Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

PERT SCHEDULING WITH RESOURCES USING QUALITATIVE SIMULATION GRAPHS

Ricki G. Ingalls

Corporate Supply Chain Operations
Compaq Computer Corporation
20555 SH 249
Houston, TX 77070

Oklahoma State University
School of Industrial Engineering and Management
322 Engineering North
Stillwater, OK 74078

Douglas J. Morrice

Department of Management Science
and Information Systems
Graduate School of Business
The University of Texas at Austin
Austin, TX 78070

ABSTRACT

The Qualitative Simulation Graph Methodology (QSGM) is well suited to address the PERT scheduling with resources problem. The coverage property of QSGM has two important implications for the PERT scheduling problem. First, it means that all possible schedules are represented. Second, it means that, as long as the delay time intervals are not violated, we can characterize all possible outcomes of a decision that needs to be made in the schedule. This gives rise to the possibility of robust point-in-time scheduling decisions without needed to re-run the simulation in order to get the results.

1 INTRODUCTION

The Qualitative Simulation Graph Methodology (QSCM) that has been developed by Ingalls, et. al. (2000) is a general purpose qualitative discrete-event simulation (QDES) framework that can be used for any type of discrete-event simulation problem. However, because of the thread explosion problem, the application of QSGM is better suited for certain types of problems. One type of problem that works better with the QSGM are problems with a fixed time horizon. Another type of problem that works well in this methodology is more strategic and less granular problems. Activity-on-arc PERT networks fit the fixed time horizon and more strategic class of problems. The QSGM brings a very important property to the PERT class of problems, and that is the coverage property. The coverage property means that all possible threads of execution are characterized in the problem. This has two important implications for the scheduling problem. First, it means that all possible schedules are represented. Second, it means that, as long as the delay time intervals are not violated, we can characterize all possible outcomes of a decision that needs to be made in the schedule.

This paper covers the application of QSGM to PERT networks with resources. Section 2 outlines the process for converting a PERT network with resources to an event graph network. Section 3 gives an example of a PERT network with resources. In this section we give one example of PERT networks with constant delay times and another example of the same PERT network with interval delay times. In this section, we also go into two key topics: the meaning of the output and the concept of point-in-time decision making for scheduling. Section 4 gives concluding remarks and possible future research.

2 CONVERTING A PERT NETWORK WITH RESOURCES TO A SIMULATION GRAPH

PERT networks with resources are complicated to model in the event graph paradigm. In a PERT network with resources, not only do all of the previous activities need to be complete, but all of the necessary resources must be available as well in order to start the new activity. In order to convert a PERT network with resources to an event graph, the following steps must be followed:

1. The event graph network adds an initialization node, called node *0* for the purpose of initializing the available number of resources in the network.
2. For each node in the PERT network, create a corresponding node in the event graph network. This node will be the hit node and count the number of previous completed activities that are necessary to execute the activities associated with

the corresponding PERT node. For our purposes, the nodes will be labeled *Hx*, where *x* is the number of the PERT node.

3. For each arc in the PERT network, there are two more nodes that are created in the event graph. The first is the start event that signifies the start of the activity and the finish event which signifies the completion of the activity. For our purposes, the start node is labeled *Sx.y* and the finish node is labeled *Fx.y*, where *x* is the number of the head node and *y* is the number of the tail node for the given arc in the PERT network.
4. Each hit node is connected to its corresponding start nodes with a scheduling condition and an execution condition. The scheduling condition checks to see if the hit node has been hit the proper number of times (i.e. all predecessor activities have been completed). The execution condition checks to see if the resources are available. The reason for this execution condition is especially critical in a qualitative simulation and its purpose will become clear later in this chapter.
5. Each finish node has arcs pointing back to start nodes of activities that could need released resources. Each one of these arcs has a scheduling condition to determine if the hit node has been triggered and an execution condition to determine if the necessary resources are available.
6. The variables defined in the model include *Hx*, which is the number of hits on the corresponding hit node, and *Qxy*, which is a boolean that allows the start of activity *x.y* if *Qxy = TRUE*.

As can be seen in the conversion of the PERT with resources network in Figure 1 to the corresponding event graph in Figure 2, the conversion can be very complicated in order to handle competing resources that are available for a given task. The qualitative simulation graph methodology is especially well suited to handle this type of network. With the execution conditions that are in place in the model, every possible combination of the deployment of resources will be simulated. With interval time on the delays of the PERT network, we can simulate every possible state and make scheduling decisions based on future possible states. Each scheduling decision will have already been run in the model and the scheduler would only need to consult the output of the model to make his resource deployment decision.

In the example in Figure 1, we have a PERT network with 5 activities and two resources. With each activity, we have the quantity of each type of resource that is needed. In Figure 2, we have the translation of the PERT network in Figure 1 to an event graph. In Figure 2, one can see the pattern described above. First, the *0* node is created so that the two resources can be initialized to their capacities. Second, each node in the PERT network in Figure 1 has a corresponding hit node. When each of the incoming arcs come into the hit node, then the scheduling conditions on the outgoing arcs become true. In the case of node *H1*, the two outgoing arcs both occur in zero time and one of the arcs is not prioritized over the other. In the QDES framework, these two pending events on the future events calendar would be members of a NOS and so two threads would be created. One thread would assume that the event *S1.2* would be executed first and the second would assume that the event *S1.3* would be executed first.

In the thread where *S1.2* is executed first, the resource availability variables would be set to $S1 = 0$ and $S2 = 1$. When the *S1.3* event comes off of the future events calendar, the execution condition, $S1 \geq 1$, would be *false* and the *S1.3* event would be ignored and taken off of the calendar. At that point, the *S1.3* event could only be executed when another activity completes and frees up resource 1. In this particular case, the *F1.2* or *F2.4* events would need to be executed.

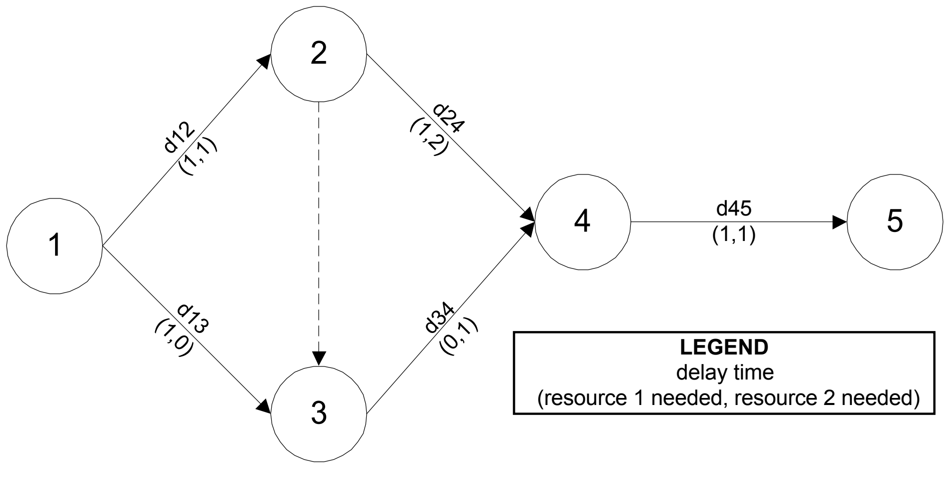


Figure 1: PERT with Resources Example

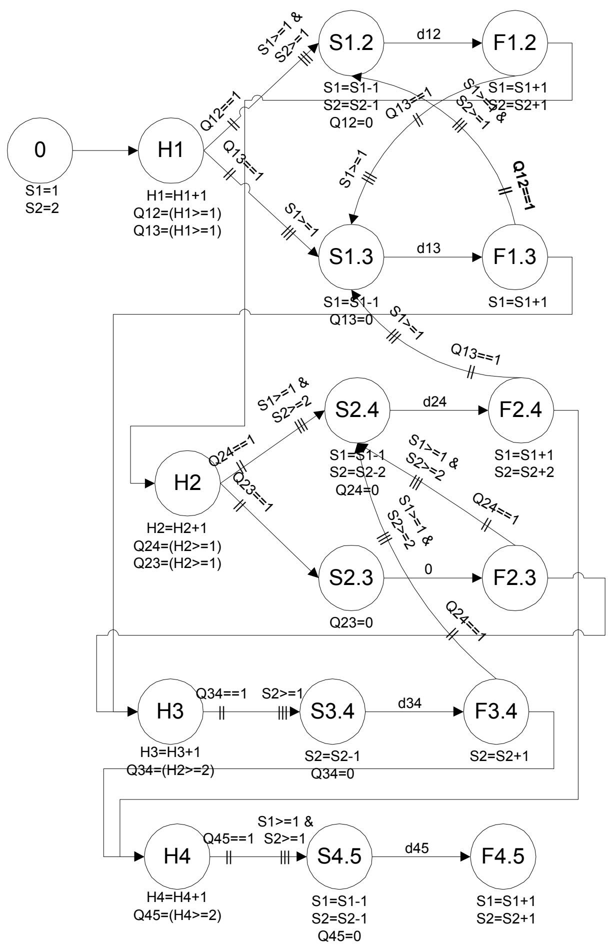


Figure 2: Figure 1 converted to an Event Graph

If any of the *start* events are executed, then they automatically schedule the activity delay in the PERT network. When the corresponding *finish* event is executed, two things occur. First, the succeeding *hit* events are scheduled to occur instantaneously with a high priority. Second, the *finish* event schedules any *start* events that have not occurred and need some of the resources that have just been released by the *finish* event.

Continuing with the thread described above, where the event *S1.2* is executed first, *F1.2* is executed when the activity is complete. At that time, the resource availability variables would be set to $S1 = 1$ and $S2 = 2$. Because *F1.2* has just released the resources that activity *1.3* needs to execute and the variable *Q13* is set to *1*, the event *S1.3* is scheduled to execute. The execution condition, $S1 \geq 1$, would be *true* and *S1.3* would execute and begin activity *1.3*.

The remainder of the event graph network follows the logic described above. Although this logic might seem complex to generate based on a PERT network description, the current implementation of QDES in SMALLTALK automatically takes the PERT network description and generates the equivalent event graph network for any PERT network that can be described. (Ingalls, 1999)

3 PERT WITH RESOURCES EXAMPLES

In Pritsker, et al (1989), the authors use “the repairman” model throughout the book. We have adopted both the constant delay time version of the model and the interval delay time version of the model. The repairman model has 2 repairmen available to perform the tasks outlined in Figure 3.

Activity	Nodes	Delay Time (Min,Expected,Max)	Resources Needed
1. Disassemble power units and instrumentation	1,2	(1,3,5)	2
2. Install new assembly	1,3	(3,6,9)	2
3. Prepare for repair check	1,4	(10,13,19)	1
4. Clean, inspect, and repair power units.	2,5	(3,9,12)	1
5. Calibrate instrumentation	2,3	(1,3,8)	1
6. Check interfaces	3,6	(8,9,16)	1
7. Check assembly	3,4	(4,7,13)	1
8. Assemble and test power units	5,6	(3,6,9)	1
9. Repair check	4,6	(1,3,8)	1

Figure 3: Repairman Model Description

3.1 Example Using Constant Delay Times

In the constant delay time version of this problem, we used the *expected* delay time in Figure 3 for the activity delays. Figure 4 shows the PERT diagram for this model. The output of the simulation would be all of the possible sequences of events and their timing. We certainly would be able to find the minimum makespan time using this type of approach. Although the QSGM can be used in this way, it is not an efficient search algorithm. Algorithms built to find optimal schedules would be more efficient than using QSGM. However, we did find that that minimum makespan is 34 and Figure 5 shows two schedules that meet that minimum makespan. The output of the model showed that 112 of the 360 threads had a completion time of 34 and qualified for minimum makespan.

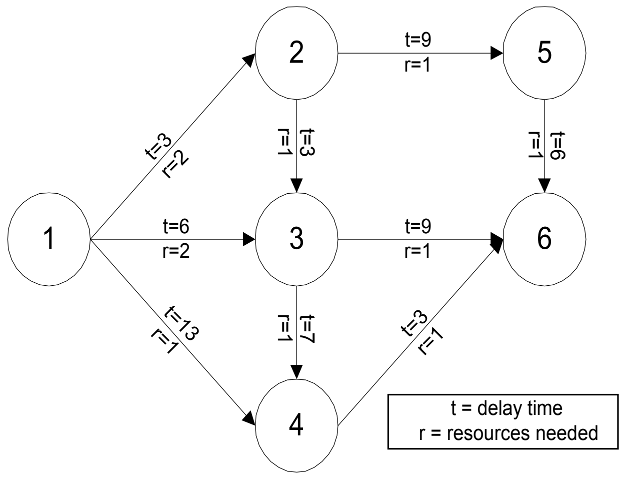


Figure 4: Repairman Model with Constant Activity Times

3.1 Example Using Interval Delay Times

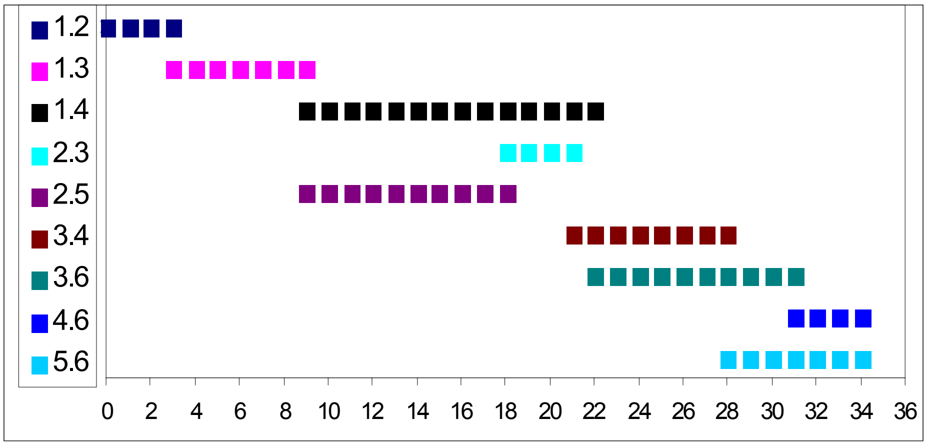
Now let us take the same Repairman Model and apply qualitative time constructs and use those qualitative time constructs as a mechanism for scheduling all these activities "real-time". Figure 6 shows the PERT representation of this model. Notice that the delay times are now intervals in $\Re$.

First of all, the QSGM gives us a new framework for scheduling. In the example, we have two resources that must be allocated to competing activities. In a traditional scheduling algorithm, the delay times would be constants and the sequence of activities would be an optimal answer based on those constant delay times. One of the weaknesses of those types of scheduling algorithms is that the actual delay times often vary from the inputs. This causes a couple of problems. First, the scheduler is not sure if the change in the timing of the activity would effect his overall optimal schedule in any significant way. The only way to assure that the schedule is still valid is to re-run the schedule with the up-to-date information. Second, if the actual delay time changes so much as to invalidate the sequence of the optimal schedule, then the scheduler is forced to reschedule for the remaining activities. Taking advantage of the coverage property of the QSGM, no rescheduling is necessary. First, we will look at some of the raw output of the simulation and explain what information we have at our disposal. Second, we will look at using that information to schedule using *point in time* decision making.

3.1.1 PERT Model Output

As an example of the output generated by the simulation on a given PERT problem, let us look at a thread generated for the Repairman problem in Figure 6. The thread we will look at has a completion time interval that is very wide for this example. In Figure 7, the four columns are the *Event #*, the *Event*, the *Time* in which that event could occur, and the *Input Delay* for each start event. If the *Event* has an *S* designation, then it is the start of a given activity. If the *Event* has an *F* designation, then it is the finish or completion of that activity. The *Input Delay* means that the delay for that activity lies somewhere in the *Input Delay* interval.

What information can we get from this thread? First, we know that if the start and finish event happen in this sequence, the schedule will finish somewhere between time 26 and time 69. For all possible threads in the simulation, the schedule will finish somewhere between time 19 and time 75. We also know the time window for this and any other sequence that has a common set of beginning events. For example, this thread's first four events are *S1.2, F1.2, S1.3,* and *F1.3.* When the *F1.3* event occurs, the time window for that event is *(4,14)*. That time window will be true for any other thread that shares this same event sequence for the first four events. This leads to another observation. When choosing which activity gets a free resource, such as happens at event 5, the user can look at the threads with a common starting sequence, such as *S1.2, F1.2, S1.3, F1.3,* and *S2.3*, and compare them with threads with the common starting sequence of *S1.2, F1.2, S1.3, F1.3,* and *S1.4,* and the threads with the starting sequence *S1.2, F1.2, S1.3, F1.3,* and *S2.5*. In this instance, *S2.3*, *S1.4*, and *S2.5* are all activities that could be started at this time. Using the information already generated by the simulation, we can make decisions based on the future of any one of these three alternatives. We use this idea in the next section.



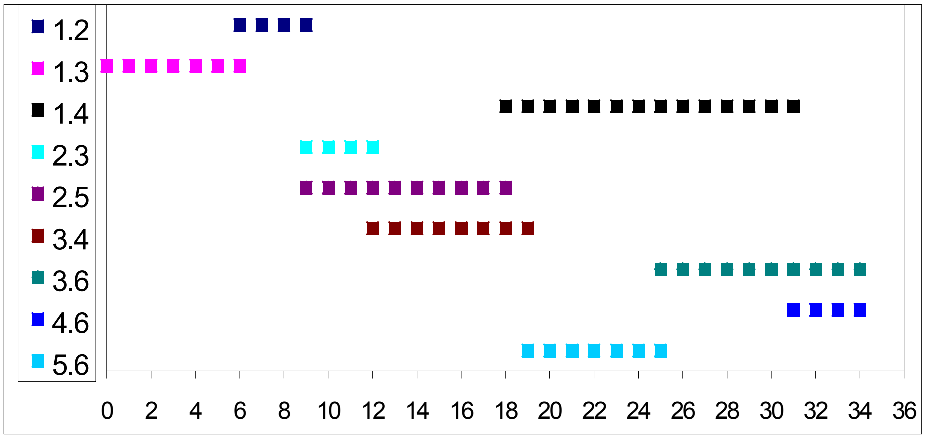


Figure 5: Two Minimum Makespan Schedules

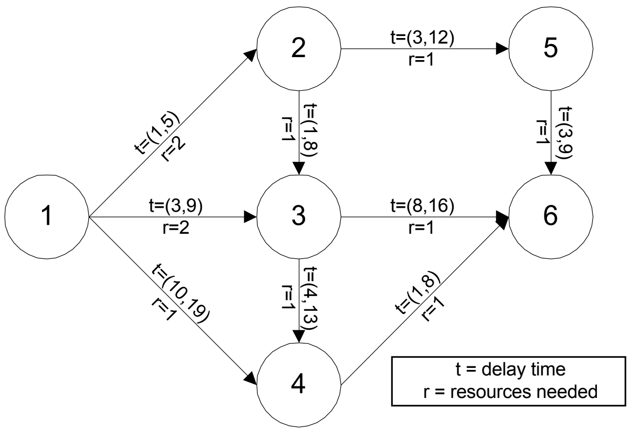


Figure 6: Repairman Model with Interval Activity Times

Event #	Event	Time	Input Delay
1	S1.2	[0.0,0]	(1,5)
2	F1.2	(1.0,5.0)	
3	S1.3	(1.0,5.0)	(3,9)
4	F1.3	(4.0,14.0)	
5	S2.3	(4.0,14.0)	(1,8)
6	S2.5	(4.0,14.0)	(3,12)
7	F2.3	(5.0,22.0)	
8	S3.4	(5.0,22.0)	(4,13)
9	F2.5	(7.0,26.0)	
10	S3.6	(7.0,26.0)	(8,16)
11	F3.4	(9.0,35.0)	
12	S5.6	(9.0,35.0)	(3,9)
13	F3.6	(15.0,42.0)	
14	S1.4	(15.0,42.0)	(10,19)
15	F5.6	(15.0,44.0)	
16	F1.4	(25.0,61.0)	
17	S4.6	(25.0,61.0)	(1,8)
18	F4.6	(26.0,69.0)	

Figure 7: PERT Thread Output Example

This also leads to another topic of interest, and that is monitoring the results of the schedule and then making adjustments to the simulation output. Again, let us assume that the first four events to occur are *S1.2, F1.2, S1.3,* and *F1.3.* At that point, we have just eliminated many of the threads that the simulation generated. With the threads remaining, we know that the latest possible to finish the schedule is now *69*. However, we also know that this event sequence has happened at a point in time. As an example, let us assume that the event sequence happened at time *10*. This means that instead of the window of *(4,14)*, we could substitute an interval of *[10,10]* as the time for the event. Although we have not worked through the mechanics of the calculations, we know that this information would adjust the upper and/or lower bounds of the remaining events in all of the threads that have this starting sequence. It might also invalidate some other threads will the same starting sequence.

3.1.2 Point in Time Decision Making

With the output of one model run, all possible sequencing outcomes are represented and that information can be used to make *point in time* decisions that take into account all the uncertainty that could happen in the future. For example, our first scheduling decision is whether to start activity 1-2, activity 1-3, or activity 1-4. Based on a user-defined scheduling criteria (see below), it is determined that activity 1-2 is the best activity to start at this time based on the possible outcomes in the future. We have no need to make any decisions on the sequencing of other activities at this point in time.

The schedule is made up of the sequence of *start* and *finish* events for each of the activities in the PERT diagram. Some of these events would occur based on a *decision*. A decision would occur when there are multiple activities that could start their process based on available resources. The scheduler would decide which event would occur next. In the QSGM output, there will be threads that simulate the outcome of each of the possible decisions that the scheduler could make. The sequence of other events occur in an *uncertain* order, which is to say that one event could occur before another based on the actual timing of the delay. These *uncertain* events are outside the control of the scheduler. Any *finish* events in a NOS would fall into this category of events. Still, other events in the sequence are *fixed*, meaning that in the thread under observation, there is no alternative for that particular event in the sequence. A simple example of a fixed event would be when only one activity is remaining and there are available resources. The *start* event for the last activity is the only event that could occur at that time. Any schedule is the sequence of the decision, uncertain, and fixed events. The scheduler only has to make decisions when a decision is necessary to continue processing the schedule.

In our research, we decided to evaluate two different scheduling criteria. The first is the *smallest upper bound completion time* criterion. At any decision point, we chose the event that guarantees the smallest upper bound on the completion time. The other scheduling criteria is the *average midpoint* completion time criterion. For all of the threads that follow any given decision, calculate the *average midpoint* of all of those threads. The decision that yields the smallest *average midpoint* is the best decision. For illustration purposes, we will show the detailed output of the *average midpoint* method in this paper.

In our example, we have to make a decision if *S1.2*, *S1.3,* or *S1.4* is the first event in our schedule. The *average midpoint* scores for these three alternatives are 37.54, 37.56, and 45.32, respectively. So the scheduler would choose *S1.2* to be the first event in the sequence using the average midpoint criteria. The *smallest upper bound completion time* scores for these three alternatives are 69, 75, and 75, respectively. Again, *S1.2* would be chosen as the first event in the schedule. This guarantees that, regardless of any other decisions or unexpected event sequences, the worst completion time will be 69.

It should be said that the criteria that we have chosen for generating the schedules are purely for illustration purposes. They may or may not be the best way to make the point-in-time decisions that are necessary. "Optimal" decision criteria can be a topic for further research.

In Figure 8, we have the schedule generated by the QSGM methodology on the Repairman problem using the *average midpoint* completion time. There are 5 designations in the schedule. First are alternatives. Alternatives require a decision. Alternatives are marked with black background and white type. After a decision is made, that decision is designated in bold. Uncertain events are italic. Fixed events are underlined. A complete schedule, based on the decisions made, the uncertain and fixed events, has a gray background. The scoring used and the final interval of a thread are in the far right-hand columns of the tables.

What can we say about these "schedules"? Certainly, each is a schedule, in that any one of them can be followed in order to complete a series of tasks. All schedules are robust, because as long as no activity times fall outside the activity time intervals designated, all of the possible schedules have been taken into account in the decision.

Our current problem was run on a 200MHz Pentium II machine in approximately 4½ minutes. We have not run industrial size problems and would have to take into account the scalability of our approach. At this point, we have not taken advantage of the highly parallel structure of QSCM. This could yield significant speed improvements and directly address the scalability problem.

Now let us take into account how this might be used. Because the output data is already generated, the point-in-time decision making of the scheduler is now a database query instead of running the algorithm over again. The quality of the decision can be much higher since it is taking into account the full range of possible events that could happen to impact the schedule in the future. Without imposing random assumptions on the intervals, we can create a robust schedule.

Alternative	**Decision**	*Uncertain*	Fixed	Complete Schedule	

Event																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	AM	Time
S1.2																		37.54	
S1.3																		37.56	
S1.4																		45.32	
S1.2	F1.2	**S1.3**																37.09	
S1.2	F1.2	**S1.4**																39.57	
S1.2	F1.2	**S2.3**																39.68	
S1.2	F1.2	**S2.5**																38.30	
S1.2	F1.2	**S1.3**	F1.3	**S1.4**														37.12	
S1.2	F1.2	**S1.3**	F1.3	**S2.3**														37.11	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**														37.05	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**													37.00	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S2.3**													37.06	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**									35.41	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.6**									35.98	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F2.5*	**S3.6**							36.09	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F2.5*	**S5.6**							36.18	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F2.5*	**S3.6**	*F3.4*	**S4.6**					36.13	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F2.5*	**S3.6**	*F3.4*	**S5.6**					35.50	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F2.5*	**S3.6**	*F3.4*	**S5.6**	*F3.6*	S4.6	*F4.6*	F5.6	36.00	24,48
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F2.5*	**S3.6**	*F3.4*	**S5.6**	*F3.6*	S4.6	*F5.6*	F4.6	37.00	24,50
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F2.5*	**S3.6**	*F3.6*	S5.6	*F3.4*	S4.6	*F4.6*	F5.6	37.00	26,48
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F2.5*	**S3.6**	*F3.6*	S5.6	*F3.4*	S4.6	*F5.6*	F4.6	36.50	26,47
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F2.5*	**S3.6**	*F3.6*	S5.6	*F5.6*	F3.4	S4.6	F4.6	37.00	27,47
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F3.4*	**S3.6**							34.31	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F3.4*	**S4.6**							34.70	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F3.4*	**S4.6**	*F2.5*	**S3.6**					34.50	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F3.4*	**S4.6**	*F2.5*	**S5.6**					35.63	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F3.4*	**S4.6**	*F2.5*	**S3.6**	*F3.6*	S5.6	*F4.6*	F5.6	36.50	30,43
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F3.4*	**S4.6**	*F2.5*	**S3.6**	*F3.6*	S5.6	*F5.6*	F4.6	32.00	30,34
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F3.4*	**S4.6**	*F2.5*	**S3.6**	*F4.6*	S5.6	*F3.6*	F5.6	35.00	27,43
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F3.4*	**S4.6**	*F2.5*	**S3.6**	*F4.6*	S5.6	*F5.6*	F3.6	34.50	27,42
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F3.4*	**S4.6**	*F4.6*	S3.6	F2.5	S5.6	*F3.6*	F5.6	31.50	28,35
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.3*	**S3.4**	*F3.4*	**S4.6**	*F4.6*	S3.6	F2.5	S5.6	*F5.6*	F3.6	35.00	28,42
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.4**							36.77	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.6**							37.50	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.4**	*F3.4*	**S3.6**					35.25	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.4**	*F3.4*	**S4.6**					36.25	
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.4**	*F3.4*	**S3.6**	*F3.6*	S4.6	*F4.6*	F5.6	31.50	28,35
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.4**	*F3.4*	**S3.6**	*F3.6*	S4.6	*F5.6*	F4.6	35.50	28,43
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.4**	*F3.4*	**S3.6**	*F5.6*	S4.6	*F3.6*	F4.6	35.00	27,43
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.4**	*F3.4*	**S3.6**	*F5.6*	S4.6	*F4.6*	F3.6	39.00	27,51
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.4**	*F5.6*	S3.6	*F3.4*	S4.6	*F3.6*	F4.6	40.00	25,55
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.4**	*F5.6*	S3.6	*F3.4*	S4.6	*F4.6*	F3.6	38.00	25,51
S1.2	F1.2	**S1.3**	F1.3	**S2.5**	**S1.4**	*F1.4*	S2.3	*F2.5*	S5.6	*F2.3*	**S3.4**	*F5.6*	S3.6	*F3.6*	F3.4	S4.6	F4.6	40.50	26,55

S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F1.4	S2.3	F2.5	S5.6	F5.6	F2.3	S3.4					39.33	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F1.4	S2.3	F2.5	S5.6	F5.6	F2.3	S3.6					39.33	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F1.4	S2.3	F2.5	S5.6	F5.6	F2.3	S3.4	S3.6	F3.4	S4.6	F3.6	F4.6	40.00	25,55
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F1.4	S2.3	F2.5	S5.6	F5.6	F2.3	S3.4	S3.6	F3.4	S4.6	F4.6	F3.6	37.50	25,50
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F1.4	S2.3	F2.5	S5.6	F5.6	F2.3	S3.4	S3.6	F3.6	F3.4	S4.6	F4.6	40.50	26,55
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3										37.26	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S5.6										37.96	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.4						39.27	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.6						39.50	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.4	F3.4	S3.6				38.25	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.4	F3.4	S4.6				39.25	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.4	F3.4	S3.6	F3.6	S4.6	F4.6	F5.6	34.50	27,42
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.4	F3.4	S3.6	F3.6	S4.6	F5.6	F4.6	38.50	27,50
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.4	F3.4	S3.6	F5.6	S4.6	F3.6	F4.6	38.00	26,50
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.4	F3.4	S3.6	F5.6	S4.6	F4.6	F3.6	42.00	26,58
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.4	F5.6	S3.6	F3.4	S4.6	F3.6	F4.6	40.00	25,55
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.4	F5.6	S3.6	F3.4	S4.6	F4.6	F3.6	41.50	25,58
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F2.3	S3.4	F5.6	S3.6	F3.6	F3.4	S4.6	F4.6	40.50	26,55
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F5.6	F2.3	S3.4					39.33	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F5.6	F2.3	S3.6					39.33	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F5.6	F2.3	S3.4	S3.6	F3.4	S4.6	F3.6	F4.6	40.00	25,55
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F5.6	F2.3	S3.4	S3.6	F3.4	S4.6	F4.6	F3.6	37.50	25,50
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F1.4	S5.6	F5.6	F2.3	S3.4	S3.6	F3.6	F3.4	S4.6	F4.6	40.50	26,55
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.4								36.48	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6								36.39	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S5.6								37.64	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S3.4						37.73	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S5.6						37.75	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S3.4	F3.4	S4.6				38.13	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S3.4	F3.4	S5.6				38.00	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S3.4	F3.4	S5.6	F3.6	S4.6	F4.6	F5.6	38.00	21,55
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S3.4	F3.4	S5.6	F3.6	S4.6	F5.6	F4.6	39.00	21,57
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S3.4	F3.4	S5.6	F5.6	S4.6	F3.6	F4.6	39.50	22,57
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S3.4	F3.4	S5.6	F5.6	S4.6	F4.6	F3.6	35.50	22,49
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S3.4	F3.6	S5.6	F3.4	S4.6	F4.6	F5.6	37.00	19,55
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S3.4	F3.6	S5.6	F3.4	S4.6	F5.6	F4.6	36.50	19,54
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F1.4	S3.4	F3.6	S5.6	F5.6	F3.4	S4.6	F4.6	37.00	20,54
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F3.6	S3.4						33.58	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F3.6	S5.6						35.20	
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F3.6	S3.4	F1.4	S5.6	F3.4	S4.6	F4.6	F5.6	31.50	21,42
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F3.6	S3.4	F1.4	S5.6	F3.4	S4.6	F5.6	F4.6	35.50	21,50
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F3.6	S3.4	F1.4	S5.6	F5.6	F3.4	S4.6	F4.6	37.50	21,54
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F3.6	S3.4	F3.4	S5.6	F1.4	S4.6	F4.6	F5.6	32.50	23,42
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F3.6	S3.4	F3.4	S5.6	F1.4	S4.6	F5.6	F4.6	32.00	23,41
S1.2	F1.2	S1.3	F1.3	S2.5	S1.4	F2.5	S2.3	F2.3	S3.6	F3.6	S3.4	F3.4	S5.6	F5.6	F1.4	S4.6	F4.6	32.50	24,41

Figure 8: Qualitative Repairman Schedule Using *Average Midpoint* Criteria

Looking at Figure 8, we can see that the schedule chooses the decision that has the lowest average midpoint at each decision point, as expected. How can we say that these schedules compare to each other? The *smallest upper bound completion time* schedule had an average maximum of 50.1 for the completion time, and an average minimum of the completion time of 24.5. Also, the latest possible completion time is 62. The *average midpoint* schedule, in Figure 8, has an average maximum of 48.9 for the completion time and an average minimum of the completion time of 24.9. The latest possible completion time for this schedule is 58. It would seem from this output that the desired effect of the *smallest upper bound completion time* criteria was actually better achieved using the *average midpoint* criteria. The data shows that a criteria that judges an extreme, such as the *smallest upper bound completion time*, takes into account only one thread and ignores all other relevant threads. However, the *average midpoint* criteria takes into account all of the threads effected in some way (in this case by including them in the average). A logical conclusion would be that using more information may be better in determining a good schedule. Again, we would stress the heuristic nature of the two scheduling criteria that we have chosen. More research should be done in order to determine which scheduling criteria is best.

4 CONCLUSION AND FUTURE RESEARCH

The PERT with resources problem is a general scheduling problem which network could be converted to a simulation graph without cycles. Taking advantage of the coverage property, the QSGM characterizes all possible schedules in its output. This information can be used to monitor and control a schedule on an ongoing basis. It can also be used to make good point-in-time scheduling decisions based on future event sequences, regardless of how randomness might effect the event sequence. This is a significant change in the way that scheduling can be done and should be a topic of significant future research.

This particular application area seems rich for future research. One of the areas for future research is how to incorporate the knowledge gained through history into the schedules that have already been generated. This would give the scheduler a way to monitor the progress of the schedule without re-running the simulation. Also, there is research that needs to be done in the area of scalability. First, the general problem structure is highly parallel and could be decomposed to run on a parallel processing computer. Second, one could make constant those activities that are less important and use intervals for the more critical activities. This would reduce the number of threads generated by the simulation. Also, any thread reduction research on the general GSCM problem would directly benefit the PERT application.

Using the qualitative simulation in the area of PERT scheduling with resources is clearly a new way to approach this problem. It breaks the assumption of constant or even probabilistic activity delay times and creates a database of schedules that take into account uncertainty.

REFERENCES

Ingalls, R. G., D. J. Morrice, A. B. Whinston, 2000. Implementation of Temporal Intervals in Qualitative Simulation Graphs (To appear in *ACM Transactions on Modeling and Computer Simulation*).

Ingalls, R. G., D. J. Morrice, A. B. Whinston, and E. Yücesan, 2000. Edge Execution Conditions: A Formalization Of Event Cancellation In Simulation Graphs (Submitted for publication)

Ingalls, R. G., 1999. Qualitative Simulation Graph Methodology and Implementation, Ph.D. Dissertation, The University of Texas at Austin, Austin, TX.

Ingalls, R. G., D. J. Morrice, and A. B. Whinston. 1996. Eliminating Canceling Edges from the Simulation Graph Model Methodology. *Proceedings of the 1996 Winter Simulation Conference*, ed. J.M. Charnes, D.J. Morrice, D.T. Brunner, and J.J. Swain, 825-832.

Pristker, A. Alan B., C. Elliot Sigal, R.D. Jack Hammesfahr, *SLAM II: Network Models For Decision Support*, Englewood Cliffs 1989.

AUTHOR BIOGRAPHIES

RICKI G. INGALLS is the Manager of Supply Chain Modeling in the Order Management and Logistics organization at Compaq Computer Corporation. In the fall of 2000, he will join the faculty in the School of Industrial Engineering and Management at Oklahoma State University as an Associate Professor. He has been involved in the application and development of operational modeling tools and techniques in the electronics industry for over 16 years. He has a B.S. in Mathematics from East Texas Baptist College, a M.S. in Industrial Engineering from Texas A&M University and a Ph.D. in Management Science from the University of Texas at Austin. Other than Compaq, He also has held management and staff positions at SEMATECH, General Electric and Motorola. His research interests include the development and application of qualitative discrete-event simulation and the development and application of large-scale models to supply chain systems.

DOUGLAS J. MORRICE is an Associate Professor in the MSIS Department at The University of Texas at Austin. He has a BA, Honours in Operations Research from Carleton University in Ottawa, Canada. He holds an M.S. and a Ph.D. in Operations Research and Industrial Engineering from Cornell University. His research interests include operations simulation design, modeling, and analysis. Dr. Morrice is a member INFORMS. He served as the Secretary for the INFORMS College on Simulation (1994-1996) and was Co-Editor of the Proceedings of the 1996 Winter Simulation Conference.