Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

MATHEMATICAL PROGRAMMING REPRESENTATIONS FOR STATE-DEPENDENT QUEUES

Wai Kin (Victor) Chan

Department of Decision Sciences
and Engineering Systems
Rensselaer Polytechnic Institute
CII 5015, 110 Eighth Street
Troy, NY 12180 U.S.A.

Lee W. Schruben

Department of Industrial Engineering
and Operations Research
University of California, Berkeley
4135 Etcheverry Hall
Berkeley, CA 94720 U.S.A

ABSTRACT

Discrete-event dynamic systems with feedback, where the behavior of the system depends on the system state, are difficult to model due to the uncertainties and dependencies of system performance on the system state. Service systems, in particular, tend to exhibit this behavior where servers may work faster (or slower) when facing an increasingly long line of impatient customers. A common example is a state-dependent queue where the service rate depends on the queue size, which can change during service. In this paper, we present a mathematical programming representation for the sample path dynamics of a state-dependent queue, and illustrate its application in sensitivity analysis.

1 INTRODUCTION

We consider a queue with single-server whose service rate depends on the queue length, called a state-dependent queue (SDQ) with notation $G/G(Q)/1$. SDQs are realistic models for discrete-event dynamic systems with a human server, whose speed may be affected by the number of jobs or customers waiting in line. These types of queueing systems have been studied by many researchers, see for example, Courtois and Georges (1971), Cheng (1993), Macgregor Smith (1994), Gupta and Srinivasa Rao (1998), Chao and Rahman (2006), Parthasarathy and Sudhesh (2007).

The goal of this paper is to develop a mathematical programming representation (MPR) for the sample path dynamics of SDQs. MPR is a mathematical programming-based technique for modeling discrete-event system (DES) dynamics as the solutions to optimization models. A DES changes its state in accordance with the occurrence of events. The trajectory of a discrete-event system, therefore, consists of a series of (state) marked event-occurrance times. Simulating such a system will give a realization of its state trajectory. However, finding the state trajectory does not necessary require simulating the underlying system in the traditional way. Modeling system state trajectories as the solutions to an optimization problem is another way of observing the system dynamics. Schruben (2000) introduced this approach, showing how MPRs can model simple queueing dynamics. Chan and Schruben (2003) derived the MPRs for open tandem queueing networks under communication blocking. Chan and Schruben (2008b) provides a procedure for representing a discrete-event system as a mathematical programming model and illustrate several applications of the optimization models, including the use of the MPR as an alternative proof for the reversibility property of single-server open tandem queues under communication and production blocking. Chan and Schruben (2008a) gives a set of MPRs for closed tandem queueing networks under various types of blocking scenarios and use these MPRs to establish properties of closed queue networks, such as reversibility, symmetry, and reverse-symmetry. Here we take a different approach that, while involving a bit more algebra, provides additional insights into the behaviors of state-dependent queues depending on how the server responds to changes in line length.

This paper focuses on single-server, state-dependent queues, and develops an MPR, called SDQ-LP, for some special cases of these systems. The derivation of the SDQ-LP optimization model consists of two steps. The first step is to develop several equations for the service time in a SDQ and establish the convexity property of the service time, under certain assumptions about server behavior. The second step is to derive the constraints from a $G/G/1$ queue simulation model from the equations for the service time, making use of the convexity property of the service time.

The rest of the paper is organized as follows. Section 2 introduces equations for computing the service time in a SDQ and introduces a convexity property of the service time under assumptions on whether the server will work faster, or will work slower, as the queue grows. Section 3 derives the MPR formulation of the SDQ using this convexity property. Applications of the SDQ-LP are suggested in Section 4. The paper is concluded in Section 5. Due to space limitation, proofs of the lemmas and theorems are omitted, and should be regarded here as conjectures.

2 BACKGROUND

We consider a queueing system with a general independent arrival process where each arriving customer or job has a general iid service requirement. There is one server, whose processing speed depends on the waiting line size, *Q*(*t*), and infinite waiting space. An important ingredient of this system is that the processing time of jobs depends on the queue size, i.e., the server may increase or decrease its speed when there are more or less jobs the queue. Specifically, the job processing speed is according a deterministic service rate function:

$$\mu(t) = f(Q(t)) \tag{1}$$

In a DES, since the queue size only changes at discrete instants, Eq.(1) can be rewritten as:

$$\mu(t) = f(Q_t), \quad t = t_1, t_2, \ldots$$

where $t_1, t_2, \ldots$ are times when the queue size changes (e.g., due to an arrival or departure of a job). Define μ_k as the service rate when there are k customers in queue.

We will somewhat realistically assume that the server is either motivated to work faster when the queue increases, or becomes discouraged and actually works slower when facing a growing queue. If the service rate increases with the queue size, then we say the service process has an increasing service rate (ISR). A simple example is where the service rate linearly increases with queue size. This has the following function for the service rate:

$$\mu_{Q_t} = (Q_t + 1)\mu_0, \quad t = t_1, t_2, \ldots$$

where μ_0 is the base service rate. For an ISR server, μ_0 is the service rate when there is only one job in the system that is currently being served, and there are no jobs waiting ($Q_t = 0$).

The second type of state-dependence we assume is that the service rate decreases when the queue size gets bigger. This type of service process will be called a decreasing service rate (DSR) server. An example is a linearly decreasing service rate:

$$\mu_{Q_t} = \mu_{\max} - Q_t\mu_0, \quad t = t_1, t_2, \ldots$$

where μ_0 is again the base service rate and $\mu_{\max}$ is the maximum service rate, which applies when the queue is empty. Because the service rate has to be strictly positive, this definition requires that $\mu_{\max}$ be larger than $Q_t\mu_0$ during the entire simulation. To ensure this, we assume that $\mu_{\max} = n\mu_0$, where n is the total number of simulated jobs specified in the simulation stopping condition. This leads to

$$\mu_{Q_t} = (n - Q_t)\,\mu_0, \quad t = t_1, t_2, \ldots.$$

As we will see in the following, defining the maximum service rate as a function of the total simulated jobs greatly simplifies our derivation. One can, however, redefine the service rate as, for example, $\max\{\mu_{\max} - Q_t\mu_0, \mu_0\}$ to allow the maximum service rate to be some arbitrary constant. The derivation of the corresponding LP would then be slightly different.

For notational convenience, we define ν_{kl} as the difference between service rate μ_k and μ_l, i.e., $\nu_{kl} = \mu_k - \mu_l$, $k, l \in \{0, 1, \ldots, n | k > l\}$.

The service rate is a monotonic increasing function in the ISR system and a monotonic decreasing function in the DSR system. Next, we show that these monotonic properties lead to useful convexity properties of the service time.

We next derive a set of sample path equations to model the service times. The service requirement of job i is a realization, d_i, of a random variable from a distribution that is identical and independent from all other jobs' service requirements. The service time for job i depends on its service requirement as well as on the service rate, which will change with queue size, which in turn is a function of the arrival times and finish times of other jobs.

For example, if the system is empty when job i arrives and no other jobs arrive during job$i$'s entire service, then the service time of job i would simply be d_i/μ_0. However, if one and only one job (i.e., job $i+1$) arrives during job i's service, then the service time of job i would become $a_{i+1} + (d_i - a_{i+1}\mu_0)/\mu_1$, where a_{i+1} is the inter-arrival time between job i and job $i+1$. The service time of job $i+1$ would then depend on the finish time of job i and also on subsequent jobs' arrival times.

Let k_i be the number of jobs that arrive during the service of job i. Denoted by $s_{ij}^{k_i}$ the service time of job i when there are k_i jobs arrive during its service with the first one to arrive being job$j$.

If job i initiates a busy period, then the queue size is zero when it arrives and k_i is the queue size when job i leaves. Let $sb_i^{k_i}$ be the service time of job i when it initiates the busy period containing k_i jobs, exclusively. The following definition gives an expression for $sb_i^{k_i}$.

Definition 1: *Define the following set of formulae:*

$$\begin{cases} sb_i^0 = & d_i/\mu_0, \\ sb_i^k = & (d_i + \sum_{l=0}^{k-1} a_{i+1+l}\nu_{kl})/\mu_k, \\ & k \in \{1, \ldots, n-i\}, \end{cases} \tag{2}$$

for $i \in \{1,\ldots,n\}$. *The* service time of job i ($sb_i^{k_i}$) initiating a busy period consisting of k_i jobs, *exclusively, can be determined by:*

$$sb_i^{k_i} = \begin{cases} sb_i^0 & \text{if } k_i = 0, \\ sb_i^k & \text{if } k_i = k \in \{1,2,\ldots,n-i\}, \end{cases}$$

for $i \in \{1,\ldots,n\}$.

In order to use Eq.(2) to find the service time, $sb_i^{k_i}$, it appears that one needs to know k_i. Moreover, knowing k_i might require as much effort as running the entire simulation. However, this is not the case since Eq.(2) is convex [concave] in k_i for a ISR [DSR] system.

More important, the minimum [maximum] of these service time values in k_i equals to the true value of $sb_i^{k_i}$. This property will allows us to drop the superscript k_i from $sb_i^{k_i}$ and use sb_i to denote the service time of job i initiating a busy period, regardless how many jobs arrive during its service time. The next lemma formally documents this.

Lemma 1: *Given* d_i *and a set of* $\{a_{i+1}, a_{i+2}, \ldots, a_n\}$, *the formulae specified in Definition* 1 *are convex* [*concave*] *in k for an ISR* [*DSR*] *system and the service time of job i* (sb_i) *initiating a busy period is:*

$$sb_i = \min_{\forall k}\{sb_i^k\},$$
$$\left[sb_i = \max_{\forall k}\{sb_i^k\}, \right]$$

for $i \in \{1,\ldots,n\}$.

Because of this convexity property, we can avoid evaluating all equations when finding the minimum or maximum, say by using a golden section search when simulating the system. As shown in Section 3, this convexity property is much more valuable in an MPR model than when running a simulation.

If job i arrives when another job is being served, its service time will depend on the finish time of the job preceding it (i.e., job $i-1$) as well as the arrival times of the jobs arriving during its service. Let $sf_{ij}^{k_i}$ be the service time of job i when the server is busy at the time job i arrives and there are k_i jobs arriving during job i's service with the first arriving job being job j. Let A_i and F_i be the arrival time and finish time of the i^{th} job, respectively. The next definition describes the computation of $sf_{ij}^{k_i}$.

Definition 2: *Define the following set of formulae:*

$$\begin{cases} sf_{ij}^0 = & d_i/\mu_{j-i-1}, \\ sf_{ij}^1 = & (d_i + (A_j - F_{i-1})\nu_{j-i,j-i-1})/\mu_{j-i}, \\ sf_{ij}^k = & (d_i + (A_j - F_{i-1})\nu_{j-i-1+k,j-i-1} + \\ & \sum_{l=1}^{k-1} a_{j+1}\nu_{j-i-1+k,j-i-1+l})/\mu_{j-i-1+k}, \\ & \qquad k \in \{2,3,\ldots,n-j+1\} \end{cases} \tag{3}$$

for $i \in \{2,3,\ldots,n-1\}$ *and* $j \in \{i+1,\ldots,n\}$. *The* service time of job i ($sf_{ij}^{k_i}$) who arrives at a non-empty queue and sees k_i arriving jobs during its service, with the first arriving job being job j, *can be computed as:*

$$sf_{ij}^{k_i} = \begin{cases} sf_{ij}^0 & \text{if } k_i = 0, \\ sf_{ij}^1 & \text{if } k_i = 1, \\ sf_{ij}^k & \text{if } k_i = k \in \{2,3,\ldots,n-j+1\}, \end{cases}$$

for $i \in \{2,3,\ldots,n-1\}$ *and* $j \in \{i+1,\ldots,n\}$.

As in Lemma 1, the formulae in Definition 2 also exhibit the convexity property for ISR systems, which allows us to find the service time for job i without knowing how many jobs arrived during job i's service period. Define sf_{ij} as the service time of job i when it arrives at a busy system and the first job arrives during its service period is job j.

Lemma 2: *Given* d_i *and a set of* $\{a_j, a_{j+1}, \ldots, a_n\}$, *the formulae specified in Definition* 2 *is convex* [*concave*] *in k for the ISR* [*DSR*] *system and the service time of job i* (sf_{ij}) *entering a busy system and the first job arrives during its service period is job* j *is:*

$$sf_{ij} = \min_{\forall k}\{sf_{ij}^k\},$$
$$\left[sf_{ij} = \max_{\forall k}\{sf_{ij}^k\}, \right]$$

for $i \in \{2,3,\ldots,n-1\}$ *and* $j \in \{i+1,\ldots,n\}$.

One issue in using Lemma 2 is that it requires the knowledge the first job (job j) arriving during job i's service period. In the next lemma, we will show that sf_{ij} possesses another concavity property that allows us to compute its value without knowing the actual identity of job j. Define sf_i as the service time of job i when it arrives at a busy system.

Lemma 3: $sf_{ij}, j = i+1,\ldots,n$ *given in Lemma* 2 *is concave* [*convex*] *in j for the ISR* [*DSR*] *system and the service time of job i* (sf_i) *entering a busy system is:*

$$sf_i = \max_{\forall j}\{sf_{ij}\},$$

$$\left[sf_i = \min_{\forall j}\{sf_{ij}\}, \right]$$

for $i \in \{2, 3, \ldots, n-1\}$.

Since we assume that the simulation stops when it simulates a total of n jobs, the last element in computing the service time is to deal with the possibility that all n simulated jobs arrived earlier than job i's service initiation. Let sl_i be the service time under this situation. The service time under this situation is given in the next lemma.

Lemma 4: *The service time of job i when all n jobs arrived earlier than its service inception can be computed as:*

$$sl_i = d_i/\mu_{n-i}, \tag{4}$$

for $i \in \{2, 3, \ldots, n-1\}$.

With all the above equations, the service time of job i can be determined by the following theorem. Let s_i be the service time of job i.

Theorem 1: *The service time of job i in a $G/G(Q)/1$ queue is given by:*

$$s_i = \min\left\{ \min_{\forall k}\left\{sb_i^k\right\}, \max\left\{ \max_{\forall j}\left\{ \min_{\forall k}\left\{sf_{ij}^k\right\}\right\}, sl_i \right\}\right\}$$

for the ISR system, and

$$s_i = \max\left\{ \max_{\forall k}\left\{sb_i^k\right\}, \min\left\{ \min_{\forall j}\left\{ \max_{\forall k}\left\{sf_{ij}^k\right\}\right\}, sl_i \right\}\right\}$$

for the DSR system, for $i \in \{1, \ldots, n\}$.

3 FORMULATION

The equations given in Theorem 1 are max-plus recursions, which can be mapped as linear constraints as described in Chan (2005). In this section, we use the equation for the DSR system to illustrate how to derive an LP for a SDQ. The derivation for the ISR system is similar.

The derivation starts with a simulation model for a *G*/*G*/1 queue where the service rate is constant. Figure 1 shows one of the many possible simulation models for a *G*/*G*/1 queue. Here we use the language of simulation event relationship graphs (ERGs) to define the system dynamics. ERGs are a general, minimalist means of explicitly expressing all the dynamic causal relationships between events in a discrete-event dynamic system model (see Schruben 1983, Law 2006 for more details). Other simulation modeling techniques can also be used to derive the corresponding LPs.

The ERG in Figure 1 is interpreted as follows. There are two events: *Arrival* (A) of jobs and *Finish* (F) of services. Unconditional arc (A, A) means that once a job arrives, the next one is scheduled to arrive after a delay of a, which is a random variable with its realization, a_i, being the inter-arrival time. Conditional arc (A, F) says that when a job arrives, if the server is idle, it can start its service and will leave the system after a service delay of s, which is a random variable with its realization, s_i, being the service time of the ith job. Conditional arc (F, F) ensures that once the server becomes available due to the departure of a finish job, it will serve the next job immediately, provided that there is at least one job waiting in line, and this job will also leave after its service delay of s, which is the same random variable as the one on Arc (A, F).

The MPR for this ERG that generates the same sample path for a given data set of $\{(a_i, s_i), i = 1, \ldots, n\}$ for n jobs is the LP shown in GG1-LP. Since the exogenous Arrival event times are all determined by the data, the sample path is simply the one that finishes all jobs as early as feasible.

<u>GG1-LP:</u>

$$\min \sum_{\forall i} F_i$$

Subject to:

$$F_i - A_i \geq s_i, i = 1, \ldots, n$$
$$F_i - F_{i-1} \geq s_i, i = 2, \ldots, n$$

and all variables are positive.

We now modify the *G*/*G*/1 ERG by replacing the independent service times $s_i, \forall i$ with the ones specified in Theorem 1. This extends the *G*/*G*/1 ERG to a *pseudo* SDQ ERG. We call this modified ERG a pseudo SDQ, because it cannot actually be simulated in the usual "next event scheduling" manner because it requires future information (i.e., future arrival times) to compute the service times. Nevertheless, the need for future information is not a problem for the corresponding LP model because all the necessary data is available to the LP (see more discussion later).

Since this pseudo ERG model for the SDQ has the same structure as the *G*/*G*/1 ERG model, the two constraints in the GG1-LP MPR model will also be needed in the SDQ-LP. In addition, we need to incorporate the equations in Theorem 1 into additional constraints on the service times. The resulting SDQ-LP is given at the end of this section.

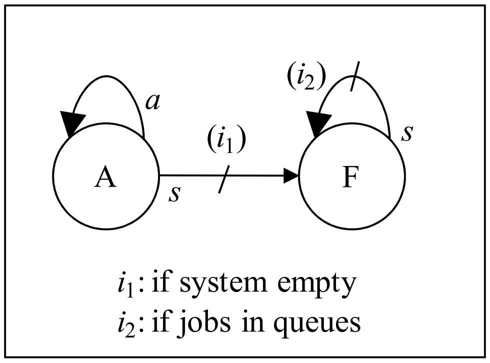

Figure 1: ERG Simulation Model of *G*/*G*/1 Queue

We add slack variables to express all constraints as equalities. All the notation for slack variables follows the same format: y^{α}_{β}, where α represents the original variable in the corresponding constraint and β is the subscript (these could be multiple subscripts) of the original variable and the corresponding constraint. For example, in Constraint (12) at the end of this Section, the slack variable is y^F_{i1}, where F represents the original variable F_i in that constraint and subscript $i1$ represents that this is the first ("1") constraint with the original variable F_i.

The two constraints in GG1-LP appear in SDQ-LP as Constraints (12) and (13), with slack variables added. We now derive the constraints for the service times. From Lemma 2, sb_i is the maximum of all sb_i^k, yielding,

$$sb_i \geq sb_i^k, \tag{5}$$

$i = 1, \ldots, n; k = 0, 1, \ldots, n - i.$

From Lemma 4, sf_{ij} is the maximum of all sf_{ij}^k, yielding,

$$sf_{ij} \geq sf_{ij}^k, \tag{6}$$

$i = 2, \ldots, n-1; j = i+1, \ldots, n; k = 0, \ldots, n-j+1.$

From Lemma 5, sf_i is the minimum of all sf_{ij}, yielding,

$$sf_i \leq sf_{ij}, \tag{7}$$

$i = 2, \ldots, n-1; j = i+1, \ldots, n.$

In the service time expression of the DSR system specified in Theorem 1, the second argument of the first maximum function is the minimum between sf_i and sl_i. Define sfl_i as this minimum, we have,

$$sfl_i \leq sf_i, \quad \text{and} \tag{8}$$

$$sfl_i \leq sl_i, \tag{9}$$

$i = 2, \ldots, n-1.$

Finally, again from Theorem 1 the service time s_i is the maximum of sb_i and sfl_i, hence,

$$s_i \geq sb_i, \tag{10}$$

$i = 1, \ldots, n$, and

$$s_i \geq sfl_i, \tag{11}$$

$i = 2, \ldots, n-1.$

Inequalities (5), (6), (10), and (11) suggest that the objective function of the LP should act as an mechanism to push sb_i, sf_i, and s_i down to the maximum of their corresponding right-hand-side, while inequalities (7), (8), and (9) call the objective function to maximize sf_i and sfl_i by pulling it up to the minimum of their respective right-hand-sides, provided that enough incentive is given to hold all the right-hand-sides unchanged during the course of minimization or maximization.

One simple objective function might be the following.

$$\min_{\forall i} \left(sb_i + \sum_{\forall j} sf_{ij} - sf_i - sfl_i + s_i \right)$$

However, this simple objective function does not produce the desired result because minimizing some of the variables might conflict with the goal of maximizing other variables. For instance, maximizing sf_i in Inequality (7) induces the inclination to have bigger sf_{ij}'s, which contradicts the goal of minimizing sf_{ij} suggested in Inequality (6).

Simply multiplying the variables with some coefficients in the objective function does not work because these coefficients would depend on the problem data and determining their values might require as much effort as running the entire simulation.

This problem has also been studied in Chan (2005), where several solution approaches have been suggested. However, all the solutions described there require either solving an LP sequentially or introducing a large number of extra constraints. In the following, we present a new technique to solving this problem

This technique is to transform the LP into a certain form so that its optimal solution will be identical to the simulation trajectories. This transformation consists of two steps. The first step is to change all constraints into equalities by adding slack variables. The second step is to define an objective function of minimizing all slack variables scaled by coefficient, c^{n-i}, where c is some constant, i is the index of the job that the variable is associated with, and

n is the total number of simulated jobs. In this paper, we select $c = 10$.

After performing the transformation on the above constraints along with the suggested objective function, we obtain an LP with the optimal solution identical to the simulation trajectory of a SDQ. This equivalence is formally stated in Theorem 2 after the SDQ-LP.

SDQ-LP:

$$\min \sum_{\forall i} c^{n-i} \left(y_{i1}^F + y_{i2}^F + \sum_{\forall k} y_{ik}^{sb} + \sum_{\forall j,k} y_{ijk}^{sfj} \right.$$
$$\left. + \sum_{\forall j} y_{ij}^{sf} + y_{i1}^{sfl} + y_{i2}^{sfl} + y_{i1}^{s} + y_{i2}^{s} \right)$$

Subject to:

$$F_i - A_i - y_{i1}^F = s_i, i = 1, \ldots, n \tag{12}$$
$$F_i - F_{i-1} - y_{i2}^F = s_i, i = 2, \ldots, n \tag{13}$$
$$sb_i - y_{ik}^{sb} = sb_i^k, i = 1, \ldots, n; k = 0, \ldots, n-i \tag{14}$$
$$\begin{aligned} sf_{ij} - y_{ijk}^{sfj} &= sf_{ij}^k, i = 2, \ldots, n-1; \\ &j = i+1, \ldots, n; k = 0, \ldots, n-j+1 \end{aligned} \tag{15}$$
$$sf_i + y_{ij}^{sf} = sf_{ij}, i = 2, \ldots, n-1; j = i+1, \ldots, n \tag{16}$$
$$sfl_i + y_{i1}^{sfl} = sf_i, i = 2, \ldots, n-1 \tag{17}$$
$$sfl_i + y_{i2}^{sfl} = sl_i, i = 2, \ldots, n-1 \tag{18}$$
$$s_i - y_{i1}^s = sb_i, i = 1, \ldots, n \tag{19}$$
$$s_i - y_{i2}^s = sfl_i, i = 2, \ldots, n-1 \tag{20}$$

and Eqs. (2), (3), and (4). All variables are positive.

Theorem 2: *The optimal solution $\{F_i, i = 1, \ldots, n\}$ of SDQ-LP is identical to the finish times of a SDQ simulation model.*

Although the pseudo ERG simulation model and its MPR model, SDQ-LP, model the same system, there is a fundamental distinction between them. An MPR model sees the entire system trajectory, including the past, present, and future, while the pseudo ERG simulation model sees only the present system state and the currently scheduled events. Using the equations in Theorem 1 in a conventional simulation would require knowledge of future job arrival times to calculate the service times. A different conventional ERG model that simulates an SDQ without looking ahead or backtracking is given in Savage and Schruben (1995).

4 APPLICATIONS

The LP model of a SDQ has opened up an application window for using techniques of mathematical programming in the analysis of the SDQ. Similar applications appeared in the development of MPR models for other queueing systems, such as tandem queue with blocking (Chan and Schruben (2008a)). These applications include finding the perturbation limit of the random variables and finite perturbation analysis, for example, the perturbation limit of G/G/1 queues (Chan and Schruben (2006)). In this section, we discuss the application of finding the perturbation limit of the random variables.

Given a sample path, there exists a limit on the changes of the random variables (or input parameters) beyond which the perturbed sample path can be different from the original sample path either temporary or permanently.

First, we transform SDQ-LP into standard form by moving all variables to the left-hand-side and all A_i and s_is to the right-hand-side. Let $\boldsymbol{s}(\lambda, \mu_0)$ be the vector of right-hand side random variables (including both A_i and s_i, $i = 1, \ldots, n$), which depend on parameters λ (parameter of the arrival process) and μ_0. When μ_0 changes $\mu_0 + \Delta\mu_0$, $\boldsymbol{s}$ becomes $\boldsymbol{s} + \Delta\boldsymbol{s}$. The objective function will change, but the dual variables will remain unchanged so long as the following primal feasible condition is satisfied,

$$\boldsymbol{B}^{-1}\boldsymbol{s} + \boldsymbol{B}^{-1}\Delta\boldsymbol{s} \geq \boldsymbol{0},$$

where $\boldsymbol{B}$ is the square matrix of the constraint matrix corresponding to the basic variables. Hence, perturbation limit is given by

$$\min \left\{ \left(\boldsymbol{x}_B + \boldsymbol{B}^{-1}\Delta\boldsymbol{s} \right)_j \geq 0, j = 1, \ldots, m \right\},$$

where $\boldsymbol{x}_B$ is the set of basic variables and m is the number of constraints.

5 CONCLUSION AND FUTURE WORK

MPRs of discrete-event system dynamics allow the integration of techniques in mathematical programming and discrete-event simulation. This paper shows how the difficult problem of simulating state-dependent systems can be modeled using math programs. The key is that an MPR has much better vision than a conventional next-event simulation—it sees the past, the present, and the future simultaneously so that it can make adjustments to its constraints to deal with the uncertain and data-dependent precedent relationships among events. Of course, such a super-vision comes with a computational price. The hope is that an MPR model and a simulation together can provide a better tool kit for studying such systems.

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation through grant CMMI-0644959 to Rensselaer Polytechnic Institute and grant DMI-0323765 to the University of California, Berkeley.

REFERENCES

Chan, W. K. V. 2005. *Mathematical Programming Representations of Discrete-Event System Dynamics.* Ph.D. dissertation, University of California, Berkeley.

Chan, W. K. V. and L. Schruben. 2006. Response Gradient Estimation Using Mathematical Programming Models of Discrete-Event System Sample Paths. In *Proceedings of the 2006 Winter Simulation Conference* (IEEE Cat No. 06CH37826), ed. Ingalls, R.G., M. D. Rossetti, J. S. Smith and B. A. Peters, 272 - 278. Piscataway, N.J.:IEEE.

Chan, W. K. V. and L. Schruben. 2008a. Mathematical Programming Models of Closed Tandem Queueing Networks. *ACM Transactions on Modeling and Computer Simulation (to appear).*

Chan, W. K. V. and L. Schruben. 2008b. Optimization Models of Discrete-Event System Dynamics. *Operations Research (to appear).*

Chao, X. and A. Rahman. 2006. Analysis and Computational Algorithm for Queues with State-Dependent Vacations I: G/M(N)/1/K. *Journal of Systems Science and Complexity* **19**(1): 36-53.

Cheng, D. W. 1993. Analysis of a Tandem Queue with State-Dependent General Blocking: A Gsmp Perspective. *Performance Evaluation* **17**(3): 169-76.

Courtois, P. J. and J. Georges.1971. On a Single-Server Finite Queueing Model with State-Dependent Arrival and Service Processes. *Operations Research* **19**(2): 424-435.

Gupta, U. C. and T. S. S. Srinivasa Rao. 1998. On the Analysis of Single Server Finite Queue with State Dependent Arrival and Service Processes: M(N)/G(N)/1/K. OR *Spektrum* **20**(2): 83-9.

Law, A. 2006. Simulation Modeling and Analysis, McGraw-Hill.

Macgregor Smith, J. 1994. Application of State-Dependent Queues to Pedestrian/Vehicular Network Design. *Operations Research* **42**(3): 414-27.

Parthasarathy, P. R. and R. Sudhesh. 2007. Time-Dependent Analysis of a Single-Server Retrial Queue with State-Dependent Rates. *Operations Research Letters* **35**(5): 601-11.

Savage, E. and L.W. Schruben. 1995, Eliminating Event Cancellation in Discrete Event Simulation, *Proc. 1995 Winter Simulation Conference*, Alexandria, VA, December 3-6: 744-750.

Schruben, L. 1983. Simulation Modeling with Event Graphs. *Communications of the ACM* **26**(11): 957-63.

Schruben, L. 2000, Mathematical Programming Models of Discrete Event System Dynamics, *Proc. of the 2000 Winter Simulation Conference*, December 10-13: 381-6.

AUTHOR BIOGRAPHIES

WAI KIN (VICTOR) CHAN is an Assistant Professor of the Department of Decision Sciences and Engineering Systems at Rensselaer Polytechnic Institute. He received a bachelor's degree and a master's degree in electrical engineering from, respectively, Shanghai Jiao Tong University, China in 1997, and Tsinghua University, China in 2000, and a M.S. and Ph.D. degrees in industrial engineering and operations research from University of California, Berkeley in 2001 and 2005, respectively. His research interests include discrete-event simulation, agent-based simulation, mathematical programming, and their applications in energy markets, social networks, biological systems, service systems, transportation networks, and manufacturing. His e-mail address is <chanw@rpi.edu>.

LEE W. SCHRUBEN is the Chancellor's Professor of the Department of Industrial Engineering and Operations Research at the University of California, Berkeley. His research is on discrete-event simulation modeling and analysis methodologies and a variety of applications – most recently in the area of bio-production. His e-mail address is <schruben@ieor.berkeley.edu>.