

Efficient Probabilistic Reanalysis and Optimization of a Discrete Event System

2011-01-1081

Published
04/12/2011

Yibo Li

Efstratios Nikolaidis
Univ. of ToledoZissimos Mourelatos
Oakland Univ

Copyright © 2011 SAE International

doi:[10.4271/2011-01-1081](https://doi.org/10.4271/2011-01-1081)

ABSTRACT

This paper presents a methodology to evaluate and optimize discrete event systems, such as an assembly line or a call center. First, the methodology estimates the performance of a system for a single probability distribution of the inputs. Probabilistic Reanalysis (PRRA) uses this information to evaluate the effect of changes in the system configuration on its performance. PRRA is integrated with a program to optimize the system. The proposed methodology is dramatically more efficient than one requiring a new Monte Carlo simulation each time we change the system. We demonstrate the approach on a drilling center and an electronic parts factory.

1. INTRODUCTION

In today's competitive business environment companies must accurately predict the performance and cost of complex systems, such as a factory, a service center or a transportation center before building them. Industrial, manufacturing and mechanical engineers in the aerospace, automotive, finance, and shipbuilding industries use sophisticated simulation programs for discrete event systems such as Arena, and Extend. These tools enable analysts to model systems, assess their performance, and optimize them.

Engineers often face the problem of minimizing (or maximizing) a function $h = g(u)$, where variables h and u are both constrained over finite feasible sets. If variable u is deterministic, the solution to the above problem can be

obtained using existing optimization algorithms. On the other hand, if u is random, with a known probability distribution, stochastic programming should be employed.

The optimization of the performance attributes of a discrete event system by changing the parameters of the probability distributions of the input variables is an example of the above optimization problem. For example, in an assembly system for electronic parts, engineers want to minimize the average total time of a part in the system by changing the mean values of the service time in each process without exceeding a budget. This requires building a simulation model and estimating repeatedly the average performance of a system's attribute (e.g. the waiting time of a part in a preparation area) for many different probability distributions of the input variables.

Development of efficient and robust optimization methods, together with increasing computational power has enabled engineers to solve such problems. However, the solution often proves impractical for the following reasons. First, there can be considerable uncertainty in the probability distributions of some random variables. Consider that in the above example, the service time in the preparation area is a random variable, and its probability distribution has one or more unknown parameters. Second, the solution requires repeated analyses of a measure of merit of a system, such as the expected utility, for many different probability distributions of the input variables. In a complex system, such as the electronic part assembly system, these analyses require

Monte Carlo simulation with thousands of replications. As a result, it is impractical to optimize the system.

This paper presents a computationally efficient methodology, which can reduce significantly the computational cost by performing only one Monte Carlo simulation. It integrates simulation and a probabilistic reanalysis (PRRA) method to solve optimization problems in which decision variables are the mean values of random variables such as the service time at a station. The methodology simulates a discrete event system using the commercial program Arena. By reweighting the data from the simulation, PRRA computes the performance attributes of a system for many different probability distributions of the decision variables without performing a new simulation. This enables a designer to identify the effect of changes in the system on its performance and to optimize the system without incurring the high computational cost of simulation.

Four areas, specifically discrete-event simulation, optimization of discrete event systems, probabilistic reanalysis, and variance reduction are important to the research presented in this paper. Leeming [1] compares three languages (GPSS, ECSL, and SIMULA 67) that are used to represent and simulate a discrete event system. Hong Zhang et al. [2] provide a method to model uncertain activity durations through triangular or trapezoidal fuzzy numbers. Ho et al. [3] propose a gradient surface method (GSM) for optimization of discrete event dynamic systems. Debus [4] presents an approach called Reactive thermostatical search (RTS), which can be used to optimize discrete event systems. Glover [5] discusses OptQuest in the broader context of recent advances in simulation optimization.

Farizal and Nikolaidis [6] introduce a PRRA approach for estimating lower and upper bounds of the probability of failure of a system when the parameters of the probability distributions of the random input variables are known up to their lower and upper bounds. They demonstrate the approach on a single degree of freedom system with a dynamic vibration absorber with uncertain natural frequencies. They show that PRRA can estimate bounds for the probability of failure at one fiftieth of the cost of Monte Carlo simulation. Fonseca et al. [7] use PRRA for optimization problems in which design variables are the mean values of some random variables. Fonseca et al. [8] also present efficient techniques to identify and quantify variability in the parameters from experimental data by maximizing the likelihood of the measurements, using Monte Carlo or perturbation methods for the likelihood computation. Beckman and McKay [9] present two efficient methods that allow an analyst to calculate the mean value of a function of random variables when the analyst changes the probability distributions of the input variables without rerunning the computer codes for Monte Carlo simulation. Au and Beck [10] present an

adaptive importance sampling method for evaluating the multidimensional integrals in reliability analysis of structures.

The primary focus of this paper is to explore a methodology that integrates simulation modeling and probabilistic reanalysis method (PRRA) to predict the average utility of a discrete event system for different probability distributions of the input variables by performing a single Monte Carlo simulation for one probability distribution of the inputs. Another objective of this work is to integrate the methodology with an optimization technique to solve the above type of optimization problems for a discrete event system.

Section 2 describes the methodology for probabilistic reanalysis and optimization of discrete event systems. The next section demonstrates the methodology on probabilistic analysis of a drilling center system. The same section and applies the methodology on optimization of an electronic part assembly system to find the best combination of input variables that minimize the average total time of a part in the system.

2. METHODOLOGY

2.1. DESCRIPTION OF PROBABILISTIC REANALYSIS (PRRA)

This section reviews the importance sampling method for estimation of the average value of a function of random variables. Then, it presents the efficient PRRA approach and briefly discusses the selection of a sampling probability distribution.

2.1.1. Importance Sampling

Importance sampling is a general Monte Carlo method for computing integrals accurately with fewer samples than in standard Monte Carlo simulation. Two major steps are involved in importance sampling. The first is variation of the original input process. Instead of drawing samples from the original probability density function (PDF), samples are drawn from other PDFs, called importance density functions, such that “important” regions of the sample space get more samples. The second step is correction of the distortion by averaging the output from different samples (realizations) using weights the method yields unbiased estimates of the mean value of the quantity of interest [11].

Let U is a random variable with a PDF $f_U(u)$. We wish to estimate the expected value of function $h = g(u)$, where g is a specified, deterministic function (operator). We compute the expected value of the quantity of interest $h = g(u)$ using:

$$E(h) = E[g(u)] = \int g(u) f_U(u) du \quad (2.1)$$

Computation of this expectation requires calculating integrals for which there is no close form. Numerical calculation depends on the ability to sample from $f_U(u)$. We can improve the efficiency of Monte Carlo simulation by sampling from a distribution $f^S_U(u)$ that is different than the true distribution $f_U(u)$ and use importance sampling to estimate the integral in (2.1) as:

$$E(h) = E[g(u)] = \int g(u) \frac{f_U(u)}{f^S_U(u)} f^S_U(u) du \quad (2.2)$$

We base our estimates on a sample of I independent points $u_1, \dots, u_I$, generated from the sampling distribution $f^S_U(u)$. For each sample value u_i , we compute the likelihood ratio as follows:

$$\omega_i = \frac{f_U(u_i)}{f^S_U(u_i)} \quad (2.3)$$

Then we can estimate the expectation of $g(u)$ with respect to the true distribution $f_U(u)$:

$$E(h)_S = E[g(u)]_S = \frac{\sum_{i=1}^I \omega_i g(u_i)}{I} \quad (2.4a)$$

where subscript “S” signifies that the expected value of h is estimated from standard importance sampling. An unbiased estimator of the standard deviation of the average value of function h is:

$$sd_{E(h)} = \frac{1}{\sqrt{I}} \sqrt{\frac{1}{I-1} \sum_{i=1}^I [\omega_i g(u_i)]^2 - \frac{I}{I-1} E(h)^2} \quad (2.5)$$

The $1 - \alpha$ level confidence interval of the mean value of h is:

$$E(h) \pm t_{I-1, 1-\frac{\alpha}{2}} \cdot \frac{sd_{E(h)}}{\sqrt{I}} \quad (2.6)$$

In the above equation $t_{I-1, 1-\frac{\alpha}{2}}$ is the point that has below it probability $1 - \frac{\alpha}{2}$ for Student's t -distribution with $I-1$ degrees of freedom. This interval covers the true system probability of failure with probability $1 - \alpha$. Usually, the 95% confidence interval is calculated, that is, $\alpha = 0.05$

In equation (2.4a), since the expected value of ω_i under $f^S_U(u_i)$ is 1, we need only to compute weights up to a constant of proportionality and then normalize:

$$E(h)_{NR} = E[g(u)]_{NR} = \frac{\sum_{i=1}^I \omega_i g(u_i)}{\sum_{i=1}^I \omega_i} \quad (2.4b)$$

where subscript “NR” signifies normalized importance sampling.

2.1.2. Efficient PRRA

In Eq. (2.1), if we change random variable U to a random vector $\mathbf{X}$, then this equation becomes:

$$E(h) = E[g(\mathbf{x})] = \int g(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} \quad (2.7)$$

where $h=g(\mathbf{x})$ is the objective function of vector $\mathbf{X}$, and $f_{\mathbf{X}}(\mathbf{x})$ is the joint probability density function of random vector $\mathbf{X}$.

The key idea of PRRA is to use the original sample values from a sampling PDF to estimate the expected value of a function $g(\mathbf{x})$ of the random variables for other PDFs. If one

generates and saves a sample from $f^S_{\mathbf{X}}(\mathbf{x})$, then one can reuse them to estimate the expected value of $g(\mathbf{x})$ for another PDF $f_{\mathbf{X}}(\mathbf{x})$

$$E(h)_S = E[g(\mathbf{x})]_S = \frac{1}{I} \sum_{i=1}^I \omega_i g(\mathbf{x}_i) \quad (2.8)$$

$$\omega_i = \frac{f_{\mathbf{X}}(\mathbf{x}_i)}{f^S_{\mathbf{X}}(\mathbf{x}_i)}$$

where $\frac{f_{\mathbf{X}}(\mathbf{x}_i)}{f^S_{\mathbf{X}}(\mathbf{x}_i)}$ is the likelihood ratio. Equation (2.8) enables one to estimate the average value of the objective function or a constraint in an optimization problem for many probability distributions of the input variables very efficiently, because this equation only requires calculation of the value of the objective function or a constraint for only one

sample drawn from sampling PDF $f^S_{\mathbf{X}}(\mathbf{x})$ Equation (2.8)

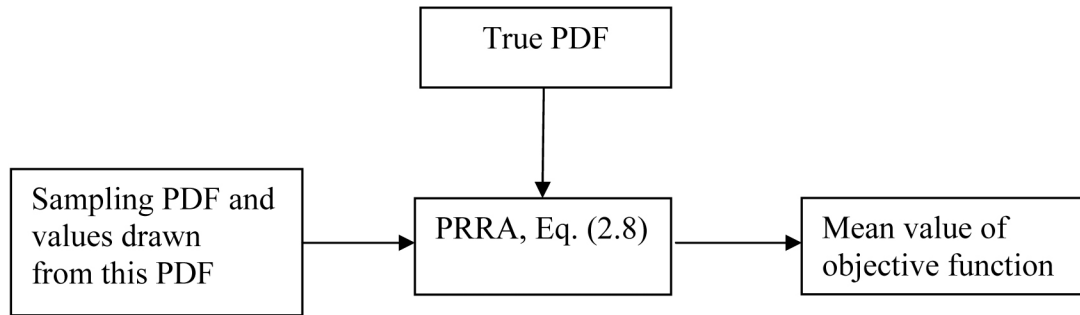


Figure 1. Procedure to estimate mean value of a function using PRRA

requires calculation of the ratios of PDFs, which is much cheaper than calculation of the objective function or the constraints.

The algorithm of PRRA for estimation of the average value of a function for many different distributions of the random variables is described below:

1. Select sampling PDFs $f_X^S(\mathbf{x})$ for the random vector.
2. Generate sample values of the random variables
3. Calculate the value of objective function $h=g(\mathbf{x}_i)$ for the sample values in step 2.
4. Estimate the average value of an objective function for other probability distribution from Eq. (2.8), the standard deviation of the estimate and a confidence interval from Eqs. (2.5) and (2.6).

Figure 1 shows the flow of the PRRA algorithm. Inputs are the sampling PDFs of the random variables and the sample values generated from the sampling PDFs. Outputs are the average value of the objective function for another probability distribution and the confidence interval for the estimated average value.

2.2. PROBABILISTIC REANALYSIS (PRRA) FOR DISCRETE EVENT SYSTEMS

In design problems involving discrete event systems, engineers often want to estimate the performance attributes of a system for many different probability distributions of the input variables. These variables can be categorized into random variables $\mathbf{X}$ and design (or decision) variables $\mathbf{Y}$. Design variables can be the distribution parameters of one or more random variables. For example, in a drilling center system, the service time of a drilling press is a random variable, and the distribution parameters of service time such as the mean value and standard deviation can be the design variables. Instead of performing repeated Monte Carlo simulations, we can estimate the expected value of a performance attribute of a discrete event system for many probability distributions of the input variables using PRRA.

We want to estimate the expected value of a performance attribute of a discrete event system. This attribute $h=g(\mathbf{x}, \mathbf{y})$ is a function of both the random and design variables, $\mathbf{X}$ and $\mathbf{Y}$, respectively. Eq. (2.8) can be used to estimate the expected value of the performance attribute. We perform K replications, each with I entities (e.g. blanks in a drill) going through the system. Therefore, $I \cdot K$ sample values are drawn from the sampling PDF of each random variable and are used to estimate the expectation of a performance attribute for many different probability distributions of the input variables. Let $\mathbf{x}_{i,k}$, $\mathbf{y}_{i,k}$ be the i^{th} sample values of $\mathbf{X}$ and $\mathbf{Y}$ in the k^{th} replication. These are ordered according to the time that the corresponding entities enter the system. Let $f_X(\mathbf{x}_{i,k})$, $f_Y(\mathbf{y}_{i,k})$ and $f_X^S(\mathbf{x}_{i,k})$, $f_Y^S(\mathbf{y}_{i,k})$ be true and sampling PDFs, respectively. We can calculate the mean value of the performance attribute $h=g(\mathbf{x}, \mathbf{y})$ for all I entities in the k^{th} replication as follows:

$$E[h(k)]_s = \frac{1}{I} \sum_{i=1}^I \omega_{i,k} g(\mathbf{x}_{i,k}, \mathbf{y}_{i,k}) \quad (2.9)$$

where $\omega_{i,k}$ is the likelihood ratio. When we calculate the likelihood ratio, the weights of all entities processed before the i^{th} entity must be included in the equation for the weight $\omega_{i,k}$. This is because in a discrete event system, the performance attributes of the current entity are affected by the previous ones. For example, in an electronic parts assembly system, the waiting time of the i^{th} part at a certain step is affected by the waiting and the service times of the previous parts. However, this calculation of weights would be too expensive. Therefore, we consider only the last δ entities before the i^{th} entity in the equation for the weight. The likelihood ratio can be calculated as follows:

$$\omega_{i,k} = \frac{\prod_{j=0}^{\delta-1} f_X(\mathbf{x}_{i-j,k}) f_Y(\mathbf{y}_{i-j,k})}{\prod_{j=0}^{\delta-1} f_X^S(\mathbf{x}_{i-j,k}) f_Y^S(\mathbf{y}_{i-j,k})} \quad (2.10)$$

Symbol δ , which is called decorrelation distance, represents the number of previous entities that affect the current one. To choose the value of the decorrelation distance, we calculate the correlation coefficient between of the performance attributes of the i^{th} and $(i - \delta)^{th}$ entities:

$$\rho[g(\mathbf{x}_{i,k}, \mathbf{y}_{i,k}), g(\mathbf{x}_{i-\delta,k}, \mathbf{y}_{i-\delta,k})] = \frac{\text{cov}[g(\mathbf{x}_{i,k}, \mathbf{y}_{i,k}), g(\mathbf{x}_{i-\delta,k}, \mathbf{y}_{i-\delta,k})]}{\text{stdev}[g(\mathbf{x}_{i,k}, \mathbf{y}_{i,k})] * \text{stdev}[g(\mathbf{x}_{i-\delta,k}, \mathbf{y}_{i-\delta,k})]} \quad (2.11)$$

where “cov” is covariance and “stdev” is standard deviation. If the correlation coefficient is less than 0.05, the effect between the two entities is ignored. The minimum value of δ in Eq. (2.11), which makes the correlation coefficient less than 0.05, is selected as decorrelation distance.

One difficulty in PRRA is that a very large number of replications are needed when the performance attributes of entities are strongly correlated. When the decorrelation distance exceeds 15 entities, a very large number of replications are required (e.g. more than 10,000) and this makes the methodology impractical.

In next step, we calculate the mean value of Eq. (2.9) for all K replications. The expectation of the performance attribute of a discrete event system for many probability distributions of the input variables is calculated as follows:

$$EP[h(\delta)]_s = \frac{\sum_{k=1}^K \omega_{i,k} g(\mathbf{x}_{i,k}, \mathbf{y}_{i,k})}{K} \quad (2.12a)$$

Eqs. (2.5) and (2.6) can be easily modified to calculate the standard deviation and confidence interval of $EP[h(\delta)]$

$$sd_{EP[h(\delta)]} = \frac{1}{\sqrt{K}} \sqrt{\frac{1}{K-1} \sum_{k=1}^K \{E[h(k)]\}^2 - \frac{K}{K-1} EP[h(\delta)]^2} \quad (2.13)$$

The $1 - \alpha$ level confidence interval of $EP[h(\delta)]$ is:

$$EP[h(\delta)] \pm t_{K-1, 1-\frac{\alpha}{2}} \cdot \frac{sd_{EP[h(\delta)]}}{\sqrt{K}} \quad (2.14)$$

The choice of the sampling PDF affects the accuracy of the estimation of the expectation of the performance attribute. Generally, we want the sampling distribution to be such that the limit converges rapidly to the value of the integral. The support of the sampling PDF should contain that of the true PDF (Figure 2); otherwise the estimator of the mean value of an attribute will be biased. In addition, the support of the sampling PDF should not extend to regions where the true PDF is very small, because this will lead to waste of sample values.

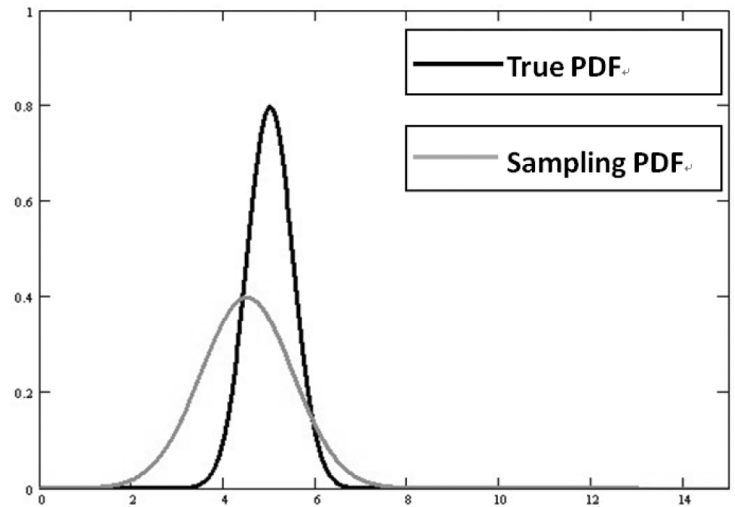


Figure 2. Comparison of the true PDF $f_X(x)$ and sampling PDF $f_X^S(x)$

We compare the efficiency of PRRA with that of standard Monte-Carlo simulation by calculating the Effective Sample Size (ESS). This is the number of replications of a standard Monte-Carlo simulation that would estimate the average performance attribute with the same accuracy as PRRA. ESS is calculated as follows,

$$ESS = \frac{\text{var}[g(\mathbf{x}_{i,k}, \mathbf{y}_{i,k})] / E[g(\mathbf{x}_{i,k}, \mathbf{y}_{i,k})]^2}{\text{var}\{EP[h(\delta)]\} / EP[h(\delta)]^2} \quad (2.15)$$

where $\text{var}[g(\mathbf{x}_{i,k}, \mathbf{y}_{i,k})]$ and $E[g(\mathbf{x}_{i,k}, \mathbf{y}_{i,k})]$ are the variance and mean value of the sample performance attribute. In most cases, ESS is less than or equal to K , because PRRA has to estimate the mean values of the performance attribute for many probability distributions. However, PRRA is still more efficient than standard Monte Carlo simulation because it

does not perform a new Monte Carlo simulation, for each probability distribution of the input variables.

From Eq. (2.4b), we can also get the normalized PRRA equation as follows:

$$EP[h(\delta)]_{NR} = \frac{\sum_{k=1}^K \sum_{i=1}^I \omega_{i,k} g(x_{i,k}, y_{i,k})}{\sum_{i=1}^I \omega_{i,k}} \quad (2.12b)$$

Equation (2.12b) could yield more accurate estimates than (2.12a) when few simulations are performed.

2.3. REANALYSIS PROGRAM AND ITS INTERFACE WITH COMMERCIAL PROGRAMS

In this section, we present a procedure to interface PRRA with a Windows-based commercial program for discrete system simulation, such as Arena. First we explain how to model and simulate a discrete event system and how to generate and sample data. Then, we explain how to export the data so that a program performing mathematical calculations, such as Mathcad, can read them. Finally, we show how to program PRRA in order to estimate a system's performance for many different probability distributions of the input variables in Mathcad. PRRA can be easily implemented on other programs such as MATLAB or Mathematica.

2.3.1. Simulation with Arena

Arena is a computer simulation tool with a graphical user interface developed by Rockwell Automation. Arena allows users to choose from various modules that are presented in various templates ranging from basic logic pieces to complex items. Each module represents a combination of SIMAN code that has been pre-packaged to allow the user to drag and drop pieces of code into the model without having to work with the code itself. Arena generates reports at the end of simulation that the user can modify. After analyzing the results, one can improve the system using various optimization methodologies located in Arena library.

In Arena's template panel, the "ReadWrite" module in "advanced process" template enables users to read data from and write data to Excel, Text, and Access files. This function enables one to read sample values from a sampling PDF in the simulation model by performing a single Monte Carlo simulation. For example, in the case of a system consisting of one drill press and a single queue, we can export the interarrival, service and waiting times of each part, for all replications, by using "ReadWrite" module.

Exporting data from Arena involves the following steps: (1) Add a "ReadWrite" module into the model. (2) By double clicking the "ReadWrite" module in the model window, set the attributes of this module. First, name this module, and select type as "Write to file". Then assign an external file, in which the sample data will be written in the "Arena File" dialog box. (3) Define the assignments. First, select the type of assignments, and then build an expression to represent it. Arena provides an expression builder to help the user complete this task easily.

After setting the attributes of the "ReadWrite" module, we can run Arena to perform the single Monte Carlo simulation. The sample points will be written into the specified file and will be read by the program that will perform PRRA

2.3.2. Using Mathcad to Perform PRRA

We implemented the PRRA approach in Mathcad because mathematical formulas in this program can be entered just as they are printed in a book, paper, or report. This feature enables users without coding experiences to program PRRA equations easily.

Mathcad also has strong graphics capabilities, which plot range variables against each other or against an index, produces two, or three dimensional plots which may be embedded anywhere in the document. In PRRA, this feature enables a user to monitor how a variable changes, and adjust some parameters involved in the calculations to improve accuracy and computational efficiency. For example, by examining the plot of the correlation coefficient between the waiting times of two entities versus their distance, we select a suitable decorrelation distance δ , in order to estimate accurately the system's performance and keep the computational cost low.

All the above features enable the symbols, equations and figures of the main steps of PRRA can be programmed just as they are shown in Eqs. (2.1), (2.2), (2.3), (2.4a), (2.5), (2.6), (2.4b), (2.7), (2.8), (2.9), (2.10), (2.11), (2.12a), (2.13), (2.14), (2.15).

2.4. RESPONSE SURFACE DEVELOPMENT FOR APPROXIMATION OF THE PERFORMANCE ATTRIBUTES

In a discrete event system, engineers want to adjust the system parameters in order to optimize one or more of its performance attributes. For example, they may want to minimize the mean time of a part in the system or the mean waiting time in a queue. An approximation of the functional relation between the value of a performance attribute and the input variables enables engineers to determine efficiently the