Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

OPTIMAL SCHEDULING OF PROBABILISTIC REPETITIVE PROJECTS USING COMPLETED UNIT AND GENETIC ALGORITHMS

Chachrist Srisuwanrat
Photios G. Ioannou

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

ABSTRACT

In this paper we introduce the *completed unit algorithm* (CU-AL), a probabilistic scheduling methodology for repetitive projects. The algorithm has two main advantages, simplicity and short computational time, that facilitate and expedite its use in simulation modeling and optimization. An integration between CU-AL and genetic algorithm (GA) is established to optimize the problem of maximizing profit for repetitive projects with probabilistic activity durations. This integration between CU-AL and GA is explained in detail through an example project with 5 activities and 10 repetitive units. A simulation model for this project is developed in Stroboscope, an activity-based simulation system. The optimization is performed by ChaStrobeGA, a Stroboscope add-on using genetic algorithm to optimize the overall objective function of project profit. Discussion of the results provides insight into the tradeoff between maintaining and relaxing resource continuity constraints in order to maximize expected project profit.

1 INTRODUCTION

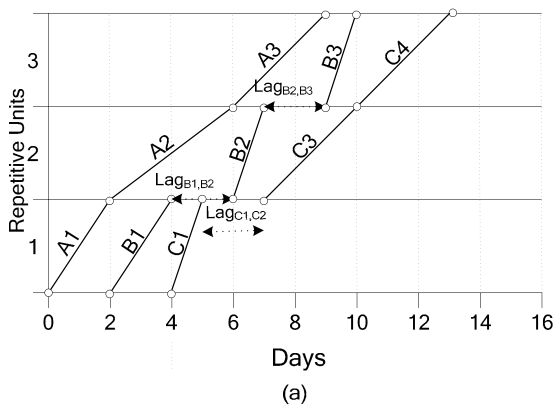
Repetitive projects where a series of activities repeat from unit to unit are common in construction. Examples of repetitive projects are high-rise buildings, housing projects, and highways, whose units are floors, houses, and road sections, respectively. The resources (crews) required by the various activities advance from one unit to the next, perform their work and move on until the last unit.

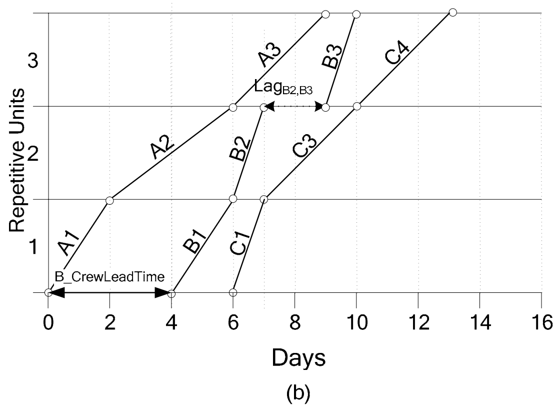
There are two main constraints that control the work sequence of resources, technological precedence and resource availability. Figure 1 shows two production diagrams for a repetitive project consisting of 3 units, each requiring 3 activities (A, B, and C). As shown in Figure 1a, technological constraints necessitate the completion of activity A (e.g., A1) before the start of activity B (e.g., B1) in the same unit, whereas resource availability constraints require activities in a preceding unit (e.g., A1) to finish before the start of the same activities in the next unit (e.g., A2). In Figure 1a, there are lags between B1 and B2, B2 and B3, and between C1 and C2. These lags imply that resources are ready to work, but the activities cannot start because their predecessors in the same unit have not been completed yet. In other words, resource availability constraints are satisfied, but technological constraints are not. Consequently, resources have to wait (be idle) until the predecessor activities are finished in that same unit. This period of waiting is called "idle time". The schedule in Figure 1a has a total idle time of 6 days: 4 days for resource B, and 2 days for resource C.

One of the main concerns when scheduling repetitive projects is how to eliminate idle time, i.e., how to keep resources working continuously without interruption. In construction, resources are usually paid from when they first arrive to the site, until they finish their work and leave. Thus, paying the cost of resources during waiting or idle time is unproductive and wasteful. In most cases, it is not cost effective to lay off resources and hire them back later because of availability issues and the cost and time associated with hiring-and-firing.

Therefore, a much more effective solution is to eliminate or minimize idle time by selectively delaying activity start dates. In Figure 1b, for example, the start date of activity B has been deliberately postponed to day 4 by using a lead time of 4 days measured from project start date (B_CrewLeadTime = 4 days). This decreases total project idle time from 6 to 2 days, without changing project duration. In Figure 1c, the start date of B has been postponed even more to day 6 (B_CrewLeadTime = 6 days). This eliminates idle time completely and provides continuous resource utilization, but also increases project duration from 13 to 15 days.

During the past 20 years, many graphical and numerical methods have been introduced to solve the problem of continuous resource utilization in repetitive project scheduling. These methods primarily focus on deterministic scheduling of repetitive projects. Examples of these methods are Line-Of-Balance (LOB), Linear Scheduling Method (LSM), and Repetitive Scheduling Method (RSM). According to these methods, there are two main approaches used to eliminate idle time: 1) balancing production rates (by changing methods or crew sizes) and 2) delaying activity start dates. Since perfect balancing of production rates is not always feasible or even an option, this paper focuses on the ever present problem of eliminating idle time by delaying activity start dates.





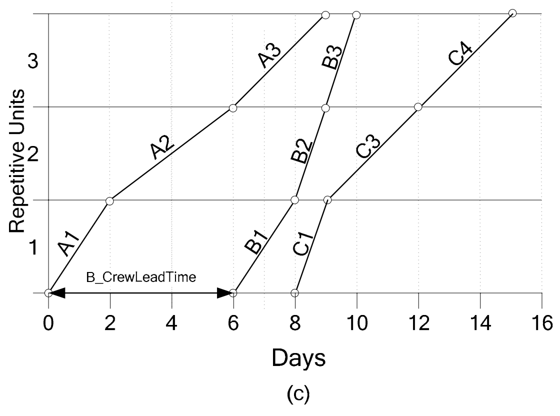


Figure 1: Delaying start date of activity B to eliminate its idle time

The concept of delaying activity start dates for the probabilistic scheduling of repetitive projects led to the development of the sequence step algorithm (SQS-AL) by Ioannou and Srisuwanrat (2006). SQS-AL was designed for repetitive projects with probabilistic activity durations, and provides a formal methodology for determining activity delays to achieve minimum project duration while achieving a given probability (confidence level) that activities will be continuous. SQS-AL uses simulation to determine the probability distributions of activity idle times and delays activities accordingly, sequence step by sequence step. The authors have shown that the algorithm is quite robust in that its effectiveness does not depend heavily on user input and judgment (Srisuwanrat and Ioannou 2007). However, SQS-AL does require significant computational time, especially for optimization problems where many alternatives must be tested.

One of the optimization problems in scheduling repetitive projects is the tradeoff between saving cost from eliminating idle time on one hand, and increasing cost due to increased project duration resulting from delaying activities' start dates on the other hand. This problem is about maintaining and relaxing resource continuity constraints in order to maximize project profit. As shown in Figure 1, it is obvious that Figure 1b with idle time of 2 days is better than Figure 1a with total idle time of 6 days, since project duration in both cases are the same. However, the decision of choosing between Figure 1b (idle time of 2 days and project duration of 14 days) and Figure 1c (zero idle time but project duration is 16 days) depends on the cost associated with idle time, interruptions, and project duration.

This paper introduces a probabilistic scheduling method called the *completed unit algorithm* (CU-AL). The algorithm uses the number of completed units in predecessors to delay their successors' start date which, in turn, reduce idle time in the successors. Since the algorithm does not perform the repetitive process of calculating idle time, delaying activities, and updating new idle time, the algorithm has two main advantages: 1) simplicity of simulation model and code, and 2) fast computational time. However, the algorithm relies on users' input (the number of predecessors' complete units) to eliminate or minimize idle time. Therefore, collaboration between CU-AL and a search methodology, Genetic Algorithm (GA), is necessary to facilitate and expedite the process of optimization. The integration between CU-AL and GA is explained in details through an example of repetitive project with 5 activities and 10 repetitive units. A simulation model for the example is developed in Stroboscope, an activity-based simulation system. The optimization is performed by ChaStrobeGA, a Stroboscope add-on using GA to optimize objective function. Results and discussions provide insightful tradeoff between maintaining and relaxing resource continuity constraints in order to maximize expected project profit.

2 COMPLETED UNIT ALGORITHM (CU-AL)

The completed unit algorithm (CU-AL) uses the number of completed repetitive units in predecessors to delay its successor's start date in order to minimize the idle time in the successor. The algorithm consists of 2 main steps: 1) determining specific successor's start date according to user-specified number of completed units in its predecessors, and 2) calculating idle time of the successor based on the determined start date from step 1. Figure 2 illustrates CU-AL in detail.

As shown in Figure 2, step 1 starts by specifying the number of completed units that are used to delay an activity. In this paper, the number of completed units that must be completed by predecessor before its successor can start the first unit is 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. After BufferXY for each activity is defined by the user, simulation model is executed for a certain number of replications, and average activity start dates are determined. It should be noted that these start dates collected from the replications are not early start dates; instead, they are the dates that their predecessors complete a certain number of units specified by the user.

Step 2 is similar to step 1, except that instead of using BufferXY, the lead time (X_CrewLeadTime) of each activity ,measured from project start date, is assigned to be the average start date determined in step 1. Note that BufferXY for each activity is now set to zero. During the replications performed in step 2, idle times of activities and project duration are collected to obtain the actual idle time and project duration corresponding to their lead times. At the end, the expected project profit is calculated from a given cost function, the expected idle time of each activity, and the expected project duration.

3 GENETIC ALGORITHM

The genetic algorithm (GA), developed by John Holland at the University of Michigan, is a search algorithm based on the mechanism of natural selection and genetics (Goldenberg 2004). This mechanism is based on the principle that strong creatures are most likely to survive, while weak creatures are most likely to become extinct. From one generation to another, survivors will reproduce offspring whose chromosomes are inherited from its parents. And occasionally, offspring genetically mutates. Inspired by this natural mechanism, GA is composed of 5

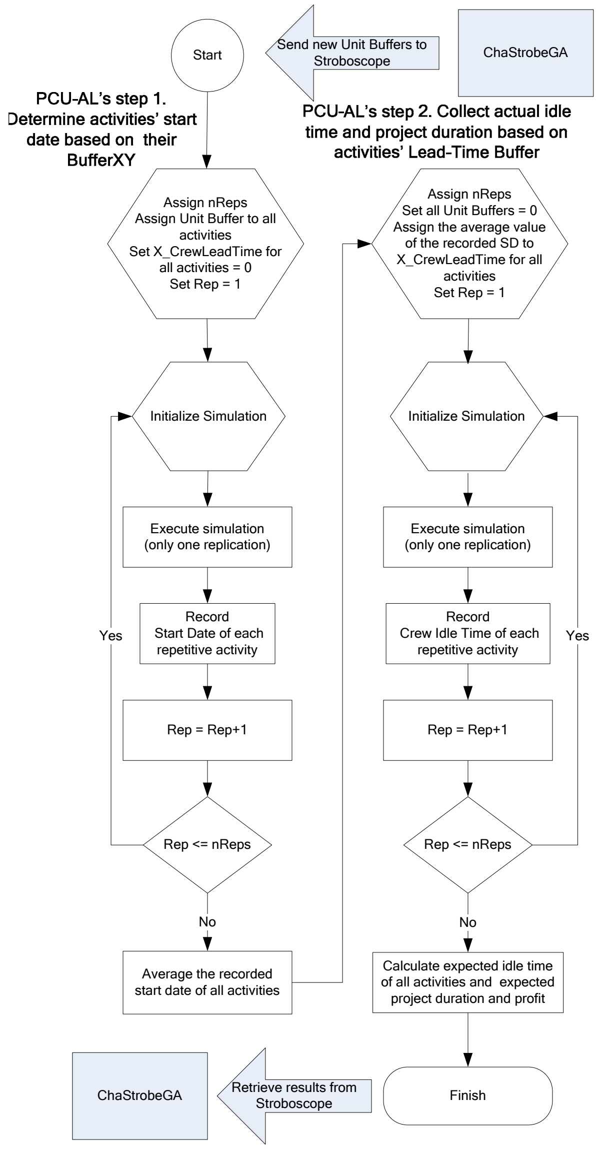

Start
Send new Unit Buffers to Stroboscope
ChaStrobeGA
PCU-AL's step 1. Determine activities' start date based on their BufferXY
PCU-AL's step 2. Collect actual idle time and project duration based on activities' Lead-Time Buffer
Assign nReps
Assign Unit Buffer to all activities
Set X_CrewLeadTime for all activities = 0
Set Rep = 1
Assign nReps
Set all Unit Buffers = 0
Assign the average value of the recorded SD to X_CrewLeadTime for all activities
Set Rep = 1
Initialize Simulation
Initialize Simulation
Execute simulation (only one replication)
Execute simulation (only one replication)
Record Start Date of each repetitive activity
Record Crew Idle Time of each repetitive activity
Yes
Yes
Rep = Rep+1
Rep = Rep+1
Rep <= nReps
Rep <= nReps
No
No
Average the recorded start date of all activities
Calculate expected idle time of all activities and expected project duration and profit
ChaStrobeGA
Retrieve results from Stroboscope
Finish

Figure 2: CU-AL workflow diagram

main processes: initialization (creating the first generation), evaluation (determining chance of surviving), reproduction (producing offspring), crossover (exchanging chromosomes inherited from parents), and mutation (alteration of genetics encodings inside the chromosomes). Figure 3a demonstrates the steps of a genetic algorithm using as an example a repetitive project consisting of 10 units requiring 5 activities in each unit. The activity-on-node network for the example is shown in Figure 3b.

In optimization, an individual (or a member) is a composition of chromosomes (or decision variables of interest) that results in a particular outcome value using an objective function. Here, an individual or member is a possible schedule. Each chromosome is a decision variable whose value may vary within its domain. In this study, the decision variables are the number of predecessors' completed units (BufferXY), and their domain of values are the integers from 1 to 10 (i.e., the total number of repetitive units).

As shown in Figure 3c, each individual contains 4 decision variables: BufferAB, BufferBC, BufferCD, and BufferDE. Since each activity has 10 units, each decision variable has domain values from 1 to 10 in an increment of 1. Accordingly, before starting optimization, users must specify the following:

- Domains for all decision variables
- Total number of generations
- Total number of members (individuals) per generation (i.e. population)
- Probabilities of crossover
- Probabilities of mutation

In this study, the above parameters are ChaStrobeGA's input variables, defined prior to the first GA process, which is initialization.

Initialization is the process of creating the first generation that does not have a preceding generation. To create the first generation, ChaStrobeGA uses uniform distributions to assign random values in the decision variables from their domain.

For the example in Figure 3, we choose to have 4 members (or 4 cases) in each generation. ChaStrobeGA randomly selects 4 values of decision variables for each of the 4 members. After the values are selected, parameters of each member (or each case study) were coded and modeled in Stroboscope.

Simulation execution based on the selected decision variables of a member involves 3 steps: 1) generating a programming script for the member's decision variables in Stroboscope language, 2) executing the simulation code, and 3) getting results from the simulation. In Figure 3d, the outcome from simulation is an assessment of expected project profit. As shown in Figure 3d, the first member with BufferAB=3, BufferBC=6, BufferCD=2, and BufferDE=4, results in an expected project profit of 123. These steps are repeated until the simulations for all members are executed. ChaStrobeGA will then start the evaluation process.

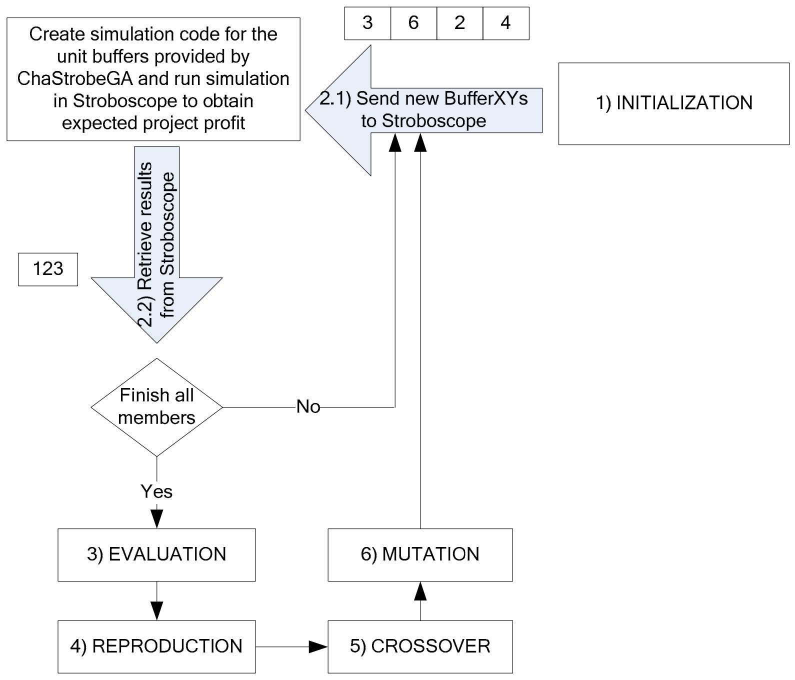
Evaluation is the process of calculating the chance of survival based on member's relative performance in the same generation. Relative performance is the ratio between the outcome from the simulation of an individual member and the sum of all outcomes in the same generation. The larger the ratio, the better chance the member has to survive. Figure 3e uses project profit to show the relative performance of each member as a percentage. Relative performance determines which member will survive to the next generation through reproduction.

Reproduction is the process of selecting members for the next generation by using a biased roulette wheel whose slot sizes are proportional to the members' relative performance. As shown in Figure 3f, the wheel is spun 4 times to get 4 members for the new generation. As shown in Figure 3g, member No 2 with relative performance of 40% is selected twice (from Figure 3f), whereas member No 3 and No 4 with relative performance of 36% and 15% respectively are each selected once. After reproduction finishes, the crossover process begins.

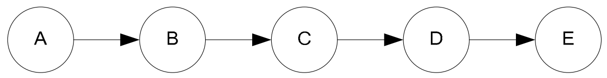
Crossover is the process of exchanging values of decision variables between members. This process consists of two steps. First, two members are randomly selected. Then, the probability of crossover (given by the user) is used to determine whether to continue the crossover operation for this pair. If the result is to proceed, one of the positions of decision variables will be selected at random. Then, the values of decision variables in that position between the two members are swapped. Figure 3h shows how members No. 1 and No. 2 are paired and values for the decision variables in the forth position are swapped.

Mutation is the process of altering decision variables to protect premature loss of important notions (Goldberg 2004). In other word, the mutation process ensures that results from GA are not limited to any specific local search. This process starts by using the mutation probability (given by the user) to determine whether to mutate decision variables of a member. If the decision is to mutate, one of the decision variables of the member will be selected at random and substituted by one of the parameters in that decision variable's domain. Figure 3i shows that the decision variable of member No. 2 in the second position is mutated, as well as the decision variable of member No. 4 in the forth position. After GA finishes the mutation operation, a new generation is ready for simulation.

The processes of simulation execution, evaluation, reproduction, crossover, and mutation are repeated until the total number of generations specified by the user is reached. Then, the results from all generations are assembled and arranged in one table and sorted by objective function (i.e., project profit), to derive the optimum solution according to GA.


3 6 2 4
Create simulation code for the unit buffers provided by ChaStrobeGA and run simulation in Stroboscope to obtain expected project profit
2.1) Send new BufferXYs to Stroboscope
1) INITIALIZATION
2.2) Retrieve results from Stroboscope
123
Finish all members
No
Yes
3) EVALUATION
6) MUTATION
4) REPRODUCTION
5) CROSSOVER

(a)



(b) An example of repetitive project consisting of 10 units requiring 5 activities

1) INITIALIZATION

No	Buffer				$	%
	AB	BC	CD	ED		
1	3	6	2	4		
2	7	2	5	3		
3	6	1	9	4		
4	2	2	5	10		

(c) Create the first generation

2) SIMULATION

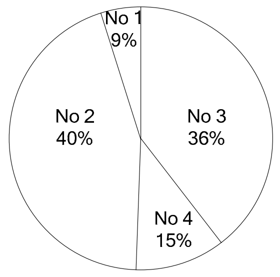
No	Buffer				$	%
	AB	BC	CD	ED		
1	3	6	2	4	123	
2	7	2	5	3	…	
3	6	1	9	4	…	
4	2	2	5	10	…	

(d) Send data and retrieve results from Stroboscope

3) EVALUATION

No	Buffer				$	%
	AB	BC	CD	ED		
1	3	6	2	4	123	9
2	7	2	5	3	541	40
3	6	1	9	4	484	36
4	2	2	5	10	210	15

(e) Calculate relative performance of each member



(f) Biased roulette wheel whose slot sizes are proportional to member's relative performance

4) REPRODUCTION

No	Buffer				$	%
	AB	BC	CD	ED		
1	7	2	5	3		
2	2	2	5	10		
3	6	1	9	4		
4	7	2	5	3		

(g) Select member by using bias roulette wheel

5) CROSSOVER

No	Buffer				$	%
	AB	BC	CD	ED		
1	7	2	5	**10**		
2	2	2	5	**3**		
3	6	1	9	4		
4	7	2	5	3		

(h) Pair members and exchange their parameters

6) MUTATION

No	Buffer				$	%
	AB	BC	CD	ED		
1	7	2	5	10		
2	2	**7**	5	3		
3	6	1	9	4		
4	7	2	5	**5**		

(i) Randomly select one decision variable and alter it value

Figure 3: Genetic Algorithm and Collaboration between CU-AL and GA

4 OBJECTIVE FUNCTION AND COST MODEL

Project profit is used as the objective function to evaluate the combined effectiveness of the two algorithms CU-AL and GA, and to study the effect of maintaining and relaxing resource continuity constraints. The standard objective function for a lump sum contact is:

```
Project Profit = Contract Price - Direct Cost -
                 Indirect Cost
```

The direct and indirect costs consist of fixed and variable costs. Moreover, there are costs associated with idle time. The contract price minus the fixed direct costs is assumed constant ($280,000), whereas unproductive direct cost is linear function of resource idle time. The indirect cost consists of two parts: a fixed project setup of $20,000 and a variable cost of $50 per day. The objective function is shown below.

Project Profit ($) = 280,000 – Unproductive Direct Cost – Indirect Cost

Unproductive Direct Cost ($) = CIT_B * 100 + CIT_C * 120 + CIT_D * 60 + CIT_E * 90

Where, *CIT = Crew Idle Time* in days

Indirect Cost ($) = 20,000 + Project Duration (in days) x $ 50/day

5 EXAMPLE PROJECT

An example of a repetitive project consisting of 5 activities performed over 10 non-identical units is used to demonstrate the application of the completed unit algorithm. Figure 3b is the activity-on-node network for each repetitive unit for the example. Table 1 shows work amounts for activities 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%. The example simulation model and the completed unit algorithm are modeled in Stroboscope. The number of replications is 3000 for each CU-AL step.

The function of project profit shown in the previous section is optimized by using a genetic algorithm (GA) as the search methodology. The decision variables are the number of predecessor completed units for each precedence link, BufferAB, BufferBC, BufferCD, and BufferDE. The value domain for these variables includes the integers from 1 to 10 (the number of repetitive units). The Stroboscope add-in ChaStrobeGA is used to implement the GA for maximizing expected project profit.

For the example presented here, the number of generations and the number of members per generation are both 30. The probabilities of crossover and mutation operations are 0.6 and 0.05 respectively.

6 DISCUSSION OF RESULTS

Table 2 shows the simulation results for the example project. They are grouped into two categories, the results from ten base cases, and the optimal results from 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. The ten base cases for this example use the same number of predecessor completed units for all activities as shown in the first rows of Table 2. Results from GA are shown in the last row, which is the best solution according to the algorithm with the given GA parameters.

In Table 2, the first base case using BufferXY = 1 for all activities is similar to scheduling activities at their early start date. Each activity is scheduled to start after its predecessor completes work in the same unit. This means that without delaying any activities, this project would take a minimum 1,049 days to complete, with an expected profit of $23,046. It should be noted that the associated total idle time is quite large at 1,989 crew working days.

The first three results in Table 2 indicate that delaying activity start date reduces total idle time significantly (from 1,989 to 1,394 and to 971 days) without increasing the expected project duration (which is about 1,050 days). This indicates clearly that the introduction of lead time (to decrease idle time) does not necessarily lengthen project duration. Hence, it is possible to introduce lead times (X_CrewLeadTime) to delay certain activity start dates which, in turn, reduce idle time while not increasing project duration.

Clearly, the effect of lead times on project duration

Table 1: Crew Production Rates and Activity Work Amount In Each Repetitive Unit

Activity	Mean	SD	Unit									
			1	2	3	4	5	6	7	8	9	10
			Work Amounts									
A	10	1.0	1200	1000	600	800	800	1000	800	1000	1200	600
B	20	2.0	600	400	800	600	600	800	400	600	800	800
C	15	1.5	800	600	200	800	200	400	800	800	600	400
D	15	1.5	600	800	400	600	600	600	600	400	400	600
E	25	2.5	400	600	200	400	800	600	600	200	400	800

Table 2: Expected Project Profit and Duration, Idle Time, and Lead Time (Coefficient of Variation, V= 10%)

Buffer				Expected Project Profit	Expected Project Duration	Idle Time (in Crew Working Days)				Lead Time (X_CrewLeadTime)			
AB	BC	CD	DE			B	C	D	E	B	C	D	E
1	1	1	1	23046	1049	505	448	434	602	121	151	205	246
2	2	2	2	76497	1050	405	328	274	387	222	272	367	461
3	3	3	3	114636	1049	343	226	158	244	283	374	482	603
4	4	4	4	161101	1067	262	114	10	56	364	486	647	809
5	5	5	5	180079	1184	182	15	0	9	444	596	771	973
6	6	6	6	182702	1384	81	0	0	0	546	738	940	1182
7	7	7	7	179787	1579	13	0	0	0	627	839	1095	1377
8	8	8	8	170424	1791	0	0	0	0	727	970	1280	1590
9	9	9	9	158952	2021	0	0	0	0	849	1132	1482	1819
10	10	10	10	150553	2189	0	0	0	0	910	1233	1610	1987
7	1	1	6	200176	1158	12	2	7	0	627	657	711	956

Table 3: Expected Project Profit and Duration, Idle Time, and Lead Time (Coefficient of Variation, V= 20%)

Buffer				Expected Project Profit	Expected Project Duration	Idle Time (in Crew Working Days)				Lead Time (X_CrewLeadTime)			
AB	BC	CD	DE			B	C	D	E	B	C	D	E
1	1	1	1	14118	1089	523	465	451	624	126	158	214	256
2	2	2	2	70084	1088	418	338	284	401	231	283	381	478
3	3	3	3	109125	1089	355	234	165	254	293	388	500	625
4	4	4	4	154022	1120	271	120	28	75	376	502	670	836
5	5	5	5	173691	1235	189	33	2	18	460	617	798	1008
6	6	6	6	178511	1437	89	3	0	5	565	763	972	1223
7	7	7	7	175410	1635	28	0	0	0	649	869	1133	1426
8	8	8	8	166671	1858	4	0	0	0	756	1007	1328	1649
9	9	9	9	155174	2095	0	0	0	0	880	1173	1536	1886
10	10	10	10	146667	2266	0	0	0	0	942	1277	1667	2057
7	2	1	6	192587	1232	29	9	21	7	650	702	758	1016

depends on many factors such as the characteristics of network, activity productivity and its variability, and the set of activities to be delayed.

As activities are pushed forward in the first six base cases in Table 2, expected profit increases from $23,046 (with BufferXY=1) to $182,702 (with BufferXY=6). However, as activities are pushed beyond 6 units, the expected profit decreases. A comparison of row 6 (the best expected project profit from the base cases) and the last row indicates that the GA results provide greater project profit with shorter project duration. Based on the example in this paper and another 3 examples, which are not shown here, expected project profit from GA is greater then that of base cases by 5% to 10% approximately.

The GA optimal solution is to use BufferAB=7, BufferBC=1, BufferCD=1, and BufferDE=6 which yield an expected project profit of $200,176 with expected project duration of 1,158 days. As shown in Table 2, the GA optimal solution results in 21 days of expected idle time. I.e., it does not eliminate idle time. This means that *in order to achieve maximum project profit, it is necessary to relax the resource continuity constraints*.

Table 3 shows similar results assuming that activity production rates have a coefficient of variation of 20%. A comparison between Table 2 (10% coefficient variation) and Table 3 (20% coefficient variation) shows that variability in productivity effects idle time, project duration, and project profit. As expected, an increase in variability increases idle times and hence lead times, and results in lower project profit and longer project duration. The number of predecessor completed units has to be increased in order to minimize the increased idle time caused by the increased variability. Similarly, the GA optimal solution in Table 3 yields less profit ($192,587) than the GA optimal solution in Table 2 ($200,176).

7 SUMMARY AND CONCLUSIONS

This paper introduced the *completed unit algorithm* (CUAL), a probabilistic scheduling method for repetitive projects. Its simplicity and fast computational processing expedites the time required to execute simulation models and facilitates optimization through a genetic algorithm.

An integration of CU-AL and the associated GA was developed within the Stroboscope simulation system and was used to solve several examples for this study. An example project with 5 probabilistic activities that repeat over 10 non-identical units was presented in detail to illustrate the tradeoff between the costs due to idle time and increased project duration. The deliberate introduction of lead times reduces resource idle time and the associated cost but may also increase project duration and indirect cost. The example results illustrate that both maintaining and relaxing resource continuity constraints must be employed in order to arrive at an optimum solution.

ACKNOWLEDGEMENTS

The writers would like to thank Sirarat Sarntivijai and Rita Awwad, graduate student at the University of Michigan for her valuable comments.

REFERENCES

Goldberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley Longman, Inc., Boston, ISBN 0201157675, 1-65.

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

Ioannou, P.G., and Likhitruangsilp, V. 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. Arm-strong, and J.A. Joines, 1483-1491.

Ioannou, P.G., and Srisuwanrat, C. 2006. Sequence Step Algorithm for Continuous Resource Utilization in Probabilistic Repetitive Project, In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1731-1740.

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.

Srisuwanrat, C., and Ioannou, P.G., 2007. The Investigation of Lead-Time Buffering Under Uncertainty Using Simulation and Cost Optimization, In *Proceedings of the 15th International Group of Lean Construction* .

AUTHOR BIOGRAPHIES

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 < csrisuwa@umich. edu.>

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.engin. umich.edu/cem/Ioannou>.