Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

EVENT RELATIONSHIP GRAPH LITE: EVENT BASED MODELING FOR SIMULATION–OPTIMIZATION OF CONTROL POLICIES IN DISCRETE EVENT SYSTEMS

Andrea Matta

Shanghai Jiao Tong University
800 Dong Chuan Road
Shanghai, 200240, CHINA

Giulia Pedrielli

National University of Singapore
12 Prince George's Park
Singapore, 118411, SINGAPORE

Arianna Alfieri

Politecnico di Torino
24 corso Duca degli Abruzzi
Torino, 10129, ITALY

ABSTRACT

Simulation–optimization has received a spectacular attention in the past decade. However, the theory still cannot meet the requirements from practice. Decision makers ask for methods solving a variety of problems with diverse aggregations and objectives. To answer these needs, the interchange of solution procedures becomes a key requirement as well as the development of (1) general modeling methodologies able to represent, extend and modify simulation–optimization as a unique problem, (2) mapping procedures between formalisms to enable the use of different tools. However, no formalism treats simulation–optimization as an integrated problem. This work aims at partially filling this gap by proposing a formalism based upon Event Relationship Graphs (ERGs) to represent the system dynamics, the problem decision variables and the constraints. The formalism can be adopted for simulation–optimization of control policies governing a queueing network. The optimization of a Kanban Control System is proposed to show the whole approach and its potential benefits.

1 INTRODUCTION

Simulation–optimization theory and practice still have to converge for satisfying the needs of decision makers, who try to solve complex problems in limited time (Fu 2002). At various stages of the decision process, problems are characterized by remarkably different levels of detail. Moreover, the decision maker considers different variables and performance to control as well as several costs/profits (Schruben 2010, Schruben 2013). For instance, at the early stage of a decision process, rough models are enough for identifying the most promising alternatives, notwithstanding the need to have more accurate representations as the best designs need to be compared. New methods are needed to support the decision maker solving this variety of problems. In light of this, the possibility to interchange solution procedures while solving a specific problem would represent a spectacular advantage. However, there exist neither a simulation nor an optimization tool which can serve such a purpose.

In fact, these needs reveal two main gaps that scientific research has not addressed yet: (1) *modeling* gap substantiated in the lack of a general modeling methodology able to represent and easily extend/modify simulation–optimization problems; (2) lack of *mapping procedures* to transform the simulation-optimization model into different languages, thus enabling the use of multiple tools to solve it.

Concerning the modeling methodologies, most of the research was devoted to develop simulation or optimization rather than simulation–optimization formalisms. Schruben (1983) proposed the Event Relationship Graph (ERG), a general language for modeling and simulation of discrete event systems (DESs). ERGs have been demonstrated to be able to simulate a Turing machine and have been successfully applied to the evaluation of the performance of DESs. Moreover, ERG solving optimization problems were proposed (Savage et al. 2005, Chan 2005). In Liu et al. (2012), an ERG is automatically generated from real time data to first simulate and then optimize the system. The LEGO framework was proposed to develop simulation model components using ERGs (Buss and Sanchez 2002). In the computing area, several modeling techniques are used for simulation and property verification, but few have the ERG modeling power and generality. An ERG can model a petri net but not vice versa. The well known GSMP (Generalized Semi-Markov Process) and DEVS (Discrete Event System Specification (Zeigler 1976)) formalisms have the same modeling power of ERG (Savage, Schruben, and Yücesan 2005) but they are not easy to use in many practical applications.
State based formalisms (e.g., finite state automata) manifest their drawbacks when complex systems composed of several components are modeled due to the state space growth, which is typically faster than the increase in the number of events (Cassandras and Lafortune 2008, Cao and Zhang 2008). Also DEVS and GSMP (Iglehart and Shedler 1983), despite their generality, suffer from the state space growth problem.

Concerning the mapping procedures, Chan and Schruben (2008) proved a general scheme to translate ERG models into their mathematical programming counterpart. However, this was done in the scope of simulation. The translation into mathematical programming was then extended to simulation–optimization of multi–stage tandem queueing systems in Matta (2008), Alfieri and Matta (2012). Recently, Latorre and Jiménez (2013) proposed a tree–based petri net model (modeling formalism) to solve a resource allocation problem.

This paper explores the possibility of developing an event–based modeling language for DES simulation-optimization problems using ERGs. More specifically, a restricted class of ERGs (namely Event Relationship Graph Lite, ERGL or ERG Lite) is used to *simultaneously* represent the *dynamics* of the DES together with the *constraints* and the *decision variables* of the optimization problem. This integrated ERG contains most of the information (the objective function can be implicitly included only in particular cases) needed to estimate the performance and to solve the optimization problem(s) related to the DES underlying the developed graph. This preliminary study deals with a subclass of ERGs. Because of this restriction, the class of optimization problems under investigation is confined to the selection of the optimal control policy for multi–stage queueing systems. Despite the current limitations of ERGLs in terms of expressiveness of the formalism, the class of DESs and the problems they can model is vast and relevant in practice (e.g., buffer allocation problems, maintenance policies, production control policies, etc.).

The contribution of this work is twofold. The proposed simplified ERG is proven to be capable of representing the useful information for solving the simulation–optimization problems of a controlled DES. Either a simulation or a simulation–optimization model with different levels of approximation are generated using mathematical programming. A second contribution is the clear distinction between system modeling and control modeling. This paper formally defines the system object of the simulation–optimization as a *controlled ERGL*, i.e., the union of a natural model (the DES without any control) and the control model (the control mechanism added to govern the DES). The decomposability is only possible thanks to the use of a unique formalism able to handle both structural and control information.

2 GENERAL NOTATION

The topology of the DES we consider can be represented by a queueing network with the set of servers $\mathbb{J} = \{0, \ldots, J+1\}$ and the set of possible transaction routes for job i ($i \in \mathbb{N} = \{0, \ldots, n\}$) between servers, represented by $\mathbb{Q}_i = \{(j, j') | j, j' \in \mathbb{J}\}$, $\forall i$. For each pair (j, j'), the arc connecting j and j' belongs to $\mathbb{Q}_i$ if and only if job i can *directly* flow from node j to node j'. The *source* node, represented by index $j = 0$, is the server j having no predecessors. The *sink* node is, instead, the server j having no successors and it

is indexed by $J+1$. The source node represents an infinite external arrival stream of customers, whereas the sink node is the output gate through which jobs are released from the network.

A stage includes a single server, its upstream buffer and output buffer B_j, both operating under a specified control policy. Buffers may have either finite or infinite capacity. We consider a general setting in which no explicit condition has to be imposed over the system layout, i.e., stage j is not the $j-$th on which each job is processed, but it simply represents the stage label. Analogously, job i is not the $i-$th in the sequence.

Let E_{ij}^{ξ} and e_{ij}^{ξ} denote the events occurring in the system and their occurrence times, respectively, where $\xi \in \mathbb{T}$ is the event type, and the pair (i,j) indicates the job i and the stage j the event refers to. We assume that job i at stage j undergoes a process activity with duration bounded by a start event E_{ij}^{s} occurring at time e_{ij}^{s} and a completion event E_{ij}^{f} occurring at time e_{ij}^{f}; the duration of the process is t_{ij} and, in case of stochastic DES, $\{t_{ij}\}$ may follow some known statistical distributions.
More generally, the flow of each job i is determined by the occurrence of a set of events $\mathbb{W}^i = \left\{E_{ij}^{\xi}, \xi \in \mathbb{T},\ j \in \mathbb{J}\right\}$, $i \in \mathbb{N}$. Each event E_{ij}^{ξ} in the set $\mathbb{W}^i$ has a set $\mathbb{W}_{ij}^{I\xi}$ of the input events, i.e., *triggering events*, and a set $\mathbb{W}_{ij}^{O\xi}$ of the output events, i.e., *triggered events*. Notice that elements in the sets $\mathbb{W}_{ij}^{I\xi}$ and $\mathbb{W}_{ij}^{O\xi}$ might not be in the set $\mathbb{W}^i$.

3 EVENT RELATIONSHIP GRAPH LITE

The DES previously described can be modeled through a graph whose set of nodes $\mathbb{W}$ is the set of events E_{ij}^{ξ}, whereas the set $\mathbb{E}$ of arcs represents the precedence relationships between events. According to Schruben (1983), we can define a *subclass* of ERGs in the scope of the presented research.

Definition 1 (Event Relationship Graph Lite, ERGL) An ERGL is an oriented weighted graph where the set of nodes $\mathbb{W} = \{(i,j,\xi), i \in \mathbb{N}, j \in \mathbb{J}, \xi \in \mathbb{T}\}$ contains the events E_{ij}^{ξ} occurring in the system. Each node is assigned a value equal to the time e_{ij}^{ξ} when the event occurs. Directed arcs connect different event pairs $\left(E_{ij}^{\xi}, E_{i'j'}^{\xi'}\right)$ and the set of arcs $\mathbb{E} = \{((\xi,i,j),(\xi',i',j')), i,i' \in \mathbb{N}, j,j' \in \mathbb{J}, \xi,\xi' \in \mathbb{T}\}$ represents the precedence relationships between events. Each arc can be assigned a weight $w_{\xi' i' j'}^{\xi i j}$ that can be continuous (positive or negative) or binary.

In an ERGL, each event E_{ij}^{ξ} has a set of triggering events $\mathbb{W}_{ij}^{I\xi}$ as well as a set of triggered events $\mathbb{W}_{ij}^{O\xi}$. In case of arc with continuous weight, the value of each node satisfies $e_{ij}^{\xi} \geq \max_{\xi',i',j'}\left(e_{i'j'}^{\xi'} + w_{\xi ij}^{\xi' i' j'} : E_{i'j'}^{\xi'} \in \mathbb{W}_{ij}^{I\xi}\right)$. This becomes $e_{ij}^{\xi} \geq \max_{\xi',i',j'}\left(e_{i'j'}^{\xi'} \cdot w_{\xi ij}^{\xi' i' j'} : E_{i'j'}^{\xi'} \in \mathbb{W}_{ij}^{I\xi}\right)$ in case of binary connections. Concerning the set $\mathbb{E}$, if an arc is assigned a binary weight, we interpret the arc as *active* when the associated weight is equal to 1 and *deactivated* otherwise. If an arc is deactivated, it does not establish any relationship between the connected events. Arcs with continuous weights are always active arcs.

Remark 1 ERGLs are a simpler subclass of ERGs that does not contain conditional arcs and in which the system state is implicitly defined by the event sequence. This characteristic makes ERGLs convenient to represent queuing systems.

4 DESIGNING A CONTROL SYSTEM THROUGH ERGLs

ERGs have been proven to be spectacularly effective in the simulation of DESs. In this paper, using the ERGL subclass, we extend their use to optimization. Under the ERGL representation perspective, modeling a system corresponds to create and connect events, i.e., populating the graph. The system dynamics is then a result of the relationships between events in the ERGL.

Definition 2 (Connected Events) Let e_{ij}^{ξ} and $e_{i'j'}^{\xi'}$ be the times when the events E_{ij}^{ξ} and $E_{i'j'}^{\xi'}$ occur, respectively. These two events are connected if and only if $\left(E_{i'j'}^{\xi'} \in \left(\mathbb{W}_{ij}^{I\xi} \cup \mathbb{W}_{ij}^{O\xi}\right) \bigwedge E_{ij}^{\xi} \in \left(\mathbb{W}_{i'j'}^{I\xi'} \cup \mathbb{W}_{i'j'}^{O\xi'}\right)\right)$. In particular, if $\left(E_{i'j'}^{\xi'} \in \mathbb{W}_{ij}^{I\xi} \bigwedge E_{ij}^{\xi} \in \mathbb{W}_{i'j'}^{O\xi'}\right)$, the connection establishes that event $E_{i'j'}^{\xi'}$ can trigger event E_{ij}^{ξ}. Analogously, if $\left(E_{ij}^{\xi} \in \mathbb{W}_{i'j'}^{I\xi'} \bigwedge E_{i'j'}^{\xi'} \in \mathbb{W}_{ij}^{O\xi}\right)$, event E_{ij}^{ξ} can trigger event $E_{i'j'}^{\xi'}$.

In the following sections, we investigate how to create a basic ERGL model (referred to as *natural model*) and how to extend it to include control mechanisms.

4.1 Natural Event Model

We define as *natural* a system that evolves solely according to its physical constraints. In this system, entities are supposed to flow according to their process plan. Referring to definition 1, this system can be mapped into a graph in which $\bigcup_{\xi,j}\left(\mathbb{W}_{ij}^{I\xi} \cup \mathbb{W}_{ij}^{O\xi}\right) \subseteq \mathbb{W}^i, \xi \in \mathbb{T}, j \in \mathbb{J}$, i.e., only events related to the same job are connected. This reflects the assumption that the *natural* knowledge is related to the processes each job has to undergo, and no control policy has been established yet. The arcs of the natural ERGL can be weighted only through continuous weights.
Figure 1 reports the graph for a serial multi–stage line. Only the events related to the same job are connected (i.e., event E_{ij}^{s} precedes event E_{ij}^{f} and event E_{ij}^{f} precedes event $E_{i,j+1}^{s}$). This example shows that, as long as no sequence is defined between jobs, the natural ERGL is a disconnected graph. A direct consequence of this graph feature is that the natural model cannot be used to estimate the performance of the underlying system.

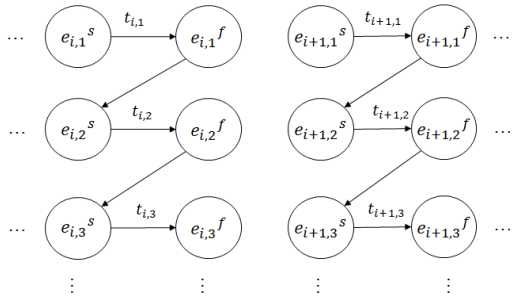


Figure 1: Single server multi–stage tandem line: the natural graph.

4.2 Control Event Model

The natural model represents a system that evolves accordingly to the process sequences assigned to each job. However, any system needs a control to enable the entities flow. Considering the general event–based model in definition 1, a *controlled ERGL* can be defined as follows.

Definition 3 (Controlled ERGL) Given a set $\mathbb{W}^N$ of natural events, a controlled ERGL is an ordered set $\mathbb{W}^{CN}$ of events that contains all the elements in $\mathbb{W}^N \subseteq \mathbb{W}$ and adds the set $\mathbb{W}^C \subseteq \mathbb{W}$ of *control* events. Elements in $\mathbb{W}^{CN}$ are connected through natural arcs ($\mathbb{E}^N \subseteq \mathbb{E}$) and control arcs ($\mathbb{E}^C \subseteq \mathbb{E}$). Control arcs are directed arcs associated to either continuous weight $s_{\xi' i' j'}^{\xi ij}$, referred to as *time buffer*, or binary weight, with $\kappa_{\xi' i' j'}^{\xi ij}$ indicating the associated binary value.

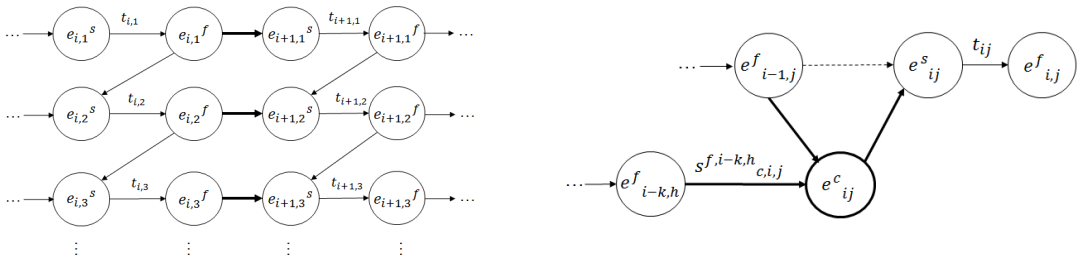
Definition 3 implies that a control policy can be represented within an ERGL only if it can be mapped into a unique event sequence. This assumption is valid for a large category of state–based policies as long as artificial control events are created to represent the system reaching an observable state. However, this assumption fails for more complex time–dependent policies that are based on measures not directly

related to a unique system state. In this case, more complex formalisms (such as the full ERGs) should be considered. GSMPs also allow to model such more complex situations. However, if GSMPs are considered, the state based modeling leads to an impractical increase in the number of nodes in the resulting graph.

It is clear that the design of a controlled ERGL requires the insertion of new active control arcs $\mathbb{E}^C$ between already existing event times and, possibly, the insertion of new events (*control events*, $\mathbb{W}^C$).

Control arc insertion The representation of a sequencing or routing control policy only requires the addition of connections between times of natural events to enable their connection according to the order established by the policy. As an example, consider the natural model in Figure 1. A sequencing rule forcing job with label i to be processed before job labeled $i+1$ at each stage j (for each i and j) will add arcs between nodes e^f_{ij} and $e^s_{i+1,j}$, as depicted in Figure 2(a).

Control nodes insertion The addition of new arcs is not sufficient when the adopted control policy does not only deal with the order of events, but further new synchronization mechanisms (conditional arcs in the full ERG) need to be added to modify the flow of entities. A typical example are blocking based control policies generated by finite capacity buffers in the line, as well as kanban tokens. In these cases, the introduction of *control* events is required. As an example, consider a multi–stage line in which a maximum capacity of c_j entities is assigned to the j–th stage. A *release* event is needed to block the occurrence of the (natural) start event of job $i+c_j$ at stage j ($e^s_{i+c_j,j}$) until the release of the job i to the stage $j+1$ has occurred, i.e., the control release event $e^r_{i,j+1}$ is responsible for triggering the control event $e^r_{i+c_j,j}$ that, eventually, triggers the natural event $e^s_{i+c_j,j}$. Under a state–based modeling perspective, this means to prevent job i to enter the queue of stage j if the queue level is equal to c_j, i.e., the buffer is full. The addition of a new control event implies adding new control arcs to connect the added node with the rest of the graph. This can also require the replacement of a natural arc with new control arcs. An additional example is depicted in Figure 2(b), where thick arrows refer to newly added control arcs, thick circles to newly added control events, and dashed arrows refer to removed natural arcs. In the natural model, the starting event time of job i at stage j (e^s_{ij}) can occur only after the finishing event time for job $i-1$ at the same stage ($e^f_{i-1,j}$). The new control mechanism breaks this connection. It is the new control event e^c_{ij}, triggered by the natural events $e^f_{i-1,j}$ and $e^f_{i-k,h}$, to trigger the event e^s_{ij}. The continuous weight $s^{f,i-k,h}_{c,i,j}$ on the control arc $((f,i-k,h),(c,i,j))$ forces a delay between $e^f_{i-k,h}$ and e^c_{ij}. As a result, according to definition 1, job i can start only at the maximum between the finish of job $i-1$ at the same stage and the finish time of job $i-k$ at stage h plus the interval $s^{f,i-k,h}_{c,i,j}$.



(a) Single server multi–stage tandem line: sequenced jobs.

(b) Example of replacing natural mechanisms.

Figure 2: Adding a control mechanism to the ERGL.

The complexity of the proposed ERGL is strongly related to the type of connections that need to be defined between event times. In general, the growth of the number of nodes in the model is linear in the number of servers, jobs and policies. The same does not hold for the arcs, whose growth is linear only in the number of servers and considered policies, while it is polynomial with order larger than 1 in the number of jobs.

The structure of the ERGL can be used as a means to determine whether a specific control policy might generate deadlocks in the system, or simply result ineffective as it does not modify the entity flow deriving from the natural graph. These two aspects are shortly investigated in the remainder of the section.

Deadlock detection In a system modeled through ERGL, a deadlock can occur when a cyclic relationship between nodes exists (necessary condition). Hence, checking if a control mechanism may lead to a deadlock means to check if the insertion of new arcs and/or new nodes (and related arcs) creates a cycle in the previously acyclic graph.

Redundancy detection We define a control mechanism as *redundant* when the resulting ERGL is characterized by duplicated connections between the same event pairs. In such cases, the added control does not change the entities flow and it is then ineffective in controlling the system dynamics.

Infeasibility and redundancy can be efficiently verified using the concept of *transitive closure* (van Leeuwen 1990, Aho, Garey, and Ullman 1972, Habib, Morvan, and Rampon 1993) guaranteeing the detectability of deadlocks and redundancies also in complex cases.

The obtained ERGL considers both the natural and controlled dynamics of the system. The natural dynamics depends on the defined process sequences, whereas the controlled dynamics depends on the policies introduced to govern the flow of entities. Different control policies lead to different ERGLs structures, i.e., different control event set $\mathbb{W}^C$ and control arc sets $\mathbb{E}^C$ and different weights $\mathbf{s}$ and/or κ. The indisputable advantage of the proposed approach is that an ERGL can be considered as an optimization model when $\mathbb{W}^C$, $\mathbb{E}^C$ and associated weights $\mathbf{s}$ and κ are interpreted as sets of decision variables. The next section investigates this aspect.

5 OPTIMIZING A CONTROLLED ERGL

The optimization of a control system can be performed, according to the classical simulation–optimization approach, through a search module generating the candidate values θ for the parameters of the control policy to be optimized, and using simulation as a black–box tool for estimating an implicit function l. Using the mathematical programming formalism, we can formulate the optimization problem P as:

$$P : \min_{\theta \in \Theta} \quad l(\theta) \tag{1}$$

where θ is the vector of decision variables and Θ defines the constraint set on θ (Fu, Glover, and April 2005). If stochastic systems are considered, function $l(\theta)$ can be interpreted as an expectation, i.e., $l = E[L(\theta, \Omega)]$, where Ω represents a probability space governing the stochastic processes that characterize the system (e.g., inter arrival times, processing times). Furthermore, the problem could be constrained to satisfy a predefined level of some performance measures (e.g., satisfaction of the service levels, throughput targets).

The approach we propose is not in contrast with this framework: the solution generated by the search procedure θ can be mapped onto control events and arcs (and related weights) following the rules discussed in section 4. The obtained ERGL has all the information for estimating function l, i.e., it behaves as a pure simulation model and, as such, it can be used to obtain an estimate of the system performance (Savage, Schruben, and Yücesan 2005).

However, the advantage of the presented modeling approach relies in the possibility to interpret control nodes, arcs and weights as decision variables in an optimization problem instead of fixed input parameters to be received from an external search procedure. This interpretation exploits the completeness of the ERG formalism and makes the ERGL a simulation–optimization modeling language.
Under this new perspective, the optimization of a control policy corresponds to the search of the best set of control events $\mathbb{W}^C = \left\{E_{ij}^{\xi}\right\}$ and related occurrence times $\left\{e_{ij}^{\xi}\right\}$, as well as the set of arcs (and related weights) such that the ERGL has the best associated value for the selected objective function.

Such a simulation–optimization model can be "solved" in many ways based on the characteristics of the objective function (1) (Chan and Schruben 2008). In this paper, we propose the use of mathematical

programming. In particular, we introduce the way to, automatically, map ERGLs in a set of equations deriving the integrated mathematical programming model for simulation-optimization.

5.1 Objective functions

When all the control nodes, arcs and related weights are established, the optimization problem related to the ERGL can be simply brought back to a simulation problem and the objective function is that proposed in Chan and Schruben (2008):

$$\min \quad \sum_{v \in \mathbb{W}} e_v \tag{2}$$

The objective function (2), together with the constraints generated from the ERGL (see section 5.2), defines a mathematical programming representation of the *simulation model* of the system. Advantages of this formulation have been investigated in Chan and Schruben (2008) and Matta (2008).

The problem is more challenging when either weights, arcs or nodes have to be determined. In particular, two main cases can be distinguished: (1) optimization of a given control policy, (2) identification and optimization of the best control policy. In both cases, the event times are variables to be optimized as in the simulation case.

Optimization of a given control policy If all nodes and arcs have already been established, the problem can be brought back to the choice of the set of optimal arc weights (the control policy parameters). A general objective function can be defined as follows:

$$\min \quad f(\mathbf{e}) + h(\mathbf{s}) + g(\kappa) \tag{3}$$

where f, h and g are real functions of the event times $\mathbf{e}$, the time buffers $\mathbf{s}$ (if present), and the boolean activations κ (if defined in the ERGL), all characterized according to definition 3. Depending on the type of considered events, several objective functions can be defined. In case κ is defined, the problem is a MILP. In the simple case in which f, g and h are summations we obtain:

$$\min \quad \sum_{v \in \mathbb{W}} \alpha_v e_v + \sum_{v \in \mathbb{E}^C} (\beta_v s_v + \gamma_v \kappa_v), \tag{4}$$

where α_v, β_v and γ_v are known function coefficients. Function (2) is a special case of (4) for $\beta_v, \gamma_v = 0$, $\forall v$, and corresponds to minimizing a function that depends only on the event times. The case $\alpha_v = 0$, $\forall v$, corresponds, instead, to the minimization of a function of the control parameters $(\mathbf{s}, \kappa)$.

Identification and optimization of the best control policy If the design of the ERGL is not given, the optimal control mechanisms $\mathbb{W}^C$ have to be selected together with the control parameters $\mathbf{s}$ and/or κ making the problem more complex. Similarly to the previous case, we can use the general objective function in equation (3). The main difference with the previous case relies in the need to add the *activation* binary decision variables which take value 1 if the event e is added, the arc with continuous weight s is chosen or the binary connection with weight κ is included in the ERGL. It is clear the need of these new elements to *create* the ERGL: in the previous model the same elements were implicitly contained within the sets defining the graph structure. We can notice that this problem is *always* a MILP thus making its solution more challenging.

The control of DES performance can also be included in the form of stochastic constraints forcing the system to achieve a configuration that meets a predefined target. An expected performance is a function of event times $\mathbf{e}$ and control variables $\mathbf{s}$ and κ. Examples of stochastic constraints are the expected value of customers waiting in queue forced to be lower than a threshold, the expected system throughput in a shop floor or the expected service level in a serial supply chain forced to be greater than a predefined value. Being concerned with optimization, stochastic constraints are not treated in Chan and Schruben (2008), Chan (2005). Performance constraints are modeled by introducing the following relationship:

$$\sum_{v \in \mathbb{W}^C} p(e_v) \leq \vartheta^*,$$

where ϑ^* is the target performance and p is any function of the control event times.

5.2 From ERGL to mathematical programming constraints

The constraints characterizing the optimization model can be partitioned into two categories: *linear dynamics constraints* and *control constraints*.

Linear dynamics constraints. These constraints involve only continuous variables and map the relationships established by the natural arcs in the control model. An example of this type of constraints is the following:

$$e^f_{ij} \geq e^s_{ij} + t_{ij},$$

stating that customer i cannot leave stage j (e^f_{ij}) before accessing the server (e^s_{ij}) and completing the service (t_{ij}). The same relationships were proposed within the LP formulations by Chan and Schruben (2008). The right hand side is a vector of values that usually are realizations of random variables and represents the weight of the arc connecting the nodes e^s_{ij} and e^f_{ij} in the example.
A procedure to automatically generate the nodes and arcs referring to linear dynamics constraint in the ERGL was proposed in Chan and Schruben (2008), Chan (2005). Thus, we refer to these works for translating the natural model.

Control constraints. Control constraints relate the occurrence times of events in presence of control variables and derive from the translation of control arcs and weights (i.e., control parameters). Control parameters can be either discrete or continuous and, when discrete, their translation leads to non linear constraints. Also for this type of constraints, a general procedure to translate the graph into the mathematical programming model has been proposed in Chan and Schruben (2008), Chan (2005). However, their models only refer to simulation, hence control arcs are treated as input parameters instead of decision variables. An example of this type of constraints is the following:

$$e^{\xi'}_{i'j'} \geq e^{\xi}_{ij} - q\left(w^{\xi ij}_{\xi' i' j'}\right), \tag{5}$$

where $e^{\xi}_{i,j}$ and $e^{\xi'}_{i'j'}$ are the time occurrence of two events relating job i on stage j and job i' on stage j' that are linked by control $q(w^{\xi ij}_{\xi' i' j'})$. If the relationship between the two event times is boolean, function q has the form $(1-\kappa^{\xi ij}_{\xi' i' j'})\cdot M$, where $\kappa^{\xi ij}_{\xi' i' j'}$ is a binary decision variable and M is a large number. Instead, in case of continuous relationship, q is a function of the continuous variable $s^{\xi ij}_{\xi' i' j'}$ time buffer.

6 APPLICATION: KANBAN CONTROL SYSTEM (KCS)

A three–stage queueing network managed by a kanban policy is represented in Figure 3. Each stage j

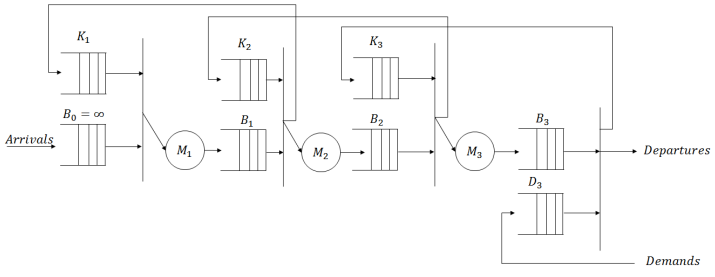


Figure 3: Three–stage queueing network with Kanban Control System.

($j = 1,\ldots,3$) is composed of a server (represented by a circle) with an incoming infinite capacity buffer

and a synchronization station consisting of two queues: K_{j+1} containing the kanban tokens of stage $j+1$, and B_j containing the finished parts from stage j. At the last stage, D_3 contains the external demands (Liberopoulos and Dallery 2000). A fixed and discrete number of kanban tokens $K_j \in \mathbb{K}_j = \{K_j^L, \ldots, K_j^U\}$ is associated to stage j.

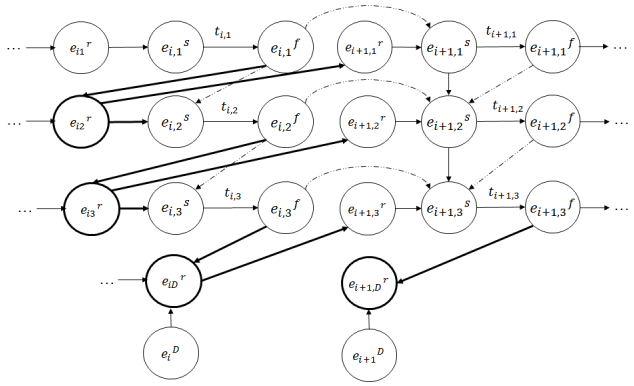


Figure 4: Kanban control system: event graph.

The *natural* ERGL related to the KCS is a multi–stage tandem line with fixed job sequence and infinite buffer capacities like the one depicted in Figure 2(a). Specifically, the nodes are: $e_{ij}^s, \forall i, j$, the time when the *start* event E_{ij}^s occurs, and $e_{ij}^f, \forall i, j$, the time of the *finish* (departure) event E_{ij}^f. Each node e_{ij}^s is connected to a node e_{ij}^f through a *natural delay arc* whose weight is the service time of job i at stage j, t_{ij}. Event E_{ij}^f can trigger the start event $E_{i+1,j}^s$, if there are queueing jobs. As a result, nodes $e_{i+1,j}^s$ and e_{ij}^f are connected. Assuming a known process sequence, job i starts being processed on stage $j+1$ once the activities at server j are completed (dashed arrow connecting e_{ij}^f and $e_{i,j+1}^s$, Figure 4). When KCS is considered, these dashed connections need to be "broken down" to be replaced by control connections and a new set of control events $\left\{e_{ij}^r\right\}$ model the *release* of jobs $i = 1, \ldots, n$ from stage $j-1$ to stage j. This new control event breaks the aforementioned connection replacing it with two control arcs $((f,i,j),(r,i,j+1))$ and $((r,i,j+1),(s,i,j+1))$. The resulting graph has the control nodes and arcs with thick borders in Figure 4 and, as a result of the control model, each customer, after being processed by stage j, can be released to the next stage only if a free kanban is available. As an example, if a single kanban token is assigned to stage j, customer $i+1$ can be released to stage j only if customer i has already been released to the next stage. Nodes $e_{i,j+1}^r$ and $e_{i+1,j}^r$ are then connected through a control arc whose "weight" is the binary variable $\kappa_{r,i+1,j}^{r,i,j+1} = 1$. Finally, the control nodes $\left\{e_i^D\right\}$ represent the demand signal constraining each job not to leave the system before the related demand has occurred. The described connections (arcs) can be mapped, as described in section 5, to the following constraints:

$$e_{ij}^f - e_{ij}^s \geq t_{ij} \qquad \forall i, j \tag{6}$$

$$e_{i+1,j}^s - e_{ij}^f \geq 0 \qquad \forall i, j \tag{7}$$

$$e_{i,j+1}^r - e_{ij}^f \geq 0 \qquad \forall i, j \tag{8}$$

$$e_{ij}^s - e_{ij}^r \geq 0 \qquad \forall i, j \tag{9}$$

$$e_{i,J+1}^r \geq e_i^D \qquad \forall i \tag{10}$$

$$e_{i+k,j}^r - e_{i,j+1}^r \geq \left(1 - \kappa_{r,i+k,j}^{r,i,j+1}\right) \cdot M \qquad \forall j,\ k \in \mathbb{K}_j,\ i = 1, \ldots, n-k \tag{11}$$

$$e_{i+k,j}^r - e_{i,j+1}^r \geq -s_{r,i+k,j}^{r,i,j+1} \qquad \forall j,\ k \in \mathbb{K}_j,\ i = 1, \ldots, n-k \tag{12}$$

To optimize the described system, we considered the following objective functions:

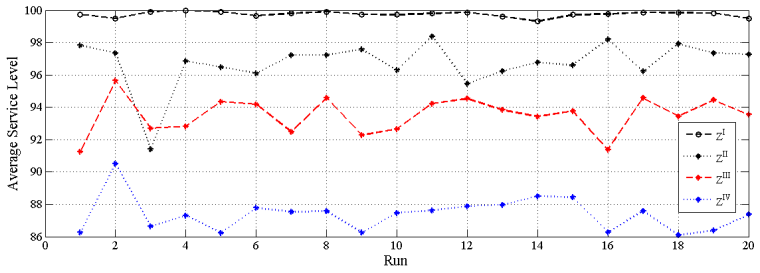
$$Z^I = \min \sum_{j\in\mathbb{J}} \sum_{i\in\mathbb{N}} \left(e^s_{ij} + e^f_{ij} + e^r_{ij} \right) \qquad Z^{II} = \min_{\kappa} \sum_{j\in\mathbb{J}} \sum_{i\in\mathbb{N}} \sum_{k\in\mathbb{K}_j} \kappa^{r,i,j+1}_{r,i+k,j} \cdot k$$

$$Z^{III} = \min_{s} \sum_{j\in\mathbb{J}} \sum_{i\in\mathbb{N}} \sum_{k\in\mathbb{K}_j} s^{r,i,j+1}_{r,i+k,j} \qquad Z^{IV} = \min \sum_{j\in\mathbb{J}} \sum_{i\in\mathbb{N}} \sum_{k\in\mathbb{K}_j} \alpha_{ijr} \left(e^r_{i,j+1} - e^r_{i,j} \right)$$

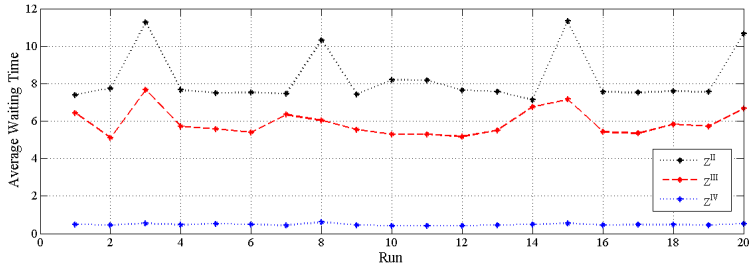
Function Z^I minimizes the sum of all the event times without constraining the number of kanban tokens (the time buffer). Functions Z^{II} and Z^{III} minimize the total number of kanban tokens and the amount of time buffer allocated to each stage of the line, respectively. Function Z^{IV} minimizes the cost of the waiting time at each stage. Since we consider identical jobs, α_{ijr} depends only on stage and event type, i.e., $\alpha_{ijr} = \alpha_{jr}$, $\forall i$. We added the following constraint to control the lateness of the jobs:

$$\frac{1}{n} \sum_{i\in\mathbb{N}} \left(e^r_{i,J+1} - e^D_i \right) \leq \vartheta^* \tag{13}$$

Furthermore, we set the event times related to the arrival of the demand for job i at the times t^D_i, which is the realization of a random variable representing the demand process, i.e. $e^D_i = t^D_i, \quad \forall i$.



(a) Average service level.



(b) Average waiting time.

Figure 5: Objective functions: the comparison through the "benchmark."

Processing times were assumed lognormal and identical for each stage ($\mu = 2.73$, $\sigma = 0.274$) as demand inter arrival times ($\mu = 3.4$, $\sigma = 1.624$). The target performance was set to $\vartheta = 0.1$ corresponding to a 91% *service level* (i.e., the ratio between the demand on time and the total demand). Figures 5(a)–5(b) report the value for the service level and the average total waiting time obtained over 20 independent simulation–optimization replications (x-axis corresponds to the replication), respectively. It is clear how a "dominant" objective function exists with respect to both measures. In Figure 5(a), the system with the highest service level is obtained from objective function Z^I. This objective function has no practical use in terms of policy choice since its implementation would require the control of the time of *all* the events,

which is impossible in a stochastic setting. Nonetheless, such a "policy" provides an upper bound on the service level performance, hence constituting the *benchmark* on this measure. Another useful benchmark is given by function Z^{IV} that provides quite a good lower bound on the service level. This is expected since the minimization of the waiting times (and inventory costs consequently) is negatively correlated to the service level. The two systems having minimum time buffer and minimum number of kanbans are in between, with kanban (i.e., Z^{II}) performing better.

Figure 5(b) reports the results on the average system waiting time (the process time t_{ij} is not included in the statistic). Also in this case, we can identify a benchmark policy, i.e., Z^{IV} which is also the worst policy in terms of service level. Moreover, the stability of the response over different sample paths suggests Z^{IV} is a good estimate of the minimum expected waiting time in the system. However, as policy Z^{I}, Z^{IV} cannot be implemented in practice since it requires to control the waiting time between each job pair, implying perfect knowledge of the, stochastic, service times. Also in this case, the time buffer and the kanban control policies are in between. Another interesting aspect can be noticed: from Figure 5(b), the kanban policy appears to generate an unstable signal with picks in some replications. This is due to the fact that the number of needed kanban tokens can only change through discrete steps. For particular realizations of the stochastic variables, the number of kanban tokens has to rise from K to $K+1$, resulting in a jump in the system performance. This phenomenon is mitigated when the time buffer policy is in place as the continuous time buffer can react with small changes to the external conditions avoiding picks in the response.

7 CONCLUSIONS

A new modeling approach has been presented that relies on a graph based system representation. We define a subclass of ERG (ERG Lite), downsizing the level of complexity of the category of systems we can represent. This enables the integration of the description of the system dynamics and the optimization of the control policy that governs the system. Each controlled ERG Lite is meant to be the union of a natural model and a control model, enhancing the modularity and flexibility of the proposed language. A mapping procedure is proposed to transform the ERG Lite model into the mathematical programming counterpart.

The advantage of this approach relies in the possibility to interpret arcs and nodes in the ERG Lite as decision variables, thus integrating the simulation and the optimization in the same model. Hence, solving the simulation–optimization model can be interpreted as identifying the "best" ERG Lite with respect to some specified objective. The presented methodology is applied to a multi–stage kanban controlled system to show its effectiveness in generating and solving different simulation–optimization problems.

REFERENCES

Aho, A., M. Garey, and J. Ullman. 1972. "The Transitive Reduction of a Directed Graph". *SIAM Journal on Computing* 1 (2): 131–137.

Alfieri, A., and A. Matta. 2012. "Mathematical Programming Formulations for Approximate Simulation of Multistage Production Systems". *European Journal of Operational Research* 219 (3): 773–783.

Buss, A. H., and P. J. Sanchez. 2002. "Modeling Very Large Scale Systems: Building Complex Models with LEGOs (Listener Event Graph Objects)". In *Proceedings of the 2002 Winter Simulation Conference*, edited by E. Y ucesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 732–737. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Cao, X.-R., and J. Zhang. 2008. "Event-Based Optimization of Markov Systems". *IEEE Transactions on Automatic Control* 53 (4): 1076–1082.

Cassandras, C. G., and S. Lafortune. 2008. *Introduction to discrete event systems*. Springer.

Chan, W., and L. Schruben. 2008. "Optimization models of Discrete–Event System Dynamics". *Operations Research* 56 (5): 1218–1237.

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

Fu, M. 2002. "Optimization for Simulation: Theory vs. Practice". *Journal on Computing* 14 (3): 192–215.

Fu, M., F. Glover, and J. April. 2005. "Simulation optimization: a review, new developments, and applications". In *Proceedings of the 2005 Winter Simulation Conference*, edited by M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 83–95: Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Habib, M., M. Morvan, and J.-X. Rampon. 1993. "On the calculation of transitive reduction—closure of orders". *Discrete Mathematics* 111 (1-3): 289–303.

Iglehart, D. L., and G. S. Shedler. 1983. "Simulation of non–Markovian systems". *IBM Journal of Research and Development* 27 (5): 472 – 479.

Latorre, J.-I., and E. Jiménez. 2013. "Simulation-based optimization of discrete event systems with alternative structural configurations using distributed computation and the Petri net paradigm". *SIMULATION* 89 (11): 1310–1334.

Liberopoulos, G., and Y. Dallery. 2000. "A unified framework for pull control mechanisms in multi–stage manufacturing systems". *Annals of Operations Research* 93 (1–4): 325–355.

Liu, Y., H. Zhang, C. Li, and R. J. Jiao. 2012. "Workflow simulation for operational decision support using event graph through process mining". *Decision Support Systems* 52 (3): 685 – 697.

Matta, A. 2008. "Simulation Optimization with Mathematical Programming Representation Of Discrete Event Systems". In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Monch, O. Rose, T. Jefferson, and J. W. Fowler, 1393–1400. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Savage, E. L., L. W. Schruben, and E. Yücesan. 2005. "On the Generality of Event-Graph Models". *INFORMS Journal on Computing* 17 (1): 3–9.

Schruben, L. 2010. "Simulation modeling for analysis". *ACM Transactions on Modeling and Computer Simulation* 20 (1): Article 2.

Schruben, L. 2013. "Simulation modeling, experimenting, analysis, and implementation". In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, 678–690. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Schruben, L. W. 1983. "Simulation modeling with event graphs". *Communications of the ACM* 26 (11): 957 – 963.

van Leeuwen, J. 1990. "Graph algorithms". In *Handbook of Theoretical Computer Science: Algorithms and Complexity*, edited by J. van Leeuwen, Volume A, 525–631. MIP Press.

Zeigler, B. P. 1976. *Theory of Modeling and Simulation*. Wiley.

AUTHOR BIOGRAPHIES

ANDREA MATTA is Distinguished Professor at the Department of Industrial Engineering & Management at Shanghai Jiao Tong University. He currently teaches stochastic models and simulation. His research area includes analysis and design of manufacturing and healthcare systems. His email address is matta@sjtu.edu.cn.

GIULIA PEDRIELLI is Research Fellow for the Centre for Maritime Studies at the National University of Singapore. Her research focuses on simulation-optimization based on math programming and budget allocation techniques applied to maritime systems. Her email address is cmsgp@nus.edu.sg.

ARIANNA ALFIERI is Associate Professor at Politecnico di Torino, where she currently teaches production planning and control and system simulation. Her research area includes scheduling and planning in production and transportation systems. Her email address is arianna.alfieri@polito.it.