Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

RESOURCE GRAPHS FOR MODELING LARGE-SCALE, HIGHLY CONGESTED SYSTEMS

Paul Hyden

Department of Mathematical Sciences
O-309 Martin Hall
Clemson University
Clemson, SC 29634-0975, U.S.A.

Lee Schruben
Theresa Roeder

Department of Industrial Engineering
and Operations Research
University of California at Berkeley
4135 Etcheverry Hall
Berkeley, CA 94720-1777, U.S.A.

ABSTRACT

Simulations often execute too slowly to be effective tools for decision-making. In particular, this problem has been found in semiconductor manufacturing where conventional job-driven simulation models explicitly track each lot of wafers as it progresses through the system. While a job-driven simulation model offers some advantages, they inherently execute slowly. This paper explicitly defines resource-driven modeling. Here jobs are implicitly tracked through their resource usage. Resource-driven simulations typically run much faster than job-driven simulations. This speed-up is insensitive to congestion and is most dramatic when the system is highly congested and therefore most interesting to the analyst. There can also be a significant reduction in memory footprint. However, there is a potential tradeoff in information loss.

1 INTRODUCTION

Most conventional simulation software packages are designed around a *job-driven* approach. Here jobs are modeled as active system entities while system resources are passive. Describing how jobs move through their processing steps, seizing available resources whenever they are needed, creates the simulation model. Records of every step of every job in the system are created and maintained. Therefore, the speed and space complexity of these simulations must be at least on the order of some polynomial of the number of active jobs in the model. Job-driven simulations are convenient for low-volume, high-mix manufacturing or when fast simulation execution speed is not as important as detailed information or system animation.

In a resource-driven simulation, see (Schruben 2001), all system entities such as jobs, tools, and operators are called resources and are treated with equal status. The models are organized around changes in resource availability. At each resource state change, the consequences for other resources are processed using integer operations, typically incrementing or decrementing the numbers of available resources and scheduling future resource changes. For example, only integer counts of jobs of particular types at different steps are necessary. The system's state is wholly described by the status of resources, expressible as integers. The main advantages of resource-driven simulations is that execution speed and memory footprint do not change significantly as the system becomes more congested. Job and resource-driven simulations are not exclusive modeling styles but can and should coexist in a simulation study.

2 DEFINITION OF RESOURCE-DRIVEN MODELING

A resource-driven simulation model randomly generates a jointly dependent, finite set of point processes. All system performance statistics can be derived from counting the numbers of occurrences of events of different types. Say the system can be described using k types of resources. The output from a single run of the simulation with random number seed ω will consist of $N(\omega)=(C_1^+,C_1^-,\ldots,C_k^+,C_k^-)$. Here C_i^+ and C_i^- are point processes, indexed by time t, that record when the availability of resource i increases and decreases, respectively. For example, if "resource" j is the number of jobs waiting in queue j, then the queue size at time t will be $C_j^+(t)-C_j^-(t)$. In resource-driven models, changes in the availability of resources constitute all of the activity of the simulation.

Computation of $N(\omega)$ evolves by processing the implications of current resource availability changes on future resource availability changes. As is common in event scheduling simulations, all events are scheduled relative to the current time. Here, *time 0* will always represent the current time. Since only the time prior to the current *time 0* is recorded, the counting processes comprising $N(\omega)$ are only well defined for $t<0$. A future resource change list serves as the list of scheduled events and is comprised of all resource availability changes scheduled after the current *time 0*. Denote $t_0(C_j^+)$ and $t_0(C_j^-)$ as the time of the next future increase and decrease in availability of resource *j*.

To facilitate the scheduling of resource availability changes, define the function $U(C,t)$ which adds an additional point at time $t\geq 0$ to counting process *C*. This schedules the resource availability change associated with *C* to occur after t time units.

3 RESOURCE GRAPHS

The basic model development tool for a resource-driven simulation is a resource graph that describes how resources interact. The resource graph provides a useful visual display of the relationships between resources, and represents every event explicitly. A resource graph is a derivative of an event graph (Schruben and Schruben 2000), so the definition is similar. However, a resource graph has no time-delayed edges. The system resource graph is composed of several subgraphs that describe the actions triggered by changes in the availability of each type of resource. These actions include scheduling future resource changes. An individual resource subgraph is initialized and run to completion for each scheduled resource change. When there are multiple resource changes scheduled at the same simulated time, the subgraph for the highest priority resource change is processed first.

A resource graph is composed of nodes and edges. Modifications of the resource state, called events, occur at the nodes while conditions are represented with edges. In Figure 1 a sample element of a resource graph is shown for events *A* and *B*. When event *A* is executed, the actions listed at *{A actions}* are executed, using $input_A$ for the values of input parameters. These actions involve changes of the state as well as random variable generation. The actions at event *A* are executed and the output parameters $output_{AB}$ are computed. If $condition_{AB}$ is true, event *B* is scheduled with priority $priority_{AB}$ and parameter values $input_B \leftarrow output_{AB}$. The highest priority event is executed next until no further events remain.

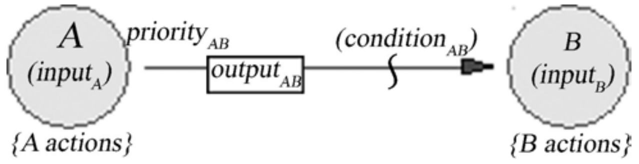

Figure 1: Resource Graph Element

Since many different resource changes may trigger the same events, the resource graph can often represent many subgraphs with few nodes by sharing common nodes across subgraphs.

3.1 Resource Change Initialization and Scheduling Events

There are several special events that will be defined for every resource graph. For each resource *R(i)*, *i=1,2,…,k,* associate 4 events: *R+(i), R-(i), +R(i,t), -R(i,t).*

The resource change events *R+(i)* and *R-(i)* schedule events and makes state changes that are caused by increasing and decreasing, respectively, the number of resources *i* available at the current *time 0*. Hence, when $t_0(C_j^+)=0$ [$t_0(C_j^-)=0$], event *R+(i)* [*R-(i)*]initializes the resource subgraph associated with an increase [decrease] of resource *i*. Event priorities are associated within each resource subgraph.

+R(i,t) [–R(i,t)] schedules a future increase [decrease] of resource *i* to occur after *t* time units. This event calls the function $U(C_i^+,t)$ $[U(C_i^-,t)]$. An increase or decrease in resource *i* should include the appropriate update of the associated point process. (No changes in clock time are permitted.)

3.2 Vertices

The resource graph consists of a set of vertices *V* representing the events. Associated with event $v\in V$ is a mapping $h_v(R_{i-1},S_{i-1},T_{i-1},A,\omega)\rightarrow(R_i,S_i)$ with

- R_i: value of resource variables at resource change *i.*
- S_i: value of statistics at resource change *i*.
- T_i: simulation clock time at resource change *i.*
- *A*: attribute values for event *v*; these values were determined at the time the event was scheduled.
- ω: the function may be a random variable in some cases.

3.3 Edges

Directed edges *D* describe the conditions between events. Edge $d_{AB}\in D$ $(A,B\in V)$ is a mapping $d_{AB}(R_i,S_i,T_i,A,\omega)\rightarrow\{true,false\}$ evaluated at the termination of event *A* that schedules *B* if *true*. When multiple events are scheduled, events are processed in order according to their priority. The priority of an event is determined by its scheduling edge. The scheduled event with the lowest valued priority is executed next.

3.4 Event List for Resource Graph

Note that time does not advance on the resource graph, so the events list acts as a priority queue on the resource subgraphs according to their priorities.

4 ALGORITHM

The basic algorithm for processing a resource-driven simulation is a serial interaction between the resource graph and the present resource availability changes.

1. Initialize a set of counting processes $N(\omega)=(C_1^+,C_1^-,\ldots,C_k^+,C_k^-)$, with at least one resource change at $t\geq 0$.
2. Advance the clock by moving the next resource change to the current time by setting $t=0$.
3. Process Resource Changes: Execute the resource subgraph scheduled with the highest priority.
4. Return to 2 until the run is terminated.

5 RESOURCE GRAPH CONSTRUCTION EXAMPLES

5.1 Resource Graph Example: *G/G/S* Queue

As a simple example, we will construct a resource graph for a *G/G/S* queue. In the *G/G/S* queue, there are two types of resources: servers [*R(1)*] and queues [*R(2)*]. As mentioned in 3.1, the required events associated with resource R(1) and R(2) are *R+(1), R-(1), +R(1,t), -R(1,t)* and *R+(1), R-(1), +R(1), -R(1),* respectively.

For mnemonic purposes, we will denote resource *R(1)* as *S* and resource *R(2)* as *Q*. The events associated with *S* and *Q* become *S+, +S(t),* and *Q+, +Q(t),* (only *increases* in server or job availability trigger resource graph executions in this example).

5.1.1 Subgraph: Increase in Jobs in Queue

The event *Q+* occurs when a new job arrives and increases in the number of jobs in the queue (resource *Q*). The associated resource subgraph includes all nodes and edges that emanate from the *Q+* node. Figure 2 displays the subgraph for this resource change.

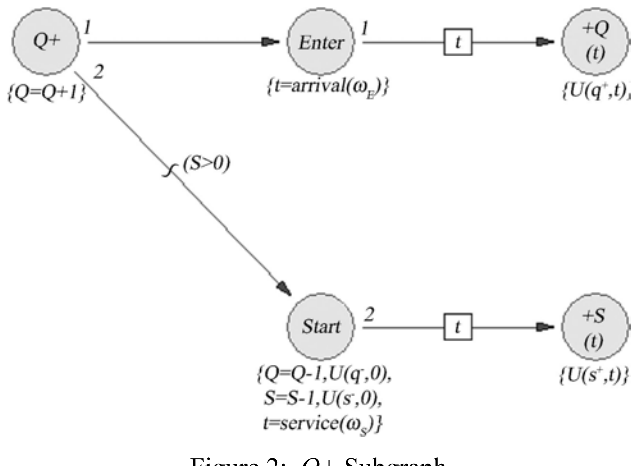

Figure 2: *Q*+ Subgraph

5.1.2 Subgraph: Increase in the Availability of Servers

The second type of resource change is an increase in server availability (resource *S*) denoted by *S*+. The resource subgraph associated with this event is executed when a server completes service. This subgraph includes all nodes and edges that emanate from the *S*+ node. Figure 3 displays the subgraph for this resource change.

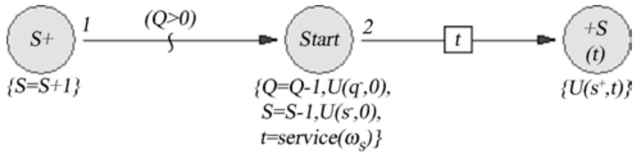

Figure 3: *S*+ Subgraph

5.1.3 *Increment Q* Event (*Q+)*

This is the trigger event for the *Q*+ resource subgraph. The immediate impact of increasing resource *Q* is included here; the queue is increased *{Q=Q+1}*. If there is a server available (*S*>*0)*, service will begin immediately in the *Start* event. The next job resource is slated to enter by scheduling event, *Enter*.

5.1.4 Increment S Event (*S*+)

This is the trigger event for the *S*+ resource subgraph. The immediate impact of increasing resource *S* is included here; the servers are increased *{S=S+1}*. If there are jobs waiting in the queue (*Q*>*0*), then service will begin immediately in the S*tart* event.

5.1.5 Scheduling Resource Change Events

The events that trigger *Q*+ and *S*+ to occur in the future are denoted +*Q(t)* and +*S(t)*, respectively.

5.1.6 Start Event

The same *Start* event appears in both the *Q*+ and *S*+ resource subgraphs. This *Start* event represents a job starting service. Resources *Q* and *S* are decremented and the associated point processed are updated. *{Q=Q-1, U(q⁻,0), S=S-1,U(s⁻,0)}*. The service completion event is then scheduled by the +*S(t)* vertex after a random service time, *t,* is generated according to the function *service* and random number stream ω_S.

5.1.7 Enter Event

A *Enter* event represents the next arrival to the queue being scheduled. The random time *t* until the next arrival occurs is generated according to the function *arrive* and the random stream ω_E. Event $Q+$ is scheduled by the $+Q(t)$ vertex to occur after t time units.

5.1.8 Resource Graph

By combining the subgraphs for $Q+$ and $S+$, the resource graph is formed. The resource graph is shown in Figure 4. Note that the *start* event node only appears once because it operates the same whether it was triggered by a server increase ($S+$) or a queue increase ($Q+$).

The relative performance of a job-driven model and a resource-driven model is shown in Figure 5. When the system becomes more congested, the job-driven simulation slows to a virtual stand still. The resource-driven simulation execution speed is virtually unaffected. It is precisely these bursts of high congestion that are typically the most interesting in simulation studies of queueing systems. Figure 6 presents the performance ratio of the job-driven style to the resource-driven style, using the same software package. The job-driven approach chokes with increased congestion and the ratio skyrockets. When the system breaks from extreme congestion, only the frequency of easily processed arrivals keeps this ratio from exploding.

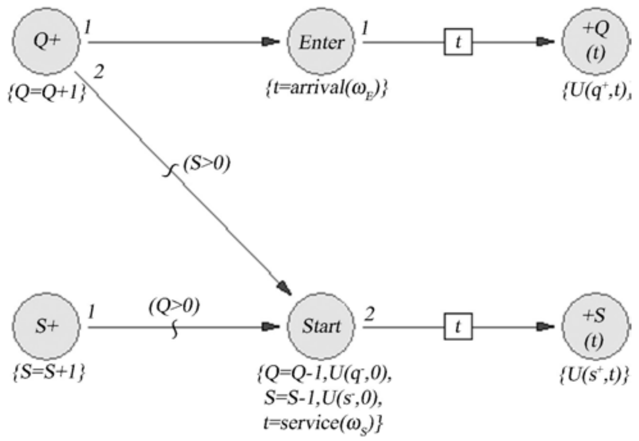

Figure 4: Resource graph for *G/G/S* queue

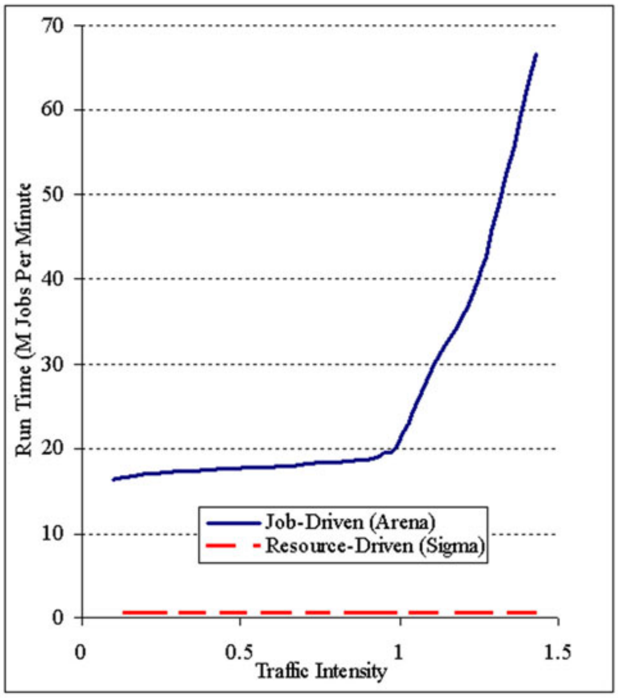

Figure 5 : Run Times for a G/G/1 Queue Simulation Using Different Styles (Altoik et al. 2001)

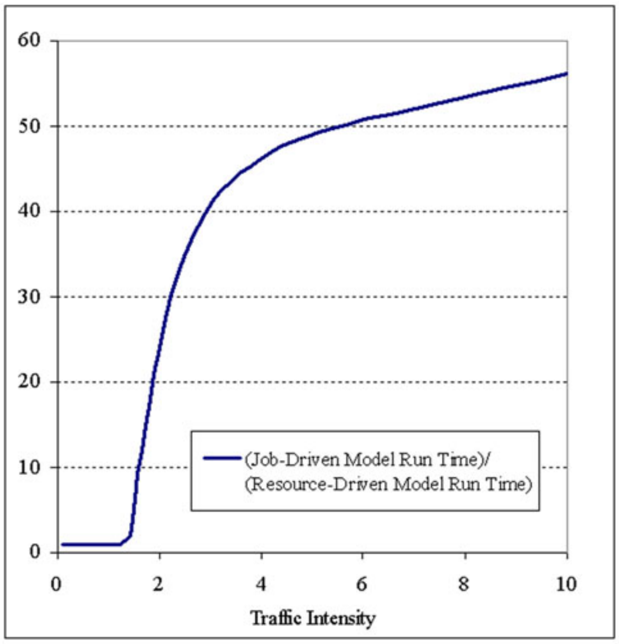

Figure 6: Ratio of Run Times for a G/G/1 Queue Simulation Using the Same Software Package but Different Styles

5.2 Job Shop Example

A more complex resource-driven example is a general job shop simulation. A job shop consists of multiple stations with different job types, each with possibly different routings through the stations. Jobs may re-enter the same station as is common in semiconductor manufacturing. This model is based on an example from Law and Kelton (2000).

As in all queueing system models, it is useful to distinguish the two types of resources, queues and servers. *Q(i)* will refer to the number of jobs at the i^{th} queue, and *S(j)* will refer to the number of available servers at the j^{th} station. The jobshop will have *I* queues, *J* stations and *a* job types. As typical to a jobshop, jobs require a single station at each step, meaning each station may serve multiple queues while each queue can only be served by one station. Different job types and jobs at different steps of their routing are kept in separate queues.

To define our model and the relationship between resources, it is useful to describe the following functions. Note that each of these functions can be assumed to execute in O(1) (constant) time.

- *next(j)* is the next queue that will be served by station *j*. *next(j)=0* indicates that all queues served by *S(j)* are empty.
- $next^+(i,j)$: update of *next(j)* when *Q(i)* is increased.
- $next^-(i,j)$: update of *next(j)* when *Q(i)* is decreased.
- *enter(k)* is the queue that an newly arriving job of type *k* increases
- *type(i):* is the job type of resources in queue *i*.
- *server(i)* is the station that serves *Q(i)*.
- *queue(j)* is the queue served by *S(j)*.
- $choose(\omega_{E,1})$:a random variable indicating which queue to increase upon the next system arrival.
- $arrive(\omega_{E,2})$: a random variable indicating the simulation time until the next arrival.
- $service_{i,j}(\omega_{S,j})$: a random variable indicating the service time for *Q(i)* on *S(j)*.
- *route(i)* is the unique queue entered by a job that last exited queue i. *route(i)=0* indicates that the job completes service after exiting queue *i*.

Although this model is much more complex, the resource graph can be written as the composition of two subgraphs for any size routing or number of servers. Using event parameters enables this efficiency.

5.2.1 Subgraph: Increase of Queue i

The first type of resource change is an increase of a queue *i* (resource *Q(i)*), which will be denoted *Q+(i)*. This subgraph occurs when there is an arrival to queue *i*. The subgraph associated with increases to the queue includes all nodes and edges that emanate from the *Q+(i)* node. Figure 7 displays the resource subgraph for this resource change.

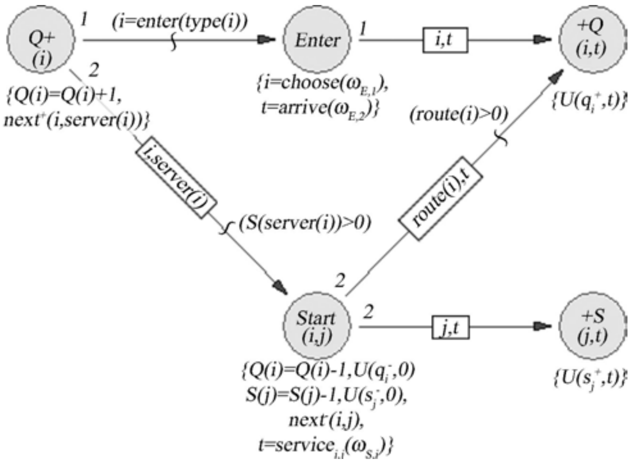
Q+ (i)
1
(i=enter(type(i))
Enter
1
i,t
+Q (i,t)
2
{Q(i)=Q(i)+1, next⁺(i,server(i))}
{i=choose(ω_E,1), t=arrive(ω_E,2)}
(route(i)>0)
{U(q_i⁺,t)}
i,server(i)
route(i),t
(S(server(i))>0)
2
Start (i,j)
2
j,t
+S (j,t)
{Q(i)=Q(i)-1,U(q_i⁻,0) S(j)=S(j)-1,U(s_j⁻,0), next⁻(i,j), t=service_i,i(ω_S,i)}
{U(s_j⁺,t)}

Figure 7: *Q+(i)* Subgraph

5.2.2 Subgraph: Increases in Available Servers

The second type of resource change is an increase of the idle servers at station *i* (resource *S(i)*), which will be denoted event *S+(i)*. This subgraph occurs when a server has completed service. The associated subgraph includes all nodes and edges that directly or indirectly emanate from the *S*+ node. Figure 8 displays the subgraph for this resource change.

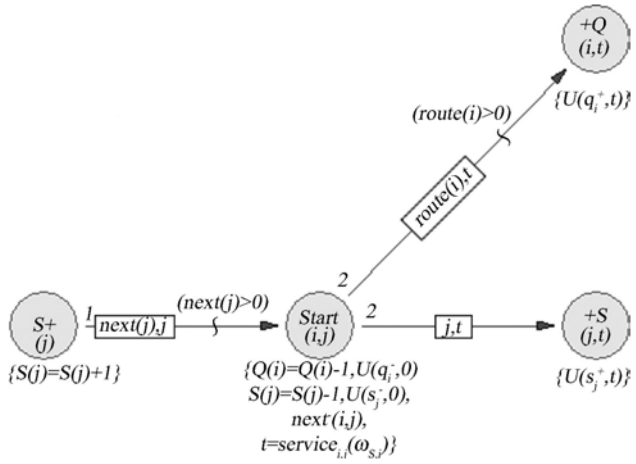

Figure 8: *S+(j)* Subgraph

5.2.3 Increase Queue Event (*Q+(i)*)

This is the trigger event for the *Q+(i)* subgraph. The immediate impact of a job entering queue *i* and increasing resource *Q(i)* is included here. The queue is increased

$\{Q(i):=Q(i)+1\}$. In addition, the next queue to be served by *server(i)* is updated. This is necessary because each station holds a one-to-many relationship with the queues. If there is a server available at the station serving queue i *(S(server(i)>0)*, service begins by scheduling the *Start* event for queue *i* and station *server(i).* If this job is entering the first queue on its routing, the next job is slated to enter by scheduling event *Enter*.

5.2.4 Increase Servers Event (*S+(j)*)

This is the trigger event for the *S+(j)* subgraph. The immediate impact of increasing resource *S(j)* is included here. The servers are increased *$\{S(j):=S(j)+1\}$,* and if there is a job waiting for service at station j (*next(j)>0*), service begins by scheduling the event *start*.

5.2.5 Scheduling Resource Change Events

The events that schedule *Q+(i)* and *S+(j)* to occur in *$t>0$* time units are denoted *+Q(j,t)* and *+S(j,t)*, respectively.

5.2.6 Start(i,j) Event

A *Start(i,j)* event represents a job leaving queue *i* to start service at station *j*. The server is freed at the completion of service. The following state changes occur for a start event:

- Resources *Q(i)* and *S(j)* are decremented and the associated point processed are updated. *$\{Q(i)=Q(i)-1, U(q_i^-,0), S(j)=S(j)-1, U(s_j^-,0)\}$*.
- A random service time, *t,* is generated according to the function $service_{i,j}$ and random stream ω_S.
- The server is slated for completion by scheduling event *+S(j,t).*
- If the job is not finished, it is moved to the next queue by scheduling event *+Q(route(i),t).*

Note that the same start event appears in both the *Q(i)+* and *S(j)+* subgraphs.

5.2.7 Enter Event

A *Enter* event represents a job entering the queue. Each time this resource enters, the next arrival to the queue is scheduled. The following state changes occur for a enter event:

- The random time *t* until the next arrival is generated according the to the function *arrive* and random stream $\omega_{E,1}$.
- The entering queue *i* of the next arrival is generated according to the function *choose* and random stream $\omega_{E,2}$.
- Event *Q+(i)* is scheduled to occur in *t* time units.

5.2.8 Resource Graph

By combining the subgraphs for *Q+(i)* and *S+(j)*, the resource graph is formed. The resource graph is shown in Figure 9. Note that the *Start* event node only appears once because it operates the same whether it was triggered by a server increase (*S+(i)*) or a queue increase (*Q+(j)*).

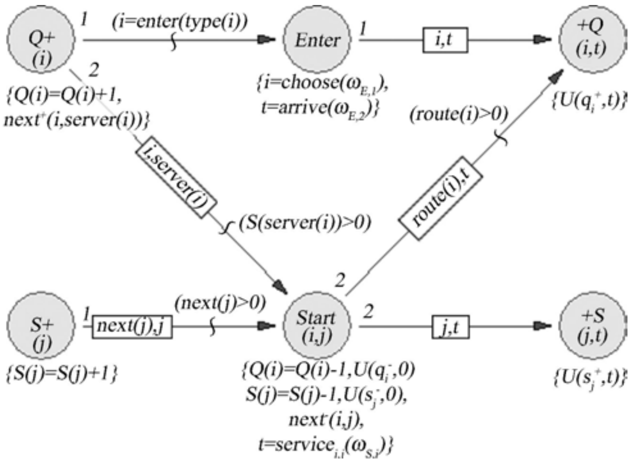

Figure 9: Resource Graph for General Job Shop of Any Size with Any Number of Job Types

6 CONCLUSION

Resource graphs are presented for developing simulation models that focus on changes in resource availability and limit resource representations to integers. Subgraphs isolate the consequences of a single change in resource availability and explicitly schedule future resource availability changes. This simplifies the initial model development by breaking the system down into components that can later be recombined. Resource-driven modeling can speed up execution time, making simulation more relevant to the decision making process.

REFERENCES

Altoik, T., Kelton, W. D., L'Ecuyer, P., Nelson, B. L., Schmeiser, B. W., Schriber, T. J., Schruben, L. W., And Wilson, J. R. (2001), *Various Ways Academics Teach Simulation: Are They All Appropriate*? Panel In The *Proceedings Of The 2001 Winter Simulation Conference*, Arlington, VA.

Kelton, W. D. and R.P. Sadowski, and D.A. Sadowski (1998), *Simulation with Arena.* McGraw-Hill.

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

Schruben, Donna A. and Lee. W. Schruben (2000). *Event Graph Modeling*. Custom Simulations.

Schruben, Lee W. (2001), "Mathematical Programming Models of Discrete Event System Dynamics", *Proceedings of the 2000 Winter Simulation Conference,* December 10-13, 2000, Orlando FL.

AUTHOR BIOGRAPHIES

PAUL HYDEN is an Assistant Professor in the Department of Mathematical Sciences at Clemson University. He received all of his degrees in Operations Research and Industrial Engineering from Cornell University. His email address is <hyden@clemson.edu>.

LEE SCHRUBEN is Professor and Chairman of the Department of Industrial Engineering and Operations Research at Berkeley. Prior to joining the Berkeley faculty he was in the School of Operations Research and Industrial Engineering at Cornell University where he held the Andrew S. Schultz Professorship in Industrial Engineering. His email address is <schruben@ieor.berkeley.edu>.

THERESA ROEDER is a Ph.D. student in the Department of Industrial Engineering and Operations Research at UC Berkeley. She received B.S. and M.S. degrees in Management Science from Case Western Reserve University, and an M.S. in IEOR from Berkeley. Her email address is <roeder@ieor.berkeley.edu>.