Time Series-Oriented Load Prediction Model and Migration Policies for Distributed Simulation Systems

Robson E. De Grande, Azzedine Boukerche, and Raed Alkharboush

***Abstract*—HLA-based simulation systems are prone to load imbalances due to lack management of shared resources in distributed environments. Such imbalances lead these simulations to exhibit performance loss in terms of execution time. As a result, many dynamic load balancing systems have been introduced to manage distributed load. These systems use specific methods, depending on load or application characteristics, to perform the required balancing. Load prediction is a technique that has been used extensively to enhance load redistribution heuristics towards preventing load imbalances. In this paper, several efficient Time Series model variants are presented and used to enhance prediction precision for large-scale distributed simulation-based systems. These variants are proposed to extend and correct the issues originating from the implementation of Holt's model for time series in the predictive module of a dynamic load balancing system for HLA-based distributed simulations. A set of migration decision-making techniques is also proposed to enable a prediction-based load balancing system to be independent of any prediction model, promoting a more modular construction.**

***Index Terms*—Load Balancing, Prediction, Holt's Model, High Level Architecture.**

I. Introduction

DISTRIBUTED simulations, such as large HLA-based simulations, have provided easy and fast implementations for different complex problems that many researchers seek answers to, such as determining the effects of atmosphere on flights [1], analyzing aircraft combat [2], or devising traffic solutions [3]. Distributed simulations depend on distributed and parallel/concurrent systems. Thus, several factors can affect the performance of the distributed simulation: improper distribution of simulation entities among the resources, the existence of external load in the resources, resource heterogeneity, and dynamic changes in the simulation load. Static load partitioning can initially distribute simulation entities based on resource heterogeneity but fails when dynamic load changes exist. As a result, several dynamic load balancing systems [4], [5], [6], [7], [8] have been designed to prevent load imbalances.

High-Level Architecture (HLA) is a framework for the design and management of distributed simulations through a set of management services. HLA simulations are very likely to undergo execution issues due to the lack of management of resources and load. Computational load stress oscillations might occur according to changes in processing or be triggered by dependencies on simulated logical processes. Consequently, HLA, as well as other simulation frameworks, cannot provide any load balancing, limiting or even disabling of the execution of these processes.

Due to the fact that distributed systems are susceptible to communication and computation load imbalances, extensive work has been conducted to prevent or detect them. Distributed simulations show some particular inter-dependencies among logical processes, which can characterize these applications as one specific type of distributed system, also prone to load oscillations. It is evident that dynamically redistributing load for non-dedicated resources is a highly complex task, which is reduced to a np-hard problem; therefore, most of the existing balancing solutions employ heuristics.

Balancing systems have used different mechanisms for considering diverse aspects in distributed environments and simulations, such as communication load [9], migration load [10], and prediction [11]. The particular predictive balancing scheme analyzed in this work manages load at the infrastructure level and uses an extension of Exponential Weighted Moving Average (EWMA) with double exponential smoothing for load forecasting, known as Holt's model. However, the efficiency of the balancing system is found to be dependent on the used prediction technique. Thus, different prediction mechanisms are considered for better suiting the model to the observed load oscillations.

In this work, we aim to increase the performance of the predictive balancing system by improving its prediction accuracy and enhancing its migration decision-making. Thus, we propose a set of suitable and different variants to Holt's model in order to introduce an innovative adaptation for the used distributed load balancing systems [12]. Such variants are built with the aim of improving the execution performance of distributed systems, with particular attention to the load behavior of distributed simulations, specifically HLA-based simulations. The variants originate from an extensive analysis of the model's limitations. Besides that, four federate migration decision-making techniques are described as independent from a particular prediction model. The biased, static method employed in the decision-making of the balancing system is generalized to become more flexible and more easily adaptable to other prediction models and load behaviors, leading the system to an improvement in performance gain.

This work expands substantially on our previous works, introduced in [13] and [14]. It adds extensive analyses on the

R. E. De Grande and A. Boukerche are with the School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada. R. Alkharboush is with Saudi Aramco. E-mail: rdgrande@site.uottawa.ca, ralkh092@uottawa.ca, and boukerch@site.uottawa.ca

Holt's variants, including a fourth variant, Genetic Algorithm Variant (GAV); it introduces new variants of migration and decision policies while significantly extending the performance evaluation of the proposed migration decision-making techniques, and it includes analyses of the implementations of every combination of Holt's variant and decision-making technique through comprehensive experimental evaluations.

The remainder of this paper is organized as follows. Section II presents related work. Section III shows a previously designed prediction-based load balancing scheme. Section IV describes the proposed Holt's model variants. Section V introduces a number of techniques to make migration decisions more efficient. Section VI defines the experiments and discusses the results. Finally, Section VII summarizes the article and draws some guidelines for future work.

II. Related Work

Many load balancing schemes have been developed to increase the performance of virtual simulation applications [4], [5], [6], [7], [8]. Some of these schemes have been developed for optimistic simulations [15], others for conservative simulations [16], [17], and still others for HLA-based simulations [18]. A predictive load balancing system, described in Section III, has been designed to introduce load forecast future scenarios. Thus, accuracy and precision are essential for balancing efficiency, showing the importance of prediction models that can be used to represent load oscillations.

1) Similar Day Approach: This approach covers the historical data search; the method seeks characteristics that are similar to those that the system experiences in a given moment and uses them to predict the load. The computed prediction provides one or more readings that follow a linear combination of historical data. The concept of similar day has been used to model a system for ionospheric forecasting [19]. Simulation execution data can be used in run-time to build the historical data, but the simulation load generally does not show a seasonal pattern. Similar day is thus unsuitable for short-term load predictions, due to its lack of seasonality.

2) Expert System: This is a rule-based intelligence system. The rules are heuristic in nature and are generated by experts in the field. This technique has been used in a different number of fields. For instance, information pertaining to the operators and the system load of the Taiwan Power system was employed to propose an automatic knowledge-based expert system [20]. Embedding an expert system is not preferred for the prediction of the resource loads that run distributed applications. It is challenging to recognize a rule that unifies the behavior of all possible scenarios and produces a precise projection.

3) Regression Methods: These methods are employed to identify the possible connections between a dependent variable and a set of independent variables. Such prediction methods are most widely used in electrical load forecasting [21]. Regression method is used to find a model that represents the relationship between the load consumption and other factors, such as weather, season, and the type of homes. Several regression models have been adopted to generate the load prediction of the next day [22], incorporating changes in weather, season, and average load. Implementing regression models in our system is not feasible as the only visible variable is the system load.

4) Artificial Neural Network: In many fields, Artificial Neural Network (ANN) has been and still is applied for forecasting. Neural networks have the ability to demonstrate non-linear fitting. For instance, the Greek Public Power Corporation [23] made use of a neural network system in its Energy Control Center for forecasting load. To predict loads for distributed systems, historical data is needed. However, casual artificial neural networks do not provide the means for the use of historical data. On the other hand, Recurrent Neural Network (RNN) uses historical data and is capable of recognizing sequences that are unrecognizable by ANN; this makes RNN suitable for pattern recognition applications.

5) Support Vector Machine: Support Vector Machine (SVM) is a well-established model and is associated with learning techniques to solve classification and regression problems. Derived from statistical learning theory [24], SVM nonlinearly maps the data into a higher dimensional space and then employs linear functions to create boundaries. SVM has been used for the projection of electrical loads in a short-term future [25], outperforming autoregressive methods. SVM requires training data to perform well, which is outside the scope of this work.

6) Fuzzy Logic: Fuzzy Logic is an approximation approach in which inputs are assigned to sets based on predefined rules. One of the advantages of using Fuzzy Logic is that it does not need a mathematical model to map inputs to outputs, resulting in a robust forecasting. A de-fuzzification process can be used to generate a precise output, but it requires detailed information. Fuzzy Logic has been used in various instances of electric load forecasting [26]. This process is not feasible for the balancing system used in this work due to the low number of available inputs, enabling a few output sets that are unhelpful for producing an exact estimation.

7) Time Series Methods: Time series approaches assume that there is a hidden relationship between the internal structure of the data, such as autocorrelation, trend, or seasonal variation. Autoregressive Moving Average (ARMA) is mostly used with stationary processes while Autoregressive Integrated Moving Average (ARIMA) is used with non-stationary processes [27]. To analyze the time series efficiency, extensive evaluations have been realized for different applications.

Unlike Fuzzy Logic and neural networks, auto-regression models do not need training and provide reasonable short-term predictions [28]. Univariate methodological approaches have been analytically studied in the scope of short-term predictions for electricity load [29]. In this context, Holt-Winters gave the best performance, being a simple yet robust technique. An empirical study has been realized to evaluate the performance of different prediction models, up to one day ahead, for seasonal data [30]. In this second study, the double seasonal Holt-Winters exponential smoothing presented the best performance against the ARIMA and Principal Components Analysis for half-hour predictions. In another study setup with historical data, Holt's model and double exponential smoothing performed the best and showed a very small error

rate [31]. Another evaluation analyzed the performance of multiple time series forecasting to predict trends [32], showing that Holt's model performed better than single exponential smoothing.

Based on the aforementioned advantages and disadvantages of each prediction method, time series [27] can provide a performance that matches our prediction goals. Time series models provide good short-term projections. As previously presented, Holt's model rises as the most suitable method for predicting computational load oscillations.

III. Prediction-based Dynamic load balancing system

The modifications and enhancements of Holt's prediction technique are incorporated into the prediction-based dynamic load balancing system described in [11], shown in Figure 1. The load balancing process consists of monitoring the resources to identify imbalances in the simulation processes, redefining load distribution, and conducting load migrations. In this work, load is restricted to computational load, or consumption of processing power, that simulation processes apply to resources.

A Cluster Load Balancer (CLB) handles all shared resources and federates within a cluster. CLBs are connected with neighbouring CLBs to exchange balancing information, as well as with Local Load Balancers (LLB). A CLB is responsible for aggregating data, determining re-distribution, and identifying migrations. The Monitoring Information Service provides each CLB with load-related information, using Grid services [33]. LLBs are placed on shared resources in order to interface with simulations, gathering simulation load data and forwarding migration calls, as described in [34].

The Prediction Engine, with the assistance of Prediction History and Prediction Model, processes the incoming data. The Prediction History provides a data history that depends on the Prediction Model as each prediction model varies in data history requirements. In this model, a *smoothed value* and a *trend* are saved in the Prediction History as our work expands on Holt's double smoothing technique. The Prediction Model provides projections based on current load data and data from the Prediction History.

A. Redistribution Algorithm

The redistribution algorithm is used to rearrange the federates in the distributed simulation to have a balanced environment, as shown in Algorithm 1. The distributed dynamic load balancing system periodically *monitors* the shared resources. This allows the system to be responsive to load changes. The monitoring is performed periodically every Δt, which is every 20 seconds, producing minimum overhead and considerable awareness of load oscillations.

Filtering and normalization are applied to the collected data. Afterwards, the collected data, previous history, and migration status are used to provide load projections in three different ranges, according to the balancing cycles of the load management system; the named short-term, medium-term, and long-term are defined as 1, 3, and 5 balancing cycles

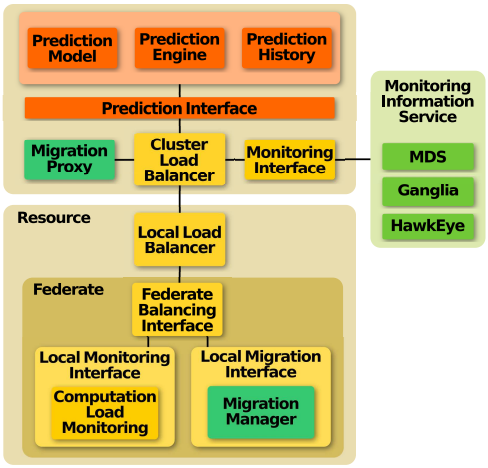


Fig. 1. General Architecture of the Predictive Balancing System

in the future, respectively. The load balancing system then applies a pair-matching mechanism between overloaded and underloaded resources in each balancing cycle. The system assigns a greater importance to projections that are closer to the current time.

A CLB first performs the load redistribution in a local scope and then in an inter-domain scope. The inter-domain balancing is needed to exchange loads between clusters of resources. For both scopes, the algorithm attempts to identify pair matches. The mechanism initiates a migration call if the difference between the load of the overloaded and the underloaded resources exceeds a threshold. The process then concludes by sending the migration calls to the requester CLB.

B. Forecasting Load Status

The predictive load balancing system performs on three different load forecasting levels: short-, medium-, and long-term. The three projections are independent of the prediction model, and their values are relative to the balancing cycle, which indicates the frequency with which the load is balanced. From experimental analysis, 5 balancing cycles corresponded to the farthest prediction in the future with an acceptable error in forecasting. Short-term projection (SP) aims to solve current load imbalances while medium- and long-term projections (MP and LP) are used to prevent load imbalances that may occur. Each projection represents a forecast of a number of balancing cycles. Short-, medium-, and long-term cycles are then defined as 1, 3, and 5 balancing cycles respectively.

The threshold used to compare overloaded and underloaded resources is adaptive and is modified based on the direction of the tendency of a resource load. The amount of adjustment needed is proportional to the balancing cycle. Thus, medium- and long-term projections receive larger adjustments.

C. Prediction Models

Load data can be naturally characterized as a time series as it is collected at fixed time intervals. Exponential Weighted Moving Average (EWMA) is a well-known time series prediction technique that observes the internal relationship of elements

Algorithm 1 Distributed Dynamic Load Balancing Algorithm

```
1: while TRUE do
2:     mng_loads ← fetch_monitoring_data()
3:     mig_moves.add(local_bal(mng_loads, SP))
4:     mig_moves.add(local_bal(mng_loads, MP))
5:     mig_moves.add(local_bal(mng_loads, LP))
6:     send_migration_moves(mig_moves)
7:     if mig_moves = 0 then
8:         data_neighbours ← reques_Neighbour_Load_Data()
9:     else
10:        if relFactor ≥ random_number(1, 100) then
11:            data_neighbours ← reques_Neighbour_Load_Data()
12:        else
13:            data_neihgbours ← 0
14:        end if
15:    end if
16:    neighbours ← identify_neighbour_Less_Load()
17:    while neighbours ≠ 0 do
18:        overloaded_resource.add(select(firstNeighbour, SP))
19:        overloaded_resource.add(select(firstNeighbour, MP))
20:        overloaded_resource.add(select(firstNeighbour, LP))
21:        federates ← select(spec_loads, overloaded)
22:        eliminate_first(neighbours)
23:    end while
24:    send_to_neighbour(overloaded_resources, federates)
25:    migration_moves ← wait_for_migration_moves()
26:    send_migration_moves(migration_moves)
27:    wait(Δ)
28: end while
```

in three different aspects [35]. Due to the lack of well-defined seasonality in load oscillations, the double EWMA has been used as the prediction model.

Holt's model, represented by Formulas 1 and 2, is applied to the collected data in order to forecast the load. This is done by computing the value of F_{m+i} in Formula 3. The double exponential smoothing computes the predicted load of the collected list of loads. It first finds the current smoothed value, sum_i, based on the current actual load, $elem_i$, the previous smoothed value, sum_{i-1}, and the previous trend, t_{i-1}. The trend enables the extrapolation of the average of the smoothed value. The tendency of the load is defined by the sign of the trend. Formula 4 calculates the forecast for time interval m composed of a series of equally spaced future time cycles. Holt's model requires setting up two constants: α, the data smoothing factor, and β, the trend smoothing factor.

$$sum_i = \alpha \cdot elem_i + (1-\alpha) \cdot (sum_{i-1} + t_{i-1}), 0 \leq \alpha \leq 1 \quad (1)$$

$$t_i = \beta \cdot (sum_i - sum_{i-1}) + (1-\beta) \cdot t_{i-1}, 0 \leq \beta \leq 1 \quad (2)$$

$$F_{i+m} = sum_i + m \cdot t_i, m \in \{1, 3, 5\} \quad (3)$$

$$SP = F_{i+1}, MP = F_{i+3}, LP = F_{i+5} \quad (4)$$

IV. Holt's Model Variants

Holt's model is one of the most popular Time Series prediction models as it provides reasonable predictions with low computational needs. However, the model presents drawbacks that affect the precision of predictions. Thus, the proposed variants comprehend adapting parameters and extending Holt's method to accommodate the different situations that affect its efficiency, especially under frequent, short-term oscillations.

A. Limitations of the Model in Distributed Load

In order to identify areas to improve load prediction under the effect of smoothed value and trend, data samples were collected from running real simulations deployed on a cluster system. Table I shows data samples that range over different load stress scenarios with different oscillation patterns. A set of 20 data samples was used for each configuration.

TABLE I
Parameters of the Collected Data Sample

Parameter	Value
Load oscillations	∼40-100 seconds
Load intensity	∼0-1 core consumption
Load oscillation switch	normal distribution
Load setups	100, 500, and 900 federates

The variables of Holt's model were evaluated to identify reactions towards different load scenarios. The results pointed out that two major weaknesses were affecting the Holt's model on the analyzed data sample.

1) Projection Linearity: This limitation originates from the relationship between the fixed time intervals of the balancing cycles and the projection obtained from the load forecast. In the forecast presented in Formula 4, the time interval, m, has a linear relationship with the computed trend, which is added to the computed smoothed value. For i, a given point in time, and a fixed sum_i and t_i, the computed projections (SP, MP, and LP) follow a linear projection.

The linear projection defines the trend, and it can be used to calculate *far-in-the-future* projections with reasonable precision in the absence of oscillations. However, oscillations are very common, so projections become less realistic. Two variants are proposed, *Cutoff* and *Separate Systems*, to provide better projections for this issue.

2) Slow Response: This issue is related to oscillations that are slow to adapt to sudden loads. The computed trend, t_i, does not always represent, or match, the actual tendency, as depicted in Figure 2. The sign of t_i represents the direction of the computed smoothed load, and the value of the computed trend shows the steepness of the tendency.

In the data sample, t_i slowly adapted its signs and value to follow the oscillations due to its dependence on the previously computed tendency, t_{i-1}. Considering Formula 2, increasing β can help create a more reactive system that is more sensitive to load oscillations. However, this causes the systems to improperly generate projections, as they are susceptible to any oscillation. Thus, a method for handling the computed trend is needed in order to obtain a more responsive system. Two approaches are proposed to resolve this matter: *Reversed Trend* and *Genetic Algorithm Variant*.

B. Proposed Solutions

A number of variants are proposed [13], and each proposed method tackles a different limitation, as listed in the previous subsection. Tests have been conducted to compare accuracy when dealing with distributed simulation data samples.

1) Cutoff: Cutoff is a naive technique that solves the problem of projection linearity. The predicted values depend on the steepness of the linear line generated from Equation 3. If this linear line is steep, projections have unrealistic and unreliable values that a normal system load cannot possibly reach. The cutoff technique restricts the projections, preventing these high projections; this is realized using a threshold.

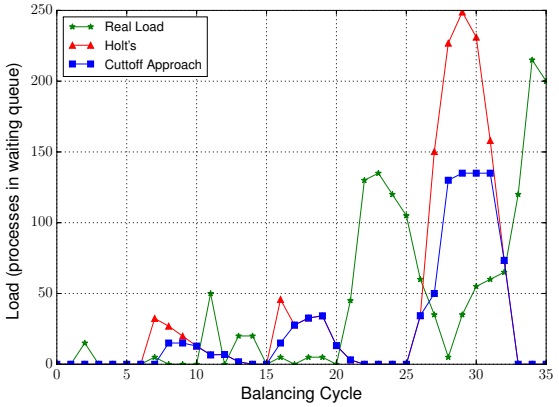


Fig. 2. Cutoff and Holt's computation of long-term prediction

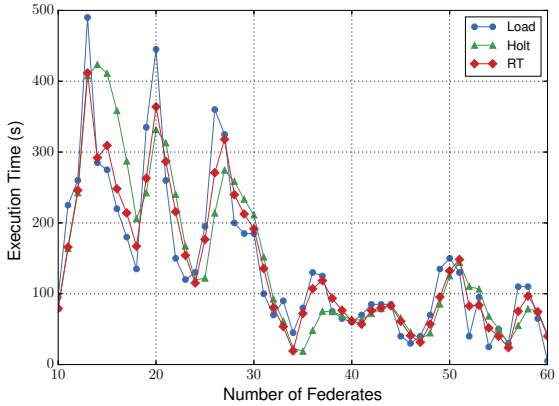


Fig. 3. Computed short-term prediction by Holt's model and RTV

At the initial state of the system, the threshold is set to the current load value. In each balancing cycle, the system computes the three projections (SP, MP, and LP). The system then compares the projection value against the threshold. Being capped by the threshold assures the system that the projections do not produce unrealistic values. The value of the threshold is not considered unrealistic, as it represents the highest value that the system has come across. Equations 5, 6, and 7 list the restrictions to predictions using the threshold.

$$SP = min(F_{m+1}, threshold) \tag{5}$$

$$MP = min(F_{m+3}, threshold) \tag{6}$$

$$LP = min(F_{m+5}, threshold) \tag{7}$$

Figure 2 shows the result of applying the Cutoff variant on the prediction model for a data sample. The implementation shows that when the resource load suddenly oscillates, the projection is *cut off* to reduce the total error that represents the differences between the actual load and the projection value. The Cutoff is expected to decrease the average error rate generated for medium- and long-term predictions; this variant, however, might limit the ability of Holt's model to predict short-term projections as it creates a hidden condition between the current projection and the threshold.

The degradation experienced in calculating the short-term projections increases performance problems when the variant is implemented in the dynamic load balancing system. As the current dynamic load balancing system assigns more importance to short-term predictions, incorrect computations for the short-term projections result in an imbalanced distributed environment when the migration list is generated.

2) Reverse Trend Variant (RTV): This variant aims to make the dynamic load balancing system respond more quickly to load oscillations. Equation 2 shows that t_i depends on t_{i-1}; the dependency on previous calculations causes the trend to respond slowly to load oscillations. The sign of the trend is important as it represents the tendency of the load and thus a major element in predicting load. Therefore, Reverse Trend changes the sign of the trend to match the actual tendency.

Reverse Trend Variant compares the computed trend of Holt's model, t_{i-1}, at the current load with the real trend, $elem_i - elem_{i-1}$. If the directions of the trends do not match, the Reverse Trend technique *reverses* the tendency of the computed trend; this is realized to match the real load tendency. Before the inversion, the Reverse Trend first calculates the differences in load. This prevents *unnecessary sign reversion* in small load oscillations. The reverse trend is not applied for small load oscillations, as they might represent slow temporary load changes and might return to normal in a couple of balancing cycles.

Figure 3 shows a comparison between implementing Holt's model against RTV and computing SP for a data sample. Reversing the trend sign makes the RTV respond to load oscillations faster than the unmodified Holt's model.

As Reverse Trend adopts more sensitivity while reacting to load oscillations and sharpness in projections, it is expected to provide better readings for short- and medium-term projections. However, the performance achieved while projecting long-term loads might decrease; this is because long-term projections must be as smooth as possible, without being sensitive to load oscillations.

3) Separate Systems Variant (SSV): This variant aims to solve the problem of *projection linearity* in Holt's model. Holt's model uses a fixed α and β in order to calculate the prediction of each time interval. α represents the percentage of the effect that the previous smoothing values have on the current smoothing value. From Equation 1, as α increases, it has a greater impact on $elem_i$, and the impact of the previous smoothing value on the current smoothed value is reduced.

From our analysis, short-term projection depends mostly on $elem_i$. On the other hand, medium- and long-term predictions depend more on previous smoothing values: sum_{i-1}. Thus, different fixed constants for each projection of SP, MP, and LP are used to give more weight to each type of prediction, resulting in three different systems. The short-term prediction system uses a larger value of α to place the greater weight of $elem_i$ on sum_i. The medium-term prediction system presents a smaller α to give more weight to sum_{i-1}. The long-term prediction system contains the smallest α.

The Separate Systems Variant is expected to provide better performance for all projections. This is a result of providing a different set of α and β parameters for each system, reflecting the importance of the previously smoothed value and trends for each range of forecasting.

4) Genetic Algorithm Variant (GAV): This variant employs genetic algorithm at each balancing cycle to adjust Holt's

Algorithm 2 Genetic Algorithm Variant (GAV)

Require: $\alpha, \beta, maximumIterations, projections, chromosomes$
$SP \leftarrow 1, MP \leftarrow 3, LP \leftarrow 5$
$smoothing_i \leftarrow smoothing(load_i, smoothing_{i-1}, trend_{i-1}, \alpha)$
$trend_i \leftarrow trend(load_i, smoothing_{i-1}, trend_{i-1}, \beta)$
{values are stored in the structure *proj*}
$proj[SP] \leftarrow \{load_i, smoothing_{i-1}, trend_{i-1}\}$
$proj[MP] \leftarrow \{load_i, smoothing_{i-1}, trend_{i-1}\}$
$proj[LP] \leftarrow \{load_i, smoothing_{i-1}, trend_{i-1}\}$
$proj.removeIndex(0)$
$proj.addNew()$
for $i \leftarrow maximumIterations$ **do**
  $calculateFitness(load_i)$
  $doSelection()$
  $findBest()$
  $doCrossover()$
  $doRandomMutation()$
  if $shouldStop()$ **then**
    $break$
  end if
end for
$\alpha, \beta \leftarrow bestCombination()$

parameters, α and β, dynamically. The optimization focuses on minimizing the difference between the produced forecast and the real load by modifying the parameters. These adjustments are expected to enhance the performance of Holt's model.

Each chromosome in this implementation of the genetic algorithm measures 12-bit in length and is divided in half. Each half represents Holt's model's parameters, α and β. The binary representation is converted to real numbers that fall between 0 and 1.

Algorithm 2 shows the flow that this variant follows to work. First, the smoothed value and the trend are computed using values of α and β. Then, different values are stored in an array that keeps track of 3 different values, the current load, previous smoothing, and previous trend. These values are stored for the three different projections, SP, MP, and LP, in the last 6 balancing cycles in a 6-by-9 array.

In the Genetic Algorithm Variant, long-, medium-, and short-term projections are calculated by the balancing cycles $i-5$, $i-3$, and $i-1$ respectively. Equation 8 shows the fitness function for α and β, which basically measures the difference between the *current load* and each type of projection.

$$f = \frac{6}{3 \cdot (\mid load - SP \mid) + 2 \cdot (\mid load - MP \mid) + (\mid load - LP \mid)} \quad (8)$$

After calculating the fitness for each chromosome, *Roulette Wheel Selection* is employed with the probability of selecting a chromosome of $P(choice = i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)}$, where $\sum_{i=1}^{n} P(i) = 1$. Through this selection, one point crossover is performed N times; subsection VI-A1 shows that generating 50 children chromosomes yields the best efficiency for GAV. Crossover consists of selecting a random number of bits from the first parent and the remaining bits from the second parent. Mutation is then applied by randomly deciding to change the value of each bit of the children chromosomes with a 0.01 probability. The maximum number of iterations, M, is used to limit the genetic algorithm's execution, and the best pair of α and β is chosen to set the Holt's in the next cycle. Subsection VI-A1 demonstrates that 50 corresponds to the best number of iterations for GAV.

Algorithm 3 Restricted Range (RR)

Require: $resourceList$
$mean, tolerance \leftarrow calculate()$
$list_{over}, list_{under} \leftarrow select()$
for all ($source$ **in** $list_{overloaded}$) **and** ($dest$ **in** $list_{underloaded}$) **do**
  if $source \neq$ SAFE **then**
    if $dest \neq$ SAFE **then**
      if $tendency(dest) =$ INCREASING **then**
        if $tendency(source) =$ INCREASING **then**
          break {Current pair is not safe. Analyze next pair of resources}
        else
          $migList[i{+}{+}] \leftarrow matchPair(source, dest)$
        end if
      else
        $migList[i{+}{+}] \leftarrow matchPair(source, dest)$
      end if
    else
      $migList[i{+}{+}] \leftarrow matchPair(source, dest)$
    end if
  else
    $migList[i{+}{+}] \leftarrow matchPair(source, dest)$
  end if
end for
return $migList$

V. Migration Decision-Making Techniques

In this section, we describe four migration decision-making techniques [14]. The pair-match algorithms of the predictive balancing scheme [11] were closely attached and conditioned to the prediction model. The new techniques are used to generate more generic, flexible, efficient pair-matching algorithms.

A. Restricted Range Technique (RR)

The Restricted Range technique replaces the dependency on predefined thresholds with a plain rule-based system. RR uses the load and tendency direction of the source and destination nodes, the mean of all nodes in the current balancing cycle, and a computed tolerance. A *balanced area* designates resources that are considered balanced. RR applies the concept of tolerance to compute a region around the value of the resource's load to define a *resource area*.

The set of rules is generated from observations and conditions in which migrations need to be triggered and is based on whether an intersection exists between the *balanced area* and the *resource area*. A *balanced flag* classifies the resources; a resource is considered *safe* when its *resource area* does not intersect with the *balanced area*. Otherwise, the resource is excluded from the list of resources selected for migrations.

Algorithm 3 summarizes the process for identifying the need to trigger a federate migration. The algorithm shows that almost all of the covered scenarios end up in a migration, eventually issuing a large number of migrations. Once a migration is set, the two involved resources are removed from the candidate list. The process is sequentially applied for each term of projections (short, medium, and long respectively), using a list of resources that gradually shortens.

Based on Algorithm 3, the migration is not only performed when both resources are in *not safe*: the destination shows an increasing load tendency, and the source presents either a decreasing or stable load tendency. Thus, this rule prevents migration calls from being issued which might unbalance the load distribution by an overloaded resource becoming underloaded or a underloaded resource becoming overloaded.

Algorithm 4 Direction Ratio (DR)

Require: $resourceList$
1: $mean, tolerance \leftarrow calculate()$
2: $list_{over}, list_{under} \leftarrow select()$
3: **for all** ($source$ **in** $list_{overloaded}$) **and** ($dest$ **in** $list_{underloaded}$) **do**
4: **if** $source \neq$ SAFE **and** $dest \neq$ SAFE **then**
5: **break** {Current pair is not safe}
6: **else if** $source =$ SAFE **and** $dest =$ SAFE **then**
7: $migList[i{+}{+}] \leftarrow matchPair(source, dest)$
8: **else**
9: **if** $tendency(dest) =$ INCREASING **then**
10: **if** $ratio(source, dest) > 0.5$ **then**
11: $migList[i{+}{+}] \leftarrow matchPair(source, dest)$
12: **else**
13: **break** {Current pair is not safe}
14: **end if**
15: **else**
16: **if** $tendency(source) =$ INCREASING **then**
17: **if** $ratio(source, dest) < 0.5$ **then**
18: $migration \leftarrow matchPair(source, dest)$
19: **else**
20: **break** {Current pair is not safe}
21: **end if**
22: **else**
23: **break** {Current pair is not safe}
24: **end if**
25: **end if**
26: **end if**
27: {Analyze next pair of resources}
28: **end for**
29: **return** $migList$

B. Direction Ratio Technique (DR)

The Restricted Range technique uses a loose set of rules that can cause inconsistency in the federate migration process. Thus, the Direction Ratio technique has been introduced. The restrictions to be applied in the new method limit the number of migrations to those that eventually lead to a balanced environment. In order to achieve this goal, Direction Ratio (DR) adds a *Ratio* to the list of metrics used by RR. The ratio reflects a comparison between load tendencies of the source resource and destination resource.

DR further limits the number of triggered migrations by analyzing the direction of the candidates' loads. Examining different possibilities of the directions gives insight to whether a migration has the potential to affect the load and status of both candidates. The ratio provides a means for verifying the effect factor of a load migration. The ratio is then compared to a fixed value, and based on the comparison, DR either triggers or ignores the migration. As shown in Figure 4, DR uses the ratio to filter the list of migrations.

C. Fuzzy Technique (FT)

Even though Fuzzy Logic cannot be applied as a tool for predicting load oscillation in the context of this work, it can be valuable in facilitating classification and categorization for migration decision-making. The Fuzzy Technique consists of categorizing the loads into sets that represent the level of load experienced by the resource. This proposed technique classifies resources, ensuring a quick convergence speed. FT assigns each load to a set and then searches for a predefined rule that best matches the pair of resources.

The technique presents six predefined sets, which classify overloaded and underloaded resources according to their load conditions: *extreme*, *normal*, and *low*. Based on these sets, the rules in Table II are introduced to achieve the goal of quickly converging to reach balance. This goal is reached by migrating more than one federate in case there is a large load difference between a pair of overloaded and underloaded resources.

TABLE II
FUZZY TECHNIQUE RULE SET

Source	Destination	Migration
extremely overloaded	extremely underloaded	2 federates
extremely overloaded	underloaded	1 federate
extremely overloaded	slightly underloaded	no migration
overloaded	extremely underloaded	1 federate
overloaded	underloaded	1 federate
overloaded	slightly underloaded	no migration
slightly overloaded	extremely underloaded	1 federate
slightly overloaded	underloaded	1 federate
slightly overloaded	slightly underloaded	no migration

At each balancing cycle, FT creates three equal sets for each status that ranges from the min/max to the balanced region. A loop starts by selecting candidates and assigning each of them to the respective category. FA then evaluates the appropriate pair-matches against the predefined logic rules. The matched rule determines the decision for migration, as well as the number of federates transferred.

D. Impact Factor Technique (IFT)

All the previous described techniques, RR, DR, and FT, as well as the method used in the original system, remove the candidates from the candidate list once a migration is set. When resources are removed from the candidate list, they are not considered in any other balancing analysis. These removed resources can be combined in more useful pair-match combinations, exploring other estimated prediction values. Thus, Impact Factor Technique is designed to propose the analysis of possible migrations and issue those that bring the most benefit.

The IFT identifies all possible migration scenarios in which resource may be involved and triggers the migrations that prove the value of the benefit of balancing the distributed load. In order to identify the value of each possible pair-match, IFT computes an *impact factor* for each possible load migration. This factor is represented by Equation 9.

$$Impact = \frac{load_{src} - \bar{l}}{load_{dst}} \tag{9}$$

where $\bar{l}$ represents the average load of resources.

The mean is subtracted from the source load to bring the source and destination loads closer together. The impact factor's value represents the degree of impact of the migration on the shared environment. The importance of triggering this migration increases with a higher value.

This technique builds all the possible migrations for short-term projections. Possible migrations are presented to the system as a unidirectional graph, from an overloaded source node to an underloaded destination node, and the impact factor as the weight of the edge. Then, the same process is performed with medium- and long-term projections, in this order, adding their migrations to the graph. Once all possible migrations are computed, IFT goes through the graph and first performs the migration with the highest impact factor. After these high-impact migrations are selected, the involved candidates and

TABLE III
PARAMETERS OF THE PREDICTIVE BALANCING SYSTEM

Parameter	Value
Balancing cycle	20 seconds
Holt's	α=0.36, β=0.36
Separate Systems - SP	α=0.85, β=0.36
Separate Systems - MP	α=0.08, β=0.09
Separate Systems - LP	α=0.05, β=0.05

respective migration edges are removed from the graph. The process continues until there are no more edges in the graph. The remaining resources are nominated as candidates for the next balancing cycle.

VI. EVALUATION OF THE VARIANTS AND TECHNIQUES

In order to evaluate the benefit that the modifications introduced into the predictive balancing system, some analyses and experiments have been conducted individually for each of the parts for which approaches have been proposed: Holt's model variants and pair match decision-making techniques. Even though conclusions can be applied to general distributed systems, these analyses have been conducted considering the load oscillations of HLA-based distributed simulations.

A. Holt's Model Variants

For the number of variants proposed in this work, a comparison method is needed to measure their reliability in giving decent projections. Therefore, an evaluation of accuracy is performed to compare their projections with the real readings of loads. The proposed variants, as well as the migration decision-making techniques, were evaluated against the gathered data samples described in Table I, after being set up according to the parameters shown in Table III. The variance in the parameters of the data samples is intended to replicate the dynamic execution of distributed simulation applications, stressing the system with low, medium and high load setups. Additionally, this section discusses the time and space complexity of each evaluation to identify the effects of the proposed enhancements and their overhead.

1) Accuracy Comparison: In the evaluation of the proposed variants, the enhancements are compared against the prediction model used in the prediction-based distributed load balancing system [11]. The evaluation process is carried out in two stages. The first stage performs the evaluation of the prediction techniques on data samples, generated by running several distributed simulations on top of a set of shared resources. A systematic computational load was used in the system to emulate simulation overhead. The aim of the first stage is to test the efficiency of the methods and to properly configure them for real simulations. The second stage is presented in the evaluation section, in which one variant is implemented on a real dynamic load balancing system.

An extensive analysis of effectiveness is first realized in order to evaluate the performance of the GAV variant; the analysis allows us to identify the combination of iterations and chromosomes that fits our specific case. An accuracy test of different configurations is conducted to compare the error

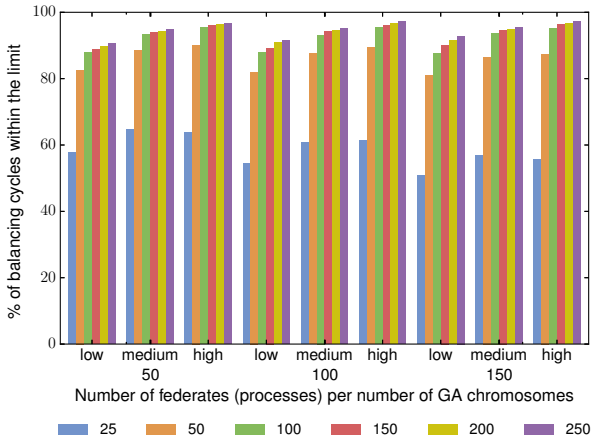


Fig. 4. Needed iterations per balancing cycle

for each computed projection in our different federate setups, which is discussed in the previous section.

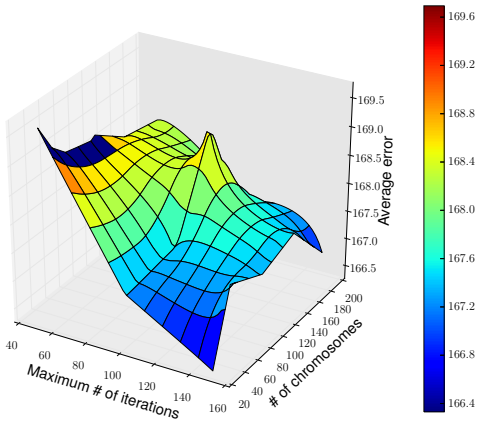
There is no simple and direct rule for defining the needed number of chromosomes, so three different numbers of chromosomes (50, 100, and 150) were chosen for the comparisons. Also, finding a suitable number of iterations requires extensive analysis to understand their distribution. Thus, GAV follows a greedy approach and stops either when the target is reached (error equals to 0) or the system cannot find a better combination of α and β for a consecutive 50 iterations.

The initial goal was to begin with a maximum number of iterations that could allow finding the suitable α and β for at least 50% of the balancing cycles, without hitting the limits. Figure 4 shows the percentage of balancing cycles in which the system could find the best chromosome without reaching the limit. GAV could find the best chromosome for 64% of the balancing cycles when the maximum number of iterations is set to 25. Changing the maximum number of iterations to 50 increased the coverage percentage from 64% to 88%. However, increasing the number of chromosomes to more than 50 does not have a noticeable effect on the total coverage.

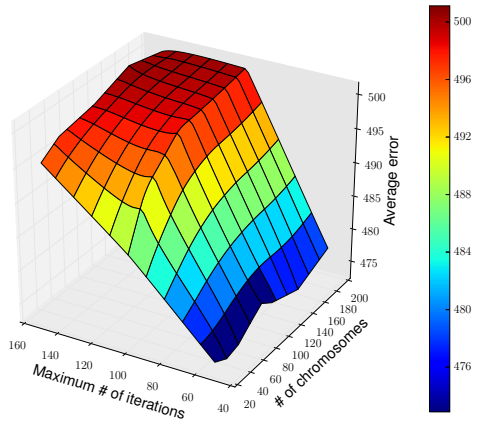
Figure 5 shows 2 samples of the average error, which occurs when SP is calculated with a medium number of federates and MP is calculated with a high number of federates. Because of the importance given to SP calculations, GAV provides the lowest error for SP even though this does not assure the lowest errors for MP. The number of chromosomes and iterations are chosen so that a reasonable error is reached with low time and space complexities. Through this analysis, the best combination for setting up our GAV was 50 chromosomes and 50 iterations.

To find α and β that result in a low error is essential for all variants. Therefore, a grid search was performed in which a set of possible values for α and β, $0 \leq [\alpha, \beta] \leq 1$, with a step of 0.01, was thoroughly compared, and the combination with the smallest value of computed errors was chosen. The predictions were extensively evaluated and compared using the calculated error. The error was obtained from the difference between the predicted and real load values. Table IV shows the rounded average error for Holt's model and its variants.

For the unmodified Holt's model, the grid search returns $\alpha = 0.36$ and $\beta = 0.36$ as a combination with the



(a) Short-term migration with medium number of federates



(b) Medium-term migration with high number of federates

Fig. 5. The average error per configuration of the Genetic Algorithm Variant

TABLE IV
AVERAGE ERROR PER VARIANT

	Holt's	Cutoff	RTV	SSV	SSRT	GAV
SP	86	88	55	31	26	97
MP	193	185	181	177	142	251
LP	217	199	205	175	151	302

TABLE V
BREAKDOWN OF LP IN CUTOFF AND RT

Frequency	Cutoff- Avg Err	Cutoff-StD	RT-Avg Err	RT-StD
Low	65.67	29.64	71.70	32.21
Medium	180.15	42.97	198.43	42.02
High	351.97	36.27	343.58	38.23

lowest error. As Cutoff and RTV present characteristics that match the features of Holt's model, the same configuration values were used to set up their parameters.

A performance comparison between Holt's model and Cutoff has been conducted in a series of experiments. The Cutoff limits the medium- and long-term projections, as shown in Figure 2, so it has set the projections to a threshold instead of to unrealistic values. The results show a decrease in the error rate of medium and long projections, as shown in Table IV. This limitation does not show a noticeable decrease as resources usually reach their peak load in the first few balancing cycles. However, this method decreased the precision of short-term projections as it restricted the projections with a threshold.

Analysis has been conducted to evaluate Reversed Trend, and results are shown in Table IV. Reversed Trend improves the response speed to load oscillations for short- and medium-term projections when compared to Holt's model. Figure 3 demonstrates the fast response of RTV to oscillations against Holt's model for short-term projections. However, the resulting effect of *sharpness* added by Reverse Trend influences the precision of long-term predictions. The *sharpness* effect adds more accuracy to the predicted value for more frequent load oscillations, which is described in Table V as the Reverse Trend provides less error for long-term projections with a high number of migrations.

A grid search was performed on each system in the Separate Systems technique. By analyzing the errors for each combination of Holt's α and β, $(\alpha = 0.85, \beta = 0.36)$ provides the lowest error for SP. Similarly, $(\alpha = 0.08, \beta = 0.09)$ and $(\alpha = 0.05, \beta = 0.05)$ return the lowest error for MP and LP, respectively. Table IV shows that by having a separate system for each projection, improved precision is achieved. Figure 6 demonstrates that the Separate Systems technique offers greater precision than Holt's model.

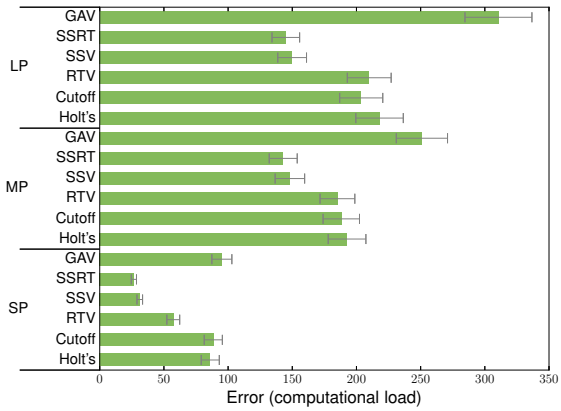
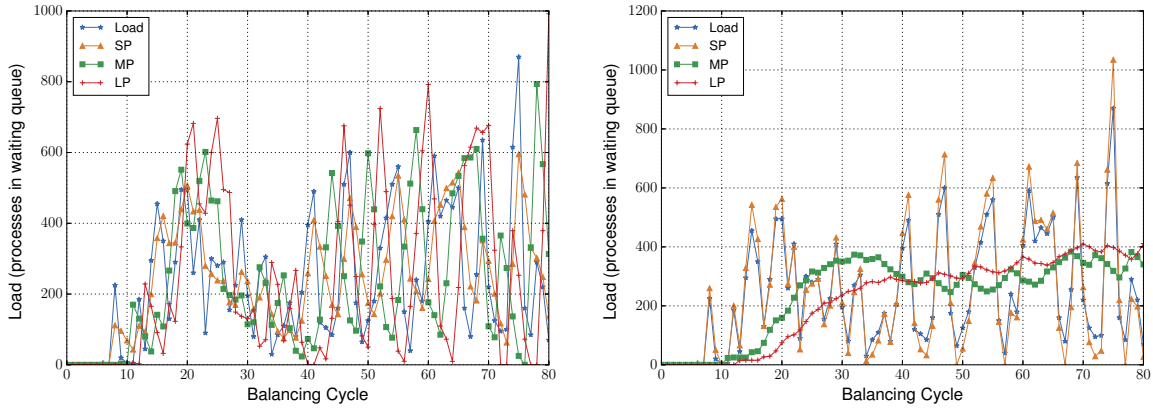


Fig. 7. Average error with 95% confidence interval for each variant

By merging the SSV and RTV techniques, and thus generating a Separate Systems Reverse Trend (SSRT), prediction performance is expected to improve. RTV adds an element of sharpness and quick reaction to sudden oscillations. On the other hand, SSV increases the precision of MP and LP. Table IV shows that the accuracy of the new variant has been improved for all predictions.

GAV's results are proof that not all extensive computations would provide better performance. GAV changes parameters continuously, affecting load projections. The time required to adapt to the frequent oscillations of the load is the main flaw in this technique. Thus, sudden load changes cause unrealistic projections for MP and LP, and indeed, this is the main cause of poor GAV performance.

Each bar in Figure 7 represents the average error of each solution, with a 95% confidence interval. Cutoff indicates worse performance by Holt's model when compared with the other variants for short-term predictions; RTV, on the other hand, performs the worst with long-term predictions.



(a) Holt's model

(b) Separate Systems (SS)

Fig. 6. The three different projections computed by each Holt's model and SSV

This shows that smoothed values work best for long-term predictions with a low and medium number of federates. However, the sharpness adds value to long-term prediction when the number of federates is high, with frequent load oscillations. The isolated systems in SSV improve predictions as each projection is given a different treatment. These different parameters show that combining two variants provides better efficiency, as merging SSV with RTV presents the lowest projection errors.

2) Complexity Comparison: As we are dealing with a balancing system that is time-dependent, a complexity analysis is needed in order to know which variant fits better than the rest in the deployment of the real system. In order to fairly analyze them, only the complexity of the *prediction model* is presented; the rest of the balancing system is untouched.

Let n be the number of resources managed by all CLBs. The number of times *smoothed value* and *trend* are calculated per *balancing cycle* is shown in Table VI.

TABLE VI
TIME AND SPACE COMPLEXITY FOR THE PROPOSED VARIANTS

Variant	Time Complexity	Space Complexity
Holt model	$\theta(2n)$	$\theta(2n)$
Cut off	$\theta(6n)$	$\theta(3n)$
RTV	$\theta(4n)$	$\theta(2n)$
SSV	$\theta(6n)$	$\theta(6n)$
SSRT	$\theta(12n)$	$\theta(6n)$
GAV	$\theta(35050n)$	$\theta(100n)$

* Note that all complexities are in the order of $O(n)$

Holt's model calculates each of the *smoothed value* and *trend* once, so there are only two calculations. Thus, the complexity is $\theta(2)$ per resource. As we have n resources, the total added time complexity is $\theta(2n)$ [1]. For space complexity, Holt's model requires 2 spaces in memory, one for the previous smoothed value and the second for the previously computed trend. Thus, the space complexity is $\theta(2n)$.

Cutoff performs the same calculations as Holt's model. However, Cutoff performs four additional operations per balancing cycle: comparison of the threshold against SP, MP, and LP, and setting the threshold. Thus, per balancing cycle, Cutoff performs $\theta(6n)$. Cutoff requires one more space in the memory to store the maximum threshold, so the trend results in $\theta(3n)$ space complexity per resource.

RTV reverses the sign of the tendencies when they are different. This requires comparisons and updates, which leads to a complexity of $\theta(4n)$. However, it does not need any more space in memory to perform its task. Thus, the space complexity is $\theta(2n)$.

For SSV, a different prediction model exists to calculate each projection, resulting in a complexity of $\theta(6n)$ [2]. The space complexity is higher than the others because it must keep different historical smoothed values and trends in memory. Thus, SSV needs three times the space complexity of Holt's model, giving a space complexity of $\theta(6n)$.

SSRT combines SSV with RTV, so the complexity is expected to be higher. The SSV part in SSRT performs $\theta(6n)$ to compute the projections. RTV needs comparisons and modifications on the trend, so this produces a final complexity of $\theta(12n)$ [3]. The space complexity, however, is the same as SSV since RTV does not have any space requirements.

The GAV model uses 50 chromosomes in 50 iterations each time it calculates the smoothed value and trend, giving 5000 [4] times. With 50 chromosomes, 50 iterations, and 12 bit per chromosome, mutation adds up 30000 [5] operations per resource on the complexity. Defining the children for the next cycle can be reduced to $\theta(50)$ [6]. Thus, GAV is a demanding approach in terms of computing power and requires significant space to execute.

B. Pair Match Decision-making Techniques

This subsection presents the results of an analysis of the pair match decision-making techniques proposed for the predictive load balancing system. This analysis is separated into two parts: the first shows an evaluation of efficiency and performance while the second describes an analysis of the complexity of the techniques used.

[1]Note that all following complexities are in the order of $O(n)$.

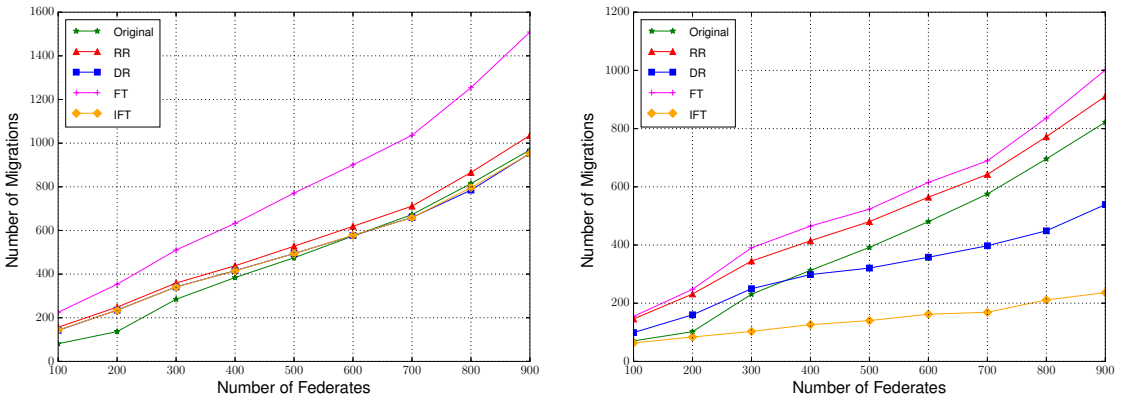
[2]Where the projection is calculated three times, giving $\theta(3 \cdot 2n)$.

[3]Where the complexity is originated from $\theta(6 \cdot 2n)$.

[4]Where the value is obtained from $2 \cdot 50 \cdot 50 = 5000$.

[5]Where the value is obtained from $50 * 50 * 12 = 30000$.

[6]Whereas $\theta(chromosomes) = \theta(50)$.



(a) Total migrations

(b) Short-term Prediction migrations

(c) Medium-term Prediction migrations

(d) Long-term Prediction migrations

Fig. 8. Efficiency analysis of the proposed migration decision-making techniques (number of migrations per approach)

1) Triggered Migrations: For the efficiency analysis, the load balancing system has been modeled, and real data has been gathered containing an average of 10 runs per setup observed in the previous subsection. The results allowed us to study the behavior of the proposed techniques and modify them accordingly. The examined behavior is the distribution of the migrations triggered per projection in a single balancing cycle. Figure 8 shows the behavior for such output per each proposed migration decision-making techniques.

In the results, FT shows an excessive number of migrations compared to the other techniques due to its candidate classification. In the defined FT rules, the first rule allows the transfer of two federates in a pair match for fast convergence, but this does not happen to the other techniques. Moreover, the FT rule that triggers two migrations is often applied because of the way FT is implemented: classifying resources as *extremely overloaded* and *extremely underloaded* based on the *current* loads instead of the *overall* load of the system. Thus, FT creates the extremely overloaded set even when the load is not considered extremely overloaded. Figure 9 shows the number of hits per rule for two load stress setups: 200 and 800 federates. For all of the analyzed series, the hits for the first rule is the highest. The first rule is a recurrent match, triggering two migrations and thus justifying the high number of migrations.

By comparing the results of the techniques with the results of the original system in Figure 8, RR produces more SP

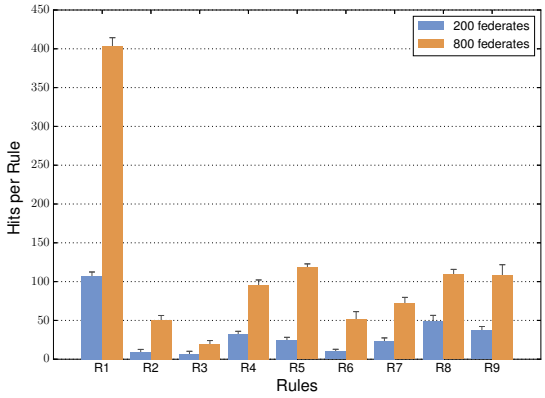
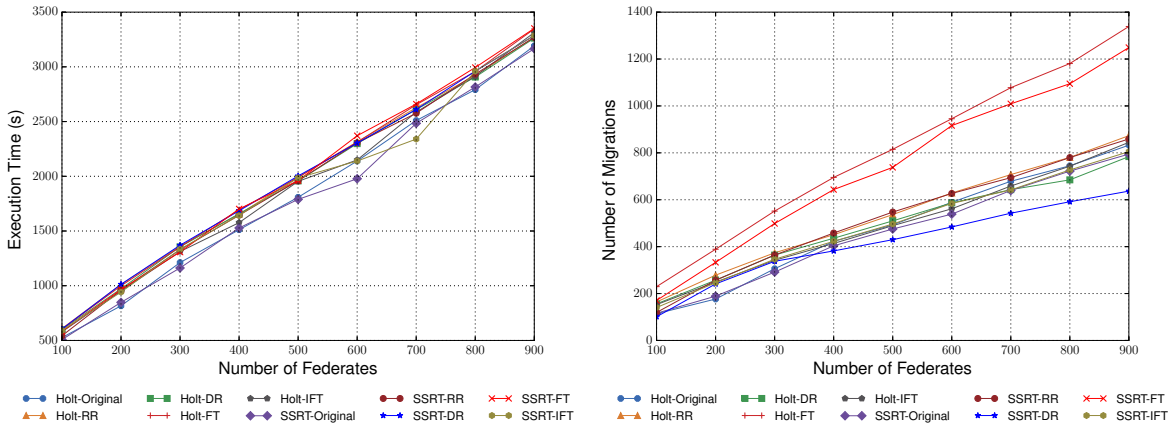


Fig. 9. Evaluation of the Fuzzy Logic technique: number of hits per each defined rule (rule ordering follows the ordering of Table II)

migrations. This is justified by the loose rules we have defined in RR. Because of the high number of migrations initiated in SP phase, fewer candidates are sent to be processed in MP phase. However, running simulations with more federates results in poor consistency, where the looseness is taken advantage from and RR starts producing more migrations than the original system. After this stage, RR has consumed most of its candidates' results in low migrations initiated in LP.

DR applied more restrictions that resulted in reducing the number of migrations in SP. This allowed for more candidates to be sent for its MP phase. At this stage, DR has provided the highest number of candidates in MP phase, producing more



(a) Execution time

(b) Number of migrations

Fig. 10. Performance analysis of the proposed variants and migration decision-making techniques

MP migrations than the other techniques. These restrictions also resulted in more candidates being sent to LP.

IFT shows the lowest number of migrations triggered in SP. The SP presents the lowest projections among the rest (MP and LP). As a result, the impact factor computed for SP is usually the smallest and is highest for LP. IFT assigns more importance to migrations with the highest impact factor first. Therefore, the number of migrations triggered in LP is the highest, followed by MP, and then SP. This shows that the migrations are issued in a manner in which LP migrations are given more importance than MP, which is higher than SP.

Because of the implementation of its rules, FT produces the highest number of migrations in SP. This suggests the implication of fewer possibilities of matching candidates for MP and LP migration. This is shown in the figure through the lower number of triggered MP and LP migrations.

2) Complexity Comparison: A complexity analysis has been performed to identify the complexity of each proposed variant and technique. In this analysis, the major features of the techniques are observed to retrieve their complexities.

In terms of processing, the technique used in the original system, RR, and DR have identical structure and functionality. They are different in the conditional statement the system adopts to verify if a migration is needed. Therefore, RR and DR do not introduce any major overhead.

FT follows every step in the original system, but it goes over two additional phases that introduce a slight overhead. FT needs to perform a one-time $\theta(n)$ evaluation to cover all loads. Once the source and the destination are assigned to sets, FT needs $\theta(9)$ per node to go over the rules, resulting in $\theta(9n)$. Thus, these two phases generate an overhead with a time complexity of $\theta(10n)$ per balancing cycle. In terms of space complexity, the extra space required by FT is a 6-long array. This storage is shared among all resources, concluding an extra $\theta(6)$ storage needed per cycle.

IFT is the most complex since it considers all migration possibilities. The worst case scenario consists of the resources being evenly divided into overloaded and underloaded, so the maximum number of possible migrations is $\theta((\frac{n}{2})^2)$, each resource presenting a connection to, at most, half of the other resources. Since this process is initiated for each projection phase (SP, MP, and LP), the complexity added is $\theta(3 \cdot (\frac{n}{2})^2)$. The impact factor shows the same complexity: $\theta(6 \cdot (\frac{n}{2})^2)$ [7]. Moreover, the search complexity for the maximum impact for all possible cases is described in Equation 10.

$$3 \cdot \sum_{i=0}^{n/2} (\frac{n-2i}{2})^2 = \frac{1}{32} n(n+2)(7n+1) \tag{10}$$

In the equation, the sum is multiplied by three because of the calculations for the three projections. In terms of space complexity, each possible migration must have 3 spaces to be represented: a space for the source node, another space for the destination node, and the last space for the impact. Based on the previous time complexity, the total number of migrations for the three projections is $\theta(9 \cdot (\frac{n}{2})^2)$ [8].

3) Experimental Evaluation of the Proposed Variants and Techniques: An experimental evaluation has been conducted with the proposed variants and techniques. The testbed consisted of simulation federates deployed in two clusters of computing servers. One of the clusters was composed of 23 computing servers interconnected by a gigabit Ethernet network; each server contained a Core 2 Duo 3.4GHz Intel Xeon CPU and 2GB of DIMM DDR RAM. The other cluster contained 16 computing servers interconnected by a 2-gigabit Myrinet optical network; each computing server contained a Quadcore 2.40GHz Intel Xeon CPU and 8GB of DIMM DDR RAM. The two clusters were interconnected through a Fast-Ethernet link. Linux operating system was installed on every server, Globus Toolkit 4.2.1 was set up, and HLA RTI 1.3 was used to coordinate simulations.

The balancing elements were deployed on all computing servers, and the CLBs were placed on each cluster management server. The baseline, as well as the initial configuration of each simulation, consisted of evenly deploying federates in the 40 shared resources; one of the servers was also dedicated to running the HLA RTI executive. In the simulation scenario, 1 to 900 federates controlled the movement of objects (vehicles) during 100 time steps. Limited to the influence of communication on the simulations, each federate managed

[7] Whereas $2 \cdot \theta(3 \cdot (\frac{n}{2})^2) = \theta(6 \cdot (\frac{n}{2})^2)$.
[8] Whereas $3 \cdot \theta(3 \cdot (\frac{n}{2})^2) = \theta(9 \cdot (\frac{n}{2})^2)$.

only one object and calculated the movement of its tank through highly intensive computational tasks.

As depicted in Figure 10, SSRT-Original performed the best, with fewer migrations compared to Holt-Original. In Figure 10a, the curves that show the best performance (lowest execution times) are those that represent SSRT-Original and SSRT-DR. SSRT-Original showed the best efficiency in Figure 10b, matching the expected behavior it presented in Figure 7; it performed the least number of migrations to achieve execution times similar to Holt-Original, as in Figure 10a.

Because the unrealistic projections of Holt's model can overcome DR's rules, Holt-DR an increase of execution times of up to 20% when compared to SSRT-Original. RA-dependent systems trigger more migrations than needed, resulting in a bad performance, and up to 25% increase in execution time compared to SSRT-Original. SSRT-DR results in larger execution time, with an increase of a maximum of 25% when compared to SSRT-Original but with 25% fewer migrations. Thus, SSRT-Original was the best matching combination of solutions for achieving the highest balancing efficiency.

Both Figures 10a and 10b show evidence of discrepancy. Even though there are curves that present a high increase in the number of migrations, the respective execution times are larger than the Original but not as proportional as the difference of the number of migrations. This behavior originates from the fact that migrations present negligible delays. By increasing the execution state of federates, the number of migrations would impose greater effect on the execution times.

VII. Conclusion

In this paper, a set of Holt's model variants and migration decision-making techniques have been proposed to improve a prediction-based dynamic load balancing system for distributed simulations. These proposed techniques were built to overcome the limitations of the previous predictive balancing scheme. The variants of the Holt's model comprise the use of Cutoff, Reverse Trend, Separate Systems, and Genetic Algorithms to enhance the load forecast. The migration decision-making was also modified using a Restricted Range, Direction-based decision, Fuzzy Logic, and Impact methods. Analyses have been conducted to evaluate all the proposed improvements on the balancing system. Separate System integrated with Reversed Trend showed the best improvement in predicting load oscillations, and the direction-based technique integrated with the restricted range technique presented the best balancing efficiency gain for enhancing decision-making.

In future work, we intend to implement other prediction models, linear or non-linear, observing the improvements they bring to the load predictions. Moreover, additional work will be conducted to tune up the conditions in order to obtain reasonable balancing efficiency for the Fuzzy Logic technique. Finally, a full analysis of the different types of balancing systems will be performed to evaluate the effectiveness of the proposed federate migration decision-making approaches.

Acknowledgment

This work is partially supported by Canada Research Chairs Program, NSERC, and DIVA Strategic Networks.

References

[1] P. Zhang and G. Gong, "Atmosphere-affected flight simulation system," in *Proc. of the Spring Simulation Multiconference*, 2009, pp. 43:1–43:7.

[2] P. Wu, B. Cai, D. Zhang, Z. Zhou, and Y. Chen, "Hla-based multi-aricraft combat simulation system," in *Proc. of 2nd Int. Asia Conference on Informatics in Control, Automation and Robotics*, vol. 3, 2010, pp. 331–334.

[3] T. Saber, A. Ventresque, and J. Murphy, "Rothar: Real-time on-line traffic assignment with load estimation," in *Proc. of the IEEE/ACM 17th Int. Symposium on Distributed Simulation and Real Time Applications*, 2013, pp. 79–86.

[4] J.-S. Yeom, A. Bhatele, K. Bisset, E. Bohm, A. Gupta, L. Kale, M. Marathe, D. Nikolopoulos, M. Schulz, and L. Wesolowski, "Overcoming the scalability challenges of epidemic simulations on blue waters," in *Proc. of IEEE 28th Int. Parallel and Distributed Processing Symposium*, 2014, pp. 755–764.

[5] A. Park and R. Fujimoto, "Efficient master/worker parallel discrete event simulation on metacomputing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp. 873–880, 2012.

[6] Y. Deng and R. W. Lau, "On delay adjustment for dynamic load balancing in distributed virtual environments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 4, pp. 529–537, 2012.

[7] H. Liu and M. Bowman, "Scale virtual worlds through dynamic load balancing," in *Proc. of the IEEE/ACM 14th Int. Symposium on Distributed Simulation and Real Time Applications*, 2010, pp. 43–52.

[8] X. Xu and G. Li, "A management and control infrastructure for integrated real-time simulation environment," in *Proc. of the IEEE/ACM 17th Int. Symposium on Distributed Simulation and Real Time Applications*, 2013, pp. 197–204.

[9] R. De Grande and A. Boukerche, "Distributed dynamic balancing of communication load for large-scale hla-based simulations," in *Proceddings of IEEE Symposium on Computers and Communications*, 2010, pp. 1109–1114.

[10] ——, "Dynamic load redistribution based on migration latency analysis for distributed virtual simulations," in *Proc. of IEEE Int. Workshop on Haptic Audio Visual Environments and Games*, 2011, pp. 88–93.

[11] ——, "Predictive dynamic load balancing for large-scale hla-based simulations," in *Proc. of IEEE/ACM 15th Int. Symposium on Distributed Simulation and Real Time Applications*, 2011, pp. 4–11.

[12] R. Alkharboush, "Improved prediction-based dynamic load balancing systems for hla-based distributed simulations," Master's thesis, EECS - University of Ottawa, 2015.

[13] R. Alkharboush, R. De Grande, and A. Boukerche, "Load prediction in hla-based distributed simulation using holt's variants," in *Proc. of IEEE/ACM Int. Symposium on Distributed Simulation and Real Time Applications*, 2013, pp. 161–168.

[14] ——, "Federate migration decision-making methods for hla-based distributed simulations," in *Proc. of IEEE/ACM Int. Symposium on Distributed Simulation and Real Time Applications*, 2014, pp. 190–197.

[15] K. Bahulkar, J. Wang, N. Abu-Ghazaleh, and D. Ponomarev, "Partitioning on dynamic behavior for parallel discrete event simulation," in *Proc. of ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation*, 2012, pp. 221–230.

[16] D. Jagtap, N. Abu-Ghazaleh, and D. Ponomarev, "Optimization of parallel discrete event simulator for multi-core systems," in *Proc. of IEEE Int. Parallel Distributed Processing Symposium*, 2012, pp. 520–531.

[17] A. Boukerche and S. K. Das, "Reducing null messages overhead through load balancing in conservative distributed simulation systems," *J. Parallel Distrib. Comput.*, vol. 64, no. 3, pp. 330–344, 2004.

[18] L. Bononi, M. Bracuto, G. D'Angelo, and L. Donatiello, "A new adaptive middleware for parallel and distributed simulation of dynamically interacting systems," in *Proc. of IEEE Int. Symposium on Distributed Simulation and Real-Time Applications*, 2004, pp. 178–187.

[19] J. Feng, "A new method for ionospheric short-term forecast using similar-day modeling," in *Proc. of Int. Symposium on Antennas, Propagation EM Theory*, 2012, pp. 472–474.

[20] K.-L. Ho, Y.-Y. Hsu, C.-F. Chen, T.-E. Lee, C.-C. Liang, T.-S. Lai, and K.-K. Chen, "Short term load forecasting of taiwan power system using a knowledge-based expert system," *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1214–1221, 1990.

[21] S. Ruzic, A. Vuckovic, and N. Nikolic, "Weather sensitive method for short term load forecasting in electric power utility of serbia," *IEEE Transactions on Power Systems*, vol. 18, no. 4, pp. 1581–1586, 2003.

[22] R. F. Engle, C. Mustafa, and J. Rice, "Modelling peak electricity demand," *Journal of Forecasting*, vol. 11, no. 3, pp. 241–251, 1992.

[23] A. Bakirtzis, V. Petridis, S. Kiartzis, M. Alexiadis, and A. Maissis, "A neural network short term load forecasting model for the greek power system," *Power Systems, IEEE Transactions on*, vol. 11, no. 2, pp. 858–863, May 1996.
[24] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York, NY, USA: Springer-Verlag New York, Inc., 1999.
[25] M. Mohandes, "Support vector machines for short-term electrical load forecasting," *Int. Journal of Energy Research*, vol. 26, no. 4, pp. 335–345, 2002.
[26] V. Miranda and C. Monteiro, "Fuzzy inference in spatial load forecasting," in *Proc. of IEEE Power Engineering Society Winter Meeting*, vol. 2, 2000, pp. 1063–1068.
[27] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*, 2nd ed. New York, USA: Springer, 2009.
[28] K. Liu, S. Subbarayan, R. Shoults, M. Manry, C. Kwan, F. Lewis, and J. Naccarino, "Comparison of very short-term load forecasting techniques," *IEEE Transactions on Power Systems*, vol. 11, no. 2, pp. 877–882, 1996.
[29] J. W. Taylor, L. M. de Menezes, and P. E. McSharry, "A comparison of univariate methods for forecasting electricity demand up to a day ahead," *Int. Journal of Forecasting*, vol. 22, no. 1, pp. 1 – 16, 2006.
[30] J. Taylor and P. McSharry, "Short-term load forecasting methods: An evaluation based on european data," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 2213–2219, 2007.
[31] S. Abdullah, N. Sapii, S. Dir, and T. Jalal, "Application of univariate forecasting models of tuberculosis cases in kelantan," in *Proc. of Int. Conference on Statistics in Science, Business, and Engineering*, 2012, pp. 1–7.
[32] L. Wu, J. Yan, and Y. Fan, "Data mining algorithms and statistical analysis for sales data forecast," in *Proc. of Int. Joint Conference on Computational Sciences and Optimization*, 2012, pp. 577–581.
[33] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, Aug. 2001.
[34] A. Boukerche and R. Grande, "Optimized federate migration for large-scale hla-based simulations," in *Proc. of IEEE/ACM Int. Symposium on Distributed Simulation and Real-Time Applications*, 2008, pp. 227–235.
[35] E. S. G. Jr., "Exponential smoothing: The state of the art - part ii," *Int. Journal of Forecasting*, vol. 22, no. 4, pp. 637 – 666, 2006.

Robson Eduardo De Grande is currently the Network Manager of the NSERC DIVA Strategic Research Network and a Senior Research Associate at the School of Electrical Engineering and Computer Science, University of Ottawa, Canada. He received his Ph.D. degree at the University of Ottawa in 2012 and his M.Sc. and B.Sc. degrees in Computer Science from the Federal University of Sao Carlos, Brazil in 2006 and 2004, respectively. He served as TPC Chair of IEEE/ACM DS-RT 2014. His research interests include large-scale distributed and mobile systems, Cloud computing, high performance computing, performance modeling and simulation, computer networks, vehicular networks, and distributed simulation systems.

Raed Alkharboush received his B.Sc. in Computer Engineering from King Saud University in 2008 and M.Sc. in ECE from the University of Ottawa, Canada in 2014. He is now a Computer Operating System Specialist at Saudi Aramco. His research interests include distributed computing, cloud computing, and performance evaluation.

Azzedine Boukerche (FIEEE, FEiC, FCAE, FAAAS) is a full professor and holds a Canada Research Chair position at the University of Ottawa (Ottawa). He is the founding director of the PARADISE Research Laboratory, School of Electrical Engineering and Computer Science (EECS), Ottawa. Prior to this, he held a faculty position at the University of North Texas, and he was a senior scientist at the Simulation Sciences Division, Metron Corp., San Diego. He was also employed as a faculty member in the School of Computer Science, McGill University, and taught at the Polytechnic of Montreal. He spent an year at the JPL/NASA-California Institute of Technology, where he contributed to a project centered on specification and verification of the software used to control interplanetary spacecraft operated by JPL/NASA Laboratory. His current research interests include Wireless Ad Hoc and sensor networks, wireless networks, mobile and pervasive computing, wireless multimedia, QoS service provisioning, performance evaluation and modeling of large-scale distributed systems, distributed computing, large-scale distributed interactive simulation, and parallel discrete-event simulation. He has published several research papers in these areas. He served as a guest editor for the Journal of Parallel and Distributed Computing (special issue for routing for mobile ad hoc, special issue for wireless communication and mobile computing, and special issue for mobile ad hoc networking and computing), ACM/Kluwer Wireless Networks, ACM/Kluwer Mobile Networks Applications, and Journal of Wireless Communication and Mobile Computing. He has been serving as an Associate Editor of ACM Computing Surveys, IEEE Transactions on Parallel and Distributed systems, IEEE Transactions on Vehicular Technology, Elsevier Ad Hoc Networks, Wiley International Journal of Wireless Communication and Mobile Computing, Wileys Security and Communication Network Journal, Elsevier Pervasive and Mobile Computing Journal, IEEE Wireless Communication Magazine, Elseviers Journal of Parallel and Distributed Computing, and SCS Transactions on Simulation. He was the recipient of the Best Research Paper Award at IEEE/ACM PADS 1997, ACM MobiWac 2006, ICC 2008, ICC 2009 and IWCMC 2009, and the recipient of the Third National Award for Telecommunication Software in 1999 for his work on a distributed security systems on mobile phone operations. He has been nominated for the Best Paper Award at the IEEE/ACM PADS 1999 and ACM MSWiM 2001. He is a recipient of an Ontario Early Research Excellence Award (previously known as Premier of Ontario Research Excellence Award), Ontario Distinguished Researcher Award, Glinski Research Excellence Award, IEEE CS Golden Core Award, IEEE Canada Gotlieb Medal Award, IEEE ComSoc Expectional Leadership Award, IEEE TCPP Excpetional Leadership Award. He is a cofounder of the QShine International Conference on Quality of Service for Wireless/Wired Heterogeneous Networks (QShine 2004). He served as the general chair for the Eighth ACM/IEEE Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, and the Ninth ACM/IEEE Symposium on Distributed Simulation and Real-Time Application (DSRT), the program chair for the ACM Workshop on QoS and Security for Wireless and Mobile Networks, ACM/IFIPS Europar 2002 Conference, IEEE/SCS Annual Simulation Symposium (ANNS 2002), ACM WWW 2002, IEEE MWCN 2002, IEEE/ACM MASCOTS 2002, IEEE Wireless Local Networks WLN 0304; IEEE WMAN 0405, and ACM MSWiM 9899, and a TPC member of numerous IEEE and ACM sponsored conferences. He served as the vice general chair for the Third IEEE Distributed Computing for Sensor Networks (DCOSS) Conference in 2007, as the program cochair for GLOBECOM 20072008 Symposium on Wireless Ad Hoc and Sensor Networks, and for the 14th IEEE ISCC 2009 Symposium on Computer and Communication Symposium, and as the finance chair for ACM Multimedia 2008. He also serves as a Steering Committee chair for the ACM Modeling, Analysis and Simulation for Wireless and Mobile Systems Conference, the ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, and IEEE/ACM DSRT.