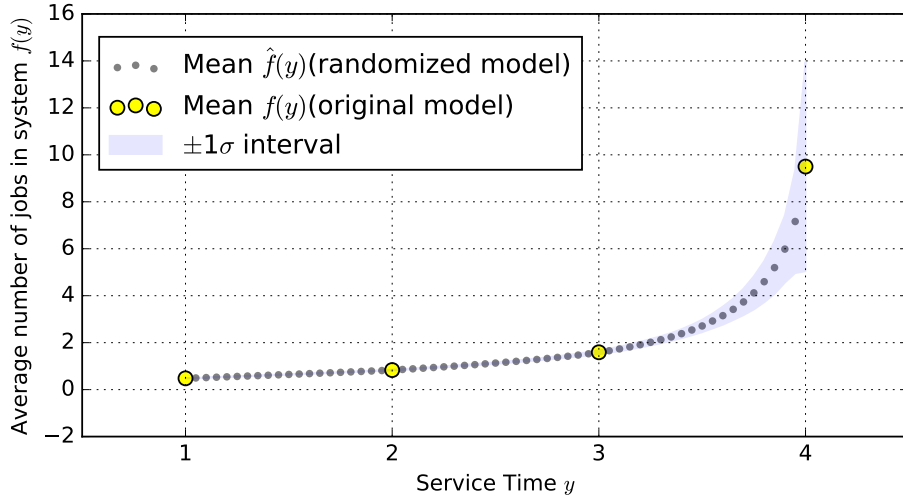


Figure 9: Average number of jobs in the system  $\hat{f}(y)$  versus the service time ( $y$ ) in a Geo/D/1 queue ( $p = 0.24$ , simulation length =  $10^4$  slots)

#### 4.4 Embedding Service Time

Consider a Geo/D/1 queue. The server is deterministic with a fixed service time of  $T \in \mathbb{N}$  slots. To embed the parameter  $T$  into continuous space, we first define the server behavior as follows:

**Definition 3.** *The server has a parameter  $T$ . At the end of each slot, the server ends jobs that have already received  $\geq T$  slots of service.*

To embed  $T$  into continuous space and evaluate the interpolation at some point  $y \in \mathbb{R}_{\geq 1}$ , we construct a randomized model where  $T$  is perturbed at the beginning of each slot and assigned the value of the random variable  $\gamma_T(y)$ . In Figure 9, we show the interpolation obtained using this randomization scheme with  $s = 1$  and  $r = 1$ . We fix the arrival probability  $p$  and sweep the service time parameter  $y$  in steps of 0.05. The interpolated function  $\hat{f}(y)$  is the average number of jobs in the system. Each point in the plot is the mean value of  $\hat{f}(y)$  measured using 100 simulation samples. The randomization produces a smooth interpolation of  $f$ .

Using the approach described in this section, multiple discrete parameters in a model can be embedded simultaneously and independently of each other, and existing continuous optimizers can be effectively applied over such an embedding. We demonstrate this using an optimization case study in the following section.

## 5 An Optimization Example

### 5.1 Problem Statement

To demonstrate the utility of the embedding technique, we consider the optimization of a queueing network in Figure 10.

- The system consists of three nodes  $n_1$ ,  $n_2$  and  $n_3$ . Each node  $n_i$  has a queue with a finite capacity  $C_i$ .
- Jobs arrive at  $n_1$  with geometrically distributed inter-arrival times (with an arrival probability  $p$ ). An arriving job that finds the queue full is lost. The node  $n_1$  consists of a single deterministic server with a service time of  $T_1$  slots. This server forwards each job to either  $n_2$  or  $n_3$  with equal probabilities, and stalls if the destination queues are full.

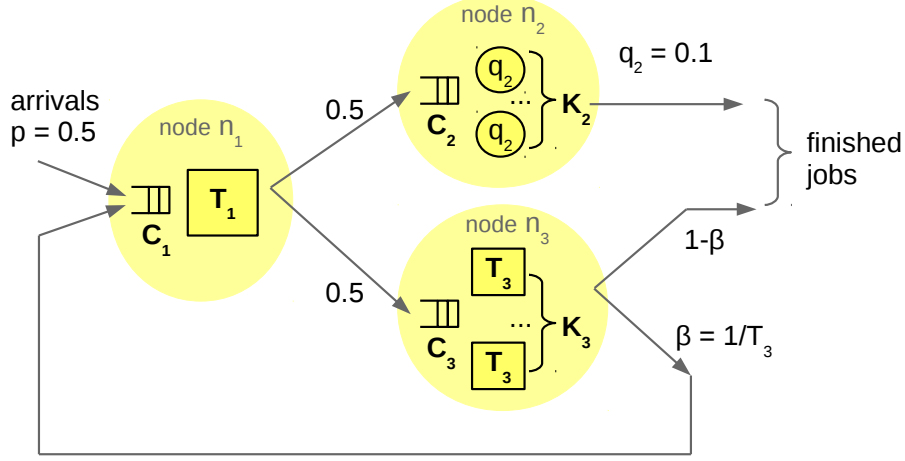


Figure 10: Queuing network to be optimized

- Nodes  $n_2$  and  $n_3$  respectively consist of  $K_2$  and  $K_3$  identical servers working in parallel. The servers in  $n_2$  have geometrically distributed service times (with service probability  $q_2$ ) whereas those in  $n_3$  are deterministic, with a service time of  $T_3$  slots.
- The servers in  $T_3$  are prone to faults. The probability of a job turning out faulty (denoted  $\beta$ ) is inversely related to the service time ( $\beta = 1/T_3$ ). A job that has received faulty service is sent back to node  $n_1$  to be re-processed as a fresh job. If the destination queue at  $n_1$  is full, the corresponding server in  $n_3$  stalls.

The parameter set for this system is  $\{C_1, C_2, C_3, T_1, T_3, K_2, K_3\}$ . Each parameter can take integer values between 1 and 10. The arrival probability  $p$  and the service parameter  $q_2$  are kept fixed ( $p = 0.5$ ,  $q_2 = 0.1$ ). Let  $X$  denote the vector of parameter values and  $\Omega_D$  denote the set of all possible values that  $X$  can take. Let  $T(X)$  denote the expected value of the long-run average throughput of the system (estimated using simulation). The throughput can be improved directly by improving the value of each parameter (that is, increasing the buffer sizes and the number of servers and reducing the service times). However, in most real-world applications improving a parameter value would also incur a certain cost. To model this trade-off we define a synthetic cost function  $C(X)$  whose value increases with increasing buffer sizes and the number of servers and reduces with increasing service-times as follows:

$$C(X) = (C_1 + C_2 + C_3) + \frac{20}{T_1} + 100K_2 + 20\frac{K_3}{T_3}. \quad (17)$$

The objective function to be minimized is the weighted sum of the normalized cost and throughput components:

$$f(X) = \frac{C(X)}{\max_{j \in \Omega_D} C(j)} - \frac{T(X)}{p} \quad (18)$$

Since an exhaustive evaluation of  $f$  over all  $10^7$  design points is infeasible, our goal is to find the best solution possible within a fixed computational budget.

## 5.2 Randomization-based Embedding

To solve this optimization problem, we first embed the discrete parameter space into a continuous one by randomizing each parameter in the model using the scheme described in Section 4. The randomization settings (listed in Table 1) were chosen via a rough tuning of the interpolation curves along each

Parameter	$r, s$	Parameter	$r, s$	Parameter	$r, s$
$C_1$	1, -2	$T_1$	1, 1	$K_2$	1, 4
$C_2$	1, 1	$T_3$	1, 1	$K_3$	1, 1
$C_3$	1, -2				

Table 1: Randomization settings (stencil size = 2 for all parameters)

parameter axis. Figure 11 shows the resulting interpolation plotted along arbitrary two-dimensional slices of the 7-dimensional parameter space. The plots were obtained by sweeping two parameters at a time (in steps of 0.25) while keeping the other parameter values fixed. The plots indicate that in the observed regions of the domain, the interpolation is reasonably smooth and suited to the application of descent-based optimization methods.

In Table 2 we list the computational overheads contributed by the additional calls to the random number generator as successive parameters in the model are embedded. The overheads were measured relative to the point  $X = (5, 5, 5, 5, 5, 5, 5)$  by successively randomizing each parameter and assigning to it the value 5.5. The values in Table 2 show that the overheads are a fraction of the total simulation time. This is in contrast to spatial interpolation methods where the computational cost of interpolation grows as integer multiples of the simulation time.

Number of parameters embedded	Avg real-time per simulation* (seconds)	Overhead
0	0.2173	0
1	0.2294	5.59 %
2	0.2304	6.06 %
3	0.2302	5.96 %
4	0.2447	12.65 %
5	0.2598	19.58 %
6	0.2701	24.31 %
7	0.2880	32.53 %

\* computed across 10 simulation runs (of length  $10^6$  slots each) with distinct randomization seeds

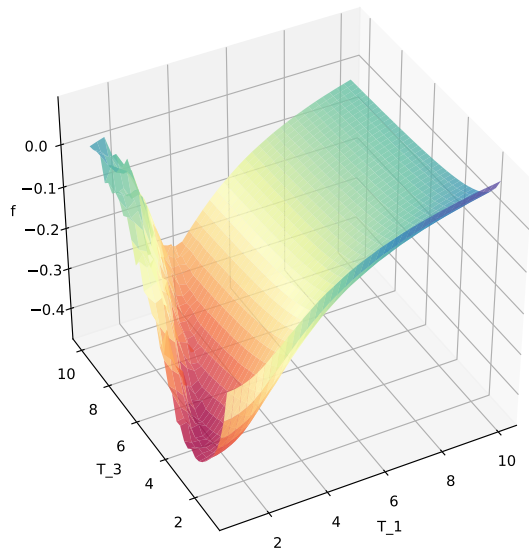
Table 2: Computational overhead of the randomization-based embedding for the 7-parameter queueing network in Figure 10

### 5.3 Optimization

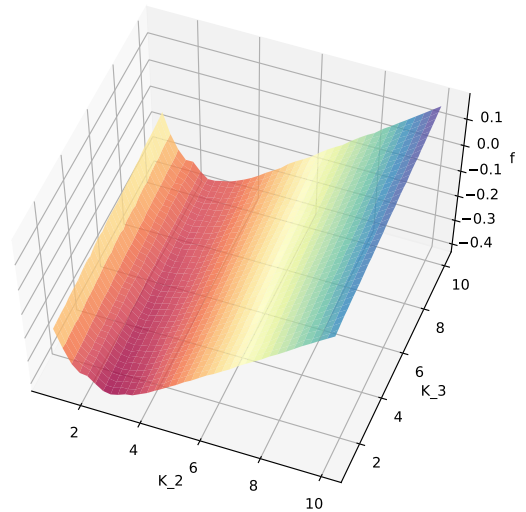
Over the randomization-based embedding, we apply two continuous optimizers: COBYLA [18, 19] and SPSA [6]. Both optimizers avoid the need for an explicit computation of numerical derivatives along each parameter axis and are thus suited to high dimensional problems. Further, both methods are suited to problems where the objective function evaluations can be noisy. COBYLA is a trust-region based method. It builds an estimate of the gradient at each step by interpolation at the vertices of a simplex in the parameter space. SPSA is a descent-based method that approximates the gradient at each step using only two objective function evaluations regardless of the dimensionality of the parameter space.

While discrete-parameter variants of SPSA exist [7], [20, Chapter 9], COBYLA cannot be applied in the discrete-parameter case. This fact illustrates the utility of our embedding technique, which makes

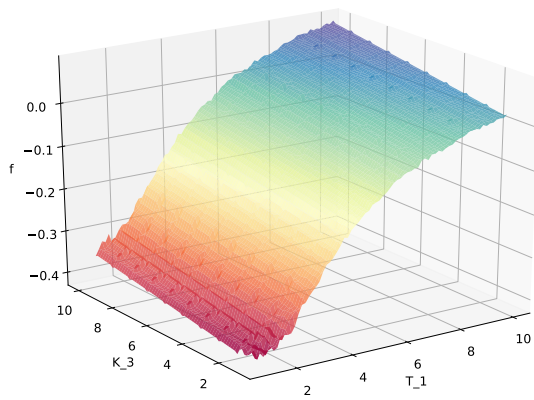




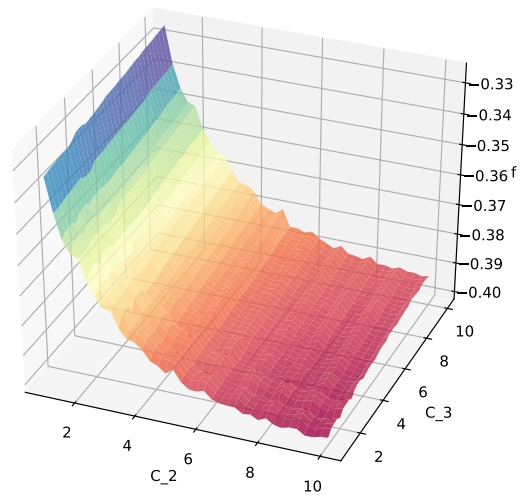
(a)  $f(T_1, T_3)$



(b)  $f(K_2, K_3)$



(c)  $f(T_1, K_3)$



(d)  $f(C_2, C_3)$

Figure 11: The interpolated objective function plotted along arbitrary 2D slices of the domain. Each point on the plot is the objective measured using a single simulation (of length  $10^4$  slots) of the randomized model.

it possible to apply existing continuous optimizers directly to solve discrete-parameter simulation optimization problems. We compare the performance of both these optimizers against a simple discrete-space variant of SPSA [7] where each objective function evaluation is restricted to points in the original discrete domain. While the focus of this work is on the embedding technique rather than specific optimization methods, we present the comparison as an evaluation of the embedding technique’s utility.

We use an implementation of COBYLA from Python’s SciPy library [21] with the settings  $\rho_{\text{beg}} = 5.0$  and  $\rho_{\text{end}} = 0.1$ . For SPSA, we use an implementation from the NoisyOpt library [22] with the default step-size schedules (suggested in [23]). The step-size parameter is identical between the continuous and discrete SPSA variants. For each method we perform 100 optimization runs using distinct, randomly chosen initial points. The set of initial points is fixed and is common across all three optimization methods. For each optimization run we set an upper limit of 1000 objective evaluations. At each objective function evaluation, the system throughput is measured using a single simulation of the randomized model (of length  $10^4$  slots) and the cost is computed analytically using Equation (17). At the end of each optimization run, we round the solution to the nearest integer point, and record the objective value at this point.

		<b>COBYLA</b>	<b>SPSA</b>	<b>Discrete-SPSA</b>
<b>Objective value at the optimum (lower is better)</b>	<b>best</b>	-0.7130	-0.7108	-0.6842
	<b>avg</b>	-0.5240	-0.1994	-0.1964
	<b>std-dev</b>	0.2930	0.3481	0.3358
<b>Avg number of objective function evaluations per optimization run</b>		52.7	1000	1000
<b>Avg time per optimization run (seconds)</b>		0.15	2.94	2.33

Table 3: Performance of COBYLA and SPSA (applied over a randomization-based embedding) and Discrete-SPSA applied directly over the discrete parameter model.

Table 3 presents the performance of the three methods measured across 100 optimization runs. We observe that COBYLA shows the best performance, both in terms of the quality of the solutions and the convergence rate. The continuous-space SPSA performs better in comparison to its discrete-parameter variant. The solutions are well-clustered. Among the top 20 solutions found by COBYLA, all have the parameter values  $T_1 = 1$ ,  $T_3 = 10$ , and  $K_2 = 3$ . The results indicate that existing continuous optimizers can be effectively applied over the embedding and their performance compares favourably against direct discrete-space search.

## 6 Conclusions

We have presented a simple and computationally efficient technique using which discrete parameters in a simulation optimization problem can be embedded into a continuous-space, enabling direct application of continuous-space methods for optimization. The technique is based on a randomization of the simulation model and is applicable to problems where the objective function is a long-run average measure. We described in detail the application of this technique to discrete-time queues for embedding queue capacities, number of servers and server-delay parameters and empirically showed that the technique can produce smooth interpolations. Further, the interpolation can be tuned by varying the shape of the coefficient functions. An analytical means to arrive at the best randomization scheme (rather than

through tuning) can be an interesting problem for future research.

Over the randomization-based embedding, existing continuous optimizers can be effectively applied to find good solutions. We demonstrated this via an optimization case study of a queueing network with 7 discrete parameters. A randomization of the model produced smooth embeddings of the objective function with a low computational overhead. Two continuous-space optimizers (COBYLA and SPSA) applied over this embedding showed good performance in comparison to a direct discrete-space search.

In summary, this work established the randomization based technique as an efficient means of embedding discrete parameter queueing systems into continuous space for simulation-based optimization over a large number of parameters. A detailed comparison across multiple continuous optimizers that can be applied over such an embedding could be the subject of future investigation. The application of the embedding technique to other kinds of systems (such as inventory models) also needs to be investigated further.

## References

- [1] C. H. Chen and L. H. Lee, *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*, 1st ed. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2010.
- [2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *SCIENCE*, vol. 220, no. 4598, pp. 671–680, 1983.
- [3] T. M. Alkhamis, M. A. Ahmed, and V. K. Tuan, "Simulated Annealing for Discrete Optimization with Estimation," *European Journal of Operational Research*, vol. 116, no. 3, pp. 530 – 544, 1999.
- [4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.
- [5] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.
- [6] J. C. Spall, "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.
- [7] J. E. Whitney, L. I. Solomon, and S. D. Hill, "Constrained optimization over discrete sets via SPSA with application to non-separable resource allocation," in *Proceeding of the 2001 Winter Simulation Conference (Cat. No.01CH37304)*, vol. 1, 2001, pp. 313–317 vol.1.
- [8] S. Bhatnagar and H. J. Kowshik, "A Discrete Parameter Stochastic Approximation Algorithm for Simulation Optimization," *SIMULATION*, vol. 81, no. 11, pp. 757–772, 2005.
- [9] 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.
- [10] H. Wang and B. W. Schmeiser, "Discrete Stochastic Optimization using Linear Interpolation," in *2008 Winter Simulation Conference*. IEEE, Dec. 2008.
- [11] G. Hazari, M. P. Desai, and G. Srinivas, "Bottleneck Identification Techniques Leading to Simplified Performance Models for Efficient Design Space Exploration in VLSI Memory Systems," in *2010 23rd International Conference on VLSI Design*. IEEE, 2010.

- [12] 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.
- [13] S. Bhatnagar, V. K. Mishra, and N. Hemachandra, “Stochastic Algorithms for Discrete Parameter Simulation Optimization,” *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 4, pp. 780–793, Oct 2011.
- [14] P. J. Schweitzer, “Perturbation Theory and Finite Markov Chains,” *Journal of Applied Probability*, vol. 5, no. 2, pp. 401–413, 1968.
- [15] A. S. Alfa, *Applied Discrete-Time Queues*, 2nd ed. Springer Publishing Company, Incorporated, 2015.
- [16] H. Bruneel, “Performance of discrete-time queueing systems,” *Computers & Operations Research*, vol. 20, no. 3, pp. 303–320, 1993.
- [17] N. Karanjkar. GitHub repository for the embedding experiments. [Online]. Available: <https://github.com/NehaKaranjkar/Embedding>
- [18] M. J. D. Powell, *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*. Dordrecht: Springer Netherlands, 1994, pp. 51–67. [Online]. Available: [https://doi.org/10.1007/978-94-015-8330-5\\_4](https://doi.org/10.1007/978-94-015-8330-5_4)
- [19] M. Powell, “On Trust Region Methods for Unconstrained Minimization without Derivatives,” *Mathematical Programming*, vol. 97, no. 3, 2003.
- [20] S. Bhatnagar, H. Prasad, and L. Prashanth, *Stochastic Recursive Algorithms for Optimization*. London: Springer London, 2013.
- [21] 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](http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.optimize.fmin_cobyla.html)
- [22] Noisyopt: A Python library for optimizing noisy functions. [Online]. Available: <https://noisyopt.readthedocs.io/en/latest/>
- [23] J. C. Spall, “Implementation of the simultaneous perturbation algorithm for stochastic optimization,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 3, pp. 817–823, Jul 1998.