Proceedings of the 1987 Winter Simulation Conference
A. Thesen, H. Grant, W. David Kelton (eds.)

DELAY PERT

Donovan Young
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205, U.S.A.

ABSTRACT

This white paper suggests a PERT simulation method applicable to critical-path project scheduling networks subject to correlated negative-exponential delays. "Delay PERT" starts with a declared schedule, in which each activity has an intended duration and an intended start time. Durations are subject to delays assumed to obey negative-exponential distributions; fixed start times, whose primary purpose is to model exogenous events such as procurements, are similarly subject to negative-exponential delays. Correlations among delays are expressed by partitioning delays into classes, and as each delay is realized in a simulation experiment the expected values are statistically updated for all the unrealized delays of the same class.

The need for "delay PERT" arose in beta testing of GITPASE, an interactive project scheduling package. This paper describes the background, shows the need for "delay PERT", gives rationales for its assumptions and statistical methodology, proposes a data structure, proposes an organization of the simulation experiments, and indicates outputs and how users might utilize them.

1. BACKGROUND

GITPASE [8] is a resource-oriented project management tool whose scheduling capabilities allow a user interactively to derive a schedule that will keep consumptions within availabilities for multiple constrained resources. Such a schedule is normally a longer one than the standard critical-path schedule that would consider only durations and precedences, because some activities may have to await resource availabilities. One GITPASE data item (among others) that facilitates expressing intentional or exogenous postponements is the fixed start time. If a fixed start time is assigned to an activity, that time overrides the activity's early-start computation; thus, for example, if the activity's predecessors were scheduled earlier or given shorter durations, the activity would not automatically be scheduled earlier. Because GITPASE is resource-oriented, the common use of fixed start times is to convert resource constraints into time constraints, i.e., to cause an activity to be scheduled at the time when resources will be available. However, a fixed start time can also be used to represent an exogenous event not represented via an explicit milestone or activity within GITPASE; for example, if an activity could begin only when a procurement was completed, the due date for the procurement could become the fixed start time for the activity, and the procurement itself would not need to be modeled. See Table 1.

A PERT simulation system for GITPASE is currently under development [2]. Like GITPASE itself, the simulation system is resource-oriented. Whereas ordinary PERT methods examine the consequences of probabilistic variation in activity durations, this simulation system varies resource consumptions, resource availabilities, timing of changes in availability levels, and exogenous events as well.

Various versions of GITPASE have undergone beta testing to manage R&D projects in automated manufacturing, oil refinery maintenance turnarounds, logistics software development projects, and now a communications installation project in which many contractors must coordinate to install various subsystems in a large military communications center. This project has pointed up the need for a PERT simulation capability that would have an interesting special structure and would not be simply a subset of the PERT simulation system already under development.

In the communications installation project the activities represent work by contractors to install parts of the system. Contractors bring their own resources. With the exception of an interesting application of access as a resource (to avoid scheduling too many people to work in a confined space at the same time, access to a confined area can be tracked as a resource that is "consumed" by activities), resource contention is of little real importance in this project. The schedule is effectively "driven" by procurement actions. Now a procurement cycle can be handled by GITPASE as a separate network or as a group of activities in the same network that represents construction activities, but in reality little procurement modeling is appropriate. Given existing management procedures, it would be redundant to force GITPASE to output such things as when bids need to be issued. Practically the only effect of procurement on the construction schedule is the effect of delivery delays. The due date for a procurement is a given element in the construction schedule, and anything the owner could have done to affect it is past history. Early deliveries are rare and unexploitable. Delays are common and disastrous, yet the most often encountered delay is zero, and it would be confusing for the schedule to be published and discussed if its milestone dates included fixed delay allowances.

Delays are believed to be negative-exponentially distributed, and are believed to be positively correlated, some projects being riddled with them while others remain relatively free of them. Each procurement delay (and each construction delay as well) is viewed as a harbinger of possible further delays for events of the same type.

Table 1: Deferments of Intended Activity Start Times in GITPASE

Deferment Mechanism	Effect on Schedule	Data Level	Typical Intent
define predecessor activity	provide early limit on start time	Network	express requirements intrinsic to work breakdown
define serialization quasi-predecessor	provide early limit on start time	Plan	spread out resource consumptions
define predecessor milestone	provide early limit on start time	File	express requirements intrinsic to work breakdown, regarding work modeled in a different network
fix start or finish time	fix start time	Plan	adjust to resource availabilities, or express requirements regarding work not modeled

A need was recognized for a PERT simulation system that would take an intended schedule as input and would simulate in such a way that delays would have some autocorrelation, and would output the usual PERT simulation statistics - estimated distributions of milestone times (especially final completion time) and estimated probabilities that each procurement and activity will be critical in terms of being on the path that determines final completion time.

2. DELAY DISTRIBUTIONS

The negative-exponential distribution has a lower limit and mode of zero and is strictly decreasing.

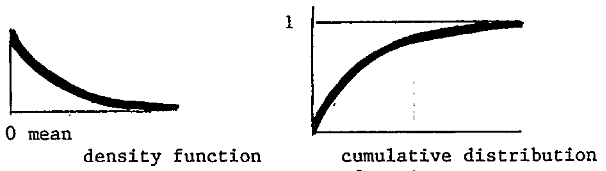

Figure 1: The Negative-Exponential Distribution

It is often associated with delays; for example, it is the distribution most used to model delays in queueing theory [1]. It is the maximum-entropy distribution for delays, that is, it is the distribution of choice if nothing is known about a delay except its expected value [7]. It is the distribution (as the continuous analog of the geometric distribution) that follows if we model final completion of a process as a series of stabs at closure, each stab having a probability of success [6]. It is the distribution that approximates the tail of most distributions. For example, if we model a process as a series of several tasks performed

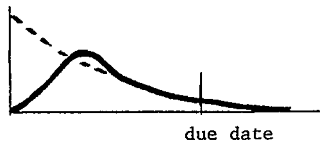

Figure 2: The Negative Exponential Distribution as an approximation of the Tail of a Gamma Distribution

in succession, each task having a negative-exponential duration, we obtain the gamma (Erlang) distribution; if the number of tasks is large (say 20 or more) this distribution tends to the normal Gaussian distribution [3]. In either case, the negative-exponential distribution closely fits the right tail. Suppose the process is a procurement, and suppose the vendor agrees to a due date such that there is a small probability the due date will be exceeded. Then the delay, defined as exceeding the due date, will have a distribution practically indistinguishable from the negative-exponential distribution.

Therefore we adopt this distribution as the "delay PERT" distribution. Let the ith delay process have an estimated average delay of d_i time units. Then, for $t > 0$, the probability of a delay between t_1 and t_2 time units is

$$P\{t_1 < t < t_2\} = \int_{t_1}^{t_2} f(t)dt \qquad (1)$$

where

$$f(t) = \frac{1}{d_i} e^{-t/d_i} \qquad (2)$$

Equation 2 gives the density function for the ith delay process. Equation 3 gives its cumulative distribution function:

$$F(t) = P\{\text{delay} < t\} = \int_0^t f(\tau)\, d\tau$$

$$F(t) = 1 - e^{-t/d_i} \qquad (3)$$

Hence the probability of a delay between t_1 and t_2 can be computed as

$$P\{t_1 < t < t_2\} = F(t)_2) - F(t_1) = e^{-t_1/d_i} - e^{-t_2/d_i} \qquad (4)$$

For example, if the expected delay is 2.0 weeks, the probability of a delay of up to 1.0 week is F(1) = 0.39347, the probability of a delay over 2.0 weeks is 1-F(2) = 0.36788, and the probability of a delay between 1.0 and 2.0 weeks is F(2)-F(1) = 0.23865.

3. POSITIVE CORRELATIONS AMONG DELAYS

If N delays were known to come from the same distribution, their average would be the estimator for d_i, and a Bayesian sequential updating scheme giving a sequence of inverted Beta-2 distributions is known to generate the appropriate sequence of estimators [5]. To avoid notational complication, let us assume all observations are from the same class and that all delays in the same class are from the same population. Since the parameter d_i is also the mean delay, we can denote $\tilde{x}_n$ as the current estimate of d_i, x_n as the current observation of t_i, $\tilde{x}_{n-1}$ as the previous estimate, and x_{n-1} as the previous observation. With this notation the Beta-2 sequential updating scheme gives successive estimates upon the n-1 and nth observations as follows (assuming no initial estimate):

$$\tilde{x}_{n-1} = \frac{1}{n-1} \sum_{j=1}^{n-1} x_j \quad \text{and} \quad \tilde{x}_n = \frac{1}{n} \sum_{j=1}^{n} x_j \qquad (5)$$

By rearrangement we can express the nth estimate in terms of the nth observation and the n-1st estimate:

$$\tilde{x}_n = \left(\frac{1}{n}\right) x_n + \left(\frac{n-1}{n}\right) \tilde{x}_{n-1} \qquad (6)$$

It is usual in Bayesian updating sequences to start with input parameters that are equivalent to an initial estimate and a measure of confidence in it; without loss of generality we can assume that the initial parameters were equivalent to an initial estimate $\tilde{x}_m$ and a weight m, such that the influence of the initial estimate is the same as if there were m observations of $\tilde{x}_m$, followed by actual observations $x_{m+1}, \ldots, x_n$, so that Equation 6 still holds, with n being the sum of the number of observations and m the weight on the initial estimate.

Now if we believed that all the delays for some class of events, say procurement completions, came from a single negative-exponential population, we could use Equation 6 to express positive correlation among delays. The simulation of the project would run until the first delay was observed; this delay, x_{m+1}, would be drawn from a population having as d_i the initial estimate $\tilde{x}_{m+1}$ from x_{m+1} and $\tilde{x}_m$, and the next delay would be drawn from a population having as d_i the revised estimate $\tilde{x}_{m+1}$. The resulting series of delays would be positively autocorrelated. A particularly long or short delay would tend to induce further delays to be somewhat longer or shorter than otherwise. Control over the amount of autocorrelation would be exercised by selection of m, the number of observations to which the initial estimate is equivalent. For very large m, observations would be essentially independent; for very small m, observations would be very dependent, and at the extreme of m=0, each delay would be drawn from a population having as d_i the mean of the previous delays.

For situations such as that of the communications installation project that motivates this development, Equation 6 is not satisfactory because it weighs all past delays equally in estimating future delays. It is felt that more recent delays should be weighed more heavily, because it is felt that the population of delays gradually drifts. The widely accepted methodology for such a situation is exponential smoothing. Given an estimate, a new observation and a smoothing constant α, the new estimate is a weighted average of the new observation and the estimate. Thus for our situation, exponential smoothing would give:

$$\tilde{x}_n = \alpha\, x_n + (1-\alpha)\, \tilde{x}_{n-1}\,, \qquad 0 < \alpha < 1 \qquad (7)$$

Comparing Equations 7 and 6, we see that they differ only in that the weight on the current observation is constant for exponential smoothing, whereas it declines for Bayesian updating so that all observations have the same weight. For exponential smoothing, older observations have smaller weights. To make this clear we can rewrite Equations 6 and 7 to express the estimate as a weighted average of observations:

3.1. Bayesian Updating

$$\tilde{x}_n = \frac{1}{n}\, x_n + \frac{1}{n}\, x_{n-1} + \dots + \frac{1}{n}\, x_m \qquad (8)$$

3.2. Exponential Smoothing

$$\tilde{x}_n = \alpha\, x_n + \alpha(1-\alpha)x_{n-1} + \alpha(1-\alpha)^2 x_{n-2} + \dots \quad + \alpha\Big(\sum_{j=1}^{n-m}(1-\alpha)^j\Big)x_m \qquad (9)$$

Control over the amount of autocorrelation given by exponential smoothing is exercised by setting the smoothing constant. For α=0 there is no autocorrelation; all delays are drawn from the population having $d_i = x_m$, the initial estimate. For α=1 there is maximal autocorrelation; each delay is drawn from a population having d_i equal to the previously observed delay. For intermediate α there is intermediate autocorrelation; with $\alpha = 0.1$, for example, the new d_i is one-tenth of the way between the previous d_i and the last observed delay.

In "delay PERT" the user will be asked to input a smoothing constant and an initial delay estimate for each class of events. The first delay for an event class will be drawn from a negative-exponential population with the initial delay estimate as its mean; subsequent delays for the class will be drawn from a negative-exponential population whose mean is updated by Equation 7.

3.3. Statistical Scaling

According to the above development, each event in a given class has the same expected delay, and observations of delays affect only delays in the same class. If a project is considered to have, for example, delay classes of 1 = "heavy construction", 2 = "minor procurement", and 3 = "major procurement," then the expected delay for all major procurements must be identical, and excessive delays in minor procurements would not tend to predict greater delays in major procurements.

Classes are partitioned, then, not only on the basis of mutual predictive power (those influencing each other are in the same class) but also on the basis of scaling: for example, if one vendor had a poor delivery record while another had a good one, they would have to be in different classes, and if some exogenous condition (e.g. a transportation strike or scarcity of resources) were likely to affect both of them the resulting correlation would be ignored because of their being in different classes.

There is a natural scaling for negative-exponential variables that could allow mutually predictive events to be in the same class even if their delay magnitudes were different. From Equation 3 we see that all delays having the same t/d_i(relative delay) have the same probability. This raises the possiblity of scaling all delays in a class and applying smoothing to relative delays, which can be done implicitly as follows:

We keep a separate current estimate for each delay in the class. When an observation is made, Equation 7 is solved for that particular delay. This updates that particular estimate (which might appear to be irrelevant since the same delay will not occur again in the same simulation). Let r be the ratio of the new estimate to the old estimate. We update the other estimates in the same class by multiplying them by r.

For example, suppose the procurements of items 16 and 17 are in the same class, that the current expected delay for item 16 is 2.00 weeks and for item 17 is 3.00 weeks. Let α=0.1. Now the procurement of item 16 is observed (simulated) to have been delayed 2.2 weeks. Equation 7 for item 16 yields 2.02 weeks. r = 2.02/2.00 = 1.01. The delay for item 17 is now estimated as 3.00r=3.03 weeks.

Not all users of "delay PERT" are expected to use this feature. Therefore the input protocol will be such that the user normally inputs an initial delay estimate only for the first member of each class and the same estimate is assigned to all other members by default, but the user may specifically revise the initial delay estimate for any member or members.

4. DELAY-PERT DATA STRUCTURE

A delay-PERT simulation study consists of one or more runs, each consisting of a user-specified number of experiments. The results of each run, alone or in combination with previous runs in the same study, consist of summary statistics on network completion delay, a completion-criticality list telling in what proportion of experiments each event was critical, summary statistics on delay of events and/or classes of events selected by the user individually and/or by selection of a criticality threshhold, and a data echo. Each study starts with a "declared Plan" (an intended schedule) that is time-feasible, and the results of a study can be interpreted as indicating likely delays in implementation compared to the Plan.

There will be a new menu block DLY (for "delay-PERT) in the auxiliary menus of appropriate GITPASE screens. Touching it will bring up a new screen - the Delay-PERT screen - for whatever Plan is currently loaded. The Delay-PERT screen will manage interactive specification of simulation studies, interactive viewing of simulation results, and user control of printed simulated results.

4.1. Simulation Curtain

The first period covered by a run of simulation experiments is called the simulation start period, and is user-specified. It can be any period later than the latest firm status report period; that is, the simulation curtain (boundary between the simulation start and the preceding period) cannot be earlier than the status curtain.

Everything before the simulation curtain is assumed to have occurred exactly as scheduled and/or reported. Everything after the simulation curtain is subject to PERT delay. If an activity crosses the curtain, its remaining duration is subject to PERT delay but its continuation is immediate - there is no "start" delay for the beginning of the after-curtain part of an activity.

Recall that with offset precedences a precedence point is defined as the boundary between a specified period associated the predecessor's schedule (e.g. 80% completion) and a specified period associated with the successor's schedule. In the definitions of direct and indirect delays to follow, the location of the precedence point relative to the simulation curtain is important.

4.2. Direct and Indirect Delays

A direct delay is one for which the user can input a mean ($\bar{x}_i$) to cause simulated delays. For example, if the user enters "3.5" the simulation experiments will draw random variates for the delay from a negative-exponential population with a mean of 3.5 periods. An indirect delay is one that follows, through precedences and quasi-precedences in the Plan, from direct or indirect delays. The direct delays drive the simulation, while the indirect delays follow along as consequences.

Direct delays include activity durations and fixed start times:

Direct Delays

- duration delay for any activity, except a dependent-duration activity (DDA).
- start delay for any activity having a fixed start time, except a DDA. A successor with no predecence point after the simulation curtain, or one with a fixed finish time, is considered for PERT purposes to have a fixed start time.
- milestone delay when the network is a dependent network for that milestone, and a milestone time exists, or when the milestone is fixed in the network (necessarily the milestone's determinant network).

Direct delays are those that are not an automatic consequence of other delays in the same network; these automatic consequences are called indirect delays.

Duration delays compare to ordinary PERT durations as follows: whereas in ordinary PERT the activity durations are considered to be PERT-beta random variables characterized by a three-point estimate (optimistic, most likely, and pessimistic durations) and the ordinary PERT schedule is defined in terms of a network having all durations at their expected times, in delay PERT the activity durations are considered to be negative-exponential random variables characterized by a single expected-delay estimate and the schedule is defined in terms of the network having no delays.

Using duration delays only (not considering start delays or milestone delays), we can carry out the equivalent of a PERT simulation. Say for some activity the PERT-beta estimates are a, m and b (optimistic duration, most likely duration and pessimistic duration). Let the same activity be characterized in delay-PERT notation by an intended duration x and an expected delay d. Then if we set a=m=x and b=x+6d the correspondence is particularly close; in both simulations the minimum and most likely duration is x, the expected duration is x + d, and the variance of duration is d^2. (To derive these results, recall that the PERT-beta expected duration is (a+4m+b)/6 and its variance is $(b-a)^2/36$ [4].)

Whereas in ordinary PERT the schedule used for discussion is that with expected activity durations, in delay-PERT the schedule used for discussion is the intended (no-delay) schedule. (We could define the expected-delay schedules as in ordinary PERT; this schedule would be a rough approximation to the expected shedule, biased to underestimate it - because of the PERT-bias phenomenon [4] - while the simulation results would give an unbiased expected schedule.)

In delay-PERT the fixed activities also have start delays. GITPASE has, for scheduling convenience, fixed-finish activities, but for delay-PERT purposes these are considered to be fixed-start activities (the user is considered to have derived the intended schedule such that if an activity had its intended duration it would finish at its fixed-finish time).

4.3. Delay Classes

Each potential delay has an expected value $\tilde{x} > 0$.

All potential delays having $\tilde{x} = 0$ (no delay) belong to the no-delay class. Each potential delay having $\tilde{x} > 0$ belongs to one delay class. Each delay class i has an autocorrelation smoothing constant α_i. Two distinct classes i and j can have the same value $\alpha_i = \alpha_j$, but all delay classes having $\alpha_i = 0$ are collapsed into the single independent-delay class having no autocorrelation.

4.4. PERT Plan Data

With delay-PERT there are two major classes of variables added to Plan-specific data: PERT input variables and PERT output variables.

PERT input variables consist of expected-delay parameters and simulation-control parameters, both of which are user-provided and user-modified either by batch data entry or interactively on the delay-PERT screen.

The expected-delay parameters are of two types, start-delay parameters and duration delay parameters.

Upon delay-PERT screen entry, the Plan is loaded, and each activity or milestone with a fixed finish time has its data changed (or implicitly treated) to have the equivalent fixed start time. Similarly, each activity that has no predecessors has its data changed (or implicitly treated) to have its start time fixed as the network start time. A milestone determined in a different network and lacking a fixed time is also treated as fixed, although it may represent ignorance rather than intent.

There is a single start-delay parameter for each fixed-start activity or milestone (including those that are represented in the Plan as fixed finish) that can constitute a direct delay (DDAs are excluded; all milestones are included). The data structure must allow for this single start-delay parameter to be expanded to three to five parameters per fixed activity in a future implementation of standard PERT simulation. The start-delay parameter is a real number that represents the mean delay, with the delay considered to be a negative-exponential random variable (with standard PERT, the 3-point PERT-beta estimates and perhaps the resulting mean and variance will be expressed here). Delay-PERT simulation will use this mean delay for the initial estimate d_i in each simulation experiment (d_i may change during the experiment if autocorrelation is in effect).

There is a single duration-delay parameter for each activity, whether or not its duration or start time is fixed, excluding milestones and DDAs. The parameter will be expanded to three or five per activity in a future implementation of standard PERT simulation. The parameter is the initial d_i for the duration delay in each simulation experiment for the activity.

Simulation control parameters include, separately for each run, its run name and run text block, its simulation start period, the random number seed, the number of experiments, the combined-experiment parameter, the delay-class membership lists, and the autocorrelation smoothing constants for each delay class. By default, all simulation control parameters except run name, run text block and random number seed are assumed equal to those of the previus run unless the user changes them.

The random number seed is an integer in the range 999 to 9999 that affects the starting point in the simulation run's random generator. Its default value is itself random. For successive experiments in the same run the next number in the generator is used. If the seed is set to a given integer before N experiments are performed, and if the seed is again set to the same integer, then the first N experiments' of the next run will use the same random number.

The run name, run text block, number of experiments and combined experiment parameter are set by the user in response to prompts given when the RUN command is executed. From 1 to 1000 experiments can be in a single run. If the combined-experiment parameter is YES, and experimental conditions remain constant (all delay parameters, delay-class memership lists and status-curtain parameters unchanged since the previous run), the simulation results will be combined with those of the previus run. Up to two delays - a start delay and a duration delay - can be associated with a given activity, and the activity can have each of its delays belong to a delay class. An activity's start delay can be declared as correlated with other delays, and its duration delay can be declared as correlated with different delays. For each class i there is an autocorrelation smoothing constant α_i.

If there is a status curtain resulting from the reporting of firm status data, then events up through the curtain period represent history and cannot be simulated. Upon delay-PERT screen entry, the data will be changed (or implicitly treated) so that the period after the curtain is the simulation start period. Thus, for an activity that had been automatically rescheduled to start at this period because it was not reported as having been started, the Plan shows it starting as early as possible; presumably this was the intent of the user (who otherwise could have rescheduled the Plan before entering the delay-PERT screen, whose entry presumes that the Plan represents the intended schedule).

After schedule preparation upon screen entry, every non-DDA activity will have a potential duration delay, some activities could have a potential start delay, and some milestones could have a potential milestone delay.

A class of delays is a set of delays having correlation among their delay parameters. We exclude the no-delay class ($\tilde{x} = 0$) from consideration. Positive delays ($\tilde{x} > 0$) each belong to some class i, and each class has an autocorrelation smoothing constant $\alpha_i > 0$. All classes having $\alpha_i = 0$ are merged into the independent class. Other classes, having $\alpha_i > 0$, can remain separate even if two classes i and j have equal constants $\alpha_i = \alpha_j$.

For specification convergence, one of the classes, called the great class, initially includes all positive delays and initially is the independent class. The great class can have any α_i, however, and is defined for the purpose of reducing user effort in specifying class memberships when one class is large.

Interactive specification of delay-class data begins with the user's giving the expected delay parameter $\tilde{x}$ for every potential delay. First the system prompts for a single initial $\tilde{x}$ (by default, $\tilde{x}$ = 0) that is initially the parameter for all delays. Then there is a protocol by which the user can change $\tilde{x}$ for each delay. At the end of this, all $\tilde{x} = 0$ delays are collected into the no-delay class and removed from display and further consideration; the positive delays from the initial population of the great class.

Next the system prompts for a single α (by default, $\alpha = \varepsilon$) for the great class.

Then there is a protocol by which the user can form specific delay classes. Classes are identified by the numbers 0,1,2,..., with 0 representing the independent class ($\alpha_0 = 0$), and with an underscsore marking the identification number of the great class. Two displays are maintained: the delay display and the class display.

The delay display is a Gantt chart without resource information, having class identifiers written in timescaled places that indicate the approximate value of the delay parameters. For example, in Figure 3 we see that the start delay for Activity 1 and the duration delays for activities 2 and 3 all are independent delays (0), that the independent class is the great class (underscore 0), that the duration delay for Activity 1 belongs to class 32 and has a rather large expected value, that the duration delay for Activity 2 is small and independent, and that Activity 3 either has no start delay or has a very large independent start delay with no duration delay (color and sort order will distinguish which).

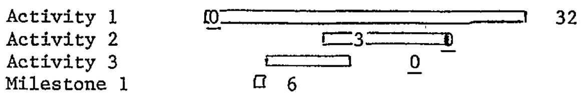

Figure 3: Delay Display

Activities in the delay display appear in a special sorted order: activities having both direct start and duration delays, activities having duration delays but no direct start delays, milestones having direct delays, milestones having no direct delays, other non-DDA activities, and DDA's.

When a delay is activated (by user touch), its $\tilde{x}$ is displayed and can be changed.

The class display lists the class identifier, all the direct activity start delays, duration delays, and milestone delays currently in the class, and α_i, which can be changed.

To collect and rearrange positive direct delays into classes, the user employs two-touch sequences on the delay display. The first delay whose identifier is touched joins or leaves the class of the second one touched, and the great class is implicit. The touch sequences for Create, Join, Leave, Leave & Create, and Leave & Join actions are exemplified in Figure 4.

It would not be necessarily inappropriate for a class to have a mixed membership of start delays, duration delays and milestone delays. If some processes are represented explicitly by an activity while similar processes are represented via a fix or a milestone, such a mixture might be appropriate.

4.5. Schedule Preparation

Upon entry of the delay-PERT screen, the loaded Plan is processed to accept PERT data. This preparation processing includes:

- checking time feasibility of the Plan
- replacing fixed finish times by fixed start times
- replacing curtain-crossing activities by their continuations
- establishing fixed start times for activities having no non-historical predecessor point.

A Plan must be time-feasible before delay-PERT simulation, because it is assumed to represent an intended schedule. It is felt that a time-infeasible schedule may not represent the user's intent, so the user is asked to make the schedule feasible before simulating. (Since the user is not required to save the feasible version, the infeasible Plan may still be the one stored on the disk.)

Fixed finish times are replaced by fixed start times for non-DDA activities. The new fixed start time is the fixed finish time, minus the intended duration, plus one period. For example, if the user intended an activity to have a duration of 4 and to finish in period 10, that is considered to be equivalent to the intent to have a duration of 4 and to start in period 7. The duration and the start time are each delayable.

When a non-DDA activity crosses the curtain, the portion to the right of the curtain is a partial activity whose start is not delayable, but whose remaining duration is delayable. It is simulated as such.

When a non-DDA activity starts in the next period after the curtain, this start time may not reflect the user's intent, because a firm status report may have automatically moved it there if the Plan had previously called for it to start earlier but it was not reported to have started. However, no check will be made of this. Any non-DDA activity that is scheduled to start in the next period after the curtain is considered to be delayable, that is, to have a fixed start for simulation purposes.

Offset precedences could cause an activity to be scheduled to start earlier than its predecessor starts (because a proportion of it does not depend on the predecessor). If the resulting period is the next period after the curtain, the activity has a delayable start.

Initial memberships: $\boxed{1\quad 2\quad 3}$ $\boxed{4\quad 5}$ 6 7 8

Class 1 Class 2 Great class

Action	Touch Sequence	Result from initial memberships
Create	6-7	$\boxed{1\quad 2\quad 3}$ $\boxed{4\quad 5}$ $\boxed{6\quad 7}$ 8
Join	6-5 or 6-4	$\boxed{1\quad 2\quad 3}$ $\boxed{4\quad 5\quad 6}$ 7 8
Leave*	4-4 or 4-5	$\boxed{1\quad 2\quad 3}$ 4 5 6 7 8
Leave & Create	5-6	$\boxed{1\quad 2\quad 3}$ 4 $\boxed{5\quad 6}$ 7 8
Leave & Join*	4-3	$\boxed{1\quad 2\quad 3\quad 4}$ 5 6 7 8
Leave	3-3 or 3-1 or 3-2	$\boxed{1\quad 2}$ 3 $\boxed{4\quad 5}$ 6 7 8
Leave & Create	1-6	$\boxed{2\quad 3}$ $\boxed{4\quad 5}$ $\boxed{1\quad 6}$ 7 8
Leave & Join	1-4	$\boxed{2\quad 3}$ $\boxed{1\quad 4\quad 5}$ 6 7 8

*Singleton class joins great class

Figure 4: Interactive Class Membership Specification

In the future, when GITPASE allows positive offset precedences where the sucessor cannot start as early as the next period after its predecessor finishes, it will be start-delayable if its precedence point is after the curtain. For example, an activity could be scheduled to start three periods after the curtain because of a precedence that was completed before the curtain. Thus, it is necessary to have a consistent method of identifying as start-delayable every start time that is not a result of precedences or quasi-precedences having the precedence point after the curtain.

5. SIMULATION RUNS IN DELAY-PERT

With an intended schedule that has been processed for delay-PERT simulation, a fixed start time exists for each delayable start time.

Let us review the definition in GITPASE of a fixed start or fixed finish time for an activity. In GITPASE a Plan (schedule) is defined by the interaction of network data with Plan-specific data. The Plan-specific data include fixed durations, fixed starts and finishes, and serialization quasi-precedences. Whenever schedule computations are performed, a precedence/quasi-precedence table is computed and then forward-pass and backward-pass schedule computations are done using the combined precedences, with the fixed start times overriding early-start times. The durations used are the scheduled durations, which are the nominal durations overridden by fixed durations.

A fixed duration is simply one where the user (by direct scheduling operations or by accepting a heuristic schedule) has set the duration to some value. The effect of a fixed duration, besides setting a scheduled duration that can be other than the nominal, is to prevent the heuristic from considering alternative durations. The effect of a fixed start time is not only to prevent the heuristic from considering alternative start times, but also to set the start time at a fixed value regardless of early start time - that is, regardless of precedences and serializations.

This definition of start fixing is an artifact of the GITPASE focus on resources. The GITPASE-typical use of a start fix is to achieve resource feasibility (by forcing the activity to span a time interval when resources are available). In such a case the fixed start might represent a process subject to delay-PERT, or not: yes if the fix purpose was to await an expected increase in resource availability where the time of the increase is delayable, no if the fix purpose was to put the activity at an arbitrary time span.

The fix definition for GITPASE can be characterized as a "strong" fix definition in contrast to a "weak" one that would simply specify an early-start candidate time. (A weak fix definition, which would give a not-earlier-than point, is not used in GITPASE.) When a start fix puts an activity later than its early start, positive slack occurs; when a start fix puts an activity earlier than its early start, negative slack (infeasibility) occurs, and there is a conflict between precedences and fixing - a conflict that leaves the scheduler's intentions unclear and whose avoidance is a reason for doing delay-PERT simulation only on time-feasible Plans.

whereupon the duration delay is of topological and chronological order zero.

Recall that each delay belongs to a unique class i.

In arbitrary sequence, a delay having topological /chronological order 0/0 is selected. Its value is generated as follows:

1. Generate $D = -\tilde{x} \ln R$, where $\tilde{x}$ is the current expected value of this delay and $\ln R$ is the natural logarithm of the next uniform (0,1) variate in the computer's random-number generator.

2. Compute $S = \text{int}(D+0.5)$, the integer closest to D. S is the value of this delay in this experiment.

Then the estimates for all delays in the class are updated as follows:

3. Exponentially smooth this expected delay using the smoothing constant α for its class:

$$\tilde{x}_{new} \leftarrow \alpha D + (1-\alpha)\tilde{x}_{old}$$

4. Compute $r = \tilde{x}_{new}/\tilde{x}_{old}$, the ratio of the new expected delay to the old.

5. For each delay in the class (excluding this one if $\tilde{x}_{new}$ already replaced $\tilde{x}_{old}$ in Step 3), multiply each delay by r:

$$\tilde{x}_{new} \leftarrow r\ \tilde{x}_{old}$$

If the class has $\alpha = 0$, Step 3 will leave $\tilde{x}$ unchanged for this delay, and Steps 4 and 5 will leave $\tilde{x}$ unchanged for other delays in the same class. Optionally, Steps 3, 4 and 5 may be suppressed for class i if $\alpha_i = 0$.

A delay having $\tilde{x} = 0$ will have $D = 0$. Optionally, Steps 1 through 5 can be omitted. Steps 4 and 5 should be null when a delay is in the independent class.

When the user touches the RUN menu block, the simulation run begins with the first experiment. Each experiment represents one simulated implementation of the network of activities, with delays, under the assumption that no replanning would be done during implementation.

5.1. Structure of an Experiment

Each delay-PERT experiment in a run consists of three phases: random generation, scheduling, and result capture.

Random generation of delays proceeds in topological/chronological order. Every start delay is of topological order zero. Every start delay that occurs at the earliest time - initially the next period after the curtain, or the netework desired start time if there is no curtain - is of chronological order zero. Each duration delay for an activity that has no start delay (e.g. for a partial activity starting at the curtain) is of topological order zero; each duration is of topological order greater than zero until that start delay is generated, As an example of generation and updating of delays, let us return to the example given the Statistical Scaling subsection of the Positive Correlations Among Delays section: Items 16 and 17 are in the same class, the autocorrelation smoothing constant for the class is 0.1, and the current expected delays for items 16 and 17 are 2.000 weeks and 3.000 weeks, respectively. We generate a delay for item 16, and update the expected delays for items 16 and 17:

Step 1. A (0,1) variate is generated, and its value turns out to be 0.33287. For item 16, D = -2.000 ℓn 0.33287 = 2.2000.

Step 2. The delay to be used in the simulation for item 16 is S = 2, which is the integer nearest to 2.200.

Step 3. The new expected delay for item 16 is 2.0200.

Step 4. r = 2.0200/2.0000 = 1.0100.

Step 5. The new expected delay for item 17 is (1.0100)x(3.0000) = 3.0300.

Now if the delay just generated was a start delay for an activity, the duration delay for the same activity becomes of topological order zero. If the delay just generated was a milestone delay or a duration delay, a succeeding delay may now become of topological order zero. A free start time (one that is not fixed but depend on precedences or quasi-precedences) becomes of topological order zero when the random genreation phase is complete for all of its predecessors.

All zero delays and secondary delays go through a null version of Steps 1 through 5. Thus every delay - every start time and every druation in the entire network - is in due course marked as having completed the random-generation phase of the experiment. The final steps are:

6. Update the topological-sort list. If this delay is an activity start, a duration delay may become of topological order zero. If it is a duration delay, a free-start (secondary) delay may become of order zero.

7. Select a next delay to generate.

The second phase in an experiment is the scheduling phase. We add the start delay to each duration. Then we call the GITPASE scheduling routine and pass to it the delayed fixed starts and delayed durations. The result is an experimental schedule.

The final phase in an experiment is the result capture phase. There are two possible approaches to this: one can write the data for the experiment to core or to mass storage, using either much memory or much time (a disk operation for each experiment) but allowing maximal flexibility for run reporting, or one can follow the traditional method of capturing cell populations and updating moments.

6. CONCLUSION

Delay PERT provides a statistical methodology for modeling, scaling and updating correlated delay statistics, and for using correlated delay statistics in PERT simulation experiments in such a way that the need for a simulation "clock" is bypassed by performing random generation in topological order.

REFERENCES

Robert B. Cooper, Introduction to Queueing Theory, New York, NY: The Macmillan Company, 1972.

James Gantt and Donovan Young, "Interactive PERT Simulation Modeling for Resource-Constrained Project Scheduling," Proceedings of the Eighteenth Annual Simulation Symposium, Simulation Week, Tampa, FL, March 1985.

William W. Hines and D. C. Montgomery, Probability and Statistics in Engineering and Management Science, New York, NY: The Ronald Press Company, 1972.

Joseph J. Modes, C. R. Phillips and E. W. Davis, Project Management with CPM, PERT and Precedence Diagramming, Third Edition, New York, NY: Van Nostrand Reinhold Company, 1983.

Howard Raiffa and R. Schlaifer, Applied Statistical Decision Theory, Boston, MA: Harvard University Press, 1961.

Sheldon M. Ross, Stochastic Processes, New York, NY: John Wiley & Sons, 1983.

Myron Tribus, Rational Descriptions, Decisions and Designs, New York, NY: Pergamon Press, 1969.

Donovan Young and Ronald L. Rardin, "GITPASE: An Interactive Planning Aid for Project Scheduling with Time-Resource Tradeoffs," chapter in Computer Augmentation of Human Decision Making, Wayne Zachary and Julie Hopson, editors, Hillsdale, NJ: Laurance E. Earlbaum Press, 1985.

AUTHOR'S BIOGRAPHY

DONOVAN YOUNG, associate professor of Industrial and Systems Engineering at Georgia Tech, pursues teaching and research interests that he first developed as a project engineer in oil refining: operations research, simulation, and decision support systems. His B.S. in Chemical Engineering (1959) and Ph.D. in Mechanical Engineering (1970) are from The University of Texas at Austin. He is a Registered Professional Engineer in Georgia. He has published papers in Management Science, The Engineering Economist, Simulation, IIE Transactions and other professional journals.