Hindawi
Advances in Civil Engineering
Volume 2021, Article ID 1567261, 13 pages
https://doi.org/10.1155/2021/1567261

Research Article

Dynamic Probability Analysis for Construction Schedule Using Subset Simulation

Shen Zhang and Xingyu Wang

Central-South Architectural Design Institute Co., Ltd., Wuhan 430071, China

Correspondence should be addressed to Shen Zhang; csadi_zhangs@zonaland.cn

Received 9 June 2021; Revised 6 August 2021; Accepted 28 August 2021; Published 9 September 2021

Academic Editor: Bang Yeon Lee

Schedule management is an essential part of construction project management. In practical management affairs, many uncertainties may lead to potential project delays and make the schedule risky. To quantify such risk, the Probabilistic Critical Path Method (PCPM) is used to compute the overdue probability. Survey shows it could help project managers understand the schedule better. However, two critical factors limited the application of PCPM: computational efficiency and timeliness. To solve these constraints, we combined subset simulation and statistical learning to build a computationally efficient and dynamic simulation system. Numerical experiment shows that this method can effectively improve the computation efficiency without losing any accuracy and outperforms the other approaches with the same assumptions. Besides, we proposed a machine learning-based way to estimate task duration distributions in PCPM automatically. It collects real-time progress data through user interactions and learns the best PERT-Beta parameters based on these historical data. Our estimator provides our simulation system the ability to handle dynamic assessment without laborious human work. These improvements reduce the limitations of PCPM, making the application of PCPM in practical management affairs possible.

1. Introduction

Schedule management is a vital part of construction project management. Overdue projects will bring potential economic losses to the project. But, on the other hand, groundless period reductions will inevitably bring risks to the project. So, applying scientific management methods is particularly important to evaluate the project schedule and control the risk of delay.

Critical Path Method (CPM) is a widely accepted and well-used tool in planning, controlling, and scheduling construction projects. It can help the project manager to identify the critical path in the schedule and determine the project duration when all tasks' durations are determined [1]. However, in practice, there exists uncertainty and volatility in projects. These will undermine classical deterministic CPM. To deal with this situation, lots of improvements were put forward to extend CPM to a nondeterministic environment. Unlike classical CPM, extended CPM computes the critical path on assumption that the task duration is nondeterministic, and it has to estimate the probability of project delay. This feature provides the project manager a quantitative way to evaluate the risk in project progress and makes extended CPM more appealing. Generally, major existing methodologies used in extended CPM could be classified into three types.

The "most critical path" method was firstly proposed by Soroush in 1994 [2]. This method focuses on the "most critical path" problem and provides a heuristic to identify the near-optimal path to such a problem. Since this method only uses the "most critical path," rather than the entire schedule network, to compute the delay probability, it runs very fast and is more accurate than the classical CPM approach. This method provides a good perspective to understand critical path [3] as well as an extensible solution framework. Some researchers combined it with artificial intelligence to find the optimal critical path [4, 5], and some extended it to fuzzy domain [6].

Unlike classical CPM, Fuzzy CPM (FCPM) represents an activity duration by replacing a real number with a fuzzy number and uses fuzzy operations to calculate the fuzzy critical path and the fuzzy project duration. By combining different kinds of fuzzy numbers and fuzzy operations, researchers put forward many constructive and useful methods. These contributions resolved some critical defects of FCPM, like negative time and invalid fuzzy subtraction, and provide an efficient way to resolve the fuzzy critical path problem for a project network. Nowadays, many researchers are still working on its variants and verifying its effectiveness and practicability in practice [7–10].

Similar to Fuzzy CPM, Probabilistic CPM (PCPM) represents an activity duration by a random variable. This makes the completion time also a random variable. The ultimate goal of PCPM is to find out the probability where the completion time is higher than the given deadline and identify the critical path in a probabilistic context. Researches showed that because of its simple probabilistic interpretation, this methodology could be easily understood by project managers and help them to manage a project better as well as minimizing delay risks [11–15].

It is worth noticing that, though formulated in a simple way, PCPM is in fact hard to solve and is still under study. Four key factors might explain such difficulty. First, PCPM is a high-dimensional integral of the probability density function on the failure domain defined by an implicit limit state function. Second, this limit state function is composed of several mutual-referenced and nested computation blocks, making it behave like a black box system. Third, since there are lots of nonlinear logical operands, like max and min, in nested blocks, the limit state function is also highly nonlinear. This fact makes some popular delegate models, like First-Order Reliability Method and Response Surface, fail to approximate the target function. Fourth, Direct Monte Carlo Simulation (DMCS) is time-consuming to reach an unbiased numerical estimation, which might be seemingly unpromising in practice.

Considering all the difficulties described above, we suggest adopting a well-designed black box system reliability algorithm, which could solve a reliability assessment problem without caring about the details of our target system. Subset simulation is such an algorithm. It is a numerical method widely used in structural engineering [16–18] and soil engineering [19, 20]. Existing research studies showed that subset simulation is computationally efficient in estimating small failure probabilities in high-dimensional black box reliability problems. But this method is rarely used in project management. Given the similarities in implicity and nonlinearity between our system and existing researches, we think subset simulation will also work on the PCPM problem.

Apart from the computation efficiency problem, there is another important issue to address. Most researches mentioned above did not consider the timeliness of assessment. As a project goes on, the task duration estimations in the schedule will change. A feasible solution is to reestimate them manually, but it will bring lots of extra workloads, which could be avoided. To realize an automated real-time reliability assessment, we introduce statistical learning into PCPM. This method can handle the dynamic evaluation in the construction life cycle without much extra work. Based on historical progress data collected from management software, the duration distribution of a task could be estimated automatically using machine learning. This feature would make PCPM more useful in practical management affairs.

In this study, we will provide a new method, combining subset simulation and statistical learning to resolve the project delay problem efficiently and dynamically. The three primary research objectives are described as follows:

(1) Introduce subset simulation-based PCPM and demonstrate its working process as well as implementation details

(2) Illustrate how statistical learning can be plugged into PCPM, propose the cost function of this learning system, and derive the formula of task duration distribution given collected data

(3) Verify the effectiveness of the proposed method and compare it with other methods

The organization of the rest of this paper is summarized as follows. Section 2 provides a short introduction to Critical Path Method (CPM) as well as a well-designed algorithm solving Critical Path Problem and then formulates the Schedule Reliability Problem. This problem will be solved in Section 3, using subset simulation. This method is a computation-efficient algorithm to estimate small failure probability. We will demonstrate how to use this method to solve the Schedule Reliability Problem. After that, we will explain how we can apply statistical learning in our PCPM system in Section 4. By combining these parts, Section 5 will demonstrate our method as a whole and display its design and implementation. Then, we used a hypothetical test case to compare our proposed method with other models in Section 6. Finally, Section 7 gives our concluding remarks.

2. Preliminaries

In this section, we will first formulate the Schedule Reliability Problem we want to resolve. Then, we will introduce a well-designed algorithm, solving the critical path problem. This algorithm will work as an implicit subexpression of the Schedule Reliability Problem. Basics of Critical Path Method and Schedule Network are omitted for conciseness.

2.1. Schedule Reliability Problem. A schedule does not always work as project managers expect. Sometimes it might fail. In other words, the real completion time of a project would exceed the time limit given by the contract. This is a disastrous thing we need to avoid. To quantify delay risk, we assume that the completion time of a project is a random variable. Given a deadline, the delay probability could be formulated as follows:

$$p_f = \int I\left(T_c\left(\mathbf{D}\right) > T_d\right)\mathrm{d}\mathbf{D}, \tag{1}$$

where $\mathbf{D} = (D_1, D_2, \ldots, D_n)^T$ is a vector collecting the duration variables D_k of every task; $T_c(\mathbf{D})$ is an implicit function transforming durations of tasks to the completion time; $I(x)$ is an indicator function which takes 1 if x is positive; otherwise, it takes 0; $p(\mathbf{D})$ is the joint probability density function of the duration variables. Thus, our goal is to compute the delay probability and ensure it is less than a given threshold, for example, 5%.

2.2. Modified Dijkstra Algorithm. To solve equation (1), we need to solve the Critical Path Problem first. Since the critical path is the longest in a schedule network, we could compute the completion time T_c and get the Critical Path using the Modified Dijkstra Algorithm (MDA). This algorithm is proposed by Ravi Shankar and Sireesha to find the Critical Path in Schedule Network [21].

MDA uses Activity-On-Node (AoN) representation and represents a schedule by a graph G, denoted as a tuple $G = (V, E)$. V are the vertices in the graph, representing the tasks in the schedule. For each task v_k, a vertex attribute d_k is assigned, defining task's duration. E are the directed edges in the graph, representing the work dependencies.

MDA improves the classic Dijkstra Algorithm by adding a topological sorting [22] before computing the critical path. This improvement could improve efficiency significantly, especially when repeat sampling is needed. After topo-sorting, the forward pass is applied to get the earliest start (ES) and earliest finish (EF) of each task. The completion time is the latest EF among all the tasks. Once completion time is computed, the backward pass is applied to get the latest start (LS) and latest finish (LF) of each task. Total Float (TF) is the difference between the Earliest Start (ES) and the Latest Start (LS) of an activity. A task is a critical task when its total float is zero.

The pseudocode of MDA is given as follows, and note that the nodes in MDA are presorted in topological order, which satisfies two indexing constraints $i < j \forall i \in \text{Node}[j] \cdot \text{predecessors}$ and $i < j \forall j \in \text{Node}[i] \cdot \text{successors}$ (Algorithms 1 and 2).

3. Subset Simulation

3.1. Basic Idea. Since $T_c(D)$ is a nested and implicit function, it is hard to solve equation (1) analytically for most schedules. Instead, Monte Carlo Simulation is used to get a numerical solution, formulated as follows:

$$\widehat{p}_f = \frac{1}{N} \sum_{k=1}^{N} I\left(T_c\left(\mathbf{D}^{(k)}\right) > T_d\right). \tag{2}$$

However, Direct Monte Carlo Simulation (DMCS) is known to be computationally inefficient. To get a stable estimation, lots of samples are required, making the computation process time-consuming. To solve equation (2) efficiently, subset simulation is applied in our study.

Subset simulation was first proposed by S. K. Au to estimate the small failure probability of a structural system in high dimensions. Because of its efficiency, this method was widely used in structural engineering and soil engineering.

The basic idea of subset simulation is to express a small probability as a product of a series of larger conditional probabilities by introducing intermediate failure events. Since intermediate failure events are much easier to solve, this decomposition could reduce unnecessary sampling and accelerate the computation. This process could be conceptually expressed as follows:

$$\widehat{p}_f = p\left(T_c > T_d^{(0)}\right) \prod_{k=2}^{n} p\left(T_c > T_d^{(k)} | T_c > T_d^{(k-1)}\right), \tag{3}$$

where $T_d^{(i)}$ is the generated time limit in ith iteration and there is $T_d^{(n)} = T_d$ in final iteration. According to the definition above, two critical questions should be answered:

(1) How to choose the intermediate failure event $T_c > T_d^{(k)}$, that is, the value of $T_d^{(k)}$?

(2) How to generate samples $D^{(k)}$ from conditional event $T_c > T_d^{(k)} | T_c > T_d^{(k-1)}$?

In the following parts, these questions will be answered by Au's standard subset simulation algorithm. And we will illustrate how it could solve Schedule Reliability Problem.

3.2. Standard Subset Simulation Algorithm. For simplicity, our study used S. K. Au's standard algorithm. In this algorithm, samples in each iteration are generated by Markov Chain Monte Carlo (MCMC) using seeds, and intermediate time limits and seed samples in iteration are chosen by a fixed threshold percentile [16].

There are only two parameters used in standard algorithm: batch size N_b which is the size of samples in each iteration and seed ratio p_s which is the ratio between seeds and samples in each iteration.

3.2.1. Intermediate Event Selection. Suppose a batch of samples $\mathbf{D}^{(k)}$ is known; we could compute their corresponding completion time $T_c^{(k)} = \left\{T_{ci}^{(k)}\right\}$ using Modified Dijkstra Algorithm. Then, we sort these times in a decreasing order. By selecting the samples with the top $[p_s N_b]$ longest completion time, we can ensure that the intermediate events $T_c > T_d^{(k)}$ satisfy $p(T_c > T_d^{(k)} | T_c > T_d^{(k-1)}) = p_s$.

$$T_d^{(k)} = \frac{T_{c(r)}^{(k)} + T_{c(r+1)}^{(k)}}{2}, \tag{4}$$

where $r = [p_s N_b]$ and $T_{\text{cr}}^{(k)}$ is the rth largest completion time among samples.

3.2.2. Conditional Batch Sampling. The simplest sampling policy is the Metropolis–Hastings sampling, which is based on a symmetric proposal with $q(\mathbf{D}_0)\pi(\mathbf{D}_0, \mathbf{D}_1) = q(\mathbf{D}_1)\pi(\mathbf{D}_1, \mathbf{D}_0)$. In our study, transition $\pi(\mathbf{D}_0, \mathbf{D}_1) \sim \mathcal{N}(0, \sigma)$ is used. We accept the move on Markov chain with an acceptance probability of $\alpha = \min\{1, q(\mathbf{D}_1)/q(\mathbf{D}_1)\}$.

$q(\mathbf{D})$ is the unnormalized conditional distribution.

(1) $T_e = 0$
(2) **for each** $i \in$ Nodes
(3) $E_S^{(i)} = 0$
(4) **for** $i = 1$ **to** Nodes · length
(5) $E_F^{(i)} = E_S^{(i)} + D^{(i)}$
(6) **if** Node$[i]$ · has Successor
(7) **foreach** $j \in$ Node$[i]$ · successor
(8) **if** $E_F^{(i)} > E_S^{(j)}$ **then** $E_S^{(j)} = E_F^{(i)}$
(9) **else**
(10) **if** $E_F^{(i)} > T_e$ **then** $T_e = E_F^{(i)}$
(11) **return** T_e

Algorithm 1: MDA forward algorithm.

(1) **for each** $i \in$ Nodes
(2) $L_F^{(i)} = T_e$
(3) **for** $j =$ Nodes · length **to** 1
(4) $L_S^{(i)} = L_F^{(i)} - D^{(i)}$
(5) $\mathrm{F}^{(i)} = L_F^{(i)} - E_F^{(i)}$
(6) **if** Node[j] · has Predecessor
(7) **for each** $i \in$ Node$[i]$ · predecessor
(8) **if** $L_S^{(j)} < L_F^{(i)}$ **then** $L_F^{(i)} = L_S^{(j)}$

Algorithm 2: MDA backward algorithm.

$$q(\mathbf{D}) = \begin{cases} \prod_i p(d_i), & \mathbf{D} \in F, \\ 0, & \mathbf{D} \notin F, \end{cases} \tag{5}$$

where $p(d)$ is PERT-Beta Distribution described in Section 4.1 and F is the intermediate failure event $F = \{\mathbf{D} | T_c(D) > T_d\}$.

3.2.3. Procedures. The pseudocode of standard algorithm is shown below (Algorithm 3).

This process is illustrated in Figure 1.

4. Dynamic Distribution Estimation

In this section, a historic data-based duration distribution estimation method is proposed. This method uses the PERT-Beta distribution as the probabilistic model. And it takes three assumptions to make the problem easier to solve. Finally, it uses gradient descending, a widely used optimization tool in machine learning, to estimate observational duration distribution.

4.1. PERT-Beta Distribution. To make PCPM work, the duration of each task should be modelled properly. An empirical distribution is usually used when there is a lack of observational data. Some survey shows beta distribution is a suitable one [23–25]. This distribution requires three empirical parameters: most likely duration b, optimistic duration a, and pessimistic duration c of the activities. The process of defining these subjective values is called the PERT three-point estimation method.

The density function of PERT-Beta Distribution is given by equation (6) and illustrated in Figure 2.

$$p(t) = \frac{1}{B(\alpha, \beta)} \frac{(t-a)^{\alpha-1}(c-t)^{\beta-1}}{(c-a)^{\alpha+\beta-1}}, \tag{6}$$

where parameters in beta function are determined by

$$\alpha = \frac{4b + c - 5a}{c - a}, \qquad \beta = \frac{5c - a - 4b}{c - a}. \tag{7}$$

However, estimation is not a static thing. With the project going on, time distribution should be corrected and reestimated according to the status quo. The project manager could do these works but will spare lots of effort. It would be time-saving if we could estimate the distribution automatically.

4.2. Assumptions. Statistical learning could be applied in our work to estimate distribution automatically. However, to build such a learning system, we need some basic assumptions to construct the cost function of this system. In our study, we make three assumptions about task progress and time distribution.

(1) *Constant Working Rate*. In project management, the relationship between time and remaining works,

(1) generate initial samples: $\mathbf{D}^{(0)} = [\mathbf{D}_1^{(0)}, \mathbf{D}_2^{(0)}, \ldots, \mathbf{D}_n^{(0)}]$
(2) **repeat**
(3) compute $T_{ci}^{(k)} = \mathrm{MDA}(\mathbf{D}_i^{(k)})$ in k -th batch samples
(4) compute $p_f^{(k)} = 1/N_b \sum_{i=1}^{N_b} \mathrm{I}(T_{ci}^{(k)} > T_{\mathrm{d}})$
(5) **if** $p_f^{(k)} < p_s$
(6) select seeds and intermediate failure event $T_d^{(k)}$ using method in Section 3.2.1
(7) generate next batch of samples $D^{(k+1)}$ using method in Section 3.2.2
(8) **else**
(9) **break**
(10) **return** $p_f = p_s^k p_f^{(k)}$

Algorithm 3: Subset simulation-based PCPM.

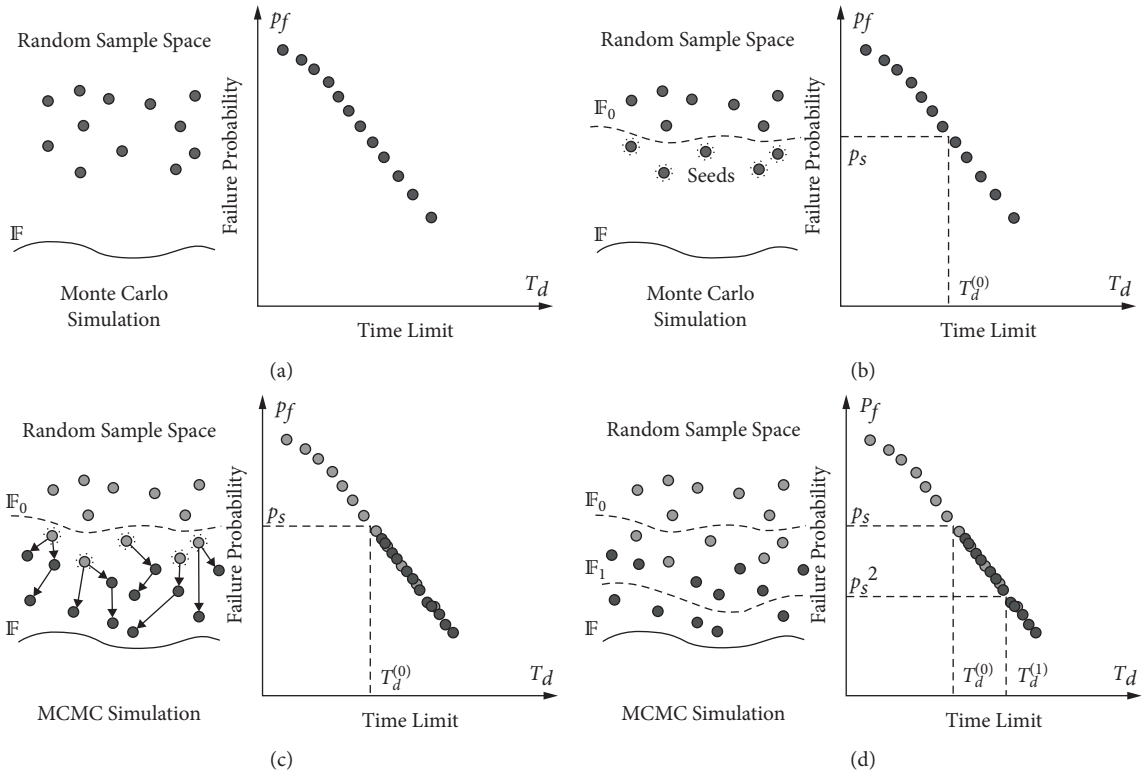


Figure 1: Illustration of subset simulation procedure. (a) Iteration 1: Direct Monte Carlo Sampling. (b) Iteration 1: Seed Selection. (c) Iteration 2: MCMC Sampling using seeds. (d) Iteration 2: Seed Selection.

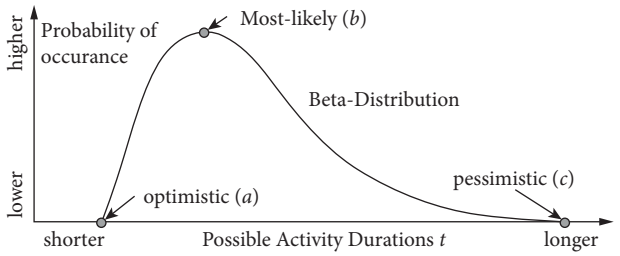


Figure 2: Typical density function of the PERT-beta distribution.

burndown chart, a curve depicting is usually an S shape curve. But to simplify the computation, we assume that this curve is a straight line. This assumption works on optimistic, average, and pessimistic working rates r_a, r_b, and r_c. Given any remaining progress p, the parameters could be computed using linear interpolation as follows:

$$a_p = r_a(1-p),$$
$$b_p = r_b(1-p), \quad (8)$$
$$c_p = r_c(1-p).$$

(2) *Fixed Risk Preference*. Parameters α and β control the shape of Standard Beta distribution, deciding its skewness, or in another perspective, a likelihood of a task finished in a short time. If a task is more likely to be completed in a shorter time, α is greater and β is

smaller. To some degree, they reflect an implicit risk preference of a project manager. So, in our study, once α and β are chosen, they do not change anymore.

(3) *Two-Step Estimation*. There is a growing uncertainty when we want to predict the future, while with progress going on, this uncertainty could be diminished. To depict this fact, we take a two-step method to estimate observational distribution. We first assume all the observations come from the "future" and use them to estimate the working rate at different conditions. Then, these working rates are used to construct the final distribution using the current state.

These assumptions are illustrated in Figure 3.

4.3. Statistical Learning. Based on the assumptions before, we could derive the cost function of our learning system. According to equation (9), the likelihood could be expressed in

$$p(\mathbf{T}) = \frac{1}{B(\alpha,\beta)} \prod_i \frac{\left(t_i - a_p\right)^{\alpha-1}\left(c_p - t_i\right)^{\beta-1}}{\left(c_p - a_p\right)^{\alpha+\beta-1}}. \tag{9}$$

Since α and β are assumed to be fixed and the working rate is constant, by taking the negative log-likelihood of observations, cost function $L(T)$ could be derived as follows:

$$\begin{aligned} L(T) \propto & \sum_i (\alpha+\beta-1)\log\left(r_c - r_a\right) \\ & - (\alpha-1)\log\left(t_i - r_a\left(1-p_i\right)\right) - (\beta-1)\log\left(r_c\left(1-p_i\right) - t_i\right). \end{aligned} \tag{10}$$

Thus, our purpose is to optimize the loss function $\min_{r_a, r_c} L(\mathbf{T})$ and estimate three working rates r_a, r_b, and r_c. Taking the partial derivative of loss function, we will have

$$\begin{aligned} \frac{\partial L(\mathbf{T})}{r_a} &= \sum_i -(\alpha+\beta-1)\frac{1}{r_c - r_a} + (\alpha-1)\frac{1-p_i}{t_i - r_a\left(1-p_i\right)}, \\ \frac{\partial L(\mathbf{T})}{r_c} &= \sum_i (\alpha+\beta-1)\frac{1}{r_c - r_a} - (\beta-1)\frac{1-p_i}{r_c\left(1-p_i\right) - t_i}. \end{aligned} \tag{11}$$

Using gradient descending, we could have numerical optima r_a^*, r_c^* using the update rule as follows:

$$\begin{aligned} r_a &\longleftarrow r_a - \eta\frac{\partial L(\mathbf{T})}{r_a}, \\ r_c &\longleftarrow r_c - \eta\frac{\partial L(\mathbf{T})}{r_c}. \end{aligned} \tag{12}$$

After determining r_a^* and r_c^*, r_b could be computed using the definition of α; see equation (7).

$$r_b^* = r_a^* + \frac{\alpha-1}{4}\left(r_c^* - r_a^*\right). \tag{13}$$

With all three working rates r_a^*, r_b^*, and r_c^* computed, the final estimation could be determined. Suppose the current remaining progress is p_c; then the updated range (a_0, b_0, c_0) could be given as follows:

$$\begin{aligned} a_0 &= r_a^* p_c + r_b^*\left(1-p_c\right), \\ b_0 &= r_b^*, \\ c_0 &= r_c^* p_c + r_b^*\left(1-p_c\right). \end{aligned} \tag{14}$$

5. System Implementation

5.1. Design of Simulation System. A detailed design of our system and the relationship between major classes are described in the class diagram in Figure 4. Some important members and methods of major classes are presented, but some trivial members or methods are omitted for lack of space.

(1) The class App is the entry class that handles all the interactions from users: we could configure the simulation using class Setting. The Setting class defines the time limit t_{d}, batch size N_b, and candidate ratio p_s. And it affects the result and performance of subset simulation. We use class RecordManager to add a progress record to the database and estimate the duration distribution of a task. We use class SubsetSimulation to evaluate our schedule and get some useful indicators.

(2) The class RecordManager is the manager class that takes in the record inputs and stores them. Besides, it also provides the function to estimate the distribution of a task using historical data. This process is described in Section 4. And when a simulation begins, the estimated beta distribution parameters would be provided to samplers to construct a batch of samples.

(3) The class SubsetSimulation is the entry class where our simulation happens. This class computes the failure probability and sets the average total float of tasks respectively. We described this process thoroughly in Section 3. SubsetSimulation will use MDA class to compute the Critical Path and completion time of the schedule. You can find the theory about MDA in preliminaries.

(4) The class Graph is a helper class that represents a schedule network using an adjacency list. It contains the topological structure of the schedule network and could be used to find the successors or predecessors of any given task. The structure is stored in database separately, using class Task and Dependency. These two classes correspond to two tables, respectively, using Object Relation Mapping (ORM).

These classes will generate three kinds of data:

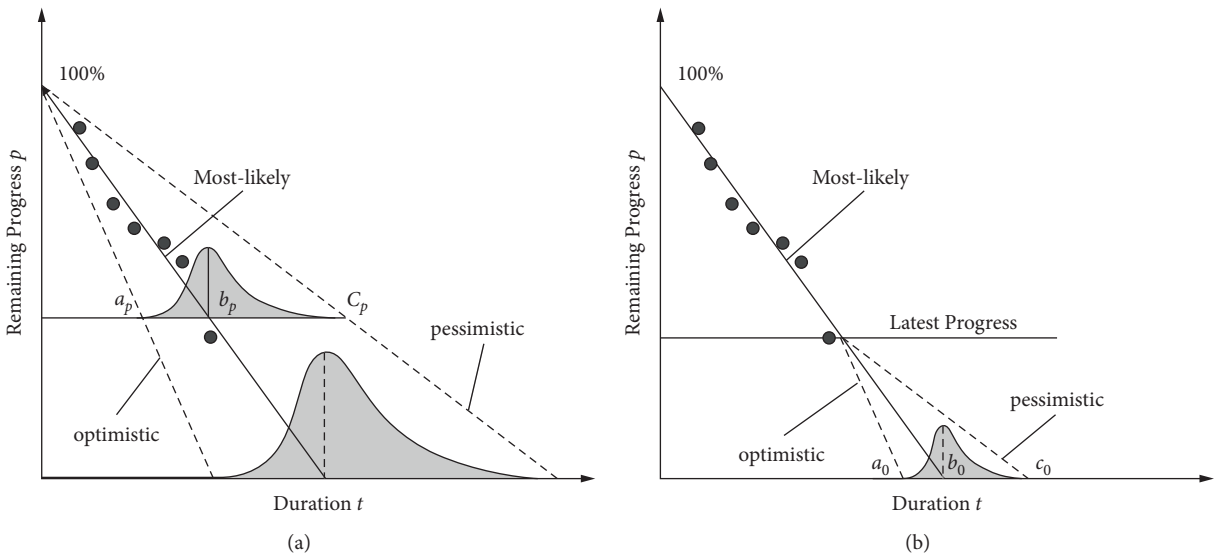


Figure 3: Assumptions in our two-step estimation. (a) Step 1: working rate estimation. (b) Step 2: final distribution estimation.

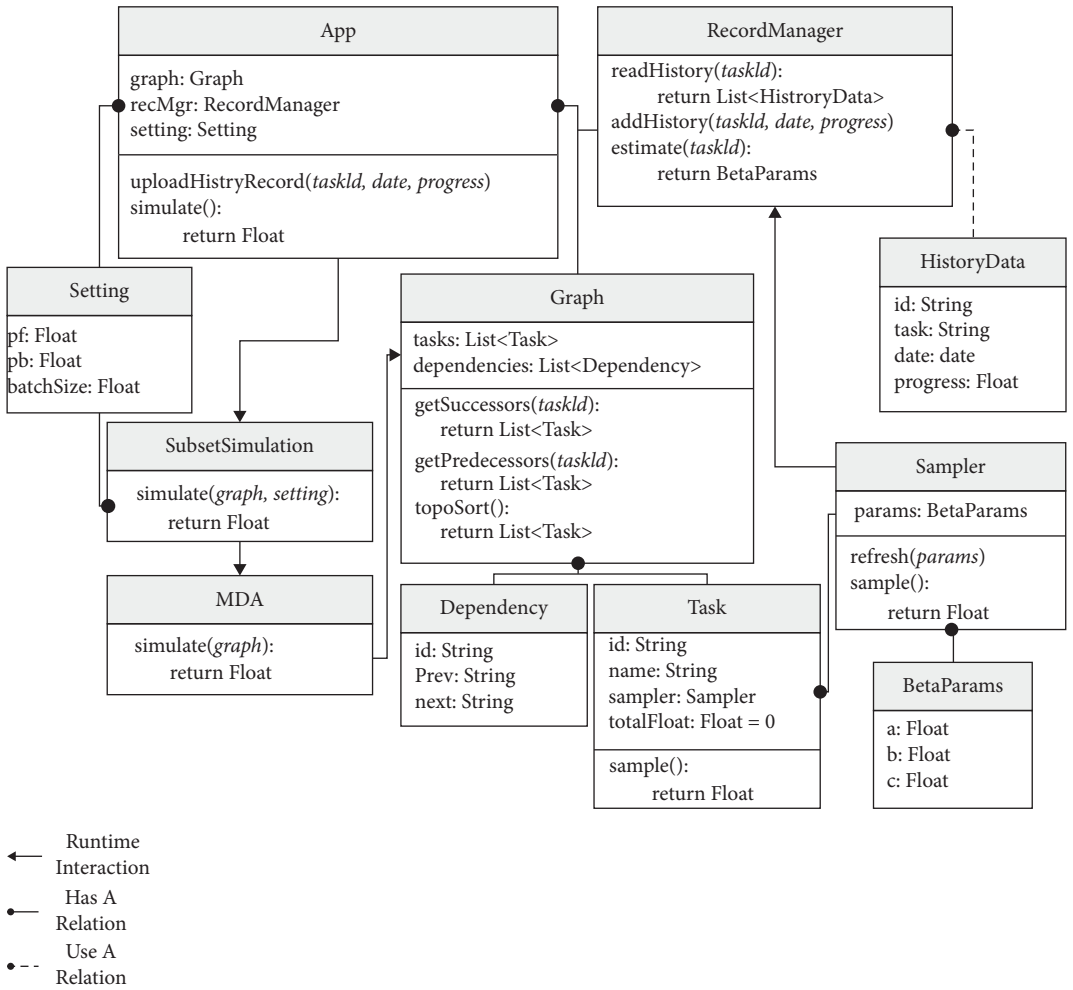


Figure 4: UML class diagram of simulation system.

(1) Persistent constant: these data cannot be changed and stored persistently in memory, like graph structure and history records

(2) User-defined variable: these data are generated by user interactions, like configuration parameters of simulation

(3) Intermediate variable: these data are generated by program and will only be used once, like beta distribution parameters and sample

The data flow diagram explaining how data are passed between classes is shown in Figure 5.

5.2. Process of Simulation System. The following process describes all the steps in our simulation system, including the creation of the schedule network, distribution estimation, and estimation for failure probability.

(1) *Schedule Network Construction*. Construct the schedule network using an adjacency list, and presort the network using Khan's algorithm to get the topological order [22]. In this step, the network could be validated; for example, circular working dependency could be detected. Besides, precomputed topo-order could save unnecessary computing time in subset simulation sampling.

(2) *Task Duration Distribution Estimation*. Use empirical parameters and observational data to estimate the time distribution for each task. This step is thoroughly described in the previous section.

(3) *Failure Probability Estimation*. Estimate the failure probability of a given schedule network using subset simulation and check whether the answer is acceptable. Usually, a criterion is chosen (e.g., 1% or 5%). If the estimation is higher than this criterion, we have to consider adjusting our plan.

(4) *Schedule Adjustment*. If failure probability is unacceptable, we have to find out the critical activities in the schedule like in classical CPM. By averaging the total float in samples, we could find those critical activities with small floats. We could allocate more resources to these activities to reduce the completion time and further reduce the failure probability.

5.3. Features and Limitations. There are three practical features in our proposed simulation system:

(1) *Schedule Evaluation*. Using the algorithm in Section 3, we could use a failure probability to quantize the delay risk. If this probability is too high, a project manager needs to change the schedule or extend the deadline. To modify the plan, we have to find critical activities and lower their duration. To prolong the deadline, we can use the empirical distribution of completion time to select a proper deadline.

(2) *Critical Activities Detection*. When there is a high possibility for a schedule to be late, critical activities could be located using a threshold of total float. If the total float of a task is lower than this threshold, it is a critical activity. This threshold describes the flexibility we can manage. A lower threshold will sift out fewer activities and usually requires a better management capability.

(3) *Task Duration Prediction*. When the records are stored, the estimation of task duration will also tell us the most likely finish date. We could compare this date to the planned finish date. If there is an unacceptable lag, this activity would be marked. And the system will alert the project manager. This process is automatic and data-driven and could improve project manager's efficiency and provide more information.

All these three features are implemented through two web pages, one for collecting progress data and one for visualizing and analyzing project progress; see Figure 6.

Though there are some practical features in our method, there are still some limitations we are striving to solve. First, the threshold may not be a proper way of finding critical activities. Since there is a possibility that an improper threshold may select all or none of the tasks, such a threshold would be unhelpful. This fact may make threshold selection a potential problem. Besides, the total float threshold only captures the average behaviour of a task. It could help reduce the risk of schedule while reducing the variance of task duration will also work. Currently, there is no indicator describing the relationship between the variance of task durations and the risk of schedule in our method.

6. Illustrative Examples and Computational Results

This section will verify the performance of our proposed method and compare it with DMCS, Soroush's LUBE [2], and Chen's FCPM [10] in terms of precision, efficiency, and stability.

6.1. Soroush's Benchmark Example. In this part, we will use an illustrative example from Soroush's work. This hypothetical test case consists of 21 activities shown in Figure 7. Our experiment also takes the assumptions in Soroush's work. The activity times are assumed to be beta distributed, and events v_1 and v_{14} are the starting and terminal events, respectively. The optimistic time a_k, most likely time m_k, and pessimistic time b_k for each activity are given in the parentheses beside that activity. To compare our work to Fuzzy CPM (FCPM), each activity's fuzzy duration is formulated in a triangular fuzzy number $d_k = (a_k, m_k, b_k)$ using the same parameters in PCPM.

For each method, we repeat simulations 20 times to get the mean estimation $\widehat{p}_f$ and its average running time $\overline{t}$. And to compare the stability of numerical results, variation coefficients $c_v = \sigma_{p_f}/\widehat{p}_f$ are also computed. As for the setting of sampling-based methods, 50,000 samples were used in DMCS, and a batch size of $N_b = 5000$ and selection ratio $p_s = 0.1$ were used to configure the Subset Simulation. Our

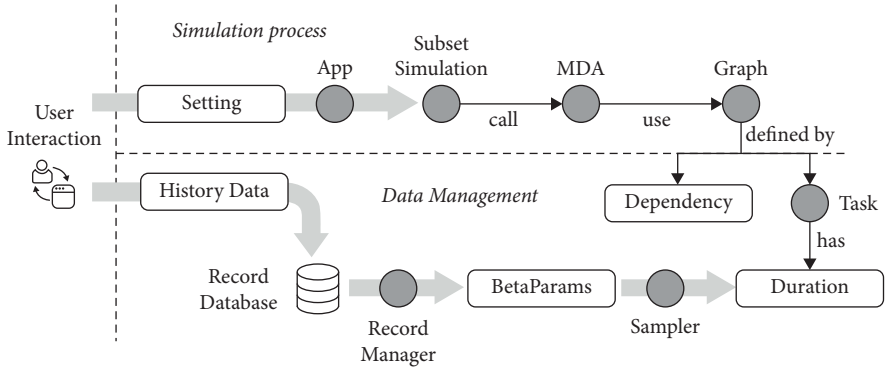
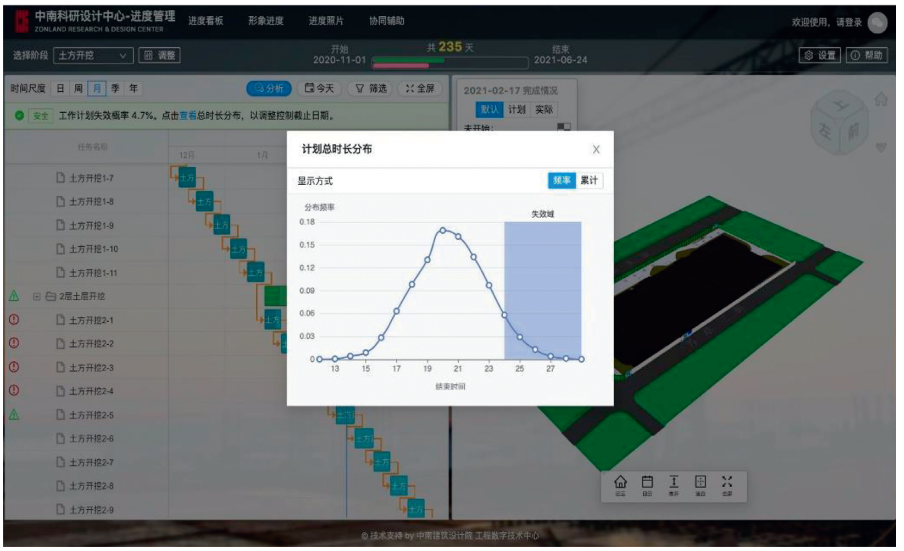


Figure 5: Data flow diagram of our system.

(a)



(b)

Figure 6: Screenshot of our simulation system. (a) Task Kanban: this page is used to display and submit current working progress; project manager could drag the slider to update the progress of a task. (b) Schedule evaluation: this page is used to visualize current working progress and evaluate the reliability of schedule. Project manager could find the dangerous tasks marked by a red exclamation in a circle from analysis.

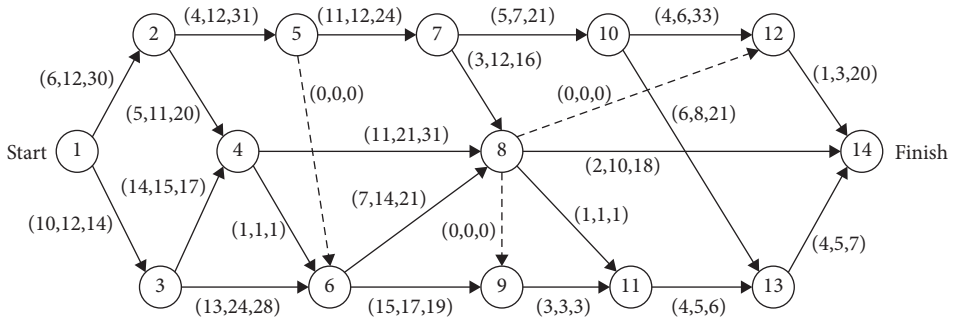


Figure 7: Hypothetical schedule network.

experiments were conducted on a PC with an 8-core Intel Core i7-0700K 3.6 GHz CPU. All the programs were coded in Python 3.7.3 and NumPy 1.20.1.

Computation results are shown in Table 1. Since Direct MCS and our method are based on sampling, generally they are slower than the one-pass method, like LUBE and FCPM. This is a common shorthand of sampling-based PCPM, but subset simulation accelerates this process while nearly retaining the same precision and stability. This makes the application of PCPM possible. On the other hand, compared to the one-pass method, subset simulation-based PCPM reaches the lowest error compared to Direct MCS. This result shows that subset simulation could provide a more reliable solution. In all, subset simulation combines computation performance and efficiency. These make it a probably good way to solve the probabilistic CPM problem.

By drawing the relationship between completion time limit and failure probability in Figure 8, we also find that the curve given by subset simulation is the closest one to the DMCS one, while there exists a systematic error in LUBE and FCPM. This observation could be explained by the fact that LUBE only focuses on one most critical path, while other paths may also lead to failure, leading to a higher failure probability. Fuzzy numbers and fuzzy operations cannot capture all the natural complexity of probabilistic modelling. Thus, deviation will occur. In contrast, subset simulation could depict this relationship very well. This would provide the project manager a reliable decision basis.

Another important indicator, total float, is also computed. A precise estimation of float would help the project manager identifying important tasks and allocate resources to them. Figure 9 shows that the total float estimation given by FCPM is lower than the ones given by PCPM, which may lead to more resource demand and hurried plans, while subset simulation gives a similar result to DMCS, and numerical errors are acceptable.

6.2. A Schedule Network in Real Project. In order to verify the performance of our algorithm in real practice, a schedule network from a real project containing 173 activities and 195 work dependencies is used. This project is a multifunctional building, including three parts: the main tower for office usage, the auxiliary building for business, and the basement for parking. The construction work is broken down in the following way: the basement is divided into 24 individual working segments each floor and auxiliary building is divided into 3 individual working segments each floor. Each segment is constructed floor by floor, which creates working dependencies. The division of our project and schedule network are illustrated in Figure 10.

Table 1: Schedule network failure evaluation comparison (deadline = 85 days).

Methods		Failure probability (%)	Variation coefficient	Running time (sec.)
MCPM	LUBE	1.93	N/A	0.00
FCPM	Triangular	19.57	N/A	0.00
PCPM	Direct MCS	2.25	2.45%	2.58
	Our method	2.24	3.11%	0.35

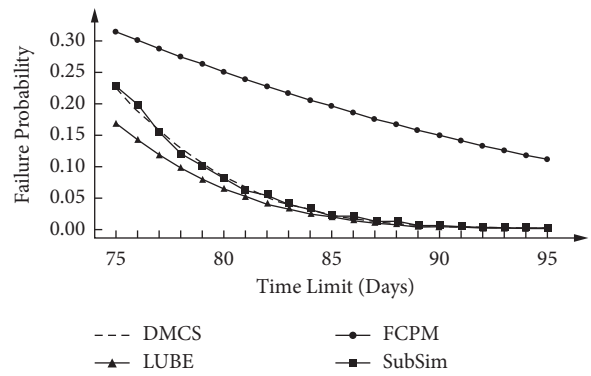


Figure 8: Failure probability curve.

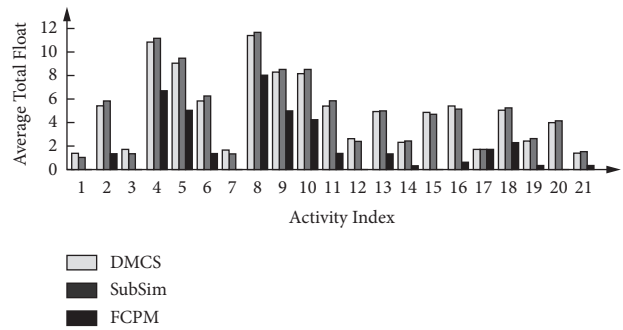


Figure 9: Average total float comparison.

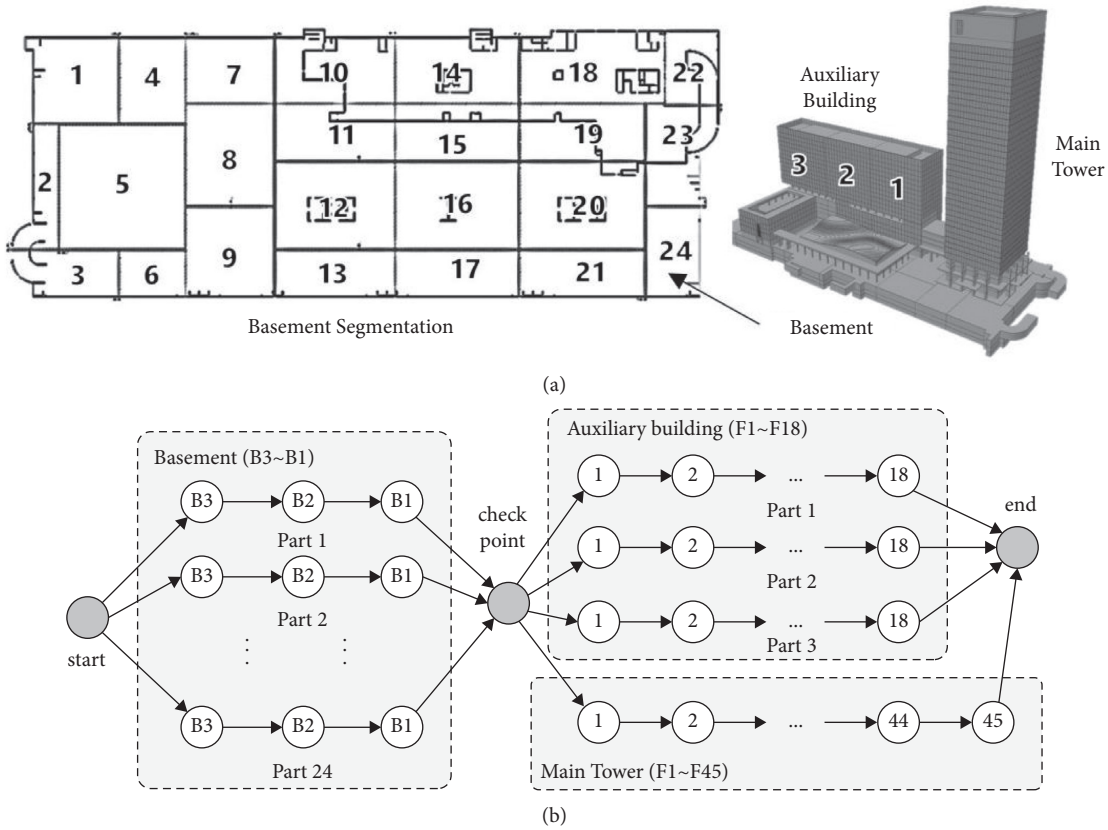


Figure 10: A schedule network of a real project. (a) Illustration of working segments: each segment is composed of several working packages and constructed floor by floor. (b) Illustration of working dependencies: there are three special nodes which are a starting point, an ending point, and a midway checkpoint. Their durations are assigned with zero day.

The activity times of each working segment are given in Table 2.

The hardware and software environments of this experiment are consistent with those of the previous one. As we can see in Table 3, comparing with other methods, our estimation is the most closet to the DMCS result. And LUBE is still lower than ours, which will be explained in the following discussion. Another point worth noting is that with the growth of the schedule network's size, our method's computational efficiency advantage becomes more obvious compared with DMCS. And its running time is acceptable in real practice. Project managers can get more accurate estimates while waiting less time.

By drawing the relationship between completion time limit and failure probability in Figure 11, we also find that curve given by our method is the closest one to the DMCS one. Still, LUBE curve is under DMCS curve, which means it underestimates the overdue risk. This fact is caused by the inherent flaw of LUBE and would mislead project managers to make an excessively optimistic estimation, which may cause potential overdue risks. But our method solved this problem well: since our method will traverse the whole schedule network, rather than one path, our PCPM will give a comprehensive consideration which depicts the risk of the network as a whole, rather than only one "most critical" path. This mechanism makes our method work well for the full range of time limit.

Table 2: Schedule network failure evaluation comparison.

Part	Floors	Working segment	Duration (days/floor/segment)
Basement	B3~B1	1, 5~9, 21	30
		2~4, 10~20, 22~24	25
Auxiliary building	F1~F18	1, 2, 3	30
Main tower	F1~F45		12

Table 3: Schedule network failure evaluation comparison (deadline = 690 days).

Methods		Failure probability (%)	Variation coefficient	Running time (sec.)
MCPM	LUBE	2.83	N/A	0.54
FCPM	Triangular	31.13	N/A	0.26
PCPM	Direct MCS	3.11	2.17%	26.32
	Our method	3.13	5.27%	1.32

6.3. Discussions. From the experiments above, we can see that although those one-pass algorithms could give results quickly, due to the inherent deficiencies of algorithm design, there will be systematic errors. For example, LUBE fails when the failure probability is relatively high. This could be explained easily by a very simple example. Supposing there is

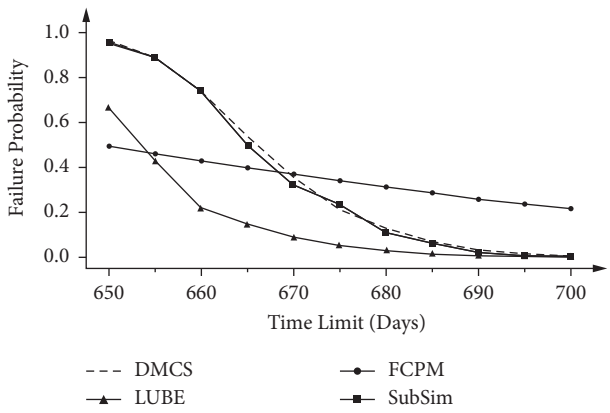


Figure 11: Failure probability curve.

a project containing N identical paths. And each path has an independent failure probability q_f. Thus, we can easily get the failure probability of this project using some basic probability theory. The right answer should be $p_f = 1 - (1 - q_f)^N$. However, according to LUBE, the failure probability is defined by the "most critical" path. Since all the paths are identical, LUBE estimation is q_f. Notice that

$$\begin{aligned} p_f &= 1 - \left(1 - q_f\right)^N > q_f \\ &= 1 - \left(1 - q_f\right)^1 \forall N > 1. \end{aligned} \tag{15}$$

So, LUBE estimation is always lower than the failure probability of the whole schedule. This difference is especially obvious when the time limit is relatively tight. In this situation, the failure probability of each path will increase. For $0 < a < 1$, $a^N \approx a$ only works when $a \approx 1$, which means q_f shall be very small. However, with a tight time limit, q_f cannot be small, which makes LUBE fail to approximate. Besides if there is more than one critical path, according to equation (15), the difference between p_f and q_f will become obvious. But this situation is common in those projects where parallel construction is needed. Besides, in most cases, paths are inherently correlated. If a node is overdue, all the paths containing this node will be affected.

These observations tell us when dealing with the schedule network, it will be better to treat the schedule as a whole, rather than a group of single paths. Though we have to admit sampling-based PCPM takes time during sampling, it is still worth doing it, since it can provide a more accurate estimation. From the experiments above, we can find that subset simulation can accelerate the sampling process greatly, making the running time of PCPM acceptable.

7. Conclusions

This paper proposed a new PCPM based on a data-driven subset simulation to solve the Schedule Network Failure Problem in a dynamic way. This method could compute the failure probability efficiently and effectively without loss of any result accuracy and outperforms the other approaches with the same assumptions. Besides, another important contribution of our study is to plug a data-based task duration distribution estimation into PCPM. This finding provides a more objective way to estimate task duration distribution, reducing the variance in project managers' experience and improving the knowledge sharing between teams. These key features would provide a good foundation for applying PCPM in practical management affairs.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this study.

References

[1] S. A. Mubarak, *Construction Project Scheduling and Control*, John Wiley & Sons, Hoboken, NJ, USA, 2015.

[2] H. M. Soroush, "The most critical path in a PERT network: a heuristic approach," *European Journal of Operational Research*, vol. 78, no. 1, pp. 93–105, 1994.

[3] F. Marle and L.-A. Vidal, "Project management traditional principles," in *Managing Complex, High Risk Projects*, pp. 1–52, Springer, Berlin, Germany, 2016.

[4] A. Rafidain, "Design genetic algorithm to find the optimal critical path network project (GAOCPN)," *Journal of Computer Sciences and Mathematics*, vol. 9, no. 1, pp. 187–210, 2012.

[5] A. Aghaie and H. Mokhtari, "Ant colony optimization algorithm for stochastic project crashing problem in PERT networks using MC simulation," *The International Journal of Advanced Manufacturing Technology*, vol. 45, no. 11-12, pp. 1051–1067, 2009.

[6] C. L. Liu and J. M. He, "An algorithm for selection of the most critical path in a project network," *Journal of Systems Engineering*, vol. 15, no. 2, pp. 136–142, 2000.

[7] S. H. Nasution, "Fuzzy critical path method," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 1, pp. 48–57, 1994.

[8] S. Chanas and P. Zieliński, "Critical path analysis in the network with fuzzy activity times," *Fuzzy Sets and Systems*, vol. 122, no. 2, pp. 195–204, 2001.

[9] M. Mazlum and A. F. Güneri, "CPM, PERT and project management with fuzzy logic technique and implementation on a business," *Procedia-Social and Behavioral Sciences*, vol. 210, pp. 348–357, 2015.

[10] S.-P. Chen and Y.-J. Hsueh, "A simple approach to fuzzy critical path analysis in project networks," *Applied Mathematical Modelling*, vol. 32, no. 7, pp. 1289–1297, 2008.

[11] D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar, "Application of a technique for research and development program evaluation," *Operations Research*, vol. 7, no. 5, pp. 646–669, 1959.

[12] D. L. Cook, *Program Evaluation and Review Technique: Applications in Education*, US Department of Health, Education, and Welfare, Office of education, Washington, DC, USA, 1966.

[13] D. Monhor, "A new probabilistic approach to the path criticality in stochastic PERT," *Central European Journal of Operations Research*, vol. 19, no. 4, pp. 615–633, 2011.

[14] D. E. Lee, T. H. Bae, and D. Arditi, "Advanced stochastic schedule simulation system," *Civil Engineering and Environmental Systems*, vol. 29, no. 1, pp. 23–40, 2012.
[15] Y. Takakura, T. Yajima, Y. Kawajiri, and S. Hashizume, "Application of critical path method to stochastic processes with historical operation data," *Chemical Engineering Research and Design*, vol. 149, pp. 195–208, 2019.
[16] S.-K. Au and J. L. Beck, "Estimation of small failure probabilities in high dimensions by subset simulation," *Probabilistic Engineering Mechanics*, vol. 16, no. 4, pp. 263–277, 2001.
[17] S. K. Au and J. L. Beck, "Subset simulation and its application to seismic risk based on dynamic analysis," *Journal of Engineering Mechanics*, vol. 129, no. 8, pp. 901–917, 2003.
[18] S. K. Au, J. Ching, and J. L. Beck, "Application of subset simulation methods to reliability benchmark problems," *Structural Safety*, vol. 29, no. 3, pp. 183–193, 2007.
[19] M. X. Wang, X. S. Tang, D. Q. Li, and X. H. Qi, "Subset simulation for efficient slope reliability analysis involving copula-based cross-correlated random fields," *Computers and Geotechnics*, vol. 118, Article ID 103326, 2020.
[20] D.-Q. Li, T. Xiao, Z.-J. Cao, C.-B. Zhou, and L.-M. Zhang, "Enhancement of random finite element method in reliability analysis and risk assessment of soil slopes using Subset Simulation," *Landslides*, vol. 13, no. 2, pp. 293–303, 2016.
[21] N. R. Shankar and V. Sireesha, "Using modified Dijkstra's algorithm for critical path method in a project network," *International Journal of Computational and Applied Mathematics*, vol. 5, no. 2, pp. 217–225, 2010.
[22] A. B. Kahn, "Topological sorting of large networks," *Communications of the ACM*, vol. 5, no. 11, pp. 558–562, 1962.
[23] W. J. McBride and C. W. Mcclelland, "PERT and the beta distribution," *IEEE Transactions on Engineering Management*, vol. 14, no. 4, pp. 166–169, 1967.
[24] D. Golenko-Ginzburg, "On the distribution of activity time in PERT," *Journal of the Operational Research Society*, vol. 39, no. 8, pp. 767–771, 1988.
[25] E. F. Udoumoh and D. W. Ebong, "A review of activity time distributions in risk analysis," *American Journal of Operations Research*, no. 6, pp. 356–371, 2017.