Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

SEQUENCE STEP ALGORITHM FOR CONTINUOUS RESOURCE UTILIZATION IN PROBABILISTIC REPETITIVE PROJECTS

Photios G. Ioannou
Chachrist Srisuwanrat

Civil & Environmental Engineering Department
2350 G.G. Brown
University of Michigan
Ann Arbor, MI 48109-2125, U.S.A.

ABSTRACT

The sequence step algorithm addresses for the first time the problem of scheduling repetitive projects with probabilistic activity durations while maintaining continuous resource utilization. This algorithm is based on generalized concepts that can be implemented in most general-purpose simulation systems. The algorithm is presented in detail and is applied to an example project with 7 activities and 4 repetitive units using a simulation model developed in Stroboscope, an activity-based simulation system. Numerical and graphical results help explain the algorithm and provide insight into the underlying tradeoff problem between reducing the expected crew idle time and increasing the expected project duration.

1 INTRODUCTION

Multiunit projects are commonly found in construction where identical or similar units require repetitive work from unit to unit. Examples of multiunit projects are multistory buildings, housing projects, highways, and tunneling projects. In these projects, the same activities are repeated from unit to unit by the same crews. For example, in a multistory building, one crew installs interior partition studs from floor to floor, while another crew follows and installs drywall. The installation of drywall in the 4th floor, for example, requires that the studwork for the 4th floor has been finished (a technological constraint) and also that the drywall crew has finished its work on the 3rd floor (a resource constraint). Of particular interest in scheduling repetitive multiunit projects is the ability to keep crews working continuously without interruption. Otherwise, crews experience periods of idle time where they receive pay without producing output. Thus, the uninterrupted (i.e., continuous) utilization of resources is of prime importance in scheduling repetitive work.

Figure 1 illustrates the problem by showing production diagrams for three activities, A, B, and C, that are repeated over three identical units (e.g., floors). These activities must be performed sequentially, one after the other. The work in each activity is performed by a separate crew. Figure 1a is a production diagram where activities are allowed to start as early as possible as is typical in CPM. Figure 1b is a production diagram where activities are deliberately delayed using the Repetitive Scheduling Method (RSM) to ensure the uninterrupted utilization of resources.

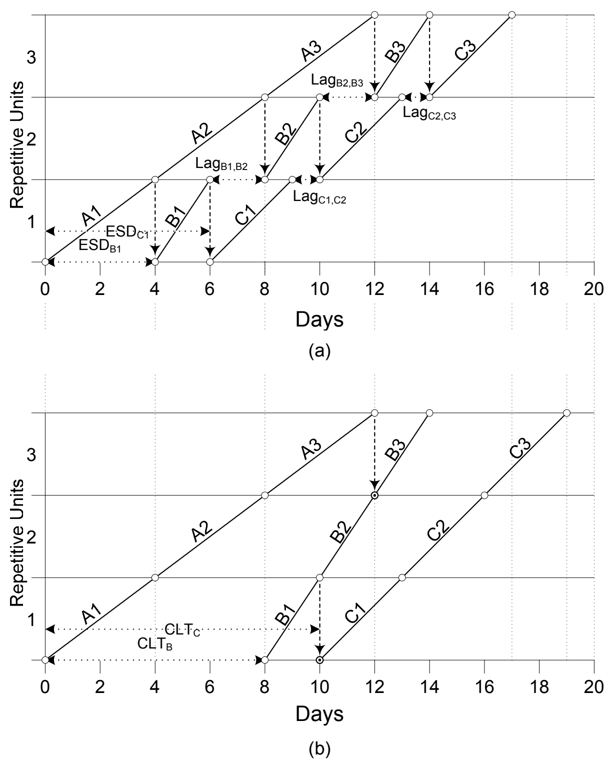

Figure 1: CPM vs. RSM Scheduling

As shown in Figure 1a, when activities A, B, and C, start on their early start dates (ESD), their differences in unit production rates (slopes) result in lags ($\text{Lag}_{I,J}$) that represent idle unproductive time for the respective crews. In practice, the crews for these activities are paid from the point they start work in the first repetitive unit, to the completion of the activity in the last unit. Thus, crew B is paid for 10 days (even though it works for only 6 days) and crew C is paid for 11 days (even though it works for only 9 days).

As shown in Figure 1b, delaying the start of activities B and C by CLT_B and CLT_C, respectively, ensures that the respective crews work continuously without interruption. The tradeoff for saving 6 crew-days over Figure 1a is that the project duration has increased from 17 to 19 days. In most practical cases, the savings in crew costs far outweigh the slight increase in project duration.

Over the years, several methods have been proposed to solve the resource continuity problem with similar approaches that either postpone the start dates of activities or alter the number of resources (crew sizes) to balance activity production rates. Harris and Ioannou (1998) unified these methods into RSM and showed how to provide solutions for repetitive projects with *deterministic* activity durations using a graphical approach.

The resource continuity problem when activity durations are *probabilistic*, however, is considerably more difficult and has not been addressed before. Probabilistic activity durations are quite common in repetitive work and stem from variability in *crew production rates* and uncertainty in the *amount of work* to be performed by each activity in each repetitive unit. Probabilistic activity durations introduce a hard-to-quantify tradeoff between activity start lead-times, the probability distributions for work discontinuities and the probability distribution for project duration.

Tackling this problem using simulation requires a fine balance between the nature of resource-based models (where an activity starts immediately if the required resources are available) and the idea of deliberately postponing activity start dates (where activities do not start before a specified lead-time has elapsed, even though resources may be available). Simulation models that rely only on precedence constraints operate in the same manner as the critical path method in that they allow activities to start as early as possible. Thus, they do not guarantee continuity in resource usage. Continuous resource utilization can only be achieved by deliberately holding the resources back by appropriately chosen lead times to make the work continuous without overly extending project duration (Ioannou and Likhitruangsilp 2005).

In this paper we present a new methodology, the *sequence step algorithm*, that allows the development of resource-based simulation models for scheduling repetitive projects with probabilistic activity durations while maintaining continuous resource utilization. The sequence step algorithm is based on generalized concepts that can be applied to most general-purpose discrete-event simulation systems. The example project presented here has been developed using Stroboscope, an activity- and resource-based simulation system (Martinez 1996). Numerical and graphical results help explain the algorithm and provide insight into the underlying tradeoff problem.

2 SEQUENCE STEPS

In project scheduling, a precedence diagram (also called an activity-on-node network) uses nodes (circles or rectangles) to model activities (tasks or work). Nodes are connected with lines (links) that represent precedence relationships between activities. An example precedence diagram with seven activities appears in Figure 2.

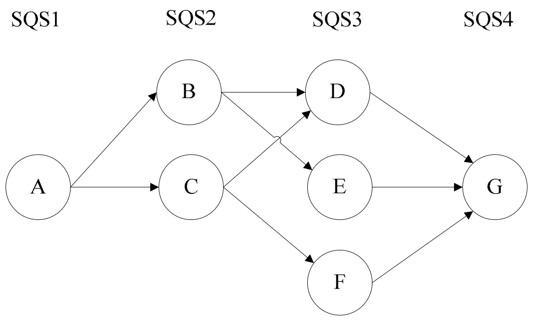

Figure 2: Precedence Diagram for Single Unit

An important property of activity-on-node networks is that they can be drawn in an organized manner using visual columns so that all precedence links have a clear left-to-right direction (Figure 2). This way, an activity to the left of a link represents a predecessor, while an activity on the right represents a successor. For example, the activities in Figure 2 belong to four visual columns, called *sequence steps*, that are indicated by SQS1, SQS2, SQS3, and SQS4. Every activity in a precedence diagram belongs to a *particular* sequence step, which is formally defined to be the *left-most* visual column in which the activity may be drawn and still maintain left-to-right precedence relationships within the network. Another way to find the sequence step for each activity is to assume that all activities have a duration of 1 and to perform the traditional CPM forward pass. The sequence step for each activity is then given by the resulting early finish dates.

The links in an activity-on-node network that has been organized by sequence step do not require arrow heads. It is implicitly understood that all links have a left-to-right direction. In fact, most modern scheduling software draw precedence diagrams by sequence step and draw links as plain lines without arrowheads.

An important property of precedence diagrams drawn by sequence step is that the traditional CPM scheduling calculations can indeed be performed by sequence step:

i.e., left-to-right for the forward pass and right-to-left for the backward pass. Calculations for activities in the same sequence step can be performed in any order (e.g., top to bottom). As explained below, this property is at the heart of the proposed *sequence step algorithm*.

3 CREW LEAD TIME AND CREW IDLE TIME

Crew lead time (CLT) is the *chosen* lead time by which the first start date of the corresponding activity is delayed. Crew idle time (CIT) is the *resulting* time a crew does not perform work during its total period of employment.

In the beginning of the proposed *sequence step algorithm*, CLT = 0, i.e., all crews are assumed available for work at time 0. Thus, the initial CIT is the early start date (e.g., ESD_{B1}) plus the sum of idle times from unit to unit (e.g., $Lag_{B1,B2} + Lag_{B2,B3}$). As explained in the next section, the reason for this assumption is computational efficiency.

As shown in Figure 1b, in order to eliminate crew idle time (CIT), it is necessary to delay the start of activities B and C by crew lead times CLT_B and CLT_C (as measured from the start of the project), respectively. Thus, to eliminate idle time for an activity, the crew lead time must equal or exceed its crew idle time (CLT >= CIT).

4 SIMULATION MODEL COMPONENTS

The implementation of the sequence step algorithm in discrete-event simulation requires that a simulation model be able to distinguish between a crew being employed (on site) and unemployed (off site), to postpone an activity's start date (CLT), and to track crew idle time (CIT).

Figure 3 is an example of a simulation sub-network that represents the generic activity X using Stroboscope modeling elements that provides these capabilities.

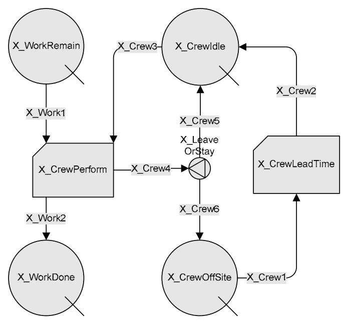

Figure 3: Simulation Sub-Network for Generic Activity X

This sub-network involves four queues, two combi activities, and one fork, through which travel two types of resources. The first resource type is X_Work which represents the repetitive units (e.g., floors) that activity X must perform. For example, if activity X represented laying drywall in a 10-story building, queue X_WorkRemain would be initialized with 10 units of resource X_Work at the start of simulation. As simulation progresses, the status of a particular work resource unit (not started, in process, or competed) would depend on it current position in the sub-network as explained below.

The second type of resource is X_Crew. As its name implies, it represents the crew(s) available to perform activity X. For example, if activity X represented laying drywall in a 10-story building, X_Crew would represent the drywall crew. In most cases there would be only one unit of X_Crew which at the start of simulation would be initialized in queue X_CrewOffSite. Resource X_Crew is used to control the first start date of activity X through CLT, and to calculate crew idle time, CIT.

Queue X_WorkRemain contains the remaining units of work (e.g., floors) where activity X (e.g., drywall installation) still needs to be performed. At the beginning of simulation, X_WorkRemain is initialized with all repetitive units to be performed (e.g., 10 units of X_Work).

Queue X_WorkDone contains the repetitive units of work (e.g., floors) where activity X (e.g., drywall installation) has already been performed. At the beginning of simulation, X_WorkDone is empty, while at the end of simulation it contains all available units of resource X_Work for the project (e.g., 10 units of X_Work).

Combi X_CrewPerform is a conditional activity that is at the heart of the sub-network. It represents the performance of activity X in a particular repetitive unit X_Work. Every time this activity starts, it draws one resource X_Work from queue X_WorkRemain, keeps it for the duration of the activity and when it finishes it releases resource X_Work to queue X_WorkDone. Thus, each resource X_Work (e.g., floor) moves from queue X_WorkRemain (not started), to combi X_CrewPerfrom (in process), to queue X_WorkDone (completed).

Figure 4 shows the sub-networks for activity A and its successor activity B which are part of the example simulation model discussed in detail below. These two sub-networks are not connected by any links. The precedence relationship (link) between A and B is implemented through the *semaphore* (i.e., the *logical* start control) of the successor activity, B. Thus, the semaphore for combi activity B_CrewPerform prevents it from starting until its predecessor combi A_CrewPerform in the same unit has been completed. In particular, the logical expression for the semaphore of activity B_CrewPerform compares the number of completed units in B_WorkDone to the number of completed units in A_WorkDone, as follows:

```
B_WorkDone.CurCount < A_WorkDone.CurCount
```

CurCount is a predefined Stroboscope variable that returns the number of resources currently residing in the referred queue. Thus, the above statement is *true* and B_CrewPerform can start only if queue A_WorkDone contains more resources (finished work units) than queue B_WorkDone.

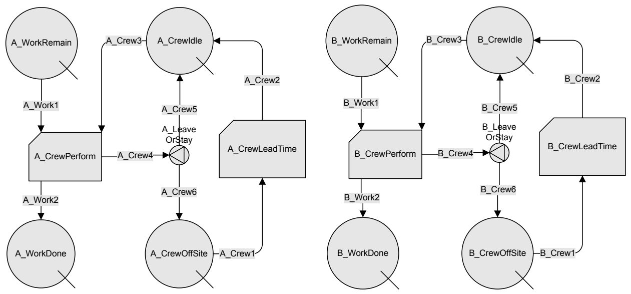

Figure 4: Sub-networks for Activities A and B

If the semaphore condition is true, *and* queues B_WorkRemain and B_CrewIdle are not empty, then B_CrewPerform will start and draw one resource from each of these queues. The duration of B_CrewPerform is the time required to finish a particular work unit and is a function of the amount of work in that unit and the production rate of the crew (both of which could be random).

X_CrewOffSite is a queue where resource X_Crew resides when not on site (not hired). At the start of simulation, resource X_Crew is initialized into this queue. After all the work is completed, resource X_Crew returns back to this queue (is laid off).

X_CrewIdle is a queue where resource X_Crew waits while it is on site (hired), but not performing any work. For example, resource B_Crew is idle and spends time in B_CrewIdle whenever activity B_CrewPerform cannot start because its predecessor activity A_CrewPerform (in the same repetitive unit) has not finished yet. The total time that resource X_Crew spends in this queue is the crew idle time (CIT). In hindsight, it is also the crew lead time (CLT), i.e., the duration of X_CrewLeadTime that should have been used to delay the first instance of X_CrewPerform to eliminate idle time. CIT is given by:

```
X_CrewIdle.AveWait*X_CrewIdle.TotCount
```

AveWait is the average visit time spent by resources in the referred queue. TotCount is the number of times that resources entered the queue (i.e., the total number of visits). Thus, the product of these two variables is the sum of the time durations of all resource visits in the queue which is equal to the crew idle time (CIT) for the activity. The sum of time each resource X_Crew spends in queue X_CrewIdle and combi X_CrewPerform represents the total time the resource is on site (hired).

The duration of activity X_CrewLeadTime is used to control the lead time for the arrival of resource X_Crew to the site and thus to delay the first start date of the corresponding activity X_CrewPerform. At the start of the sequence step algorithm, the duration of X_CrewLeadTime (CLT_X) for all activities X is set to zero. Thus, all crews are assumed to be available for work at the beginning of the project and the time spent by resource X_Crew in queue X_CrewIdle (CIT) includes not just the lags (idle time) between units but also the time interval from project start to the first start of activity X_CrewPerform.

As the sequence step algorithm advances from one sequence step to the next, it becomes possible to assign the final durations of X_CrewLeadTime (CLT) for certain activities. The details of this procedure are explained in detail in the sequence step algorithm section below. At this point, it suffices to note that the total time spent by resource X_Crew in combi X_CrewLeadTime and queue X_CrewOffSite represents its period of unemployment.

Fork X_LeaveOrStay is a decision point that determines whether to send resource X_Crew to X_CrewIdle (crew is still employed on site) or to send X_Crew to X_CrewOffSite (crew is unemployed). The outcome of this decision depends on whether there is any remaining work left. If queue X_WorkRemain contains any X_Work units (i.e., X_WorkRemain.CurCount is greater than zero), fork X_LeaveOrStay will send resource X_Crew back to queue X_CrewIdle. On the other hand, if there is no X_Work in X_WorkRemain, indicating that all work is completed, fork X_LeaveOrStay will send X_Crew to queue X_CrewOffSite.

The above simulation modeling elements provide the means for determining and controlling crew lead time (CLT), the period under employment, crew idle time (CIT), productive time, as well as the number of completed work units and the number of remaining work units. The suggested network modeling format can be applied to both deterministic or stochastic scheduling problems, to typical or non-typical repetitive projects, and any combinations thereof.

5 SEQUENCE STEP ALGORITHM

The purpose of the sequence step algorithm is to schedule repetitive projects, where activity durations in each unit may vary due to resource productivity or differences in work amounts among units, so that crews may work continuously without interruption. The algorithm consists of three general steps.

The first step is to simulate the network and collect crew idle times (CIT) for each activity in each project replication. After performing a number of replications, the collected CIT samples are arranged into histogram-like intervals based on relative frequency. For example, in sequence step 1, after a total of 10,000 simulation replications, we can arrange the 10,000 crew idle times for activity B into intervals of 5 days: 0-5, 5-10, 10-15, and so on.

In the second step, we select a desired confidence level (cumulative probability) for the crew idle times collected in the first step and we assign the corresponding time value to be the duration of X_CrewLeadTime (CLT_X) . For example, if for activity B 9,427 crew idle times out of 10,000 were less than 55 days, then assigning 55 to be the duration of B_CrewLeadTime ($CLT_B = 55$) should give a confidence of 94% that crew B should be able to work continuously.

As pointed out earlier, at the beginning of the algorithm the duration of X_CrewLeadTime was set to zero for all activities. As we finish a particular sequence step we can assign the duration of X_CrewLeadTime for activities in the next sequence step. This duration (CLT) will remain constant through the end of the algorithm. For example, at the end of 10,000 project replications for sequence step 1, the duration of B_CrewLeadTime (CLT_B) can be set to 55 days with 94% confidence. This duration will not change for the rest of the algorithm.

In the third step of the algorithm we reset the simulation model and clear all previously collected crew idle time (CIT) statistics for all activities. Using the already assigned X_CrewLeadTime durations (CLT) for all activities in previous sequence steps, we move to the next sequence step and repeat the first and second algorithm steps until we reach the last sequence step.

Figure 5 is the flowchart of the sequence step algorithm. As shown in the flowchart, the algorithm contains two nested loops. The inner loop is the project replication loop while the outer loop is the sequence step loop. The inner loop represents step one, where the outer loop represents steps two and three. nSQSs and nReps are the total number of sequence steps and the total number of replications. SQS and Rep are the current sequence step and replication.

By repeating these three algorithm steps, we collect samples of crew idle times (CIT) that reflect the already chosen crew lead times (CLT) for activities on this and previous sequence steps. In turn, these CIT are used to select crew lead times (CLT) for the activities on the next sequence step using the appropriate confidence level. These CLT are then assigned to X_CrewLeadTime durations for the activities in the next sequence step and the process is repeated.

6 EXAMPLE PROJECT

An example repetitive project consisting of 4 similar but non-identical units, with 7 activities each, is used to demonstrate the application of the sequence step algorithm. The example model and the sequence step algorithm have been implemented using the Stroboscope system.

6.1 Precedence Network And Input

The example project includes 4 non-identical units (e.g., floors, houses, etc.) each requiring 7 work activities (A, B, C, D, E, F, and G) performed by different crews. The activ-

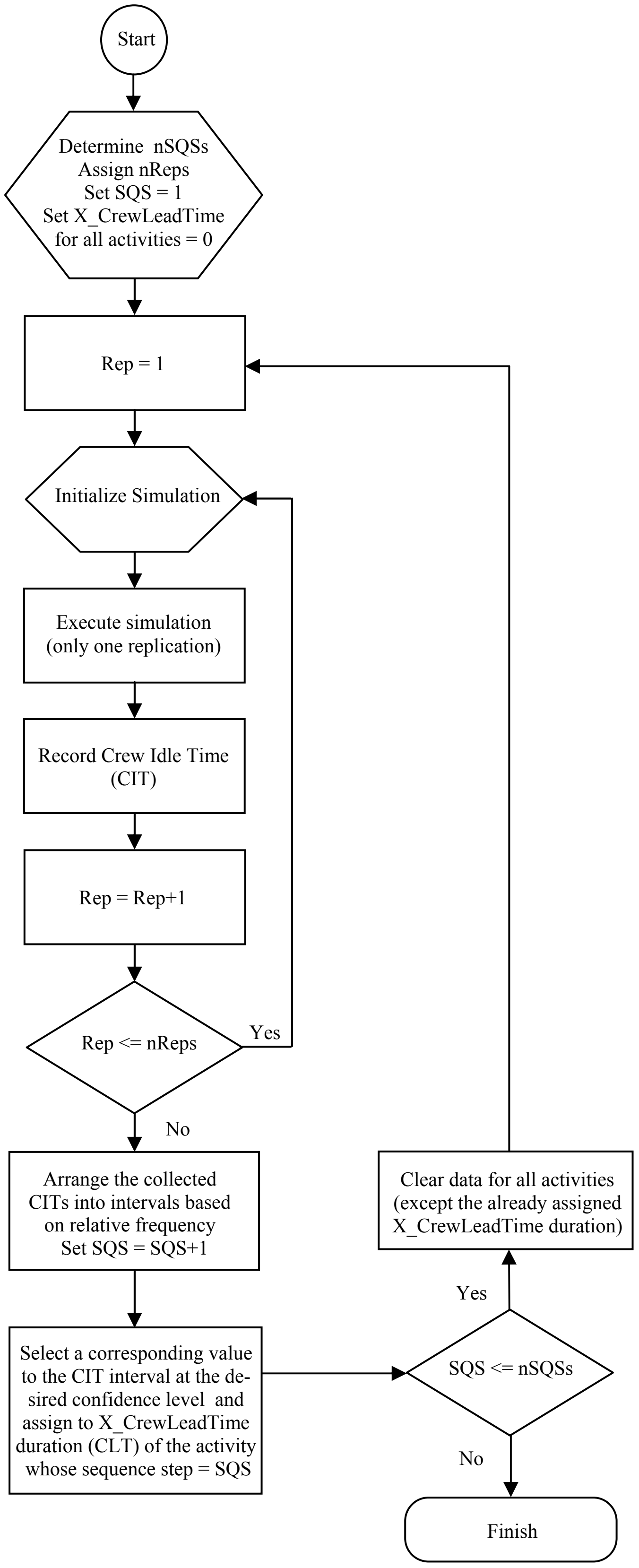
Start
Determine nSQSs
Assign nReps
Set SQS = 1
Set X_CrewLeadTime for all activities = 0
Rep = 1
Initialize Simulation
Execute simulation (only one replication)
Record Crew Idle Time (CIT)
Rep = Rep+1
Rep <= nReps
Yes
No
Arrange the collected CITs into intervals based on relative frequency
Set SQS = SQS+1
Select a corresponding value to the CIT interval at the desired confidence level and assign to X_CrewLeadTime duration (CLT) of the activity whose sequence step = SQS
SQS <= nSQSs
Yes
No
Clear data for all activities (except the already assigned X_CrewLeadTime duration)
Finish

Figure 5: Sequence Step Algorithm Flow Chart

ity-on-node network for each repetitive unit is the same and appears in Figure 1. As indicated in Figure 1, activity A is in sequence step 1, activities B and C are in sequence step 2, etc. The amount of work for each activity in each of the 4 units is different and is shown in Table 1. For example, the work amounts for activity A in units 1, 2, 3, and 4 are 100, 250, 150, and 200 work units respectively.

Table 1: Activity Work Amount In Each Repetitive Unit

	Activity						
Unit	A	B	C	D	E	F	G
1	100	150	200	150	100	150	50
2	250	100	150	200	150	250	200
3	150	200	50	100	50	50	50
4	200	150	200	150	100	100	150

In each of the 4 repetitive units, crew production rates (in work amounts per day) for each of the 7 activities are assumed to follow normal distributions with the means and standard deviations shown in Table 2. Consequently, the duration of each activity in each of the 4 repetitive units varies because of differences in work amounts from unit to unit and because of random production rates.

Table 2: Moments of Daily Crew Production Rates

Activity	Mean	SD
A	10	1.0
B	20	2.0
C	15	1.5
D	15	1.5
E	25	2.5
F	15	1.5
G	20	2.0

6.2 Assigning Crew Lead Times

The schedule resulting from the sequence step algorithm after the first simulation replication for sequence step 1 is shown in Figure 6. "SQS1" in the title indicates the first sequence step, while "n1" indicates the first replication. The resulting project schedule at the end of the first replication is shown as a production diagram.

At this point, the durations CLT for all activities X_CrewLeadTime have been set to 0. Thus, the first data point for the crew idle time, CIT, for each activity (measured from the beginning of the project) is given by the ESD of the activity in the first unit plus the sum of all its lags between units. For example, the first data point for the B crew idle time, CIT_B, is given by the sum ESD_{B1} + $Lag_{B1,B2}$ + $Lag_{B2,B3}$ + $Lag_{B3,B4}$.

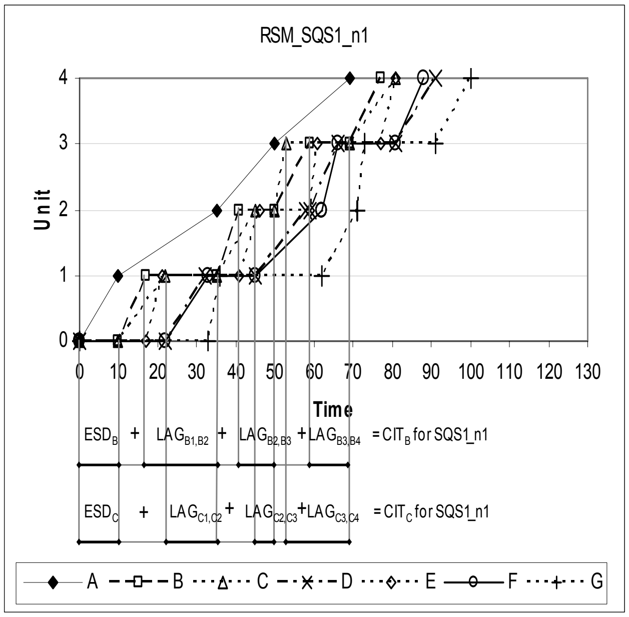

Figure 6: Measurement of Crew Idle Time (Production Diagram, 1st Replication, SQS1)

For this example, 10,000 replications were simulated within each sequence step. Thus, 10,000 data points for the crew idle times of activities B and C were collected by the end of processing sequence step 1. These 10,000 crew idle times for C and B each were then arranged by relative frequency in cumulative bins using an interval of 5 time units as shown in Table 3. Each row of this table shows a time value and the percent of the 10,000 crew idle times for activity B that were less than that value. For example, out of a total of 10,000 values, 9,427 crew idle times for activity B were less than 55 days. Thus, if activity B is scheduled to start on day 55 there is 94.27% probability that its continuity in resource utilization will be maintained. On the other hand, If CLT_B is set to 75, then continuous resource usage is virtually guaranteed.

Table 3: B Crew Idle Times from SQS1

CIT_B	
Range	% Frequency
< 35	0.01
< 40	1.26
< 45	19.37
< 50	65.92
< 55	**94.27**
< 60	99.40
< 65	99.95
< 70	99.99
< 75	100.00

In this example, an 80% confidence level is used to select crew lead times (CLT) for all activities. Hence, B_CrewLeadTime is set to a duration of CLT_B = 55 days (the first value in Table 3 that exceeds 80%). Activity B cannot start earlier than time 55 because its crew will not be available. Similarly, at 80% confidence, the duration for C_CrewLeadTime is set to 50 days.

Once B_CrewLeadTime and C_CrewLeadTime are set to 55 and 50 days respectively, the algorithm moves to sequence step 2. The assigned crew lead times CLT_B = 55 days and CLT_C = 50 days will remain constant until the algorithm finishes.

Table 4 shows the average total idle time (average CIT) that each crew spends on site from the time it arrives. Notice that in SQS1 crews B and C arrive at the site at time 0 and thus have average total idle times of 48 and 44 days respectively. In SQS2 crews B and C are assigned to arrive on days 55 and 50 respectively and thus have 0 average idle time from then on.

Table 4: Average Total Crew Idle Time On Site (CIT)

	Crew						
SQS	A	B	C	D	E	F	G
1	0	48	44	54	66	54	79
2	0	0	0	64	73	64	90
3	0	0	0	0	0	0	96
4	0	0	0	0	0	0	0

In sequence step 2, the algorithm collects 10,000 samples of crew idle times (CIT) for activities D, E, F in sequence step 3. These collected idle times reflect the impact of the chosen crew lead times for activities B and C (55 and 50 days) as well as the impact of variability in activity durations. Figure 7 shows the resulting schedule for the 1st replication. As shown in Figure 7, activities B and C cannot start at their earliest start date anymore because their crews have been deliberately delayed to days 55 and 50 respectively. A comparison of Figures 6 and 7 indicates that the chosen delays for activities B and C provide uninterrupted work for crews B and C, at least for this replication. The assigned crew lead times for all activities appear in Table 5.

From the 10,000 project replications in sequence step 2, we collect 10,000 samples of crew idle times (CIT) for activities D, E, F that are then organized in cumulative frequency tables similar to Table 3. As shown in Table 5, the assigned lead times for crews D, E, and F, at the end of processing sequence step 2 are CLT_D = 70, CLT_E = 80, and CLT_F = 70 days. These lead times are then assigned to the durations of D_CrewLeadTime, E_CrewLeadTime, and F_CrewLeadTime respectively, and are used in processing the next step of the algorithm in SQS3.

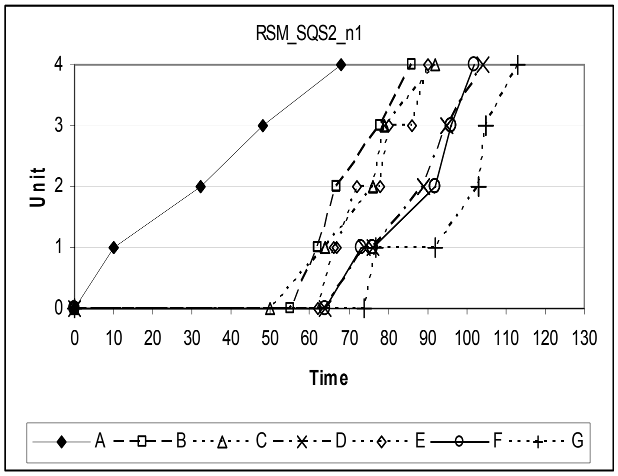

Figure 7: Production Diagram, 1st Replication, SQS2

Figure 8 shows the project schedule from processing the first replication of sequence step 3. Activities D, E, F, have no idle time (at least for this replication) and as can be seen, even the continuity in activity G is improved as shown in the figure and in Table 5. This is a byproduct of the assigned crew lead times, CLT, for activities B, C, D, E, and F, in all preceding sequence steps 1-3. At the end of the sequence step process, the crew lead time for activity G is assigned to CLT_G = 100 days and the resulting average total crew idle time has dropped to zero.

Table 5: Sum of Lags Between Units, Assigned Crew Lead Time, and Average Project Duration

	Sum of Lags Between Units							**Assigned Crew Lead Time** (CLT)							**Average Project Duration**	**Average Total Idle Time**
SQS	A	B	C	D	E	F	G	A	B	C	D	E	F	G		
1	0	38	34	30	48	30	45	0	0	0	0	0	0	0	102	225
2	0	0	0	1	11	1	16	0	55	50	0	0	0	0	113	29
3	0	0	0	0	0	0	12	0	55	50	70	80	70	0	119	12
4	0	0	0	0	0	0	0	0	55	50	70	80	70	100	123	0

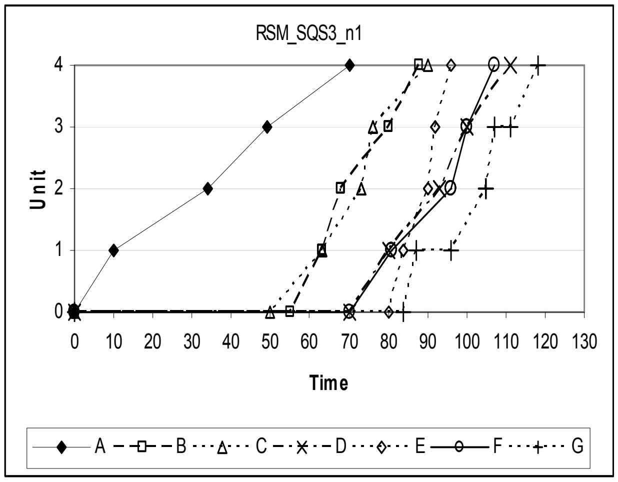

Figure 8: Production Diagram, 1st Replication, SQS3

Figure 9 shows the final schedule where crew lead times for all activities have been assigned. As can be seen, all activities start at their respective CLT. From Table 5 we see that the average total idle time for all crews has been reduced from 225 crew-days to 0, while the average project duration has grown from 102 to 123 days.

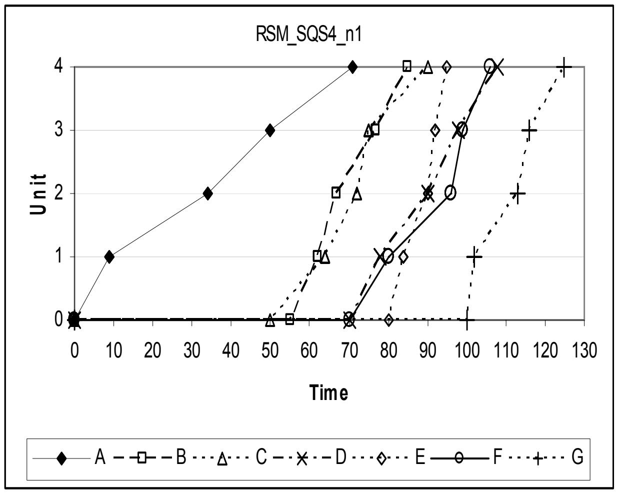

Figure 9: Production Diagram, 1st Replication, SQS4

6.3 Discussion Of Results

Table 4 shows the average total idle time (average CIT) that each crew spends on site from the time it arrives. Table 5, on the other hand, shows the average idle time between units. The difference is that values in Table 4 include not just lags, but also the idle time from the time a crew comes to the site until it starts work in the first unit.

It is interesting to notice that in both tables the average idle times (CIT) for all crews drop to 0 once activities are assigned the crew lead times shown in Table 5. Table 4 shows that as some crew lead times are assigned from one SQS to the next, the average CIT for succeeding activities increase. In contrast, in Table 5 the average idle time between units actually decreases. This is as expected, because the assignment of crew lead times to predecessor activities delays the first start of successor activities even more (and thus increases their initial idle time), but also reduces their idle time between units.

Table 5 shows that as the sequence step algorithm progresses, the average project duration increases from 102 days to 123 days. The corresponding cumulative distributions for project duration for each sequence step in the algorithm are shown in Figure 10. The greatest horizontal time shift between successive distributions occurs between sequence steps 1 and 2 which gives an average increase in project duration from 102 to 113 days. The reason for this can be observed in Figure 6. Activity A is the slowest activity in the project and as a result contributes the most to discontinuities in the work of other activities. Thus, to avoid discontinuities in its successor activities B and C, it is necessary to shift their crew arrival date significantly, which in turn increases project duration the most. Figure 7 shows that once the crew lead times for B and C were assigned, the rest of the activities (except G) had almost no work interruption between units, at least in the first replication of the project.

6.4 Selection of Confidence Level

An important question in the optimal use of the sequence step algorithm is how to select an appropriate confidence level for the occurrence of crew work interruptions to balance the increase in project duration. This is an important issue because high confidence levels (to virtually eliminate idle time) can lead to significant increases in project duration.

To address this issue, Figure 11 shows 5 lines that relate average total crew idle time (total CIT in crew-days) and average project duration (in days). Each line corresponds to a different confidence level: 20%, 40%, 60%, 80%, and 100%. Moreover, each line consists of four points that correspond, from left to the right (or top to bottom), to the four SQS steps 1-4 of the algorithm. Thus, the expected total idle time between units decreases as the algorithm proceeds from one sequence step to the next. Clearly, selecting a greater confidence level decreases the expected crew idle time but also increases the expected project duration. Thus, allowing some interruption can benefit project duration.

For all confidence levels, the most dramatic reduction in average total crew idle time occurs between sequence steps 1 and 2 in the algorithm. From that point on, the reduction in average idle time from sequence step 2 to 4 is relatively small and results in ever higher increases in average project duration as the confidence level increases.

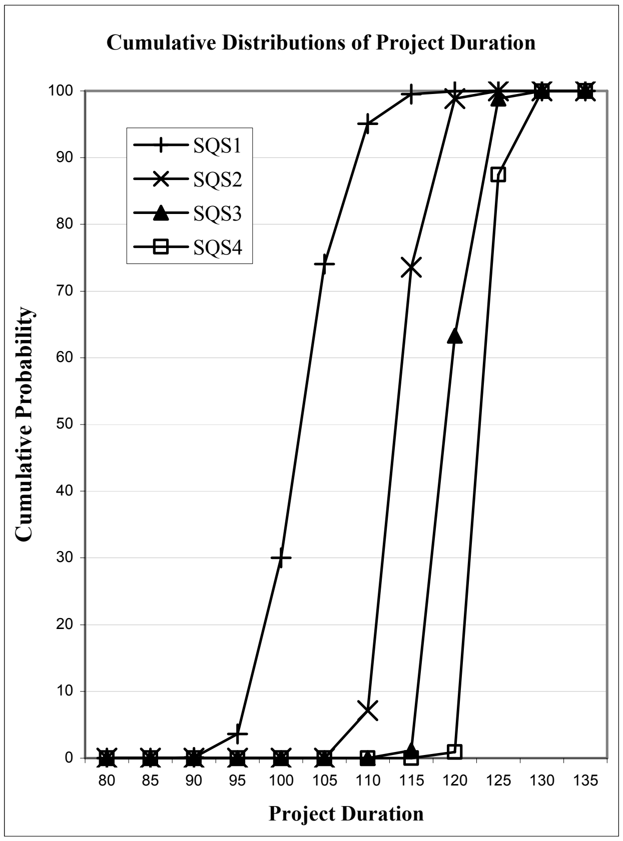

Figure 10: Cumulative Distributions of Project Duration at 80% Confidence Level of Continuous Resource Utilization

It is very important to notice that irrespective of confidence level, the average total idle times at the completion of the algorithm (bottom point in each line) are *very small*. In particular, the expected idle time for confidence levels of 60%, 80%, and 100% is practically zero. Yet, the expected project durations for confidence levels of 60%, 80%, and 100% increase from 118 to 123 to 158 days. It is not hard to conclude that the optimal confidence level is between 60% and 80%, but definitely not 100%.

Figure 12 shows the corresponding probability density functions for project duration for different confidence levels. Here it is again evident that there is little difference between confidence levels of 60% and 80% but there is substantial difference between 80% and 100%.

This figure also shows the probability density functions for project duration when the project is scheduled using CPM and RSM. For the CPM case, activities in each replication are allowed to start as early as their predecessors allow. Thus, project duration tends to be the shortest but has large variability. For the RSM case, activities in each replication are scheduled with perfect hindsight so as to eliminate crew idle time. This produces a slight increase in expected project duration over the CPM case but completely eliminates idle time. The difference between the RSM expected project duration and that for (say) an 80% confidence level represents the value of perfect information about the true activity durations that will be experienced during construction. If these were known ahead of time, then the project could be scheduled with even shorter crew lead times and have an even shorter project duration. However, that is not the case in real life, and hence the need for this algorithm.

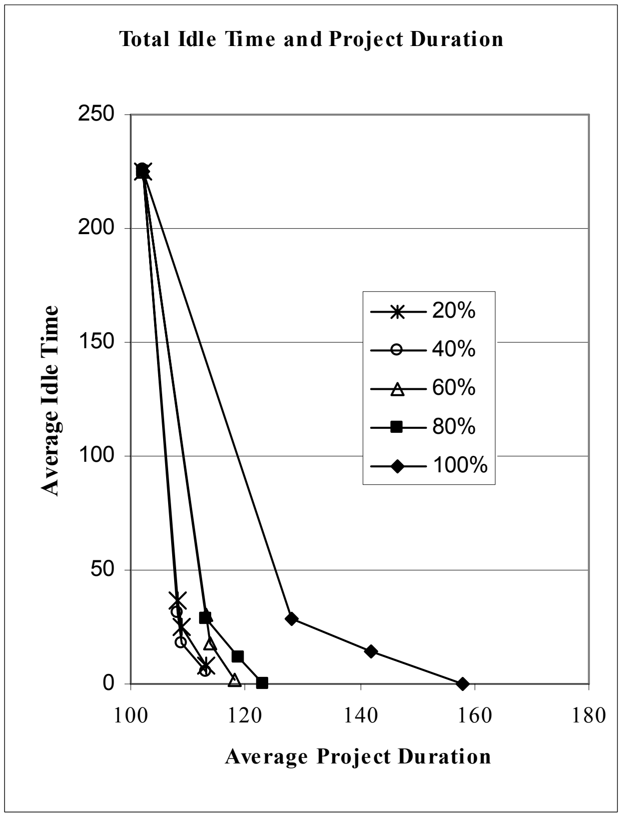

Figure 11: Average Idle Time vs. Average Project duration for Different Confidence Levels

7 CONCLUSION

The sequence step algorithm is the first to address the problem of scheduling probabilistic repetitive projects to eliminate crew idle time. The algorithm can be easily adapted to different resource-based simulation software by adding two nested loops: an inner replication loop and an outer sequence step loop. Its application in Stroboscope has proven very effective and can help tackle a difficult problem that had hitherto eluded formal treatment. The problem of continuous resource utilization is central to many types of simulation models (and not just repetitive projects) that can now be analyzed and given the necessary attention.

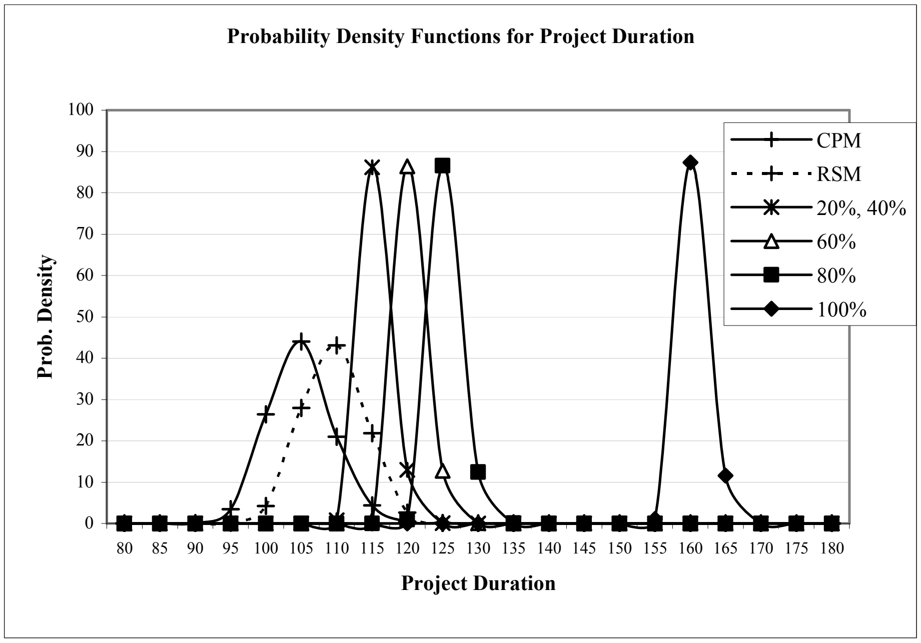

Figure 12: Project Duration Density Functions for Different Confidence Levels of Continuous Resource Utilization

ACKNOWLEDGEMENTS

The writers would like to thank Julio C. Martinez, Professor at Virginia Tech, for his enthusiastic technical support with Stroboscope. Additionally, the writers would like to thank Sirarat Sarntivijai and George A. Ioannou, graduate students at the University of Michigan for their help.

REFERENCES

Harris, R. B. 1978. *Precedence and Arrow Networking Techniques for Construction*. John Wiley & Sons, Inc., New York, ISBN 0-471-04123-8, 50-51.

Harris, R.B., and P. G. Ioannou. 1998. Scheduling projects with repeating activities. *Journal of Construction Engineering and Management*, ASCE, July/August 1998, 269-278.

Ioannou, P. G., and V. Likhitruangsilp. 2005. Simulation of multiple-drift tunnel construction with limited resources, In *Proceedings of the 2005 Winter Simulation Conference*, ed. M.E. Kuhl, N.M. Steiger, F.B. Armstrong, and J.A. Joines, 1483-1491. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Martinez, J. C. 1996. *STROBOSCOPE: State and resource based simulation of construction processes.* Doctoral Dissertation, Dept. of Civil and Environ. Engineering, University of Michigan, Ann Arbor, Michigan.

Martinez, J. C. and P. G. Ioannou. 1999. General Purpose Systems For Effective Construction Simulation. *Journal of Construction Engineering and Management*, ASCE, (125)4, July-August 1999.

AUTHOR BIOGRAPHIES

PHOTIOS G. IOANNOU is Professor of Civil and Environmental Engineering at the University of Michigan. He received a Dipl. Civil Eng. from the National Technical University of Athens, Greece, in 1979, and a SMCE and Ph.D. in Civil Engineering from MIT in 1981 and 1984. From 1989-1995 he served as Chairman of the Computing in Construction Technical Committee of the ASCE. He co-developed three construction simulation systems: UM-CYCLONE with R.I. Carr (1989), COOPS with L.Y. Liu (1991), and STROBOSCOPE with J.C. Martinez (1996). His research is in construction engineering and management, and in particular in decision support systems and construction process modeling and simulation. His e-mail is <photios@umich.edu> and his website is <www.cem.umich.edu/Ioannou>.

CHACHRIST SRISUWANRAT is Ph.D. student in Construction Engineering Management at the University of Michigan. His research is in the area of resource utilization and construction simulation under the direction of P.G. Ioannou. His e-mail is <csriuwa@umich.edu>.