A Genetic Algorithm Approach for Adjusting Time Series Based Load Prediction †

Raed Alkharboush
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
Email: ralkh092@uottawa.ca

Robson E. De Grande
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
Email: rdgrande@site.uottawa.ca

Azzedine Boukerche
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
Email: boukerch@site.uottawa.ca

***Abstract*—Distributed virtual simulation are prone to load oscillations, as well as load imbalances during run-time. Detecting such imbalances and responding accordingly using load redistribution can be of great utility in keeping execution performance close to the aimed optimal. A dynamic balancing scheme can introduce a reactive approach, but a predictive scheme can prevent imbalances before they occur. Several models can be employed for predicting load, but due to the characteristics in which the load is collected and presented, time series offer reasonable load forecasting in a short time. However, the Holt's model, well known model for time series representation, shows limitations on the forecasting of load. In order to correct this issue, a genetic algorithm approach is introduced to dynamically adjust the model based on the recent modifications on the load behaviour. The convergence of the algorithm can substantially influence the response time of the predictive balancing system, so an analysis is conducted to identify the minimum number of iterations for generating a reasonable adjustment.**

I. Introduction

Distributed virtual simulations have been receiving growing attention due to their substantial potential on aiding the analysis and evaluation of complex systems, training, testing, and entertainment. Dynamic oscillations on the load can generate imbalances and negatively impact on the performance of such simulations. Many platforms can be used for the design and implementation of distributed virtual simulations, and the High Level Architecture (HLA) is commonly used to provide a design framework and methods to facilitate the creation and coordination of distributed simulations. Aiming at re-usability and interoperability aspects, it enforces the reuse of simulation entities, federates, and promotes the transparent communication of parts across multiple platforms. The framework is essentially structured on a set of design rules, object model templates, and an interface specification. Management services are provided through the Run-time Infrastructure (RTI) component to maintain the consistency of HLA-compliant simulations. Federates access the services to inter-communicate and progress with simulation processing. However, HLA only comprises the management of simulation aspects, and it is not able to prevent performance loss arising from load imbalances.

† This work is partially supported by NSERC Canada Research Chairs Program and ORF/MRI Research Funds.

Dynamic load balancing systems have been developed to cope with the issues load oscillations generate. Such systems are basically concerned with either communication or computation load of simulations. In order to modify the distribution of load during run-time, these systems need to monitor the environment and the simulation constantly, and they react upon the identification of certain load behaviour characteristics on the collected data sample. Even though these balancing systems effectively react to imbalances and correct load distribution, they are incapable of preventively reacting to load oscillations. Since such systems are reactive, some performance loss has already been present before it is detected. A preventive form of reaction, when possible, can avoid imbalances and losses before they occur or with minimal consequences on the performance.

A predictive balancing system [1] has been proposed to add analysis of load forecasting on top of the already existing dynamic load balancing system. Consequently, some prevention is offered based on the load tendencies as it is detected based on the most recent collected data samples. In such balancing system, a prediction model is used to map and identify the load oscillations, making use of time series and Holt's model for the load forecasting. However, the model presents some limitations closely connected to its linear method to calculate load forecasts.

Proposed solution using genetic algorithm is introduced to cope with some of the limitations of the Holt's model. Since these limitations are generated by the static linear projections that are used on the forecasting, dynamic adjustments are applied to modify the model as the behaviour of the load changes. The genetic algorithm is an essential tool in this approach because it provides bio-inspired heuristics that can be easily applied to produce a reasonable sub-optimal answer in a feasible short time. The converge of the genetic algorithm is critical for providing the answer in the short time frame required by the balancing system.

The remainder of this paper is organized as follows. Section 2 shows the related work, which delineates the existing prediction models that potentially can be used for load forecasting. Section 3 describes a prediction based load balancing scheme, which is used as a the basis for implementing a time-series

prediction model. Section 4 introduces the proposed genetic algorithm approach to dynamically modify the prediction model, as well as the predictive balancing system. Section 5 presents the analysis that have been conducted to the evaluated the best conditions in which the genetic algorithm needs to be setup for a better matching with restrictions on time and precision. Finally, Section 6 concludes the paper and provides directions for future work.

II. Related Work

Accurate and precise load prediction techniques are important for the efficiency of load balancing systems. Load prediction helps such systems to migrate federates among resources to have a better balanced shared environment in a preventive fashion. This shows the importance to look at the different prediction models [2] in the distributed simulations field and other fields in order to understand the possible models we can use.

A. Similar day Approach

Similar day is a method that searches historical data looking for similar characteristics to what the system is going through at the moment to predict the load. The historical data has to be seasonal where the characteristics of the historical data repeat themselves. The prediction computed by this approach can be the a single similar reading or a linear combination of several historical readings. Feng [3] has used the concept of similar day in his research to model a system for ionospheric forecasting. Implementing this approach is not considered challenging once historical data is available. In our experiments, historical data can be gathered as the simulation is running. However, the load of distributed simulation does not follow a seasonal pattern. The load does not follow any pattern. Thus using this approach is not suitable because of the differences of their data behaviour.

B. Expert Systems

This is a rule-based intelligence system. The rules are heuristic in nature and are incorporated by experts in the field to a list of statements in produce prediction without any human involvement. This however raises a set of disadvantages of such prediction methods: the quality of output based on the proper expert input. This technique has been used in a different number of fields. Ho *et al.* [4] have employed the knowledge of the operators and the system load of the Taiwan Power system to propose a knowledge-based expert system. Embedding an expert system is not preferred in predicting the load of resources that run distributed simulations. The behaviour of the load in each time the simulation runs is different. This makes it difficult for experts to recognize a rule that unifies the behaviour of all possible scenarios in order to produce a precise projection.

C. Regression methods

Regression is used to find the relationships between a dependent variable and a set of independent variables. This prediction method is the most widely used in the electrical load forecasting [5] [6] [7] [8]. It is used to find a model that represents the relationship of load consumption and other factors, such as weather, season, and the type of home. Engle *et al.* [9] presented different regression models to forecast the load of the next day. The models proposed incorporated the changes in weather, season of a holiday, and the average load. Implementing regression models in our systems is not feasible as the only visible variable is the system's load. Building a relationship between two different elements is impossible.

D. Artificial Neural Network

Artificial Neural Network (ANN) has been one of the most widely used forecasting techniques in many fields, especially electric load forecasting. Neural Network have capabilities to demonstrate non-linear fitting. Bakirtzis *et al.* [10] implemented a neural network system for load projection for the Energy Control Center of the Greek Public Power Corporation. To predict loads for distributed systems, historical data is needed to better project future loads. However, casual artificial neural network do not provide meanings to use historical data. Tapped delay is an approach where previous historical data are fed to the neural network along the current data. On the other hand, Recurrent Neural Network (RNN) uses historical data with neural network. RNN is capable of recognizing sequences that are unrecognisable by ANN; this makes RNN suitable for pattern recognition applications.

E. Support Vector Machine

Support Vector Machine (SVM) is a recent technique to solve classification and regression problems. It was inspired from Vapniks statistical learning theory [11]. SVM nonlinearly maps the data into a higher dimensional space. SVM then employs linear functions to create boundaries in the new space. Choosing a suitable kernel is an issue that is usually raised when using SVM [12]. Mohandes [13] used SVM for projection electrical load in the short-term future. Mohandes compared the performance of SVM against autoregressive methods and the conclusion is that SVM was favoured over the autoregressive model. However, it needed training data to perform well.

F. Fuzzy Logic

Fuzzy logic is an approximation approach where 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 that maps inputs to outputs, giving a robust forecasting. A defuzzification process can be used to generate a precise output, but it needs a lot of information. Fuzzy has been used in a different number of electric load forecasting [14] [15]. This process is not feasible in the balancing system we are working on because of the low number of inputs we have, which produce few output sets that are not helpful in producing an exact output. Another issue is that the range of the inputs is needed to assign them to proper sets.

G. 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 process while Autoregressive Integrated Moving Average (ARIMA) is used with non-stationary processes. Several evaluations have been conducted in order to analyse the performance of time-series predictions in different applications.

A study [16] has been conducted to compare Autoregression model against Fuzzy logic and Neural Network. Unlike Fuzzy Logic and Neural Networks, Autoregression models do not need training, and the autoregression model provided reasonable predictions for a short period of time, in minute-by-minute predictions. A comparison of the existing approaches of univariant methods (PCA, double seasonal ARMA, double seasonal Holt-Winters, ANN) for short-term, half-hour and hourly electricity load predictions [17]. In this context, Holt-Winters had the best performance being a simple yet robust technique. An empirical study has been realized to evaluate the performance of different prediction models for up to a day-ahead for a seasonal data [18]. In this second study, the double seasonal Holt-Winters exponential smoothing has the best performance against ARIMA and PCA for half-hour predictions. An analysis has been conducted to predict the Tuberculosis cases for two years based on historical data [19]. In their evaluation setup with historical data, Holts model and Double Exponential Smoothing performed the best and showed a really small error rate. Another evaluation analysed the performance of multiple time series forecasting methods to forecast new product sales with and increasing trend [20]. In this analysis, Holts model performed better than Single Exponential Smoothing. However, traditional ARMA outperformed Holts model.

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

III. Predictive Balancing System

The modifications and enhancements of Holt's prediction technique are incorporated into the prediction-based dynamic load balancing system described in [1], which is presented in Figure 1. The load balancing process consists of monitoring the resources in order to define a set of under-loaded and over-loaded resources.

As defined in its architectural design, the balancing system works together with a prediction mechanism, and it is basically composed of a Cluster Load Balancer (CLB) and a Local Load Balancer (LLB). These two components control and enable the fundamental functions of the load balancing system. The CLB controls all the balancing actions: monitoring, load redistribution, and migration. The LLB works as a interface between the balancing system and the simulation elements.

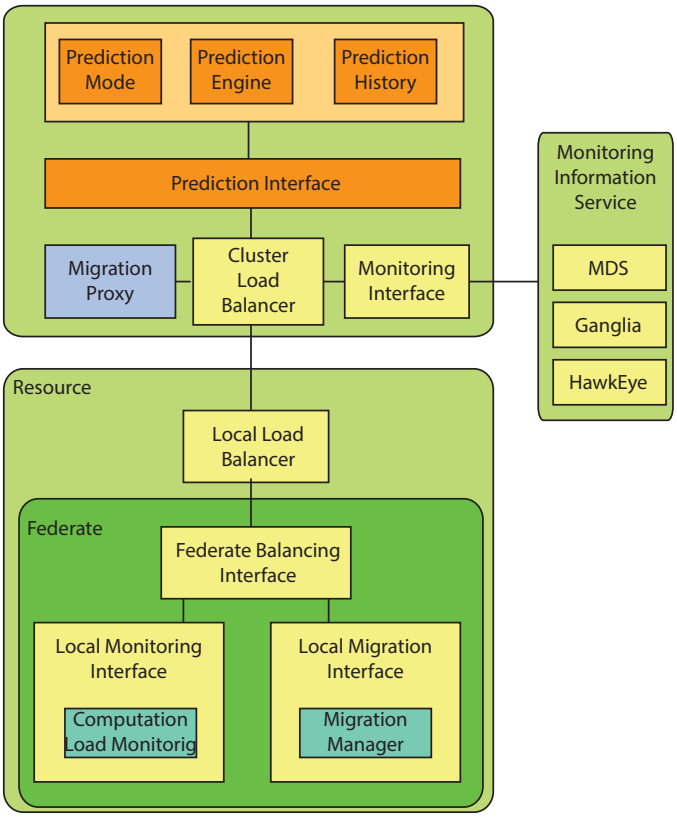


Fig. 1. General Architecture of the Simulation/Visualization System

The LLB collects load information and aggregate data through its Local Monitoring Interface; it also forwards migration calls and manages migration procedures through its Local Migration Interface. A Migration Proxy is also in the system to enable and guarantee the reachability of remote parts in large-scale environments.

In order for the balancing system to collect load data from distributed resources periodically, it accesses a Monitoring Information System through its Monitoring Interface. The interface brings up transparency on the access to the data, which can be collected through different monitoring tools.

The analysis of collect data of this balancing system is extend with predictions on load tendencies. In order for this extension to be enabled, a Prediction Model, a Prediction Engine, and a Prediction Interface are introduced in the architecture, as depicted in architectural design of the system. These components, more importantly the interface, allow modular use of different models and engines for the forecasting of load, which is used in the decision-making of the balancing system.

A. Balancing Scheme

The redistribution algorithm is used to re-arrange 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 (line 2 of Algorithm 1). This allows the system to be responsive to load changes. The Monitoring is performed periodically

Algorithm 1 Distributed Dynamic Load Balancing Algorithm

```
1: while TRUE do
2:     loads ← query_MDS()
3:     current_loads ← filter_MDS_data(loads)
4:     current_loads ← normalize_loads(current_loads, benchmark)
5:     current_loads ← prediction(current_loads, old_loads, mig_RSCs)
6:     old_loads ← current_loads
7:     overload_cand ← select_overload(current_loads)
8:     spec_loads ← request_LLBs(overload_cand)
9:     mng_loads ← filter(current_loads, spec_loads)
10:    mig_moves ← local_bal(mng_loads, SP)
11:    mig_moves ← local_bal(mng_loads, MP)
12:    mig_moves ← local_bal(mng_loads, LP)
13:    send_migration_moves(mig_moves)
14:    if mig_moves = 0 then
15:        data_neighbours ← reques_Neighbour_Load_Data()
16:    else
17:        if relFactor ≥ random_number(1, 100) then
18:            data_neighbours ← reques_Neighbour_Load_Data()
19:        else
20:            data_neihgbours ← 0
21:        end if
22:    end if
23:    neighbours ← identify_neighbour_Less_Load()
24:    while neighbours ≠ 0 do
25:        overloaded_resource ← select(firstNeighbour, SP)
26:        overloaded_resource ← select(firstNeighbour, MP)
27:        overloaded_resource ← select(firstNeighbour, LP)
28:        federates ← select(spec_loads, overloaded)
29:        eliminate_first(neighbours)
30:    end while
31:    send_to_neightbour(overloaded_resources, federates)
32:    migration_moves ← wait_for_migration_moves()
33:    send_migration_moves(migration_moves)
34:    wait(Δ)
35: end while
```

every Δt (line 34 of Algorithm 1). Restricted by the periodic refresh rate offered by the monitoring tool in this particular design, Δt is set to 20 seconds, producing minimum overhead and considerable awareness of load oscillations.

Filtering and normalization is applied on the collected data. Afterwards, the collected data, previous history, and migration status are used to provide load projection in the following three different balancing cycle ranges: short-term, medium-term, and long-term (line 5 of Algorithm 1). The load balancing system then applies a pair matching mechanism between the local over-loaded and under-loaded resources in each balancing cycle (lines $6-12$ of Algorithm 1). The load balancing system assigns a greater importance to projections that are closer to the current time (lines $10-12$ of Algorithm 1).

A CLB performs the redistribution of load first in a local scope and then in an inter-domain scope. The inter-domain balancing is needed to exchange load between clusters of resources (lines $14-32$ of Algorithm 1). For both scopes, the algorithm attempts to identify pair matches. The mechanism initiates a migration call if the difference between the load of the over-loaded and the under-loaded resources is beyond a threshold. However in the inter-domain balancing, load from the neighbouring CLBs need to be collected to identify potential destinations for load transfers. The selection of a CLB candidate is based on the average load of its resource; the average provides means to determine if there imbalances among clusters of resources. At the end of the selection process, all migration calls are sent at once to the requester CLB (lines $31-33$ of Algorithm 1).

B. Load Prediction

As previously stated, the distributed load balancing system performs on three different load forecasting levels: short, medium, and long-term. The three projections are independent from the prediction model. Each projection is designed to provide a goal to the system. 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 might happen. 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.

Due to be collected in fixed time intervals, the load data can be naturally characterized as a time series. Exponential Weighted Moving Average (EWMA) is a well-known time series prediction technique that observes the internal relationship of elements in three different aspects [21]. Because of the lack of well-defined seasonality on the load oscillations, the Double EWMa has been used as the prediction model for the balancing system. The Double EWMA adds a trend to the exponentially calculated average. The trend of predicting the load of shared resources is used to identify the tendency of the resource load.

Upon a deep analysis on this prediction method, two different weaknesses were identified with how Holt's model deals with loads similar to the load generated by running distributed simulation:

- The limitation on the load projection in a future balancing cycle has a linear relationship with the time interval of the future balancing cycle that we need to have a load projection for. By looking at the forecasting formula of Holt's model;
- The slow response of Holt's model to adapt itself to sudden load oscillations. After thoroughly analyzing the Holt's model and identifying its different variables, it was noticed that the computed trend, t_i, does not always represent, or match, the actual tendency.

IV. Genetic Algorithm

A variant is introduced the prediction model of the predictive balancing system to improve its load projection. This variant uses Genetic Algorithms to solve the second issue. The basic principle is that it adopt the usage of Genetic Algorithms at each balancing cycle in order to dynamically adjust Holt's parameters, α and β. Changes to the parameters affect the performance of Holt's model and how it react to changes to the load, more specifically load imbalances. This is what we are aiming for.

Genetic Algorithm runs in iterations where certain type of processing is done on chromosomes to find a chromosome that has characteristics that fits our model the best. Each chromosome in our example is 12-bit in length that is divided in half. Each half is a binary representation Holt's model's parameters, α and β. The binary representation is parsed to

integers and then divided by 10^6 to get a 6-digit precision value that are between 0 and 1.

Algorithm 2 shows the flow this variant follows to work. First, the smoothed value and the trend is computed using values of α and β. Then stores different values in an array that keeps track of 3 different values (current load, previous smoothing, previous trend) for the three different projection (SP, MP, and LP) for the last 6 balancing cycles in a 6-by-9 array. This array is important as it directs this variant to properly change the values of Holt's parameters. Afterwards, the GA kicks in.

GA runs over the set of chromosomes to calculate the fitness. The fitness is calculated with the help of the 6-by-9 array; for a balancing cycle i, the long term projection is calculated by balancing cycle $i-5$, the medium term projection is calculated by $i-3$, and the short term projection is computed by balancing cycle $i-1$. To calculate the fitness, all chromosomes are parsed to give numerical values for α and β. Afterwards, the algorithm makes use of the previous smoothed value and trend per previous balancing cycle to calculate a projection. These projections of these previous data should match the current load of the node. Therefore, the fitness is recognized as the difference between the projections and the current load. GA goes through all chromosomes to extract α and β to compute the projections, then the difference is computed against the current load, as shown in following equation:

$$f = \frac{3 \times (\mid load - SP \mid) + 2 \times (\mid load - MP \mid) + (\mid load - LP \mid)}{6} \quad (1)$$

Because of the importance to the SP projections, the difference between SP and MP is given the highest weight, followed by MP, and the lowest weight for LP.

Algorithm 2 Genetic Algorithm Approach

Require: $\alpha, \beta, MaximumIterations, projections, chromosomes$
$shortPeriod \leftarrow 1, mediumPeriod \leftarrow 3, longPeriod \leftarrow 5$
$Smoothing_{curr} \leftarrow smooth(Load_{curr}, Val_{prevSmoothed}, Trend_{prev}, \alpha)$
$Trend_{curr} \leftarrow trend(Load_{curr}, Val_{prevSmoothed}, Trend_{prev}, \beta)$
$proj[shortPeriod][shortCurrLoad] \leftarrow Load_{curr}$
$proj[shortPeriod][shortPrevSmoothing] \leftarrow Smoothing_{prev}$
$proj[shortPeriod][shortPrevTrend] \leftarrow Trend_{prev}$
$proj[mediumPeriod][medCurrLoad] \leftarrow Load_{curr}$
$proj[mediumPeriod][medPrevSmoothing] \leftarrow Smoothing_{prev}$
$proj[mediumPeriod][medPrevTrend] \leftarrow Trend_{prev}$
$proj[longPeriod][longCurrLoad] \leftarrow Load_{curr}$
$proj[longPeriod][longPrevSmoothing] \leftarrow Smoothing_{prev}$
$proj[longPeriod][longPrevTrend] \leftarrow Trend_{prev}$
$proj.removeIndex(0)$
$proj.addNew()$
for $i \leftarrow MaximumIterations$ **do**
  $calculateFitness(Load_{curr})$
  $doSelection()$
  $findBest()$
  $doCrossover()$
  $doRandomMutation()$
  if $shouldStop()$ **then**
    $break$
  end if
end for
$\alpha, \beta \leftarrow bestCombination()$

The equation shows that the higher the error appears, higher the fitness becomes. As we prefer to have a higher fitness for the chromosomes with the lower error, the fitness is reversed $f = \frac{1}{f}$ after it is calculated.

After calculating the fitness for each chromosome, a natural selection approach is used to calculate the changes of selecting a chromosome. The used selection approach is Roulette Wheel Selection as it considered the fastest among the other selection methods, such as Rank and Tournament approaches. However, it lacks providing more diversified chromosomes. Having more diversified chromosomes allows the process of reaching to an ideal chromosome for our model faster. As we are using a system that is time constrained, the least resource demanding approach was selected.

The probability of choosing a chromosome using the Russian Wheel Selection is $P(choice = i) = \frac{fitness(i)}{\sum_{j=1}^{n} fitness(j)}$, where $\sum_{i=1}^{n} P(i) = 1$. Two random numbers are generated. The chromosomes that their probabilities contain the random numbers are selected as a candidate parent chromosomes for a child chromosome in the next iteration. The child chromosome is a combination of the two parent chromosomes, which contains a random number of bits from the first parent and the rest of its size, 12-bits in total, is filled from from the other parent.

After creating a list of all children, a random mutation is applied to the children where a random number is generated at each bit to see if a bit mutation is needed. The bit mutation's goal is to create a diversified child chromosomes. This process happens for a number of iterations. As we are dealing with a real system, an additional measurement is enforced to when the process should exit from the iteration. In the greedy approach, it stops when the goal is reached or no better chromosomes are generated. Waiting until one of the previous statements happen would take a lot of time. Thus, the variant applied a maximum iteration that represents the maximum number of allowed iteration before it stops. After it stops, the best combination of α and β is chosen to be the Holt's parameters in the next balancing cycle.

Algorithm 3 Fitness calculation

for $c \leftarrow NumberOfChromosomes$ **do**
  $\alpha, \beta \leftarrow parseChromosome(chromosome)$
  $smoothness \leftarrow smoothness(proj[0][Smoothness_{shortPrev}], \alpha)$
  $trend \leftarrow trend(proj[0][Trend_{shortPrev}], \beta)$
  $SP \leftarrow smoothness + trend$
  $smoothness \leftarrow smoothness(proj[0][Smoothness_{mediumPrev}], \alpha)$
  $trend \leftarrow trend(proj[0][Trend_{mediumPrev}], \beta)$
  $MP \leftarrow smoothness + trend$
  $smoothness \leftarrow smoothness(proj[0][Smoothness_{longPrev}], \alpha)$
  $trend \leftarrow trend(proj[0][Trend_{longPrev}], \beta)$
  $LP \leftarrow smoothness + trend$
  $f \leftarrow \frac{(\mid current - SP \mid) + (\mid current - MP \mid) + (\mid current - LP \mid)}{3}$
end for

V. Performance Analysis

In the evaluation of the proposed approach, the enhancements are compared against the prediction model used in the prediction-based distributed load balancing system [1]. The evaluation process is done in two stages. The first stage performs the evaluation of the prediction techniques on data samples, generated by running several distributed simulations

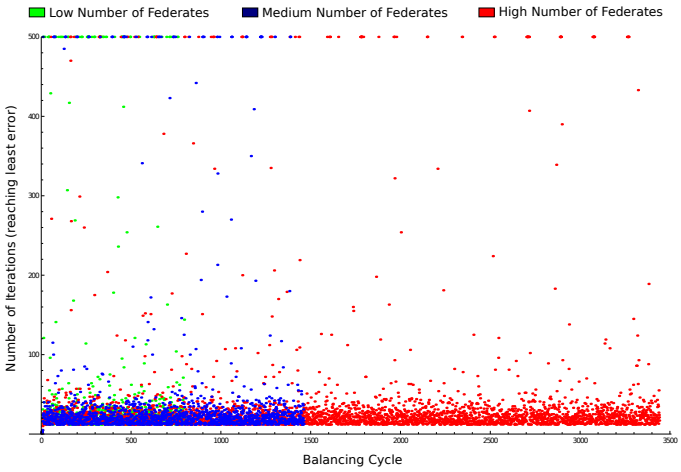


Fig. 2. Sample Coverage Percentage per # Iterations

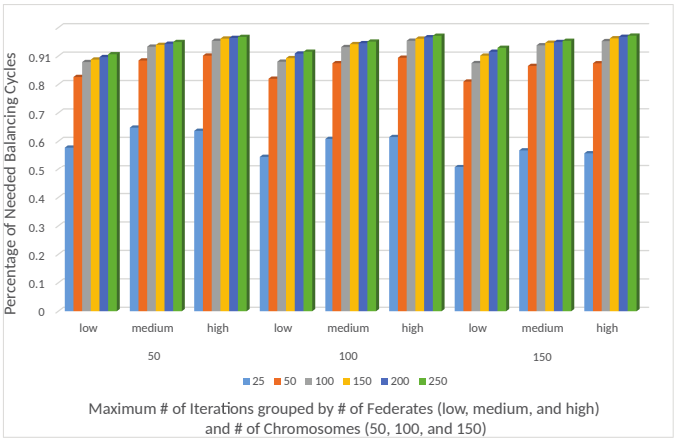


Fig. 3. Needed iterations per balancing cycle

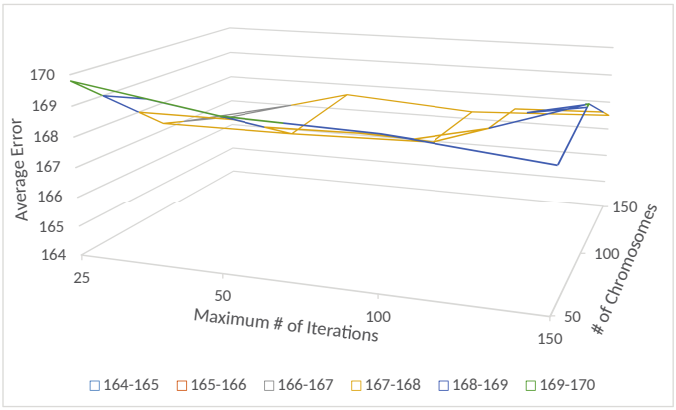
on top of a set of shared resources. A systematic computational load was used in the system to emulate simulation overhead. The aim of the first stage is to test the efficiency of the methods and to properly configure them for real simulations. The second stage is presented in the Evaluation section where one of the variants is implemented on real dynamic load balancing system.

First, in order to examine GA's performance, a thorough analysis is established to identify the combination of iterations and chromosomes that fits our case. An accuracy test of different configurations is conducted to compare the error for each computed projection in the different federate setups we have, which is discussed in the previous section.

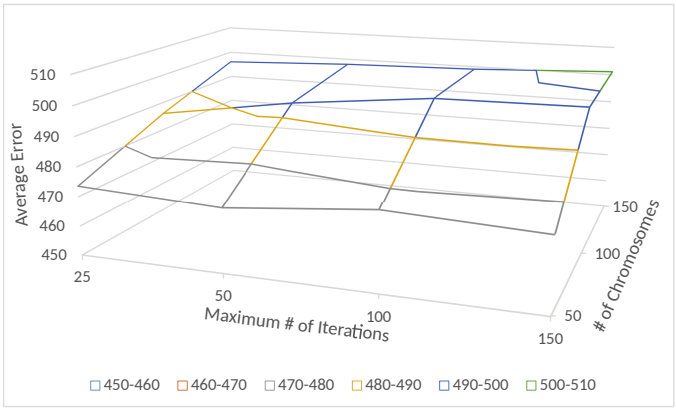
There is not any simple and direct rule to define the needed number of chromosomes. However, the more chromosomes we have the more diversified combinations of α and β we get. Therefore, three different numbers (50, 100, and 150) of chromosomes were chosen for the accuracy comparison. On the other hand, finding the suitable number of iterations requires deep analysis to understand their distribution. To do so, the simulation was executed 10 times per each federate configuration with a loose maximum number of iteration that was set to 500. Besides the additional iteration metric, GA still follows the greedy approach and stops when the target is reached (error equals to 0) or the system could not find a better combination of α and β for a consecutive 10 iterations. Figure 2 shows the average number of iterations the system needed per each balancing cycle. The least number of iterations needed by the system is 10 and it shows that the system started with a certain chromosome that resulted in the least error between the actual and the projected load for 10 consecutive runs; therefore, it stops and returns that chromosome as the one with the best combination of α and β for the balancing cycle. The maximum, however, is 500 and it is expected as sometimes GA requires more iterations to find the best chromosome, especially when the error is decreasing in each iteration. The introduced metric enforces the system to stop GA in order to comply with the real-time restriction imposed by the processing time needed per each balancing cycle.

Our initial goal was to start 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, and then increase the number of iterations to cover more. Figure 3 shows the percentage of balancing cycles where the system could find a best chromosome without reaching the limit. For example, GA could find the best chromosome for 64% of the balancing cycles when a medium number of federates is chosen and 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 has not shown a noticeable and profitable effect on the total coverage.

Figure 4 shows 2 samples of the average error of different configurations according to the prediction and load stress on the system; graph (a) shows the errors obtained when SP is calculated with medium number of federates, and graph (b) shows the errors when MP is calculated with high number of federates. Because of the ordering in analysing pair matches, a higher importance is assigned to SP calculations, so GA focuses on providing the lowest error for SP. However, achieving the lowest errors for SP does not always assure the lowest errors for MP. As a result, choosing a high number of chromosomes that run for a long time, long iterations, does not provide any noticeable differences in terms of the total average error. Thus, a number of chromosomes and iterations are chosen so that it provides a decent error within low time and space complexity. By analysing the different errors for the different case, we identified and chose the following parameters for configuring the GA mechanism for the load balancing system: 50 chromosomes and 50 iterations.



(a) SP Calculated with Medium Number of Federates



(b) MP Calculated with High Number of Federates

Fig. 4. The average error per configuration

VI. Conclusion

In this paper, a genetic algorithm has been employed to dynamically adjust the parameters that configure a prediction model, Holt's model. This model is used in the predictive load balancing system, and it provides means for load forecasting using projection to define load tendency in short time intervals. The load forecasting presents limitations due to the static linearity of the projections. The genetic algorithm allows the adaptation of the model to accommodate the latest changes on the load behaviour. However, the algorithm introduces likely introduces delays on the prediction calculations since it requires some time, iterations, to converge. Analysis has been conducted in a predefined computational intensive simulation scenario to identify the best configuration on the genetic algorithm to fulfil the trade-off between precision and response time. The results showed the implemented algorithm can reach reasonable values in a very short number of iterations, 50. This allows the system to be suitable to be applied on the balancing system. However, the gain that the modifications that the genetic algorithm may offer needs to be thoroughly evaluated on a range of different execution scenarios to identify the adaptability of the algorithm.

References

[1] R. De Grande and A. Boukerche, "Predictive dynamic load balancing for large-scale hla-based simulations," in *Distributed Simulation and Real Time Applications (DS-RT), 2011 IEEE/ACM 15th International Symposium on*, Sept., pp. 4–11.

[2] E. A. Feinberg and D. Genethliou, "Chapter 12 load forecasting."

[3] J. Feng, "A new method for ionospheric short-term forecast using similar-day modeling," in *Antennas, Propagation EM Theory (ISAPE), 2012 10th International Symposium on*, Oct 2012, pp. 472–474.

[4] 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," *Power Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 1214–1221, Nov 1990.

[5] O. Hyde and P. F. Hodnett, "An adaptable automated procedure for short-term electricity load forecasting," *Power Systems, IEEE Transactions on*, vol. 12, no. 1, pp. 84–94, Feb 1997.

[6] T. Haida and S. Muto, "Regression based peak load forecasting using a transformation technique," *Power Systems, IEEE Transactions on*, vol. 9, no. 4, pp. 1788–1794, Nov 1994.

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

[8] W. Charytoniuk, M.-S. Chen, and P. Van Olinda, "Nonparametric regression based short-term load forecasting," *Power Systems, IEEE Transactions on*, vol. 13, no. 3, pp. 725–730, Aug 1998.

[9] R. F. Engle, C. Mustafa, and J. Rice, "Modelling peak electricity demand," *Journal of Forecasting*, vol. 11, no. 3, pp. 241–251, 1992. [Online]. Available: http://dx.doi.org/10.1002/for.3980110306

[10] 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.

[11] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[12] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. New York, NY, USA: Cambridge University Press, 2000.

[13] M. Mohandes, "Support vector machines for short-term electrical load forecasting," *International Journal of Energy Research*, vol. 26, no. 4, pp. 335–345, 2002. [Online]. Available: http://dx.doi.org/10.1002/er.787

[14] S. J. Kiartzis, A. Bakirtzis, J. Theocharis, and G. Tsagas, "A fuzzy expert system for peak load forecasting. application to the greek power system," in *Electrotechnical Conference, 2000. MELECON 2000. 10th Mediterranean*, vol. 3, May 2000, pp. 1097–1100 vol.3.

[15] V. Miranda and C. Monteiro, "Fuzzy inference in spatial load forecasting," in *Power Engineering Society Winter Meeting, 2000. IEEE*, vol. 2, 2000, pp. 1063–1068 vol.2.

[16] K. Liu, S. Subbarayan, R. Shoults, M. Manry, C. Kwan, F. Lewis, and J. Naccarino, "Comparison of very short-term load forecasting techniques," *Power Systems, IEEE Transactions on*, vol. 11, no. 2, pp. 877–882, May 1996.

[17] 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," *International Journal of Forecasting*, vol. 22, no. 1, pp. 1 – 16, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207005000907

[18] J. Taylor and P. McSharry, "Short-term load forecasting methods: An evaluation based on european data," *Power Systems, IEEE Transactions on*, vol. 22, no. 4, pp. 2213–2219, Nov 2007.

[19] S. Abdullah, N. Sapii, S. Dir, and T. Jalal, "Application of univariate forecasting models of tuberculosis cases in kelantan," in *Statistics in Science, Business, and Engineering (ICSSBE), 2012 International Conference on*, Sept 2012, pp. 1–7.

[20] L. Wu, J. Yan, and Y. Fan, "Data mining algorithms and statistical analysis for sales data forecast," in *Fifth International Joint Conference on Computational Sciences and Optimization*, 2012, pp. 577–581.

[21] E. S. G. Jr., "Exponential smoothing: The state of the art - part ii," *International Journal of Forecasting*, vol. 22, no. 4, pp. 637 – 666, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207006000392