Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

GENERATING SCHEDULING CONSTRAINTS FOR DISCRETE EVENT DYNAMIC SYSTEMS

Wai Kin Chan
Lee W. Schruben

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

ABSTRACT

In most scheduling literature, constraints are seemingly generated in an ad-hoc manner using intuitive arguments. This could result in overlooking some constraints or including unnecessary constraints. Schruben (2000) has shown how the dynamics of some discrete event systems can be modeled as the solutions of optimization programs. In this paper, we use this idea to generate mathematical programming models systematically for scheduling resources in discrete event dynamic systems. Two examples are presented: a multiple server queue and a semiconductor manufacturing cluster tool. An interesting result was that the mathematical structure of the scheduling program generated from a simulation of a cluster tool found in the literature leads to a different, more concise and illuminating cluster tool simulation model that would have been difficult to discover otherwise. The corresponding optimal scheduling problem is surprising in that *it does not include explicit representation of the resource that is actually being scheduled!*

1 INTRODUCTION

Scheduling plays an important role in many decision making contexts. Important scheduling optimization problems are notoriously intractable. In fact, many of them are NP-hard. During the last few decades, heuristics algorithms have played an important role in the solution of scheduling problems. While not directly solving explicit optimal resource scheduling programs, these heuristics often result from an analysis of their explicit formulations. This paper is concerned with formulating such problems for discrete event dynamic systems.

Specifically, we consider two typical classes of scheduling problems: (i) parallel resource scheduling problems with the objective of minimizing the makespan and (ii) cluster tools scheduling problems with the objective of maximizing the throughput. A parallel resource scheduling problem with the objective of minimizing the makespan is a problem of allocating *N* jobs to be processed in *m* identical resources such that the time needed to complete all *N* jobs is minimized. This problem is of interest because minimizing the makespan balances the workload over all recourses.

A cluster tool is an integrated, environmentally isolated manufacturing system consisting of processing chambers, internal robots (transport modules) to transport jobs (e.g., wafers), and load locks where the jobs enter and leave the system (see Figure 1). Each of entering wafers is transported from the load lock to one (or more) chamber(s) for processing according to predefined routing sequences. After all processing steps, the wafers are returned to the load lock and leave the system. There are many types of routing sequences, for example, sequential processing and parallel processing (see, e.g., Srinivasan 1998 for details). In this paper, for the ease of exposition, we shall assume sequential processing.

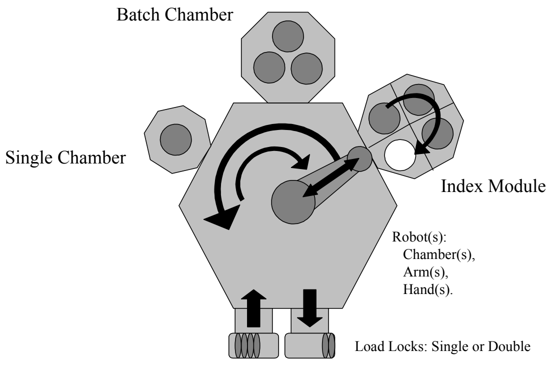

Figure 1: A Typical Cluster Tool

The analysis of cluster tools, given a processing sequence, usually involves answering one (or both) of the following two questions: (i) what is the tool throughput? and (ii) what wafer move sequence maximizes the throughput? The tool throughput, defined as the number of wafers produced per unit time, is an important performance measure because higher throughput usually implies higher revenue.

For the first question, there are two popular methods for evaluating throughput, static spreadsheet analysis and dynamic discrete event simulations. Simulation models are typically much more accurate than spreadsheet models (see Pederson and Trout 2002).

Perkinson et al. (1994) and Venkatesh et al. (1997) provided analytical models that predict the relationship between steady-state throughput and tool parameters, such as process times and transport times. Simulation models based on Petri Nets can be found in Srinivasan (1998) and Zuberek (2001). However, Petri Net models of cluster tools, especially for multi-cluster tools, are much more cumbersome than models developed using simulation event relationship graphs (Schruben 1983). Using event graphs, very complex cluster tools can often be modeled using only three events. The simplicity of event graph simulations not only dramatically reduces execution times and makes logical errors less likely, it also facilitates the analysis of the system performance as well as the design of algorithms for finding optimal scheduling sequences (see, e.g., Nehme and Pierce 1994, Pederson and Trout 2002). The benefits of the event graph approach over the Petri Nets approach become more significant when one tries to analyze multi-cluster (nested) tools. Ding and Yi (2004) proposed a three-event event graph simulation model for multi-cluster tools. By integrating their simulation model into their tree search algorithm, they efficiently found the optimal scheduling sequence for a complex, integrated system of cluster tools.

For the second question, different approaches have been proposed in the literature. Rostami and Hamidzadeh (2002) presented an heuristic algorithm for finding the optimal scheduling sequence for cluster tools with residency constraints. In each permutation of their algorithm, they create and solve a linear programming (LP) problem.

We note that problems of scheduling cluster tools with residency constraints are mathematically similar to the hoist scheduling problems. Phillips and Unger (1976) described a mixed integer programming model for scheduling such systems. Subsequently, Shapiro and Nuttle (1988) and Chen, Chu, and Proth (1998) solved scheduling problems using a branch and bound approach, in which solutions from linear programming sub-problems were used to reduce the number of branching nodes as well as provide a lower bound of the cycle time.

Constraints in above mathematical programming models are apparently generated in a rather ad-hoc manner using intuitive arguments. For complicated systems, this could lead to the problems of overlooking necessary constraints or including redundant or unnecessary constraints in the model formulation. Moreover, if structural characteristics of the system change, the optimization model must also be altered to reflect the changes. In this paper, we propose a *systematic* approach for formulating mathematical programming models for scheduling problems. Specifically, we first model of the system dynamics as an event graph that incorporates the processing sequence of the jobs. This can be used to simulate the system for analysis. We can also generate the constraints for optimal resource scheduling from the event graph. The mathematical programming formulations provide an analytical approach to answering the *how* and *what-if* questions regarding the performance of a system.

Another benefit from representing discrete event system dynamics as mathematical programs is illustrated in the cluster tool example presented in this paper. The mathematical characteristics of the optimization model for a cluster tool simulator found in the literature guided us to a new, more concise and illuminating, cluster tool simulation model that would have been difficult to formulate without the help of the optimization model.

The reminder of this paper is organized as follows: In Section 2, a general framework of generating mathematical programming formulations for discrete event system dynamics is described. In Section 3, we illustrate mapping an event graph into a linear program (LP) using an example of mapping a $G/G/1$ queue. Section 4 considers parallel resource scheduling problems. We derive LP formulations for cluster tool scheduling problems in Section 5. Section 6 concludes the paper.

2 GENERAL FORMULATION

Schruben (2000) proposed representing the dynamics of discrete event systems as optimization programs, where the solutions are the system trajectories. The discrete event system is first modeled as a simulation event relationship graph, or event graph (EG). Event graphs are a simple, completely general, graphical representation of discrete event dynamics. EGs are a system of state changes (often simple difference equations) analogous to systems of differential equations used to represent continuous system dynamics. Then the constraints for an optimization model of the system are derived from the timed edges of the EG along with feasibility conditions for the state variables.

Certain Petri Nets can also be modeled as mathematical programs (Yen 1999). However, event graph modeling (Schruben and Schruben 2000) is more general than Petri Net modeling since all Petri Net models can be transformed into EGs, but the reverse does not hold (see Schruben 2003). Our derivation will be based on the EG representation. It has been shown that the system dynamics of many EG queueing models can be represented as an LP, e.g., Chan and Schruben (2003) provides the LP formulations for single-server tandem queues and Chan and Schruben (2004) describes the LP formulations for multi-server tandem queues.

Section 2.1 introduces a framework of generating constraints with integer or binary variables. The methodology of generating constraints without integer or binary variables is described in Section 2.2.

2.1 Generating Constraints with Integer or Binary Variables

The variables in the optimization model are the event times. The time of the *i*-th occurrence of a generic event *E* is denoted as E_i. The objective function is simply to execute all system events as soon as possible, subject to constraints imposed in the system event graph.

To model the constraints on the event times, consider the generic edge between events *E* and *F* in Figure 2.

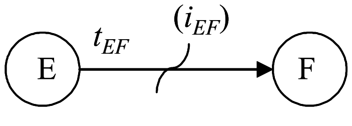

Figure 2: A Generic Edge of an Event Graph

This edge indicates that event *E* will schedule event *F* after a delay of t_{EF}, provided the condition i_{EF} is true. The truth of the edge condition, i_{EF}, will be tested with the binary variable $\gamma_{E,j,F}$ (discussed later):

$\gamma_{E,j,F} = 1$ iff condition i_{EF} is true immediately after the *j*-th execution of event *E* (at E_j).

If condition i_{EF} is true at time E_j, then one and only one occurrence of event *F* (say, the *k*-th) will be scheduled. This is controlled by another binary variable,

$\delta_{E,j,F,k} = 1$ iff the *j*-th execution of event *E* (at time, E_j) schedules the *k*-th execution of event *F* (for time F_k).

If i_{EF} is true, then one and only one occurrence of event *F* will be scheduled at time F_k. Let $t_{EF}(j)$ be the *j*-th value of the edge delay time stochastic process. If M_1 is a number at least as large as the duration of the simulation, this edge imposes five constraints on the system dynamics,

$$F_k \le E_j + t_{EF}(j) + M_1(1-\delta_{E,j,F,k}) \tag{2.1}$$
$$F_k \ge E_j + t_{EF}(j) - M_1(1-\delta_{E,j,F,k}) \tag{2.2}$$
$$\sum\nolimits_j \delta_{E,j,F,k} \le 1$$
$$\sum\nolimits_k \delta_{E,j,F,k} \le 1$$

and

$$\sum\nolimits_k \delta_{E,j,F,k} \ge \gamma_{E,j,F}\,.$$

This simply states that, if condition i_{EF} is true when event *E* occurs at time E_j, then one and only one event *F* will be scheduled at time $E_j + t_{EF}(j)$. Note that changing inequalities (2.1) and (2.2) to the following inequalities gives us a tighter (stronger) formulation.

$$F_k \le E_j + t_{EF}(j) + M_1(1-\sum\nolimits_{l\ge k}\delta_{E,j,F,k}) \tag{2.1$'$}$$
$$F_k \ge E_j + t_{EF}(j) - M_1(1-\sum\nolimits_{l\le k}\delta_{E,j,F,k})\,. \tag{2.2$'$}$$

To test the truth of edge conditions, a counting process is used. The number of times that event *E* has occurred by time *t* is given by the right-continuous event counting point process:

$$C_E(t) = \lim_{\varepsilon\to 0}\max\{j : E_j \le t+\varepsilon\}\,.$$

To count the number of times event *E* has occurred by the *k*-th occurrence of event *F* (at F_k), we will use another binary variable, σ. For a suitably large value of M_2, define

$$\sigma_{E,j,F,k} \ge \frac{F_k - E_j}{M_2} \text{ and } \sigma_{E,j,F,k} \in \{0,1\} \tag{2.3}$$

and

$$C_E(F_k) = \sum\nolimits_j \sigma_{E,j,F,k}\,.$$

Note that if simultaneous events are considered, e.g., $F_k = E_j$, then constraint (2.3) should be changed from '≥' to '>'.

Denote by $X(t)$ the value of a generic state variable at time *t* with X_{0^-} as its initial value. Let I_X denote the set of events that *increment* state variable, $X(t)$, by the integer amounts, $u_i^{\prime}$, and D_X denote the set of events that *decrement* $X(t)$ by amounts $d_i^{\prime}$. Then $X(E_j)$ is simply the amount $X(t)$ has increased minus the amount it has decreased right after the *j*-th occurrence of event *E*, or,

$$X(E_j) = X_{0^-} + \sum\nolimits_{e\in I_X} u_i^{\prime} C_e(E_j) - \sum\nolimits_{e\in D_X} d_i^{\prime} C_e(E_j)\,. \tag{2.4}$$

If i_{EF} is the condition like $(X > B)$, then, for a suitably large value of M_3,

$$\gamma_{E,j,F} \ge \frac{X(E_j)-B}{M_3} \text{ and } \gamma_{E,j,F} \in \{0,1\}\,.$$

The sums of the binary variables $\gamma_{E,j,F}$, $\delta_{E,j,F,k}$, and $\sigma_{E,j,F,k}$ are added to the objective function to be minimized with suitable weights. To the set of constraints imposed by the edge E-F, we add the definitions of $C_E(F_k)$ and the associated constraints. We will denote a set of constraints like these as

$$F = E + t_{EF} \text{ iff } i_{EF}.$$

2.2 Generating Constraints without Integer or Binary Variables

Most of the aforementioned constraints involve integer (or binary) variables and therefore the resulting mathematical programming formulations are mixed integer programming formulations. Sometimes, however, we might want to represent a discrete event system as an LP, for example, for duality sensitivity analysis. Therefore, we provide a method of transforming equation (2.4) into a set of linear constraints. Since equation (2.4) consists of more than two events that contribute in changing the value of $X(t)$, the derivation will be based on ‘convolutions’ of these events. For the ease of exposition, we shall illustrate the derivation using an example of six events (i.e., $I_X = \{a, b, c\}$ and $D_X = \{d, e, f\}$, see inequality (2.5)). We will extend the result to general cases after presenting this example. First, we note that if all simultaneous events are handled correctly, then without loss of generality, we can assume $u_i' = d_i' = 1$ for all i. For example, if one event increments $X(t)$ by 2 units (i.e., $u_i' = 2$), then it can be replaced by two events, each increments $X(t)$ by 1 unit. Let A_X be the initial value of $X(t)$. The feasibility condition of $X(t)$ says

$$\begin{aligned} X(t) &= C_a(t) + C_b(t) + C_c(t) \\ &\quad - C_d(t) - C_e(t) - C_f(t) + A_X \geq 0 \qquad \forall t \geq 0. \quad (2.5) \end{aligned}$$

Let $U(t)$ be a “super” event representing the event that either of event times $a(t)$, $b(t)$, or $c(t)$ has occurred by time t. Similarly, let $V(t)$ be a super event representing the event of either $d(t)$, $e(t)$, or $f(t)$ having occurred by time t. The feasibility condition of $X(t)$ can now be re-written in terms of the event counting processes as

$$X(t) = C_U(t) - C_V(t) + A_X \geq 0,$$

which is equivalent to the event times being constrained by

$$U_{i-A_X} \leq V_i \qquad , i = A_X + 1, \ldots, N. \qquad (2.6)$$

To represent the relationships between U_i and a_i, b_i, and c_i (and the relationships between V_i and d_i, e_i, and f_i), notice that there are $(i+1)(i+2)/2$ possible permutations that could cause U_i (V_i) to happen. This is depicted in Figure 3. Let $\mathrm{A}(i)$ be the set of all possible permutations and α_{i,i_1,i_2,i_3} be one of these permutations. Therefore, for each $i = 1,\ldots,N$, we have:

$$\begin{array}{lll} \alpha_{i,i_1,i_2,i_3} & \geq \alpha_{i-1,i_1-1,i_2,i_3} & \\ \alpha_{i,i_1,i_2,i_3} & \geq \alpha_{i-1,i_1,i_2-1,i_3} & \\ \alpha_{i,i_1,i_2,i_3} & \geq \alpha_{i-1,i_1,i_2,i_3-1} & \\ U_i & \leq \alpha_{i,i_1,i_2,i_3} & , \forall (i_1,i_2,i_3) \in \mathrm{A}(i). \end{array} \qquad (2.7)$$

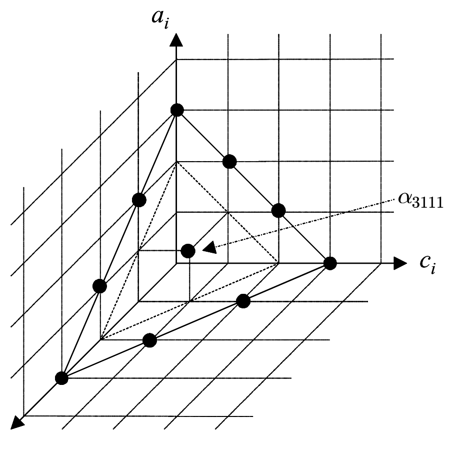

Figure 3: The Geometry Interpretation of Constraint (2.7)

Similarly, let $\mathrm{B}(i)$ be the set of all possible permutations and β_{i,i_1,i_2,i_3} be one of these permutations, for each $i = 1,\ldots,N$, we have:

$$\begin{array}{lll} \beta_{i,i_1,i_2,i_3} & \geq \beta_{i-1,i_1-1,i_2,i_3} & \\ \beta_{i,i_1,i_2,i_3} & \geq \beta_{i-1,i_1,i_2-1,i_3} & \\ \beta_{i,i_1,i_2,i_3} & \geq \beta_{i-1,i_1,i_2,i_3-1} & \\ V_i & \leq \beta_{i,i_1,i_2,i_3} & , \forall (i_1,i_2,i_3) \in \mathrm{B}(i) \end{array} \qquad (2.8)$$

and the boundary inequalities are:

$$\begin{array}{lll} \alpha_{i,i,0,0} & \geq a_i & \\ \alpha_{i,0i,0} & \geq b_i & \\ \alpha_{i,0,0,i} & \geq c_i & \\ \beta_{i,i,0,0} & \geq d_i & \\ \beta_{i,0i,0} & \geq e_i & \\ \beta_{i,0,0,i} & \geq f_i & , i = 1,\ldots,N. \end{array} \qquad (2.9)$$

Finally, we will add the following terms into our objective function to be minimized:

$$\sum_{i=1}^{N}\Big[\sum_{(i_1,i_2,i_3)\in \mathrm{A}(i)}(\alpha_{i,i_1,i_2,i_3}) + \sum_{(i_1,i_2,i_3)\in \mathrm{B}(i)}(\beta_{i,i_1,i_2,i_3})\Big] \\ -\sum_{i=1}^{N-A_X} U_i - \sum_{i=1}^{N} V_i.$$

The minus sign before U_i and V_i is crucial. Also, exactly one of the constraints governing U_i and α_{i,i_1,i_2,i_3} (V_i and β_{i,i_1,i_2,i_3}) must be binding in an optimal solution.

Now, for a general case with $|I_X|$ number of events that increment $X(t)$ and $|D_X|$ number of events that decrement $X(t)$, there are $\sum_{i_m=0}^{i}\sum_{i_{m-1}=0}^{i_m}\cdots\sum_{i_2=0}^{i_3} 1$ (m = $|I_X|$ or

$|D_X|$) permutations for each i and constraints (2.6) to (2.9) will be changed to:

$$U_{i-A_X} \le V_i \qquad ,i = A_X + 1,\dots,N \qquad (2.6')$$

$$\begin{aligned} \alpha_{i,i_1,\dots,i_{|I_X|}} &\ge \alpha_{i-1,i_1-1,\dots,i_{|I_X|}} \\ &\vdots \\ \alpha_{i,i_1,\dots,i_{|I_X|}} &\ge \alpha_{i-1,i_1,\dots,i_{|I_X|}-1} \\ U_i &\le \alpha_{i,i_1,\dots,i_{|I_X|}} \quad ,\forall(i_1,\dots,i_{|I_X|}) \in \mathrm{A}(i) \end{aligned} \qquad (2.7')$$

$$\begin{aligned} \beta_{i,i_1,\dots,i_{|D_X|}} &\ge \beta_{i-1,i_1-1,\dots,i_{|D_X|}} \\ &\vdots \\ \beta_{i,i_1,\dots,i_{|D_X|}} &\ge \beta_{i-1,i_1,\dots,i_{|D_X|}-1} \\ V_i &\le \beta_{i,i_1,\dots,i_{|D_X|}} \quad ,\forall(i_1,\dots,i_{|D_X|}) \in B(i) \end{aligned} \qquad (2.8')$$

$$\begin{aligned} \alpha_{i,i,0,\dots,0} &\ge E_{1i} \\ &\vdots \\ \alpha_{i,0,\dots,0,i} &\ge E_{|I_X|,i} \\ \beta_{i,i,0,\dots,0} &\ge E'_{1i} \\ &\vdots \\ \beta_{i,0,\dots,0,i} &\ge E'_{|D_X|,i} \qquad ,i = 1,\dots,N, \end{aligned} \qquad (2.9')$$

where $E_{ji}, j = 1,\dots,|I_X|$ and $E'_{ji}, j = 1,\dots,|D_X|$ are the events that increment and decrement $X(t)$ respectively.

3 *G/G/*1 QUEUE FORMULATION

In this section, we illustrate the mapping an EG into an LP using a *G/G/*1 queue example. A more general example is given in Schruben (2000). Figure 4 shows the event graph for a *G/G/*1 queue with single-class of customers, a batch size of 1, and a non-preemptive service discipline. $Q(t)$ is the number of jobs waiting in queue at time t and $R(t)$ is the number of available resources at time t. Let ta_i be the time between the arrival of the i-1-th job and the i-th job; ts_i be the service time of the i-th service.

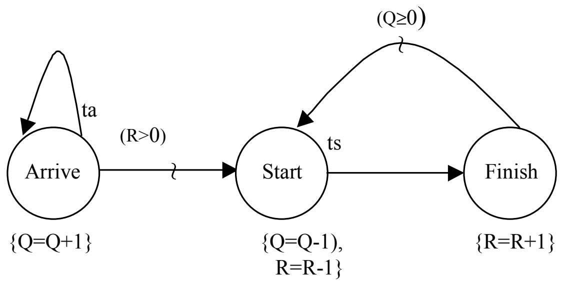

Figure 4: EG for a G/G/1 Queue

For this system, the linear program that specifies the dynamic system trajectory is almost obvious. The non-negative decision variables in our linear program will be the event times. Denote A_i, S_i, and F_i as the time of the i-th *A*rrival event, *S*tart event, and *F*inish event respectively.

From the feasibility conditions of the two state variables $Q(t)$ and $R(t)$ and the two unconditional edges, we will have the following LP formulation:

$$\min \quad \textstyle\sum_{i=1}^{N}(A_i + S_i + F_i)$$

subject to:

$$\begin{aligned} A_{i+1} &= A_i + ta_{i+1} &&,i = 2,\dots,N-1 &&(3.1) \\ F_i &= S_i + ts_i &&,i = 1,\dots,N &&(3.2) \\ S_i &\ge A_i &&,i = 1,\dots,N &&(3.3) \\ S_{i+1} &\ge F_i &&,i = 1,\dots,N-1 &&(3.4) \end{aligned}$$

In fact, we do not need to include the job arrival times in our objective function since they are not scheduled by conditional edges. Constraints (3.1) and (3.2) merely say that time is non-negative. Constraint (3.3) states that the server cannot start the i-th job until at least i jobs have arrived. Constraint (3.4) ensures that the single server cannot process more than one job at a time.

4 FORMULATING SCHEDULING PROBLEMS

The linear program for the multiple resource simulation becomes more interesting when the order in which J jobs are to be processed is to be determined. Translating the event graph into what now becomes a mixed integer scheduling program (MIP) gives a formulation of the classical parallel server 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 (say, to derive an order- release rule).

For the parallel resource event graph, 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.

The MIP formulation of the scheduling problem derived from the simulation event graph requires the definition of two additional binary variables defined as:

θ_{ij} = 1 if job j is the i-th arrival, 0 otherwise, and
η_{ij} = 1 if job j is the i-th service, 0 otherwise.

The optimization model of the parallel server job-scheduling problem with R servers, batch size 1, and J jobs to minimize makespan is defined as:

$$\min \quad F_J$$

subject to:

$$\begin{aligned} F_i &\le F_J &&,i = 1,..,J &&(4.1) \\ A_{i+1} &\ge A_i + ta_j\theta_{ij} &&,i = 1,\dots,J-1, j = 1,..,J &&(4.2) \end{aligned}$$

$$F_i \geq S_i + ts_j\eta_{ij} \quad ,i = 1,...,J, j = 1,..,J \qquad (4.3)$$
$$S_i \geq A_i \quad ,i = 1,...,J \qquad (4.4)$$
$$S_{i+R} \geq F_i \quad ,i = 1,...,J - R \qquad (4.5)$$
$$\sum\nolimits_i \theta_{ij} = 1 \quad ,j = 1,...,J \qquad (4.6)$$
$$\sum\nolimits_j \theta_{ij} = 1 \quad ,i = 1,...,J \qquad (4.7)$$
$$\sum\nolimits_i \eta_{ij} = 1 \quad ,j = 1,...,J \qquad (4.8)$$
$$\sum\nolimits_j \eta_{ij} = 1 \quad ,i = 1,...,J. \qquad (4.9)$$

Constraint (4.1) defines the makespan. Constraints (4.2) to (4.5) are from the Arrive-Arrive edge, Start-Finish edge, Arrive-Start edge, and Finish-Start edge respectively. Constraints (4.6) to (4.9) are the assignment constraints.

This formulation can be made tighter by changing constraints (4.2) and (4.3) to

$$A_{i+1} \geq A_i + \sum\nolimits_j ta_j\theta_{ij}, i = 1,...,J-1 \qquad (4.2')$$
$$F_i \geq S_i + \sum\nolimits_j ts_j\eta_{ij}, i = 1,...,J \qquad (4.3')$$

Small instances (say for $J = 15$) of the above scheduling problem with $\alpha_{ij} = 0$ can be solved quickly.

5 GENERATING CONSTRAINTS FOR CLUSTER TOOLS

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. If LP formulations for cluster tools are available, performance analysis, such as sensitivity analysis (varying processing times, robot speeds, or number of chambers) could be much easier. Therefore, in this section, we illustrate how one might derive LP formulations for cluster tool scheduling.

A generic three-event event graph for p-chamber cluster tools is shown in Figure 5, in which chambers are differentiated by the parameter k and the notation 'M', 'S', and 'F' stand for 'Move', 'Start', and 'Finish' events respectively. The edge M(k)-S(k) represents the activity that the robot moves a wafer to chamber k. For a sequential processing sequence, the duration of this activity equals the time needed to pick up a finished wafer at chamber k-1 plus the time required to move and load this wafer in chamber k; for a parallel processing sequence, other information is needed to represent the time for moving a wafer from the current location to chamber k. The edge S(k)-F(k) symbolizes the activity of processing a wafer at chamber k. Once the robot loads the wafer into chamber k, its next activity is decided according to condition (i_1). Similarly, once a wafer is finished at chamber k, the system will choose the next activity to perform according to condition (i_2). By changing conditions (i_1) and (i_2), one can model cluster tools under different settings.

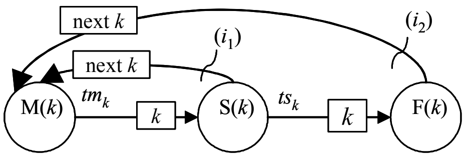

Figure 5: A Generic Event Graph for Cluster Tools

To simplify exposition, we shall first consider a two-chamber sequential processing cluster tool with a single robot and single load lock. Then, we will extend the LP for two-chamber cluster tools to the LP for p-chamber cluster tools. The expanded event graph of a two-chamber cluster tool (without using event parameters to distinguish similar events) is shown in Figure 6. This event graph is the same as the one given in Ding and Yi (2004). Events M_k, S_k, and F_k, k = 1,2 are the events of moving a wafer to chamber k, start processing a wafer at chamber k, and finish processing a wafer at chamber k respectively. Events M_3 and F_3 are the event of moving a wafer from chamber 2 to the load lock and the event that a finished wafer leaves the system. 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 respective events. Define the following state variables: $R(t)$ is the number of available robot(s) at time t; $P_k(t)$ is the number of available slots at chamber k (k = 1,2); $W_k(t)$ is the number of finished wafers waiting at chamber k (k = 1,2). In this two-chamber cluster tool example, all these variables are initialized at 1 and can only take on the values of either 0 or 1 during a simulation.

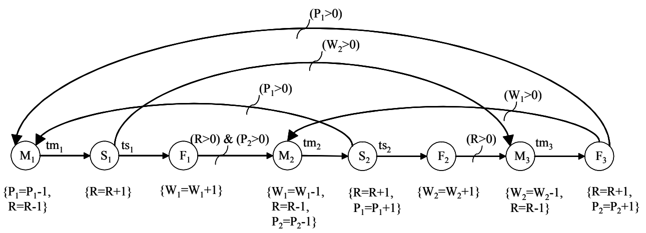

Figure 6: Conventional EG for 2-Chamber Cluster Tools

In this event graph, there are six events that contribute in changing the value of $R(t)$, which is exactly the case considered in Section 2 and therefore can be handled by constraints (2.6) to (2.9).

Examining the constraints (2.6) to (2.9) more closely, we find that most of these constraints are either dominated or identical to constraints generated by the feasibility conditions of other state variables. The only constraints that are not redundant to those for the feasibility conditions of other state variables are constraints governing events S_{1i} and M_{3i}. This special characteristic of the LP leads us to the following theorem.

Theorem 1 For event graphs of cluster tools with sequential processing, the feasibility condition of (2.5) is enforced implicitly by other constraints imposed in the event graph and are hence redundant.

Proof Using Figure 6 as an example, the constraint enforced by edges M$_1$-S$_1$ and the fact that when the *i*-th event of S_1 happens, the next event (i.e., *i*+1-th) of M_1 will only happen after some positive delay imply:

$$S_{1i} = M_{1i} + tm_{1i} \geq M_{1i}$$

and

$$M_{1i} \geq S_{1,i-1}$$

which are equivalent to

$$C_{S_1}(t) - C_{M_1}(t) = 0 \ or \ -1$$

Moreover, the events relationships (enforced by the feasibility conditions of other state variables) on the graph make sure that at any time *t*, if $C_{S_1}(t) - C_{M_1}(t) = -1$, then $C_{S_2}(t) - C_{M_2}(t) = 0$ and $C_{F_3}(t) - C_{M_3}(t) = 0$. Therefore, the feasibility condition of $R(t) \geq 0$ is satisfied. □

In light of Theorem 1, we can eliminate the variable $R(t)$ from the EG in Figure 6 by introducing a new "activity" state variable, $P_3(t)$. $P_3(t)$ is the number of wafers in the activities requiring either of the two chambers. This includes any wafers that may be in the activity of being moved by the robot from the load lock to chamber 1 or being moved by the robot from chamber 1 to chamber 2, but not wafers in the activity of being moved by the robot from chamber 2 to load lock. The modified event graph is shown in Figure 7. Note that the edge F$_3$-M$_1$ is no longer needed.

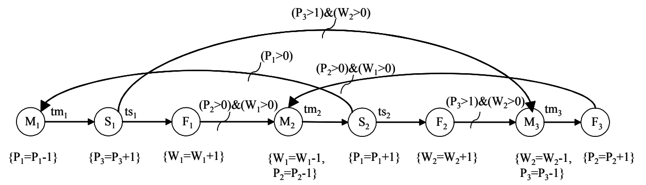

Figure 7: EG without State Variable *R*

What more surprising is that the variable *R*, representing the availability of the robot, is not necessary in the simplified simulation. *The corresponding simplified scheduling model does not include explicit representation of the resource actually being scheduled!* Constraints on this resource are all implicit from the activity indicators. This can be done in this case because it is never optimal to delay moving jobs into chambers of lower indexes when possible (e.g., once S_2 event happens, the robot always will move a wafer from the load lock to chamber 1, regardless of how soon the wafer will be finished at chamber 2).

The EG model in Figure 7 is more concise than Figure 6 in that all the redundancies introduced by $R(t)$ are eliminated and the feasibility of transporting wafers (i.e., robot cannot move two wafers at a time) is enforced by conditions on the activity variable, $P_3(t)$. Figure 7 is also illuminating because the availability of the robot (busy or idle) can be captured by looking at the number of wafers in activities involving chambers 1 and 2, which is equal to the value of $P_3(t)$.

Figure 7 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 the similar state chances and conditions. The only difference is that the conditions on edges S$_1$-M$_{p+1}$ and F$_p$-M$_{p+1}$ are changed to $(P_{p+1}>p-1)\&(W_p>0)$. This condition can be interpreted as starvation avoidance, in that it is optimal for finished wafers be removed from the system if and only if all chambers are full. Although this seems strange at the first glance, it can be shown that this condition will not change the optimal schedule (or the steady-state throughput).

To derive the constraints, we start from the feasibility conditions of the state variables. For example, the feasibility condition of $P_1(t)$ states

$$P_1(t) = C_{S_2}(t) - C_{M_1}(t) + 1 \geq 0$$

which is equivalent to

$$M_{1i} \geq S_{2,i-1} \qquad , i = 2, ..., N.$$

The equalities governed by edges M$_1$-S$_1$, S$_1$-F$_1$, and S$_2$-F$_2$ give:

$$F_{1i} - F_{2,i-1} \geq tm_{1i} + ts_{1i} - ts_{2,i-1} \quad , i = 2, ..., N \tag{5.1}$$

The constraints of state variables $P_2(t)$, $P_3(t)$, $W_1(t)$, $W_2(t)$ are derived using a similar procedure, which yields:

$$F_{2i} - F_{3,i-1} \geq tm_{2i} + ts_{2i} \qquad , i = 2, ..., N \tag{5.2}$$
$$F_{3i} - F_{1,i+1} \geq tm_{3i} - ts_{1,i+1} \qquad , i = 1, ..., N - p + 1 \tag{5.3}$$
$$F_{2i} - F_{1i} \quad \geq tm_{2i} + ts_{2i} \qquad , i = 1, ..., N \tag{5.4}$$
$$F_{3i} - F_{2i} \quad \geq tm_{3i} \qquad , i = 1, ..., N \tag{5.5}$$

Therefore, the LP formulation of a 2-chamber tool is to minimize all event times subject to constraints (5.1) to (5.5).

The above LP can be generalized to model a *p*-chamber cluster tool. This generalized LP is given below without detailed derivations. The constraints BC_{1k} and BC_{2k} are constraints for initial states and terminating states of the tool ('*BC*' is mnemonic for *B*oundary *C*onditions). These conditions do not exist in the 2-chamber cluster tools.

$$\min \quad \sum_{k=1}^{p+1} \sum_{i=1}^{N} F_{ki}$$

subject to:

$$\begin{array}{llll}
P_k(t): & F_{ki} & -F_{k+1,i-1} & \geq tm_{ki} + ts_{ki} - ts_{k+1,i-1} \\
P_p(t): & F_{pi} & -F_{p+1,i-1} & \geq tm_{pi} + ts_{pi} \\
P_{p+1}(t): & F_{p+1,i} & -F_{1,i+p-1} & \geq tm_{p+1,i} - ts_{1,i+p-1} \\
W_k(t): & F_{k+1,i} & -F_{ki} & \geq tm_{k+1i} + ts_{k+1,i} \\
W_p(t): & F_{p+1,i} & -F_{pi} & \geq tm_{p+1,i} \\
BC1_k: & F_{k+2,1} & -F_{1,k+1} & \geq tm_{k+2,1} + ts_{k+2,1} - ts_{1,i+1} \\
BC2_k: & F_{P+1,N-k} & -F_{p-k,N} & \geq tm_{P+1,N-k} - ts_{p-k,N}
\end{array}$$

6 CONCLUSION

In this paper, we introduced a methodology for systematically generating mathematical programming models for optimal resource scheduling. The general approach can be used to create mathematical programming models for most discrete event queueing systems. The ideas are illustrated with two examples: (i) the parallel resource scheduling problem as a mixed integer programming formulation and (ii) a cluster tool scheduling problem with sequential processing as an LP formulation (For some other processing sequences, the LP formulations can also be derived using the procedure described in Section 2).

An interesting result is that the special structure of the mathematical programming formulation of cluster tool dynamics guided us to a new concise and illuminating cluster tools simulation model, which would have been difficult to construct without the help of the mathematics of optimization model. This event graph simulation used "activity" indicator variables – a concept that should be useful in other types of EG simulation applications.

Other advantages of the LP formulation include the benefits we gain when answering the *how* and *what-if* questions regarding the performance of cluster tools, e.g., the structure of the LP tells us some properties of the tools since sensitivity analyses using its dual can be carried out giving infinite perturbation analysis gradient estimators for the processing and move times.

Finally, the structure of the LP for 2-chamber cluster tool tells us that in an optimal solution, constraint (5.1) is always binding and exactly two of the constraints from (5.2) to (5.5) are binding, depending on the processing times and moving times; this results in only four possible optimal schedules – a dramatic reduction in the combinatorial complexity of this scheduling problem. These four schedules can be easily compared to get the optimal schedule. We also found from the mathematical programming representation and EG models together, that the move time from chamber 1 to chamber 2 does not affect the optimal solution at all. Similar results for *p*-chamber cluster tools can be deduced from their corresponding LPs.

ACKNOWLEDGMENTS

The authors wish to thank the National Science Foundation through grant DMI–0323765 for partial support of the research reported here.

REFERENCES

Chan, Wai Kin and L. W. Schruben (2003). Properties of discrete event systems from their mathematical programming representations. *Proceedings of the 2003 Winter Simulation Conference,* ed. Chick, S. E., Sanchez, P. J., Ferrin, D., and Morrice, D. J., *496-502. Piscataway, NJ, USA, IEEE.*

Chan, Wai Kin and Lee W. Schruben (2004). Mathematical Programming Representations for Multi-server Tandem Queueing Networks. Technical Report, UC Berkeley, Berkeley.

Chen, H. X., C. B. Chu and J. M. Proth (1998). Cyclic scheduling of a hoist with time window constraints. *IEEE Transactions on Robotics and Automation* 14(1): 144-152.

Ding, Shengwei and Jingang Yi (2004). An event graph based simulation and analysis of multi-cluster tools. *Proceedings of the 2004 Winter Simulation Conference,* ed. Ingalls, R. G., Rossetti, M. D., Smith, J. S., and Peters, B.A., *Piscataway, NJ, USA.*

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, 189-192. New York, NY, USA, IEEE.*

Pederson, D. E. and C. E. Trout (2002). Demonstrated Benefits of Cluster Tool Simulation. *Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing (MASM 2002), 58-63, Tempe, AZ,.*

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.

Phillips, L. W. and P. S. Unger (1976). Mathematical programming solution of a hoist scheduling program. *AIIE Transactions* 8(2): 219-225.

Rostami, S. and B. Hamidzadeh (2002). Optimal scheduling techniques for cluster tools with process-module and transport-module residency constraints. *IEEE Transactions on Semiconductor Manufacturing* 15(3): 341-349.

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

Schruben, Lee W. (1983). Simulation Modeling with Event Graphs. *Communications of the Association of Computing Machinery*.

Schruben, L. W. (2000). Mathematical programming models of discrete event system dynamics. *Proceedings of the 2000 Winter Simulation Conference,* ed. Joines, J. A., Barton, R. R., Kang, K., and Fishwick, P. A., *381-385. Piscataway, NJ, USA, IEEE.*

Schruben, Lee W. (2003). Conditional Parametric Petri Nets and their Mapping to Simulation Event Graphs. Technical Report, Italy.

Shapiro, G. W. and H. L. W. Nuttle (1988). Hoist Scheduling for a Pcb Electroplating Facility. *IIE Transactions* 20(2): 157-167.

Srinivasan, R. S. (1998). Modeling and performance analysis of cluster tools using Petri nets. *IEEE Transactions on Semiconductor Manufacturing* 11(3): 394-403.

Venkatesh, S., R. Davenport, P. Foxhoven and J. Nulman (1997). A steady-state throughput analysis of cluster tools: dual-blade versus single-blade robots. *IEEE Transactions on Semiconductor Manufacturing* 10(4): 418-424.

Yen, H.-C. (1999). Integer Linear Programming and the Analysis of Some Petri Net Problems. *Theory Computing Systems*.

Zuberek, W. M. (2001). Timed Petri nets in modeling and analysis of cluster tools. *IEEE Transactions on Robotics & Automation* 17(5): 562-575.

AUTHOR BIOGRAPHIES

WAI KIN CHAN is a PhD candidate in the Industrial Engineering and Operations Research Department at the University of California, Berkeley. His research interests include designs of simulation experiments, queueing theory, and computational optimization. His e-mail is <kin@ieor.berkeley.edu> and his web address is <www.ieor.berkeley.edu/~kin>.

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