THE INVESTIGATION OF LEAD-TIME BUFFERING UNDER UNCERTAINTY USING SIMULATION AND COST OPTIMIZATION

Chachrist Srisuwanrat[1] and Photios G. Ioannou[2]

ABSTRACT

The impact of uncertainty and variability in the productivity of trades aggravates the problem of work interruptions and idle time within repetitive activities. To eliminate the interruptions and idle time in order to achieve smooth work flow of resources, activities are deliberately delayed from their early start date. However, this practice induces a problem of tradeoffs between project cost and duration. Many recent studies have suggested that different types of buffers can be used to absorb the impact of uncertainty and variability on production work flow and most studies focus on using buffer to determine "when to halt an on-going production line". In contrast, this paper focuses on "when to start a production line so that there is no interruption". Two different approaches to lead-time buffering, the *sequence step algorithm* (SQS-AL) and the *completed unit algorithm* (CU-AL), are investigated using STROBOSCOPE (a discrete-event simulation system) with a special search add-in that implements a genetic-algorithm (GA). The investigation reveals that applying lead-time buffer provides better work flow and greater project profit; however, these depend on the penalty cost of work flow interruption and indirect cost. Both algorithms have implications that translate to advantages and limitations depending on assumptions, simplicity of simulation model, project characteristics, and uncertainty.

KEY WORDS

Lead-time buffering, the *sequence step algorithm*, the *completed unit algorithm*, production work flow, continuous resource utilization, idle time, simulation, STROBOSCOPE, genetic algorithm, profit maximization.

INTRODUCTION

One of the main concerns in scheduling repetitive activities is how to keep resources working continuously without interruptions or idle time. Interruptions and idle time in repetitive work are caused by differences in activity production rates and by when repetitive activities are scheduled to start for the first time.

In general, scheduling repetitive activities effectively requires that the following characteristics and constraints must be considered:

- Production rates
- Variation in production rates
- Precedence constraints

[1] Ph.D. Candidate, Civil and Env. Engineering. Department, 2350 G.G. Brown, University. of Michigan, Ann Arbor, MI 48109-2125, christ_cv@hotmail.com

[2] Professor, Civil and Env. Engineering. Department, 2350 G.G. Brown, University of Michigan, Ann Arbor, MI 48109-2125, photios@umich.edu

- Resource availability constraints
- Resource continuity constraints
- Direct and indirect project cost
- Penalty cost and time from interruptions in production lines

Omitting one of these factors would incur waste in time and capital, which can be seen when repetitive activities are scheduled at their early start dates as in CPM or PERT. Early-start scheduling ignores resource continuity constraints and causes work flow interruptions that result in idle time, lower productivity, and, consequently, penalty cost and time.

To eliminate interruptions and idle time, resource availability and resource continuity constraints must be taken into account in scheduling production lines. Examples of scheduling methods considering these constraints are Line-Of-Balance (LOB), Linear Scheduling Method (LSM), and Repetitive Scheduling Method (RSM). Taking constraints into their calculation, these methods could employ one or more of the following techniques to minimize the number of interruptions and idle time:

- Balancing production rates
- Delaying activities to eliminate interruptions and idle time within the activities
- Introducing work breaks

Even so, current repetitive scheduling methods (e.g., LOB, LSM, and RSM) are effective to a certain extent since they do not explicitly consider uncertainty and variability in construction tasks. They are deterministic scheduling methods and as such rely on average production rates or activity durations to schedule repetitive work. Unfortunately, average production rates alone are not sufficient to capture the behavior of actual construction (Tommelein et al. 1998). Consequently, when these methods are used in practice, they still result in work interruptions between repetitive units, and hence, in resource idle time. Thus, new approaches are needed to solve the problem of scheduling production lines or repetitive activities in a probabilistic manner.

One of the often-used and facilitating approaches for scheduling repetitive activities under uncertainty and variability is to use discrete-event simulation in conjunction with user-specified buffers. Several studies have suggested that different types of buffers can be used to absorb the impact of uncertainty and variability on production work flow (Ioannou and Likhitruangsilp 2005, Ioannou and Srisuwanrat 2006, Tommelein et al. 1999, Alarcon and Ashley 1999, Sakamoto et al. 2002, Alves and Tommelein 2004). Many of these studies focus on using buffers to determine “when to halt an on-going production line”. In this paper we focus on the underlying problem of using buffers to control “when to start a production line so that there is no interruption”. We will call this second type of buffer “a lead-time buffer”.

A lead-time buffer (or lead time) is the time interval, measured from project start, that specifies when a particular resource should arrive at the site. The purpose of lead-time buffers is to control the arrival of resources to the jobsite, and hence to control the start of the associated activities. In this paper we discuss those situations where each repetitive activity requires a separate resource (no sharing resources between activities), and where there is no variability in specified crew arrival dates. Hence, the terms lead-time buffer, crew lead time, crew arrival date, and activity start date are used interchangeably.

Figure 1 illustrates the use of a lead-time buffer to eliminate idle time in a successor activity (activity B), which is caused by a difference in production rates from its

predecessor (activity A). As shown in Figure 1.1, activity B has idle time of 20 days ($Lag_{B1,B2}$ + $Lag_{B2,B3}$). To eliminate this idle time, we can introduce a lead-time buffer of 35 days measured from the project start date, as shown in Figure 1.2. Thus, resource B is scheduled to arrive at day 35. Detailed information about calculating lead-times and eliminating idle time in deterministic scheduling problem for repetitive projects appears in (Harris and Ioannou 1998).

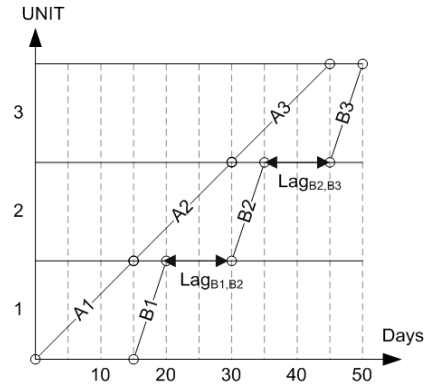

Figure 1.1: Early start date schedule with 20 days of idle time in activity B

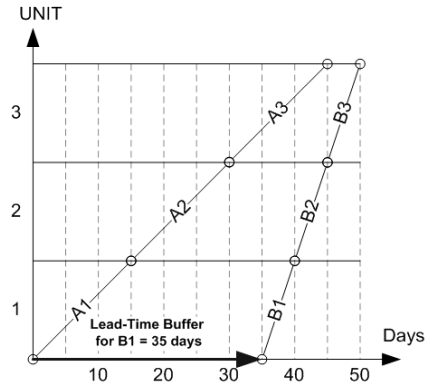

Figure 1.2: Using lead-time buffer of 35 days to eliminate idle time in activity B

In this paper we use simulation and genetic algorithms to investigate two different methodologies developed by the authors for determining lead-time buffers for repetitive activities under uncertainty and variability. The first is the "*sequence step algorithm (SQS-AL)*" and the second is the "*completed unit algorithm (CU-AL)*".

The main difference between the two algorithms is how they derive the lead-time buffer for a production line (or a repetitive activity). In brief, SQS-AL derives lead-time buffers for activities on a particular sequence step from their expected *crew idle times* that result from the *selected lead-time buffers* for activities on previous sequence steps. In CU-AL lead-time buffers are selected to coincide with the completion of a user-specified number of predecessors' completed units. Intuitively, SQS-AL should provide better results than CU-AL, since lead-time buffers from SQS-AL can be any real number, whereas lead-time buffers from CU-AL can only be selected from the predecessors' finish dates of particular activity units. However, both approaches have implications that translate to advantages and limitations depending on assumptions, simplicity of simulation model, project characteristics, and uncertainty.

These algorithms are presented through an example project with 5 activities that repeat over 10 units, that we have modeled using STROBOSCOPE, a discrete-event simulation system. The effectiveness of the algorithms is evaluated and compared on the basis of an objective function that measures project profit and which is sensitive to project duration and the penalty cost of idle time. To perform the investigation of the simulation models we have developed ChaStrobeGA, a STROBOSCOPE add-on, that implements an optimization system using a Genetic Algorithm approach. The results from the example are shown and discussed at the end.

GENERAL CONSIDERATIONS

In the discussion that follows, it is important to keep in mind that lead-time buffers are always measured from project start date. Their purpose is to provide continuous

production work flow under uncertainty and variability, and not to halt an on-going production line. These lead-time buffers absorb the effects of uncertainty and variability in the predecessors' production rates which, in turn, reduce waiting times and the number of interruptions. This is similar to what should happen in construction where a general contractor should be able to specify to specialty trades and subcontractors ahead of time the specific date for their arrival to the site so as to maximize their productivity with no interruption, and also to maximize project profit.

SEQUENCE STEP ALGORITHM (SQS-AL)

The sequence step algorithm (SQS-AL) uses the cumulative frequency of an activity's idle time and a user-specified confidence level to determine the activity's lead-time. The lead-time buffers introduced to preceding activities decrease the idle time of succeeding activities. Hence, SQS-AL determines lead-time buffers and activity idle times in sequence step order. The process of constructing cumulative frequency and determining lead-time buffers moves from activities in one sequence step to activities in the next sequence step until the end of the network.

An advantage of SQS-AL is that the lead-time buffers calculated by SQS-AL are derived directly from resource idle times and can be any real value. Moreover, the process of progressing from one sequence step to another results in accurate values of true idle times prior to determining lead-time buffers of activities in that sequence step. A detailed discussion of SQS-AL can be found in (Ioannou and Srisuwanrat 2006).

COMPLETED UNIT ALGORITHM (CU-AL)

For each precedence link in a network, the completed unit algorithm (CU-AL) minimizes the idle time in the successor by delaying the successor's start date until a specified number of repetitive units have been completed by the predecessor. After executing a certain number of replications, the activity start dates for each activity from these replications are averaged and used as the lead-time buffer measured from project start.

The start dates collected from these replications are not early start dates, but rather the dates when their predecessors complete a certain number of units specified by the user. In this paper, the number of predecessors' completed units shall be called "BufferXY"; where X is the predecessor's name, and Y is the successor's name. For example, the statement "BufferAB equals 5" indicates that activity A is a predecessor of activity B and that activity B can start only when five units or more of activity A are completed. For example, after 1000 replications are executed, the start date values of B from these replications are averaged to derive the lead-time buffer for activity B. Then, another 1000 replications are executed, but this time activity B is scheduled to start at the calculated average value (lead-time buffer for B) from the first 1000 replications. During these replications, the idle times for activity B are collected to obtain the actual idle time corresponding to scheduled start date of B (the averaged value of B's start date from the first 1000 replications).

EXAMPLE PROJECT

To investigate the efficiency of the sequence step algorithm and the completed unit algorithm, an example of 5 repetitive activities with 10 units each are modeled and tested in STROBOSCOPE (Martinez and Ioannou 1999). Figure 2 is the activity-on-node network for each repetitive unit for the example. Table 1 shows the amount of work for

each activity in each of the 10 units, and the mean values of activity productivity. In each repetitive activity, productivity per day is assumed to follow a normal distribution with the mean shown in Table 1 and a coefficient of variation of 10%.

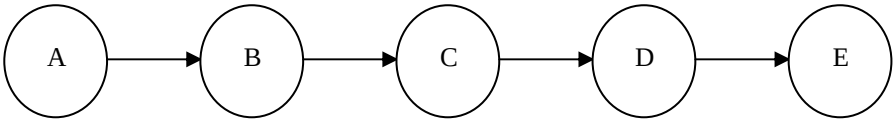

Figure 2: Activity-On-Node Networks for Single Unit

Table 1: Activity Daily Productivity and Work Amounts

Activity	Mean	SD	Unit									
			1	2	3	4	5	6	7	8	9	10
			Work Amounts									
A	10	1.0	300	250	150	200	200	250	200	250	300	150
B	20	2.0	150	100	200	150	150	200	100	150	200	200
C	15	1.5	200	150	50	200	50	100	200	200	150	100
D	15	1.5	150	200	100	150	150	150	150	100	100	150
E	25	2.5	100	150	50	100	200	150	150	50	100	200

SIMULATION

The simulation models for the sequence step algorithm (SQS-AL) and the completed unit algorithm (CU-AL) are implemented using STROBOSCOPE (Martinez and Ioannou 1999). STROBOSCOPE is a resource-driven discrete-event simulation system that is well suited for this task because it is activity oriented and has powerful capabilities to collect data and alter variable inputs during simulation. Nevertheless, the models could also be developed in other simulation environments. The activity sub-network models suggested in (Ioannou and Srisuwanrat 2006) are used for both algorithms. More details about the simulation model network can be found in (Ioannou and Srisuwanrat 2006).

For SQS-AL, each sequence step is simulated 1000 times in order to derive lead-time buffers for activities in that sequence step. Therefore for this network with 5 sequence steps, SQS-AL requires a total of 5000 replications to derive the lead-time buffers for all activities. After the lead-time buffers are obtained, another 1000 replications are executed to collect idle times for each activity, average project duration, and average project profit.

For CU-AL, the entire network is simulated 1000 times in order to derive lead-time buffers, and another 1000 replications to collect idle times, average project duration, and average project profit according to the derived lead-time buffers. There are differences in the means of deriving lead-time buffers and total number of replications required by SQS-AL and CU-AL. The differences and implications of using the two different approaches are discussed later in the results section.

OPTIMIZATION USING GENETIC ALGORITHM

The optimization in this paper is achieved by a STROBOSCOPE add-on called “ChaStrobeGA”, developed by the authors. ChaStrobeGA employs a genetic algorithm to optimize an objective function by selecting values for decision variables within user-specified domains. For SQS-AL the decision variables are the activity confidence levels

(for B, C, D, and E) ranging from 0 to 0.9 with a step of 0.1 (these are the domain values). For CU-AL, the decision variables are the number of predecessors' completed units for each activity relationship (BufferAB, BufferBC, BufferCD, and BufferDE) ranging from 1 to 10 with an interval of 1. ChaStrobeGA automatically alters the input variables, creates STROBOSCOPE code for the input variables, runs the STROBOSCOPE model, collects the results, and performs the genetic algorithm.

For the example presented in this paper, the genetic algorithm parameters are 31 generations, 30 populations for each generation, 0.6 cross over probability, and 0.05 mutation probability. In brief, 930 cases, which could be the same or different cases, are tested. In each case, ChaStrobeGA creates simulation code based on 1) user-given alternative parameters, 2) result values of the objective function, and 3) genetic algorithm (reproduction, cross over, and mutation). For example, ChaStrobeGA creates simulation code for SQS-AL using confidence levels of 0.2, 0.4, 0.8, and 0.5 for activities B, C, D, and E, respectively. This step of selecting confidence level for each case is executed 30 times (called 30 populations for the first generation), and then these 30 cases are executed using STROBOSCOPE to get the project profit (objective function) from each case. After the 30 cases (populations) in the first generation are executed, ChaStrobeGA creates another 30 populations (the 2^{nd} generation) by reproducing, crossover, and mutation processes according to genetic algorithm. Then, these steps are repeated until 31 generations are obtained. As the GA progresses from one generation to another, the average value of project profits from each generation tends to increase. In the end, 930 cases are sorted by their expected project profits and durations in order to derive the best solution according to GA.

OBJECTIVE FUNCTION AND COST MODEL

An objective function has been developed to investigate the effectiveness of the sequence step algorithm (SQS-AL) and the completed unit algorithm (CU-AL). Since maximizing the profit of construction project is one of the main purposes, the objective function used here is to maximize the following project profit function:

$$Project\ Profit = Contract\ Price - Direct\ Cost - Indirect\ Cost$$

Our focus is to eliminate the waste stemming from resource idle time, to allow the relaxation of resource continuity constraints under uncertainty and variability, and to minimize the cost of delaying the project completion date. Thus, the direct cost is categorized into two groups: productive direct cost (assumed constant) and unproductive direct cost (linear function of time). The objective function is shown below.

$Project\ Profit = 70{,}000 - Unproductive\ Direct\ Cost - Indirect\ Cost$
$Indirect\ Cost = 5000 + Project\ Duration\ (in\ days) \times \$\ 50/day$
$Unproductive\ Direct\ Cost = CIT_B * 100 + CIT_C * 120 + CIT_D * 60 + CIT_E * 90$
Where, *CIT = Crew Idle Time* in days

ChaStrobeGA uses GA to maximize *Project Profit* by changing the decision variables (confidence level for SQS-AL; and BufferAct1Act2 for CU-AL). Results from GA are shown in the next section.

DISCUSSION OF RESULTS

Tables 2 and 3 show the results for the example project using the sequence step algorithm (SQS-AL) and the completed unit algorithm (CU-AL), respectively. In each table, the first ten rows are results from *base cases* and the last row shows results from the genetic algorithm (GA). Base cases are usually employed to experiment with the scope and the possible range of project duration, project profit, and, more generally, the behavior of the system. Here, base cases (the first 10 rows in each table) use the same alternative parameters for all activities. For example, confidence levels for all activities are 0.1 for SQS-AL, and predecessors' completed units for all activities are 1 for CU-AL. Also, the effectiveness of base cases for both algorithms shall be evaluated and a comparison shall be made to evaluate how good base cases are in terms of expected project profit and project duration relative to the optimal results from GA.

Table 2: SQS-AL Results

Confidence Level				Project Profit ($)	Project Duration (days)	Idle Time (crew days)					Lead-Time Buffer (days)			
B	C	D	E			B	C	D	E	Total	B	C	D	E
0	0	0	0	6,173	267	125	110	107	148	490	31	39	53	63
0.1	0.1	0.1	0.1	50,389	284	2	0	0	2	4	154	163	177	227
0.2	0.2	0.2	0.2	50,463	286	1	0	0	1	2	155	164	179	230
0.3	0.3	0.3	0.3	50,480	286	1	0	0	1	2	155	164	179	230
0.4	**0.4**	**0.4**	**0.4**	**50,493**	**288**	**1**	**0**	**0**	**1**	**2**	**156**	**165**	**180**	**232**
0.5	0.5	0.5	0.5	50,483	288	1	0	0	0	1	156	165	180	233
0.6	0.6	0.6	0.6	50,476	289	0	0	0	0	0	157	166	181	234
0.7	0.7	0.7	0.7	50,416	291	0	0	0	0	0	158	167	182	236
0.8	0.8	0.8	0.8	50,328	293	0	0	0	0	0	158	167	183	238
0.9	0.9	0.9	0.9	50,289	294	0	0	0	0	0	159	168	184	239
0.2	**0**	**0**	**0.6**	**50,562**	**285**	**1**	**0**	**0**	**0**	**1**	**155**	**163**	**177**	**230**

(Best base case result) / (GA result) = 50,493 / 50,562 = 99.86%

Table 3: CU-AL Results

Buffer (Units)				Project Profit ($)	Project Duration (days)	Idle Time (crew days)					Lead-Time Buffer (days)			
AB	BC	CD	DE			B	C	D	E	Total	B	C	D	E
1	1	1	1	6,051	267	125	111	108	149	493	31	39	52	63
2	2	2	2	19,817	267	99	79	66	93	337	56	70	94	118
3	3	3	3	29,600	267	84	54	36	56	230	72	96	124	155
4	4	4	4	41,146	273	63	25	0	10	98	92	124	166	208
5	**5**	**5**	**5**	**45,315**	**307**	**43**	**0**	**0**	**1**	**44**	**113**	**153**	**199**	**251**
6	6	6	6	45,251	360	18	0	0	0	18	138	188	242	305
7	7	7	7	44,478	410	0	0	0	0	0	159	215	282	355
8	8	8	8	41,779	464	0	0	0	0	0	184	248	329	409
9	9	9	9	38,842	523	0	0	0	0	0	214	289	380	468
10	10	10	10	36,664	567	0	0	0	0	0	230	315	414	512
7	**1**	**1**	**5**	**50,478**	**289**	**0**	**0**	**0**	**1**	**1**	**159**	**167**	**180**	**233**

(Best base case result) / (GA result) = 45,315 / 50,478 = 89.77%

The optimal results from GA appear in the last rows of Tables 2 and 3. In contrast to the base cases, the GA results that yield maximum project profit call for combinations of different decision values for each activity, i.e., (0.2, 0, 0, 0.6) and (7, 1, 1, 5). It should also be noted that the GA results shown in each table may not be the best solution for that methodology, since GA does not provide an exhaustive search among all possibilities. Nevertheless, the GA results should be very close to the anticipated best solutions. Moreover, there could be more than one solution that results in project profits with less than 0.1% difference from the best solution shown in each table. For this project, for example, there are more than 50 solutions that yield project profits within 0.1% difference from the project profit given by the GA optimum shown.

The first base cases in Tables 2 and 3 have no lead-time buffers and correspond to early start solutions. Hence, they give results similar to PERT. The numbers shown under the Lead-Time Buffer headings are simply average early start dates. Thus, without lead-time buffers, we have the minimum expected project duration at 267 days, but also the maximum total idle time of about 490 crew days. The introduction of even small lead-time buffers (e.g., the second base case) reduces total idle time significantly to 4 crew-days for SQS-AL and 337 crew-days for CU-AL, while project duration increases by 17 days for SQS-AL and 0 days for CU-AL.

For CU-AL in Table 3, the first three cases have the *same* (minimum) expected project duration, but correspond to different buffer sizes which cause *different* expected total amounts of idle time. It is interesting that the expected project duration could remain at minimum, while lead-time buffers increase and idle time is being reduced. This shows that the relationships between *buffer size*, *total idle time* and *project duration* are neither necessarily direct nor linear. Thus, it is possible, for example, to introduce lead-time buffers to reduce idle time while not increasing project duration.

The best SQS-AL results from the 10 base cases in Table 2 correspond to a confidence level of 0.4 for all activities and yield project profit and project duration of ($50,493, 288 days). These results are very close to those from GA ($50,562, 285 days). The best CU-AL base case in Table 3 gives ($45,315, 307 days) for a lead-time buffer of 5 units. These results are not as close to those from GA ($50,478, 289 days). This is indeed true for other examples, as well. The optimum from SQS-AL base cases is very close to the result from GA, while the optimum from CU-AL base cases is not.

The following questions focus on the effectiveness of the sequence step algorithm (SQS-AL) and the completed unit algorithm (PUC-AL) in terms of maximum expected project profit, optimization, and efficacy of simulation modelling.

FOR WHICH ALGORITHM DOES GA WORK BEST?

The best result obtained from SQS-AL and GA is very close to that of CU-AL and GA. The two algorithms appear almost equally effective in terms of providing the optimal result if GA is used. In addition to the example shown in this paper, we have tested three other examples. Results from all four examples indicate that the differences in the optima obtained from both algorithms when using GA is less than 5%. However, the expected project duration from SQS-AL is always shorter than that from CU-AL. As explained below, one reason is the use of *integer* number of completed units in CU-AL which provides less resolution than the confidence level used in SQS-AL. In order to improve

the resulting project duration in CU-AL, it would be necessary to divide the repetitive units into smaller sub-units to reduce the expected duration of each sub-unit.

FOR WHICH ALGORITHM DO BASE CASES PROVIDE BETTER RESULTS?

The optimum results obtained from SQS-AL base cases and from GA are very close (less than 1% difference). The optimum results from CU-AL base cases are always less than those from GA by about 10%. More importantly, the expected project profits from all SQS-AL base cases are near the optimum and fluctuate much less than those from CU-AL. Thus, SQS-AL base cases always yield better and more reliable results.

There are two reasons for this. First, SQS-AL derives lead-time buffers directly from activity idle times. In contrast, CU-AL does not consider the existence or the magnitude of activity idle times and blindly delays each activity's start date by the number of completed units specified by the user. In other words, CU-AL causes unnecessary postponement because it keeps pushing each activity without taking idle time into account when determining lead-time buffers. Moreover, there are situations where CU-AL cannot eliminate idle time nor avoid unnecessary postponement as shown in Figure 3. Figure 3.1 shows that if BufferXY is 2 units, Y will have an idle time of 10 days. Figure 3.2 shows that if BufferXY is 3 units, then project completion is delayed by 20 days. Figure 3.3 shows the ideal solution that can be achieved by SQS-AL but not by CU-AL.

Second, the expected duration of each unit has a significant impact on the size of lead-time buffers and the results of CU-AL. For example, if all activity durations in Figure 3 are doubled, the idle time in Figure 3.1 will also double and become 20 days while the unnecessary postponement in Figure 3.2 will double and become 40 days. This drawback does not exist in SQS-AL.

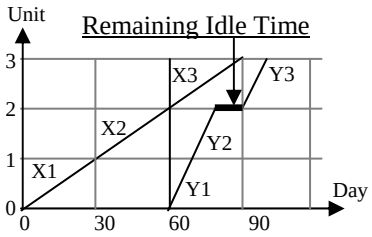

Figure 3.1: Delaying activity Y to 2 units of X incurs idle time in Y by 10 days.

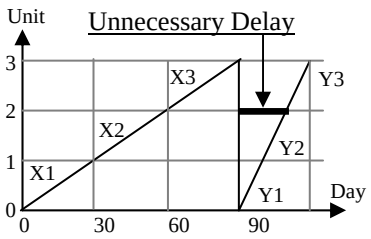

Figure 3.2: Delaying activity Y to 3 units of X incurs unnecessary delay by 20 days.

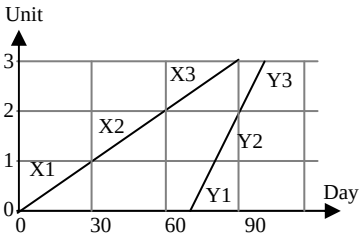

Figure 3.3: Delaying activity Y by 40 days from its early start date results the best solution.

The inaccuracies in CU-AL caused by the magnitude of activity durations per unit should be examined carefully. CU-AL is a very natural modeling approach for resource-based activity systems, such as Stroboscope. This problem may not exist if the project is modeled at a high level of detail (e.g., operational level) where expected durations for each unit are relatively small. However, studies that use an approach similar to CU-AL, but model the project with relatively large units (e.g., floors or road sections), should reconsider this particular problem. In particular, when CU-AL is used to obtain base case solutions, analysts should realize that the derived results may greatly deviate from the optimal solution, especially with respect to project duration.

Subdividing repetitive units is not without its own cost. As the number of repetitive units increases, so do the number of base cases for CU-AL. For example, if there are 100

units in each activity, 100 base cases are required for CU-AL in order to get the best result. This is not the case for SQS-AL as the number of base cases does not depend on the number of repetitive units. Usually, the ten base cases shown in Table 2 are enough.

CONCLUSION

The effectiveness of the sequence step algorithm (SQS-AL) and the completed unit algorithm (CU-AL) is investigated depending on assumptions, simplicity of simulation model, project characteristics, and uncertainty. In general, the advantage of SQS-AL is its ability to derive very effective lead-time buffers that eliminate idle time without unnecessary delay in project duration. However, SQS-AL requires longer computation time than does CU-AL. The advantage of CU-AL is its simplicity in simulation modeling and coding. The disadvantages of CU-AL are mainly two. It depends greatly on the user's level of effort to perform sensitivity analysis and find a solution close to the optimum, and its accuracy is vulnerable to the length of activity duration.

ACKNOWLEDGEMENTS

The authors would like to thank Rita Awwad and Sirarat Sarntivijai, graduate students at the University of Michigan, for their valuable comments on the content of this paper.

REFERENCES

Alarcon, L.F., and Ashley, D.B. (1999). "Playing Games: Evaluating the Impact of Lean Production Strategies on Project Cost and Schedule" *Proc. 7th Ann. Conf. Intl. Group for Lean Constr.*, IGLC-7, 26-28 July held in Berkeley, CA, USA, 263 – 173.

Alves, T.C.L. and Tommelein, I.D. (2004). "Simulation of Buffering and Batching Practices in the Interface Detailing-Fabrication-Installation of HVAC Ductwork" *Proc. 10th Ann. Conf. Intl. Group for Lean Constr.*, IGLC-12, August held in Ellsinore, Denmark, 13 pp.

Harris, R.B., and P. G. Ioannou. (1998). "Scheduling projects with repeating activities" ASCE, *J. of Constr. Engrg. and Mgmt*., 1998, 269-278, July/August Issue.

Ioannou P.G. and V. Likhitruangsilp. (2005). "Simulation of Multiple-Drift Tunnel Construction With Limited Resources", *Proc. 2005 Winter Sim. Conf.*, eds. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, IEEE, Piscataway, NJ, 1765.

Ioannou P.G. and C. Srisuwanrat. (2006). "Sequence Step Algorithm for Continuous Resource Utilization in Probabilistic Repetitive Projects", In *Proc. of the 2005 Winter Simulation Conference*, IEEE, Piscataway, New Jersey.

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

Sakamoto, M., Horman, M.J., and Thomas, H.R. (2002). "A Study of the Relationship between Buffers and Performance in Construction" *Proc. 10th Ann. Conf. Intl. Group for Lean Constr.*, IGLC-10, August held in Gramado, Brazil, 13 pp.

Tommelein, I.D. (1997) "Discrete-event Simulation of Lean Construction Processes" *Proc. 5th Ann. Conf. Intl. Group for Lean Constr.*, IGLC-5, 121 – 135.

Tommelein, I.D., Riley, D., and Howell, G.A. (1998). "Parade Game: Impact of Work Flow Variability on Trade Performance." *Proc. 6th Ann. Conf. Intl. Group for Lean Constr.*, IGLC-6, 13-15 August held in Guaruja, Brazil, 14 pp.