OPTIMIZATION MODELS OF DISCRETE-EVENT SYSTEM DYNAMICS

Wai Kin (Victor) Chan[*]
Lee Schruben[**]

[*] Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, CII 5015, 110 Eight Street, Troy, NY 12180
[**] Department of Industrial Engineering and Operations Research, University of California, Berkeley, 4135 Etcheverry Hall, Berkeley, CA 94720

ABSTRACT

A methodology is given for modeling the dynamics of discrete-event stochastic systems as optimization problems. The intent is to provide a means to utilize the rich mathematical theory and algorithms of optimization in the study of this important class of systems. A procedure for mapping a simulation event relationship graph into a mixed-integer program is presented along with examples of queueing networks and manufacturing systems that illustrate the approach. Several potential applications are examined including automatic constraint generation for optimal resource scheduling, representations of max-plus algebra models for queueing system dynamics, response gradient estimation, and an unconventional technique for simulating queueing systems using virtual resources that are identified from the optimization models for these systems.

1 INTRODUCTION

The goal of this paper is to produce a methodology for modeling stochastic discrete-event dynamic systems as optimization problems. This allows the rich theory and algorithms of mathematical programming to be applied to the modeling and analysis of such systems, which include many queueing networks as well as certain manufacturing, transportation, and communications systems.

1.1 Background

Schruben (2000) proposed modeling sample paths from event relationship graphs (ERGs) used to simulate discrete-event system dynamics as the solutions to mathematical programming models. Chan and Schruben (2003) derived mathematical formulations for queueing networks. In this paper, we provide a procedure that generates these optimization models directly from the explicit event relationships in a simulation model and examine several examples and potential applications. The optimization representations model simulation sample paths as the solutions to mathematical programs; they are still simulation models but their mathematical representations allow us to obtain more information from each run (realization) of the simulation experiment (see applications in Section 3).

Analogous to using a system of differential equations for modeling continuous system dynamics, an ERG is a system of difference equations that, along with the initial and terminating boundary conditions, completely specifies the behavior of a discrete-event system. The dynamics of many of these simulations can be modeled as linear programs (LPs) and mixed-integer pro-

grams (MIPs) using the procedure presented in this paper. The variables in these optimization programs are the event times. Causal relationships between simulation events are explicit in its ERG, which imposes timing and logical constraints on the event times. The objective of the mathematical program, as in the simulation, is simply to execute events as early as possible. The input for the simulation and corresponding optimization program are identical and includes inter-event times (such as customer interarrival times and service times in a queue) as well as information such as job routings and batch sizes.

Perhaps the first reference on modeling queueing system dynamics as network optimization programs can be found in Maxwell and Wilson (1981). Baccelli et al. (1992), and Cohen, Gaubert and Quadrat (1999) also provide a linear theory for certain discrete-event systems (e.g., timed event graph Petri nets—very different from ERGs) using max-plus algebra and similar algebraic tools, e.g., semirings or dioids (Gondran and Minoux 1984). Using the mapping provided in Schruben and Yucesan (1994), one can translate stochastic timed Petri nets into more general ERGs and thus model Petri Net dynamics as the solutions to optimization programs using the procedure given in this paper.

The benefits to formulating and simulating discrete-event dynamic systems as optimization models include 1) a methodical approach to deriving dynamic equations for studying the properties of queueing systems, 2) a systematic method for constraint generation when formulating optimal resource scheduling models, 3) alternative simulation models that have no obvious representation using current simulation world views, and 4) an alternative representation of infinitesimal perturbation gradient estimators using duality that may unify or enrich this set of simulation sensitivity analysis techniques. In addition, there are some interesting discrete optimization problems with special structure that arise when modeling discrete-event stochastic systems in this manner, e.g., an explicit duality between the classical lot-sizing problem and a $G/G/1$ queue.

The paper is organized as follows. The rest of Section 1 gives an overview on ERG models as well as the intuition behind representing ERGs as mathematical programs. The general methodology is described in Section 2 while the technical details of the approach are provided in Appendix A. Motivating applications are illustrated in Section 3. Section 4 concludes the paper with some observations and suggestions for further research.

1.2 Event Relationship Graph Models

Event relationship graphs (ERGs) are a general, minimalist means of explicitly expressing all the relationships between events in a discrete-event dynamic system model (Schruben 1983, Pegden 1986, Som and Sargent 1990, Wu and Chung 1991, Askin and Standridge 1993, Law and Kelton 2000, Seila, Ceric and Tadikamalla 2003).

The vertices of an ERG represent state changes that take place when a particular type of event occurs. The directed arcs of the graph represent the relationships between pairs of events. The state changes associated with each event vertex appear in braces. Labels on directed arcs—representing all the dynamic and logical relationships between events—specify the conditions and time delays between the occurrences of events. Following the ERG notation in Askin and Standridge (1993), a generic arc in an ERG is shown in Figure 1.1 and defined as follows: *after event A occurs, if condition i_{AB} is then true, event B will immediately be scheduled to occur t_{AB} time units into the future.*

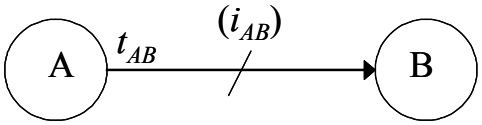


Figure 1.1 : Basic Element of an Event Relationship Graph (ERG)

A simple batch processing queueing system with R_0 identical parallel servers that processes jobs in batch sizes of b will be used for illustration in this paper. One of several possible ERGs for this system is shown in Figure 1.2 where the state is described by two integers: R = the number of currently idle resources and Q = the number of jobs currently waiting in line for service. The input data for simulating the ERG are the (random) customer interarrival times, a, and the (random) service times, s.

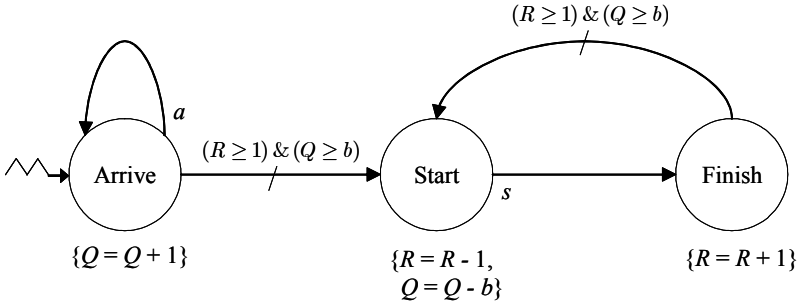


Figure 1.2 : ERG for a Batch Processing Queue with Parallel Resources
State: Q = number of waiting jobs, R = number of idle resources
Data: b = batch size, s = (random) service times, a = (random) interarrival times

The queue in Figure 1.2 is assumed to be initially empty with all the servers idle. Therefore, the initial value of the number of available resources, R, is the total number of parallel servers in the system, R_0, and Q is initially set equal to zero. Initially scheduled events are indicated by a broken arrow as in Law and Kelton (2000). The only event initially scheduled here is the first job to arrive at time zero. The "&" is used here to denote the Boolean "AND" operator.

We will give a formal treatment of the translation of an ERG into an optimization model in Section 2. To give a general idea of the method, we first present a simple example. The following notation for queueing networks will be used throughout the paper. For $i = 1,\ldots,n$,

a_i: time interval between the i^{th} external arrival and the i-1^{th} external arrival
s_{ki}: time interval needed for the i^{th} service at stage k
A_{ki}: the time of the i^{th} external arrival event occurrence at stage k
S_{ki}: the time of the i^{th} service start event occurrence at stage k
F_{ki}: the time of the i^{th} service finish event occurrence at stage k

For single server systems, the subscript k will be dropped. Setting $R_0 = 1$ and $b = 1$ in Figure 1.2 gives the ERG for a single-server, first-in-first-out queue ($G/G/1$). Intuitively translating this

ERG to an optimization program that generates the sample path for given interarrival times $\{a_i\}$ and service times $\{s_i\}$ for n jobs gives the following LP. (The derivation of this program using the procedure in the next section is given in Example 2.1).

GG1-LP1:

$$\begin{array}{lll} \min & \sum_{i=1}^{n}(A_i + S_i + F_i) & \\ st. & & \\ A_{i+1} & = A_i + a_{i+1} & ,i = 1,...,n-1 \\ F_i & = S_i + s_i & ,i = 1,...,n \\ S_i & \geq A_i & ,i = 1,...,n \\ S_i & \geq F_{i-1} & ,i = 2,...,n \\ A_1 = 0; A_i, S_i, F_i & \text{free}, \forall i\,. & \end{array}$$

The objective function, as in a simulation, is simply to execute all the events as early as feasible. The optimal solution of this LP is identical to the state trajectories (sample paths) generated by simulating an ERG model of the system (see Theorem 2.1).

For this simple system, there is one constraint for every arc in the ERG. The first constraint is the definition of the interarrival time: the i+1$^{\text{th}}$ job will arrive (at time A_{i+1}) in a_{i+1} time units after the i^{th} job arrives at the system. The second constraint is the definition of the service time and states that the i^{th} job (started at time S_i) will finish its service s_i time units later (at time F_i). The third constraint ensures that the i^{th} service cannot start before the i^{th} job actually arrives. The last constraint enforces the constraint that the server cannot start work on the next job before it finishes the previous job.

An ERG, when observed from a simulation point of view, is a graphical representation for the underlying event scheduling process. When an ERG is considered from a mathematical programming perspective, unconditional arcs in the ERG impose *equality* constraints and conditional arcs in the ERG impose *greater than and equal to* constraints on event occurrence times. From this point of view, an ERG is a set of constraints on event occurrence times. Each equality or inequality constraint impacts at most the two event types connected by the arc with a right-hand side for an equality constraint being the time delay for the arc. Simulating an ERG, with a sequence of generated random variates as input data, involves executing all event instances as soon as feasible. This is equivalent to the objective of minimizing the times of all event instances subject to the constraints imposed by the arcs in the ERG.

For modeling large-scale systems, such as huge networks of queues, the ERG represents a generic element in an array of ERGs. The events in the ERG are subscripted with the particular station in the queueing network to which it applies by using event parameters. For this we will expand the basic definition of an ERG to allow the values of a string of expressions computed after an event is executed to be passed to a string of state variables when an event is scheduled. The values of expressions are included in boxes on the arcs.

Treating event parameters as subscripts, the graph becomes an element in an array of ERGs that model a network composed of a large number of similar systems. For example, Figure 1.3 is an ERG for m multiple-server queues in tandem. (The state changes are omitted in Figure 1.3 since these merely increment and decrement state variables $Q[k]$ and $R[k]$ for station k like in Figure 1.2.) The only change to the graph in Figure 1.2 is an additional arc from the Finish(k) event for station k to the Arrival(k) event for the next station, k+1, and the arc attributes.

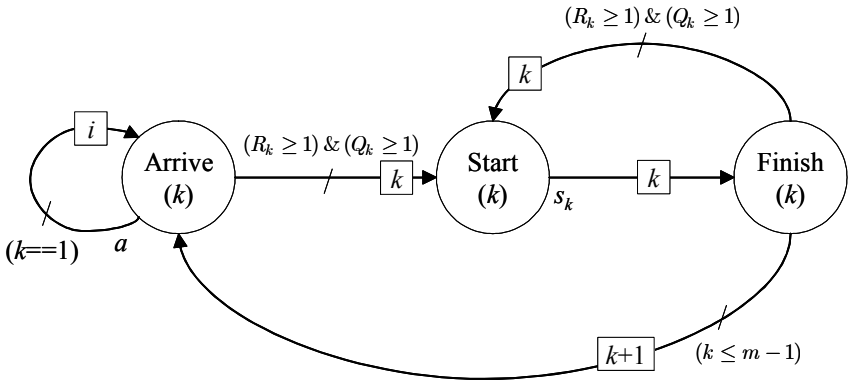


Figure 1.3 : ERG for *m* Multiple-Server Queues in Tandem

The ERG in Figure 1.3 can be easily extended to model more general queueing networks. For example, a generic jobshop can be modeled by adding a parameter, *j*, to the objects in the graph that indicates job type. The attribute, *k*+1, on arc from Finish(*k*,*j*) to Arrival(*k*,*j*) in the resulting ERG can be replaced by a general routing function. The ERG then becomes an element of a 2-dimensional array (k^{th} station by j^{th} job type) of ERGs. Transportation resources and move times as well as resource failures and repairs can be modeled by altering the conditions and delays on the arcs in this ERG (see Schruben and Schruben 2000). We defer a discussion of finite buffer tandem queues under different blocking scenarios to Section 3.1.

Event parameters, while convenient for simulating very large systems, do not increase the fundamental modeling power of ERGs. For many of our examples, we present the fully expanded event graphs (like in Figure 3.2) without resorting to using parameters to make their relationships to the corresponding optimization models clearer.

Specializations of ERGs include resource cycle graphs for simulating queueing networks (Hyden, Schruben and Roeder 2001). In a resource cycle ERG, the state variables are all integer arrays and the resource state changes associated with every event are expressed as one or more integer difference equations. Simulating these models involves increasing and decreasing the values of elements in the state arrays when specific events occur (e.g., the number of idle servers of a particular class would decrease whenever a start service event occurs for this class). Event relationship graph modeling of such systems has certain advantages in terms of simplicity and efficiency in simulation (Schruben and Schruben 2000). For example, a simulated resource cycle ERG for a semiconductor factory ran orders of magnitude faster than the most popular commercial simulator (Schruben and Roeder 2003). Object oriented ERGs (called LEGOs for Listening Event Graph Objects) have been developed by Buss and Sanchez (2002). Savage, Schruben and Yucesan (2004) showed that any discrete-event dynamic system can likely be modeled as an ERG by proving they have Turing Machine modeling power.

ERGs are used here to provide explicit examples; however, the approach is not dependent on this representation of a discrete-event model. ERGs define a discrete-event system model at a more fundamental level than most commercial simulation languages by expressing the relationships between the different event functions in the simulation code. Process interaction and activity scanning simulations that are more typical of commercial simulation codes can be translated into their basic ERGs (Schruben and Yucesan 1994) and also modeled as optimization problems.

2 METHODOLOGY

We will employ the definition of an ERG given in Yucesan and Schruben (1992). An *Event Relationship Graph* (ERG) is defined as an ordered triple $G = \{V(G), E(G), \Psi_G\}$, where $V(G)$ is the set of event vertices, $E(G)$ is the set of directed arcs, and Ψ_G is the incidence function that associates with each arc of *G* an ordered pair of vertices (they could be the same vertex) of *G*. For each directed arc $e \in E(G)$, let *O*(*e*) and *D*(*e*) be the originating and destination vertices, respectively. Defining STATES as in Zeigler (1984) an *Event Relationship Graph Model* (ERGM) is defined as $\Sigma = (\Phi, \Pi, T, G)$, where

$$\Phi = \{ f_v : \text{STATES} \rightarrow \text{STATES} \mid v \in V(G) \} \text{ (a set of state transitions functions),}$$
$$\Pi = \{ \pi_e : \text{STATES} \rightarrow \{0,1\} \mid e \in E(G) \} \text{ (a set of Boolean arc conditions), and}$$
$$T = \{ t_e : \text{STATES} \rightarrow \mathbb{R}^+ \mid e \in E(G) \} \text{ (a set of arc delay times).}$$

In this paper, we assume that all state variables are integers (fractional rational numbers can be converted into integers by multiplying by a power of ten), for otherwise the relationship in (2.1) will not be well-defined. The arc delay times, $\{t_e \in E(G)\}$ (typically the realization of a state-dependent stochastic process), along with the initial states and termination conditions define a sample path for a stochastic process called a simulation run. We are going to model these simulation run sample paths as the solutions to optimization programs.

We will express the procedure for translating ERGMs into optimization models in terms of elementary event graphs as defined in Yucesan and Schruben (1992). They showed that any ERGM can be expanded to yield an *Elementary Event Relationship Graph Model* (EERGM)—an EERGM is an ERGM in which every vertex contains only at most one state variable change and every arc condition consists of only two arithmetic expressions connected by relational operators such as "≥," "≤," "<," ">," "=," and "≠." We modify this definition by only allowing the relational operator "≥" because other relational operators can be modeled by the "≥" operator and/or multiplication of "–1." For example, for a generic integer state variable *R*, the condition ($R \neq 1$) is equivalent to conditions ($-R \geq 0$) OR ($R \geq 2$). Boolean AND and OR operations will result in conjunctive and disjunctive constraints in the optimization models. The definition of an EERGM only allows conditions with a single relational operator; therefore, ANDs and ORs will not appear in the arc conditions of an EERGM. We assume that a state variable can only be increased or decreased by at most 1 unit at one time (one occurrence of an event). For example, an event incrementing *R* by 2 units can be replaced by two events occurring in succession without delay or interruption, each of which increments *R* by 1 unit. We will assume positive arc delay times.

Yucesan and Schruben (1992) also showed that any ERG can be expanded into graphs with only conditional zero-delay arcs (denoted by E^c) and unconditional delay arcs (denoted by E^u); therefore, one need only consider these two types of arcs when deriving constraints. A vertex name with subscript *i* denotes the instant of the i^{th} occurrence of that event (vertex); e.g., A_i is the instant at which the i^{th} *A* event occurs. Define the count of the number of occurrences of event *A* by time *t* as a right-continuous function,

$$C_A(t) = \max\{i : A_i \leq t\}.$$

Since this event counting function is a jump function, we define times *just before* and *right after* the instant at which event A occurs for i times as $A_i^- = \sup\{t : C_A(t) < i\}$ and $A_i^+ = \inf\{t : C_A(t) \geq i\}$ (= $\min\{t : C_A(t) \geq i\}$). Since time is continuous, $A_i^- = A_i^+ = A_i$, but $C_A(A_i^-) = i - 1$ and $C_A(A_i^+) = i$. If the i^{th} occurrence of event A happens at or before time t, then event A will have occurred at least i times by time t. We have the usual relationships between event counting processes and times,

$$A_i \leq t \Leftrightarrow C_A(t) \geq i \tag{2.1}$$

and

$$A_{i+1} > t \Leftrightarrow C_A(t) \leq i.$$

Given an EERGM defined above, the methodology is described in the following procedure.

ERG2MP (Event Relationship Graph To Mathematical Program):

Step 1. For every unconditional timed arc $e \in E^u$, define a set of equality constraints $D(e)_i = O(e)_j + t_{ej}$, $i, j = 1,...,n$, where n is the maximum number of occurrences of event $O(e)$ in the sample path, capturing the fact that $O(e)_j$ causes event $D(e)_i$ to occur unconditionally after a delay t_{ej}.

Step 2. For every conditional zero-delay arc $e \in E^c$, define a set of inequality constraints reflecting the fact that condition on arc e is true at both times $O(e)_j^+$ and $I(e)_i^-$ for a unique pair of indices i and j (see Appendix A for details).

Step 3. Combine all constraints along with an objective function of executing all events as soon as feasible into a mathematical program. ◊

If multiple instances of the same event can be scheduled, binary assignment variables can be added to determine the assignments of i and j and the equality will be changed to inequality, otherwise, set $i = j$. Appendix A gives details on which particular instance, j, of event $O(e)$ will schedule which instance, i, of event $D(e)$. This situation could happen, for example, when simulating a parallel-server queue where customers might leave a service station in a different order from that in which they arrived (called "*overtaking*" in the queueing literature). The procedure assumes a simulated sample path of n events. Stopping rules that result in a random n are not considered here.

EXAMPLE 2.1. We provide the details for deriving GG1-LP1 given in Section 1.

Step 1: The two unconditional delay arcs: (Arrive, Arrive) and (Start, Finish) give, respectively,

(Arrive, Arrive) $A_{i+1} = A_i + a_{i+1}$, $i = 1,...,n-1$
(Start, Finish) $F_i = S_i + s_i$, $i = 1,...,n$

Step 2: The Start vertex is scheduled by two conditional arcs with same conditions, each of which contains two relational operators. If this ERGM were transformed into an EERGM, the two relational operators on each of these conditional arcs could be implemented by two one-relational-operator conditional arcs connected by a dummy event vertex. However, for ease of exposition, we will work with the original ERGM directly; one can also work with the EERGM and obtain an LP similar to GG1-LP1 by eliminating some redundant constraints.

At the time just before the occurrence of the i^{th} Start event, the two conditions on these two conditional arcs must be true. Hence,

($R \geq 1$):
$$\begin{aligned} R(S_i^-) &= C_F(S_i^-) - C_S(S_i^-) + 1 \geq 1 \\ &\Leftrightarrow C_F(S_i^-) \geq i - 1 \\ &\Leftrightarrow F_{i-1} \leq S_i^- = S_i \end{aligned}$$

($Q \geq 1$):
$$\begin{aligned} Q(S_i^-) &= C_A(S_i^-) - C_S(S_i^-) \geq 1 \\ &\Leftrightarrow C_A(S_i^-) \geq i \\ &\Leftrightarrow A_i \leq S_i^- = S_i \end{aligned}$$

Step 3: The above four constraints along with an objective function of executing all events as soon as feasible (i.e., $\sum_{i=1}^{n} A_i + S_i + F_i$) constitutes GG1-LP1. Since delay times are assumed positive, we do not include the nonnegative constraints on the variables. ◊

When there is no overtaking, an LP formulation can be obtained. The following theorem states that the optimal solution of a resulting LP (with an objective function of minimizing all event times) is identical to the system dynamics—the sample path obtained from a simulation of the original ERG. A proof is provided in Appendix B.

THEOREM 2.1. *Any feasible sample path for an ERG without overtaking is an optimal solution to the corresponding optimization model generated by Procedure ERG2MP.*

Simpler objective functions will work as well for GG1-LP1. For example, we do not need to include the job arrival times in the objective function because the times for these exogenous events are completely determined by the input data. Also, if we knew that the n jobs occurred in the same busy period, then the simple objective of minimizing the length of the busy period, F_n, could be used. Changing the objective to $\sum_{i=1}^{n} F_i^k$ may provide information for estimating different moments of some interesting decision variables, which is the subject of further study.

Other performance measures can also be evaluated by using the solutions (event times) of the optimization representation since the state of a DES is piecewise constant. For example, if $L_n(x_1,\ldots,x_n;\ \theta)$ is a sample performance measure evaluated over a sample path containing n events, $f(\boldsymbol{Z}_i)$ is a bounded cost when the system is at state $\boldsymbol{Z}_i$ immediately after the occurrence of

the i^{th} event, and x_i is event time of the i^{th} event, then one has $L_n(x_1,...,x_n;\theta) = \sum_{i=0}^{n-1} f(\boldsymbol{Z}_i)[x_{i+1} - x_i]$ (see Glasserman 1991 for more details).

For cases where overtaking is possible, binary variables are added to ensure correct assignments of events. It might also be possible to add as many binary variables as needed so that the optimal solution under an objective function of minimizing all event times is identical to the system dynamics. The following example is a formulation with binary variables generated by procedure ERG2MP.

EXAMPLE 2.2. Consider the *G*/*G*/*R* ERGM given in Figure 1.2 with R_0>1 parallel servers and a sample path of *n* completed batches of jobs, each of size *b*. We go through the details of each of the steps in procedure EGM2MP.

Step 1: The unconditional delay arc (Arrive, Arrive) gives

(Arrive, Arrive) $$A_{i+1} = A_i + a_{i+1}, i = 1,...,b \cdot n - 1 \quad .$$

For the unconditional delay arc (Start, Finish), since overtaking can happen with multiple servers, we need to insure that there is one and only one equality constraint binding for every pair of Start and Finish events. That is, we need to assign a single Finish event to each Start event. We can use the usual assignment constraints to do this with binary variables determining the assignments of which of several possible Finish events is scheduled by each Start event. Let binary variable $\delta_{ij} = 1$ if the i^{th} Start event schedules the j^{th} Finish event, and 0 otherwise. For a suitably large real number *M* (e.g., the upper bound of $S_i + s_i - F_j, \forall i, j$ or simply bigger than the simulation duration), we have

(Start, Finish) $$F_j \geq S_i + s_i - M(1 - \delta_{ij}),\ i = 1,\ldots,n;\ j = \max\{i\text{–}R_0\text{+}1,\ 1\},...,n$$

Since event times are ordered (i.e., $F_j \geq F_{j-1}$), if the i^{th} Start event schedules the j–1^{th} Finish event, then the above constraint should also be enforced. Therefore, the formulation could become tighter if these assignment constraints are replaced by

$$F_j \geq S_i + s_i - M(1 - \sum_{l=\max\{i-R_0+1,1\}}^{j} \delta_{il}),\ i = 1,\ldots,n;\ j = \max\{i\text{–}R_0\text{+}1,\ 1\},...,n$$

Step 2: In EERGMs we assumed for completeness that a state variable can only be increased or decreased by at most 1 at a time. Since our batch size is b, this means here that the Start event needs to be split into *b* events, each decreasing *Q* by 1. However, this added complication is not necessary when the *b* events *always* occur simultaneously as they do here. Just before the i^{th} occurrence the Start event, the condition $(R \geq 1)\&(Q \geq b)$ on the two conditional arcs must be true. Hence,

$$(Q \geq b): \quad \begin{aligned} Q(S_i^-) &= C_A(S_i^-) - b \cdot C_S(S_i^-) \geq b \\ &\Leftrightarrow C_A(S_i^-) \geq b \cdot i \\ &\Leftrightarrow A_{b \cdot i} \leq S_i^- = S_i \end{aligned}$$

This constraint, $A_{b \cdot i} \leq S_i$, although derived mechanically, has the intuitive interpretation that simply says that a server cannot start the i^{th} service until after at least the i^{th} full batch of b jobs has arrived. We also need

$$(R \geq 1): \quad \begin{aligned} R(S_i^-) &= C_F(S_i^-) - C_S(S_i^-) + R_0 \geq 1 \\ &\Leftrightarrow C_F(S_i^-) \geq i - R_0 \\ &\Leftrightarrow F_{i-R_0} \leq S_i^- = S_i \end{aligned}$$

Again, although derived mechanically, this constraint has the simple interpretation that, since there are only R_0 resources, the number of Start events cannot exceed the number of Finish events by more than R_0.

Step 3: The objective function is simply to execute all events as soon as feasible. To summarize, the following mixed-integer program generates the system trajectory for a batch-service $G/G/R$ queue with n batches of jobs.

GGRb-MIP:

$$\begin{aligned} &\min \quad \textstyle\sum_{i=1}^{b \cdot n}(A_i) + \sum_{i=1}^{n}(S_i + F_i) \\ &st. \\ &A_{i+1} \;=\; A_i + a_{i+1} \;, i = 1,\ldots,b \cdot n - 1 \\ &F_j \quad \geq S_i + s_i - M\Big(1 - \textstyle\sum_{l=\max\{i-R_0+1,1\}}^{j} \delta_{il}\Big), \begin{matrix} i = 1,\ldots,n; \\ j = \max\{i - R_0 + 1,1\},\ldots,n \end{matrix} \\ &S_i \quad \geq \; A_{b \cdot i} \qquad , i = 1,\ldots,n \\ &S_i \quad \geq \; F_{i-R_0} \qquad , i = R_0 + 1,\ldots,n \\ &\textstyle\sum_{j=\max\{i-R_0+1\}}^{n} \delta_{ij} = 1, i = 1,\ldots,n \\ &\textstyle\sum_{i=1}^{\min\{j+R_0-1,n\}} \delta_{ij} = 1, j = 1,\ldots,n \\ &A_1 = 0; A_i, S_i, F_i \text{ free}, \delta_{ij} \in \{0,1\}, \forall i, j\,. \end{aligned}$$ ◊

The big-M method is used for ease of exposition. Techniques for developing stronger MIP formulations can be found in Nemhauser and Wolsey (1999).

Of course, we could solve this optimization problem easily by simply running the simulation. This would give us the optimal values for all the binary variables. This in turn tells us which linear constraints are binding in a relaxation of the MIP into an LP whose solution is the system state trajectory corresponding to the input data. The mathematical tools of optimization can then be used to analyze the sample path. For example, fixing the values of the binary variables at their

optimal values found from running the simulation changes the MIP into an LP, which has a dual (more on this later). In this way, explicit sample-path dual models for more general discrete-event dynamic systems can be derived, the subject of ongoing research. The queue length process, $Q(t) = C_A(t) - C_S(t)$, which is independent of the queueing discipline, can be computed from the resulting dual variables. Without batched service ($b = 1$), the sequence of customer waiting times, $W_i = S_i - A_i$, are the excess variables—the reduced costs of the dual—for the third constraint (see Chan 2005 for more details). We emphasize again that our goal here is to derive an optimization representation for a simulation model for its mathematical properties. We are actually going to solve the resulting mathematical program.

3 APPLICATIONS

The fundamental motivation here for representing discrete-event systems as optimization models is to be able to use the rich mathematical and algorithmic theory of combinatorial optimization in the study of queues and other discrete-event dynamic systems. In this section, we illustrate some specific motivations for formulating discrete-event systems as optimization models by showing several potential applications.

A straightforward, but probably not very good, reason for representing simulations as analytical optimization models is to be able to solve an LP as an alternative to executing a simulation—without resorting to sequentially processing a future events list. However, it was interesting to observe that the run times for solving moderate sized cases of the optimization model (GG1-LP1) for a $G/G/1$ queue were generally much faster than running the simulation. This is because the optimization software exploited the nearly diagonal form of the matrix of constraint coefficients. This should carry over to modeling networks of such queues. The mathematical programs might also be useful in modeling simple subsystems in hierarchical simulation models. We focus in this paper on other incentives.

3.1 Tandem Queueing Networks

In this section we model open queueing systems as LPs. The optimization models for these systems provide straightforward alternative proofs of reversibility properties under various blocking scenarios, which typically involve notation dense induction proofs on max-plus models or network flow analysis. A condition under which a general blocking open tandem queue is reversible is presented.

Consider a tandem queueing system with m consecutive stages, labeled $k = 1,\ldots,m$. In each stage, there is a single server and a finite storage space for intermediate jobs. Job $i = 1,\ldots,n$ is processed at all stages in sequence with service times s_{ki} (see Figure 3.1). It is assumed that $\{(s_{1i}, s_{2i},\ldots,s_{mi}), i = 1,...,n\}$ are i.i.d. random variables.

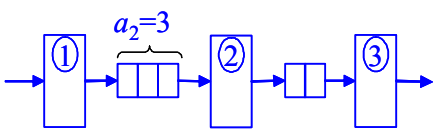
Since buffer sizes between stages are finite, a control policy is needed in order to coordinate the service process between stages. Two common control policies considered in the literature are communication blocking and production blocking control policies (Buzacott and Shanthikumar 1993). Other types of blocking include kanban blocking, variations of kanban blocking (Liberopoulos and Dallery 2000), and general blocking (Cheng and Yao 1993). In the following, we shall briefly introduce communication, production, and general blockings; for detailed explana-

tion and other blocking control policies, the reader is referred to Onvural (1990) and Chan (2005).

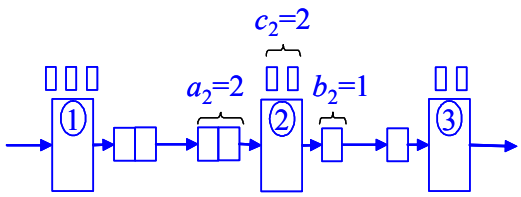
Denote by TQmC($\rightarrow/G_k/1/a_k$, k = 1,...,m) and TQmP($\rightarrow/G_k/1/a_k$, k = 1,...,m) an m-stage communication-blocking tandem queue and an m-stage production-blocking tandem queue, respectively, where the arrow "→" represents that the arrival process at stage k ($2 \le k \le m$) is the departure process at stage k-1 except that stage 1 has an external general arrival process, a_k is the buffer size for jobs at stage k including the space of the server at stage k. The notation a_k is consistent with the tandem queueing literature. As a result, t_{ai} is used to denote the time interval between the i^{th} external arrival and the i-1^{th} external arrival. In a communication-blocking tandem queue, a server at stage k can start processing a job if the following conditions are satisfied: (C1) a job is available for processing, (C2) a server is available, and (C3) there is at least one empty space at the next stage. Upon the arrival of a job at stage k, if a server at stage k is available but the next stage is full, then the server is blocked. Therefore communication blocking is also called "*blocked-before-service.*" In a production-blocking tandem queue, a server at stage k can start processing a job if the following conditions are satisfied: (P1) a job is available for processing, (P2) a server is available, and (P3) the server is not holding a finished job. When a server at stage k finishes processing a job, if there is no empty space at the next stage, then the finished job will stay with the server, blocking it from processing the next job. Therefore production blocking is also called "*blocked-after-service.*"

Denote by TQmG($\rightarrow/G_k/1/(a_k, b_k, c_k)$, k = 1,...,m) an m-stage tandem queue under general blocking, where a_k, b_k, and c_k are, respectively, the maximum number of jobs allowed *in queue and in service* (i.e., raw jobs), the maximum number of *finished jobs* allowed at the stage, and the maximum number of jobs allowed at the stage (*in queue, in service, and finished*). The relationship between a_k, b_k, and c_k, is $c_k \le a_k + b_k$. The limit, c_k is used to control all jobs in a stage, including raw jobs and finished jobs. In Figure 3.1.b, $c_k = 2$; this means that it does not matter in which buffer the jobs are, the maximum spaces in stage 2 is 2. Under general blocking, a server at stage k can start processing a job if (G1) a job is available for processing, (G2) a server is available, and (G3) there are strictly less than b_k finished jobs blocked at stage k.

Let B_k be the number of finished jobs blocked at stage k. Figure 3.2.a–c are the ERGs for a 3-stage tandem queue under communication blocking, production blocking, and general blocking, respectively. To make the exposition more transparent, we have not used event parameters as in Figure 1.3 at the expense of much more complicated looking ERGs. For simplicity, we have dropped the time dependency from all parameters in these figures (e.g., instead of $a_k(t)$, we write a_k). In the LP representations, these parameters are fixed to be the initial values (e.g., $a_k(0)$). In Figure 3.2.c, since services are assumed to be uninterrupted, having $b_k = b_k - 1$ occurred at event S_k instead of at event F_k can prevent more than $b_k(0)$ jobs from being served concurrently.

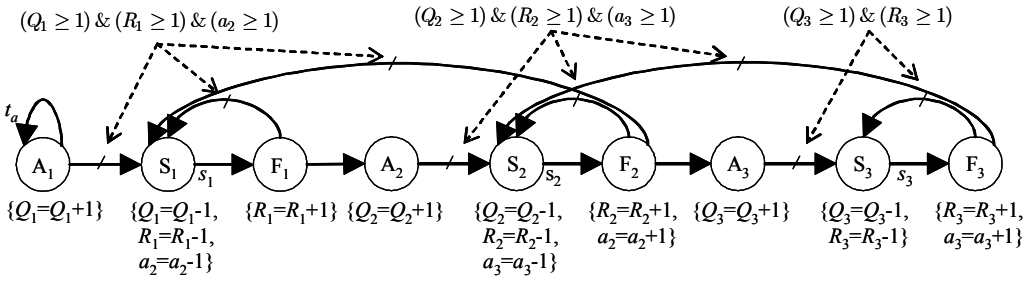


(a) Communication or Production Blocking

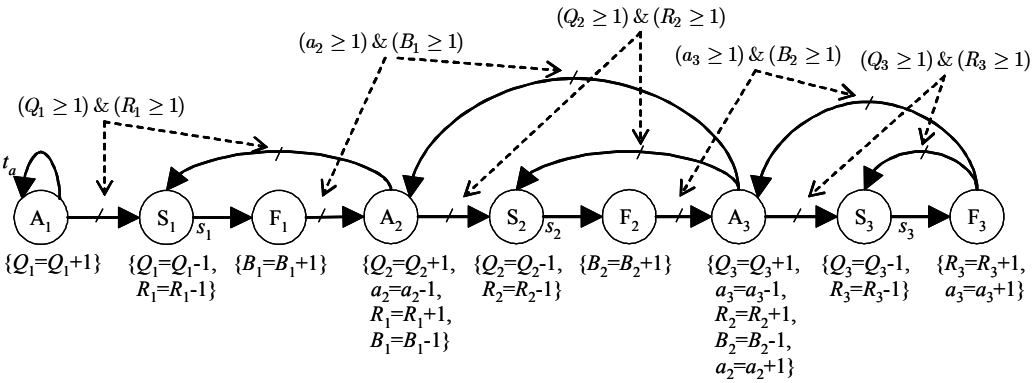


(b) General Blocking

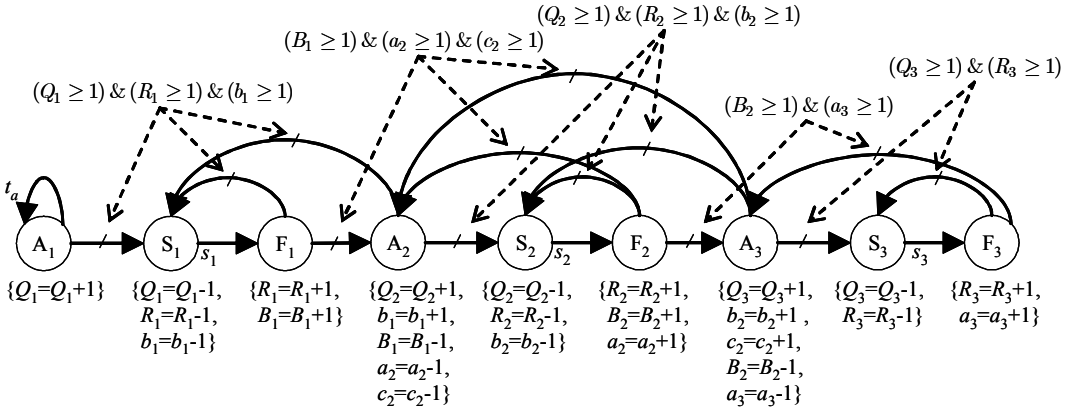
Figure 3.1: Tandem Queues under Different Blocking Control Policies



(a) ERG of TQ3C($G/G_1/R_1/\infty, \rightarrow/G_2/R_2/a_2, \rightarrow/G_3/R_3/a_3$)



(b) ERG of TQ3P($G/G_1/R_1/\infty, \rightarrow/G_2/R_2/a_2, \rightarrow/G_3/R_3/a_3$)



(c) ERG of TQ3G($G/G_1/R_1/(\infty, b_1, \infty), \rightarrow/G_2/R_2/(a_2, b_2, c_2), \rightarrow/G_3/R_3/(a_3, \infty, c_3)$)

Figure 3.2 : ERG of Open Tandem Queueing Networks

3.1.1 Deriving Linear Program Representations for TQ3C(.)

We use procedure ERG2MP to derive the LP formulation for TQ3C($G/G_1/1/\infty$, $\rightarrow/G_2/1/a_2$, $\rightarrow/G_3/1/a_3$) in the following.

Step 1: Directly from the unconditional timed arcs in Figure 3.2.a, we get

(A_1, A_1) $$A_{1,i+1} = A_{1i} + t_{ai} \tag{3.1}$$

(S_1, F_1) $$F_{1i} = S_{1i} + s_{1i} \tag{3.2}$$

(F_1, A_2) $$F_{1i} = A_{2i}$$

(S_2, F_2) $$F_{2i} = S_{2i} + s_{2i}$$

(F_2, A_3) $$F_{2i} = A_{3i}$$

(S_3, F_3) $$F_{3i} = S_{3i} + s_{3i}$$

All these equalities will be used in the next step to simplify the formulation. In particular, (3.1) will allow the use of arrival-time sequence $\{A_i, i = 1,\ldots,n\}$ as the input data to the formulation, instead of the interarrival-time sequence.

Step 2: Vertices S_1, S_2, and S_3 are scheduled by more than one conditional arc with the same conditions. At the time just before the occurrence of the i^{th} S_1 event, the three conditions ($Q_1 \geq 1$)&($R_1 \geq 1$)&($a_2 \geq 1$) on the three conditional arcs must be true, yielding the following constraints.

($Q_1 \geq 1$)
$$\begin{aligned} Q_1(S_{1i}^-) &= C_{A_1}(S_{1i}^-) - C_{S_1}(S_{1i}^-) \geq 1 \\ C_{A_1}(S_{1i}^-) &\geq i \\ A_{1i} &\leq S_{1i}^- = S_{1i} \end{aligned}$$

Using equation (3.2) gives the constraint,

$$F_{1i} \geq A_{1i} + s_{1i}$$

($R_1 \geq 1$)
$$\begin{aligned} R_1(S_{1i}^-) &= C_{F_1}(S_{1i}^-) - C_{S_1}(S_{1i}^-) + 1 \geq 1 \\ C_{F_1}(S_{1i}^-) &\geq i - 1 \\ F_{1,i-1} &\leq S_{1i}^- = S_{1i} \end{aligned}$$

Again, using equation (3.2) gives the constraint,

$$F_{1i} - F_{1,i-1} \geq s_{1i}$$

($a_2 \geq 1$)
$$\begin{aligned} a_2(t) &= C_{F_2}(S_{1i}^-) - C_{S_1}(S_{1i}^-) + a_2 \geq 1 \\ C_{F_2}(S_{1i}^-) &\geq i - a_2 \\ F_{2,i-a_2} &\leq S_{1i}^- = S_{1i} \end{aligned}$$

Using equation (3.2) gives the constraint,

$$F_{1i} - F_{2,i-a_2} \geq s_{1i}$$

Similar for events S_2 and S_3, five more constraints for Q_2, R_2, a_2, Q_3, and R_3 can be derived.

Step 3: Combining all these eight constraints along with an objective function of executing all events as soon as feasible into an LP, we have a mathematical programming representation for a 3-stage communication-blocking open tandem queue. This LP will be called "TQ3C-LP(F)," where the letter "F" reflects that the decision variables are the finish times. The formulation with n jobs is given below (The dual variables for each set of constraints are in parenthesis).

TQ3C-LP(F):

$$\begin{aligned}
&\min \quad \sum_{i=1}^{n} (F_{1i} + F_{2i} + F_{3i}) \\
&st. \\
&F_{1i} \geq A_{1i} + s_{1i} \quad , i = 1,...,n \quad (U_{1i}) \\
&F_{1i} - F_{1,i-1} \geq s_{1i} \quad , i = 2,...,n \quad (V_{1i}) \\
&F_{1i} - F_{2,i-a_2} \geq s_{1i} \quad , i = a_2 + 1,...,n \quad (W_{1i}) \\
&F_{2i} - F_{1i} \geq s_{2i} \quad , i = 1,...,n \quad (U_{2i}) \\
&F_{2i} - F_{2,i-1} \geq s_{2i} \quad , i = 2,...,n \quad (V_{2i}) \\
&F_{2i} - F_{3,i-a_3} \geq s_{2i} \quad , i = a_3 + 1,...,n \quad (W_{2i}) \\
&F_{3i} - F_{2i} \geq s_{3i} \quad , i = 1,...,n \quad (U_{3i}) \\
&F_{3i} - F_{3,i-1} \geq s_{3i} \quad , i = 2,...,n \quad (V_{3i})
\end{aligned}$$

where F_{ki}, k = 1,2,3, i = 1,...,n, are unrestricted in sign; and U_{ki}, V_{ki}, and W_{ki} are the dual variables.

For an m-stage tandem queue, the derivation is similar and the LP representation is

TQmC-LP(F):

$$\begin{aligned}
&\min \sum_{k=1}^{m} \sum_{i=1}^{n} F_{ki} \\
&st. \\
&F_{ki} - F_{k-1,i} \geq s_{ki} \quad , k = 1,...,m, i = 1,...,n \quad (U_{ki}) \\
&F_{ki} - F_{k,i-1} \geq s_{ki} \quad , k = 1,...,m, i = 2,...,n \quad (V_{ki}) \\
&F_{ki} - F_{k+1,i-a_{k+1}} \geq s_{ki} \quad , k = 1,...,m-1, i = a_{k+1} + 1,...,n \quad (W_{ki})
\end{aligned}$$

where $F_{0i} = A_{1i}$, i = 1,...,n; F_{ki}, k = 1,...,m, i = 1,...,n, are unrestricted in sign; and U_{ki}, V_{ki}, and W_{ki} are the dual variables. Using ERG2MP, we can also derive LP representations for production-blocking (called TQmP-LP(F)) and general-blocking (called TQmG-LP(F)) tandem queues, which are given below.

TQmP-LP(F):

$$\begin{aligned}
&\min \sum_{k=1}^{m}\sum_{i=1}^{n} F_{ki} \\
&st. \\
&F_{ki} - F_{k-1,i} && \geq s_{ki} \;, k = 1,...,m, i = 1,...,n && (U_{ki}) \\
&F_{ki} - F_{k,i-1} && \geq s_{ki} \;, k = 1,...,m, i = 2,...,n && (V_{ki}) \\
&F_{ki} - F_{j,i-\left(\sum_{l=k+1}^{j} a_l + 1\right)} && \geq s_{ki} \;, k = 1,...,m-1, j = k+1,...,m && (W_{kji}) \\
& && \quad , i = \textstyle\sum_{l=k+1}^{j} a_l + 2,...,n
\end{aligned}$$

where $F_{0i} = A_{1i}$, $i = 1,...,n$; F_{ki}, $k = 1,...,m$, $i = 1,...,n$, are unrestricted in sign; and U_{ki}, V_{ki}, and W_{kji} are the dual variables.

TQmG-LP(*F*):

$$\begin{aligned}
&\min \sum_{k=1}^{m}\sum_{i=1}^{n} F_{ki} \\
&st. \\
&F_{ki} - F_{k-1,i} && \geq s_{ki} \;, k = 1,...,m, i = 1,...,n && (U_{ki}) \\
&F_{ki} - F_{k,i-1} && \geq s_{ki} \;, k = 1,...,m, i = 2,...,n && (V_{ki}) \\
&F_{ki} - F_{j,i-\left(b_k + \sum_{l=k+1}^{j-1} c_l + a_j\right)} && \geq s_{ki} \;, k = 1,...,m-1, j = k+1,...,m && (W_{kji}) \\
& && \quad , i = b_k + \textstyle\sum_{l=k+1}^{j-1} c_l + a_j + 1,...,n
\end{aligned}$$

where $F_{0i} = A_{1i}$, $i = 1,...,n$; F_{ki}, $k = 1,...,m$, $i = 1,...,n$, are unrestricted in sign; and U_{ki}, V_{ki}, and W_{kji} are the dual variables.

REMARK 3.1. Relationships between stochastic processes are often used to establish properties of queueing systems (see for example, Cheng 1997). Deterministic analysis of these system dynamics equations is used with coupling arguments for problems such as developing performance bounds and proving reversibility. For example, using intuitive arguments, a representation of the departure processes in blocked tandem queues is developed in Buzacott and Shanthikumar (1993). There it is shown that properties of the departure processes are shared by the system work in process and throughput.

Using the variables we defined in Section 1.2, the max-plus equation 5.38 on page 184 of Buzacott and Shanthikumar (1993) for communications blocking is

$$F_{ki} = \max\left\{ F_{k-1,i}, F_{k,i-1}, F_{k+1,i-a_{k+1}} \right\} + s_{ki} \,,$$

and $F_{0i} = A_{1i}$ = the arrival times of jobs to the first queue. The relationships between F_{ki} and the three variable inside the $\max\{\cdot\}$ function can be written as three inequalities, which are equivalent to the three constraints in TQmC-LP(*F*) with an objective function of minimizing all variables. Therefore, TQmC-LP(*F*) is equivalent to these system dynamics equations. It can also be shown that TQmP-LP(*F*) is equivalent to equation 5.37 in Buzacott and Shanthikumar (1993) and TQmG-LP(*F*) is identical to the max-plus recursion given in Cheng 1995. ◊

In addition to being developed mechanically, each of the LP representations presented here has easily interpreted constraints and a dual that will be discussed in subsequent sections.

3.1.2 Comparison of Communication Blocking and Production Blocking

The above max-plus system-dynamics equations are used in Buzacott and Shanthikumar (1993) with induction arguments to prove bounds on the performance of production-blocking and communication-blocking systems. Such performance bounds can be seen immediately from their LP representations. For example, one can compare the number of constraints in the two LPs (without solving them) and immediately observes that the performance of communication blocking (as reflected in functions of job completion times) can be no better than that of production blocking since the production-blocking LP has less constraints, all else being equal. This appears in the first result listed below. Other results proven in Buzacott and Shanthikumar (1993) by induction are also given below.

Before proceeding to the proofs, we introduce two departure-time formulations for tandem queues. In particular, using only the departure times as variables, we modify TQmC-LP(*F*) and TQmP-LP(*F*) into two formulations, called TQmC-LP(*D*) and TQmP-LP(*D*). Let D_{ki}^C [F_{ki}^C] and D_{ki}^P [F_{ki}^P] denote the departure times [finish times] of communication and production blocking tandem queues, respectively. The modification from TQmC-LP(*F*) to TQmC-LP(*D*) is immediate because $F_{ki}^C = D_{ki}^C$, $k = 1,...,m$, $i = 1,...,n$, in a communication blocking tandem queue. This gives:

TQmC-LP(*D*):

$$\begin{aligned}
&\min \sum_{k=1}^{m} \sum_{i=1}^{n} D_{ki}^C \\
&st. \\
&D_{ki}^C - D_{k-1,i}^C \geq s_{ki} \quad , k = 1,...,m, i = 1,...,n \\
&D_{ki}^C - D_{k,i-1}^C \geq s_{ki} \quad , k = 1,...,m, i = 2,...,n \\
&D_{ki}^C - D_{k+1,i-a_{k+1}}^C \geq s_{ki} \quad , k = 1,...,m-1, i = a_{k+1}+1,...,n
\end{aligned}$$

where $D_{0i}^C = A_{1i}$, $i = 1,..,n$; and D_{ki}^C, $k = 1,...,m$; $i = 1,...,n$, are unrestricted in sign. TQmP-LP(*D*) is, however, not immediate. The formal way to get it is by applying ERG2MP to Figure 3.2.b (generalized to *m* stages) and simplifying the formulation using the fact that in a production blocking tandem queue $D_{mi}^P = F_{mi}^P$ and $D_{k-1,i}^P = A_{ki}$, $k = 2,...,m$, $i = 1,...,n$. We will skip the details and present it here using an informal argument. In Figure 3.2.b, observe that the i^{th} job cannot depart from stage *k* until the following three conditions are satisfied: (1) the i^{th} job arrives at stage *k* (at time $D_{k-1,i}^P$) and finishes its service requirement at stage *k*, (2) the previous job (the i-1^{th} job) departs stage *k* (at time $D_{k,i-1}^P$) and the i^{th} job finishes its service requirement at stage *k*, and (3) there is at least one empty buffer space at stage k+1, which happens when the i-$a_{k+1}{}^{th}$ job departs stage k+1 (at time $D_{k+1,i-a_{k+1}}^P$). The third condition does not include the requirement that the i^{th} job must finish its service requirement because such requirement has already been captured in the first two conditions. These three conditions give us the following formulation for an *m*-stage production tandem queue:

TQmP-LP(*D*):

$$\begin{aligned}
&\min \textstyle\sum_{k=1}^{m}\sum_{i=1}^{n} D_{ki}^{P} \\
&st. \\
&D_{ki}^{P} - D_{k-1,i}^{P} && \geq s_{ki} \quad ,k = 1,\ldots,m; i = 1,\ldots,n \\
&D_{ki}^{P} - D_{k,i-1}^{P} && \geq s_{ki} \quad ,k = 1,\ldots,m; i = 2,\ldots,n \\
&D_{ki}^{P} - D_{k+1,i-a_{k+1}}^{P} && \geq 0 \quad ,k = 1,\ldots,m-1; i = a_{k+1}+1,\ldots,n
\end{aligned}$$

where $D_{0i}^{P} = A_{1i}$, $i = 1,..,n$; D_{ki}^{P}, $k = 1,...,m$, $i = 1,...,n$, are unrestricted in sign.

With TQmC-LP(D) and TQmP-LP(D), the following results are relatively straightforward.

1) $D_{ki}^{C} \geq D_{ki}^{P}$, $k = 1,...,m$, $i = 1,...,n$.
Alternative Proof:
This is true because the right-hand side of TQmP-LP(D) is non-negative and less than that of TQmC-LP(D). □

Adding one additional space to all buffers in the m-stage communication blocking tandem queue reverses the direction of above inequality, that is, making the communication blocking tandem queue no worse than the production blocking tandem queue. Observe that this one additional space when added to a 2-stage communication blocking tandem queue will make it equal to (i.e., as good as) a 2-stage production blocking tandem queue.

2) The departure time D_{ki}^{P} under production blocking with buffer capacities $\{a_1, a_2,...,a_m\}$ is not less than the departure time D_{ki}^{C} under communication blocking with buffer capacities $\{a_1+1, a_2+1,...,a_m+1\}$.
Alternative Proof:
Observe that this additional buffer space, while has no effect to the first two constraints in TQmC-LP(F), will cause the third constraint to be the same as the third constraint in TQmP-LP(F) when $j = k+1$. Therefore, the feasible region of TQmC-LP(F) with buffer capacities $\{a_1+1, a_2+1,\ldots,a_m+1\}$ contains the feasible region of TQmC-LP(F) with buffer capacities $\{a_1, a_2,\ldots,a_m\}$, yielding $F_{ki}^{P} \geq F_{ki}^{C}$. The result follows from the facts that $D_{ki}^{P} \geq F_{ki}^{P}$ and $D_{ki}^{C} = F_{ki}^{C}$.
□

By defining the throughput as the largest departure rate, i.e., $\lim_{i\to\infty} i / D_{ki}^{C}$ and $\lim_{i\to\infty} i / D_{ki}^{P}$, $k = 1,...,m$ (usually only the last state is of interest, but in equilibrium all stages have the same departure rate), the following result is a direct application of the above two results (for a more detailed definition of throughput, see Buzacott and Shanthikumar 1993).

3) The throughput of a communication blocking tandem queue with buffer capacity $\{a_1, a_2,...,a_m\}$ is not greater than that of a production blocking tandem queue with buffer capacity $\{a_1, a_2,...,a_m\}$, which is, in turn, not greater than that of a communication blocking tandem queue with $\{a_1+1, a_2+1,...,a_m+1\}$.

3.1.3 Reversibility Property for Tandem Queues

The *reverse queue* of a tandem queue is achieved by reversing the order of all stages. A tandem queue is said to be *reversible* if it has the same asymptotic throughput as its reverse queue. Motivations for studying the reversibility property are summarized in Buzacott and Shanthikumar (1993) and Chan (2005). Reversibility is considered to be an important property in design and performance analysis of tandem queues because properties of the original system can be inferred from the reverse system.

The reversibility property has been established and proven for communication blocking, production blocking (Yamazaki and Sakasegawa 1975, Dattatreya 1978, Muth 1979) and general blocking under certain conditions (Cheng 1995). Pinedo (2002) proved the reversibility property for deterministic scheduling problems. Most of these proofs start with the so-called "*activity network*," a directed acyclic graph (DAG) with arcs representing the time required to process jobs in the system, and show that the longest path in the network of original is equal to that of the reverse system. These activity networks are obtained intuitively either from max-plus recursions or from the precedence relations among activities of the system.

We will show in the following that these activity networks can be constructed more systematically from the dual of the LP representations of tandem queues under various blocking scenarios. We then prove the reversibility property by simply rotating the networks. For systems that are not reversible in general (e.g., general blocking tandem queues), the LP representations will also help in finding conditions for reversibility. For example, we will give a new condition under which a general blocking tandem queue is reversible by using the LP network representation.

Observe that the LP representations can also be used to derive similar properties for closed tandem queueing networks under the conditions of stationary and ergodic service times (see Chan 2005). These conditions are required for closed tandem queues because they ensure that the asymptotic throughput defined in the previous section is independent of the initial state of the system (see Dallery, Zhen and Towsley 1994).

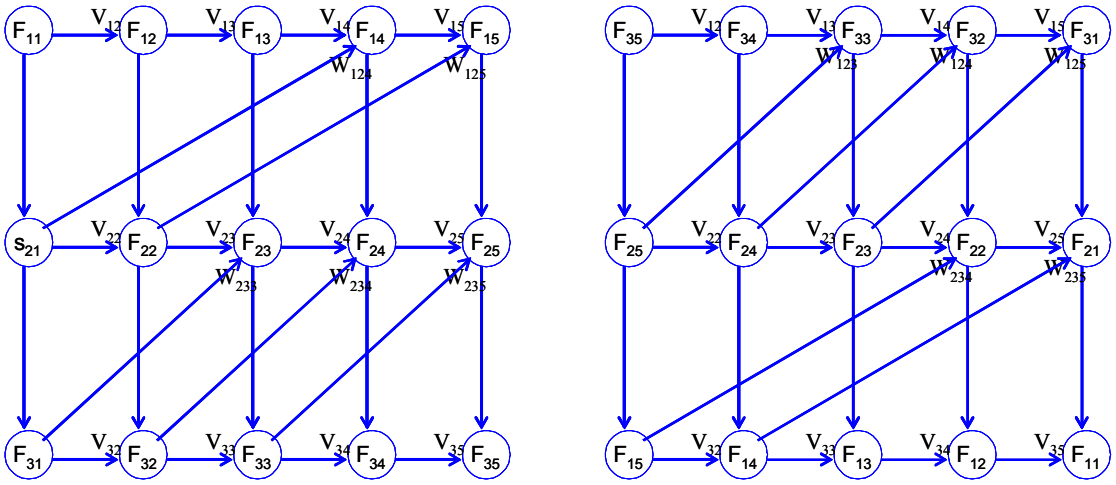
We assume that all service times are positive (a dummy server with zero processing time is equivalent to an empty buffer). Moreover, in throughput analysis one can assume infinite number of jobs waiting in front of the first stage (for otherwise, a dummy server with service times equal the interarrival times can be added in front of stage 1). The following theorem is well-known, but we give an alternative LP-based proof that will be extended to general blocking.

THEOREM 3.1. *An m-stage tandem queue is reversible under communication blocking or production blocking.*

PROOF. We prove the communication case, the production case is similar. First, we observe that the dual of TQmC-LP(F) is the network flow problem of finding the longest paths from the first node to all other nodes (the longest path from the first node to the last node is the makespan—the time to finish all jobs); e.g., the dual network for TQ3C($\rightarrow/G_1/1/\infty$, $\rightarrow/G_2/1/3$, $\rightarrow/G_3/1/2$) and n = 5 is given in Figure 3.3.a, and the makespan is the longest path from node F_{11} to node F_{35}. The length of each arc represents the service time of a job at that stage (the index of the node at which an arc terminates tells which job it is), e.g., the length of arc (F_{22}, F_{23}) is s_{23}, the service time of the 3rd job at stage 2. Variables U_{ki}, V_{ki}, and W_{ki} are the flows on the arcs.

The reverse queue is also a tandem queue (called TQ3C-R($\cdot$)) and its reverse dual network is shown in Figure 3.3.b. By rotating Figure 3.3.b 180 degrees and reversing the directions of the

arcs, one can see that these two networks are identical. Therefore, the makespans (longest paths) are the same. Since the service times are i.i.d., this reversal does not change the throughput. Therefore, the queue is reversible. □



a. Original Queue b. Reverse Queue

Figure 3.3: Dual network of TQ3C and TQ3C-R($\rightarrow/G_1/1/\infty$, $\rightarrow/G_2/1/3$, $\rightarrow/G_3/1/2$) and $n = 5$

The next theorem provides a new condition under which a general blocking tandem queue is reversible. This condition includes Cheng (1995)'s condition as a special case.

THEOREM 3.2. *A general blocking tandem queue is reversible if its control parameters satisfy* $a_k - b_k = Z$, $k = 1,...,m$, *where Z is an arbitrary integer.*

PROOF. From the LP representation, one can see that the condition $a_k - b_k = Z$, $k = 1,...,m$ ensures that the LP for reverse queue has the same graph structure as that of the original queue. The rest of the proof is identical to that of Theorem 3.1. □

3.2 Formulating Scheduling Problems

The scheduling literature is vast. The reader is referred to the scheduling book by Pinedo (2002), which gives an introduction to this topic and provides formulations to many scheduling problems. Here, we shall focus on generating constraints for optimal formulations of optimization models for scheduling problems directly from ERGs.

Traditionally, constraints for optimal scheduling problems are found in an ad-hoc manner using intuitive arguments. For complex systems, there may be a risk of missing constraints, which may eventually lead to an infeasible schedule. Here, we show that resource scheduling constraints can be obtained methodically from an ERG of the system using the ERG2MP procedure. The mathematical programming model for the system dynamics includes all the feasibility constraints that must be enforced in an associated optimization problem for resource scheduling.

For the parallel resource ERG, if the objective function is changed to minimize the last finish time and binary variables are added assigning the i^{th} job processing time to the j^{th} Start event then we have a formulation of the parallel resource scheduling problem with minimum makespan as our objective.

Translating the ERG for a *G*/*G*/*s* queue into what now becomes a mixed-integer scheduling program gives a formulation of the classical parallel-resource non-preemptive job scheduling problem, which is in the class of *NP*-hard problems. Nevertheless, solving this MIP for small numbers of jobs gives us insights into possible efficient heuristics. To add generality, the order in which jobs arrive can also become a policy decision (e.g., to derive an order-release rule).

Attaching different job indices to the sequence of job arrival events and to the sequence of service start events allows these to be scheduled optimally. We define two binary variables used to assign the i^{th} processing time to the j^{th} Start event and the i^{th} interarrival time to the j^{th} Arrival event. Let

$\theta_{ij} = 1$ if job j is the i^{th} arrival, 0 otherwise, and
$\eta_{ij} = 1$ if job j is the i^{th} service, 0 otherwise.

The optimization model for the parallel-resource ERG simulation model that allows for optimal job scheduling of n jobs to R servers, batch size 1, to minimize makespan then becomes

Makespan-MIP:

$$
\begin{aligned}
&\min \quad F_n \\
&st. \\
&F_i \;\le\; F_n && ,i = 1,..,n \\
&A_{i+1} \;\ge\; A_i + a_j\theta_{ij} && ,i = 1,...,n-1, j = 1,..,n \\
&F_i \;\ge\; S_i + s_j\eta_{ij} && ,i = 1,...,n, j = 1,..,n \\
&S_i \;\ge\; A_i && ,i = 1,...,n \\
&S_{i+R} \;\ge\; F_i && ,i = 1,...,n-R \\
&\textstyle\sum_i \theta_{ij} = 1 && ,j = 1,...,n \\
&\textstyle\sum_j \theta_{ij} = 1 && ,i = 1,...,n \\
&\textstyle\sum_i \eta_{ij} = 1 && ,j = 1,...,n \\
&\textstyle\sum_j \eta_{ij} = 1 && ,i = 1,...,n \\
&A_1 \ge 0; A_i, S_i, F_i \text{ free}; \eta_{ij}, \theta_{ij} \in \{0,1\}, \forall i, j
\end{aligned}
$$

The first constraint defines the makespan. The second to the fifth constraints are from the arcs (Arrive, Arrive), (Start, Finish), (Arrive, Start), and (Finish, Start) in Figure 1.2, respectively. The last four constraints are the assignment constraints for scheduling. This formulation can be made tighter by changing the second and the third constraints to

$$A_{i+1} \;\ge\; A_i + \textstyle\sum_j a_j\theta_{ij} \quad ,i = 1,...,n-1$$

$$F_i \quad \geq S_i + \sum\nolimits_j s_j \eta_{ij} \quad , i = 1,\ldots,n.$$

Note that formulating the optimal scheduling model for batched service from its ERG simulation model is done by adding binary job assignment variables in a similar manner to the more general model with batch size *b* given earlier as GGRb-MIP.

Methodically formulating optimal resource scheduling programs directly from their discrete-event simulations can be a tool for confirming or for correcting intuitive ad-hoc scheduling constraint generation. Of course, we can argue only that we have derived the optimal scheduling program for the ERG, which itself might have been created intuitively. But one can easily simulate an ERG and logical errors and omissions have an opportunity for exposure.

3.3 Using Virtual Resources for Efficient Simulation and Scheduling

We next consider a somewhat more complicated scheduling problem. We show that studying the simulation model can produce a more efficient optimization model, and that studying the special structure of the optimization model can lead to a more efficient simulation. The two modeling methodologies indeed complement one another, much like looking at the same system with two different eyes.

The system that we consider is a tool commonly found in semiconductor manufacturing called a cluster tool (Figure 3.4). In the recent years, the use of cluster tools in semiconductor manufacturing has increased rapidly, causing the performance of cluster tools to become more and more important (see Chan and Schruben 2004, Dawande et al. 2002, Ding and Yi 2004, Perkinson et al. 1994, and Rostami and Hamidzadeh 2002). The cluster tool scheduling problem with residency constraints is similar to hoist scheduling problems (Shapiro and Nuttle 1988). A survey on recent development on scheduling cluster tools can be found in Dawande et al. (2005). If LP formulations for cluster tool dynamics are available, performance analysis, such as sensitivity analysis to varying processing times, robot speeds, or number of chambers as well as optimally scheduling robot moves could become easier.

Many ERG simulations of cluster tools have been developed: two of the more elegant ERGs for a generic cluster tool, both modeled with only three events, are in Nehme and Pierce (1994) and Ding and Yi (2004). In the following, we formulate the ERG given in Ding and Yi (2004) as an LP and simplify it to obtain a faster simulation model for cluster tools.

We first consider a 2-chamber cluster tool with a single-blade robot and generalize the result to a *p*-chamber cluster tool later. For clarity of exposition, the expanded (without using event parameters) ERG of the three-event ERG given in Ding and Yi (2004) for a 2-chamber single-robot cluster tool is shown in Figure 3.5. There $R(t)$ is the number of available robots at time t; $P_k(t)$ is the number of available processing slots at chamber k; $W_k(t)$ is the number of finished wafers waiting at chamber k; m_{ki} is the moving time from chamber k-1 to k; and s_{ki} is the processing time for the i^{th} wafer at chamber k, k = 1,2 (subscript k = 3 is the single load lock capable of loading and removing wafers to and from the cluster tool.). We do not consider the moving time between chambers when the robot is empty. If the time to load/unload a wafer to/from a chamber is important, it can be added into m_{ki}. In this 2-chamber example, since there is only one robot and no intermediate buffers, all state variables (while initialized at 1) can only be 0 or 1 (busy or idle).

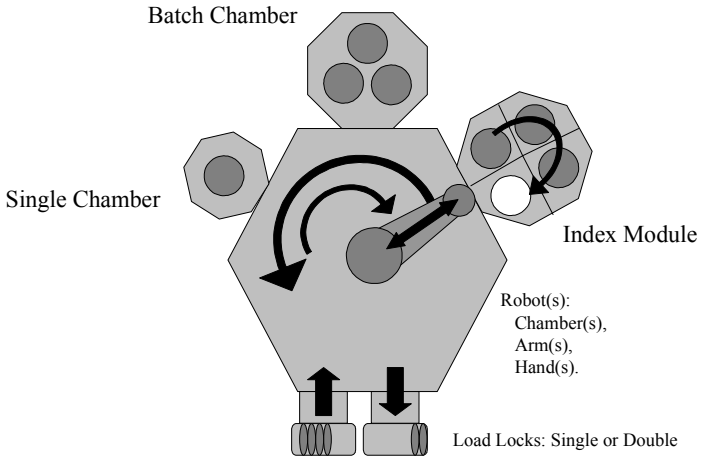


Figure 3.4: A Typical Cluster Tool

The events are *M*, *S*, and *F* for *Move*, *Start* and *Finish*, indexed for the different processing chambers with 3 again for the load lock. Let M_{1i}, S_{1i}, F_{1i}, M_{2i}, S_{2i}, F_{2i}, M_{3i}, and F_{3i} denote the times of the i^{th} occurrence of the corresponding events. In Figure 3.5, there are more than two events that increment or decrement *R*(*t*); this is case (e) in Figure A.2 of Appendix A. Applying the constraint generation procedure outlined in Section 2 and examining the resulting constraints, we find that most of these constraints are redundant. The only non-redundant constraints are constraints governing events S_1 and M_3. (Approaches for finding redundant constraints in a general math program can be found in Gal 1979.) This special characteristic of this LP suggests a new ERG simulation model (Figure 3.6) without the state variable *R*(*t*). Observe also that the arc (F_3, M_1) is no longer necessary in the new simulation ERG. The translation from a math program to an ERG, however, is carried out in an ad-hoc manner. Methodical approaches for such translations is a future research topic.

The following proposition shows the validity of the new ERG, i.e., no "phantom" robots (Schruben and Schruben 2004).

PROPOSITION 3.1. *The feasibility condition of R(t) (R(t) ≥ 0) is implied in* Figure 3.6.

PROOF. From Figure 3.5, the feasibility condition of *R*(*t*) is

$$R(t) = 1 + C_{S_1}(t) - C_{M_1}(t) + C_{S_2}(t) - C_{M_2}(t) + C_{F_3}(t) - C_{M_3}(t) \geq 0\,.$$

To prove that this condition is implied in Figure 3.6, we show that the following conditions are satisfied: (1) $C_{S_k}(t) - C_{M_k}(t) \in \{0, -1\}$, k = 1,2, and $C_{F_3}(t) - C_{M_3}(t) \in \{0, -1\}$; and (2) at any time, at most one of the three terms in (1) can be –1 and the other two terms are 0.

First consider $C_{S_1}(t) - C_{M_1}(t)$ and condition (1). Arc (M_1, S_1) implies $S_{1i} = M_{1i} + m_{1i} \geq M_{1i}$. Also, event S_1 cannot schedule an M_1 event unless it schedules some other events first, e.g., S_1 -> F_1 -> M_2 -> S_2 -> M_1. This gives $M_{1i} \geq S_{1,i-1}$. Minimizing all event

times and using (2.1), these two inequalities are equivalent to $C_{S_1}(t) - C_{M_1}(t) \in \{0, -1\}$. For condition (2), during the time period when $C_{S_1}(t) - C_{M_1}(t) = -1$, W_1 must be zero because event F_1 has not happened yet; this ensures that event M_2 cannot occur during this time and hence $C_{S_2}(t) - C_{M_2}(t) = 0$. Also, P_3 must less than or equal to 1 because event S_1 has not happened yet; this ensures that event M_3 cannot occur during this time period and hence $C_{F_3}(t) - C_{M_3}(t) = 0$. The proof is completed by extending a similar argument to the other two terms. □

The introduction of variable $P_3(t)$ makes explicit the notion of a "*virtual resource*"—defined as an object that captures redundancies in a simulation model by simulating only necessary operations on resources. In this example, $P_3(t)$ is the number of wafers in the two chambers. As seen from the optimization formulation, simulating this virtual resource provides enough information for determining the status of the robot; therefore, R can be eliminated from the ERG. Figure 3.6 can easily be generalized to model p-chamber cluster tools. For the k^{th} chamber, the three event nodes (M_k, S_k, and F_k) are added with similar state chances and conditions. The only difference is that the conditions on arcs (S_1, M_{p+1}) and (F_p, M_{p+1}) are changed to $(W_p \geq 1) \& (P_{p+1} \geq p)$.

The LP for a p-chamber cluster tool is given below. The last two constraints (with dual variables BC_{1k} and BC_{2k}) are constraints for the initial states and terminating states of the tool ('*BC*' is mnemonic for Boundary Conditions). These conditions do not exist in the 2-chamber cluster tools. The throughput of the tool can be computed by calculating the reciprocal of the difference between two consecutive finish events at a chamber when the tool reaches a steady state, for example, $(F_{ki} - F_{k,i-1})^{-1}$.

CLp-LP:

$$\begin{aligned}
&\min \textstyle\sum_{k=1}^{p+1} \sum_{i=1}^{n} F_{ki} \\
&s.t. \\
&F_{ki} - F_{k+1,i-1} \geq m_{ki} + s_{ki} - s_{k+1,i-1} && ,k = 1,\ldots,p, i = 1,\ldots,n && (P_k) \\
&F_{k,i} - F_{k-1,i} \geq m_{k,i} + s_{k,i} && ,k = 2,\ldots,p, i = 1,\ldots,n && (W_k) \\
&F_{p+1,i} - F_{1,i+p-1} \geq m_{p+1,i} - s_{1,i+p-1} && ,i = 1,..,n-(p-1) && (P_{p+1}) \\
&F_{p+1,i} - F_{pi} \geq m_{p+1,i} && ,i = 1,\ldots,n && (W_{p+1}) \\
&F_{k,1} - F_{1,k-1} \geq m_{k,1} + s_{k,1} - s_{1,k-1} && ,k = 3,\ldots,p && (BC_{1k}) \\
&F_{p+1,n-(p-k)} - F_{k,n} \geq m_{p+1,n-(p-k)} - s_{k,n} && ,k = 2,\ldots,p-1 && (BC_{2k}) \\
&F_{ki} \text{ free}, s_{p+1,i} = 0, \forall k, i
\end{aligned}$$

Figure 3.7 shows some of the benefits gained from using the new LP and the new simulation model together: (1) starting with the original simulation model (whose run time is depicted by the dashed line in Figure 3.7.a); (2) we formulated this simulation model as an LP (whose run time is given by the dashed line in Figure 3.7.b); (3) we then reduced the size of this LP by eliminating redundant constraints and achieved a new LP (whose run time is given by the solid line in Figure 3.7.b); (4) we then came full circle and used the new LP to construct a new simulation model in Figure 3.6 (whose run time is depicted by the solid line in Figure 3.7.a). The execution efficiency of both the LP model and the ERG simulation were dramatically improved by exploiting their individual special structures. Figure 3.7 does not include the time spent in devel-

oping the new LP and the new simulation model. However, once they are developed, we could, if necessary, run the new simulation model for many times. Thus, the saving is in a magnitude of the number of runs times the difference between the speeds of the new and old models.

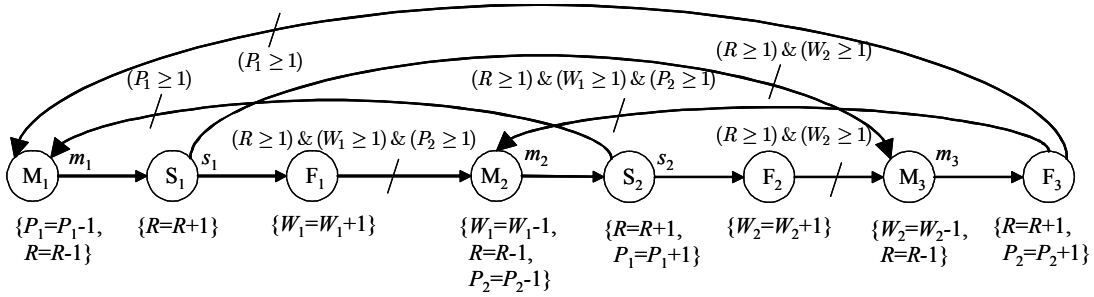


Figure 3.5: Original ERG Simulation Model for a 2-chamber Cluster Tool (Ding and Yi 2004)

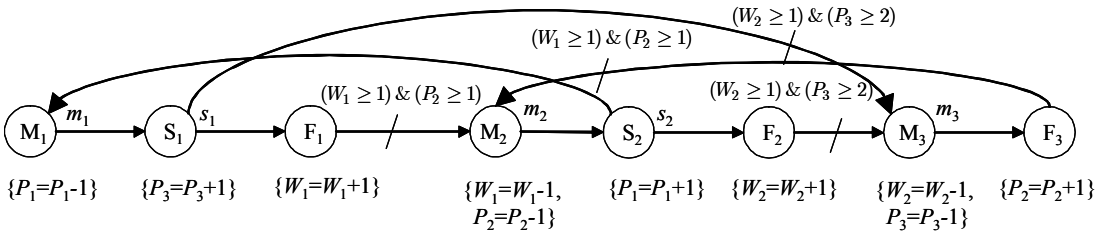
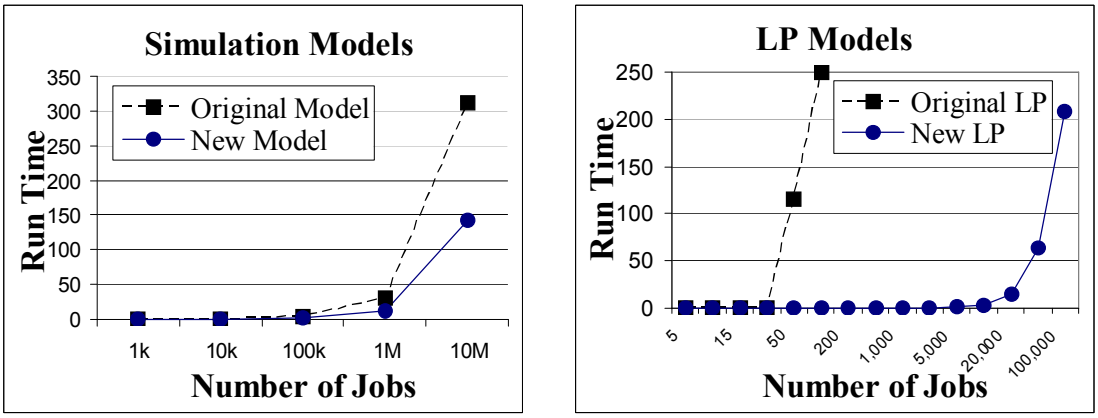


Figure 3.6: New ERG Simulation Model (without State Variable R)



(a) Run Times of the Simulation Models (b) Run Times of the LP Models

Figure 3.7: Original Simulation Model → Original LP → New LP → New Simulation Model

REMARK 3.2. Through the derivation from the original model to the new one, we also showed that the so-called "*push*" schedule is optimal for cluster tools under sequential processing—in a

sequential schedule, since jobs must be processed in sequence, the optimal schedule is to push jobs into the system as early as possible (see e.g., Perkinson et al. 1994, Dawande et al. 2002). ◊

REMARK 3.3. For complex systems, it could be very difficult to identify all redundant constraints. However, many optimization software now have presolver routines that can sometimes identify redundant constraints; therefore, better (not necessary optimal) simulation models may still be achievable. This could dramatically reduce the CPU times for optimization algorithms involving thousands of calls to the simulation model—in the cluster tools literature, most algorithms use simulation to evaluate the throughput of the tool at each iteration of a scheduling heuristics (see e.g., Rostami and Hamidzadeh 2002). ◊

3.4 Equivalent Linear Programs and Event Reduction in Their Corresponding Simulations

Different ERGs can be created to model the same discrete-event system. However, some of these ERGs could be equivalent in the sense that all their sample paths are identical given the same input. Some of these have distinct mathematical programming representations. These mathematical programs also may have distinct but equivalent formulations that correspond to different simulation models of a system.

For example, the mathematical program, GGR-LP1 for a single server queue (i.e., GG1-LP1) was generated directly from the ERG in Figure 1.2 for simulating this system. Other ERGs can be used to simulate this system and each of them will lead us to a different mathematical program. In the simulation literature it is common to model a *G*/*G*/1 queue using only two events, a customer Arrival event and a Finish service event (see, for example, page 57 of Law and Kelton 2000). In fact, the Start event is not necessary, since it is always scheduled with zero delay so its associated state changes could be conditionally incorporated in the state changes for the Arrival and Finish events, making these events more complex, but reducing the number of events by one (see page 58 of Law and Kelton 2000). Heuristics and algorithms for eliminating redundant events in an ERG have been proposed (Schruben 1983 and Som and Sargent 1990, Seila, Ceric and Tadikamalla 2003).

We also discovered that the Finish event is not necessary in the simulation model by reformulating its LP representation. This is somewhat surprising since we could not find any simulation of a *G*/*G*/1 queue in the literature that contains only a single Start event—although there are simulations of *M*/*G*/1 queues with only a Finish event (Schruben and Schruben 2000) and *G*/*G*/*R* queues with only a single Finish event (Chan 2005).

In GG1-LP1, the first constraint is only for the Arrival event times, which are input data for the simulation model and therefore can be removed from the formulation. The second constraint, being an equality, can also be eliminated from the formulation by incorporating it into the last two constraints, that is, substituting $F_i - s_i$ for S_i, $i = 1,...,n$. Dropping the constant terms, i.e., $\sum_{i=1}^{n}(a_i + s_i)$, from the objective function yields an LP for the *G*/*G*/1 queue that has only Finish event times as its variables,

GG1-LP1(*F*):

$$\begin{array}{lll}\min & \sum_{i=1}^{n} F_i & \\ st. & & \\ F_i & \geq A_i + s_i \ , i = 1,...,n & (U_i) \\ F_i - F_{i-1} & \geq s_i \qquad , i = 2,...,n & (V_i) \\ F_i \text{ free}, \forall i & & \end{array}$$

A more interesting LP for a *G*/*G*/1 queue is obtained by replacing F_i with $S_i + s_i$, $i = 1,...,n$, in the last constraint in GG1-LP1, resulting in a formulation for a *G*/*G*/1 queue with only Start events as variables,

GG1-LP1(*S*):

$$\begin{array}{lll}\min & \sum_{i=1}^{n} S_i & \\ st. & & \\ S_i & \geq A_i \quad , i = 1,...,n & (U_i) \\ S_i - S_{i-1} & \geq s_{i-1} \ , i = 2,...,n & (V_i) \\ S_i \text{ free } \forall i & & \end{array}$$

This is another example where developing ERG models from their equivalent optimization formulations can sometimes give different, and potentially more efficient, simulations.

3.5 Queueing System Duality

GG1-LP1(*S*) can be modified to a new formulation with an objective function of minimizing the average job waiting time ($W = \frac{1}{n}\sum_{i=1}^{n}(S_i - A_i)$) by replacing S_i with $W_i + A_i$, $i = 1,...,n$, in the formulation and dropping the constant term of sum of the A_i's from the objective function and dividing it by *n*. This new formulation and its dual are

GG1-LP1(*W*):

$$\begin{array}{lll}\min & \frac{1}{n}\sum_{i=1}^{n} W_i & \\ st. & & \\ W_i & \geq 0 \qquad\quad , i = 1,...,n & (U_i) \\ W_i - W_{i-1} & \geq s_{i-1} - a_i \ , i = 2,...,n & (V_i) \\ W_i \text{ free } \forall i & & \end{array}$$

GG1-LP1-Dual(*W*):

$$\begin{array}{llll}\max & \frac{1}{n}\sum_{i=1}^{n}(s_i - a_{i+1})V_i & & \\ st. & & & \\ U_1 & -V_2 & = 1 & (W_1) \\ U_i + V_i & -V_{i+1} & = 1 \ , i = 2,...,n-1 & (W_i) \\ U_n + V_n & & = 1 & (W_n) \\ U_i, V_i \geq 0, \forall i & & & \end{array}$$

Here U_i, and V_i, are the corresponding dual variables for each constraint. GG1-LP(*W*) is equivalent to the well-known Lindley recursion (Lindley 1952), $W_i = \max\{W_{i-1} + s_{i-1} - a_i, 0\}$. In fact, many of the max-plus representations of stochastic system sample paths have direct LP representations and hence, duals.

This dual looks at the dynamics of a $G/G/1$ queue from the server's viewpoint (reminiscent of Palm Calculus, Baccelli and Bremaud 2003). The LP GG1-LP-Dual(W), when expressed as a minimization, is equivalent to the classical uncapacitated lot-sizing problem with only inventory holding costs: The $G/G/1$ FIFO queue is the sample path dual of an uncapacitated lot-sizing problem and vice versa. This gives us a new way of thinking about queueing system dynamics.

The constraints in GG1-LP1-Dual(W) are similar to the inventory balance equation, where U_i is the production quantity at period i and V_i is the inventory level carried from period $i-1$ to period i. Each customer requires a single unit of "customer service" (or simply, a service unit) which is produced by the server and can also be stored in inventory. Each customer's departure represents a potential production period of the server. Upon the departure of a customer, the server can either produce a new service unit for the next customer in line or use a service unit in inventory. At the beginning of each busy period, the server needs to decide how many service units (the U's) to produce. After the first customer is served, the rest of the service units (the V's) will be stored in inventory and can be used throughout the rest of the current busy period. The penalty of producing too many service units during a busy period is the service holding cost (i.e., the time waiting for an arrival $a_{i+1} - s_i$). In the current model, since the costs from node A_0 to node W_i, $i = 1,...,n$, are zero, there is no setup cost or penalty cost for producing too few service units during a busy period; therefore, the server will produce just enough to satisfy all the customers in a busy period.

3.6 Response Gradient Estimation

Infinitesimal perturbation analysis (IPA) is a technique for estimating the gradient of a system performance measure by observing the sample path from a single simulation run. There are at least two ways of computing the sample path derivatives. The first one computes the sample path derivatives by using an algorithm integrated into a simulation (Ho et al. 1979, Ho and Cao 1991, Glasserman 1991, Fu and Hu 1997, Suri and Zazanis 1988, Freimer and Schruben 2001). The second approach makes use of linear programming duality. For example, Homem-de-Mello et al. (1999) gives a network formulation for unlimited buffer tandem queues and calculates sample-path derivatives from the dual variables. Kim (2006) considers the newsvendor problem and derives an IPA estimator from the dual variables. Our method extends this second approach. Specifically, we use the dual variables of a mathematical programming representation for an ERG to compute the sample path derivatives for the underlying simulation.

We will focus our discussion on formulations without integer variables. Consider an LP generated by ERG2MP. The dual variables (shadow prices) represent how sensitive the objective function (event times) is to changes in the right-hand-side random variables (input data) and therefore, provide information necessary for computing gradient estimators using the chain-rule as done in IPA gradient estimation. In fact, perturbations are propagated through all the binding constraints (constraints with zero excess) and the value of each dual variable represents the marginal effect of the corresponding right-hand-side random variable to the objective function. Therefore, all the binding constraints constitute an event-tree (the solution of the dual LP) similar to the one defined in Suri (1987). Moreover, the LP solution obtained from running the simulation might provides more information for a single simulation run because other perturbed sample paths can be reached from the current sample path by some additional computation (pivots), which might be easier than running a new simulation. From the computational point of view, the

LP representations would be a potentially effective tool for other types of sensitivity analysis (in particularly, finite difference gradient analysis) when IPA fails, for example, using the dual-simplex method to get new sample paths. Performance of such sensitivity analysis is under investigation.

Let us now examine the consistency property of IPA gradient estimators computed using the dual variables. For ease of exposition, we shall consider the average waiting time of a *GI*/*G*/1 queue. Derivatives of other performance measures in which IPA works can also be computed using the dual variables (see discussion under Theorem 2.1). Let $\boldsymbol{d}(\theta) = (d_1(\theta), d_2(\theta),\ldots,d_m(\theta))$ be the right-hand-side vector which is a function of parameter θ, and $\boldsymbol{u} = (u_1, u_2,\ldots,u_m)$ be a vector of dual variables, where m is the number of constraints, usually greater than n—the number of variables. Dividing the objective function by n (this will not alter the optimal basis) and taking the limit $n \to \infty$ gives a consistent estimator of the average waiting time, i.e., $\bar{W}(\theta) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} W_i^*$ a.s., where W_i^*'s are the optimal primal solutions, or equivalently working with the dual, $\bar{W}(\theta) = \lim_{n\to\infty} \frac{1}{n} z(\theta) = \lim_{n\to\infty} \frac{1}{n} \boldsymbol{d}(\theta)\boldsymbol{u}^*$ a.s., where $\boldsymbol{u}^*$ is the optimal dual vector. For a small perturbation $\Delta\theta$ provided that the order of events remains unchanged—an usual assumption of IPA—the current dual variables remain optimal and therefore, the objective function is perturbed by an amount of $\Delta z = \Delta \boldsymbol{d}\boldsymbol{u}^*$, where $\Delta\boldsymbol{d}$ is the amount of perturbation of the right-hand side due to the change in θ. The change to the average waiting time is then

$$\bar{W}(\theta + \Delta\theta) - \bar{W}(\theta) = \lim_{n\to\infty} \frac{1}{n} [\boldsymbol{d}(\theta + \Delta\theta) - \boldsymbol{d}(\theta)]\boldsymbol{u}^* = \lim_{n\to\infty} \frac{1}{n} \Delta \boldsymbol{d}\boldsymbol{u}^* \text{ a.s.}$$

Divided by $\Delta\theta$ and letting $\Delta\theta \to 0$ yields the derivative of the average waiting time. Now, using the same assumptions typically made in IPA, i.e., the random variables $d_i(\theta)$'s are uniformly differentiable—a condition such that the random variables are smooth enough or well-behavior so that IPA works (see Cao 1985 or Ho and Cao 1991 for more details), we have

$$\begin{aligned} \frac{d\bar{W}(\theta)}{d\theta} &= \lim_{\Delta\theta\to 0} \lim_{n\to\infty} \frac{1}{n} \frac{\Delta \boldsymbol{d}(\theta)}{\Delta\theta} \boldsymbol{u}^* \qquad a.s. \\ &= \lim_{\Delta\theta\to 0} \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{m} \frac{\Delta d_i(\theta)}{\Delta\theta} u_i^* \\ &= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{m} d_i'(\theta) u_i^* \end{aligned}$$

where $d_i'(\theta)$ is the derivative of $d_i(\theta)$ w.r.t. θ (assume exists) and the last equation uses the uniform differentiability condition. Observe that as n goes to infinity, so does m. Therefore, the dual variables provide a consistent estimator for the derivative of the average waiting time under the usual IPA assumptions. If θ is a location or scale parameter of the arrival or service distributions, then the chain rule can be used in the usual IPA manner to compute the gradient estimates.

Gradient estimators for tandem queueing networks can also be calculated directly from the shadow prices of TQmC-LP(*F*), TQmF-LP(*F*), or TQmG-LP(*F*). It can also be shown (Chan 2005) that the service time shadow process (values for the U_{ki}, V_{ki}, and W_{ki}'s) are the number of

customers in a busy period or a local busy period (for definition of local busy period, see Fu and Hu 1997).

4 CONCLUSION

This paper is intended to further develop the idea of expressing the dynamics of a discrete-event stochastic system as the solutions to optimization programs, first proposed by Schruben in (2000), as a general framework. ERGs are a convenient, completely general, way of defining the dynamics of a discrete-event system and are used here. Other approaches to modeling discrete-event dynamics, most notably the use of max-plus algebra, Petri Nets, process flows, and generalized semi-Markov processes can also be translated into ERGs and hence into optimization programs. The goal of this paper is to demonstrate a close relationship between discrete-event simulations to optimization programs, permitting the application of the rich mathematical theory and algorithms of optimization to the study of discrete-event stochastic systems. It is also hoped that further research into these relationships might provide insights to develop new algorithms for solving some hard deterministic optimization problems using properties of their equivalent ERG simulation models.

ACKNOWLEDGMENTS

The authors wish to thank Professor J. George Shanthikumar for his helpful suggestions and comments on this paper. The encouragement, insights, and criticisms of the referees and Associate Editor to an earlier version of this paper resulted in significant improvements and extensions. The authors also appreciate the support for this research by the National Science Foundation through grant DMI0323765 to the University of California, Berkeley.

APPENDIX A

Available online in the *Operations Research* web site.

APPENDIX B

In short, an LP generated by ERG2MP has a dual of network-type formulations. An objective function of minimizing all event times in the primal is equivalent to an objective function of finding an earliest feasible processing path (longest path) from the first node (the first event of a simulation) to all nodes (all events in a simulation) in the dual. This will result in a tree where nodes are events and branches are relationships among events. A more formal argument is given below.

We first note that the constraints generated for unconditional timed arcs and conditional untimed arcs are of the types $x_i - x_j = d_i$ and $x_i - x_j \geq 0$, respectively. Both constraints are of network type; therefore, each row of the constraint matrix has one +1 and one –1 entries and zeroes in all the other entries. In addition, the generated constraints are valid and complete because the facts that the causal relationships among the events in an ERG are governed completely by the arcs and their conditions; and that the constraints are derived by using the equivalency relation in (2.1).

Let $\min\left\{\sum_{i=1}^{n} x_i : \boldsymbol{A}\boldsymbol{x} \geqq \boldsymbol{d}, \boldsymbol{x} \in \boldsymbol{R}^n\right\}$ be a generated LP, where $\boldsymbol{R}^n$ is the set of real n-dimensional vectors, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is the event-times variables, $\boldsymbol{A}$ is the $m \times n$ constraint matrix containing one +1 and one –1 in each row and all the other entries are zeroes, and $\boldsymbol{d} \in \boldsymbol{R}_+^m$ is an m-vector of nonnegative-real right-hand sides, where m is the number of constraints. The operator "$\geqq$" here means that some constraints are equality and some are inequality. All LP examples given in this paper are of this type.

PROOF of Theorem 2.1:
The proof is by induction on the number of events, k, in the sample path. We first show that the LP can be solved variable by variable. That is, to solve an LP with k+1 event-time variables, one can first solve the LP with k event-time variables and then use the results to solve the LP with k+1 event-time variables. We use superscript "[k]" to associate notation with an LP that has k event-time variables. It suffices to show that the optimal solution $\boldsymbol{x}^{*[k]}$ for LP$^{[k]}$ is still in the optimal solution for LP$^{[k+1]}$, i.e., $\boldsymbol{x}^{*[k+1]} = ({x_1}^{*[k+1]}, \ldots, {x_k}^{*[k+1]}, {x_{k+1}}^{*[k+1]}) = ({x_1}^{*[k]}, \ldots, {x_k}^{*[k]}, {x_{k+1}}^{*[k+1]})$, and ${x_{k+1}}^{*[k+1]}$ is determined by $\boldsymbol{x}^{*[k]}$. Since this should be true for all k, our goal is to show $\boldsymbol{x}^{*[k+1]} = ({x_1}^{*[k+1]}, \ldots, {x_{k+1}}^{*[k+1]}) = ({x_1}^{*[1]}, {x_2}^{*[2]}, \ldots, {x_k}^{*[k]}, {x_{k+1}}^{*[k+1]})$.

By induction, when $k = 1$, there is only one event and the constraints would be $x_1 = d_1$, $x_1 \geq d_1$, or both; therefore, minimizing x_1 gives optimal solution $\boldsymbol{x}^{*[1]} = ({x_1}^{*[1]}) = (d_1)$. The second event can also be an initial event similar to the first event (i.e., $x_2 = d_2$ or $x_2 \geq d_2$) or it can be scheduled by the first event (i.e., $x_2 - x_1 = d_2$ or $x_2 - x_1 \geq d_2$). In either case, by minimizing $x_1 + x_2$ the optimal solution $\boldsymbol{x}^{*[1]}$ for LP$^{[1]}$ is still optimal for x_1 in LP$^{[2]}$, yielding $\boldsymbol{x}^{*[2]} = ({x_1}^{*[2]}, {x_2}^{*[2]}) = ({x_1}^{*[1]}, {x_2}^{*[2]})$, which equals either (d_1, d_2) or (d_1, d_1+d_2). Therefore, ${x_1}^{*[2]}$ and ${x_2}^{*[2]}$ are the earliest time for the first and second events to occur, respectively. Using a similar argument, one can show that $\boldsymbol{x}^{*[3]} = ({x_1}^{*[3]}, {x_2}^{*[3]}, {x_3}^{*[3]}) = ({x_1}^{*[1]}, {x_2}^{*[2]}, {x_3}^{*[3]})$ and ${x_3}^{*[3]}$ is the earliest feasible time for the third event to occur.

Next, assume that $\boldsymbol{x}^{*[k]} = ({x_1}^{*[1]}, \ldots, {x_k}^{*[k]})$ is the optimal solution for LP$^{[k]}$, i.e., the earliest time for the 1st, 2nd,…,kth events. Now, consider LP$^{[k+1]}$. Since x_{k+1} must be scheduled by some previous event, which is realized by constraints $x_{k+1} - x_j \geq d_{k+1},\ \forall j \in \Gamma(k+1)$, where $\Gamma(k+1)$ is the set of all possible scheduling events for the k+1th event, any feasible solution for x_{k+1} must satisfy

$$x_{k+1}{}^{[k+1]} \geq {x_j}^{*[k]} + d_{k+1},\ \forall j \in \Gamma(k+1). \qquad \text{(A.2)}$$

Let ${x_{k+1}}^{*[k+1]} = \max_{j \in \Gamma(k+1)} \{{x_j}^{*[k]}\} + d_{k+1}$ be the earliest time for the k+1th event to occur. We first observe that $({x_1}^{*[k]}, \ldots, {x_k}^{*[k]}, {x_{k+1}}^{*[k+1]})$ is feasible for LP$^{[k+1]}$ because $({x_1}^{*[k]}, \ldots, {x_k}^{*[k]})$ is feasible for LP$^{[k]}$ and LP$^{[k+1]}$ includes only constraints from LP$^{[k]}$ and constraints A.2 (constraints A.2 only enforce the earliest time for the k+1th event). Second, $({x_1}^{*[k]}, \ldots, {x_k}^{*[k]}, {x_{k+1}}^{*[k+1]})$ is also optimal for LP$^{[k+1]}$. This is true because for any feasible solution $\boldsymbol{x}'^{[k+1]} = ({x_1}'^{[k+1]}, \ldots, {x_k}'^{[k+1]})$ for LP$^{[k+1]}$, we have an objective function

$$\begin{aligned}\sum_{i=1}^{k+1} {x_i'}^{[k+1]} &= \sum_{i=1}^{k} {x_i'}^{[k+1]} + {x'_{k+1}}^{[k+1]} \\ &\geq \sum_{i=1}^{k} {x_i}^{*[k]} + {x'_{k+1}}^{[k+1]} \\ &\geq \sum_{i=1}^{k} {x_i}^{*[k]} + \max_{j\in\Gamma(k+1)} \left\{ {x_j}^{*[k]} \right\} + d_{k+1}, \\ &= \sum_{i=1}^{k} {x_i}^{*[i]} + {x_{k+1}}^{*[k+1]} \\ &= \sum_{i=1}^{k+1} {x_i}^{*[k+1]}\end{aligned}$$

where the first inequality is because $\sum_{i=1}^{k} {x_i}^{*[k]}$ is the optimal solution for LP$^{[k]}$ (by assumption) and $(x_1'^{[k+1]},\ldots,x_k'^{[k+1]})$ is also a feasible solution for LP$^{[k]}$ (since all constraints in LP$^{[k]}$ are included in LP$^{[k+1]}$); and the second inequality follows from A.2. Multiple optimal solutions could happen, which is the case in an event scheduling simulation.

The proof is then complete by noting that the event scheduling function for a discrete-event simulation of an ERG executes events one by one in the same manner as the LP is solved variable by variable above (see, for example, Law and Kelton, 2000). □

REFERENCES

Askin, R.G. and C. R. Standridge (1993), *Modeling and Analysis of Manufacturing Systems*, New York, Wiley.

Baccelli, F. and P. Bremaud (2003). *Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences*, Springer Verlag.

Baccelli, F., G. Cohen, G. J. Olsder and J. P. Quadrat (1992). *Synchronization and Linearity: An Algebra for Discrete-event Systems*, Wiley.

Buss, A. H. and Sanchez, P. J. (2002). Building Complex Models with LEGOs (Listener Event Graph Objects). *Proceedings of the 2002 Winter Simulation Conference (Cat. No.02CH37393).* IEEE. Part vol.1, 2002, pp.732-737. San Diego, CA, USA.

Buzacott, J. A. and J. G. Shanthikumar (1993). *Stochastic Models of Manufacturing Systems*, Prentice Hall, New Jersey.

Cao, X. R. (1985). Convergence of Parameter Sensitivity Estimates in a Stochastic Experiment. *IEEE Transactions on Automatic Control* 30(9): 845-853.

Chan, W. K.V. and L. W. Schruben (2003). Properties of Discrete-event Systems from Their Mathematical Programming Representations. *Proceedings of the 2003 Winter Simulation Conference (IEEE Cat. No.03CH7499).* IEEE. Part vol.1, 2003, pp.496-502. Piscataway, NJ, USA.

Chan, W. K. V. and Schruben, L. W. (2004). Generating Scheduling Constraints for Discrete Event Dynamic Systems. *Proceedings of the 2004 Winter Simulation Conference (IEEE Cat. No.04CH37614C),* IEEE. Part vol.2, 2004, pp.568-576. Piscataway, NJ, USA.

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

Cheng, D. W. (1995). Line Reversibility of Tandem Queues with General Blocking. *Management Science* 41(5): 864-73.

Cheng, D. W. (1997). Line Reversibility of Multiserver Systems. *Probability in the Engineering and Informational Sciences* 11: 177-188.

Cohen, G., S. Gaubert and J. P. Quadrat (1999). Max-plus Algebra and System Theory: Where We Are and Where to Go Now. *Annual Reviews in Control* 23: 207-19.

Dattatreya, E. S. (1978). *Tandem Queueing Systems with Blocking*, Ph.D. dissertation, University of California, Berkeley.

Dawande, M., C. Sriskandarajah and S. Sethi (2002). On Throughput Maximization in Constant Travel-Time Robotic Cells. *Manufacturing & Service Operations Management* 4(4): 296-312.

Dawande, M., H. N. Geismar, S. P. Sethi and C. Sriskandarajah (2005). Sequencing and Scheduling in Robotic Cells: Recent Developments. *Journal of Scheduling* 8(5): 387-426.

Ding, S. and J. Yi (2004). An Event Relationship Graph Based Simulation and Analysis of Multi-Cluster Tools. *Proceedings of the 2004 Winter Simulation Conference (IEEE Cat. No.04CH37614C),* IEEE. Part vol.2, 2004, pp.1915-1924. Piscataway, NJ, USA.

Freimer, M. and L. Schruben (2001). Graphical representation of IPA estimation. *Proceeding of the 2001 Winter Simulation Conference (Cat. No.01CH37304)*. IEEE. Part vol.1, 2001, pp.422-427. Piscataway, NJ, USA.

Fu, M. and J. Q. Hu (1997). *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*, Kluwer Academic Publishers, Boston.

Gal, T. (1979). *Postoptimal Analyses, Parametric Programming and Related Topics*, McGraw-Hill, London.

Glasserman, P. (1991). *Gradient Estimation via Perturbation Analysis*, Kluwer Academic Publishers, Boston.

Gondran, M. and M. Minoux (1984). Linear Algebra in Dioids: A Survey of Recent Results. *Annals Discrete Math* 19: 147-164.

Hooker, J. N. (2002). Logic, Optimization, and Constraint Programming. *INFORMS Journal on Computing* 14(4): 295-321.

Ho, Y. C., M. A. Eyler and T. T. Chien (1979). A Gradient Technique for General Buffer Storage Design in a Production Line. *International Journal of Production Research* 17(6): 557-580.

Ho, Y. C. and X. Cao (1991). *Perturbation Analysis of Discrete-event Dynamic Systems*, Kluwer Academic Publishers, Boston.

Homem-de-Mello, T., A. Shapiro and M. L. Spearman (1999). Finding Optimal Material Release Times Using Simulation-Based Optimization. *Management Science* 45(1): 86-102.

Hyden, P., L. Schruben and T. Roeder (2001). Resource Graphs for Modeling Large-Scale, Highly Congested Systems. *Proceeding of the 2001 Winter Simulation Conference (Cat. No.01CH37304).* IEEE. Part vol.1, 2001, pp.523-529. Piscataway, NJ, USA.

Kim, S. (2006). Gradient-Based Simulation Optimization. *Proceeding of the 2006 Winter Simulation Conference. IEEE, pp.159-167, Piscataway, NJ, USA.*

Law, A. and W. D. Kelton (2000). *Simulation Modeling and Analysis*, McGraw-Hill.

Lindley, D. V. (1952). The Theory of Queues with a Single Server. *Proceedings of the Cambridge Philosophical Society* 48(2): 277-289.

Maxwell, W. L. and R. C. Wilson (1981). Dynamic Network Flow Modeling of Fixed Path Material Handling Systems. *IIE Transactions* 13(1): 12-21.

Muth, E. J. (1979). The Reversibility Property of Production Lines. *Management Science* 25(2): 152-8.

Nehme, D. A. and N. G. Pierce (1994). Evaluating the Throughput of Cluster Tools using Event-Graph Simulations. *IEEE/SEMI 1994 Advanced Semiconductor Manufacturing Conference and Workshop. Theme - Manufacturing Excellence: A Global Challenge. ASMC '94 Proceedings (Cat. No.94CH3475-1).* IEEE. 1994, pp.189-92. New York, NY, USA.

Nemhauser, G. L. and L. A. Wolsey (1999). *Integer and Combinatorial Optimization*, Wiley, New York.

Onvural, R. O. (1990). Closed Queueing Networks with Blocking. *Stochastic Analysis of Computer and Communication Systems*. Takagi, Hideaki, Elsevier Science Publishers B.V. (North-Holland): 499-528.

Pegden, C. D. (1986). *Introduction to SIMAN*, 2nd ed., Systems Modelin Corporation.

Pinedo, M. (2002). *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, Upper Saddle River, N.J.

Perkinson, T. L., P. K. McLarty, R. S. Gyurcsik and R. K. Cavin, III (1994). Single-Wafer Cluster Tool Performance: An Analysis of Throughput. *IEEE Transactions on Semiconductor Manufacturing* 7(3): 369-373.

Savage, E. L., L. W. Schruben and E. Yucesan (2004). On the Generality of Event Relationship Graph Models. *INFORMS Journal on Computing* 16(4).

Schruben, D. and L. W. Schruben (2000). *Graphical Simulation Modeling using SIGMA*, Custom Simulation.

Schruben, L. (1983). Simulation Modeling with Event Relationship Graphs. *Communications of the ACM* 26(11): 957-963.

Schruben, L. and Yucesan, E. (1994). Transforming Petri Nets into Event Graph Models. *Proceedings of the 1994 Winter Simulation Conference (IEEE Cat. No. 94CH35705). IEEE. 1994, pp.560-565. New York, NY, USA.*

Schruben, L. W. (2000). Mathematical programming models of discrete event system dynamics. *Proceedings of the 2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165). IEEE. Part vol.1, 2000, pp.381-385. Piscataway, NJ, USA.*

Schruben, L. W. and T. M. Roeder (2003). Fast simulations of Large-Scale Highly Congested Systems. *Simulation, Transactions of the Society for Modeling and Simulation International* 79(3): 115-125.

Seila, A. F., V. Ceric and P. Tadikamalla (2003). *Applied Simulation Modeling*, Thomson.

Som, T. K. and R. G. Sargent (1990). A Formal Development of Event Relationship Graphs as an Aid to Structured and Efficient Simulation Programs. *ORSA Journal on Computing* 1(2): 107-25.

Suri, R. (1987). Infinitesimal Perturbation Analysis for General Discrete-event Systems. *Journal of the ACM* 34(3): 686-717.

Suri, R. and M. A. Zazanis (1988). Perturbation Analysis Gives Strongly Consistent Sensitivity Estimates for the M/G/1 Queue. *Management Science* 34(1): 39-64.

Williams, H. P. (1995). Logic Applied to Integer Programming and Integer Programming Applied to Logic. *European Journal of Operational Research* 81(3): 605-616.

Wu, J. H. and C. N. Chung (1991). *Timed Finite Automata as The Theoretical Foundation for Simulation Modeling with Event Relationship Graphs.* Technical Report, Department of Decision Science. and Information Systems, University of Kentucky, Lexington, KY.

Yamazaki, G., H. Sakasegawa and T. Kawashima (1978). Production Rate Estimated with Flow-Shop Reversibility. *Bulletin of the JSME* 21: 167-171.

Yucesan, E. and L. Schruben (1992). Structural and Behavioral Equivalence of Simulation Models. *ACM Transactions on Modeling and Computer Simulation* 2(1): 82-103.

Zeigler, B. P. A. (1984). *Theory of Modelling and Simulation*, Krieger Publishing Company, Melbourne.