A SIMULATION-BASED TECHNIQUE FOR CONTINUOUS-SPACE EMBEDDING OF DISCRETE-PARAMETER QUEUEING SYSTEMS

Neha Karanjkar*
Robert Bosch Centre for Cyber-Physical Systems, IISc Bangalore

Madhav Desai
Department of Electrical Engineering, IIT Bombay

Shalabh Bhatnagar
Department of Computer Science and Automation, IISc Bangalore

KEYWORDS
Simulation Optimization, Queueing, Interpolation

ABSTRACT

This paper presents a simulation-based technique for embedding the discrete-valued parameters in queueing systems (such as buffer capacities) into a continuous space. The significance of this technique is that it enables the direct application of existing descent-based continuous optimizers for solving simulation optimization problems efficiently over a discrete parameter set. The embedding technique is based on a randomization of the simulation model itself, and is applicable 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 well-suited to high-dimensional problems. We present a theoretical basis for this embedding technique and demonstrate its utility in the optimization of discrete-time queueing systems.

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, the number of servers and service-times in slotted-time queues. More concretely, consider a queueing system with a parameter set $X = (x_1, x_2, \ldots, x_n)$ 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 Ω_D. Let $f : \Omega_D \rightarrow \mathbb{R}$ be some cost/performance measure of the system that we wish to optimize. In many problems, 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. Further, most discrete-space search methods do not scale well with the number of parameters.

For small parameter sets, ranking and selection-based procedures such as Optimal Computing Budget Allocation (Chen and Lee (2010)) 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 and genetic algorithms or heuristic-based local search methods such as Tabu search have been employed. A detailed survey of simulation optimization approaches can be found in Swisher et al. (2000). Discrete-space variants of continuous optimizers such as Simultaneous Perturbation Stochastic Approximation (SPSA) have also been proposed wherein the parameter estimate is projected back to the discrete space at each iteration (see Whitney et al. (2001) and Bhatnagar and Kowshik (2005)). In all of the above methods, the objective function evaluation is limited strictly to points in the original discrete domain.

A Continuous-space Embedding Approach:

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. Such continuous optimizers often scale well with respect to the number of parameters, in comparison to discrete-space methods. This is because gradient information can be utilized at each step to converge rapidly to local minima. To search for a global optimum, random multi-starts can be used. The solutions thus found in the continuous domain are projected back to the discrete domain (for example, using nearest-neighbour rounding). Let $\Omega_C \subset \mathbb{R}^n$ denote the convex hull of the original discrete space Ω_D. An embedding of the discrete parameter space into a continuous one is essentially an interpolation (say $\hat{f}$) of f defined over Ω_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.

Such an embedding-based approach was shown to be effective in the optimization of queueing and inventory sys-

*Corresponding author. Email: neha.karanjkar@rbccps.org

tems in the past (see Lim (2012) and Wang and Schmeiser (2008)). However the embedding reported in existing literature was achieved via spatial interpolation (for instance, using piece-wise simplex interpolation) which requires multiple simulations to be performed at each parameter point. More concretely, given some point $Y \in \Omega_C$, and a set of points $X^1, X^2, \ldots, X^p \in \Omega_D$ in the neighbourhood of Y, the interpolated objective value $\hat{f}(Y)$ can be computed as a weighted average of the values $f(X^1), f(X^2), \ldots, f(X^p)$, each of which is obtained via a single simulation. Thus 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, spatial interpolation as a means of embedding is computationally expensive.

Instead, this paper presents an embedding technique which requires a single simulation to measure the interpolated value at a given point in the continuous domain (irrespective of the number of parameters). The technique is based on a randomization of the simulation model and is applicable when the objective f is composed of long-run average measures. To perform the embedding, each parameter in the model is perturbed periodically and assigned values of a discrete random variable, instead of a fixed constant value over a single simulation run. Now, the distribution of this random variable can be varied continuously, even though the model parameter itself is discrete-valued. The value of the objective function measured using a single, long simulation of this randomized model directly gives us the interpolated objective value. In essence, the technique relies on an averaging in *time*, in contrast to spatial interpolation methods which perform an averaging over the parameter space. We describe the embedding technique in more detail in the following section. In this paper, we present a theoretical justification for the randomization-based embedding technique and describe its application to discrete-time queueing systems.

Related Work and Our Contributions

The use of randomization for embedding discrete-valued parameters in a simulation model was reported by Karanjkar and Desai (2015) for the design optimization of multi-core systems, however without a theoretical justification. A theoretical basis for such a technique was introduced by Bhatnagar et al. (2011) in the context of two specific algorithms for solving the DPSO problem. This work proposed variants of two continuous optimizers 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 essentially 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 paper is on the randomization-based embedding technique itself. We present a general randomization scheme and prove that it produces continuous interpolations of the objective. The proof extends (Bhatnagar et al. 2011; Lemma 3) by relaxing the assumption on the ergodicity of the constituent Markov chains, making the analysis applicable to a wider set of parameters and systems (including systems with transient and/or periodic states), such as the examples considered in this paper. We then describe the application of the embedding technique to discrete-time (slotted) queues for embedding queue-capacity, number of servers and service-time parameters into a continuous space. Such queues are of importance in several applications such as communication networks, manufacturing lines and transportation systems (see Alfa (2015)). To demonstrate the utility of the embedding technique, we consider the optimization of a queueing network with 7 parameters. We observe that a randomization of the simulation model produces continuous, smooth embeddings of the objective and two continuous optimizers applied directly over this embedding perform favourably in comparison to a direct discrete-space search method. However the focus of the current paper is on the embedding technique itself, rather than on specific optimization methods. In-fact, once an embedding has been achieved, a rich set of existing continuous optimizers become applicable to the original discrete-parameter problem. A detailed performance comparison between discrete-space methods and continuous optimizers applied over an embedding can be the topic for future research and is beyond the scope of the current paper. In summary, this paper introduces the randomization-based embedding technique as a useful tool in the optimization of queueing networks over discrete parameters. The simulation models and scripts used in this work are available in an online repository (see Karanjkar (2017)).

RANDOMIZATION-BASED EMBEDDING

Consider a system with n integer-valued parameters and let $X = (x_1, x_2, \ldots, x_n)$ denote the parameter vector, where each x_i can take values from some finite set D_i consisting of successive integers. The set of all possible values that X can take is the n-dimensional discrete parameter space Ω_D which, in this case is the Cartesian product $\Omega_D = \prod_{i=1}^{n} D_i$. Given a point $X \in \Omega_D$, our simulation model allows us to measure the value of some long-run average objective $f(X)$. Let Ω_C denote the convex hull of Ω_D. We wish to obtain the interpolation $\hat{f} : \Omega_C \rightarrow \mathbb{R}$ of f, and measure its value at a given point $Y \in \Omega_C$. Let $Y = (y_1, y_2, \ldots, y_n)$. Thus the i^{th} parameter needs to be assigned a value $y_i \in \mathbb{R}$ in the embedded model.

To obtain $\hat{f}(Y)$, we construct a randomized version of the model where the value of each parameter in the model is perturbed periodically (for example, at the beginning of each time-slot) and assigned values of a discrete random variable which we denote as γ. For each $i \in \{1, \ldots, n\}$, the i^{th} parameter in the model is assigned values of the random variable $\gamma_i(y_i)$. The random variable γ_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 $y_i \in D_i$. The simplest example of such a random

variable is:

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

For instance, if $y_i = 10.3$, then at each time-slot the parameter will take the value 11 with probability 0.3 and the value 10 with probability 0.7 so that its average value over a single simulation run would be 10.3. All parameters in the model are embedded simultaneously and assigned values of independent random variables $\gamma_1(y_1), \gamma_2(y_2), \ldots \gamma_n(y_n)$. 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. Thus, the interpolation is achieved by averaging over time, instead of space.

ANALYSIS

In this section we present a general randomization scheme and prove that it produces continuous interpolations of the long-run average objective. Consider the i^{th} parameter in the model which can take integer values from the set D_i. Let $D_i = \{x_i^1, x_i^2, \ldots, x_i^p\}$ and let $C_i \subset \mathbb{R}$ denote the range $[x_i^1, x_i^p]$. To embed the i^{th} parameter into a continuous space and assign to it some value $y_i \in C_i$, we perturb the parameter periodically and assign to it values of a discrete random variable $\gamma_i(y_i)$. In general, the random variable γ_i can have a multi-point distribution, taking values from the set D_i. Let $\alpha_i^1, \alpha_i^2, \ldots, \alpha_i^p$ be a set of functions which map values from the domain C_i to the range $[0, 1]$ such that:

- $\sum_{k=1}^{p} \alpha_i^k(y) = 1 \quad \forall y \in C_i,$ (2)
- $\alpha_i^k(y) = 1$ when $y = x_i^k$ for $k \in \{1, 2, \ldots, p\},$ (3)
- $\alpha_i^1, \ldots, \alpha_i^p$ are continuous at all points in C_i. (4)

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

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

There are an infinite number of choices for the set of functions $\{\alpha_i^1, \ldots, \alpha_i^p\}$ that satisfy 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 $\hat{f}$.) Let $X^1, X^2, \ldots, X^m$ be points in the n-dimensional discrete parameter space Ω_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 π 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

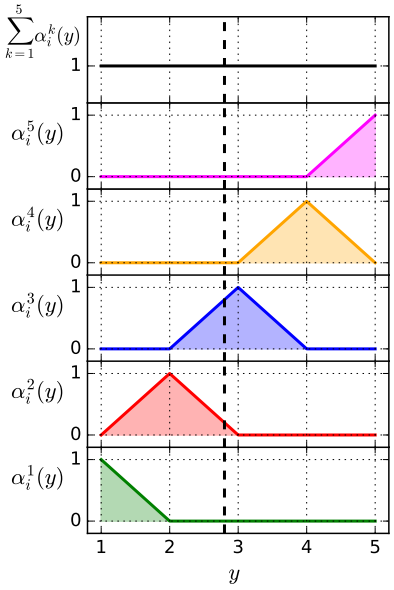


Figure 1: An example for the set of functions $\alpha_i^1, \alpha_i^2, \ldots$ that satisfy conditions (2)-(4) over the domain $D_i = \{1, 2, \ldots, 5\}$. At a given point $y = y_i$, $\alpha_i^k(y_i)$ represents the probability with which the random variable γ_i takes the value k. For example, at $y = 2.8$ indicated by the dashed line, $\alpha_i^2 = 0.2$ and $\alpha_i^3 = 0.8$ whereas $\alpha_i^k = 0$ for $k \notin \{2, 3\}$.

has a unique stationary distribution π^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 $X^1, \ldots, X^m$ all share a common state-space $\mathcal{S}$. 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 $c : \mathcal{S} \to \mathbb{R}$ be a cost function that assigns a fixed cost to every occurrence of a state in the Markov chain. Let π_s^j denote the probability of occurrence of a state $s \in \mathcal{S}$ under the distribution π^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 $Y = (y_1, y_2, \ldots, y_n) \in \Omega_C$ where the i^{th} parameter in the model is assigned values of the random variable $\gamma_i(y_i)$. Here $\gamma_1, \gamma_2, \ldots, \gamma_n$ 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 the set of functions $\{\alpha_i^k : C_i \to [0, 1] \mid i \in \{1, 2, \ldots, n\}, k \in \{1, 2, \ldots, |D_i|\}\}$ used during randomization, be continuous and have K continuous derivates, where K is some non-negative integer (that is, let each function be of differentiability class C^K where $K \in \{0, 1, 2, \ldots\}$.) Let $\hat{f} : \Omega_C \to \mathbb{R}$ be the corresponding long-run average measure of the randomized model. Then, we show that the following statement is true:

Theorem 1. *$\hat{f}$ is a continuous interpolation of f and also belongs to differentiability class C^K.*

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

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

where m is the total number of points in the discrete design space ($m = \prod_{i=1}^{n} |D_i|$). Let $X^j = (x_1^j, x_2^j, \ldots, x_n^j)$ be a point in Ω_D and let $I_i(x)$ denote the index of element x in the set D_i. The coefficients β^j can then be obtained as:

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

From equations (8) and (4) it follows that the functions $\beta^1, \beta^2, \ldots, \beta^m$ which map values from the domain Ω_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 $j \in \{1, 2, \ldots, m\}$, (10)
- $\beta^1, \ldots, \beta^m$ are continuous at all points in Ω_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

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

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 (Schweitzer 1968; 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 λ for $\lambda \in [0, 1)$.

In Equation (12) $P^1, P^2, \ldots, P^m$ 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 $\beta^1(Y), \ldots, \beta^m(Y)$ 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 $j \in \{1, 2, \ldots, m\}$. Thus $\hat{f}(Y) = f(Y)$ whenever $Y \in \Omega_D$. Therefore $\hat{f}$ is a C^K interpolation of f. □

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. It should be noted that for $\hat{f}$ to be of class C^K, it is sufficient but not necessary for the coefficient functions $\alpha_i^1, \alpha_i^2, \ldots$ to be C^K functions. For instance, it may be possible to obtain a continuously differentiable interpolation $\hat{f}$ using coefficient functions that are continuous but not differentiable at the integer points.

APPLICATION TO DISCRETE-TIME QUEUES

In this section we describe the application of the embedding technique to discrete-time queues for embedding queue-capacity, number of servers and service-time parameters into a continuous space and present simulation results for several concrete examples.

We assume that in each slot, 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. S_0 denotes the initial state of the queue 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 an ongoing job in the current slot (irrespective of the amount of time for which the job has already received service) is denoted as q. For a deterministic server (denoted D), every job takes a constant amount of time to be processed by the server. The number of slots taken to process a job is denoted as T.

For performing the randomization, we will use a more general form for the random variable γ, which was first introduced in Equation (1) as follows:

$$\gamma(y) = \begin{cases} \lfloor y \rfloor & \text{with probability } \alpha^1(y) = \alpha(y) \\ \lceil y \rceil & \text{with probability } \alpha^2(y) = 1 - \alpha(y) \end{cases}$$

$$\text{where } \alpha(y) = \begin{cases} \dfrac{\lceil y \rceil^s - y^s}{\lceil y \rceil^s - \lfloor y \rfloor^s} & \text{when } y \notin \mathbb{Z} \\ 0 & \text{when } y \in \mathbb{Z} \end{cases} \tag{13}$$

where $s \in \mathbb{R}$ is a constant. It can be seen that the probability functions $\alpha^1(y)$ and $\alpha^2(y)$ will satisfy the conditions (2)-(4) for $s \neq 0$.

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:

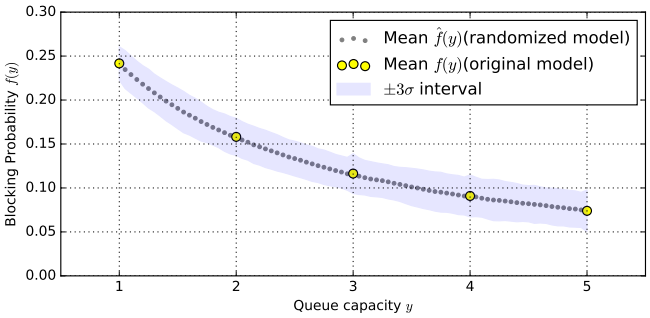


Figure 2: Interpolation produced using the randomization-based embedding technique for a finite-capacity Geo/Geo/1 queue. The embedded parameter y is the queue-capacity and $\hat{f}(y)$ is the blocking probability of the queue. (Simulation length=10^4 slots, $p = 0.5$, $q = 0.51$).

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. For a queue with a constant value of C, the states $\{S_t \mid S_t > C\}$ are transient and unreachable from states $\{S_t \mid S_t \leq C\}$. The definition also makes it intuitive for the parameter C to be updated dynamically within a simulation, in the randomized model. Let $f : \mathbb{N} \to \mathbb{R}$ be some long-run average measure of this system expressed as a function of 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 $\gamma(y)$ as described in Equation (13) with $s = -1$. 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 $S_t > C_t$ at some time instants t.

In Figure 2, we show the simulation results obtained with this randomization scheme. 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 ± 3 standard-deviation interval.

We observe that a randomization of the model produces a smooth interpolation of the objective. The standard deviation values (which represent the simulation error) are similar at the discrete and the interpolated points, indicating that the stochastic error contributed by the randomization is negligible. The computational overhead of the embedding, contributed primarily by the additional calls to a random number generator, was found to be 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 $y_0 + 0.5$). This overhead is very small in comparison to the computational cost involved when using spatial interpolation.

Thus a smooth embedding of the queue capacity parameter could be obtained efficiently through a randomization of the simulation model.

The shape of the interpolation curve is sensitive to the choice of the interpolation coefficients $\alpha^1(y), \alpha^2(y)$, and thus the value of the parameter s used for generating these coefficients. For most of the examples considered in this study, we observe that setting $s = 1$ results in a smooth interpolation. For other examples, we have tuned s to obtain a smooth interpolation. It may be possible to arrive at the best randomization settings (the choice of the functions α^k) analytically rather than through tuning. However this is beyond the scope of the current work and can be an interesting direction for future study.

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 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 $\gamma(y)$ defined in Equation(13) with $s = 1$. 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 3, we show the interpolation obtained using this scheme. 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. We observe that a randomization of the

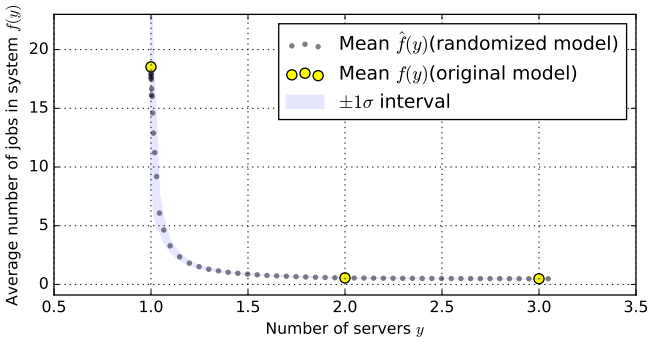


Figure 3: Interpolation results for embedding the number of servers (K) in a Geo/Geo/1/K queue. The embedded parameter $y = K$ is the number of servers and the interpolated function $\hat{f}(y)$ is the average number of jobs in the system. ($p = 0.5$, $q = 0.51$, simulation length$= 10^4$ slots)

model produces smooth interpolations of f in a Geo/Geo/1/K queue. Similarly, for a queue with a deterministic service time (a Geo/D/1/K queue), a smooth interpolation was obtained using the same randomization settings.

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 $\gamma(y)$ defined in Equation (13) with $s = 1$. In Figure 4, we show the interpolation obtained using this randomization scheme. 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 applied over such an embedding.

AN OPTIMIZATION CASE STUDY

To demonstrate the utility of the embedding technique, we present an optimization case study for a queueing network shown in Figure 5. We optimize this queueing network using the embedding-based approach described in this paper and compare its performance to a direct discrete-space search method. Although the focus of this paper is on the embedding technique itself (and not on a comparison between optimization algorithms), the preliminary results obtained for this case-study indicate that descent-based continuous optimizers can be effectively applied over the embedding and may perform significantly better in comparison to discrete-space methods.

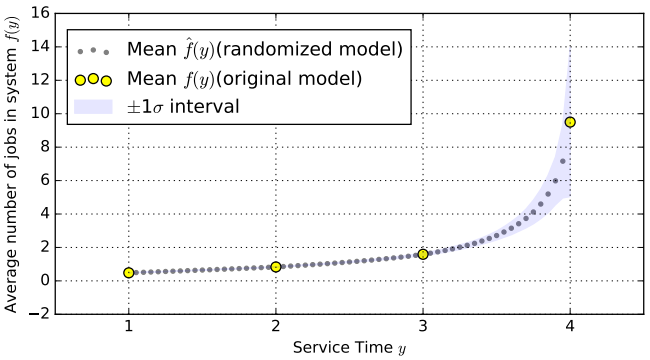


Figure 4: Interpolation results obtained by embedding the service-time in a Geo/D/1 queue. The embedded parameter y is the service time (in slots) and $\hat{f}(y)$ is the average number of jobs in the system. ($p = 0.24$, simulation length$= 10^4$ slots)

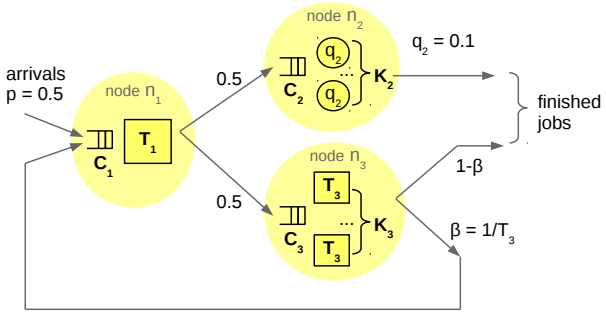
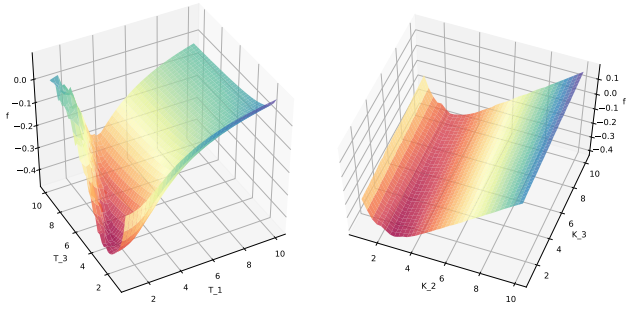


Figure 5: Queueing network to be optimized

Problem Statement: The queueing network in Figure 5 consists of three nodes n_1, n_2 and n_3. Each node n_i has a queue with a finite capacity C_i in front of it. 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. 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 β) 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 arrival probability p and the service parameter q_2 are kept fixed ($p = 0.5$, $q_2 = 0.1$). 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. Thus the parameter



(a) $f(T1, T3)$ (b) $f(K2, K3)$

Figure 6: The interpolated objective function $\hat{f}$ (obtained via simulation of the queueing network in Figure 5) plotted along 2-dimensional slices of the parameter space.

space has 10^7 design points.

Objective Function: Let X denote the vector of parameter values and Ω_D denote the set of all possible values that X can take. We define the objective function $f(X)$ as a weighted sum of throughput and cost components. The throughput component, denoted $T(X)$ is the expected value of the long-run average throughput of the system (estimated using simulation). The cost component, denoted $C(X)$ is modeled using a synthetic function as follows:

$$C(X) = (C_1 + C_2 + C_3) + \frac{20}{T_1} + 100K_2 + 20\frac{K_3}{T_3}. \quad (14)$$

Thus the cost increases with increasing buffer sizes and the number of servers, and reduces with increasing service-times. The objective function to be minimized is a weighted sum of the normalized cost and throughput components, defined as follows:

$$f(X) = \frac{C(X)}{\max\limits_{j \in \Omega_D} C(j)} - \frac{T(X)}{p} \quad (15)$$

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. 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 technique described in the previous section. To understand the nature of the resulting interpolation (in terms of continuity, convexity and smoothness), we plot the interpolated objective along arbitrary 2-dimensional slices of the 7-dimensional domain. Figure 6 shows the interpolated objective $\hat{f}$ plotted along two such slices. The plots were obtained by sweeping two parameters at a time (in steps of 0.25) while keeping the other parameter values fixed. Each point on this plot is obtained via a single simulation of the randomized model of length 10^4 slots. Along each slice, the interpolation is found to be reasonably smooth. The plots obtained along several other slices in the domain were similar, indicating that the interpolation obtained via the randomization is well-suited to the application of continuous-space optimizers.

We evaluate the performance of two continuous optimizers, COBYLA (Powell (1994; 2003)) and SPSA (Spall (1992)) applied *directly* over the randomization-based embedding and compare their performance against a discrete-space version of SPSA (Whitney et al. (2001)) applied over the original discrete-parameter model. While discrete-parameter variants of SPSA exist, COBYLA has not been conventionally applied in the discrete-parameter case. This fact illustrates the utility of our embedding technique, which makes it possible for a large set of existing continuous optimizers to be applied for solving discrete-parameter problems. Both COBYLA and SPSA were chosen as they do not require 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. The settings for each optimizer were selected via tuning and were identical across the continuous and discrete versions of SPSA.

Results: For each optimization 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 (14). At the end of each optimization run, we round the solution to the nearest integer point, and record the objective value at this point. Table 1 presents the performance results for the three methods measured across 100 optimization runs. The results show 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 were found to be well-clustered. (Among the top 20 solutions found by COBYLA, all were found to 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.

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

Table 1: Performance of two continuous optimizers (COBYLA and SPSA) applied directly over the randomization-based embedding, and a discrete-space optimizer (Discrete-SPSA) applied over the original discrete-parameter space.

CONCLUSIONS

This paper presented a simple and computationally efficient technique using which discrete parameters in queueing systems can be embedded into a continuous-space, enabling direct application of continuous-space methods for simulation-based optimization. The technique is based on a randomization of the parameter values in the simulation model and is applicable to problems where the objective function is a long-run average measure. Unlike spatial interpolation, the interpolated value can be measured using a single simulation of this randomized model irrespective of the dimensionality of the design space. We presented a theoretical basis for this embedding technique and described its application to discrete-time queues for embedding queue capacities, number of servers and service-time parameters into a continuous space. We then demonstrated the utility of this embedding technique via an optimization case study of a queueing network with 7 parameters. A randomization of the simulation model produced reasonably smooth and continuous interpolations of the objective. Two continuous-space optimizers (COBYLA, SPSA) applied directly over this embedding were found to perform better in comparison to a direct discrete-space search (Discrete-SPSA).

The shape of the generated interpolation curves is affected by the choice of the randomization settings (in particular, the choice of the functions $\alpha_i^1, \alpha_i^2, \ldots$ used during randomization). For obtaining a smooth interpolation, we have performed a tuning of these randomization parameters in the current work. However, for some systems it may be possible to arrive at the best randomization scheme analytically, and this can be a direction for future investigation. Further, the effect of the perturbation interval on the generated interpolation curves also needs to be investigated further. A limitation of this embedding technique is that it may require a modification of the simulation program. Some state transitions that were not possible in the original (fixed-parameter) model may now become possible when the parameter values are randomized. These transitions have to be accounted-for by the programmer.

In summary, this paper showed that the randomization-based embedding technique can be a useful and effective tool in simulation-based optimization of queueing systems. Future work can extend the embedding technique to other kinds of discrete-event systems such as inventory models.

REFERENCES

Alfa A.S., 2015. *"Applied Discrete-Time Queues"*. Springer Publishing Company, Incorporated, 2nd ed. ISBN 149393418X, 9781493934188.

Bhatnagar S. and Kowshik H.J., 2005. "A Discrete Parameter Stochastic Approximation Algorithm for Simulation Optimization". *SIMULATION*, 81, no. 11, 757–772.

Bhatnagar S.; Mishra V.K.; and Hemachandra N., 2011. "Stochastic Algorithms for Discrete Parameter Simulation Optimization". *IEEE Transactions on Automation Science and Engineering*, 8, no. 4, 780–793. ISSN 1545-5955.

Chen C.H. and Lee L.H., 2010. *"Stochastic Simulation Optimization: An Optimal Computing Budget Allocation"*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1st ed. ISBN 9789814282642, 9814282642.

Karanjkar N.V. and Desai M.P., 2015. "An Approach to Discrete Parameter Design Space Exploration of Multicore 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*. ISSN 1526-7539, 85–88.

Lim E., 2012. "Stochastic Approximation over Multidimensional Discrete Sets with Applications to Inventory Systems and Admission Control of Queueing Networks". *ACM Trans Model Comput Simul*, 22, no. 4, 19:1–19:23. ISSN 1049-3301.

Powell M., 2003. "On Trust Region Methods for Unconstrained Minimization without Derivatives". *Mathematical Programming*, 97, no. 3. ISSN 0025-5610.

Powell M.J.D., 1994. *"A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation"*, Springer Netherlands, Dordrecht, chap. 4. ISBN 978-94-015-8330-5, 51–67.

Schweitzer P.J., 1968. "Perturbation Theory and Finite Markov Chains". *Journal of Applied Probability*, 5, no. 2, 401–413. ISSN 00219002.

Spall J.C., 1992. "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation". *IEEE Transactions on Automatic Control*, 37, no. 3, 332–341.

Swisher J.R.; Hyden P.D.; Jacobson S.H.; and Schruben L.W., 2000. "A Survey of Simulation Optimization Techniques and Procedures". In *Simulation Conference, 2000. Proceedings. Winter*. vol. 1, 119–128 vol.1.

Wang H. and Schmeiser B.W., 2008. "Discrete Stochastic Optimization using Linear Interpolation". In *2008 Winter Simulation Conference*. IEEE. ISBN 978-1-4244-2707-9, 502–508.

Whitney J.E.; Solomon L.I.; and Hill S.D., 2001. "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, 313–317 vol.1.

WEB REFERENCES

Karanjkar N., 2017. *Online repository of queueing models and scripts used for the embedding experiments*. `https://github.com/NehaKaranjkar/Embedding`.