Simulation Analysis Framework Based on TRIAD.NET

Grigorii Kolevatov
Department of Mechanics and Mathematics
Perm State University
Perm
kolevatov@prognoz

Elena Zamyatina
Department of Mechanics and Mathematics
Perm State University
Perm
e_zamyatina@mail.ru

Abstract — **the objective of using simulations is to produce true knowledge about complex dynamic systems. Due to increasing popularity of simulation, complexity of simulation models is increasing too. In recent years there were researches with simulations which consist of thousands of interacting objects. The process of analysis such models becomes too complicated and makes the whole modeling process more expensive. That influence tends to be very strong if, as usual, an iteration process is used. Moreover, simulation output analysis based on number of math statistics techniques which creates extra requirements to analysis team members and makes simulation too difficult to mass implementing. This paper is dedicated to attempt of the problem resolving.**

simulation, simulation output analysis, data mining, time series, linear regression

I. Introduction

Computer simulations have many applications in many areas, such as: logistic, manufacturing, medicine and service operations. Simulation is used when traditional Operations Research tools such as linear programming, stochastic modeling or queuing network models cannot capture the details or the dynamic of the system [1].

Although simulation is good for representing complex systems, analysis of simulation seems to be too complicated. As shown in paper [2] traditional simulation analysis looks like selection the best variant by some criteria from different scenarios. Setting the set of scenarios and selection criteria has a major influence on analysis results. Decision, which criteria is most appropriate and what scenarios need to be considered, is often made on basis of intuition and modeler experience. That form of analysis does not help researcher to understand why the selected scenario is the best and if the number of scenarios is enough or not.

On the other hand, in last three decades, a huge number of data analysis techniques called Data Mining have been developed. Application of those techniques to the problem of simulation output analysis could significantly decrease analysis complexity and improves real profit of simulation applications.

Many attempts to solve the problem were made in recent scientific researches. Paper [3] shows the benefits of using heuristic-based automated optimizations for finding better scenario with pre-set selection criteria, called objective functions. New approach to automate selection criteria was described in [4]. Offered approach helps to reduce the requirements to the modeler due to automating statistic's calculations. In [1] the way of reducing simulation output log by mining information about entire model variables relationships were demonstrated. Number of data mining techniques, which was applied in [1], estimate the correlation between model variables. The approach was tested on complex model with huge number of interacting elements. It earnestly shows how significantly data mining applications can decrease volume of output data without losing important information.

All of described here approaches have one main disadvantage – they were designed for particular specific tasks and have not got facilities to simplify the analysis process in general case. The decision of the problem could be a special designed simulation analysis framework which could provide an expert with set of instruments which would help an expert to answer every question about simulation model that an expert could have. It should be able to adjust dynamically to particular area specifications. This framework should combine information about analyzed system (extracted from model describing) and new data mining techniques. Also it should be able to adapt dynamically to an expert needs. This paper describes the attempt of creating such framework. Also it is necessary to mention, that developing the most common framework which suits to every particular case is a very difficult task. Thus, in the paper only attempt of creating such framework is considering. Framework creating activities were undertaken on base of discrete-event simulation environment TRIAD.NET. In section II formal problem definition is described. TRIAD.NET Special features' description is put into Section III. Suggested problem solution is described in Section IV. Section V is dedicated to particular framework architecture. At the end of the paper brief conclusion is made.

II. Problem Definition

A. User Dialog

At first, it is necessary to mention that all knowledge about the real system: its purpose and its place in the world is situated in expert's mind. Modeling environment has only information about model itself. That's why it is impossible to put all of research activities into environment. Only an expert can determine what information is important: if model reflects real system in proper way, if all necessary information is delivered

or not. So, the expert should deliver to the environment necessary information about research objectives. The second assumption is that the expert's knowledge grows dynamically since research was started. It means that an expert needs system to automatically adapt to expert's requirements. By that reasons it's logically to decide that the interaction between an expert and the environment should be organized in terms of dialog: expert asks questions about his filed of interest, environment makes an investigation and returns answers [5].

In traditional analysis all questions, that analyzer could ask an environment, in general, was: "Is *A* true?" where *A* is some statement about model values. It's not enough, because usually, expert develops simulation not only for checking some suggestion, but for changing suggestion to a more appropriate value. That's why it is necessary to add another type of question: "Is *A* determined by *B*?" It means that *A* is true when *B* is true, and when *B* is false – *A* is false too. There is one more is "Is *A* conditioned by *B*?" It means that *A* is true when *B* is true, and *A* is undefined when *B* is false. To answer this question an expert usually does many different experiments with different parameters. But model parameters' domain is known to environment, model is also known. That means, consequently, environment could answer these questions itself. In fact, even answering these questions can make analysis process simpler and clearer. But also those types of question could be combined into more general: "What determine *A*" and "What condition *A*", and replaced by one: "Why *A* is true". Thus, we have two general questions:

1) Is A true?

2) Why A is true?

B. Statements.

Simulation models consist of some objects of different types and connections between them. Model variables reflect states of these objects or groups of objects or model in general. Thus, every statement about a model could be impressed by first-order predicate language. Therefore, such statements could consist of:

- the quantifier symbols $\forall$ and $\exists$;
- the logical connectives;
- parentheses, brackets, and other punctuation symbols;
- an infinite set of variables;
- an equality symbol;
- first-order predicates;
- functions.

Before saying a word about sense of those symbols we should determine that there are two types of simulations analysis: *single experiment analysis* (SEA), *multiply experiment analysis* (MEA). In different situations the particular sense of alphabet symbols should be defined in different ways.

Let *M* is a model, and *X* is a set of model variables:

$$X = \{x_1, \ldots x_n\} \tag{1}$$

In SEA case, particular value of variable x_i depends of system time and could be written as:

$$x_i : T \rightarrow D(x_i), \tag{2}$$

where *T* is system time. Hence, in SEA situation we have only one variable – *t*, and variables $x_i(t)$ would be the function of *t*.

In MEA case, value of a variable depends on experiment where it is calculated and system time. Experiment itself is determined by model parameters value and initial values for random numbers generator (RNG). To not depend from probability factors, better determine x_i in such way:

$$x_i : T \cap E \rightarrow D(x_i), \tag{3}$$

where *E* is set of all possible experiments.

Also, it's usually necessary to use some integrated characteristic, such as average or divergence. Set of those variables could be called *Y*, and every y_i could be defined in that way:

$$y_i : E \rightarrow D(x_i), \tag{4}$$

We can easily translate our question "Is *B* determine *A*" into first-order predicate form as it shown on (5) and "Is *C* determine *A*" as it shown on (6).

$$B \Leftrightarrow A \tag{5}$$

$$C \Rightarrow A \tag{6}$$

Thus, answering question "What determine *A*", we should find such statement *B* that suit next condition:

$$B \Leftrightarrow A \vee (\forall S \mid S \Leftrightarrow A) : B \vee S = 1 \tag{7}$$

Analogically, question "What condition A" could be defined as:

$$C \Rightarrow A \vee (\forall S \mid S \Rightarrow A) : C \vee S = 1 \tag{8}$$

Thus, we can define our objective as:

1) calculate value of statement A;

2) find statemnts B and C which satisfy to (7) and (8).

III. TRIAD.NET FEATURES

Distributed simulation system Triad.Net includes following components: TriadCompile – compiler from Triad modeling language, TriadCore – simulation core, GUI, TriadDebugger – validating and debugging system, TriadBalance – distributed components synchronization system, TriadEditor – remote access system, TriadSecurity – external and internal security threats detection system, TriadBuilder – automated model redefining system and TriadMining – simulation output analysis system.

Simulation model in Triad defined as:

$$M = (STR, ROUT, MES), \tag{9}$$

where *STR* – structure layer, *ROUT* – routine layer, *MES* – message layer. Structure layer represents itself as objects'

aggregate, interacting to each other sending messages. Each object has input and output poles which servers as receivers and senders messages. Structure layer representation based on graphs. Separate objects play a role of graph nodes. Graph arches determine connections between objects.

Objects' behavior is determined by routine layer. Routine is a sequence of events which plans each other. When event occurs state of associate object changes. Routine layer is separated from structure layer, thus, routines could be reused when structure is defined, and different routines could be associated with different nodes in structure layer. Message layer is used for complex message defining.

One of the advantages of Triad, which play the main role in choosing Triad as a platform for analysis framework is its special feature: special objects, called *information procedures* and *simulation conditions*. Information procedures are objects that collect information about model variables changing. When change of a model variable occur information procedure is executed and data is saved in modelling output. Specific content of data depends on an algorithm of specific information procedure. Triad has facilities to code every formal algorithm as an information procedure. Simulation conditions determine a situation when simulation ends. They also have special algorithmic faculties that help to set complicated conditions.

Simulation conditions and information procedures are separated from model definition. Hence, information procedures could be changed without model changing and the same simulation conditions could be used for different models. Algorithmic facilities of information procedures and simulation conditions and their separate (from model definition) character has not got analogues in other modelling environment and languages such as GPSS, ProModel, Witness, AnyLogic and etc. This is the main reason, why analysis framework should be developed on base of Tried.

IV. Problem Solution

As described in Section III problem consists of two parts:

1) Calculate statement value;

2) Find determining and conditioning statements.

Further, an approach of solution each part of the problem is considered.

A. Statement Value Calculation

When SEA case is considered, there are no difficulties in calculating statement value. After an experiment took place it's easy to calculate actual value of statement, based on values collected through the experiment. All model variables depend on system time, since experiment is finite then system time is finite, and we got a finite number of variables' values.

Situation becomes difficult when MEA case took place. Due to probabilistic nature of experiment, E is infinite. It makes it impossible to calculate the sentence using all of possible data. In [4] it's earnestly shown that it's simple to determine the necessary value of replications by specifying the significance level and deviation of confident interval which is suitable for situation. Back to Section II, modeling environment does not have enough knowledge about what significance level and deviation is enough to be confident in results. Hence, these parameters should be user-defined.

The second problem connected with parameters' domain. Since it can be infinite or too large it becomes too difficult to calculate all possible experiments. Problem could be solved by setting number of intervals, which could divide the domain into finite number of values.

Hence we can calculate any value of any variable, function or predicate both in SEA and MEA cases. Consequently we can calculate the actual meaning of statement.

B. Determinating and Conditioning Statement Search

The corner stone of DE terminating and conditioning statement search is to find a statement which satisfy criteria, defined in (7) or (8). These criteria could be divided into two separated criteria: dependence criteria and completeness criteria. Dependence criteria for (7) described in (5) and dependence criteria for (8) described in (6). Completeness criterion for (7) is shown in (10), and completeness criterion for (8) is shown in (11).

$$(\forall S \mid S \Leftrightarrow A) : B \vee S = 1 \quad (10)$$

$$(\forall S \mid S \Rightarrow A) : C \vee S = 1 \quad (11)$$

Total search algorithm for (7) could be described in follows way:

1) Let B be false.

2) Find new S, which suits for (5).

3) If S was found then continue, else exit.

4) Check (10) criteria.

5) If (10) criteria is false then let B = BvC, simplify B.

6) Return to step 2.

The only undefined step in these criteria is step 2. It could be divided into two steps:

1) Make new suggestion.

2) If it is impossible to make new suggestion then exit.

3) Check criteria (5).

4) Return to step 1.

Step 1 is still undefined. The way we can define it is to make father suggestion: "Variables from statement A and from statement B should have dependence". Dependence estimation is widely spread technique used in data mining problems. Different types of correlation could be estimated by special measures which are described in next section. In common way dependence estimator could be defined as it shown in (12).

$$de : X \times X \rightarrow R \quad (13)$$

Thus, dependence estimator *de* is reflecting all possible pairs of model variables into real numbers. Thus, using dependence measure of some kind, it's possible to estimate relationships between variables and choose set of variables that could be used for creating possible B-statement.

C. Dependance estimation

The objective of using dependence estimator is to estimate function dependence between variable. Consider the offered approach.

Let M – certain model. Model has parameters $P = (P_1, ..., P_n)$. Each parameter is determined on domain D_{P_i} and domains organize space D_P. Model also has set of variables $X = (X_1, ..., X_n)$ each variable determined on D_{X_i} what construct space D_X. Let's declare $X_i^{'}$ which satisfy the follows:

$$X_i \notin X_i^{'}, X_j \in X_i^{'}, \forall j \neq i \tag{14}$$

The result of experiment e would be matrix:

$$X^e = (X_1^e, ..., X_m^e), \tag{15}$$

$$X_i^e = (x_{1,i}^e, ..., x^e{}_{T,i}), \tag{16}$$

We can determine now a dependence estimation problem in such way: for the results of experiment e on model M find functions F_i which satisfy (17):

$$F_i^e : D_{A_i^{'}} \to D_{A_i}, F_i(A^{e'}) \in \left[A_i^e - \partial, A_i^e + \partial\right], \partial \in D(A_j) \tag{17}$$

Hence, dependence estimator should have a view:

$$F' = (F_1', ..., F_m') \tag{18}$$

V. TRIAD.MINING ARCHITECTURE

In general, Triad.Mining algorithm is follows:

1) Get an expert request.

2) Translate request into set of information procedures and modeling conditions.

3) Simulate with conditions and information procedures from step 2.

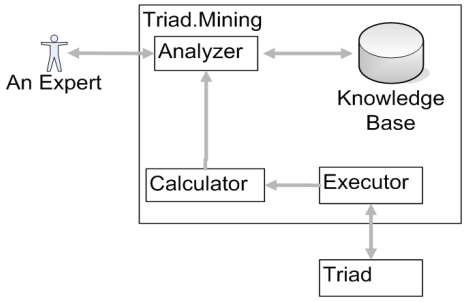
4) Process the result:

a) Calculate statement if request type is 1;

b) Start the algorithm from section IV B.

To solve the problem from section II TRIAD.Mining uses four components:

1) Analyzer
2) Executor
3) Calculator
4) Knowledge Base



Picture 1. The Architecture of Triad.Mining

Relationships between components are shown on picture 1. Analyzer gets the request from an expert in a first-order predicate form. Then it translates the request into particular modeling conditions and information procedures and sends it to the executor. Executor does simulation tasks and collects the results. Results go to Calculator. Calculator interprets the results and calculates statement value or checks the criteria. The result of the calculations goes to analyzer, which formulate the final answer to the expert.

VI. CONCLUSION

Triad.Mining shows a better result in understanding the real message of simulation output and improves efficiency of simulation research significantly in comparison with traditional simulation analysis tools. It based on common approach and does not depend on specific research area, such as logistic. It also forms up communications with and expert with and natural question-reply way, and does not demand an expert to have a deep knowledge in mathematical statistics. All of these arguments say that Triad.Mining could be used for improving efficiency of simulation research process.

[1] T. Brady, E. Yellig, Simulation Data Mining: a new form of simulation output, 37th Winter Simulation Conference, Orlando, USA, 2005, pp 285-289.

[2] Akbay, S. Kunter, Using simulation optimization to find the best solution. Industrial Engineering Solutions 28, San Fernando, Argentine, 1996, pp 24-29.

[3] Brady, Thomas F. and Bowden, Royce A. The effectiveness of generic optimization routines in computer simulation languages. In Proceedings of the 10th Industrial Engineering Research Conference, [CD-ROM], Dallas, USA, 2001.

[4] Robinson S., Automated Analysis of Simulation Output Data, proceeding of the 37th Winter Simulation Conference, Orlando, USA, 2005, pp 763-770.

[5] G. Neumann, J. Tolujew, From Tracefile Analysis to Understanding the Message of Simulation Results, proceeding of the 7th EUROSIM Congress on Modeling and Simulation, Prague, Czechia, 2010.