Noname manuscript No.
(will be inserted by the editor)

Combining Time Series Prediction Models Using Genetic Algorithm to Auto-scaling Web Applications Hosted in the Cloud Infrastructure

Valter Rogério Messias · Julio Cezar Estrella · Ricardo Ehlers · Marcos José Santana · Regina Carlucci Santana · Stephan Reiff-Marganiec

Received: date / Accepted: date

Abstract In a cloud computing environment, companies have the ability to allocate resources according to demand. However, there is a delay that may take minutes between the request for a new resource and it is ready for using. This causes the reactive techniques, which request a new resource only when the system reaches a certain load threshold, are not suitable for the resource allocation process. To address this problem, it is necessary to predict requests that arrive at the system in the next period of time to allocate the necessary resources, before the system becomes overloaded. There are several time-series forecasting models to calculate the workload predictions based on history of monitoring data. However, it is difficult to know which is the best time series forecasting model to be used in each case. The work becomes even more complicated when the user does not have much historical data to be analyzed. Most related work, considers only single methods to evaluate the results of the forecast. Other work propose an approach that selects suitable forecasting methods for a given context. But in this case, it is necessary to have a significant amount of data to train the classifier. Moreover, the best solution may not be a specific model, but rather a combination of models. In this paper we propose an adaptive prediction method using genetic algorithms to combine time-series forecasting models. Our method does not require a previous phase of training, because it constantly adapts the extent to which the data is coming. To evaluate our proposal we use three logs extracted from real web servers. The results show that our proposal often brings the best result, and is generic enough to adapt to various types of time series.

Keywords Genetic Algorithm · Time Series Prediction Models · Elasticity · Cloud Computing

1 Introduction

With the emergence of cloud computing, computing resources, such as storage, processing and bandwidth are available as services, in which the user pays only for the resources used during a period of time. The elasticity, the main characteristic of cloud computing, allows to quickly allocate and deallocate large amounts of resources at runtime. These resources can be acquired automatically and quickly, with increasing demand, and released on lower demand. For users, the resources available to be used appear to be unlimited and can be purchased in any amount and at any time [7].

To take advantage of this new reality, companies are increasingly using the cloud infrastructure to host their Web applications [29]. The reasons for that are practically two: avoid non-compliance with SLAs (Service Level Agreements), considering an increase in the number of requests; and save money, in situations of low amount of requests. The main challenge is to join these two objectives.

Infrastructure providers, such as Amazon, offer the possibility that the user allocates and deallocates resources by paying a fixed price per hour. Thus, the

V. R. Messias · J. C. Estrella · R. Ehlers · M. J. Santana · R. C. Santana
Institute of Mathematics and Computer Sciences
University of Sao Paolo
P.O. Box 668
E-mail: {valterrm, jcezar, ehlers, mjs, rcs}@icmc.usp.br

S. Reiff-Marganiec
Department of Computer Science
University of Leicester
Leicester - UK
E-mail: srm13@le.ac.uk

companies using such infrastructures must decide the amount of resources used at each time.

The techniques for deciding when and how to allocate the resources are named auto-scaling and can be separated into two groups: reactive and proactive. The reactive techniques keep monitoring the system events (CPU usage, number of requests in the system, queue length, etc.) and choose to allocate or release resources when these events exceed a threshold. On the other hand, proactive techniques attempt to predict the amount of resources at any given time, to anticipate unwanted events.

The problem is that in reactive techniques, often, time to react is insufficient. The time between the request for a resource in the cloud (virtual machine) and it being ready for use can be minutes [26], sufficient time for overloading the system. Concerning proactive techniques, the time series analysis, based on classical statistical models offers a spectrum of forecasting methods. But as none of these methods is the best in all cases, we are going to use genetic algorithm to combine the benefits of the individual forecasting methods to achieve higher forecast accuracy to auto-scaling web applications hosted on cloud infrastructure.

The goal is use several time series models and combine them using genetic algorithm. The advantage of this technique is that the auto-scaling can adapt to new types of workloads. That is important because the workloads, specially in web applications, usually change characteristics over time. The genetic algorithm will work to adjust a suitable weight for each prediction time series model used in the system.

Our forecasts are based on five statistical models: naive model (Naive); autoregressive model (AR); autoregressive moving average model (ARMA); autoregressive integrated moving average model (ARIMA); and extended exponential smoothing model (ETS). After predicting demand, a queue $M/M/m$ model is used to calculate the amount of resources. The goal is to determine the minimum amount of resources to meet the demand without violating service level agreements.

To perform the experiments three real web logs were used. The logs have different characteristics which makes the results and conclusions more realistic and reliable to be used in real web applications.

In summary, the contributions of this paper are as follows: (i) We propose a novel forecasting methodology that uses genetic algorithm to combine time series-based forecasting approaches. (ii) We propose a new metric for measuring the elasticity, named MEI (Mean Elasticity Index). (iii) We evaluate our proposal in the context of multiple different experiments and case studies based on real web server logs.

The results of our evaluation considering multiple different scenarios show that combination of time series forecasting models using genetic algorithms offers a generic and adaptative model, which often brings the best results, and is therefore more appropriate than statically selected fixed forecasting methods.

The remainder of this paper is organized as follows: in the next section related work will be presented and discussed. Next, in section 3, we will present an introduction to genetic algorithms and how we will use them in our proposal, including a brief explanation about time series forecasting models. After that, in section 4, our System Architecture will be presented. Next, in section 5, the evaluation of the proposal and discussion of results will be presented. Finally, in section 6, the conclusion and proposals for future work are announced.

2 Related Work

This section aims to present a survey on work related to resource allocation and demand forecasting in cloud computing environments.

Threshold-based rules or policies are very popular in cloud providers such as Amazon EC2, and third-party tools such as RightScale [14]. This tool allows setting the rules such as the upper and lower thresholds for the performance variable (e.g. 30% and 70% of CPU load). This technique requires an extra effort from the user, who needs to select suitable performance variables or logical combination of variables, and also to set several parameters [27]. Several authors have adopted auto-scaling techniques based on *rightScales auto-scaling algorithm* [4]. [13] propose a set of reactive rules based on the number of active sessions. [12] extends the previous work following the RightScale approach: if all VMs have active sessions above the given upper threshold, a new VM is provisioned; if there are VMs with active sessions below a given lower threshold and with at least one VM that has no active session, the idle one will be shut down. Some work such as [9] and [25] use adaptive techniques to dynamically define threshold. In these techniques initial values are set-up, but they are automatically modified as a consequence of the observed SLA violations. In conclusion, due to their simplicity, rules become a popular way to auto-scaling applications without much effort, specially in the case of applications with quite regular, predictable patterns. However, in case of bursty workloads the client should consider a more advanced and powerful auto-scaling system.

Other works propose more sophisticated reactive techniques, based on control theory. In [5] the authors propose combining two proactive and adaptive controllers for scaling down with dynamic gain parameters based on input workload, and a reactive approach for scaling up. In [30] the authors discussed a MIMO (Multiple-input multiple-output) adaptive controller that uses a second-order ARMA (Auto Regressive Moving Average) to model the non-linear and time-varying relationship between the resource allocation and its normalized performance. The controller is able to adjust the CPU and disk I/O usage. The work proposed in [23] designs different SISO (Single-input single-output) and MIMO controllers to determine the CPU allocation of VMs, relying on Kalman filters. [32] and [33] apply an adaptive fuzzy controller to the application, and estimate the required CPU load for the input workload. The problem of the control theory based approaches is that they are difficult to set up and are essentially reactive, which can cause problems during the allocation of resources, since these take minutes until they are ready for use.

Finally, we highlight the work using proactive techniques for demand forecasting and resource allocation in the cloud. In [22] an optimal VM-level auto-scaling scheme with cost-latency trade-off is proposed. The number of requests was predicted in each re-allocation time-unit using linear regression. Next, the optimal number of VMs was calculated based on queuing theory, using a $M/M/m$ model. In [31], the authors proposed a look-ahead resource allocation algorithm based on model predictive control. A second order ARMA model was used on Fifa World Cup 98 workload [6] to predict the number of requests that arrive in the system. After forecasting, an optimization function was used to determine the number of resources to be allocated.

In [17] the authors present an auto-scaling system that supports heterogeneous cloud infrastructures and different client requirements. Several statistical models were considered for workload prediction and a decision tree was used to determine optimal resource combination. An approach classifying WIB (Workload Intensity Behaviour) to dynamically select appropriate forecasting methods is presented in [19]. Based on user-specified forecasting objectives, a decision tree decides on the appropriate forecasting method through direct feedback mechanisms that evaluate and compare the recent accuracy of different forecasting methods. The authors in [8] presented a comparative study about the performance of two predictive models (Holt-Winters and ARIMA) using a workload extracted from a NASA WWW server [3].

Proactive techniques, using time series analysis, are very appealing for implementing auto-scalers, as they are able to predict future demands arriving to elastic applications. Having this information, it is possible to provide resources in advance and deal with the time required to start up new VMs. However, their main drawback is the prediction accuracy, that highly depends on several factors, including: the target application, input workload pattern and/or burstiness, the selected metric, the history window and prediction interval, as well as on the specific technique being used. To address these problems we need a model that suits the changes of the factors mentioned above, combining the various forecasting techniques for every time interval. That is the goal of this work.

3 Genetic Algorithm

Genetic algorithms (GAs) use concepts from the principle of natural selection to address a wide range of problems, in particular optimization. They are robust, generic and easily adaptable. Inspired by the way Darwinism explains the process of evolution of species, the GAs are broken down into the following steps: initialization, evaluation, selection, crossover, mutation, update and completion. Figure 1 describes the overall structure of a simple GA.

Basically, what a genetic algorithm does is create a population of possible answers to the problem to be treated, and then submit it to the process of evolution. Next, will be described each step:

– **Evaluation**: the ability of solutions is evaluated (individuals of the population) by means of an analysis in order to establish which individuals are more likely within the population (best solution to the problem);
– **Selection**: individuals are selected for reproduction. The probability of a given solution be selected is proportional to its fitness;
– **Crossing**: characteristics of the chosen solutions are recombined, generating new individuals;
– **Mutation**: characteristics of individuals resulting from the reproduction process are changed, thus adding variety to the population;
– **Update**: individuals created in this generation are inserted in the population;
– **Finishing**: it is checked whether the conditions for the end of evolution has been reached, returning to the evaluation stage if not, and ending execution otherwise.

3.1 Representation

Individuals are the fundamental unit of a genetic algorithm: they encode possible solutions to the problem to be treated, and it is through its manipulation (by the process of evolution) that answers are found. The

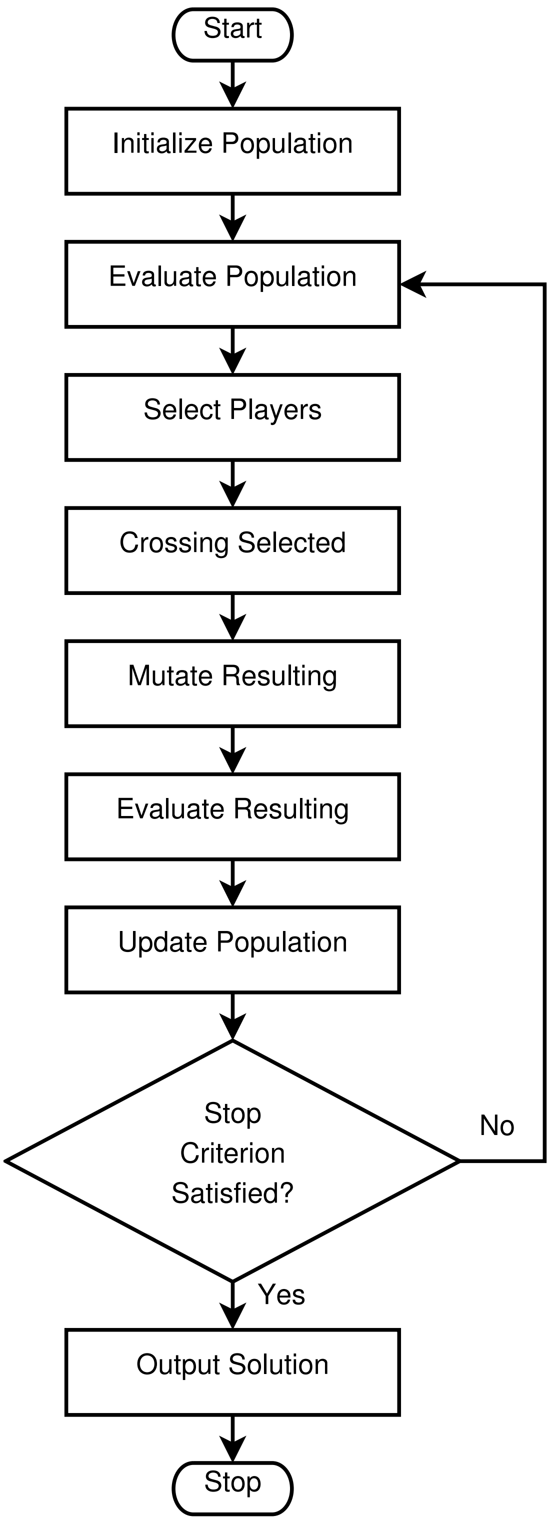


Fig. 1: The overall structure of genetic algorithm

choice of representation for individuals is the most important step in development of a GA, since it will have primary responsibility for program performance. In this study, each individual is represented by a set of five real numbers. Each of these numbers are called gene. Each gene represents a model of time series forecasting. The values that each gene may assume belong to the range between zero and one. The sum of the values of all genes must be equal to one. Each individual is composed of a combination of genes representing each of the prediction models used. These models will be described in section 3.2. The value of each gene represents the weight of the model has the individual. Equations 1, 2 and 3 show the formalized representation of genes and individuals.

$$gene \in [0,1] \tag{1}$$

$$individual = \{gene_1, gene_2, gene_3, gene_4, gene_5\} \tag{2}$$

$$\sum_{i=1}^{5} gene_i = 1 \tag{3}$$

3.2 Classical Statistical Models for Time Series Forecast

Naive Forecasting

The naive method is very simple because it is based on the assumption that the last observed value will occur in the next time interval. This method requires only a single time series point to be applied.

Auto-Regressive (AR)

In this model, the variable of interest is forecasted using a linear combination of past values of the variable. The term auto-regression indicates that it is a regression of the variable against itself [20]. Equation 4 shows an autoregressive model of order p.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + e_t \tag{4}$$

where c is a constant and e_t is white noise. We refer to this as an AR(p) model.

Despite being a simple method with optimal computational performance [19], according to [20] auto-regressive models are remarkably flexible at handling a wide range of different time series patterns.

Moving Average (MA)

Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model [20]. Equation 5 represents a moving average model.

$$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q} \tag{5}$$

where e_t is white noise. This is denoted an MA(q) model. The values of e_t are not observed, so it is not really a regression in the usual sense. Each value of y_t can be thought of as a weighted moving average of the past few forecast errors [20].

The strength of this method is the simplicity, and the weakness is the lack of sensitivity to trends and seasonal components [19].

Simple Exponential Smoothing (SES)

Exponential smoothing is a technique that assigns exponentially decreasing weights over time, whereas in the simple moving average the past observations are weighted equally, as described in equation 6 [20].

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots \quad (6)$$

where $0 \leq \alpha \leq 1$ is the smoothing parameter. The one-step-ahead forecast for time $T + 1$ is a weighted average of all the observations in the series $y_1, \ldots, y_T$. The rate at which the weights decrease is controlled by the parameter α [20]. Since the weights are adjusted to be larger on more recent observations, this model has more flexible reaction to trends or other developments than the moving average model, however, has no seasonal component or interpolation.

Auto-Regressive Moving Averages (ARMA)

ARMA models provide a parsimonious description of a stationary stochastic process in terms of two polynomials, one for the auto-regression and the second for the moving average. The model is usually then referred to as the ARMA(p, q) model where p is the order of the autoregressive part and q is the order of the moving average part.

The ARMA model is useful for times series with some noise and changes within trend, but no seasonal behavior [19].

Holt-Winters Method

Holt-Winters method was created to capture seasonality. This method comprises the forecast equation and three smoothing equations: one for the level ℓ_t, one for trend b_t, and one for the seasonal component denoted by s_t, with smoothing parameters α, β^* and γ. There are two variations to this method: additive and multiplicative, that differ in the nature of the seasonal component. The first one is preferred when the seasonal variations are roughly constant through the series, while the second one is preferred when the seasonal variations are changed proportional to the level of the series [20].

Extended Exponential Smoothing (ETS)

The ETS model [21] works by choosing the best forecast model between SES model, additive or multiplicative Holt-Winters model. The forecasting process starts selecting an optimized model instance for a given time series. After, the parameters of the equations are estimated. Having the model and the parameters adapted to the time series data, point forecasts are computed. [19].

ARIMA (Auto-Regressive Integrated Moving Averages)

The ARIMA model is a stochastic process modeling framework [10] defined by six parameters (p, d, q) and (P, D, Q), where the first triple defines the model concerning trend and noise component, and the second vector is optional and defines a model for the seasonal component. P or p stands for the order of the AR(p) process, D or d for the order of integration (needed for the transformation into a stationary stochastic process), and Q or q for the order of the MA(q) process [19].

The weakness of this model is the difficulty of selecting parameters. In general, the selection is done using different unit-root tests and Akaike information criterion (AIC). To address this problem, a process for automated model selection implemented in the auto.arima() function of the R forecast package is proposed in [21]. That function fits an ARIMA model to the time series data and select the best parameters to compute point forecasts.

3.3 Genetic Operators

Initialization

After choosing a representation method, the next step will be choose the population size and a method for generating the initial population. The initial population should has a gene pool as large as possible in order to be able to explore the whole search space, and thus, in this paper, the initial population has been generated considering five individuals. Each individual is a time series forecasting model, that is, has one of its genes with value equal to one and the other genes with value equal to zero. In sequence, all individuals are combined each other (arithmetic average) in groups of two, three, four and five individuals. Thus is formed the initial population.

Evaluation

The evaluation is a process of assigning a value to each individual according to a fitness function. This value shows the goodness of an individual. In this work, we propose a metric named Elasticity Index (EI) as fitness function. The aim is to maximize the sum of EI (SEI) in previous forecasts. Equation 7 shows the sum of EI (SEI). We can see in Equation 7 that EI can assume values between 0 and 1. The closer to 1 the value, the better is the solution. The parameter N represents the number of past predictions. Thus, the best individual (solution) is the one that maximizes the sum of EI. Figure 2 illustrates the evaluation process.

$$\text{SEI} = \sum_{i=1}^{N} \frac{min(V_{\text{exp},i}, V_{\text{pred},i})}{max(V_{\text{exp},i}, V_{\text{pred},i})} \tag{7}$$

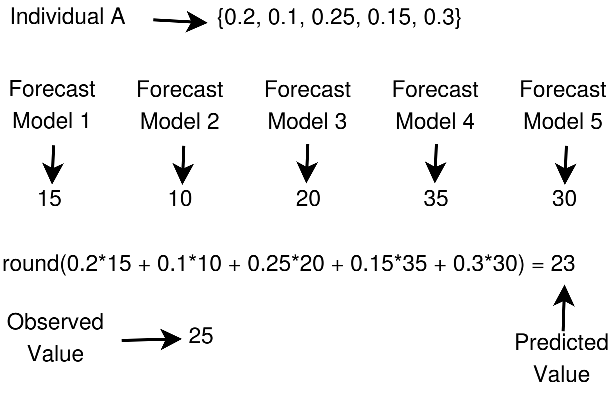


Fig. 2: Evaluation Example

In Figure 2 we can see how the forecast models are combined using the individual genes as weights for your values. After that, we have the predicted value and we can use the Equation 7 to calculate the EI. This process is repeated for each individual in all previous prediction, and each EI value is added to previous values. The individual with the highest value for the SEI will be better able to solve the problem.

We want to use the metric EI as fitness function because it favors the model that has the best performance most of the time, i.e., it is less sensitive to outliers. Other metrics such as MAE and RMSE, are very sensitive to outliers, that is, if an individual has very poor performance at some point, it can be misjudged by these metrics, even if it has done well most of the time.

Selection

Selection is the process of choosing two parents from the population for crossing. In this study, we opted to select the best 50 individuals ranked in the evaluation phase. These individuals are going to the crossover phase.

Crossover

Crossover is done to explore new solutions. This operator changes defined parts of two members that are selected and obtains different members that give new points in the search space. In this work, the chosen crossover operator was the arithmetic average of the values assigned to the genes of each individual participant of the intersection. After the parents were selected as described above, in the Selection phase, will be made the combination of all parents, two by two, where the couple will mate with probability of 0.90. Figure 3 shows a crossover operation.

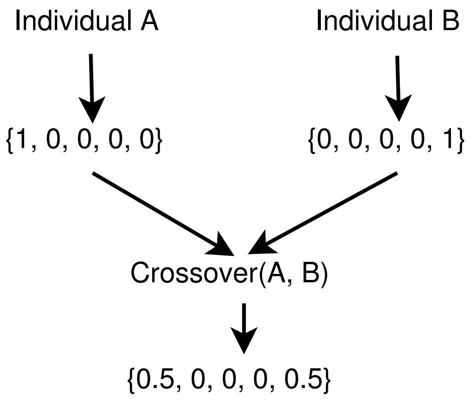


Fig. 3: Crossover Example

Mutation

The mutation operator is critical to the success of genetic algorithms since it determines the search directions and avoids early convergence. However, unlike the crossover, mutation is usually done by modifying genes within a chromosome and its probability is low. In this paper, we used the swap operator as mutation operator with probability of 0.10. In the swap operator, two randomly selected genes will exchange their values. Figure 4 shows a mutation operation.

Update

At this point, individuals resulting from the crossover and mutation process are entered in the population ac-

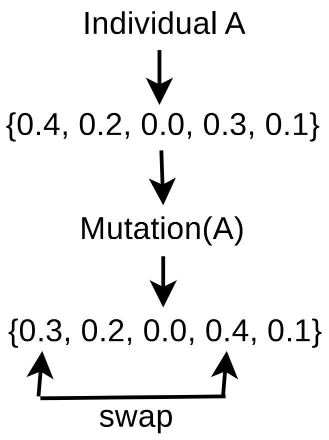


Fig. 4: Mutation Example

cording to the policy adopted by the GA. In the most traditional way, the population maintains a fixed size and individuals are created in same number as its predecessors and replace them altogether. However, there are alternatives to this approach, for example, all of the N best individuals can always be maintained. In our case, we want to maintain the original individuals and their sons resulting of crossover and mutation process.

Finishing

The finish does not involve the use of any genetic operator. It is simply composed of a test that gives order to the process of evolution if the GA has reached a predetermined stopping point. The criteria for arrest may be different from the number of generations have grown to the degree of convergence of the current population. In our case, the stopping rule is when individuals stop to evolve, that is, when the best solution (individual) of the current generation is the same as the previous generation, or when the algorithm reach one hundred iterations.

4 System Architecture

A typical Web Architecture deployed in a cloud environment is shown in Figure 5. A Front-End is responsible to receive requests from clients and distribute it between the several servers in Back-End. Each server is a Virtual Machine (VM) rented by a Cloud Provider. It is easy to see that the architecture shown represents a Queue $M/M/m$ model, in agreement with Queuing Theory [16].

A queue $M/M/m$ has a single entry point to queue and m servers to meet demand. In the architecture shown in Figure 5, the load balancer would be the point where all requests have to pass (single line). Thereafter, the load balancer distributes requests among the m servers who will be responsible for meeting them.

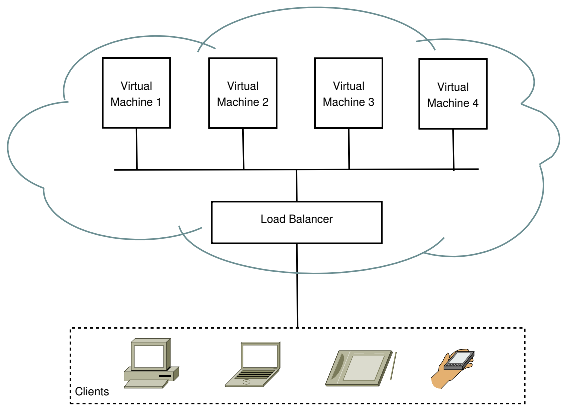


Fig. 5: Web Applications Architecture Deployed on Cloud Infrastructure.

The function of an auto-scaler is to find the minimum number of servers (m) to meet requests within the expected time (SLA) and at the same time save resources. Figure 6 shows our proposed proactive autoscaler in two phases. The first stage is responsible for forecasting demand for the next time interval. Then, in the second phase, the minimum amount of resources required is calculated to meet the demand.

Our auto-scaling system scales a Web Application in response to change in amount of requests at fixed intervals, which we denote by reconfiguration intervals set to one hour. This value (one hour) was chosen because most cloud infrastructure providers define this period as minimum to charge for resources. That is, if a resource is allocated by a lower range at a time, this will be charged as if it had been used throughout the period (one hour). Therefore, to optimize the amount paid to providers is necessary to plan the resources that will be used every hour.

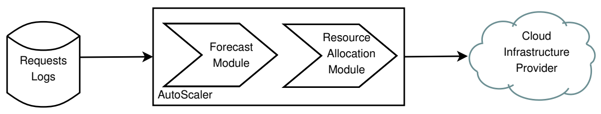


Fig. 6: Autoscaler Architecture.

4.1 Requests Forecast

For this work, we used, in addition to naive model, the functions *arima(p, d, q)*, *auto.arima()* and *ets()* of the

R forecast package. The first of them is used to implement the autoregressive model. The second one adjust the best arima model for determined time serie. The third one choose the best exponential smoothing model in agreement with data. We tried our system using three real web applications logs: three months of Fifa World Cup 98 web servers [1], two months of Nasa web servers [3] and two weeks of ClarkNet web server [2].

To compose the time series, the maximum amount of requests per second in each hour was considered. The model is adjusted hour by hour with new data that arrives to the system. After, next time series point is predicted. The algorithm 1 shows the forecast process.

```
  Data: time serie
  Result: next point forecast
  initialization;
  foreach hour do
1     $timeSerie \leftarrow readLog(allData)$;
      foreach model do
2         $model \leftarrow estimateModel(timeSerie)$;
3         $value[model] \leftarrow$
          $predictOneStepAhead(model)$;
      end
4     $nextValue \leftarrow combineModelsUsingGA(value)$;
  end
```

Algorithm 1: Forecast Process

In the line 1 the information are extracted from the web logs. Then, this information is transformed into a time series. Next, in line 2, a function is used to estimate the most appropriate parameters for a particular forecasting model, in relation to the time-series data. All the forecasting models used and their parameters are described in Section 3.2. The functions naive, *arima(1, 0, 0)*, *arima(1, 0, 1)*, *auto.arima()* and *ets()* were considered. All these functions are implemented in the statistical package R [11]. In line 3, the prediction is calculated for the next time interval, based on the estimated model in the previous step. Finally, in line 4, the combination of models using Genetic Algorithms (GA) is done. Algorithm 2 shows the process of combining models. This entire process is repeated every one-hour interval.

Before the forecast, the data were converted to logarithmic scale. This procedure is useful, especially in series that have great variation in their values, such as the FIFA world cup series. After the prediction, its value is transformed to normal range.

The algorithm 2 shows the process of combination of predictive models. Each of the steps shown in the algorithm is explained in section 3.3.

```
  Data: previous forecasts
  Result: weights for each model
  initialization;
1 $initializatePopulation()$;
  while not reach stop condition do
2     $evaluatePopulation()$;
3     $selectPlayers()$;
4     $crossSelected()$;
5     $mutateResulting()$;
6     $evaluateResulting()$;
7     $updatePopulation()$;
  end
8 $outputSolution()$;
  finalization;
```

Algorithm 2: Combine Models Process

4.2 Resource Allocation

With next point forecasted such as described in previous section, we allocated resources following the principles of a Queue model $M/M/m$. In this category of Queue, system utilization is modeled as follows [18]:

$$\rho = \frac{\lambda}{m\mu} \tag{8}$$

In equation 8, ρ is the system utilization, λ is the arrive rate, μ is the processing rate and m is the number of servers.

For the system to be stable, ρ must be less than 1 [18]. Our aim is find the smallest value to m that keep the system stability. So let's rewrite the equation 8 in m function.

$$m = \lceil \frac{\lambda}{\rho\mu} \rceil \tag{9}$$

To find an adequate ρ value it is needed consider the μ value and the desired response time to requests. The response time can be calculated as follows [18]:

$$R = \frac{\frac{1}{\mu}}{1-\rho} \tag{10}$$

Putting equation 10 in ρ function we obtain next equation:

$$\rho = 1 - \frac{1}{R\mu} \tag{11}$$

Where R represents the response time promised to the clients in Service Level Agreements (SLA).

Joining equations 9 and 11 we have the next equation:

$$m = \lceil \frac{R\lambda}{R\mu - 1} \rceil \tag{12}$$

So, the solution of our problem is to substitute the parameters in equation 12 by point forecast for next hour, server processing capacity and SLA values, to find

```
    Data: next point forecast
    Result: amount of resources to be allocated
    initialization;
    foreach hour do
1       $arriveRate \leftarrow getForecastOneStepAhead(model)$;
2       $processingRate \leftarrow getProcessingRate()$;
3       $sla \leftarrow getMaxResponseTime()$;
4       $amountServers \leftarrow$
        $calculateResources(arriveRate, processingRate, sla)$;
    end
```

Algorithm 3: Resource Allocation Process

the number of VMs needed in next hour. The algorithm 3 shows the allocating resources process.

Algorithm 3 starts picking predicting the arrival rate of requests for the next period of time. Next, the values of the processing rate for each server (virtual machine) and the maximum response time required are obtained. Based on these parameters, the amount of resources to be allocated for the next time interval is calculated according to equation 12.

5 Evaluation

In this section we will present results about accuracy of forecast models and how it works in resource allocation considering several scenarios. After that, results are analysed and discussed.

5.1 Data Logs Utilized

We evaluate our proposal using three real web server logs: three months of Fifa World Cup 98 web servers [1], two months of Nasa web servers [3] and two weeks of ClarkNet web server [2].

5.1.1 ClarkNet Log

The ClarkNet Log was extracted from a Web server for the Metro Baltimore-Washington DC area. It is based in two traces that contains all HTTP requests to the ClarkNet WWW server during two weeks. The first log was collected from 00:00:00 August 28, 1995 through 23:59:59 September 3, 1995, a total of 7 days. The second log was collected from 00:00:00 September 4, 1995 through 23:59:59 September 10, 1995, a total of 7 days. In this two weeks period, there were 3,328,587 requests. Timestamps have 1 second resolution [2].

Looking at Figure 7, we see that the ClarkNet series presents a behavior that has similarity over time, and indicate periods with higher demand. Their values are of the order of tens.

5.1.2 Nasa Log

The Nasa series was based in two traces that contains all HTTP requests to the NASA Kennedy Space Center WWW server in Florida during two weeks. The first log was collected from 00:00:00 July 1, 1995 through 23:59:59 July 31, 1995, a total of 31 days. The second log was collected from 00:00:00 August 1, 1995 through 23:59:59 Agust 31, 1995, a total of 31 days. In this two months period, there were 3,461,612 requests. Timestamps have 1 second resolution. From 01/Aug/1995:14-:52:01 until 03/Aug/1995:04:36:13 there are no accesses recorded, as the Web server was shut down, due to Hurricane Erin [3].

In NASA series, shown in Figure 7, we see a repetition in your pattern in certain periods, more prominent than in ClarkNet series. Also, their values are of the order of tens, and do not vary much among themselves.

5.1.3 Fifa World Cup Log

The Fifa World Cup logs consists of all the requests made to the 1998 World Cup Web site between April 30, 1998 and July 26, 1998. During this period of time the site received 1,352,804,107 requests. The requests timestamps have 1 second resolution. The time on each server was coordinated with the local time in France (+0200). The first access logs were collected on April 30th, 1998; the final access logs were collected on July 26th, 1998. During this 88 day period, 1,352,804,107 requests were received by the World Cup site. [1].

Fifa World Cup time series, showed in figure 7, is the most complicated of all considered in this work because it has a high degree of variation in their values, does not have a repeating pattern of data (many random components), and has peaks, wich are complicated to predict. Their values vary with great randomness of the order of tens to the thousands. The challenge is achieve good results with this type of series.

5.2 Accuracy Forecast Results

To evaluate the accuracy of the models four metrics were considered: Mean Absolute Error (MAE); Mean Elasticity Index (MEI); Root Mean Squared Error (RM-SE); and Mean Absolute Percentage Error (MAPE).

$$\mathrm{MAE} = \frac{1}{N} \sum\nolimits_{i=1}^{N} |V_{\mathrm{exp},i} - V_{\mathrm{pred},i}| \tag{13}$$

$$\mathrm{MEI} = \frac{1}{N} \sum_{i=1}^{N} \frac{min(V_{\mathrm{exp},i}, V_{\mathrm{pred},i})}{max(V_{\mathrm{exp},i}, V_{\mathrm{pred},i})} \tag{14}$$

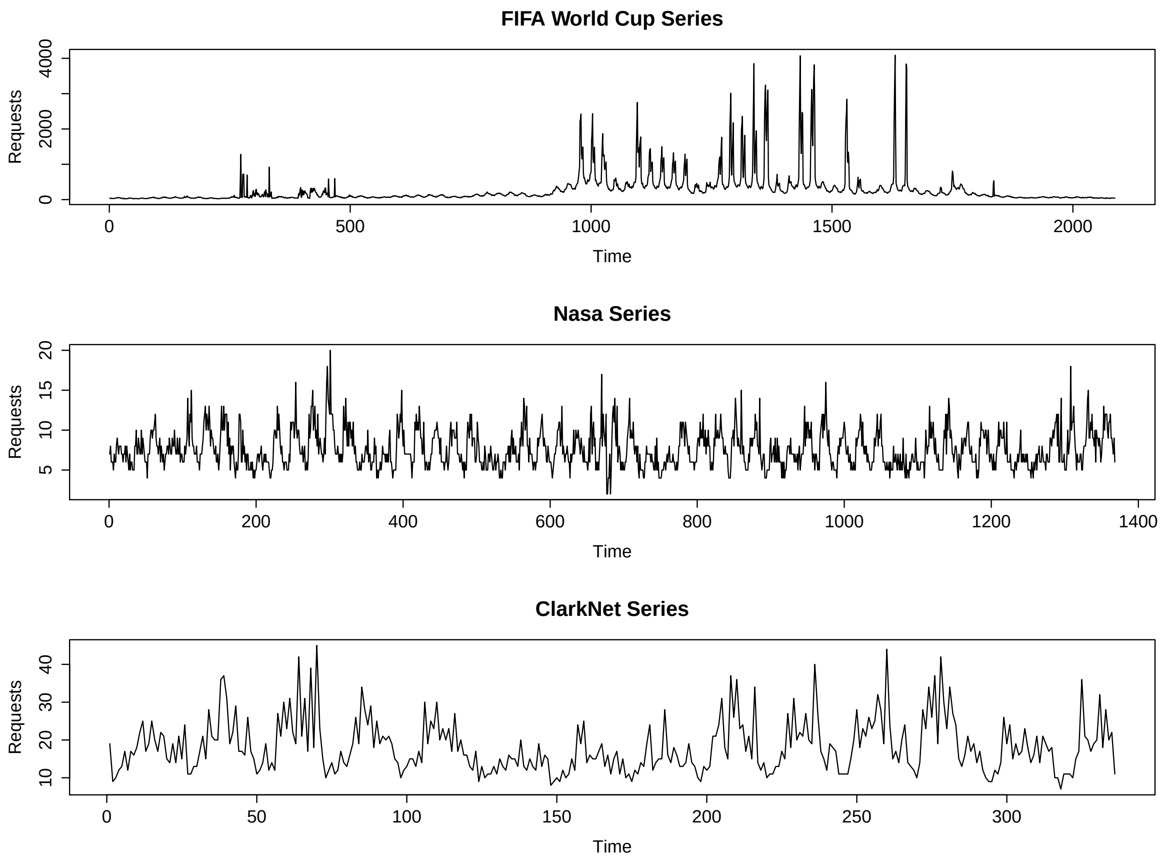


Fig. 7: Utilized Time Series.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (V_{\text{exp},i} - V_{\text{pred},i})^2} \quad (15)$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N} \left| \frac{V_{\text{exp},i} - V_{\text{pred},i}}{V_{\text{exp},i}} \right| \times 100\% \quad (16)$$

The smaller the values according to the MAE, RMSE and MAPE metrics, the better is the model performance. However, for MEI metric that may assume values between 0 and 1, the closer the value of 1, the better is the model performance. The parameter N is the number of past predictions from the beginning of the time series.

5.2.1 ClarkNet Series

The figures 8 and 9 shows the distribution of forecast errors for each model, with and without outliers respectively, considering ClarkNet series. It can be seen that the ARMA (1, 1) model had the best performance, showing a more symmetrical distribution of errors and a smaller difference between the first and the third quartile. We also see that the ARMA (1, 1) model also showed lower values for outliers. Our proposal (GA) was the one with the lowest number of outliers. The worst performance was of the NAIVE model, which showed the greatest distance among the quartiles and high values for outliers. To calculate these outliers the following formula was used:

$$outlier = \begin{cases} value < -1.5(Q3 - Q1), \\ value > +1.5(Q3 - Q1). \end{cases}$$

Where Q1 is the first quartile and Q3 is the third quartile.

Regarding the metrics used, according to Table 1, ARMA model obtained the best result for ClarkNet series, being better in all metrics. Our proposal was the second best, with close proximity to the ARMA model results. The NAIVE model had poor performance in all metrics.

Observing figure 10, which shows the evolution of the weights of each model for Clarknet series, we note that the ETS and ARMA models were the most relevant to the Clarknet series. The ETS model had more weight at the beginning of the series and the ARMA model became more important after time 200, receiving more weight. This shows the evolution of our proposal

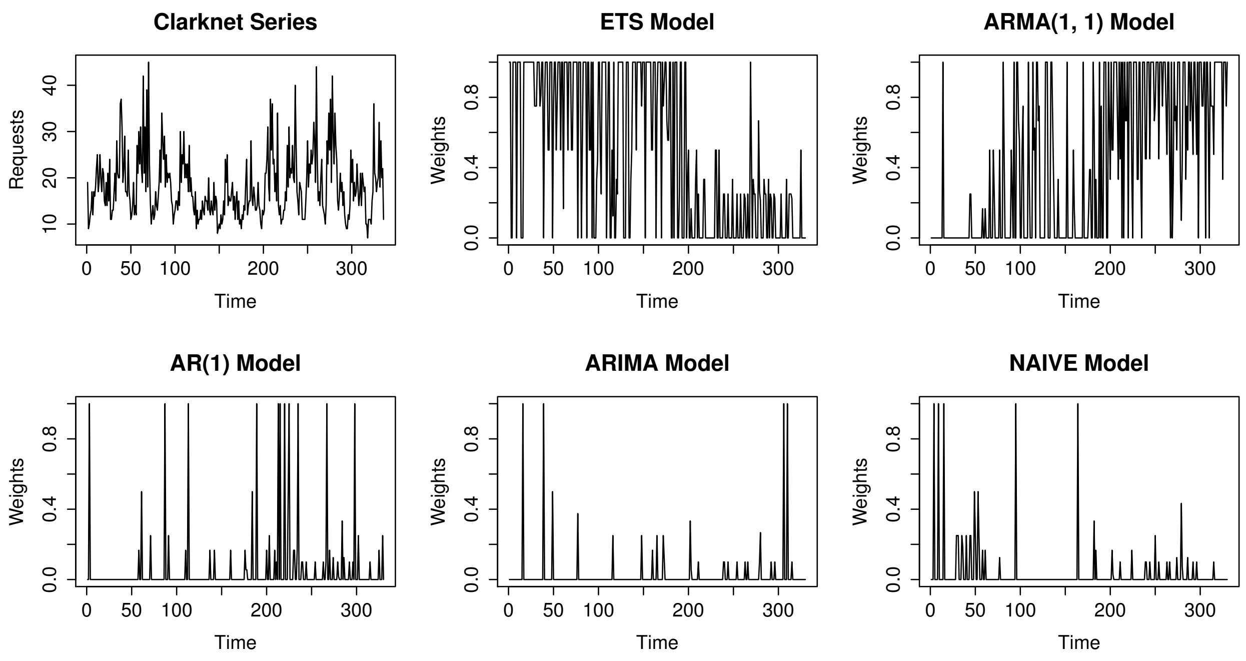


Fig. 10: Weights of the forecasting models for Clarknet series.

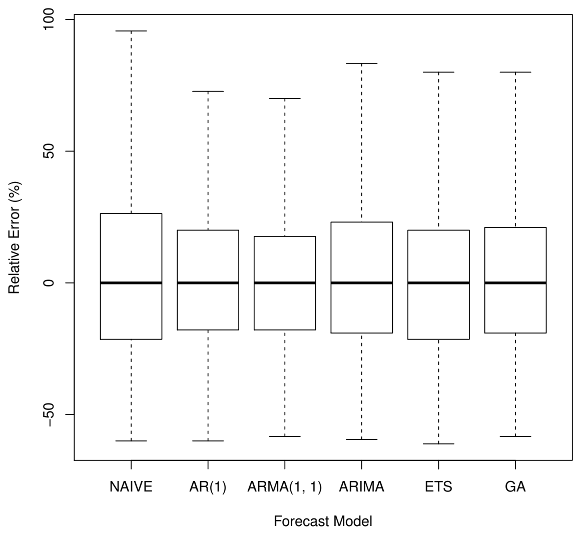


Fig. 8: Forecast Error ClarkNet Log Without Outliers.

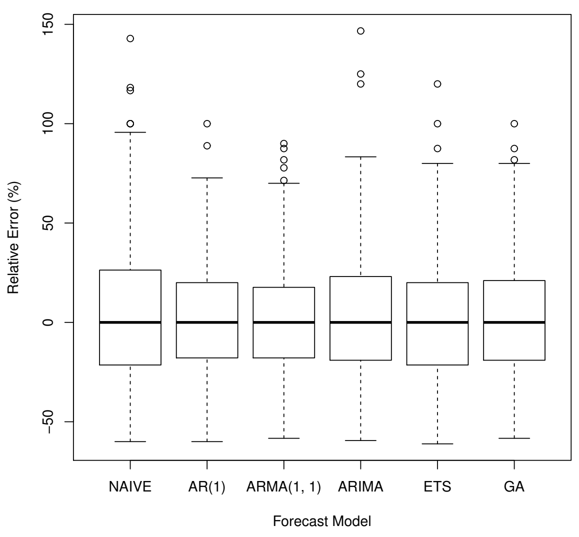


Fig. 9: Forecast Error ClarkNet Log With Outliers.

over time, adapting and converging to the model with the best performance.

5.2.2 Nasa Series

In Figures 11 and 12 we can observe that, as in Nasa series, the ARIMA model showed a better distribution of the errors. but in contrast, had a large number of outliers. Our proposal had the lowest amount of outliers, as well as lower values for its.

Table 2 shows that ARIMA model was better in all metrics. Again, our proposal has a very good performance, getting very close to the ARIMA model, according to the metrics used. Note that the ARIMA model had a bad performance considering the ClarkNet series.

Table 1: Performance of Forecast Models (ClarkNet)

Forecast Model	MAE	MEI	RMSE	MAPE
NAIVE	5.37	0.8554	7.41	28.01
AR(1)	4.46	0.8590	6.44	22.90
ARMA(1, 1)	4.29	0.8601	6.16	22.07
ARIMA	4.51	0.8588	6.33	23.96
ETS	4.45	0.8592	6.24	23.24
GA	4.35	0.8600	6.17	22.40

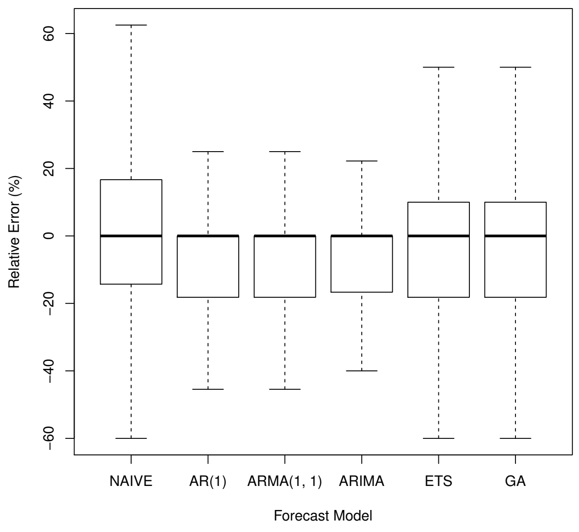


Fig. 11: Forecast Error Nasa Log Without Outliers.

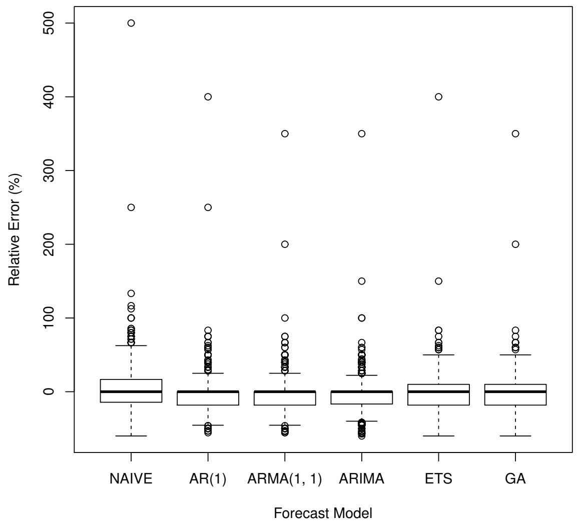


Fig. 12: Forecast Error Nasa Log With Outliers.

This shows that the choice of appropriate model is very dependent on the type of observed series. Our proposal can adapt to the series, always converging on the best model.

Table 2: Performance of Forecast Models (Nasa)

Forecast Model	MAE	MEI	RMSE	MAPE
NAIVE	1.40	0.8636	1.96	18.56
AR(1)	1.34	0.8651	1.92	16.88
ARMA(1, 1)	1.31	0.8668	1.87	16.39
ARIMA	1.28	0.8686	1.84	16.07
ETS	1.35	0.8633	1.89	17.05
GA	1.30	0.8676	1.85	16.37

Figure 13, which shows the evolution of the weights of each model for NASA series, we observed that the models had similar weights over time. Anyway, we can see that the ARIMA model received the highest weights for most of the time. Again, our proposal converged on the best model.

5.2.3 FIFA World Cup Series

In Fifa World Cup serie, there was almost a tie between our proposal (GA) and the NAIVE model, as can be seen in Figure 14. However, according to Figure 15, we see that the GA model had the smaller amount, and the lowest values for outliers.

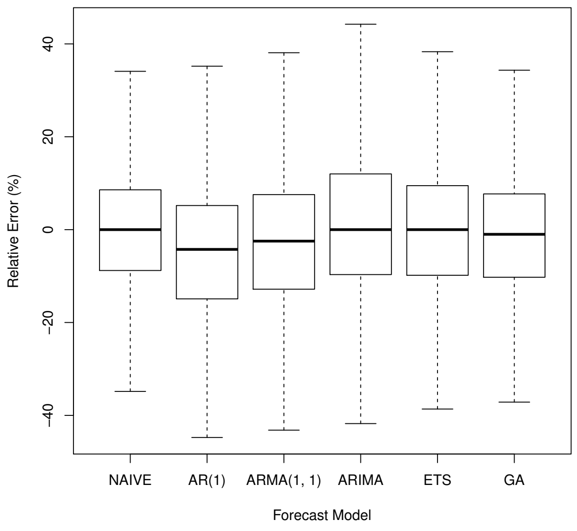


Fig. 14: Forecast Error Fifa World Cup 98 Log Without Outliers.

The results for the Fifa World Cup series reflect its complexity. Observing Table 3, NAIVE model was the

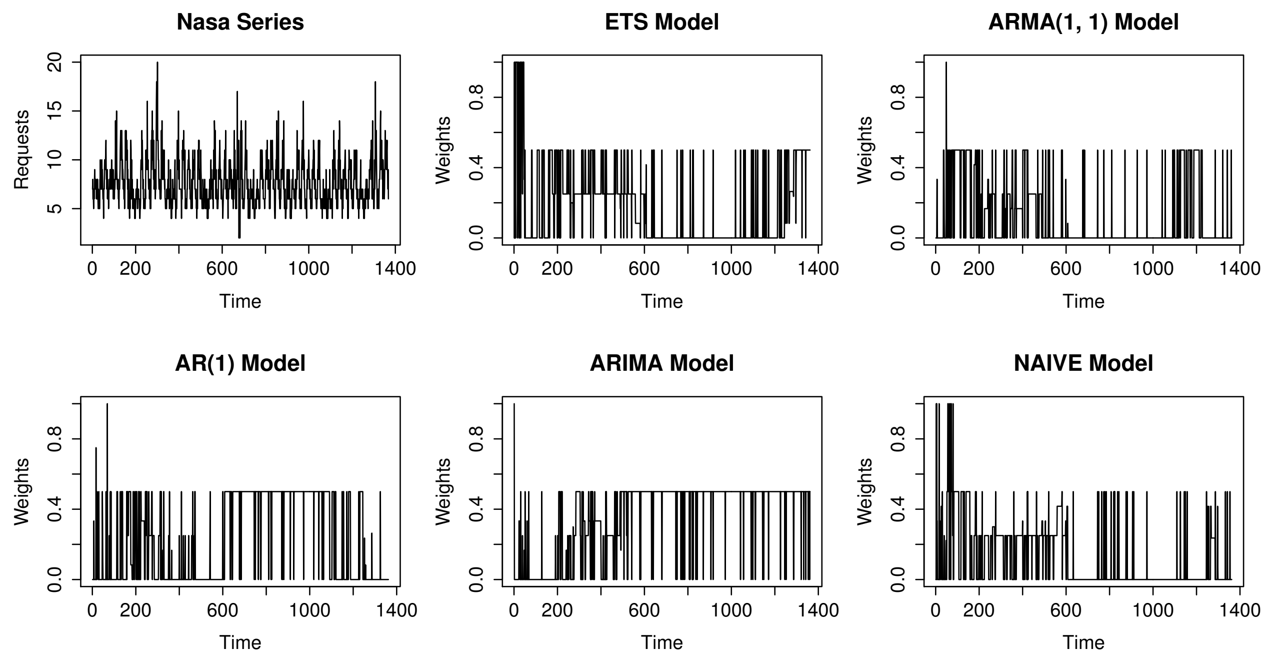


Fig. 13: Weights of the forecasting models for Nasa series.

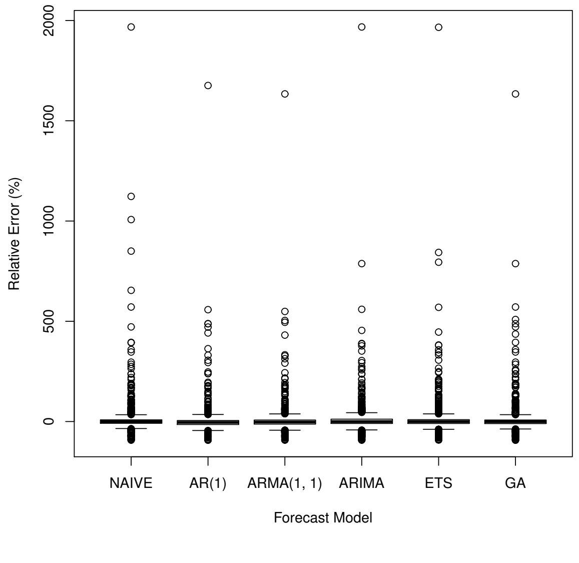


Fig. 15: Forecast Error Fifa World Cup 98 Log With Outliers.

best in the MAE and MEI metrics, while the AR (1) model was the best in the RMSE metric and the GA model was the best in MAPE metric. Interestingly, the NAIVE model, which had a poor performance in the previous series (NASA and ClarkNet), was one of the best for the FIFA World Cup series. And, once again, our proposal was able to adapt the series, converging to the best result.

Table 3: Performance of Forecast Models (World Cup)

Forecast Model	MAE	MEI	RMSE	MAPE
NAIVE	71.23	0.8760	249.46	18.64
AR(1)	76.87	0.8612	240.61	17.85
ARMA(1, 1)	81.04	0.8601	244.67	18.21
ARIMA	80.07	0.8567	248.82	19.78
ETS	80.08	0.8654	260.74	19.15
GA	71.55	0.8740	248.25	17.68

According to the figure 16, we can observe the evolution of the weights of each model to FIFA World Cup Series. We note that the ETS and AR models have received the greatest weight at the beginning of the series. From the time in 1000, the two models have lost relevance and the NAIVE model started to prevail, receiving the highest weights. Coincidentally, from the time in 1000, the series now has high variability, with a lot of bursts. The results show that none of the models used is suitable for treating this type of series. For this reason, the NAIVE model prevailed. Either way, our proposal again proved to be able to indicate into the best result.

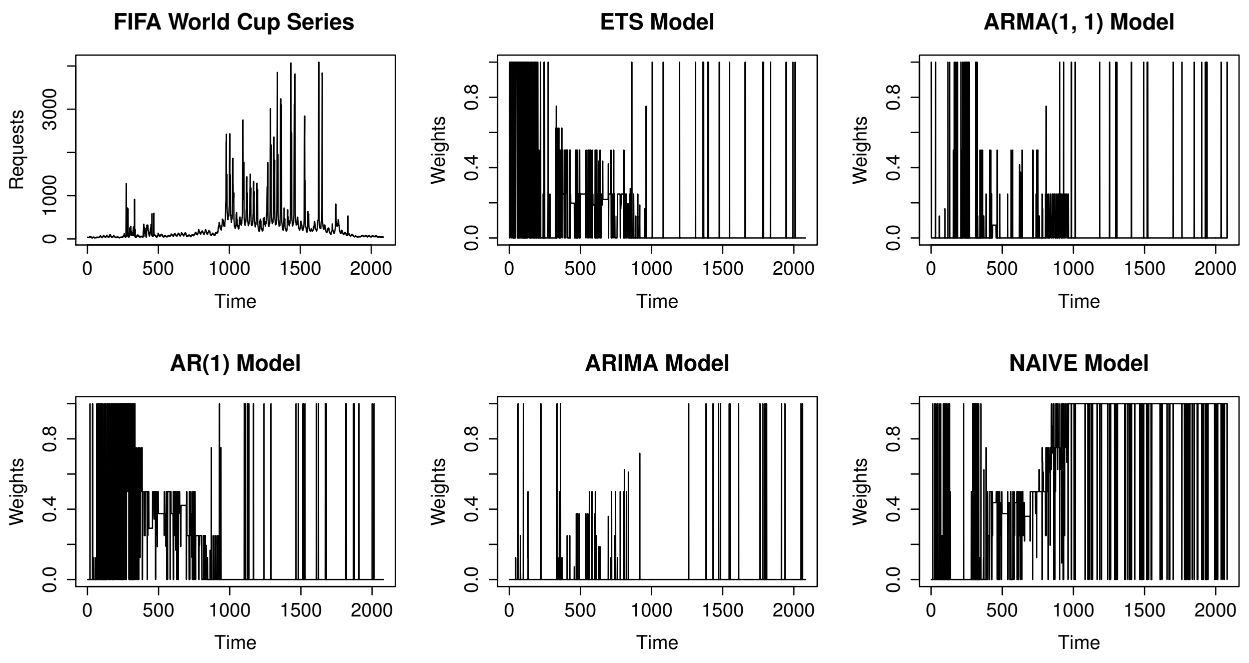


Fig. 16: Weights of the forecasting models for FIFA World Cup series.

5.3 Resource Allocation Results

To measure the efficiency of our proposal in the proper allocation of resources, we compare it with the use of a linear combination of models, by means of linear programming, as shown in Equations 17, 18, 19 and 20. To solve the Equations 17, 18, 19 and 20 we use the *SimplexSolver* class from Apache Commons Math library [28]. This class implements the two-phase Simplex algorithm [15]. We chose the Simplex algorithm to compare with our proposal (GA) because it is widely used in linear optimization problems. Initially, the zero value is assigned for the MEI of each forecasting model. After this, in each time interval, we will: a) update the MEI in accordance with the past predictions of each model; b) set the weight of each forecasting model by solving Equations 17, 18, 19 and 20; c) combine the future forecast of each model, by means of the weights calculated in the previous stage; and, finally, d) calculate and allocate the required number of resources for the next hour using the Equation 12. Figures 17, 18 and 19 illustrate the results obtained, and include three time series used in this study and two different scenarios outlined in Table 4. The X axis represents the number of under-provisioned resources and the Y axis represents the number of over-provisioned resources.

$$Z = max \sum_{i=1}^{5} M_i x_i \tag{17}$$

subject to:

$$x_i \in [0, 1] \tag{18}$$

$$\sum_{i=1}^{5} x_i = 1 \tag{19}$$

where:

M_i = mean elasticity index according to the forecasting model i

x_i = weight of the forecasting model i in the next forecast

Table 4: Scenarios considered in the experiments

Scenario	Processing rate (μ)	SLA (R)
First	10 requests per second	0.4 second
Second	20 requests per second	0.8 second

5.3.1 ClarkNet Series

Figure 17 shows the results for ClarkNet series. We can see that the GA method is in both scenarios evaluated, among the Pareto optimal solutions. The ETS and ARMA models also are among the optimal solutions, in both scenarios. The Simplex method was dominated in the second scenario, it is not an optimal solution in this case.

In the Tables 5 and 6 we can observe the exact amounts of under and over-provisioned resources generated by the GA and Simplex methods, and by each isolated prediction model. The 5 table shows that the GA method was the one that had the least value for the sum of under-provisioned and over-provisioned resources, in the first scenario. The Simplex method was only in fourth. In the second scenario, shown in Table 6, again the GA method showed the best results for the total of under and over-provisioned resources generated, tied with the ETS model. The Simplex method was fourth, next to the ARMA model.

Table 5: First Scenario: ClarkNet Series

Model	Under-provisioned resources	Over-provisioned resources	Total
NAIVE	117	117	234
AR(1)	138	70	208
ARMA(1, 1)	127	63	190
ARIMA	126	82	208
ETS	115	71	186
Simplex	106	95	201
GA	116	65	181

Table 6: Second Scenario: ClarkNet Series

Model	Under-provisioned resources	Over-provisioned resources	Total
NAIVE	52	52	104
AR(1)	81	29	110
ARMA(1, 1)	65	34	99
ARIMA	58	40	98
ETS	57	39	96
Simplex	59	40	99
GA	59	37	96

5.3.2 Nasa Series

Figure 18 shows the results for Nasa series. We can see that all models except the ARMA in the first scenario and the NAIVE in the second, were among the Pareto optimal solutions. In Table 7, which shows the results for the first scenario, we can see that the GA method had the second best result for the amounts of under and over-provisioned resources generated, getting very close to the first place, the ARIMA model. In the second scenario, presented in the Table 8, only the NAIVE model was slightly worse than the others. We can notice also that the ARIMA model, which had one of the worst performances in ClarkNet series, this time it was the model that had the best performance.

Table 7: First Scenario: Nasa Series

Model	Under-provisioned resources	Over-provisioned resources	Total
NAIVE	148	148	296
AR(1)	244	56	300
ARMA(1, 1)	223	67	290
ARIMA	215	64	279
ETS	204	87	291
Simplex	154	135	289
GA	209	73	282

Table 8: Second Scenario: Nasa Series

Model	Under-provisioned resources	Over-provisioned resources	Total
NAIVE	1	1	2
AR(1)	1	0	1
ARMA(1, 1)	1	0	1
ARIMA	1	0	1
ETS	1	0	1
Simplex	1	0	1
GA	1	0	1

5.3.3 FIFA World Cup Series

In the FIFA World Cup series, shown in Figure 19, the only model dominated in both scenarios was the ETS. The others were among the Pareto optimal solutions. However, observing the Tables 9 and 10, we note that three solutions stand out from others: NAIVE model, the GA method and Simplex method. The difference between the three solutions was very small, in both scenarios, with NAIVE model a little better than the GA method, which was slightly better than the Simplex method. Another interesting fact is that the NAIVE model, which had a very poor performance in the previous series (ClarkNet and NASA), this time was the model that had the best performance. Instead, the ETS model, which in previous cases performed well, this time was the worst performing model.

5.4 Complexity and Overhead

Figure 20 shows the overhead caused by predictive models and our proposal. To implement the models considered in the experiments we use the following functions of the statistical package R: the *arima (1, 0, 0)*

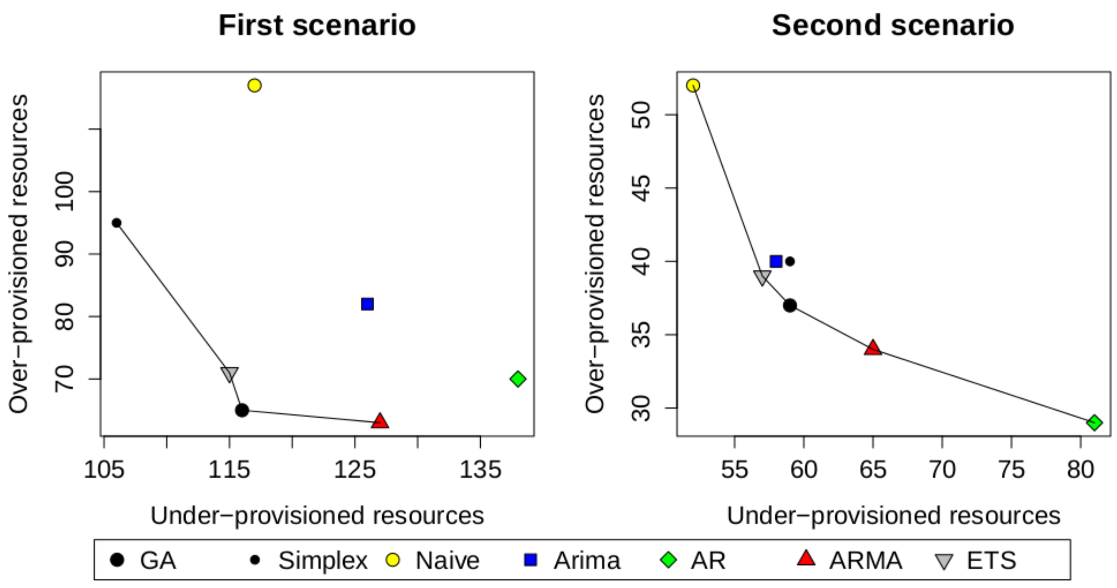


Fig. 17: Solutions for ClarkNet series.

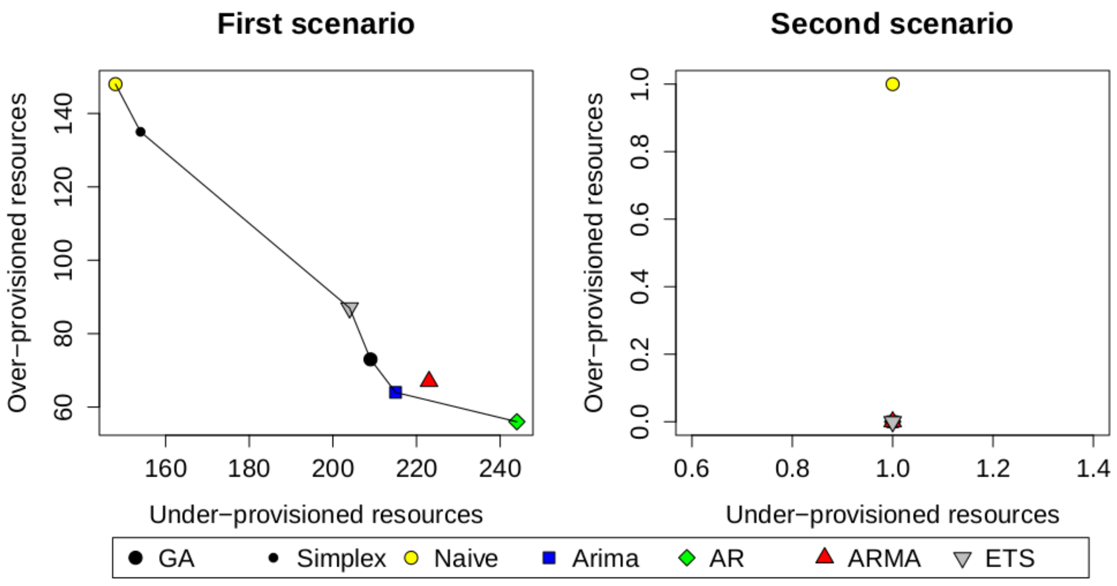


Fig. 18: Solutions for NASA series.

Table 9: First Scenario: FIFA World Cup Series

Model	Under-provisioned resources	Over-provisioned resources	Total
NAIVE	9,883	9,882	19,765
AR(1)	14,707	6,622	21,329
ARMA(1, 1)	14,655	7,800	22,455
ARIMA	13,264	8,924	22,188
ETS	11,947	10,234	22,181
Simplex	10,051	9,806	19,857
GA	10,537	9,248	19,785

Table 10: Second Scenario: FIFA World Cup Series

Model	Under-provisioned resources	Over-provisioned resources	Total
NAIVE	3,956	3,955	7,911
AR(1)	5,908	2,641	8,549
ARMA(1, 1)	5,895	3,123	9,018
ARIMA	5,356	3,540	8,896
ETS	4,807	4,057	8,864
Simplex	4,025	3,924	7,949
GA	4,238	3,696	7,934

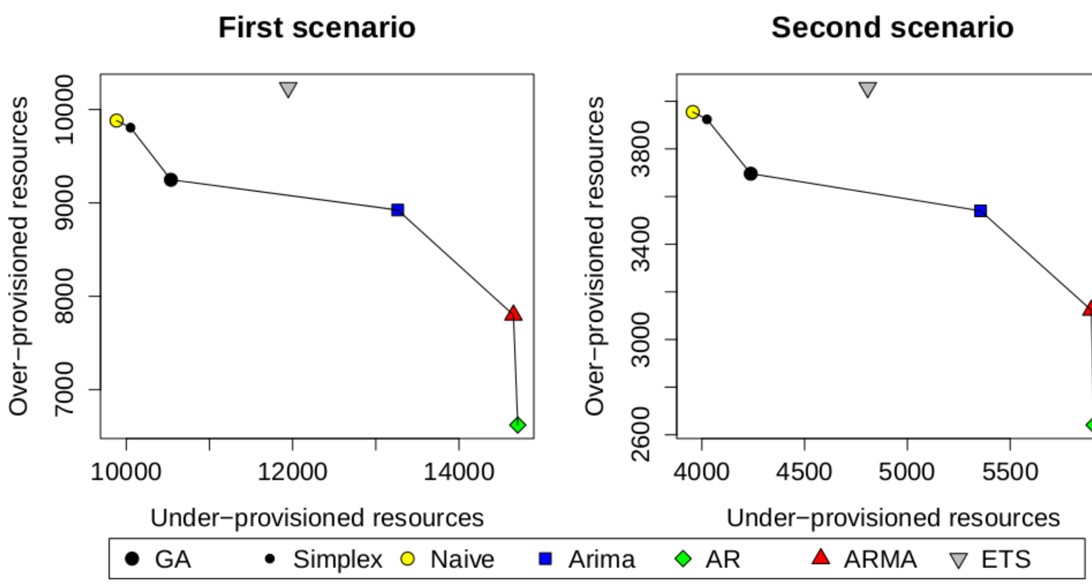


Fig. 19: Solutions for the Fifa World Cup series.

function as AR(1) model, the *arima (1, 0, 1)* function as ARMA(1, 1) model, the *auto.arima ()* as ARIMA model and the *ets ()* as ETS model. The genetic algorithm for combining models was implemented using the Java language. To perform the experiments we used a quad core machine with 2 GB of speed and 16 GB of RAM.

According to the Figure 20, the AR (1) model has 5 ms overhead for 300 data points and the ARMA (1, 1) has 8 ms overhead for 300 data points. The ARIMA model shows a considered increase in overhead over the previous models. The model has an overhead of 100 ms to 300 data points. ETS model was more complex, with an overhead of approximately 300 ms to 300 data points. Our proposal to combine the models using genetic algorithm (GA), showed an overhead slightly higher, around 400 ms to 300 data points. The Simplex method showed a overhead very similar to our proposal. However, it is important to clarify that this overhead is negligible for the application, since the prediction is performed only once every hour. The NAIVE model is not shown in the Figure because has virtually no overhead.

To analyze the complexity of our proposal, we will consider the number of comparisons that is performed to find the best solution to the problem. Whereas, in each iteration (evolution), the top 50 assessed individuals are selected for reproduction, and that intersect in pairs, with probability 0.9, we will have approximately $\binom{50}{2}$ individuals (solutions) at the end of the crossover phase.

To select the best individuals, we use the Merge-sort algorithm for ordering them according to their respective valuations. The Merge-sort complexity is $O(n * log_2 n)$, where n is the number of individuals (candidate solutions). Therefore, in our case, this complexity would be $\binom{50}{2} * log_2 \binom{50}{2}$.

At the end of each iteration, it is necessary to find the best individual of the current generation. Therefore, it is needed $\binom{50}{2}$ comparisons. As our solution allows up to 100 iterations, we have, in total, up to $100 * \binom{50}{2} * \binom{50}{2} * log_2 \binom{50}{2}$ comparisons. We can conclude, then, that the complexity of our proposal depends on two parameters: the number of selected individuals to cross (N); and maximum number of allowed iterations (M). In our case N equals 50 and M equals 100.

Therefore, given the above facts, the complexity of our proposal is $M * \binom{N}{2}^2 * log_2 \binom{N}{2}$, or $O(M * N^2 * log_2 N)$. Where N is the number of individuals selected to cross and M is the maximum number of iterations (evolutions) allowed.

Regarding the Simplex algorithm, according to the literature [24] the maximum complexity is considered exponential in K, where K is the number of problem variables. However, according to the literature [24], few problems that need this complexity to be solved. This is why the Simplex is widely used. Figure 20 shows that for a problem of five variables, as in our case, the Simplex algorithm processing time is very similar to the time spent for the solution using genetic algorithms (GA).

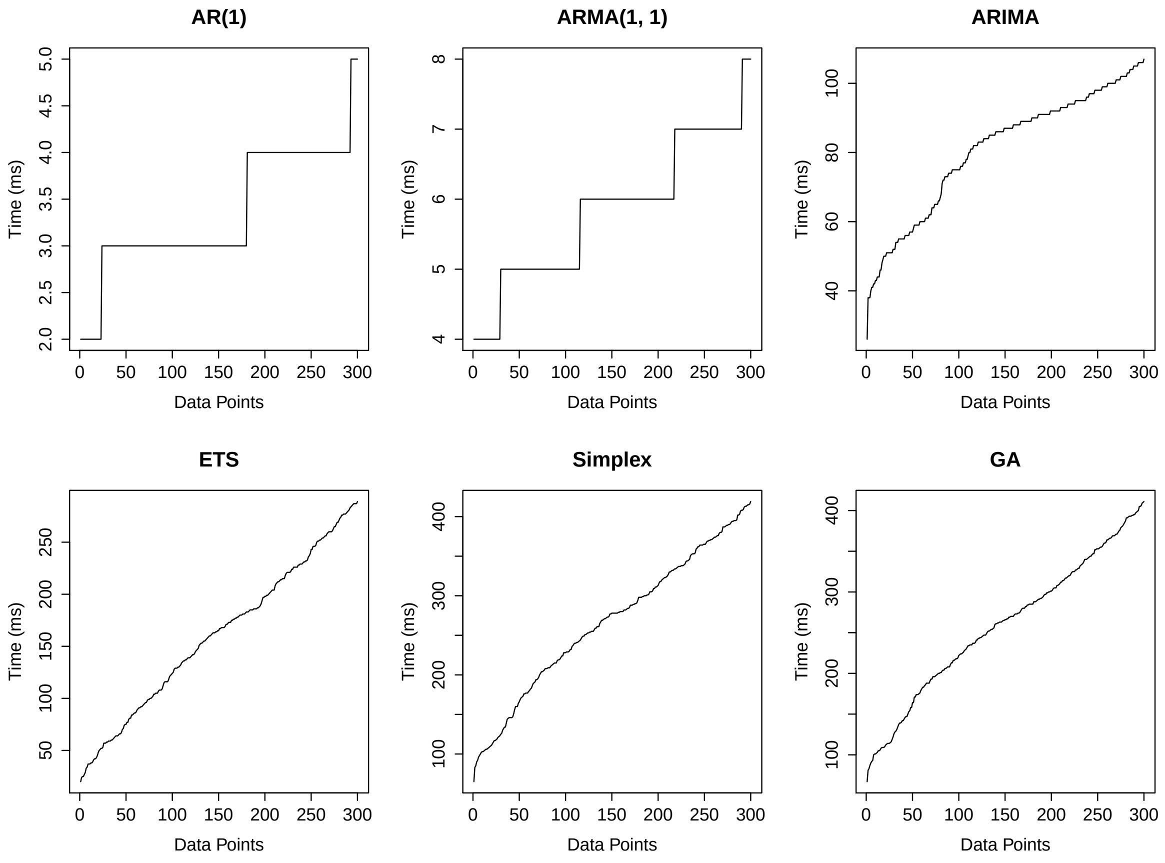


Fig. 20: Overhead of the forecasting models.

5.5 Bootstrap

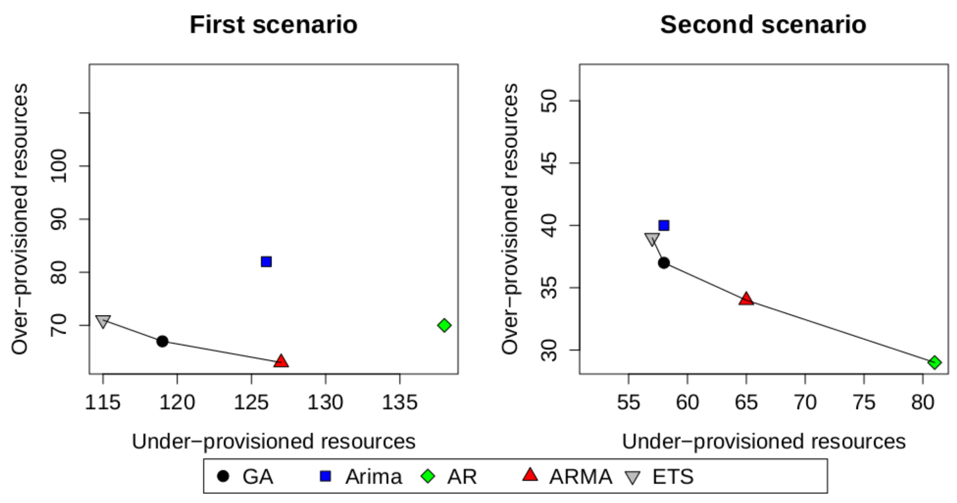
To show that our proposal does not depend on a specific forecasting model, we retrace the experiments, removing in every execution, one of the predictive models. Thus, in the first round, we removed the NAIVE model from the initial set of five models: NAIVE, AR, ARMA, ARIMA and ETS. In the second round, we removed the AR model. In the third, the ARMA model, and so on. In total there were five rounds of experiments, each without one of the five models considered in our proposal.

The Figures 21, 22, 23, 24, and 25 illustrate the bootstrap results. We note that in all cases and scenarios, our proposal (GA) has always been among the Pareto optimal solutions. All isolates forecasting models, were dominated in some scenery. This shows the adjustment capacity of our proposal, which, regardless of the series, scenario or set of models considered, always converges to an optimal solution, in relation to this set.

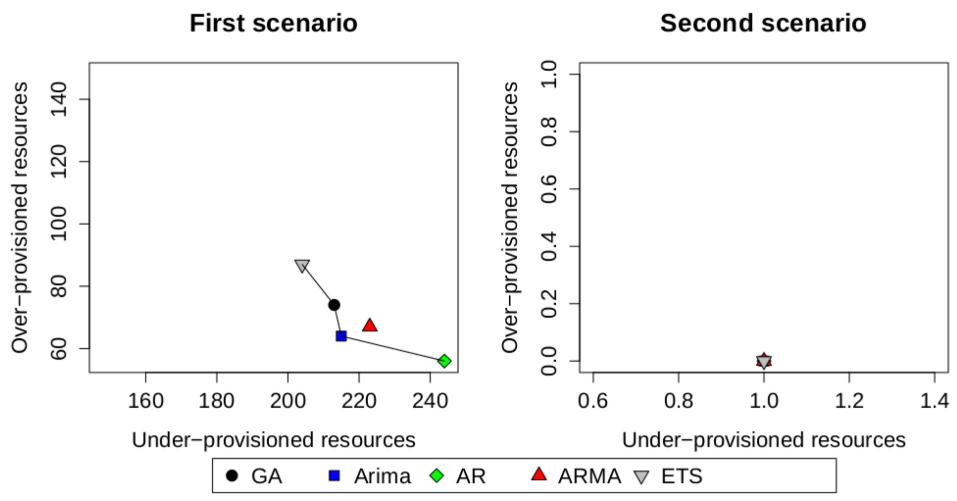
5.6 Feasibility Test

To show the feasibility of our proposal we made a projection of cost with under-provisioned and over-provisioned resources for a year, considering the nominal price of \$ 1 for either a under-provisioned resource as a over-provisioned resource. For the purposes of this paper, a resource is a virtual machine rented in an infrastructure cloud provider. In addition to the projection of costs, we also consider the overhead of each model and deployment difficulty.

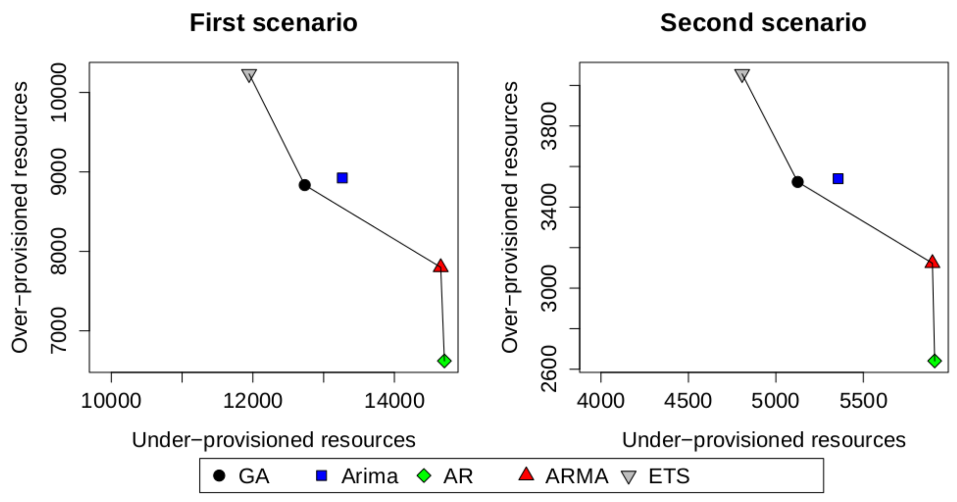
To calculate the estimated annual average cost, we use the values shown in the Tables 5, 6, 7, 8, 9 and 10, for under-provisioned and over-provisioned resources, in each series used. From these values we project the average cost in a year, multiplying these values by the number of days in a year, and divided by the number of days in the respective log, in each scenario. Then we calculated the average of the values presented in each scenario. Finally, we calculated the average of the values presented in three series and get the overall average among all series. The Table 11 illustrates the results of our study.



(a) ClarkNet series

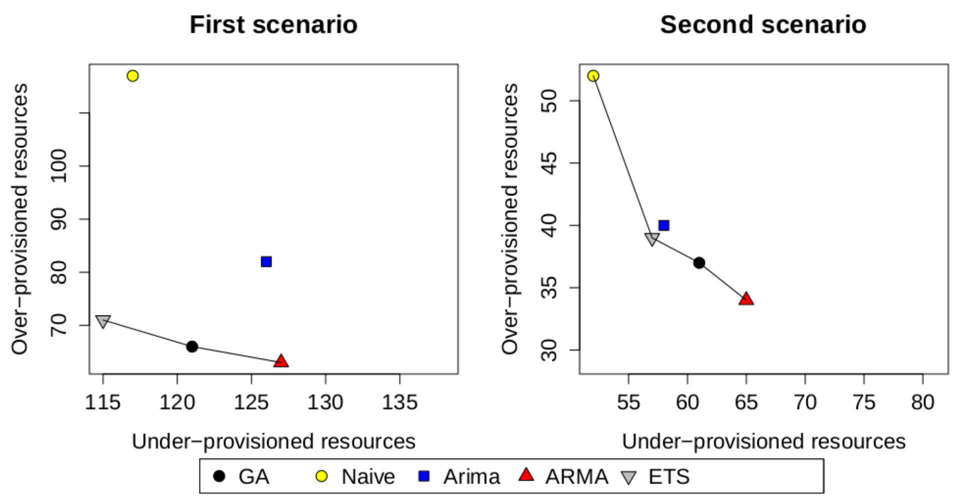


(b) Nasa series

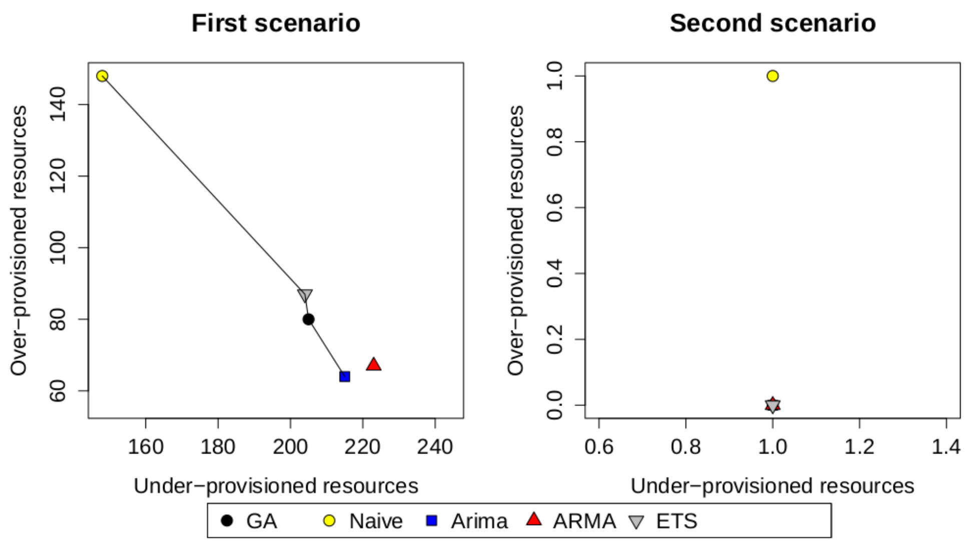


(c) FIFA World Cup series

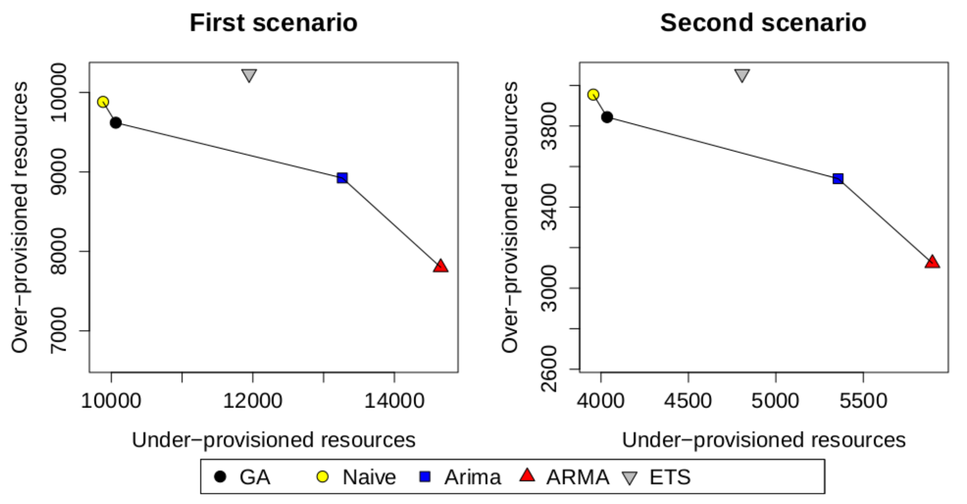
Fig. 21: Bootstrap for NAIVE Model.



(a) ClarkNet series

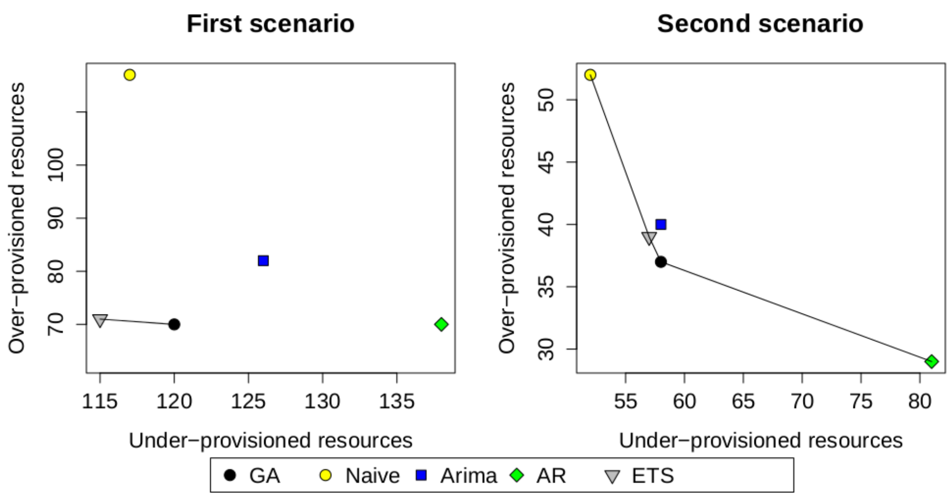


(b) Nasa series

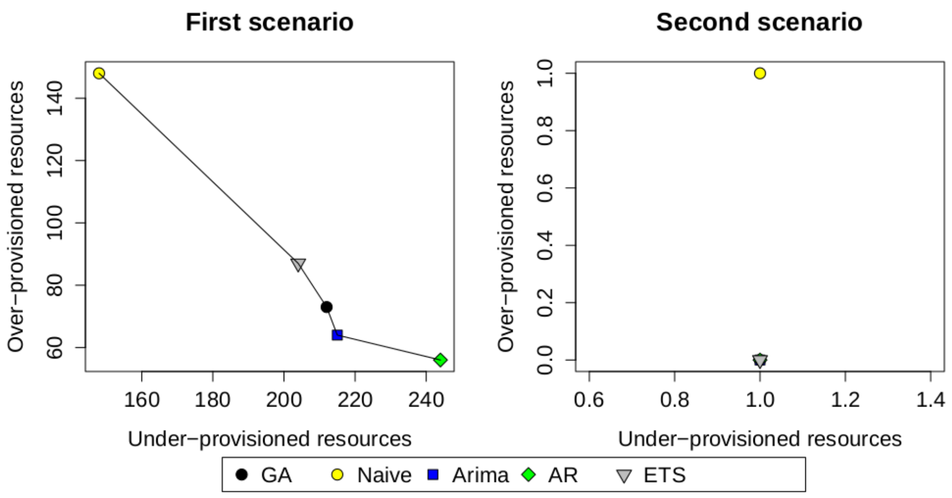


(c) FIFA World Cup series

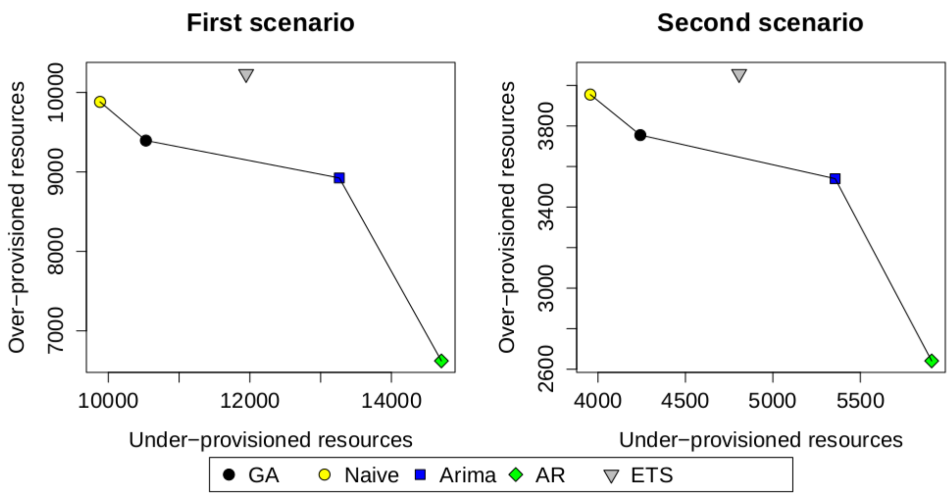
Fig. 22: Bootstrap for AR Model.



(a) ClarkNet series

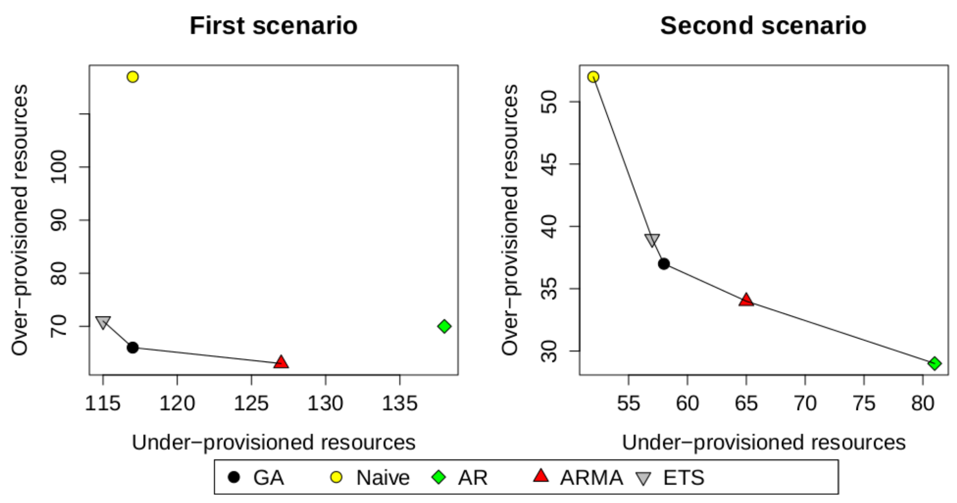


(b) Nasa series

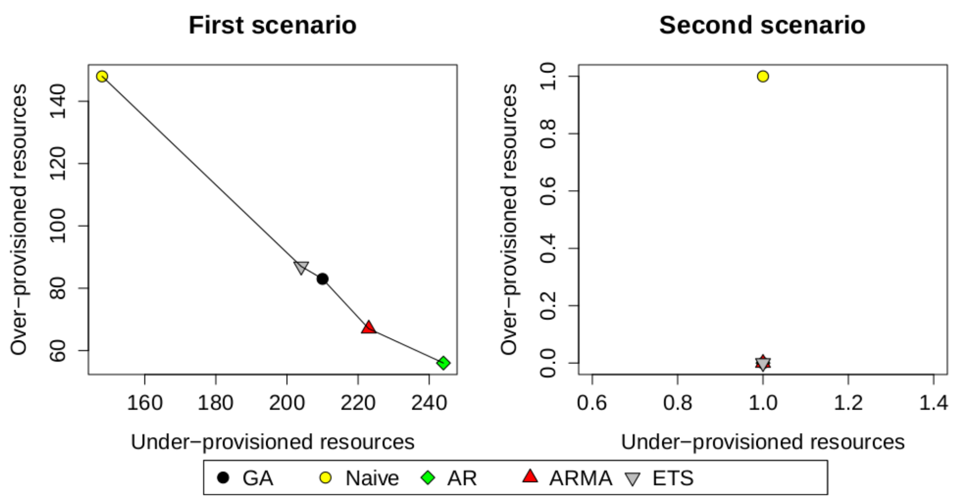


(c) FIFA World Cup series

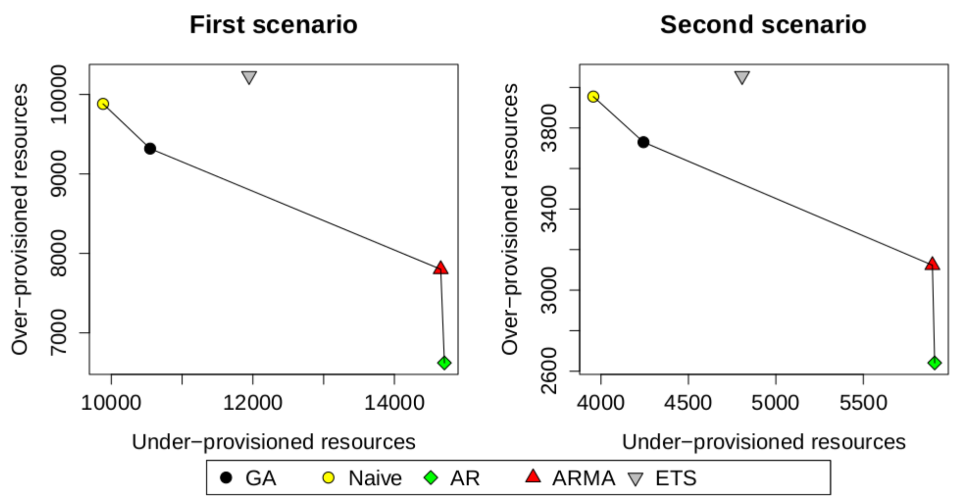
Fig. 23: Bootstrap for ARMA Model.



(a) ClarkNet series

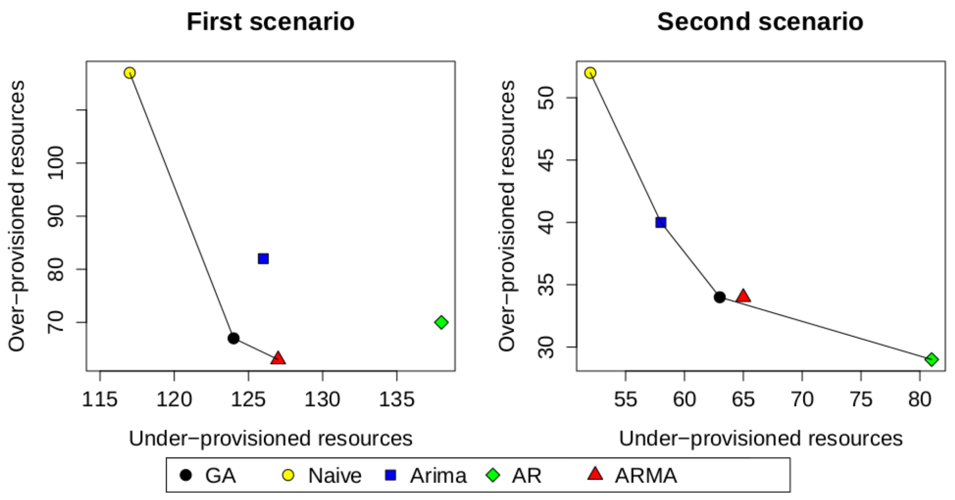


(b) Nasa series

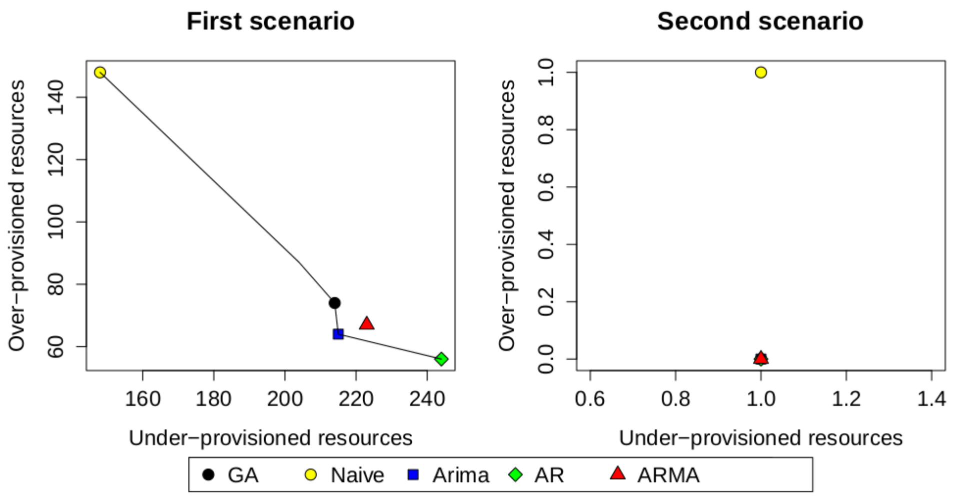


(c) FIFA World Cup series

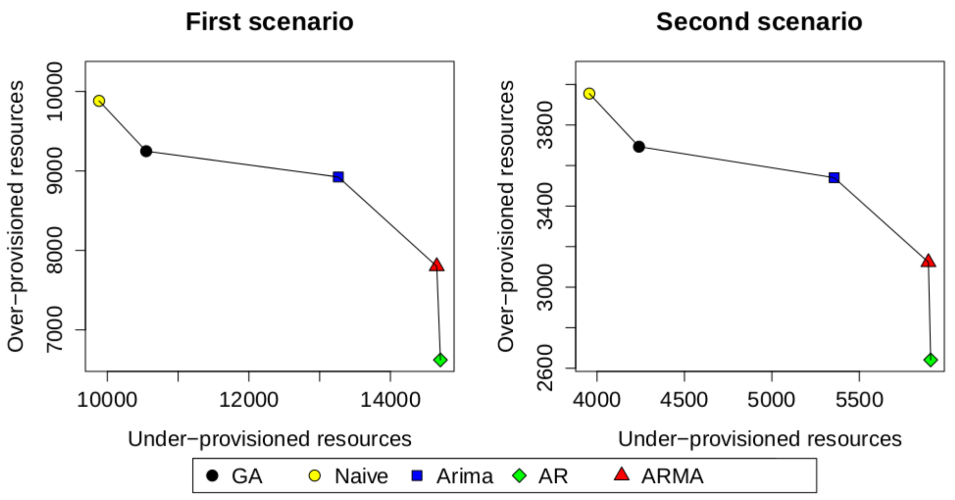
Fig. 24: Bootstrap for ARIMA Model.



(a) ClarkNet series



(b) Nasa series



(c) FIFA World Cup series

Fig. 25: Bootstrap for ETS Model.

In Table 11 we can see that combining predictive models reduces costs with under-provisioned and over-provisioned resources. Both on linear programming approach using the simplex method as the solution based on genetic algorithms (GA) showed better results than the use of a single prediction model in isolation. However, between the two, the solution based on genetic algorithms was the one with the best result.

Another important point to be considered is the overhead raised by the models to make a prediction. After all, a very high overhead could make the solution unfeasible. Those overhead were taken from experiments conducted and were detailed in Section 5.4. We can see in the Table 11 that the overhead raised by solutions based on combination of predictive models is superior to the overhead of solutions based on a single model. However, still, the overhead of around 400 ms, is extremely low considering that a prediction will be made only every one hour time interval. Therefore, this does not make in any way, the proposed solution unfeasible.

Finally, we can analyze the difficulty of implementing the proposals. Of course that solutions based on the combination of models has a slightly higher difficulty of implementing than the solutions based on a single model. But even so, this difficulty can be considered low. That is due to the fact that both the Simplex algorithm as the use of genetic algorithms are widely described and documented in the literature. Thus, it is very easy to find professionals able to work with such technologies.

According to the data presented above, we can conclude that our proposal is feasible to solve the resource allocation problem for Web applications hosted on the cloud infrastructure. Compared to the linear programming based solution and the solutions based on a single forecast model, our solution (GA) was the one that proved the most suitable to resolve the issue.

Table 11: Estimated Costs per Year

Average Cost per Year (estimate)				
	Series			
Model	ClarkNet	Nasa	World Cup	Total
NAIVE	4394	968.5	59964.67	21775.72
AR	4134	978.25	64735.67	23282.64
ARMA	3757	945.75	68191.5	24298.08
ARIMA	3978	910	67348.67	24078.89
ETS	3666	949	67264.17	23959.72
Simplex	3900	942.5	60246.34	21696.28
GA	3601	919.75	60057.84	21526.2

5.7 Final Remarks About Results

We can notice by the results that there is not a better predictive model for all cases. The NAIVE model, for example, had the best result for the FIFA world cup series and the worst for the other series. Similarly, the ARIMA model had the best performance for the Nasa series and the worst for the ClarkNet series. Our proposed method had good results in all scenarios and series used. It showed itself a generic and efficient model, able to adapt to each series and scenery.

The results show that the higher the rate of processing of the server and the lower the response time is, less error will be in the allocation of resources. This rule applies to all models and series. From this premise, we can conclude that, when decrease the server processing rate and maximum response time (SLA), the prediction accuracy becomes more important.

The models evaluated, in general, showed low error rates in resource allocation, especially for ClarkNet and NASA series. For FIFA World Cup series, the error rate was higher than in the two other series analyzed. However, the results can not be considered bad at all. The Fifa World Cup series presents high variability of values, which makes it difficult to forecast. We have also to consider that the values achieved by the series is of the order of thousands, unlike the other two where the values achieved are on the order of tens.

Regarding the metrics used, MEI metric is presented as the best option to evaluate the elasticity of auto-scaling techniques. Our proposed metric was less sensitive to outliers, rewarding the model which had the best regularity.

6 Conclusion and Future Works

The focus of this work was to conduct a study on the use genetic algorithm to combine classical statistical models in predicting demand for web applications hosted in cloud infrastructure. The demand forecast is important because reactive techniques are not able to cope with variations in demand without causing system instability. This is due to the fact that the allocation of resources in the cloud infrastructure is not instantaneous, and there is a delay until the resource is ready for use. It is therefore necessary to predict the future demand, to request the necessary resources in advance. However, choosing the best prediction model is a difficult task. As seen from the results, there is not a better prediction model for all cases. A model that has a good performance for a given time series can have a bad performance to another. The task becomes even more complicated when the user does not have much historical data to analyze. For example, a company that is starting a business and want to host your Web application in the cloud. Our proposal strikes precisely this problem, because our proposed method is able to adapt to various types of time series, and does not need a lot of historical data to work, because it does not require a previous training phase, being able to adapt the extent to which the data is coming. Thus, our method is generic enough to be used by any user wishing to host your Web application in the cloud and be guaranteed to get a good result. It was also shown that the smaller the parameters: server processing rate and maximum response time, becomes more important to choose a good predictive model. Finally, the MEI metric proposed in this work, is effective in evaluating the performance of the auto-scaling techniques, and can be useful for future work, since there is a lack of metrics to measure elasticity in the cloud. For future work will consider the use of a cost model to optimize the fitness function. Also, we will consider other kinds of crossover and mutation. Other forecasting models may also be considered as well.

Acknowledgements We would like to thank the Federal Institute of Sao Paulo (IFSP) and the Higher Education Personnel Training Coordination (CAPES) for financial support. The logs used in this work were obtained in [1].

7 Compliance with Ethical Standards and Disclosure of potential conflicts of interest

Our research did not involve human participants and neither animals.There are no conflicts of interest with reviwers, funding agencies, etc and neither financial support for the development of this work, that could have direct or potential influence or impart bias on the work. The reviewers are given the following countries: Spain, Italy, Germany, Australia and the Netherlands. None of the reviewers are from the same country or the same institution of the authors. The content of this paper are according with of ethical and professional conduct described by the Springer.

References

1. 1998 world cup web site access logs. http://ita.ee.lbl.gov/html/contrib/WorldCup.html. Accessed: 2014-10-15
2. Clarknet-http - two weeks of http logs from the clarknet www server. http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html. Accessed: 2014-10-15
3. Nasa-http - two months of http logs from the ksc-nasa www server. http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html. Accessed: 2014-10-15
4. Rightscale. set up autoscaling using voting tags. https://support.rightscale.com/. Accessed: 2015-01-10
5. Ali-Eldin, A., Tordsson, J., Elmroth, E.: An adaptive hybrid elasticity controller for cloud infrastructures. In: Network Operations and Management Symposium (NOMS), 2012 IEEE, pp. 204–212. IEEE (2012)
6. Arlitt, M., Jin, T.: A workload characterization study of the 1998 world cup web site. Network, IEEE **14**(3), 30–37 (2000)
7. Armbrust, M., Fox, O., Griffith, R., Joseph, A.D., Katz, Y., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: M.: Above the clouds: A berkeley view of cloud computing (2009)
8. Balaji, M., Rao, G., Kumar, C., et al.: A comparitive study of predictive models for cloud infrastructure management. In: Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, pp. 923–926. IEEE (2014)
9. Beloglazov, A., Buyya, R.: Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In: Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, p. 4. ACM (2010)
10. Box, G.E., Jenkins, G.M., Reinsel, G.C.: Time series analysis: forecasting and control. John Wiley & Sons (2013)
11. Braun, J., Murdoch, D.J.: A first course in statistical programming with R, vol. 25. Cambridge University Press Cambridge (2007)
12. Chieu, T.C., Mohindra, A., Karve, A.A.: Scalability and performance of web applications in a compute cloud. In: e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on, pp. 317–323. IEEE (2011)
13. Chieu, T.C., Mohindra, A., Karve, A.A., Segal, A.: Dynamic scaling of web applications in a virtualized cloud computing environment. In: e-Business Engineering, 2009. ICEBE'09. IEEE International Conference on, pp. 281–286. IEEE (2009)
14. Clark, T.: Quantifying the benefits of the rightscale cloud management platform. Fact Point Group Whitepaper, funded by Rightscale (2010)
15. Dantzig, G.B.: Linear programming and extensions. Princeton university press (1998)
16. Di Penta, M., Casazza, G., Antoniol, G., Merlo, E.: Modeling web maintenance centers through queue models. In: Software Maintenance and Reengineering, 2001. Fifth European Conference on, pp. 131–138. IEEE (2001)
17. Fernandez, H., Pierre, G., Kielmann, T., et al.: Autoscaling web applications in heterogeneous cloud infrastructures. In: IEEE International Conference on Cloud Engineering (2014)
18. Gross, D., Shortle, J.F., Thompson, J.M., Harris, C.M.: Fundamentals of queueing theory. John Wiley & Sons (2013)
19. Herbst, N.R., Huber, N., Kounev, S., Amrehn, E.: Self-adaptive workload classification and forecasting for proactive resource provisioning. Concurrency and Computation: Practice and Experience (2014)
20. Hyndman, R.J., Athanasopoulos, G.: Forecasting: principles and practice. OTexts (2014)
21. Hyndman, R.J., Khandakar, Y.: Automatic time series for forecasting: the forecast package for r. Tech. rep., Monash University, Department of Econometrics and Business Statistics (2007)

[1] http://ita.ee.lbl.gov/html/traces.html

22. Jiang, J., Lu, J., Zhang, G., Long, G.: Optimal cloud resource auto-scaling for web applications. In: Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on, pp. 58–65. IEEE (2013)
23. Kalyvianaki, E., Charalambous, T., Hand, S.: Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In: Proceedings of the 6th international conference on Autonomic computing, pp. 117–126. ACM (2009)
24. Kleinberg, J., Tardos, É.: Algorithm design. Pearson Education India (2006)
25. Lim, H.C., Babu, S., Chase, J.S., Parekh, S.S.: Automated control in cloud computing: challenges and opportunities. In: Proceedings of the 1st workshop on Automated control for datacenters and clouds, pp. 13–18. ACM (2009)
26. Lorido-Botrán, T., Miguel-Alonso, J., Lozano, J.A.: Auto-scaling techniques for elastic applications in cloud environments. Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09 **12** (2012)
27. Lorido-Botran, T., Miguel-Alonso, J., Lozano, J.A.: A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing pp. 1–34 (2014)
28. Math, C.: The apache commons mathematics library (2014)
29. Miller, M.: Cloud computing: Web-based applications that change the way you work and collaborate online. Que publishing (2008)
30. Padala, P., Hou, K.Y., Shin, K.G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A.: Automated control of multiple virtualized resources. In: Proceedings of the 4th ACM European conference on Computer systems, pp. 13–26. ACM (2009)
31. Roy, N., Dubey, A., Gokhale, A.: Efficient autoscaling in the cloud using predictive models for workload forecasting. In: Cloud Computing (CLOUD), 2011 IEEE International Conference on, pp. 500–507. IEEE (2011)
32. Wang, L., Xu, J., Zhao, M., Tu, Y., Fortes, J.A.: Fuzzy modeling based resource management for virtualized database systems. In: Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on, pp. 32–42. IEEE (2011)
33. Xu, J., Zhao, M., Fortes, J., Carpenter, R., Yousif, M.: On the use of fuzzy modeling in virtualized data center management. In: Autonomic Computing, 2007. ICAC'07. Fourth International Conference on, pp. 25–25. IEEE (2007)