Predicting Job Wait Time in Grid Environment by Applying Machine Learning Methods on Historical Information

Somayeh Kianpisheh, Saeed Jalili and Nasrolah Moghadam Charkari

Faculty of Electrical and Computer Engineering,
Tarbiat Modares University, Tehran, Iran
{s_kianpisheh, sjalili, moghadam}@modares.ac.ir

Abstract

To have high performance scheduling mechanisms in grid computing, we need accurate methods for estimating parameters like jobs' wait time and run time. In this paper, we consider wait time prediction problem. Different regression techniques are examined on AuverGrid data set to predict wait time. To improve the quality of prediction, some extra features are proposed. Simulation results show that adding these features reduces prediction error between 13% and 60% in different methods. Results also show that K-nearest neighbor outperforms other regression techniques. We have compared the k-nearest neighbor method in both original and enriched data set with Last-M. K-Nearest neighbor in enriched data set outperforms both Last-M and K-nearest neighbor in original data set in accuracy and perfect prediction percentage.

Keywords: *Grid computing; Wait time prediction; Machine learning*

1. Introduction

Grid computing technologies emerged within the scientific community with the objective of connecting different resources to provide powerful computational capabilities. Nowadays, it has gained popularity in research areas. For effective use of grid computational power, scheduling plays a critical role. Perfect predictions of the wait time and run time of jobs, lead us to high performance scheduling mechanisms [1]. Unfortunately, predicting these parameters is not trivial and researches have shown low accuracy in prediction [2, 3]. Thus, enhancing the accuracy of prediction is currently the main interest in many studies.

In this paper, we concentrate on wait time prediction and focus on AuverGrid [4] data set which recently has attracted researchers' attention. The related works can be divided into two categories. One that predicts wait time in various workload archives available for grid computing, and the second which concentrates on AuverGrid. We explain each part separately in the following;

A prediction mechanism based on instance based learning has been proposed in [5]. By storing historical information, for each query it returns the N nearest neighbors as the relevant experiences. Using a kernel regression in returned experiences, the wait time on TACC lonestar system archive is predicted. [6] Uses statistical techniques to predict job run times, and makes scheduler simulation to obtain job start times on grid sites. Some other approaches based on run time prediction have been presented in [7, 8]. The authors of [9] provide an upper bound with some degree of confidence for wait time. [10] Proposes exploiting multiple time series predictors to predict job response time. The predictors' estimation is validated according to the queue theory. Among predictors, the most probable predictor which makes the queue be equalized will be accepted. [11] Models each resource with queue and uses queue theory to estimate job wait time.

So far, little researches have been done on AuverGrid data set. [12] Proposes a mechanism to predict failure of jobs submitted to this production grid. An adaptive model has been presented for load prediction in [13, 14]. To predict wait time, [2] presents a similar approach to [6-8]. They evaluate some run time prediction methods on AuverGrid and choose the best predictor. Then based on run time prediction and simulating a scheduling policy, wait time is predicted. The disadvantage of these approaches is that they require detailed knowledge of scheduling algorithm. It seems that little works have been done for wait time prediction on this data set.

In contrast with other works to predict wait time, we emphasize on both job and site features. Using historical information, we model the problem as regression. Different machine learning techniques include linear and quadratic model, decision trees, support vector machines and k-nearest neighbor estimator will be evaluated. In order to reduce the prediction error, some extra features have been taken into account. Simulation results showed significant reduction in the error of prediction. We have also compared our approach with Last-M which is a generalized of Last-2 algorithm [15]. There are two contributions for this work. First, In contrast with other works which advocate one specific estimation method, we evaluate different regression techniques to predict wait time. Second, we propose some extra features which have significant effect on error reduction. The experiences show this effect in different methods. In Section 2, we describe the data set and its original features. In Section 3, the problem is formulized and the used learning methods are summarized. In this section, the proposed features are introduced. The experimental results have been demonstrated in Section 4, and finally conclusion is discussed in Section 5.

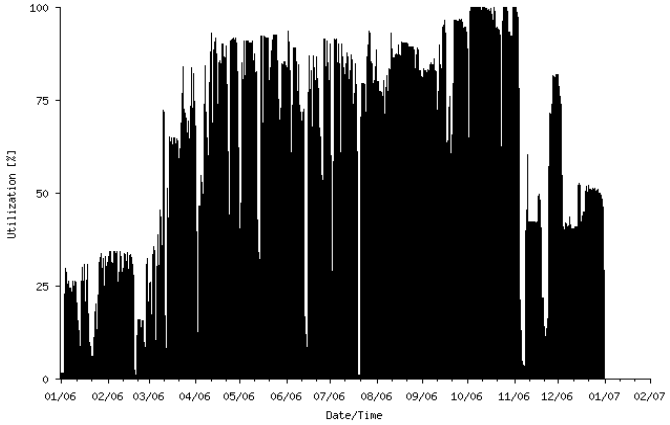


Figure 1. System Utilization in AuverGrid

2. AuverGrid

The Grid workload Archive contains traces of job execution for a number of grids. The traces have been recorded for a period of time in Grid Workload Format. Although according to this format a total of 29 attributes have been defined, not all of them are available. In our work we use workload traces of a well known multi site grid called AuverGrid [4]. This grid is a part of EGEE project and consists of 475 CPUs within 5 sites which are geographically distributed in the Auvergne region, France. The trace contains a total of 404,176 records of submitted jobs to AuverGrid from 1st of January 2006 till 1st of January 2007. The average system utilization is 58.48% (Figure 1). Table 1 shows job features we have used for wait time prediction.

3. Wait Time Prediction

The purpose of this paper is to predict wait time with high accuracy. Considering job features we can model the problem as regression and use machine learning techniques for estimation. In Section 3.1, we describe the learning methods. In the first phase of simulations, we observe that the accuracy with original features in the data set is not acceptable. Thus, to enhance the accuracy, we propose some extra features that can be derived from original features in the data set. Section 3.2 discusses on these extra features. The simulation results in Section 4 show significant improvement via adding these features.

Table 1. Original Features in AuverGrid

feature	Description
Submit time	The time job submitted to grid (original time is 1st of Jan 2006)
Average CPU time used	Average of the CPU time used during job execution
Used memory	Average of used memory per processor
Requested time	User runtime estimate
User ID	User identifier
Group ID	Group identifier
Executable ID	Identifier used for categorizing or describing the application
Queue ID	Queue identifier job has assigned to
Site ID	Site identifier job has executed on
Run time	Job run time
Wait time	**Queue waiting time before job execution**

*** Wait time is required to be predicted.**
**** Run time is not used in learning. However, it is used to drive extra features.**

3.1. An Overview of Used Learning Methods

We have examined some regression methods such as linear and quadratic models, decision tree (j.48), support vector machine and K-nearest neighbor. The effect of varying parameters like pruning factor, number of support vectors, kernel function and K has been considered. We also studied the effect of applying regression on principle components. In this Section, we shortly describe these methods [16, 17] express these algorithms in detail.

A. Linear/Quadratic regression

The most common approach in regression is to fit the data to a global parametric function. In linear regression, the value of a dependent variable like y is expressed as a function of some independent variables (x's) as below;

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i \quad (1)$$

The subscript i denotes the observations or instances; the second subscript designates p independent variables. There are $p + 1$ parameters, β_j to be estimated according to a fitting criterion. Very often, this is a least squares criterion, which minimizes the sum of the squares of the prediction errors for all instances. The error term ε_i denotes the error of estimation for each instance i, and it is assumed to be normally distributed. The parameters can be found by solving the equations as a result of taking derivation from error function with respect to parameters. The quadratic model can be defined in a similar way.

B. Decision tree

Tree induction algorithms construct the model by partitioning the data set. The task of constructing a tree is accomplished by employing a search to select an attribute(s) to be used for partitioning the data at each node of the tree. To avoid over fitting and form simpler models, pruning strategies are employed. Among different algorithms, we have chosen the simple and popular j.48 algorithm [17]. The splitting is carried in a recursive manner. According to formula 2, the goodness of a split is measured by the mean square error of the estimated value g_m from the actual value r^t. The estimation is computed as the mean of the outputs of instances X_m reaching the node *m*. The number of those instances is defined by N_m. In this formula, $b_m(\boldsymbol{x^t})$ defines if instance *t* reaches node *m*.

$$E_m = \frac{1}{N_m}\sum_t (r^t - g_m)^2 b_m(\boldsymbol{x^t}) \tag{2}$$

When the error is small enough a leaf node is created. Otherwise, data is split further such that the sum of the errors in the branches is minimized.

C. Support vector machine Regression

SVM was introduced in 1992 [18] as a pattern classification and regression technique. Assuming a linear regression model as the following:

$$f(\boldsymbol{x}) = \boldsymbol{w^T x} + w_0 \tag{3}$$

In support vector regression, the ε-sensitive loss function is defined as following:

$$e_{\in}\left(r^t, f(\boldsymbol{x^t})\right) = \begin{cases} 0 & if\ |y^t - f(\boldsymbol{x^t})| < \epsilon \\ |y^t - f(\boldsymbol{x^t})| - \epsilon & otherwise \end{cases} \tag{4}$$

Which means that we tolerate errors up to ∈. To account for deviations out of ∈ −zone, the slack variables are introduced and the optimization problem will be as bellow:

$$min\frac{1}{2}\parallel \boldsymbol{w} \parallel^2 + C\sum_t (\xi_+^t + \xi_-^t) \tag{5}$$

Subject to:

$$\begin{gathered} r^t - (\boldsymbol{w^T x} + w_0) \leq \in + \xi_+^t \\ (\boldsymbol{w^T x} + w_0) - r^t \leq \in + \xi_-^t \\ \xi_+^t, \xi_-^t \geq 0 \end{gathered}$$

It has been shown that this optimization problem can be written as Lagrange function and its dual can be taken. Kernel functions can also be used. The result will choose certain instances as support vectors and the regression line is computed as a weighted sum of them.

D. K-nearest neighbor

In instance-based regression, each instance is usually represented as a set of attribute value pairs. The problem to be solved is, for a given query instance, to predict the target value as a function of other instances whose target values are known. The K-nearest neighbor is the most popular instance-based algorithm. The target values of the K most similar neighbors are used in estimation. In this paper we have used the Euclidean distance between instances as similarity metric. However, other metrics can be used.

E. Regression On principle components

PCA [19] is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has as a high variance as possible and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to the preceding components. Hwang has showed that regression on principle components can reduce the mean square error [20]. Thus, in this paper we also have studied the effect of regression on principle components.

3.2. Proposed Features

According to the mentioned learning methods, the mean absolute error has been found about 5600 seconds in the best case, which is relatively high. To reduce the prediction error some new features from the original features, are introduced. Table 2 shows these features.

- Submit hour: This feature can easily be computed from submit time.
- CPU Intensity: This feature reflexes the amount of computation in the job. It can be computed as below:

$$CPU\ intensity = \frac{Average\ used\ CPU\ time}{run\ time} \tag{6}$$

- Queue load: This feature shows the load of queue when submitting job. It can be computed via a sequential scanning of data set from top to bottom (Fig.2).
- Site CPU utilization: This feature shows the CPU utilization in the site that job is going to be run. There are five sites which the total number of their CPUs has been shown in Table 3. In a similar manner to Queue load, the values for this feature can be computed via a sequential scanning. The general formula is as following:

$$Site\ CPU\ utilization = \frac{number\ of\ used\ CPUs\ in\ site\ at\ submit\ time}{number\ of\ total\ CPUs\ in\ the\ site} \tag{7}$$

We have chosen submit hour, since in some hours during the day the workload is high [21], and consequently, the jobs submitted during those hours will wait longer. Similarly, in hours with light workload, jobs will be served sooner. The more the job needs CPU, the more difficult is the providing of the required processing power. So, it makes sense to consider CPU intensity as a feature in wait time prediction. Obviously the queue load has a direct effect on wait time. And, at the end, the site CPU Utilization is important too. Whenever the CPU utilization is high, the jobs must wait more till computing units become idle. In the next section, we show how these extra features will significantly reduce the prediction error.

Table 2. Proposed Features Derived from Original Features

feature	description
Submit hour	The hour job submitted to grid
CPU intensity	CPU intension in the job
Queue load	Load of queue (at submit time)
Site CPU utilization	CPU utilization in the site job is going to be run (at submit time)

Table 3. Number of CPUs in Different Sites in AuverGrid

Site name	clrlcgce01	clrlcgce02	clrlcgce03	iut15	opgc
Number of CPUs	112	84	186	38	55

Sequential algorithm for computing Queue Load feature

```
1.  Initialize: Q1_Load = 0; Q2_Load = 0; …. Q13_Load = 0;
2.  Initialize sets: Q1_ExitTime = []; Q2_ExitTime = []; …. Q13_ExitTime = [];
3.  For each row in data set
4.      Current time = Job.Submit time;
5.      Queue = Job.Queue ID;
6.      Switch Queue
7.      Case 'Q1':
8.          Remove entries from Q1_ExitTime which are less than Current time;
9.          Q1_Load = Q1_Load – number of removed entries;
10.         Store Q1_Load as new feature for this row in AuverGrid data set;
11.         Job.Exit time = Job.Submit time + Job.Wait time;
12.         Add Job.Exit time to array Q1_ExitTime;
13.         Q1_Load = Q1_Load + 1;
14.     Case 'Q2':  //Similar to Q1
                .
                .
                .
15.     Case 'Q13': //Similar to Q1
16.     End switch
17. End
```

Figure 2. Pseudo Code for Computing Queue Load Feature

4. Results

To ensure the correctness of the results, the machine learning techniques have been done via Weka. For simplicity, we have chosen 25000 jobs randomly in the data set. 70% of these jobs (17500) have been randomly chosen as training set and the rest (7500) form the test set. The simulation has been done in two parts. In part 1, we examine different machine learning techniques and study the effect of the proposed features on the error of prediction. Then, the best method will be chosen for the prediction. In part 2, we compare the accuracy of prediction with a time series method called Last-M. In the following sections each part of simulation will be described separately.

4.1. Examining The Effect of Proposed Features

In this part of simulation, we examine different machine learning techniques described in Section 3 and study the effect of added features on the prediction error. Figure 3 shows the mean absolute error for different methods of the prediction. We have studied the effect of varying parameters. These parameters include confidence factor of pruning in decisions trees (0.25, 0.45, 0.65), kernel function (RBFKernel, PolyKernel), complexity parameter defining number of support vectors in SVM (1, 3, 5) and K in K-nearest neighbor method (1, 3, 5, 7, 11). As it can be seen, except for decision trees, in all methods proposed features has caused a reduction in the prediction error. The amount of reduction changes between 13% and 60% in different methods. Applying PCA before regression in linear model, decision tree and SVM has not improved the accuracy and only when extra features are considered in KNN (K = 1) there is a bit reduction. In general, it can be seen that the best result with the mean absolute error equals to 2255 seconds has been achieved when we apply K-nearest neighbor in the enriched data set.

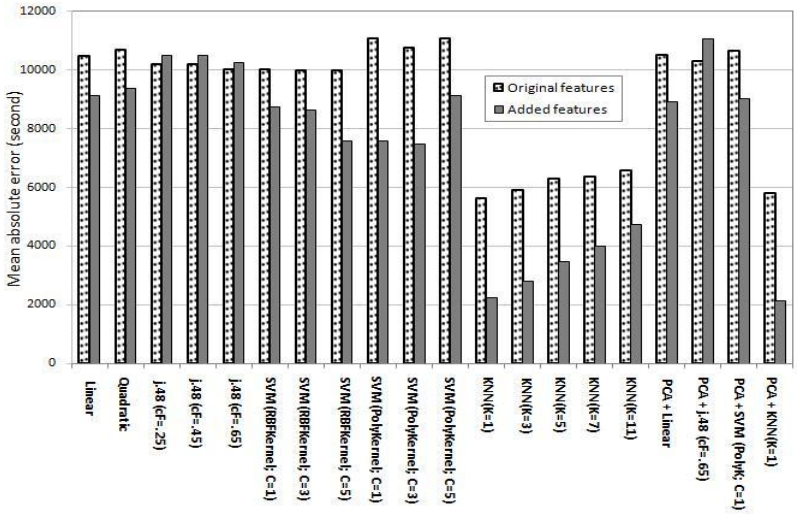


Figure 3. Mean Absolute Error for Different Methods of the Prediction. The Added Features caused an Observable Reduction in Error.

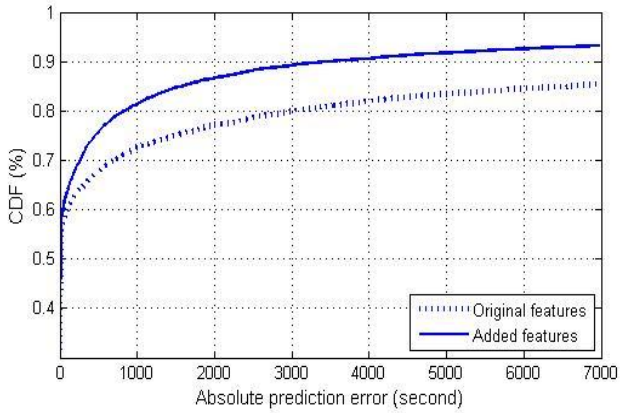


Figure 4. Cumulative Distribution Function of Absolute Prediction Error in KNN (K = 1)

Figure 4 shows the cumulative distribution function of absolute prediction error in KNN (K = 1). 80% of instances have error less than 1000 seconds, when proposed features are considered. However, this value reduces to 70% with original features. Thus, these features showed an observable reduction in error.

4.2. Nearest Neighbor Versus Time Series Predictor

Last-2 [15] is a time series predictor which has been introduced for estimating jobs' run time. It simply averages the runtime of the last two jobs in order to predict the current job's run time. It is found that Lats-2 has a good prediction capability. In this paper, we generalize this algorithm to Last-M and use it to predict waiting time. In order to predict the waiting time of a job in a specific queue like 'Q', it takes average of waiting times of last M completed jobs that had been assigned to 'Q'. As in most systems the queue status is the same during small periods, this method seems to be logical. In the rest of paper, we compare this method with the results of applying KNN (K = 1) in the data set with

original and added features. To make Last-M and K-nearest neighbor method comparable, we perform the experiments according to the following steps:

1. We choose 3 out of 13 queues: Q1 (26441 assigned jobs), Q4 (11506 assigned jobs) and Q2 (7808 assigned jobs) which demonstrates three sizes of available historical data instances, large, medium and small. Thus, we have three data sets.
2. For each data set, we apply Last-M to predict jobs' wait time.
3. 70% of each data set is used as training and the rest as test set. Then the K-nearest neighbor is applied.
4. The quality of prediction for instances of test sets in Last-M and K-nearest neighbor methods have been compared.

Figure 5 shows mean absolute error for these queues, when M varies in the range of 1 to 15. In all cases, applying KNN (K = 1) in the enriched data set (with added features) outperforms two other cases: Last-M and applying nearest neighbor in original data set. Note that only Last-M results depend on M. As a result of adding proposed features, we will have approximately 5100, 10300 and 7300 seconds reduction in mean absolute error for Q1, Q4 and Q2 respectively when considering the best M for each queue. As size of training set decreases, the estimation error in original data set will increase. In the other hand, there is no observable change in the error of enriched data set. This emphasizes on the usefulness of added features when there are small history information available.

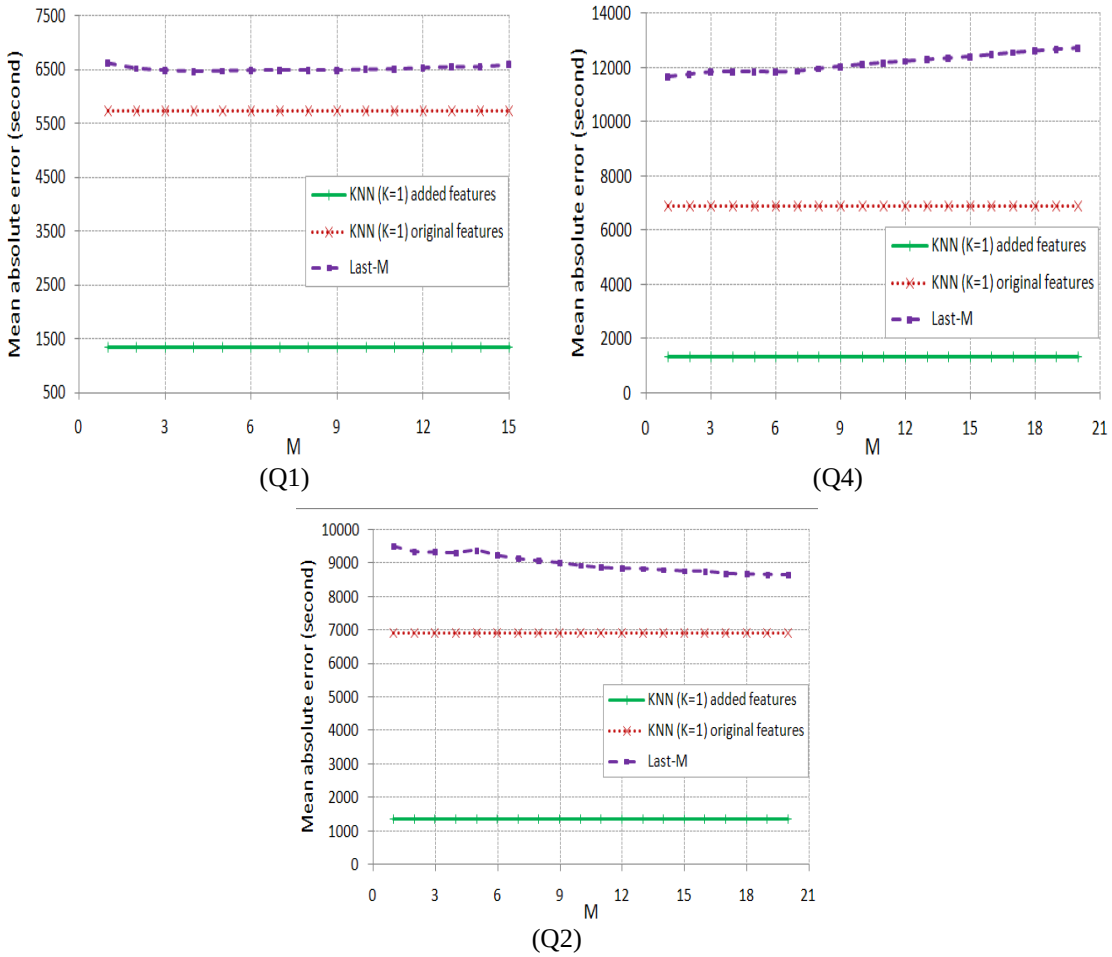
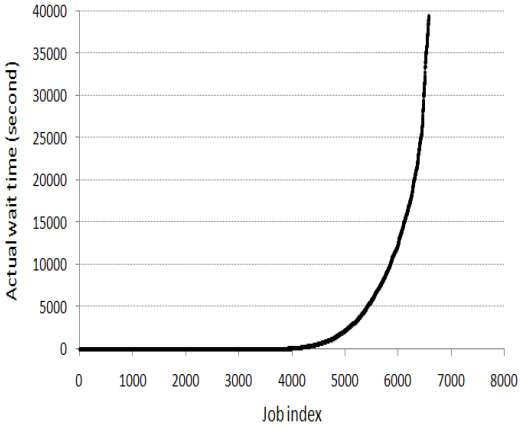
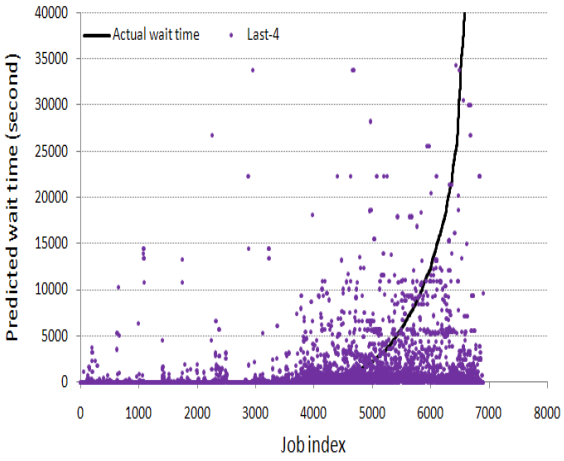


Figure 5. Mean Absolute Error Changes when Varying M

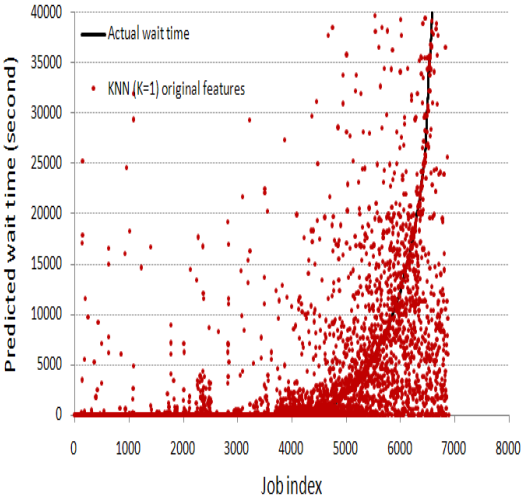
Figure 6 demonstrates the difference between the actual and the predicted wait time in Q1. The jobs in the test set have been sorted by their actual wait time. Thus, form a quadratic shape as shown in part (a) of the figure. In Figure 6, parts (b), (c) and (d) show the predicted wait time in Last-4 and nearest neighbor in original and enriched data set respectively. We have chosen Last-4 since in Q1 it shows the best performance (Figure 5). As it can be seen, the distribution of predicted points in Last-4 and K-nearest neighbor in original data set is relatively far from quadratic shape. However, proposed features have made the distribution of the predicted points close to the actual distribution.



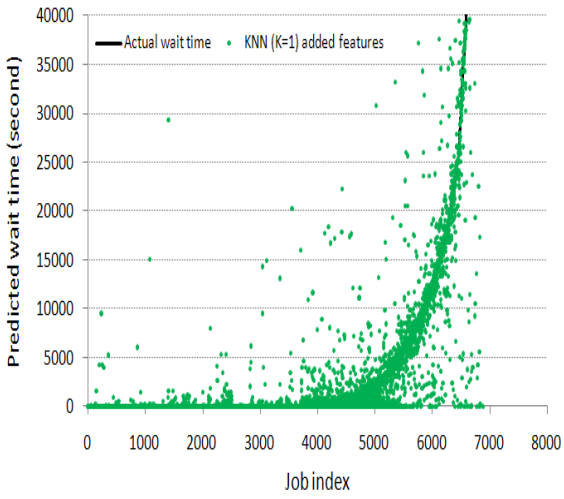
(a) Actual wait time



(b) Last-4



c) KNN (K=1) with original features



(d) KNN (K=1) with proposed features

Figure 6. The Difference between Actual and Predicted Wait Time in Q1. Jobs have been Sorted by Actual Wait Time. (a) Actual Wait Time (b) Last-4 Prediction (c) Nearest Neighbor in Original Data Set (d) Nearest Neighbor in Enriched Data Set

To measure the quality of the prediction more precisely, we define the accuracy of the prediction as the following formula:

$$accuracy = \begin{cases} 1 & if\ P = A \\ \frac{P}{A} & if\ P < A \\ \frac{A}{P} & if\ A < P \end{cases} \quad (8)$$

Where P and A is the predicted and actual wait time respectively. In this way, the mean accuracy would be the average of the accuracy of all predictions over test set. The mean accuracy and the amount of perfect, under and over prediction in estimating wait time for data points of three queues have been shown in table 4. For each queue, the best M according to Figure 5 has been chosen. In average, the proposed features have enhanced the accuracy about 0.12 in nearest neighbor method. In all queues, the accuracy of Last-M is the least. The K-nearest neighbor (K=1) in enriched data set has the highest amount of perfect prediction. Thus, as a whole enriching the data set by proposed features and using nearest neighbor as a method of the prediction can enhance the quality of prediction.

Table 4. Prediction Quality in Estimating Wait Time for Data Points of Three Queues

		Mean accuracy	Perfect prediction	Under prediction	Over prediction
Q1	**KNN(K=1); added features**	0.67	31.5%	34.6%	33.8%
	KNN(K=1); original features	0.58	31%	35.1%	33.9%
	Last-4	0.43	14.5%	45.5%	40%
Q4	**KNN(K=1); added features**	0.72	30.6%	36%	33.4%
	KNN(K=1); original features	0.56	28%	36.5%	35.5%
	Last-1	0.39	28.7%	55.6%	15.57%
Q2	**KNN(K=1); added features**	0.71	35%	33.5%	31.5%
	KNN(K=1); original features	0.61	34.5%	33.7%	31.7%
	Last-20	0.37	1.5%	36.1%	62.4%

5. Conclusion

In this paper, a study on predicting job wait time has been carried out. We examined the AuverGrid data set and chose some features to estimate wait time via regression. Different learning machine methods including linear and quadratic regression model, decision trees, support vector machine and K-nearest neighborhood have been used for prediction. As the prediction accuracy was low four new features have been proposed to improve the quality of prediction. Simulation results show that considering these features which can easily be derived from original features, make a significant reduction in the prediction error. Simulations also showed that K-nearest neighbor (K=1) outperforms other methods in prediction. We compared the nearest method in both original and enriched data set with a time series method called Last-M. Different values of M have been studied and the results showed the outperformance of the nearest neighbor. We also have shown how adding proposed features makes the distribution of predicted points close to their actual distribution. As a whole, enriching the data set by adding our proposed features and using nearest neighbor as a method for prediction showed a good improvement in both mean accuracy and perfect prediction.

References

[1] F. Guim and J. Corbalan, "A Job Self-scheduling Policy for HPC Infrastructures", 13th International Workshop on Job Scheduling Strategies for Parallel Processing, **(2008)**, pp. 51-75.
[2] O. Sonmez, N. Yigitbasi, A. Iosup and D. Epema, "Trace-Based Evaluation of Job Runtime and Queue Wait Time Predictions in Grids", 18th ACM international symposium on High performance distributed computing **(2009)**.
[3] W. Smith, "A Service for Queue Prediction and Job Statistics", Workshop on Gateway Computing Environments, **(2010)**, pp. 1 - 8.
[4] http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Workloads.Gwa-t-4.
[5] W. Smith, "Prediction Services for Distributed Computing", Symposium on Parallel and Distributed Processing, **(2007)**.
[6] L. Hui, D. Groep, J. Templon and L. Wolters, "Predicting job start times on clusters", International Symposium on Cluster Computing and the Grid, **(2004)**, pp. 301 - 308.
[7] W. Smith, V. Taylor and I. Foster, "Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance", Workshop on Job Scheduling Strategies for Parallel Processing, **(1999)**.
[8] A. B. Downey, "Using Queue Time Predictions for Processor Allocation", workshop on Job Scheduling Strategies for Parallel Processing, **(1997)**, pp. 35-57.
[9] J. Brevik, D. Nurmi and R. Wolski, "Predicting Bounds on Queuing Delay for Batch-scheduled Parallel Machines", symposium on Principles and practice of parallel programming, **(2006)**.
[10] M. Tao, S. Dong and L. Zhang, "A multi-strategy collaborative prediction model for the runtime of online tasks in computing cluster/grid", J. Cluster Computing, vol. 14, **(2011)**, pp. 199–210.
[11] G. Tian, C. Xiao, X. Xu, C. Gao, Nuslati and Mardan, "Grid Workflow Scheduling Based on Time Prediction of Queuing Theory", Conference on Information and Automation, **(2010)**, pp. 2036-2039.
[12] H. Fadishei, H. Saadatfar and H. Deldari, "Job failure prediction in grid environment based on workload characteristics", Computer Conference, **(2009)**, pp. 329 - 334.
[13] Y. Yuan, Y. Wu, G. Yang and W. Zheng, "Adaptive Hybrid Model for Long Term Load Prediction in Computational Grid", International Symposium on Cluster Computing and the Grid, **(2008)**, pp. 340 - 347.
[14] Y. Wu, K. Hwang, Y. Yuan and W. Zheng, "Adaptive Workload Prediction of Grid Performance in Confidence Windows", J. IEEE Transactions on Parallel and Distributed Systems, vol. 21, **(2010)**, pp. 925-938.
[15] D. Tsafrir, Y. Etsion and D. G. Feitelson, "Backfilling Using System-Generated Predictions Rather than User Runtime Estimates", J. IEEE Transactions on Parallel and Distributed Systems, vol. 18, **(2007)**, pp. 789–803.
[16] L. Uysal and H. A. Guvenir, "An overview of regression techniques for knowledge discovery", J. The Knowledge Engineering Review, vol. 14, **(1999)**, pp. 319-340.
[17] E. Alpaydin, "Introduction to Machine Learning", The MIT Press, London, England, **(2004)**.
[18] B. E. Boser, I. M. Guyon and V. N. Vapnik, "A training algorithm for optimal margin classifiers", Workshop Comput. Learning Theory, **(1992)**, pp. 144–152.
[19] I. Jolliffe, "Principal Component Analysis", Springer, New York, **(2001)**.
[20] J. T. G. Hwang and D. Nettleton, "Principal Components Regression With Data Chosen Components and Related Methods", J. Technometrics, vol. 45, **(2003)**, pp. 70-79.
[21] Y. Fei, J. Changjun, D. Rong and Y. Jianjun, "Grid resource management policies for load-balancing and energy-saving by vacation queuing theory", J. Computers and Electrical Engineering, vol. 35, **(2009)**, pp. 966-979.

Authors

Somayeh Kianpisheh is currently a Ph.D. student in software computer engineering at the Faculty of Electrical and Computer Engineering in Tarbiat Modares University, Tehran, Iran. She received her B.S and M.S degrees in computer engineering from Tehran and Tarbiat Modares University in 2004 and 2010, respectively. Her main research interests include grid computing, performance modeling and task scheduling.

Saeed Jalili received his Ph.D. in computer science from Bradford University, Bradford, England in 1991. He is currently an associate professor in faculty of electrical and computer engineering, Tehran, Iran. His main research interests include machine learning, network security, unconventional computing and software runtime verification.

Nasrolah Moghadam Charkari received his B.S. in computer science at Shahid Beheshti University, Tehran, Iran in 1986 and his M.S. and Ph.D. in computer engineering and information systems engineering from Yamanashi University, Japan in 1992 and 1995, respectively. Currently, he is an assistant professor in faculty of electrical and computer engineering, Tehran, Iran. His main research interest include distributed systems, computer vision and image processing.