Evaluation and Improvement of Parallel Discrete Event Simulation Performance Predictions: A Rough-Set-based Approach

László Muka, István Derka

Department of Telecommunications, Széchenyi István University
Egyetem tér 1, H-9026 Győr, Hungary; muka@sze.hu, steve@sze.hu

Abstract: Simulation performance prediction methods make possible the realization of performance improvement potentials of Parallel Discrete Event Simulation (PDES) methods, important in the analysis of complex systems and large-scale networks. Currently, high performance execution environments (emerging clusters and computing clouds) advance the development of quality/cost analysis capabilities of performance prediction methods. In this paper, for the evaluation and management of prediction correctness/cost, the efficacy, efficiency and effectiveness coefficients and improvement operations are defined for predictions. The performance coefficients and improvement operations are embedded in the rough-set-modeling and learning process and presented as an enhancement approach of the conventional Coupling Factor Method (CFM). A case study based on the CFM analysis of PDES of a closed queuing network model is presented. In the example, after rough-modeling and train-and-test analysis, the correctness/cost evaluation and effectiveness improvement operations are shown for series of predictions and the feedback connection to modeling refinement phase is demonstrated too.

Keywords: Parallel Discrete Event Simulation; simulation performance prediction; quality/cost analysis; Systems Performance Criteria; Rough Set Theory; Coupling Factor Method

1 Introduction

Over the last few years, various research efforts have been made regarding Parallel Discrete Event Simulation (PDES) modeling and execution methods [1, 2, 3, 4], since parallel execution turned out to be an appropriate approach to meet high computing capacity requirements of Discrete Event Simulation (DES) analysis of complex systems and large-scale networks [5, 6].

In the present paper, PDES is defined as the execution of a single DES model on some high performance computing platform which can be clusters of homogeneous or heterogeneous computers and other emerging execution environments (cloud, grid, etc.) too.

Development of simulation models having high runtime performance features in a PDES execution environment is not an easy task even today, mainly because it is hard to tell the parallel execution features of a model, for example, possible resource capacity bottlenecks, in the development phase [7]. The *simulation performance prediction* of PDES execution can help to realize higher performance by providing preliminary knowledge about the behavior of the simulation model in the parallel execution environment [7, 8, 9, 10, 11]. The PDES performance prediction methods can support the performance increase throughout the whole modeling and simulation process including model development support, simulation setup and evaluation phases and taking into account different simulation framework specifics too [7].

Nowadays, emerging simulation operation forms, like on-line, real-time modeling and simulation and especially cloud computing with on-demand network access and pay-per-use feature and the developing simulation as a service (Simulation Software-as-a-Service) in public clouds [3, 21] put an increasing emphasis on quality/cost analysis capability of performance prediction methods.

The *motivation* of the authors to make the research presented in the paper was the lack of methods which can manage together the correctness and cost of performance prediction for a model in parallel simulation execution environment and which can also provide an easy feedback to the modeling and setup phases in the simulation.

The authors have developed the *Enhanced Simulation Performance Prediction Method (ESPPM)*. ESPPM is enhancing the standard *Coupling Factor Method (CFM)* of PDES performance prediction [14] by a method of improvement allowing handle together correctness and cost in speedup predictions. The improvement approach based on *Rough Set Theory (RST)* [13] train-and-test analysis with embedded *Systems Performance Criteria (SPC)* [30] of *efficacy (E1), efficiency (E2)* and *effectiveness (E3)* for complex evaluation and quality/cost performance improvements steps.

In this work, the authors make the following key *contributions*:

- Definitions of performance improvement operations, based on definitions of cost measures and on definitions of performance coefficients of *efficacy (E1), efficiency (E2)* and *effectiveness (E3)* are described for single predictions and for series of predictions.
- A case study of the work of ESPPM is introduced using the example of a CFM PDES performance prediction of a Closed Queuing Network (CQN) model. In the analysis, the RST model of CFM modeling and its train-and-test examination is presented, the E1, E2 and E3 rule and attribute dropping operations are shown and the discussion of analysis results of performance predictions with feedback to the CFM and RST model-identification and refinement phase.

The rest of paper is organized as follows: Section 2 summarizes the related work, estimates the simulation performance prediction methods and relevant RST applications. Section 3 describes the components of the new method (RST, SPC and CFM). Section 4 introduces prediction performance improvement operations based on cost and prediction performance coefficients SPC-type definitions which are embedded into the process of train-and-test RST analysis. In Section 4, the process of ESPPM algorithm is described too. Section 5 introduces an example case study to demonstrate the use of the new method for the prediction analysis. Section 6 discusses the results of analysis. Finally, the conclusions are presented.

2 Related Work

The related research is overviewed in following two points.

2.1 Simulation Performance Prediction Methods

The method introduced in [8] uses execution *event-trace data* of sequential and parallel, simulation runs, for creating the *execution graph model* and for the subsequent *critical path analysis* to predict parallel simulation performance. The method takes into account, in the prediction, the characteristics of the simulation hardware (with hardware parameters and mapping algorithms). The use of a wide variety of conservative and optimistic parallel simulation synchronization protocols is allowed by the method, which is an advantage. Unfortunately, the evaluation of prediction quality and the common evaluation with the cost of prediction are not treated in the method.

As an alternative to the conservative synchronization, the time driven version of *the statistical synchronization method (SSM-T)* with its loose synchronization [18, 19, 20] is a less well-known, but promising PDES synchronization method, that can be applied for the parallel simulation of certain types of systems such as communication networks. The increased performance can be predicted based on the *trace of frequency of statistics exchange between segments*. The frequency of statistics exchange can be used similarly to *lookahead* [15]. There is no performance model used in the method, thus, performance prediction requires time-consuming analysis of the simulation model operation.

The method described in [7], uses a *hybrid approach* to define the theoretical limit of the execution improvement for conservative synchronization protocol. In the *trace-based part*, the method analyses only a definite part of the sequence of events of the simulation model; while in an *analytical modeling part*, for the calculation of the lower bound on the runtime of the parallel simulation, a simplified scheduling problem model and the linear programming approach of is applied. An important advantage of this method is the definition of the performance improvement limit for a model (or model version) but the method does not pay attention to the quality/cost analysis of predictions.

Paper [14] introduces the CFM (details are described in point 3.3), a simulation performance prediction approach, based on the *coupling factor PDES performance model*. The coupling factor helps to predict the *parallelization potential* of the simulation models and can be defined in sequential simulation model runs. The method is appropriate only for conservative null message-based algorithm. Unfortunately, CFM introduced in [14] does not predict the speedup value and does not support the quality/cost evaluation in predictions either.

2.2 Rough Set-based Performance Prediction

Paper [22] presents a rough set modeling and application runtime prediction method. The method is based on identifying and extracting properties defining runtime similarity of applications using available past data. The method does not include any consideration on prediction cost and on the improvement of prediction quality.

Paper [23] introduces a scheduling optimization approach for a dynamic remanufacturing situation with uncertain data. The proposed method is using linear programming and rough set evaluation and learning in an iterative process. Monte Carlo simulation is also involved to improve consistency of data. Unfortunately, complex evaluation, allowing quality/cost analysis in the learning process, is not included in the method.

3 Method Components

3.1 Rough Set Elements and the Rough Prediction Algorithm

The RST (Rough Set Theory) is a mathematical framework suitable for modeling and analysis of information systems with imprecise relations, with uncertain data [25, 26, 27, 28, 29].

A *rough set information system* with embedded knowledge consists of two sets: the set of *objects* called the *universe* and the set of *attributes*.

More formally, $I = (U, A, f, V)$ denotes an *information system* of RST, where set U is the universe, A is the set of attributes. Sets U and A are finite nonempty sets where ($U = \{x_1, x_2, \dots, x_{|U|}\}$ and $A = \{a_1, a_2, \dots, a_{|A|}\}$). The attributes define an *information function* $f: U \to V$ for U where the set V is the set of values of A ($V = V_{a_1} \cup V_{a_2} \cup \dots \cup V_{a_{|A|}}$). The set V_{a_i}– named also the *domain* of a_i – contains the collection of values of a_i and $V_{a_i} = \left\{v_{1_{a_i}}, v_{2_{a_i}}, \dots, v_{\left|V_{a_i}\right|_{a_i}}\right\}$ where $\left|V_{a_i}\right|$ is the size of the domain of a_i.

Discretization is the operation of mapping the primary values and ranges of all attributes to *selected* (possibly optimized) sets of discrete values: $f_{V'}: V' \to V$. The set V' stands for the values of a before discretization.

The B*-indiscernibility relation* $IND(B)$for a set of attributes $B \subseteq A$is defined in the following way:

$IND(B) = \left\{(x_i, x_j) \in U^2 \middle| \forall(a \in B)(\, a(x_i) = a(x_j))\right\}$.

If $(x_i, x_j) \in IND(B),$ then the objects x_i and x_j are indiscernible from each other in B and the *equivalence classes* $[x]_{IND(B)}$ of $IND(B)$ are formed by the objects indiscernible in B.

Rough sets are defined by their lower approximation and upper approximation sets. The set $B^*(X)$ and the set $B_*(X)$ is the B*-lower* and B*-upper approximation* of the set X and defined as follows:

$B_*(X) = \bigcup_{x \in U}\left\{x \middle| [x]_{IND(B)} \subseteq X\right\}$ and

$B^*(X) = \bigcup_{x \in U}\left\{x \middle| [x]_{IND(B)} \cap X \neq \emptyset\right\}$.

The set $BN_B(X)$ defined by the equation $BN_B(X) = B^*(X) \setminus B_*(X)$ is the B*-boundary region* of X. If X is a crisp set then, $X = B_*(X)$ thus $BN_B(X) = \emptyset$ which means the boundary region is empty.

A *reduct* R_B is the minimal subset of attributes B that allows the same classification of objects of U as the set of attributes B. This feature of a reduct may be described by indiscernibility function as follows:

$IND_{\forall x(x \in U)}(R_B) = IND_{\forall x(x \in U)}(B),\, B \subseteq A$.

In general, the information system may take the form of $I = (U, A = C \cup D, f, V)$ which is a *decision information system (DIS)*. The set $C = \left\{c_1, c_2, \dots, c_{|C|}\right\}$ denotes the set of condition attributes and D is the set of decision attributes $D = \left\{d_1, d_2, \dots, d_{|D|}\right\}$. The information function $f: U \to V$ may be expressed by information functions $f_C: C \to V_C$ and $f_D: D \to V_D$, where $V = V_C \cup V_D$ ($V_C = V_{c_1} \cup V_{c_2} \cup V_{c_3} \cup, \dots, \cup V_{c_{|C|}}$ and $V_D = V_{d_1} \cup V_{d_2} \cup \dots \cup V_{d_{|D|}}$) and

$V_C = \bigcup_{i=1}^{|C|} V_{c_i}$ where V_{c_i}=$\left\{v_{1_{c_i}}, v_{2_{c_i}}, \dots, v_{\left|V_{c_i}\right|_{c_i}}\right\}$ and $V_D = \bigcup_{i=1}^{|D|} V_{d_i}$. where V_{d_i}=$\left\{v_{1_{d_i}}, v_{2_{d_i}}, \dots, v_{\left|V_{d_i}\right|_{d_i}}\right\}$.

In a *decision table*$I = (U, A = C \cup \{d\}, f, V)$ based on a DIS, d denotes the *distinguished decision attribute*. Furthermore, a decision information system having the form of $I = (U, C \cup D, f_{V'}, f, V', V)$ denotes a DIS with *discretization information functions* $f_{V_C'}: V_C' \to V_C$and $f_{V_D'}: V_D' \to V_D$.

The *classification* may also be described by a *set decision rules* $S = \{s_1, s_2, \dots, s_{|S|}\}$ in the form of implication $s_k = (\varphi_k \Rightarrow \kappa_k), (s_k \in S)$, where φ_k and κ_k are *logical expressions* of the condition and decision attributes respectively. The formulas φ_k and κ_k may also be quoted as *LHS (Left Hand Side)* and *RHS (Right Hand Side)* part of the rule. A decision rule s_k may be evaluated by its $Match_U(s_k)$ and $Supp_U(s_k)$ values, where $Match_U(s_k)$ is the number of objects in U the attribute values of which satisfy φ_k (*matching* with the *LHS* part of s_k), and $Supp_U(s_k)$ denotes the number of objects in decision table the attribute values of which satisfy both φ_k and κ_k (matching with both the *LHS* and *RHS* parts of s_k).

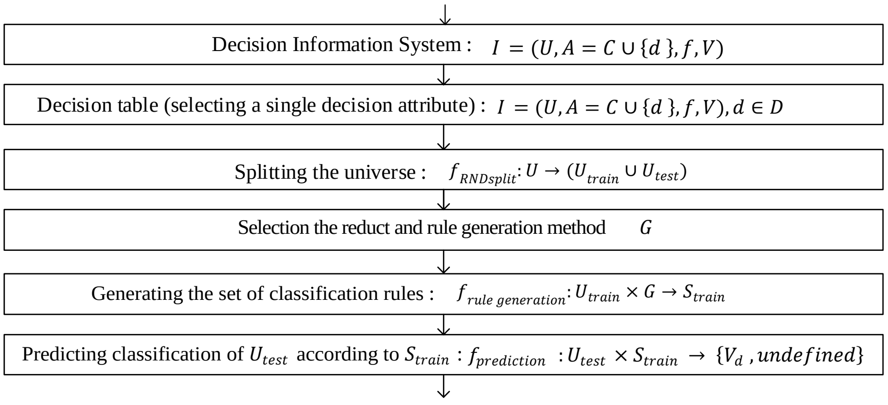
Decision Information System : $I = (U, A = C \cup \{d\}, f, V)$
Decision table (selecting a single decision attribute) : $I = (U, A = C \cup \{d\}, f, V), d \in D$
Splitting the universe : $f_{RNDsplit}: U \to (U_{train} \cup U_{test})$
Selection the reduct and rule generation method G
Generating the set of classification rules : $f_{rule\ generation}: U_{train} \times G \to S_{train}$
Predicting classification of U_{test} according to S_{train} : $f_{prediction}$: $U_{test} \times S_{train} \to \{V_d, undefined\}$

Figure 1

The Traditional Rough Set Theory (TRST) analysis algorithm for performance prediction

Figure 1 shows the process of the rough prediction algorithm. The essence of the algorithm is the random split of the universe into train and test partitions, generation a set of classification rules using the train partition and then generation classification predictions for the test partition using the generated rules.

3.2 Systems Performance Criteria

Efficacy (E1), *efficiency (E2)* and *effectiveness (E3)* are *Systems Performance Criteria (SPC)* [11, 30, 31]. E1, E2 and E3 coefficients are in a hierarchy-like relationship with each other. On the longer term, the performance of a system is checked by the effectiveness criterion, the efficacy criterion shows whether the performance is suitable at all, and the efficiency criterion characterizes the relation of the required output and the resources used to produce the output.

3.3 The Coupling Factor Method

Based on some theoretical considerations about the connectedness of PDES model segments, paper [14] describes a practical simulation performance prediction approach the *Coupling Factor Method (CFM)*. The method – using results of sequential simulation runs – predicts the *parallelization potential* of simulation models with conservative null message-based synchronization algorithm. The CFM performance model can be described by the formula:

$\lambda = L * E/\tau * P$

where L is the lookahead value characterizing the simulation model $[simsec]$ [6], E is the event density generated by the model $[event/simsec]$, τ is the latency of messages between logical processes (LPs) of the simulation model during the execution $[sec]$, and P is the event processing hardware performance $[event/sec]$. In this performance model, parameters L and Echaracterize the simulation model itself, parameters τ and P describe the execution environment. The performance model involves only four parameters for the performance prediction calculations that can be measured in simple *sequential simulation runs*. According to the method, the PDES speedup value cannot be predicted, but the high value of λ (λvalue is a couple of time 100 or higher [15]) shows the good potential for simulation model parallelization. For a separate process, the λ_N parallelization potential of a process is only a part of the whole potential:

$$\lambda_N = \lambda/N_{LP}$$

where N_{LP} the number of LPs [15]. (The four parameters of CFM formulates requirement on the simulation model and on how the parallelization potential can be exploited.)

The method has been validated by a series of simulation investigations for *homogeneous* and *heterogeneous* clusters of computers [15, 16, 17]. Example applications for telecommunication systems and for cloud computing systems have been introduced too in [16] and [32].

4 The Prediction Performance Enhancement

4.1 Defining Prediction Performance Coefficients

In a TRST simulation performance analysis, in a decision table $I = (U, A = C \cup \{d\}, f, V)$ the o*bjects* of the universe U are *computer simulation experiments.* Set of attributes (A) consists of independent (C) and dependent $\{d\}$ variables that are taken into account in performance evaluation and in classification prediction.

The classification of experiments in $U_{test(i)}$ ($U = U_{train(i)} \cup U_{test(i)}$) is *predicted* by $f_{prediction(i)}: U_{test(i)} \times S_{train(i)} \rightarrow \{V_d, undefined\}, (i = 1,2, \dots, \text{n})$, using the set of rules $S_{train(i)} = \left\{s_1, s_2, \dots, s_{|S_{train(i)}|}\right\}, s_k = \varphi_k \Rightarrow \kappa_k$ that are generated according to function $f_{train}: U_{test} \times G \rightarrow S_{train}$ in TRST.

In the following the *E1*, *E2* and *E3* SPC will be defined in the form of *prediction performance coefficients*: efficacy (E1) to measure prediction correctness, efficiency (E2) to assess correctness-to-cost relationship and effectiveness (E3) to take into account E1 and E2 for series of predictions. To calculate cost relationships *cost of experiments, attributes, rules and predictions* will be defined using the consumed computation time.

For the improvement of prediction performance, attribute and rule dropping operations are defined.

Definition 1. Efficacy (E1) of a prediction

$E1_{prediction}$ is calculated according to formula

$$E1_{prediction} = \frac{\sum_{l=1}^{|U_{test}|} p(x_l)}{|U_{test}|}$$

where $p(x_l)$ (*prediction correctness*) is equal to

$p(x_l) = \begin{cases} 1, & |\langle d(x_l)_{predicted}\rangle = \langle d(x_l)_{observed}\rangle \\ & 0|otherwise \end{cases}$ and where$d(x_l)_{predicted}$ is

$d(x_l)_{predicted} = \begin{cases} v_d, \langle \kappa_k \rangle = v_d, v_d \in V_d \big| (Match_{U_{test}}(s_k) = 1 \\ \quad undefined\ |otherwise \end{cases}$ and

$(s_k \in S_{train})$, $\langle d(s_k)_{predicted}\rangle = v_{d(x_l, s_k)}$.

The interval of $E1$ coefficient is $0 \leq E1 \leq 1$. If $E1_{prediction} \geq E1_{limit}$ is true then the prediction is *efficacious*. The $E1_{limit}$ denotes the lower limit of efficacy and the inequality $E1_{limit} > 0.5$ should be satisfied that is the efficacy of prediction is required to be better than random guess.

Definition 2. Cost of experiments, attributes, rules and predictions

For a decision table $I = (U, A = C \cup \{d\}, f, V)$ supposing that a homogeneous cluster of computers with equal cores is examined and supposing that all the cores are continuously working during the execution time, the *cost of a simulation experiment* $x_l (x_l \in U)$ is defined as: $K(x_l) = N_{cores} * execution\ time\ [sec]$.

Cost of a rule s_k is calculated as $K(s_k) = \sum_{l=1}^{|U|} K(x_l | Supp(s_k)) = 1)\ [sec]$.

Cost of a prediction $K(S_{prediction})$ is defined as $K(S_{train}) = \sum_{k=1}^{|S_{train}|} K(s_k)\ [sec]$, $(s_k \in S_{train})$.

Definition 3. Efficiency (E2) of a prediction

$E2_{prediction} = E1_{prediction}/K_{prediction}\ [\frac{1}{sec}]$

Definition 4. Effectiveness (E3) of a series of predictions

$E3_{series} = E3\left(S_{prediction}\right) = \overline{E2}_n \middle| \overline{E1}_n$ where

$\overline{E1}_n = \frac{1}{n}\sum_{i=1}^{n} E1_{(i)}$ and $\overline{E2}_n = \frac{1}{n}\sum_{i=1}^{n} E1_{(i)} / \frac{1}{n}\sum_{i=1}^{n} K(\,S_{train(i)})\ [\frac{1}{sec}]$

and where i denotes the i-th prediction and n is the number of predictions in the series($n \geq 2$). For $n = 1$ $E3_{series}$ is *undefined* and for $\overline{E1}_n < E_{limit}$ $E3_{series}$ is *not effective*.

Definition 5. Rule and attribute dropping

Dropping of a rule s_k and dropping of an attribute c_j for the pair (S_{train}, U_{test}) are the replacement operations $S_{train} := S_{train} \setminus \{s_k\}$ and $C := C \setminus \{c_j\}$ respectively.

4.2 The Enhanced Simulation Performance Prediction Method

Figure 2 shows the process diagram of the *Enhanced Simulation Performance Prediction Method (ESPPM).*

The ESPPM process can be described as follows:

- The ESPPM input data are produced both in sequential (CFM parameter measurements) and parallel simulation model runs (PDES, CFM parameter measurements, runtime measurements for cost calculations are executed and other relevant (or possibly relevant) simulation modeling hardware and software environmental data are collected.
- The rough model is made for performance prediction (objects and attributes of DISs before and after discretization).
- The TRST train-and-test method is used in interactive manner for generation of predictions
- The E1, E2 and E3 coefficients (built in the TRST cycle) are used for evaluation and attribute and rule dropping operations are applied for improvement of correctness and cost of predictions.
- The method supports setting up feedback to identification and refinement phase of the simulation performance prediction models.

The method can be implemented by using the OMNet++ [12] and the Rosetta System [13] free software for DES modeling and RST examinations respectively.

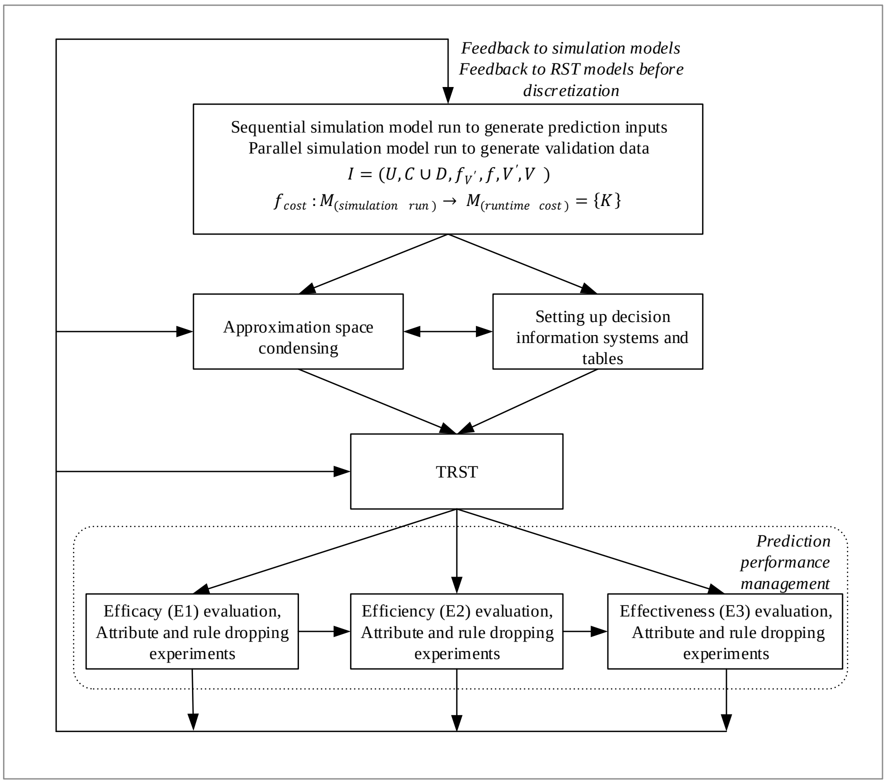
Feedback to simulation models
Feedback to RST models before discretization
Sequential simulation model run to generate prediction inputs
Parallel simulation model run to generate validation data
$I = (U, C \cup D, f_{V'}, f, V', V)$
$f_{cost} : M_{(simulation\ \ run)} \rightarrow M_{(runtime\ \ cost)} = \{K\}$
Approximation space condensing
Setting up decision information systems and tables
TRST
Prediction performance management
Efficacy (E1) evaluation, Attribute and rule dropping experiments
Efficiency (E2) evaluation, Attribute and rule dropping experiments
Effectiveness (E3) evaluation, Attribute and rule dropping experiments

Figure 2
The process of the Enhanced Simulation Performance Prediction Method (ESPPM)

5 Prediction Performance Analysis Case Study

5.1 Simulation Performance Evaluation Example

The example analysis of the CFM approach described in [15] investigates the simulation runs of a CQN model with various configurations in a homogeneous cluster execution environment with different number of processors.

In Figure 3, the number of *tandem queues* in the CQN model is 24 ($Q_{1,\ldots,24}$), the number of *simple queues* in a tandem queue is 50 ($q_{1,\ldots,50}$). The switching between tandem queues is performed by *switches* ($sw_{1,\ldots,24}$) according to *uniform probability distribution*. The delay of switching between tandem queues (shown by 2D arrow shapes, in Figure 3) will model the *lookahead* (L). The *lookahead delay* is the delay of *a job* before entering the next tandem queue and it is defined by the L value.

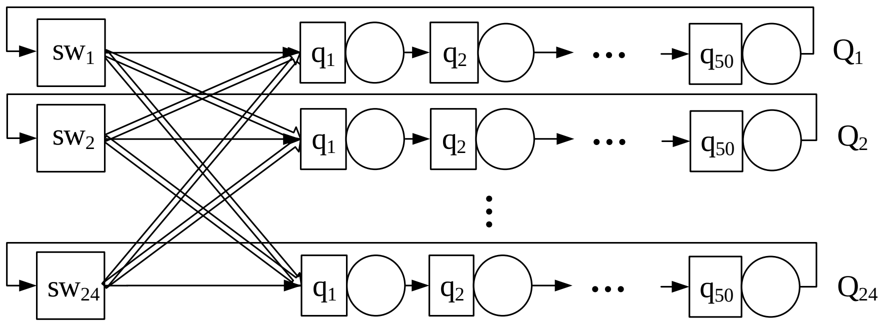

Figure 3

CQN model for PDES performance simulation

The starting number of jobs in every simple queue is 2 jobs, thus the *number of jobs in CQN* is 2400 and this value remains constant. Jobs have exponential *inter-arrival time* and exponential *service time* distributions with FCFS service discipline. The expected value for both the arrival and service time distributions is $10simsec$. The *delay* on links between *simple queues* is $1simsec$.

The *software environment* of simulation runs includes: *Linux (Debian) operating system, MPI, NFS, OMNet++ network simulator*. The *hardware environment* of simulation runs is a *homogeneous cluster* of 12 two-core host PCs. The communication latency of messages – measured by the OpenMPI PingPong benchmark and used as τ value – over MPI between the host PCs of the cluster is $25\mu sec$.

To calculate coupling factor λ the values of$E$ and P variableswere measured in *sequential simulation runs* on *one core* of a host PC (N_{cores}=1) with L =0.1, 1, 10, 100 and 1000 $simsec$ of lookahead values. The result of *sequential simulation* runs is summarized in Table 1 [15].

Table1

Coupling factor determination by sequential simulations

L [simsec]	0.1	1	10	100	1000
Execution time [sec]	524.18	521.36	523.09	516.54	415.73
P [ev/sec]	263502	264868	263465	261132	247653
E [ev/simsec]	159.86	159.83	159.51	156.12	119.16
Coupling factor λ	2.43	24.14	242.17	2391.39	19246.76

The *PDES experiments* were executed using equal number of cores and LPs: N_{LP} = 2, 4, 6, 8, 12 and 24 ($N_{cores} = N_{LP}$). The *relative speedup* results measured in *PDES executions* are summarized in Table 2.

The *simulated virtual time* both for sequential and PDES runs was $864000simsec$. The total number of execution runs for sequential simulation and for PDES were 55 and 330 respectively (11 runs with the same setup of an experiment).

Table 2
PDES relative speedup performance at different $\mathrm{N_{cores}}$ and L values

L [simsec]	0.1	1	10	100	1000
$\mathrm{N_{cores}}$=2	0.43	0.78	0.85	0.99	0.99
$\mathrm{N_{cores}}$=4	0.06	0.48	0.83	0.92	0.94
$\mathrm{N_{cores}}$=6	0.03	0.28	0.76	0.91	0.94
$\mathrm{N_{cores}}$=8	0.02	0.18	0.66	0.89	0.95
$\mathrm{N_{cores}}$=12	0.01	0.09	0.48	0.86	0.94
$\mathrm{N_{cores}}$=24	0.00	0.02	0.18	0.63	0.87

The *speedup* is defined as a proportion of the sequential and the PDES execution time of the simulation. The *relative speedup* is calculated as the proportion between the speedup values and the *number cores* used to achieve the speedup.

5.2 RST Modeling and Analysis

The *RST model of simulation performance prediction* is presented in the form of *decision information systems* and *decision tables:*

$I = (U, C \cup D, f_{V'}, f, V', V), I = (U, A = C \cup D, f, V),$

$I = (U, A = C \cup \{d\}, f, V,),\ d \in D.$

An extended but relevant list of elements of U, C and D sets are described below.

The *simulation experiments* are the *objects* of the universe $U = \{x_1, x_2, x_3, \dots, x_{30}\}$ where x_i denotes the object of the universe with index i.

An *extended set of condition attributes* may be defined as follows

$C = \{\ P, E, L(c_{04}), \lambda(c_{01}), \lambda_N, N_{LP}, N_{cores}(c_{03}), \tau, simulated\ virtual\ time,\ \} \cup$

$\{conservative\ synchronisation\ protocol\ with\ null\ message\ algorithm\} \ \cup$

$\{\ Q, q, number\ of\ jobs\ in\ CQN, inter\ arrival\ time,$

$service\ time, FCFS\ service\ discipline, switching\ time\ between\ tandems,$

$propagation\ delay\ between\ simple\ queues\ \ \} \cup$

$\{\ OMNet++, MPI, Linux\ Debian\ \}.$

The set of *decision* attributes under consideration can be set as

$D = \{relative\ speedup(d_{01}), speedup(c_{02}, d_{02}), utilization\ of\ resources\}.$
(The variables in brackets (for example, c_{04}) shows the variables selected for the analysis.)

5.2.1 Generating Predictions and Costs

For the prediction performance analysis the following decision table after discretization and appropriate coding is used:

$I_{d_{01}} = (U, A = C \cup \{d_{01}\}, f, V)$, $U = \{x_1, x_2, \dots, x_{30}\}$, $C = \{c_{01}, c_{02}, c_{03}, c_{04}\}$,

$V_{c_{01}} = \{l, m, h, s, e\}$, $V_{c_{02}} = \{a, n\}$, $V_{c_{03}} = \{2,4,6,8,12,24\}$,

$V_{c_{04}} = \{f(v'_{c_{04}} = 0.1), g, h, o, t(v'_{c_{04}} = 1000)\}$, $V_{d_{01}} = \{L(0 \le v'_{d_{01}} < 0.6), H\}$.

(For the later analysis, some values are shown before coding too (for example, $v'_{c_{04}} = 0.1$))

The examination is performed in a form of *TRST train-and-test* analysis. For the *train-and-test* examination, based on our previous results presented in [24], the *Rosetta System* [13] is used with *G=Johnson's RSES (Rough Set Exploration System)* reduct and rule generation method and with the subsequent classification of objects. The algorithm of examination is as follows:

Input: $I_{d_{01}}$, simulation runtime and configuration data, TRST configuration setup data: $\left(U = U_{train(i)} \cup U_{test(i)}\right) \wedge ((U_{train(i)} \cap U_{test(i)}) = \emptyset) \wedge \left(|U_{train(i)}|/|U| = = \frac{15}{30}\right) \wedge (RND_{seed} = i) \wedge (G = Johnson's\ RSES)$

Output: approximation of classification prediction data, and SPC evaluation

```
begin
        for i=1 to 8 for each prediction case do
        //prediction cases = 0, 3, 4,
                compute TRST
                //for the series of computation use the same $(U_{train(i)}, U_{test(i)})$
                //pairs $(i = 1,2,\dots,8)$
        return  classification prediction rules and classification prediction data,
                E1, E2 and E3 evaluation data
end
```

Table 3 shows the *costs of simulation experiments* $K(x_i)$ ($x_i \in U, i = 1,2, \dots ,30$).

Table 3
Cost of PDES simulation experiments $K(x_i)$ [ks]

N_{cores}	L=0.1simsec		L=1simsec		L=10simsec		L=100simsec		L=1000simsec	
	x_i	$K(x_i)$	x_i	$K(x_i)$	x_i	$K(x_i)$	x_i	$K(x_i)$	x_i	$K(x_i)$
N_{cores}=2	x_{01}	209.60	x_{07}	22.73	x_{13}	28.56	x_{19}	0.82	x_{25}	0.48
N_{cores}=4	x_{02}	52.40	x_{08}	56.88	x_{14}	10.92	x_{20}	0.60	x_{26}	0.44
N_{cores}=6	x_{03}	24.66	x_{09}	28.72	x_{15}	0.79	x_{21}	0.58	x_{27}	0.44
N_{cores}=8	x_{04}	16.55	x_{10}	18.48	x_{16}	0.69	x_{22}	0.56	x_{28}	0.44
N_{cores}=12	x_{05}	8.73	x_{11}	10.84	x_{17}	0.63	x_{23}	0.56	x_{29}	0.44
N_{cores}=24	x_{06}	1.23	x_{12}	0.67	x_{18}	0.62	x_{24}	0.58	x_{30}	0.42

In Table 4, values of c*ost of rules* $K(s_j)$ are shown in order of their occurrence in predictions ($s_j \in \bigcup_{i=1}^{8} S_{train(i)}, j = 1,2,\dots,24$).

Table 4
Cost of rules $K(s_j)$ [ks]

s_j	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$K(s_j)$	313.17	3.70	2.67	2.87	1.85	2.90	0.79	2.29
s_j	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}
$K(s_j)$	1.63	1.09	0.69	22.74	0.67	1.81	34.90	1.30
s_j	s_{17}	s_{18}	s_{19}	s_{20}	s_{21}	s_{22}	s_{23}	s_{24}
$K(s_j)$	0.79	0.63	2.87	1.09	1.10	1.70	6.68	5.68

In Table 5, *costs of predictions* $K(S_{train(i)})$ are shown together with the number of prediction rules $|S_{train(i)}|$ in the prediction ($i = 1,2,\dots,8$).

Table 5
Cost of predictions $K(S_{train(i)})$ [ks] and number of rules in $S_{train(i)}$

Prediction(i)	1	2	3	4	5	6	7	8
$K(S_{train(i)})$	331.85	322.97	351.79	357.85	329.67	364.82	336.05	357.99
$\lvert S_{train(i)} \rvert$	9	7	10	8	10	10	7	6

In the following points, based on simulation results and on RST modeling and train-and-test examination data, cases of single predictions and series of predictions, with and without dropping, are analyzed using *prediction performance coefficients*. The basic series of predictions (prediction case 0) is introduced in point 5.2.2. All the other prediction analysis examples (prediction case 1-5) introduced in the analysis are derived from prediction case 0. The efficacy minimum requirement for all prediction cases is $E1_{limit} \geq 60\%$.

5.2.2 Predictions with Full Set of Attributes and Rules

Prediction case 0: $\forall_{i=1}^{8} S_{prediction(i)} := S_{train(i)}$

Cost data and rule set sizes for the series of predictions are shown in Table 3, 4 and 5. The effectiveness coefficient of prediction series is

$E3_{series} = E3(S_{train}) = \overline{E2}_8 \big| \overline{E1}_8$ where

$\overline{E2}_8 = \frac{1}{8}\sum_{i=1}^{8} E1_i / \frac{1}{8}\sum_{i=1}^{8} K\left(S_{train(i)}\right) = 77\%/344.12ks = 0.224\frac{\%}{ks}$, and

$\overline{E1}_8 = \frac{1}{8}\sum_{i=1}^{8} E1(S_{train(i)}) = 77\% > (E1_{limit} = 60\%)$.

5.2.3 Dropping in Single Predictions: Efficacy and Efficiency Analysis

Prediction case 1: dropping $S_{prediction(1)} := S_{train(1)} \setminus \{s_3\}$

Set of prediction rules $S_{train(1)}$ consists of nine rules ($S_{train(1)} = \{s_1, s_2, \dots, s_9\}$). The limit $E1_{limit} \geq 0.6$ has been accepted, thus the rule $s_3 = (c_{04}(t) \Rightarrow d_{01}(H))$, ($s_3 \in S_{train(1)}$) can be dropped *efficaciously* since the effect of dropping on $E1$ is

$$\left(\left(\left(\sum_{l=1}^{15=|U_{test(1)}|} p(x_l)\right)_{S_{train(1)}} = 11\right) - \left(\left|Match_{U_{test(1)}}(s_3)\right| = 2\right) = 9\right) \geq \left(E1_{limit} * \left|U_{test(1)}\right| = 0.6 * 15 = 9\right).$$

Decision rule $s_1 = (c_{04}(f) \Rightarrow d_{01}(L))$, ($s_1 \in S_{train(1)}$) *cannot* be dropped *efficaciously* because

$$\left(\left(\left(\sum_{l=1}^{15=|U_{test(1)}|} p(x_l)\right)_{S_{train(1)}} = 11\right) - \left(\left|Match_{U_{test(1)}}(s_1)\right| = 3\right) = 8\right) \ngeq \left(E1_{limit} * \left|U_{test(1)}\right| = 0.6 * 15 = 9\right).$$

Cost of rule s_3 is $K(s_3) = 2.67ks$ thus after dropping s_3 $K\left(S_{train(1)} \setminus \{s_3\}\right) = 329.18ks$. The efficacy and efficiency coefficients after dropping are

$E1\left(S_{train(1)} \setminus \{s_3\}\right) = 60\%$, $E2\left(S_{train(1)} \setminus \{s_3\}\right) = 0.182\frac{\%}{ks}$ respectively. Comparing $E2_{limit} = E1_{limit} / \frac{1}{8}\sum_{i=1}^{8} K\left(S_{prediction}\right)$ for both before and after dropping cases, the next inequality has been got: $E2_{limit}\left(S_{train(1)}\right) = 0.181\frac{\%}{ks} <$

$< E2_{limit}\left(S_{train(1)} \setminus \{s_3\}\right) = 0.182\frac{\%}{ks}$, that is $S_{train(1)}$ is worse in effectiveness.

Prediction case 2: dropping $C_{prediction(1)} := C \setminus \{c_{03}\}$

Extending the previous case, the *attribute* c_{03} may be dropped *efficaciously* if the following set of *sufficient* conditions can be satisfied:

$$\exists S_{EF(1)} \left(S_{train(1)} = \left(S_{EF(1)} \cup S_{FE(1)}\right)\right) \wedge \left(\{c_{03}\} \notin C_{S_{EF(1)}}\right) \wedge E1\left(S_{EF(1)}\right) \geq$$

$\geq E1_{limit}$, where set $C_{S_{EF(1)}}$ denotes the set of condition attributes used by rules of set $S_{EF(1)}$. The partitioning $S_{train(1)} = (S_{EF(1)} = \{s_1, s_2, s_3\}) \cup S_{FE(1)} = \{s_4, s_5, s_6, s_7, s_8, s_9\}$, satisfies the conditions since $E1\left(S_{\mathrm{EF}(1)}\right) = 60\ \% = E1_{limit}$ and $\left(C_{S_{FE(1)}} \setminus C_{S_{EF(1)}}\right) = \{c_{03}, c_{04}\} \setminus \{c_{04}\} = \{c_{03}\}$.

The cost of prediction after dropping c_{03} is $K\left(S_{train(1) \setminus \{c_{03}\}}\right) = K\left(S_{EF(1)}\right) =$

$= 319.54ks$, thus the efficiency got is $E2\left(S_{train(1) \setminus \{c_{03}\}}\right) = 0.188\frac{\%}{ks}$.

5.2.4 Dropping in Series of Predictions: Effectiveness Analysis

Prediction case 3: dropping $\forall_{i=1}^{8} S_{prediction(i)} := S_{train(i)} \setminus \{s_1\}$

The effectiveness *after* dropping the *rule* s_1 is

$E3_{series} = E3(S_{train} \setminus \{s_1\}) = \overline{E2}_8 | \overline{E1}_8$ where

$\overline{E2}_8 = \frac{1}{8}\sum_{i=1}^{8} E1_{(i)} / \frac{1}{8}\sum_{i=1}^{8} K\,(S_{train(i)} \setminus \{s_1\}) = 1.680\frac{\%}{ks}$, and

$\overline{E1}_8 = \frac{1}{8}\sum_{i=1}^{8} E1(S_{train(i)} \setminus \{s_1\}) = 52\%$.

In this case, dropping s_1 is highly *efficient* (since $\overline{K}(S_{train}) = 344.12ks$ and $K(s_1) = 313.17ks$) but the dropping is not *efficacious* because $52\% < E1_{limit} = 60\%$ and thus the dropping is not *effective* too.

Prediction case 4: dropping $\forall_{i=1}^{8} S_{prediction(i)} := \{s_1, s_2, s_3\}$

For the case of this rule dropping effectiveness is

$E3_{series} = E3(\{s_1, s_2, s_3\}) = \overline{E2}_8 | \overline{E1}_8$ where

$\overline{E2}_8 = \frac{1}{8}\sum_{i=1}^{8} E1_i / \frac{1}{8}\sum_{i=1}^{8} K\,(S_{train(i)} = \{s_1, s_2, s_3\}) = 0.194\frac{\%}{ks}$, and

$\overline{E1}_8 = \frac{1}{8}\sum_{i=1}^{8} E1(S_{train(i)} = \{s_1, s_2, s_3\}) = 62\% > (E1_{limit} = 60\%)$.

Prediction case 5: dropping $\forall_{i=1}^{8} S_{prediction(i)} = S_{train(i)} | C := C \setminus \{c_{03}\}$

Attribute c_{03} could be dropped *efficaciously* for the series of predictions (continuing the analysis of prediction case 2) since the *sufficient* set of conditions,

$\bigvee_{i=1}^{8} \exists S_{EF(i)} \left(S_{train(\mathrm{i})} = \left(S_{EF(i)} \cup S_{FE(i)}\right)\right) \wedge (\{c_{03}\} \notin C_{S_{EF(i)}})) \wedge$

$\wedge \left(\left(\frac{1}{8}\sum_{i=1}^{8} E1\left(S_{EF(i)}\right)\right) \geq E1_{limit}\right)$, can be satisfied by simply taking $\bigvee_{i=1}^{8} S_{EF(i)} = \{s_1, s_2, s_3\}$ for this case too. Thus, the effectiveness evaluation gives the same numbers as for prediction case 4:

$E3_{series} = E3(S_{EF} | C \setminus \{c_{03}\}) = \overline{E2}_8 | \overline{E1}_8$ where

$\overline{E2}_8 = \frac{1}{8}\sum_{i=1}^{8} E1_i / \frac{1}{8}\sum_{i=1}^{8} K\,(S_{train(i)} = S_{EF(i)}) = 0.194\frac{\%}{ks}$, and

$\overline{E1}_8 = \frac{1}{8}\sum_{i=1}^{8} E1(S_{EF(i)}) = 62\% > (E1_{limit} = 60\%)$.

6 Comparison and Discussion of Predictions

Figure 4 compares three series of predictions $S_{prediction} = S_{train}$ (case 0), $S_{prediction} = \{s_1, s_2, s_3\}$ (case 4) and $S_{prediction} = S_{train} \setminus \{s_1\}$ (case 3). The diagram shows $E1$ and $E2$ coefficients for every single prediction and shows $\overline{E2}$ and $\overline{E1}$ values of $E3$ for the series of predictions too. To make the comparison of predictions easier, the efficiency coefficients are calculated such that $K\left(S_{prediction} = \{s_1, s_2, s_3\}\right) = 319.54ks = 100\%$.

For the single predictions of $S_{prediction} = S_{train}$, the relation $\forall_{i=1}^{8} E1_{(i)}\big(E1_{(i)} \geq \geq E1_{limit}\big)$ is hold, and the relative efficiency of the series $\overline{E2}$ is 72% ($\overline{E1}$= 77%).

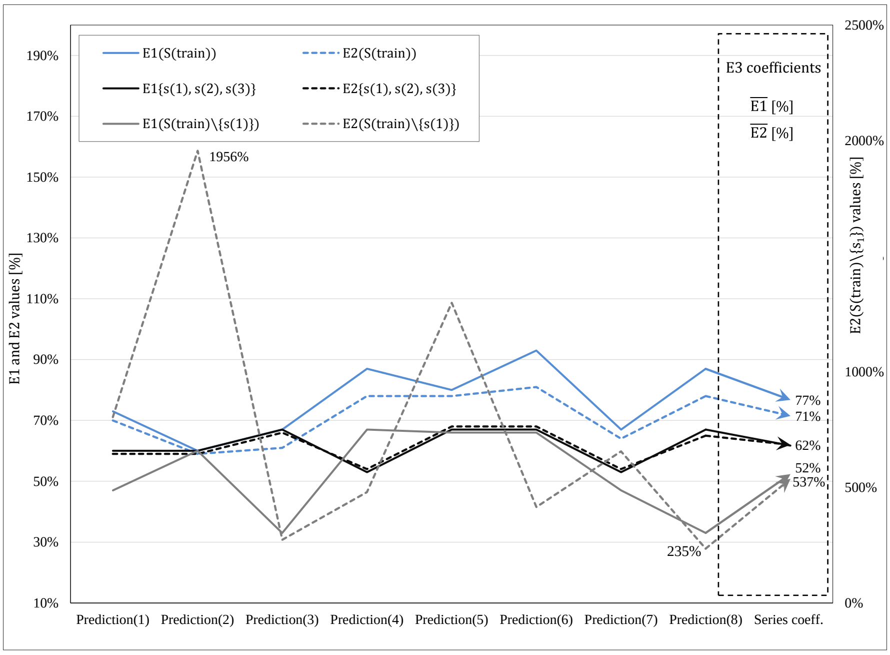

Figure 4
E1, E2 and E3 evaluation of predictions and droppings

For $S_{prediction} = \{s_1, s_2, s_3\}$, the relation $\forall_{i=1}^{8} E1_{(i)}(E1_{(i)} > 50\%)$ is true for every single prediction. Efficacy of series is $\overline{E1} = 62\%$, with a relative efficiency of $\overline{E2} = 62\%$ too. If $\overline{E1} = 60\% = E1_{limit}$ has been accepted then $S_{prediction} = \{s_1, s_2, s_3\}$ performs better: $\overline{E2}(\{s_1, s_2, s_3\})$ =60% and $\overline{E2}\big(S_{prediction} = S_{train}\big) =$ =56%.

Dropping s_1 ($S_{prediction} = S_{train} \setminus \{s_1\}$) leads to a significant decrease of $\overline{E1}$ (77% → 52%), and results in an *ineffective* series of predictions. All the single predictions of $E2(S_{train} \setminus \{s_1\})$ are in the range between 235% and 1956%. The efficiency of the series $\overline{E2}(S_{train} \setminus \{s_1\})$ is 537%. The rule with the highest cost s_1 has significant influence on efficacy and dominating influence on efficiency of predictions. (Rule s_1 is included in $S_{prediction} = \{s_1, s_2, s_3\}$ and its proportion in the cost of the whole set is 98% (Table 4).)

This evaluation indicates the need for the feedback to model identification phase.

The s_1 rule has the form $s_1 = (c_{04}(f\,) \Rightarrow d_{01}(L))$ where $c_{01}(f) = v'_{c_{04}} = 0.1$ and $d_{01}(L) = v'_{d_{01}} < 0.6$. The relation $Supp(s_1) = 1$ is true for the

$lookahead = 0.1$ column in Table 2. The cost of the rule $K(s_1) = \sum_{l=1}^{6} K(x_l) = 313.17ks$ is the sum of costs of 6 experiments of the universe $x_{01} - x_{06}$ in Table 3. It means that if objects $x_{01} - x_{06}$ are excluded from the model, then there is no need for the high cost s_1 rule.

This modification of the model allows decreasing prediction costs without losing the prediction power in ranges with higher potential speed increases.

Conclusions

In past years, numerous simulation performance prediction methods have been developed that support simulation model development for PDES, since PDES execution can significantly decrease model runtime and developing simulation models with high PDES runtime features have remained challenging tasks. The emerging execution platforms with on-demand access and charge-per-use services bring into focus, the importance of cost/quality evaluation in performance predictions. Here, we have defined prediction performance improvement operations, based on the system of performance coefficients of efficacy (E1), efficiency (E2), effectiveness (E3), characterizing prediction correctness, evaluating correctness to cost relationship and describing correctness cost behavior for a series of predictions, respectively. We included the evaluation and improvement steps in a traditional RST train-and-test algorithm and added it to a classic CFM to form an integrated prediction method. We presented the work of the improved prediction method on a case study of a Closed Queuing Network (CQN) model PDES CFM analysis. In the analysis of the CFM RST model and results of the train-and-test examination, we presented the use of E1, E2 and E3 coefficients for rule and attribute dropping operations and the feedback to the CFM and RST model-identification phase was shown. For the case study implementation, the OMNet++ DES framework and the Rosetta Rough Set Software System were used. Future research is planned to focus on the effectiveness evaluation of simulation model output data, preprocessing for rough modeling and on the application of the method, for methods other than CFM.

References

[1] Perumalla, K. S., (2010) μ*π*: a Scalable and Transparent System for Simulating MPI Programs, In Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p. 62

[2] Perumalla, K. S., (2006) Parallel and Distributed Simulation: Traditional Techniques and Recent Advances, In Proceedings of the 38th Winter Simulation Conference, pp. 84-95

[3] D’Angelo, G., (2011) Parallel and Distributed Simulation from Many Cores to the Public Cloud (Extended Version), In: Proceedings of the 2011

International Conference on High Performance Computing and Simulation (HPCS 2011) Istanbul (Turkey) IEEE, July 2011, ISBN 978-1-61284-382-7 http://dx.doi.org/10.1109/HPCSim.2011.5999802,arXiv preprint arXiv:1105.2301. pp.1-9

[4] Yoginath, S. B., Perumalla, K. S., (2013) Empirical Evaluation of Conservative and Optimistic Discrete Event Execution on Cloud and VM platforms, In Proceedings of the 2013 ACM SIGSIM conference on Principles of advanced discrete simulation, pp. 201-210

[5] Fujimoto, R. M., (1990) Performance of Time Warp under Synthetic Workloads, In Proceedings of the SCS Multiconference on Distributed Simulation, San Diego, CA, USA, 17-19 January, 1990, pp. 23-28

[6] Fujimoto, R. M., (2000) Parallel and Distributed Simulation Systems, John Wiley & Sons, USA, ISBN: 0-471-18383-0

[7] Kunz, G.; Tenbusch, S., Gross, J., Wehrle, K., (2011) Predicting Runtime Performance Bounds of Expanded Parallel Discrete Event Simulations, In Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on IEEE, pp. 359-368

[8] Juhasz, Z., Turner, S., Kuntner, K., Gerzson, M., (2003) A Performance Analyser and Prediction Tool for Parallel Discrete Event Simulation, International Journal of Simulation, Vol. 4, No. 1, May 2003, pp. 7-22

[9] Muka, L., Lencse, G., (2008) Cooperating Modelling Methods for Performance Evaluation of Interconnected Infocommunication and Business Process Systems, In Proceedings of the 2008 European Simulation and Modelling Conference (ESM'2008), Le Havre, France, Oct. 27-29 EUROSIS-ETI, pp. 404-411

[10] Muka, L., Lencse, G., (2011) Method for Improving the Efficiency of Simulation of ICT and BP Systems by Using Fast and Detailed Models, Acta Technica Jaurinensis, Vol. 5, No. 1, pp. 31-42

[11] Muka, L., Benko, B. K., (2013) Meta-Level Performance Management of Simulation: The Problem Context Retrieval Approach, Periodica Polytechnica, Electrical Engineering and Computer Science, 55(1-2), pp. 53-64, DOI: 10.3311/pp.ee.2011-1-2.06

[12] Varga, A., Hornig, R., (2008) An Overview of the OMNeT++ Simulation Environment, In Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 60-69

[13] Komorowski, J., Øhrn, A., Skowron, A., (2002) The ROSETTA Rough Set Software System, In Handbook of Data Mining and Knowledge Discovery,

W. Klösgen and J. Zytkow (eds.), Oxford University Press, ch. 2-3, ISBN 0-19-511831-6

[14] Varga, A., Sekercioglu, Y. A., Egan, G. K. (2003) A Practical Efficiency Criterion for the Null Message Algorithm, Proceedings of the European Simulation Symposium (ESS 2003), Oct. 26-29, 2003, Delft, The Netherlands, SCS International, pp. 81-92

[15] Lencse, G., Varga, A., (2010) Performance Prediction of Conservative Parallel Discrete Event Simulation, Proceedings of the 2010 Industrial Simulation Conference (ISC'2010), Budapest, Hungary, 7-9, June, 2010, EUROSIS-ETI, pp. 214-219

[16] Lencse, G., Derka I., Muka, L., (2013) Towards the Efficient Simulation of Telecommunication Systems in Heterogeneous Execution Environments", In proceedings of: TSP2013, Edited by Norbert Herencsar, Karol Molnar, the 36th International Conference on Telecommunications and Signal Processing, Rome, Italy, pp. 304-310

[17] Lencse, G., Derka, I., (2013) Testing the Speed-up of Parallel Discrete Event Simulation in Heterogeneous Execution Environments, In proceeding of: ISC'2013, Edited by Veronique Limere and El-Houssaine Aghezzaf, ISBN 978-90-77381-76-2, 11th Annual Industrial Simulation Conference, Ghent University, Ghent, Belgium, pp. 101-107

[18] Lencse, G., (1998) Efficient Parallel Simulation with the Statistical Synchronization Method, Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation (CNDS'98), San Diego, CA. Jan. 11-14, SCS International, pp. 3-8

[19] Lencse, G., (2002) Parallel Simulation with OMNeT++ using the Statistical Synchronization Method, Proceedings of the 2nd International OMNeT++ Workshop, Jan. 8-9, Technical University Berlin, Berlin, Germany, pp. 24-32

[20] Lencse, G., (2009) An Efficient Statistical Synchronization Method for Parallel Simulation, Journal on Information Technologies and Communications, (ISSN 0866-7039) Volume E-1, 1(5), pp. 15-23

[21] Preisler, T., Dethlefs, T., Renz, W., (2015) Simulation as a Service: A Design Approach for Large-Scale Energy Network Simulations, In Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on IEEE, pp. 1765-1772

[22] Krishnaswamy, S., Zaslavsky A., Loke, S. W., (2002) Predicting Run Times of Applications Using Rough Sets, In Ninth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002), pp. 455-462

[23] Song, C., Guan, X., Zhao, Q., Ho, Y. C., (2005) Machine Learning Approach for Determining Feasible Plans of a Remanufacturing System,

Automation Science and Engineering, IEEE Transactions on, 2(3), pp. 262-275

[24] Muka, L., Derka, I., (2013) Improving Performance Prediction of Parallel and Distributed Discrete Event Simulation: A Rough Sets-based Approach, In proceeding of: ISC'2013, Edited by Veronique Limere and El-Houssaine Aghezzaf, ISBN 978-90-77381-76-2, 11th Annual Industrial Simulation Conference, Ghent University, Ghent, Belgium, pp. 95-100

[25] Pawlak, Z.; (1982) Rough sets, International Journal of Parallel Programming, 11(5), pp. 341-356

[26] Pawlak, Z., Skowron, A., (2007) Rudiments of Rough Sets, Information sciences, 177.1, pp. 3-27

[27] Zhang, J., Li, T., Ruan, D., Gao, Z., Zhao, C., (2012) A Parallel Method for Computing Rough Set Approximations, Information Sciences,194.2, pp. 209-223

[28] Bazan, J. G., Szczuka, M, (2005) The Rough Set Exploration System, In Transactions on Rough Sets III, Springer Berlin Heidelberg, pp. 37-56

[29] Suraj, Z., (2004) An Introduction to Rough Set Theory and Its Applications, ICENCO, Cairo, Egypt, pp. 1-39

[30] Gregory, F., (1993) Cause, Effect, Efficiency and Soft Systems Models, Journal of the Operational Research Society, Vol. 44 (4), pp. 333-344

[31] Checkland, P., (2000) Soft Systems Methodology: a Thirty Year Retrospective, Systems Research and Behavioral Science,17, S11-S58

[32] Muka, L., Derka, I., (2014) Simulation Performance Prediction in Clouds, Proceedings of 9th International Symposium on Applied Informatics and Related Areas (AIS2014), Székesfehérvár, Hungary, November 12, 2014, pp. 142-147, ISBN: 978-615-5460-21-010