PROXEL-BASED SIMULATION OF PROJECT SCHEDULES

Claudia Isensee
Graham Horton
Fakultät für Informatik
Otto-von-Guericke Universität Magdeburg
39016 Magdeburg, Germany

KEYWORDS
Proxels, project scheduling.

ABSTRACT

Project scheduling is an important tool that breaks down a complex process into its individual tasks and assigns durations to these. When the task durations are given as statistical distributions, rather than constants, the total project duration also becomes a random variable. This paper demonstrates the application of the proxel-based simulation method to the simulation of project schedules. Taking a project schedule as an input, the distribution of the total project duration is computed algorithmically, rather than by analytically combining the statistical distributions of the individual tasks. The experiments performed verify the method and point towards future work.

INTRODUCTION

Project Scheduling

Project scheduling systems such as MS-Project are very helpful in project management today. They help to plan, organize and control the use of time and other resources. The project itself is broken down into smaller parts, which can then be treated separately and result in an overall plan.

In order to represent project schedules throughout this paper, a simple schema is introduced here for the sake of brevity:

A, B, C	Tasks of the schedule
A;B	A must be finished, before B can be started
A|B	A and B run in parallel; the combination is only finished when both tasks are done

('|' has a higher priority than ';')

Figure 1 shows an example schedule where the first task *A* is succeeded by tasks *B* and *C*, which run in parallel, and the last task *D* cannot start before tasks *B* and *C* are finished. The representation of the schedule would be *A;B|C;D*. The splitting and synchronization points *1* and *2* in Figure 1 represent *milestones* of the project.

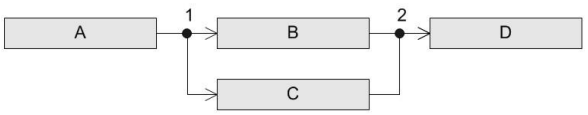

Figure 1: Example Schedule

What are Proxels?

Proxels are a new approach to the simulation of discrete stochastic models. They enumerate all possible states of the system at discrete points in time and compute their specific probabilities. Each proxel ("probability element") stores not only the discrete state of the model, but also the point in simulation time, the amount of time already spent in this state, and the path, that the state has been reached by. The resulting proxel tree structure is a discrete-time Markov chain. The probability starts in the initial system state and is then distributed through the proxel tree. This is done by calculating the probabilities for a state change by the next point in simulation time, deducing from it the probability to stay in that state, and then tracing all of those possible paths recursively. A detailed introduction into proxel-based simulation can be found in (Horton 2002) and (Lazarova-Molnar and Horton 2003).

MOTIVATION

Why Simulate Project Schedules?

Generic project schedule simulation methods used today, such as PERT (Program evaluation and Review Technique) and CPM (Critical Path Method), support the actual scheduling process and resource assignment, and help to identify critical paths. Most of these can only handle fixed task times. Special purpose tools on the other hand, such as the add-on to STROBOSCOPE presented in (Ioannou and Martinez 98), can handle variable task durations, but these are tailored to a specific application field.

Usually the scheduled time for a task is specified as a scalar number that can change as the project goes along, according to the actual progress made. However, at any point in time, the schedule only has one single finishing date. A statistical distribution that remains basically the same throughout the project is a much more expressive statement than a single number that can change while the project progresses.

Theoretically, it is also possible to specify a range or even a statistical distribution for the duration of a task. In practice these numbers mostly result from the experience of the managers and the physical limitations of the project. Therefore it would be useful to incorporate such information into a project schedule. Using this, a distribution for the total project duration could be obtained via simulation.

This paper documents the use of proxel-based simulation to obtain a distribution for the time needed to finish a project. The starting point is an already established schedule with durations for each task represented by statistical distributions. The simulation does not change the scheduling, assign resources or identify the critical path.

SCHEDULE SIMULATION ALGORITHM

This section describes the phases of the project schedule simulation. The first step is to export the relevant schedule information from MS-Project and import it into the simulation program. In a preprocessing step, two data structures are created: a net, representing the schedule, and using this information, the set of all possible states of this net is enumerated and stored in a structure resembling a reachability graph. The actual generation and processing of the proxels is a process which tracks the probability throughout the set of states as time progresses. As a last step, statistics for the final state, which represents the finished project, are computed.

Project Schedule Specification

MS-Project was used as a scheduling tool to specify the project schedules and their parameters. Since task times are usually represented as scalar values, several user-defined fields were customized to represent the distribution of each task's duration, its parameters and the maximum task duration. In addition, a flag was introduced, that distinguishes between atomic (sub-) tasks and summary tasks. Only the atomic tasks, which make up the most detailed schedule, are necessary for the computations. An export schema was devised, that dumps the schedule information into a text file containing one row per task. For each task only the necessary information is put out: *id*, *predecessors* and *successors*, the *distribution* with its *parameters* and the *maximum task duration*.

Task Input & Preprocessing

The text file is taken as an input for the simulator that first builds an internal list of the tasks with their parameters. Each task is connected to its predecessors and successors. If there is no single last project task defined, then a final dummy task with a duration of $DET(0)$ is introduced.

Proxel-based simulation utilizes the method of supplementary variables. Each proxel contains an *age intensity vector* of elapsed enabling times of some timed transactions. The equivalent of an enabling time in a project schedule is the duration that a specific task has been worked on so far. Therefore every task has to be assigned a supplementary variable. This happens according to the following rule:

If two tasks share the same supplementary variable, then one of the tasks must precede the other in the schedule, it must be finished in order for the other task to start.

The heuristic used here does not produce the minimum number of supplementary variables for certain schedules, but the resulting assignment does not increase the state space of the model later on. An example is the above used schedule *A*;*B*|*C*;*D*. In this case, all tasks could share the same supplementary variable, since all milestones are connected linearly, and there are no parallel tasks with independent starting times. The chosen heuristic assigns two separate supplementary variables to *B* and *C*, but in each simulation time step their values are the same. Consequently, the same number of proxels is created, as in the case with only one supplementary variable, and thus no unnecessary overhead is incurred.

A minimal assignment of supplementary variables introduces more complexity into the handling of those during simulation, but does not reduce the actual number of proxels that have to be processed. The chosen tactic is easier to implement, and does not affect the algorithm.

Create Reachability Graph

Starting from the project schedule, all possible states of the schedule are enumerated. This corresponds to the reachability graph of a Petri net. A schedule state is uniquely defined through the tasks that have been finished so far, assuming that there are no unnecessary slack times between tasks. This *donelist* only needs to contain the set of tasks that have been finished, and whose immediate successors are still or not yet in progress. The transitions between the schedule states are associated with the distributions of the task durations. The reachability graph of the schedule *A*;*B*|*C*;*D* is shown in Figure 2. The firing of transition *TA* from *started* to *done(A)* represents the finishing of task *A*.

Figure 2: Reachability Graph

The reachability graph is built starting with the last state of the schedule (the finished project), which is initialised with a *donelist* that only contains the last task of the project schedule, which is known. Starting from there, the reachability graph grows recursively using the following algorithm:

```
start(sched_state s){
  FOR ALL tasks t IN s.done
    s1.done = s.done – t;
    FOR ALL tasks tp IN t.pred
      s1.done += tp;
    END FOR
    IF check(s1) = OK THEN
      add(s1);
      start(s1);
    END IF
  END FOR
}

check(s) – checks if state is possible (no
           precedence restrictions violated)
add(s)   - adds state to reachability graph,
           eliminating doubles
t.pred   - list of immediate task predecessors
s.done   - donelist of schedule state
```

Each task in the *donelist* of a schedule state is replaced by its predecessors, and the resulting state is checked for inconsistencies. These occur, if a task is removed from the list that is a predecessor of another task in the *donelist*. If the state is legal, then it is added to the reachability graph. Schedule states with the same *donelist* are merged by combining their successor lists. Each possible transition from one state to another has to be represented and associated with the corresponding task duration distribution. During this process a state is created with an empty *donelist*. This corresponds to the starting state of the schedule.

Simulation Phase

This phase constitutes the actual processing of the schedule information to determine the distribution of the total project duration. A proxel incorporates the information about the current schedule state, the enabling times of the transitions, the current simulation time and the probability of this combination.

At the beginning of the simulation the proxel of the schedule starting state is assigned a probability of one, and the age intensity vector is initiated with all zeros. As the simulation time progresses in discrete steps Δt, this probability is propagated to the follow-up proxels, according to the *instantaneous rate functions* of the transitions. The *irf* describes the rate of probability flowing from one proxel to the next, for small Δt. The simulation process can be formalized as follows:

```
WHILE dt*step < tmax
  FOR ALL proxels p
    sol[p.state][step] += p.prob
    p1 = probability for leaving p.state
    IF p1 < 1 THEN
      add(p.state,p.prob*(1-p1),p.tau',step+1)
    END IF
    FOR ALL schedule states s in p.succ
      ps = propability of statechange to s
      IF ps > 0 THEN
        add(s, p.prob * ps, p.tau'',step+1)
      END IF

    END FOR
  END FOR
END WHILE

add()    - adds new proxel
sol[][]  - solution array for summing schedule
           state probabilities
p.state  - schedule state
p.prob   - proxel probability
p.tau    - age intensity vector
p.succ   - list of successors
```

First the probability for remaining in the current schedule state is determined, and if it is nonzero, a proxel with the same state and accordingly increased *age intensity variables* is created. Then all schedule states that can be reached from the current one are determined, and if the transition probability is nonzero, the follow-up proxel is created. In each time step this process is repeated for every proxel, until the final simulation time is reached.

The probability for each schedule state per time step is accumulated in a solution array. This array also contains the probabilities for being in the final state of the reachability graph, representing the finishing of the project. In a post processing step, this data is then analysed and some statistical values are computed, such as mean and standard deviation.

EXPERIMENTS AND EXAMPLES

This section presents some experimental results obtained. Comparisons with other project schedule simulation tools have not yet been made, due to time constraints. The section also points out a problem that became apparent through the application of proxel-based simulation to project schedules.

Example

The example schedule $A;B|C;D$ was used with the following distributions of the task durations:

$A \sim U(2,5)$; $B \sim U(5,15)$; $C \sim U(7,12)$; $D \sim U(2,4)$

Figure 3 shows the probabilities of all six schedule states over time, with a step size of 0.5. The flow of probability through the state space can easily be seen. For example, the *done(D)* curve shows, that it is impossible to be done with the project before time 12 and as time progresses further, this becomes more and more likely until about time 24. Afterwards the project is completed with a probability of 1.

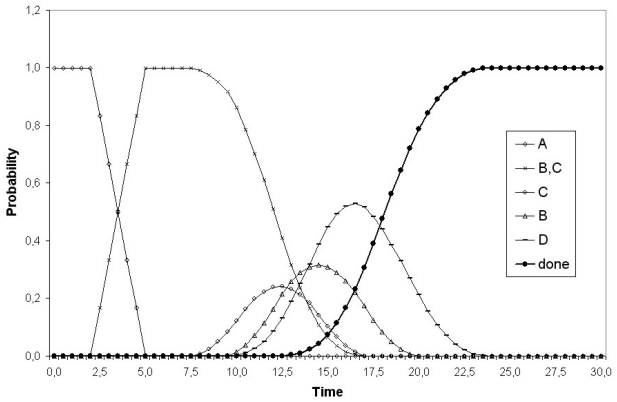

Figure 3: Example Result Graph

Result Variable Development

Previous research suggests that as Δt (simulation step size) decreases, the target variables converge towards their true value, if Δt were zero. The mean value of the total project duration also seems to exhibit this behavior, as Figure 5 and Figure 6 show. Figure 5 depicts the development of the mean of the total project duration for a schedule with the following structure: *A;(B|C;E)|D;F;G|(H|I);J*, where the tasks have various distributions including exponential, uniform and deterministic. The schedule is shown in Figure 4. Figure 6 shows the same development for an example taken from (Ioannou and Martinez 98). The example is a schedule for a road construction and contains 26 tasks. Both graphs suggest a polynomial relationship between Δt and the target variable mean. Further experiments are needed to investigate this property, and possible relationships of other target variables such as the standard deviation.

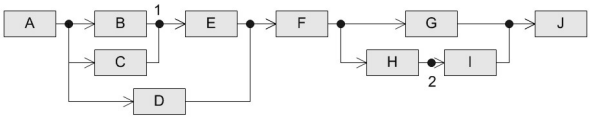

Figure 4: Example Schedule with 10 Tasks

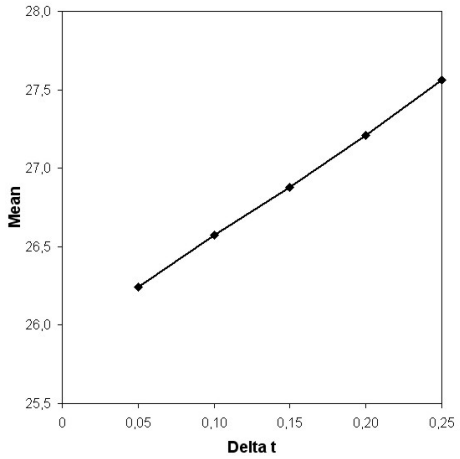

Figure 5: Development of Mean Value with Decreasing Δt

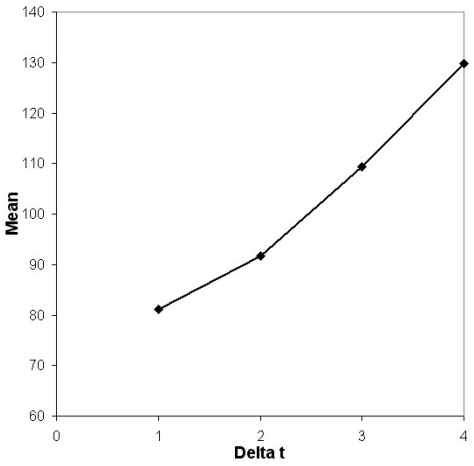

Figure 6: Mean Value Development for Construction Schedule Example with Decreasing Δt

Performance

The performance of the algorithm is directly related to the number of proxels processed during the simulation. An upper bound for the total number of proxels needed is the following:

$$\left(\frac{t\max_{task}}{\Delta t}\right)^{optTau} * numStates * \frac{t\max}{\Delta t}$$

Here $tmax_{task}$ is the longest of the maximum task durations, *tmax* the total simulation time and *numStates* the total number of schedule states in the reachability graph. The variable *optTau* does not represent the actual number of supplementary variables used, but the optimum one. This number is equivalent to the maximum number of independent parallel milestones. The schedule in Figure 4 only has two independent supplementary variables, since both milestones 1 and 2 are independent of the parallel strands, as tasks D or G can still be in progress, when they occur. Therefore the number of total proxels processed grows polynomial with an order of *optTau* as Δt decreases. This can also be seen in Figure 7, which shows the total number of proxels processed for a schedule with *optTau=3*.

The processing time needed grows exponentially with the optimal number of supplementary variables of a schedule. Exponential growth of computation time is a major disadvantage of an algorithm, since an increase in accuracy has to be paid for with an unreasonable amount of computation time. Nevertheless this method is not directly exponential in the size of the schedule, but just in the degree of parallelism within the schedule. Nevertheless, other methods have to be found to improve the quality of the solution, such as extrapolation.

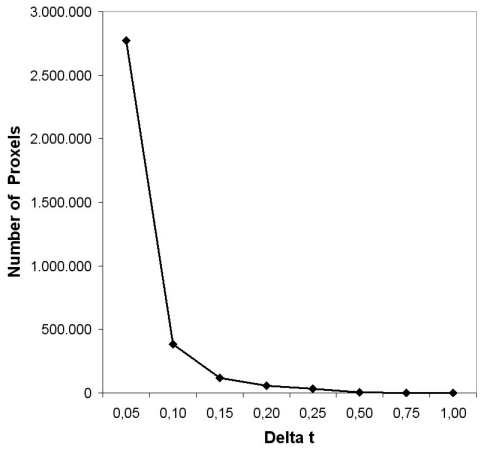

Figure 7: Increase in Number of Proxels with Decreasing Δt

Comparison with Discrete-Event Simulation

In comparison to the proxel method, discrete-event simulation is a random algorithm, which samples the probability distributions and requires replications to produce statistically significant results. This can be a disadvantage when the specific choice of distributions leads to a large variance in the output variable and thus requires a very large number of replications to be computed accurately. The proxel method is insensitive to this effect.

The processing time of a discrete-event simulation algorithm is not only dependent on the size of the input schedule, but also linearly related to the number of replications done, as can be seen in Figure 8. To produce result distributions, similar in accuracy to the proxel-based method, a large number of replications is needed.

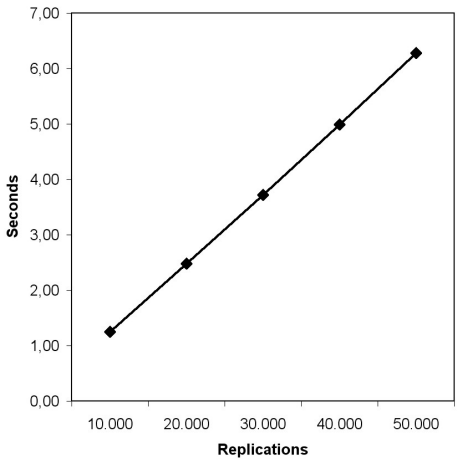

Figure 8: Linear Increase in Computation Time with Increasing Number of Replications

When using a sufficiently small Δt for the proxel method to get good results, the performance of the tested discrete-event simulator is much better, since it does not use a discretization stepsize. However, as already mentioned, results from the proxel method, calculated using large Δt can be extrapolated to having a theoretical Δt of zero, and thereby lead to an accurate solution at much lower cost. When comparing the computation time needed to compute few proxel results using large Δt and with the time needed to let the discrete-event simulator perform a sufficient number of replications, the proxel method is faster, although the exponential relationship between processing time and input size suggests otherwise.

Problems

One problem arising from the proxel simulation of project schedules is an initial bias in the solution value, which is caused by the minimal number of steps needed for probability to reach the final milestone. Since the simulation progresses in discrete steps, there can be a maximum offset of Δt between the actual and the simulation time at which a state change occurs. This offset is accumulated along the paths of the reachability graph, and consequently results in a positive offset or bias in the resulting distribution, and especially the mean and maximum project duration. As expected, this bias decreases as Δt is chosen smaller, but this also greatly increases the processing time needed. The bias in the maximum project duration is easy to predict through the critical path, calculated using the maximum task durations. Nevertheless, it might be possible to also predict and then eliminate the bias from the mean value. This question needs to be investigated further.

Another problem of the proposed algorithm is the undesirable property of an exponential relationship between the optimum number of supplementary variables and the total number of proxels that need to be processed, which is directly related to the total computation time needed. This is due to the fact that the proxel-method enumerates the total state space of the model with all possible combinations of the *age intensity vector* elements. However, the performance is not directly exponential in relation to the size of the schedule, but just in relation to the degree of parallelism of the schedule. This is a major difference to real exponential problems such as enumerating the possible paths of the travelling salesman.

DISCUSSION

The reasons why proxels would be particularly useful to simulate project schedules lie both within the problem and the structure of the proxel-method itself.

Taking the schedule and the distributions of the individual tasks as an input, it is the goal to determine the statistical distribution of the total project time. Theoretically it is be possible to analytically combine the distributions for the tasks making up the schedule.

The distributions of two parallel tasks would have to be multiplied, and the combination of two tasks scheduled in a direct sequence would require the convolution of the individual distributions. Computing a product of density functions is fairly easy, whereas computing the convolution of two density functions has quadratic complexity.

The proxel-based method leads accurate results through extrapolation of few runs with large Δt, while a discrete-event simulator needs a larger number of replications, that is also dependent on the variance of the input distributions. Furthermore, the results of the proxel method can be used to extract more information than just the project end date, such as a probabilistic critical path. Therefore, although the proxel method is exponentially related to an input parameter compared to a linear relationship in discrete-event simulation, other properties of the method tend to compensate that negative effect.

The flow of probability in a project schedule is unidirectional, i.e. it flows from left to right as time progresses. When translating this into a proxel structure, the state space of the model exhibits a specific behaviour. It only grows up to a particular point in time. As the end of the project approaches, more and more tasks are finished, and the number of concurrent proxels decreases. Another advantage of this forward-flowing structure is the predetermined maximum simulation time.

CONCLUSION

This paper documents the proxel-based simulation of project schedules. The goal was to obtain a probability distribution for the total project duration. The algorithm used was described, starting from the export of the schedule information from MS-Project through pre-processing and simulation to the computation of statistics for the obtained distribution. Some experiments were performed to verify the method and measure its performance. The performance of the algorithm was estimated and problems discussed.

In comparison to existing tools the presented method does not modify the project schedule; it only evaluates the schedule and does not give any feedback in terms of better scheduling strategies or resource assignments. Nevertheless, it demonstrates the easy applicability of proxel-based simulation to another application field. The method is specifically tailored to obtain the total project durations' distribution without analytically combining the task distributions.

Future Work

The presented method needs to be investigated further concerning its performance. The results and performance parameters have to be further compared to the ones of other methods or tools.

The bias in the result distribution variable mean, which is a result of the linear schedule structure and the discrete simulation steps, should be eliminated. This can either happen through extrapolation with different Δt or through formalizing the effect of Δt on the mean, and then eliminating it.

Using statistical distributions as task durations, it is usually not possible to determine one specific critical path. Nevertheless, an equivalent could be found that would necessarily contain probabilistic elements. Defining a probabilistic critical path and including its discovery as a post processing step could be an addition to the present algorithm.

REFERENCES

Horton, G. 2002. *A new paradigm for the numerical simulation of stochastic Petri nets with general firing times.* In: Proceedings of the European Simulation Symposium 2002. Dresden: Society for Computer Simulation.

Ioannou, P. G. and Martinez, J. C. 1998. *PROJECT SCHEDULING USING STATE-BASED PROBABILISTIC DECISION NETWORKS.* In: Proceedings of the 1998 Winter Simulation Conference. Eds: D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan.

Lazarova-Molnar, S. and Horton, G. 2003. *Proxel-Based Simulation of Stochastic Petri Nets containing Immediate Transitions.* In: On-Site Proceedings of the Satellite Workshop of ICALP 2003 in Eindhoven, Netherlands. Forschungsbericht Universität Dortmund: SCS Verlag.

AUTHOR BIOGRAPHIES

CLAUDIA ISENSEE studied Computer Science at the Otto-von-Guericke-Universität Magdeburg and spent an exchange year at the University of Wisconsin, Stevens Point, where she graduated in 2002. She was awarded her German University diploma in September 2003. Since October 2003 she is working at the "Lehrstuhl für Simulation" at the Otto-von-Guericke-Universität Magdeburg. Her e-mail address is: `claudia@isg.cs.uni-magdeburg.de`.

GRAHAM HORTON studied Computer Science at the University of Erlangen, obtaining his Masters degree ("Diplom") in 1989. He obtained his PhD in Computer Science in 1991 and his "Habilitation" in 1998 at the same university, in the field of simulation. Since 2002, he is Professor for Simulation and Modelling at the Computer Science department of the University of Magdeburg. His email address is: `graham@isg.cs.uni-magdeburg.de`