An Evaluation of Learning and Heuristic Techniques for Application Run Time Predictions

Hui Li[*] David Groep[†] Lex Wolters[*]

[*]Leiden Institute of Advanced Computer Science (LIACS), Leiden University
2333 CA, Leiden, The Netherlands

[†]National Institute for Nuclear and High Energy Physics (NIKHEF)
1098 SJ, Amsterdam, The Netherlands

E-mail: hli@liacs.nl davidg@nikhef.nl llexx@liacs.nl

Abstract

This paper evaluates several main learning and heuristic techniques for application run time predictions on clusters and parallel supercomputers. Techniques being investigated are based on mining the similarities in historical workload traces. Firstly several feature selection algorithms are introduced and compared, such as Rough Set based "Improved Reduct" and Genetic Algorithms. These algorithms are able to select relevant attributes that identify similar jobs in the past. Secondly two induction methods, namely Similarity Templates and Instance Based Learning (IBL), are empirically evaluated for prediction generation. Our evaluation is based on real workloads with diverse characteristics, which are collected from the LHC Computing Grid, the DAS-2 research cluster, and San Diego Supercomputing Center. The experimental results indicate that more computationally efficient methods such as Improved Reduct is able to achieve comparable results in terms of prediction accuracy. Moreover, the simple Similarity Templates approach, when combined with proper feature selection algorithms, outperforms more sophisticated IBL algorithms on our workloads.

1 Introduction

Application run time is an important performance metric in a heterogeneous Grid environment and many techniques have been proposed for its prediction. In the literature, there are two major categories of solutions to the run time prediction problem. The first category of techniques attempts to build application models and machine profiles and combine them to produce an estimate [5, 6]. This approach has the disadvantage of requiring direct knowledge of the internal design of the algorithm or the machine. It also lacks a proven mechanism for producing a run time prediction from the application and machine profiling data over a wide range of algorithms and architectures [13]. Nevertheless, this approach could generate relatively more accurate predictions for the targeted applications as more specific information is incorporated and analyzed.

The second category of techniques is to derive predictions from the historical performance data or workload traces. For time-sharing Unix machines, Dinda has extensively investigated time series models for host load predictions and it is shown that the relationship between the host load and the execution time is almost linear [9]. For the space-shared supercomputers and clusters, however, different techniques have been investigated as the workload data is in the feature domain. This paper focuses on evaluating a variety of learning and heuristic techniques for application run time predictions using workload traces.

It is believed that the knowledge about resource or application properties can be discovered through historical behavior. We adopt a data mining approach for the run time prediction problem and the predictions are made by mining the similarities in the performance data. We select two algorithms for feature selection. One is a modified version based on Rough Set Reduct algorithm [] (so-called "Improved Reduct"), the other is based on Genetic Algorithms []. These feature selection methods can be combined with simple induction models like mean or linear regression for prediction generation, which is known as the "Similarity Templates" approach. We also use feature selection and other techniques to improve induction algorithms from Instance Based Learning [3]. The proposed techniques are empirically evaluated on three real-world workloads with diverse characteristics, which are collected from the LHC Computing Grid (LCG), DAS-2 research cluster (DAS), and San Diego Supercomputing Center (SDSC).

The rest of the paper is organized as follows. Section 2 discusses the related work in application run time predictions using workload traces. Section 3 describes our selected prediction techniques, which include the feature selection algorithms and the induction methods. Section 4 gives an extensive empirical evaluation of prediction techniques based on the real-world workloads. Conclusions and future work are presented in Section 5.

2 Related Work

Statistically application run time in the workload traces is distributed with heavy tails and shows a high level of self-similarity [10]. This means that run times vary significantly across all time scales and global models such as non-parametric regression or multi-layer sigmoidal neural networks are not likely to work. Local models that exploit the internal structure of workloads become a more viable solution. Workloads typically consist of jobs from a pool of users with different activity levels. Same users tend to run same or similar applications many times, resulting in highly correlated run times for the same users. There are also other job attributes or features (e.g. job name and number of processors) that can be utilized to partition workloads into a more fine-grained level. The key question here becomes how to efficiently identify and select the relevant features and exclude the irrelevant ones.

Different feature selection techniques have been proposed in the literature. In [11, 21], features are predefined manually by expert information, upon which templates are constructed to categorize similar jobs. Compared with manual selection, automatic techniques are shown to be more generic and be able to achieve higher prediction accuracy. In [25], greedy search and Genetic Algorithms (GA) are empirically studied for automatic template definition. GA is able to heuristically search through the template space and is shown to be a preferable method. In [18], a Rough Set technique based on Hu's Reduct algorithm [12] is introduced for automatic attribute selection. This approach is able to take the dependency between run time and other attributes into account during attribute selection. It can obtain good results in general, although the comparison with GA shows varying performance on different workloads.

Techniques based on Instance Based Learning (IBL) algorithms have also been investigated in run time predictions [15]. IBL, or local learning techniques, use historical data near the query point to build a local model for approximation. A proper distance metric has to be defined to measure the "nearness" between data instances. Feature selection or instance caching techniques are important enhancement techniques to make IBL practically useful. For instance, via feature selection a better distance metric can be formed as the irrelevant attributes have been filtered out. Via instance caching the learning rate can be greatly accelerated by removing irrelevant inputs [14]. We elaborate our selected prediction techniques in the next section.

3 Prediction Techniques

In this section the prediction techniques are introduced as follows. Firstly two feature selection algorithms, namely Improved Reduct and Genetic Algorithms, are introduced. The proposed Improved Reduct method is described in more detail. Secondly two induction techniques, the Similarity Templates and Instance Based Learning (IBL), are discussed briefly. The corresponding improvements are also introduced.

3.1 Feature Selection

3.1.1 Improved Reduct

The first algorithm we introduce for feature selection is Improved Reduct, which is based on Rough Set theory [17]. Firstly several important notions are defined to facilitate algorithm description. The workload traces can be represented as *information systems* or *tables*, where rows represent objects (case, event, etc) and columns represent attributes (or features). The set of attributes can be further classified into two disjoint subsets, application run time is called *decision attribute* and the rest serves as *condition attributes*. *Degree of Efficiency* (DoE) represents how good a subset of condition attributes can categorize data objects in an information system and it is defined as the weighted sum of CoV (Coefficient of Variance) of categorized decision attribute values. DoE is a quantitative measure of dependency between a subset of condition attributes and decision attribute. Condition attribute a is said to be *incrementally useful* to decision attribute D with respect to condition attribute subset R if the DoE of set $\{a\} \cup R$ is smaller than that of just set R. The *Significant Value of Attributes* is a quantitative measure on how incrementally useful of attribute a to decision attribute D with respected to condition attribute set R, defined as $DoE(R, D) - DoE(R + \{a\}, D)$ and denoted as *SVA*(a, R, D).

With the definitions above, the Improved Reduct algorithm is presented as follows:

Algorithm 1 (Improved Reduct) *Compute the best condition attribute subset with respect to Degree of Efficiency of the decision attribute set*

Input: An information system S, with attributes A being classified into condition attributes C and decision attributes D

Output: A set of condition attributes R (reduct set)

Steps

1. $R = \emptyset$ and $C^{'} = C$;

```
2. Compute the significant value for each attribute $a \in C^{'}$
   with respect to $R$, sort the set of attributes $C^{'}$ descend-
   ingly based on SVA values;

3. WHILE $DoE(R, D) - DoE(C, D) > Threshold$ DO
   Select an attribute $a$ in $C^{'}$ with the highest significant
   value;
   $R = R \cup \{a\}, C^{'} = C^{'} - \{a\}$;
   Recalculate the significant value of attributes in $C^{'}$
   with respect to $R$ and sort the values descendingly;
   compute $DoE(R, D)$;
   END WHILE
```

The algorithm starts with an empty reduct set. Significant values for condition attributes are calculated and sorted. A hill-climbing forward selection method is then applied to iteratively add the most significant attribute to the reduct set. At every round the significant values of attributes are recalculated with respect to the current reduct set. The selection process stops when the reduct set reaches or exceeds the level of clustering efficiency as the whole condition attribute set. Our algorithm adapts the original Reduct algorithm in [12] to a clustering problem with the introduction of Degree of Efficiency. We then improve the search process to be able to incrementally select attributes based on their incremental usefulness with respect to the previously selected attributes. The resulting reduct set is a subset of relevant condition attributes with descendant "significance".

In terms of search method the Improved Reduct algorithm is very similar to greedy search in [25], both adopt decedent forward selection. The main difference reside on the evaluation criteria: Improve Reduct uses the weighted average of CoV (DoE) to evaluate the quality of conditional attributes while greedy search does it via actually predicting the run times. This makes Improved Reduct computationally more efficient than the greedy counterparts.

3.1.2 Genetic Algorithms

The second feature selection algorithm we consider is Genetic Algorithms (GA). As a heuristic search technique based on the mechanics of natural selection and genetics, GA is a popular choice in the feature selection literature [4, 22] and has been applied in run time predictions [25]. A genetic algorithm typically evolves a population of individuals over a sequence of generations. We define an *individual* as a subset of condition attributes. Each attribute can be encoded into one or more bits, depending on whether it is a selection or weighting task. For instance, GA-1 (GA-i, i - number of bits) is equivalent to feature selection. GA-2 has four different weights for each attribute, which are equally spaced from 0 to 1 (namely, $0, 1/3, 2/3, 1$). Using i bits will result in 2^i discrete weights being evaluated. Evolving a generation of individuals to the next requires a set of operators, which we choose regularly used *selection*, *mutation*, and *crossover*. The fitness of the individual is evaluated by the *prediction accuracy* using the subset represented by the individual. The genetic search is halted on certain predefined maximum generation number or the fitness of all individuals reaches consensus. Our GA implementation is based on [28]

3.2 Prediction Generation

3.2.1 Similarity Templates

Similarity Templates is a simple yet efficient approach for run time predictions. Templates that consist of selected features are used to cluster jobs into categories. Jobs in one category are considered similar and induction algorithms are applied to generate predictions for future jobs in the same category. Simple induction algorithms such as *Windowed Mean* and *Linear Regression* have shown good performance/accuracy tradeoff. The simplest model - *WM(1)*, namely using the last available run time as prediction, is found to be the best model on many workloads.

Proper adaptive strategies can further improve the prediction accuracy and should be included in the algorithm. For instance, the most recent job entries in the category should be used in prediction and the feature selection process should be automatically redone after certain period, so the algorithm can adapt with *time*. Instead of using one global model for all categories, we can match an appropriate model for each category so it adapts with *model*. Moreover, if there are more attributes in the template, the workload will be partitioned into more fine-grained categories thus less data will be in each category. Chances that no valid predictions will be derived are higher in finer granularities. Given the relevant significance of attributes, as can be obtained by Improved Reduct, the attribute with the least significance can be automatically removed and a coarser grained template is formed until a valid prediction is made. By using this mechanism the algorithm can adapt with *template* to minimize the overall prediction errors.

3.2.2 Instance Based Learning

Instance based learning (IBL) algorithms are empirically studied for run time predictions by Kapadia et al. [15] for limited datasets. Combined with instance caching and editing techniques, IBL is a practically powerful tool to predict application run times. Not like other global models, IBL algorithms try to build a local model for historical data "near" the query point. By measuring nearness, a proper distance metric should be defined. For instance, a normalized Heterogeneous Euclidean-Overlap Metric (HEOM) [26] is suitable as it can handle both numeric and nominal attributes.

Workloads	Format	System	#CPUs	Period	#Jobs	Attributes
LCG	openPBS	Linux cluster	288	Apr - Sep, 2004	127920	(g, u, q, e, n, r, eq)
DAS	openPBS	Linux cluster	144	Jun - Dec, 2003	114043	(g, u, q, e, n, r)
SDSC	Standard	IBM SP	1152	Mar - Dec, 2002	67742	(g, u, q, e, n, r)

Table 1. Characteristics of workload traces. *sdsc* **is in standard workload format and is obtained from [24]. Attributes: g - group name, u - user name, q - queue name, e - job name, n - requested CPUs, r - requested run time, eq - GHz equivalent of allocated CPUs.**

Three IBL induction algorithms are considered in this paper, namely k-Nearest-Neighbor (kNN), Weighted Average (WA), and Linear Locally Weighted Regression (LLWR). kNN predicts the run time by using an unweighted average of the nearest k instances as defined by the distance metric, whereas WA uses a weighted average. LLWR tries to minimize a weighted sum of the squared errors to determine the linear parameters, in which weights are calculated as distances smoothed by a kernel function. Theoretically more sophisticated LLWR algorithm can track polynomial surfaces with smaller errors than k-Nearest-Neighbor and Weighted Average [3].

The basic IBL algorithms have to be improved to be practically useful. We employ several techniques to reduce search time and to improve prediction accuracy. Firstly the proposed feature selection methods can serve as a preprocess phase to IBL so only the relevant features are considered in the distance metric. Secondly instances in the historical workloads should be pruned to reduce search time. We introduce two levels of instance reduction. The first level is similar to IB2 [2], which incrementally incorporates incorrectly predicted instances into the historical base. This reduction technique is, however, very sensitive to noise. The second level of reduction is what we call "scoped search", where search is limited to a certain scope, such as in the category defined by the most significant attribute (e.g. "user name"). This technique is proved to be very efficient in reducing search time while at the same time maintaining the same level of prediction accuracy. Other instance reduction techniques are surveyed in [27].

The distance-based IBL algorithm is intrinsically a generalization of the similarity template, in which distances are simplified to binary values (belong or not belong to a specific category defined by the template). In many real world situations, same instances will be selected for prediction by both approaches as jobs in a fine-grained category will most likely score high in the IBL distance measurement. How well the technique performs when no valid prediction can be derived from a desired category becomes a very important factor in terms of prediction accuracy. The experimental results and analysis will be provided in next section.

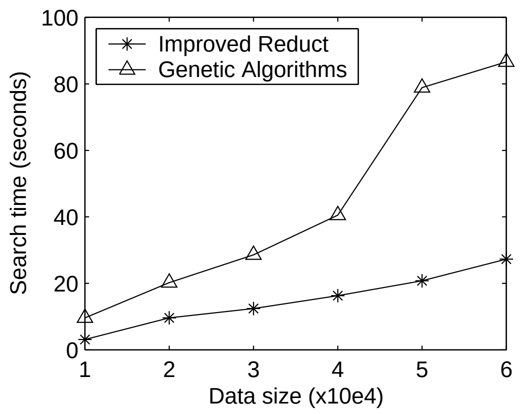

Figure 1. Search times of feature selection by Improved Reduct and Genetic Algorithms ($WM(1)$ is used as the induction model in Genetic search. Executions are performed on a Pentium III 933MHz Linux workstation).

4 Empirical Evaluation

The workload traces that we use for evaluation are summarized in Table 1. These workloads have diverse characteristics in terms of job entries and available attributes. LCG is collected from the production cluster in NIKHEF LCG [19] testbed. The system contains a mixture of Intel PIII, Xeon, and AMD K7 work nodes with different clock speeds. Most of the groups and users are from the High Energy Physics community. DAS is from the biggest cluster (at Vrije Universiteit) of the DAS-2 supercomputer [8] in the Netherlands, which is primarily used for computer science research. SDSC is obtained from the Parallel Workload Archive [24].

We use three types of metrics to measure the performance of techniques. *Prediction accuracy* is measured by the average prediction error divided by the average run time. *Search time* shows the average time needed for a feature selection technique to search for the best feature subset. *Prediction time* is calculated as the average execution time in milliseconds per prediction. The workload datasets are di-

Workload Name	Improved Reduct		Genetic Algorithms	
	Best Feature Set	Prediction Error	Best Feature Set	Prediction Error
LCG	(u, e, r)	0.272	(u, q, e, r)	0.265
DAS	(u, r, n, e, q)	0.529	(g, u, q, e, n, r)	0.529
SDSC	(u, r, n)	0.388	(u, q, n, r)	0.386

Table 2. Comparisons of Improved Reduct and Genetic Algorithms. Prediction error is measured by average prediction error divided by average run time. Attributes: g - group name, u - user name, q - queue name, e - job name, n - requested CPUs, r - requested run time.

Workload Name	Average Prediction Error / Average Run Time			
	Without TA (requested)	With TA (requested)	Without TA (last)	With TA (last)
LCG	0.454	0.299	0.298	0.299
DAS	1.98	0.734	0.548	0.693
SDSC	0.521	0.508	0.439	0.484

Table 3. Comparison of prediction accuracy with and without Template Adaption (predictor: Improved Reduct + $WM(1)$. "Requested" and "last" means that *requested run time* **and** *last available run time* **is used as estimation when no template can derive valid predictions, respectively).**

vided into independent training and test sets throughout the evaluation. Our empirical evaluation is conducted to answer the following questions.

1. How different feature selection techniques, namely, Improved Reduct and Genetic Algorithms, perform for the Similarity Templates approach?

Table 2 shows the best feature subsets found by the Improve Reduct and Genetic Algorithms for the similarity template approach. Both techniques are able to eliminate redundant attributes in the condition attribute set. In Improved Reduct, the measuring of relevant significance of attributes leads to more features being ruled out. For instance, on *lcg* queue name and requested run time have the same clustering efficiency during certain period, since users typically don't specify requested run times and use the queue default wall time instead. While during some other period, users do supply requested run times for their jobs. This makes requested run time relatively more important than queue name so only requested run time is selected. In terms of prediction accuracy which is also shown in Table 2, Improved Reduct achieves comparable prediction accuracy as Genetic Algorithms. It is also more computationally efficient in terms of search time, which is shown in Figure 1.

2. How effective are various adaptive strategies in improving prediction accuracy?

Improved Reduct is able to rank the selected attributes according to their relevant significance and the ranking of attributes can be used in template adaption as is introduced in section 3.2. Table 3 shows the comparison of prediction accuracy with and without template adaption. When the requested run time is used as estimation when no valid prediction is made by the templates, we can see that template adaption is able to reduce the prediction errors, especially on the research workload DAS where huge variations exist. However, when the last available run time is used, the best results are achieved mostly without template adaption. The reason could be that the last available value is itself a better estimator than the coarse-grained templates and it can generalize better. To sum up, the best similarity template approach is formed by combining Improved Reduct with Windowed Mean, and use last available run time as estimation if necessary.

3. What is the performance of IBL-based algorithms when different induction models are applied, namely, k Nearest Neighbor, Weighted Average, and Linear Locally Weighted Regression?

We now focus on the evaluation of the IBL-based algorithms. Figure 2 shows the prediction accuracy and prediction time for the three IBL-based induction algorithms. We can see that simpler and more computationally efficient algorithms such as 1-NN and 3-WA perform better than more sophisticated LLWR model. In most cases 1-NN becomes the best choice. This is in accordance with the results obtained in [14]. We can also see that without improvement even the best performed IBL model (1-NN) is still not practically useful in a real-time environment, with the average execution time per prediction exceeds 2 milliseconds.

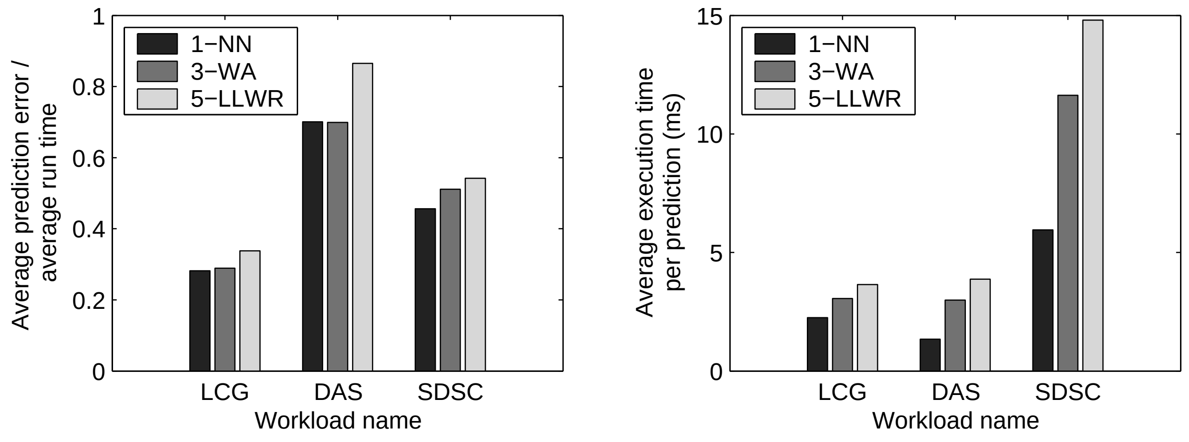

Figure 2. Comparison of prediction accuracy and prediction time of three IBL-based induction algorithms (without feature selection and instance reduction. Maximum look-back number is set to 5,000. Executions are performed on a PowerPC G4 550Mhz Mac with 512MB memory).

4. What is the impact of feature selection, feature weighting, and instance reduction on IBL in terms of search time and prediction accuracy?

The performance of the basic IBL algorithms can be improvd via feature selection and instance reduction. Feature selection (and weighting) is used to weight the attributes in the distance function. The above mentioned Improved Reduct and Genetic Algorithms are used, and we extend the GA encoding bits (GA-i, $i = 1, 2, 3$) to enable feature weighting. A relatively small weight set (not more than $2^3 = 8$ different weights) are evaluated since it typically gives better results than using a larger set [16]. We select the best feature weights and induction models by evaluating a training set and obtain the performance results on a test set, as are listed in table 4. Firstly, we can see that algorithms with features selected by Improved Reduct achieve similar or slightly better prediction accuracy then their GA counterparts. For GA selections using smaller feature weight set is indeed giving better results. As far as prediction time is concerned, scoped search gives the most significant reduction at the same time preserves or even improves the prediction accuracy. In some cases IB2 can also slightly add improvement. Overall, the best IBL algorithm is formed by improving the basic 1-NN or 3-WA model with features selected by Improved Reduct and instances reduced by scoped search plus IB2.

5. How is the similarity template approach, when combined with proper feature selection and adaption techniques, compared with the improved IBL-based algorithms?

Table 5 shows the comparison of the prediction accuracy as well as the prediction time of the two improved techniques, respectively. We can see that the improved Similarity Templates (ST) technique achieves comparable or considerably higher prediction accuracy than the improved IBL algorithm. The Similarity Templates approach is also more computationally efficient. Through instance reduction the IBL algorithm can have comparable prediction times on some workloads. However, the IBL algorithm still depends heavily on how fast the search for nearest neighbor(s) can be performed thus it is significantly slower than ST on other workloads. Generally speaking, the Similarity Template approach outperforms the IBL algorithm on most of our workloads.

5. Conclusions and Future Work

In this paper we have evaluated several main learning and heuristic techniques for application run time predictions. We take the run time prediction problem as a data mining problem and predictions are made by mining the similarities in the historical performance data. We compared two feature selection algorithms and empirically evaluated two major induction models, namely Similarity Templates and Instance Based Learning (IBL). The experimental results show that more computationally efficient Improved Reduct is able to achieve comparable results as Genetic Algorithms in terms of prediction accuracy. Moreover, the simple Similarity Templates approach, when combined with proper feature selection algorithms, outperforms more sophisticated IBL algorithms on a variety of workloads ranging from research to production.

There are many techniques from data mining and compu-

Workload Name	Reduction Techniques	Prediction Accuracy | Prediction Time			
		IReduct	GA-1	GA-2	GA-3
LCG		1-NN	1-NN	1-NN	1-NN
	None	0.289 | 2.05	0.323 | 0.26	0.323 | 0.26	0.332 | 0.28
	IB2	0.290 | 2.13	0.323 | 0.26	0.323 | 0.25	0.332 | 0.28
	Scoped	0.295 | 0.58	0.324 | 0.02	0.324 | 0.02	0.333 | 0.02
	IB2+Scoped	0.293 | 0.56	0.324 | 0.02	0.324 | 0.02	0.333 | 0.02
DAS		3-WA	1-NN	1-NN	1-NN
	None	0.702 | 2.48	0.707 | 1.08	0.707 | 1.08	0.711 | 1.14
	IB2	0.700 | 2.63	0.700 | 1.16	0.700 | 1.15	0.707 | 1.22
	Scoped	0.656 | 0.18	0.682 | 0.07	0.682 | 0.07	0.679 | 0.07
	IB2+Scoped	0.656 | 0.19	0.682 | 0.07	0.682 | 0.07	0.679 | 0.07
SDSC		1-NN	1-NN	1-NN	1-NN
	None	0.448 | 5.04	0.448 | 5.00	0.453 | 4.95	0.451 | 4.98
	IB2	0.451 | 5.18	0.451 | 4.80	0.455 | 5.13	0.452 | 5.14
	Scoped	0.448 | 0.06	0.448 | 0.06	0.457 | 0.07	0.453 | 0.07
	IB2+Scoped	0.447 | 0.06	0.447 | 0.06	0.457 | 0.06	0.452 | 0.06

Table 4. Performance comparison of different feature selection and instance reduction techniques in improving IBL algorithms (prediction accuracy = average prediction error / average run time, prediction time = average execution time in milliseconds per prediction. Maximum look-back number is set to 5,000. Executions are performed on a PowerPC G4 550Mhz Mac with 512MB memory).

Workload Name	Prediction Accuracy		Prediction Time (ms)	
	Improved ST	Improved IBL	Improved ST	Improved IBL
LCG	0.298	0.293	0.08	0.56
DAS	0.548	0.656	0.08	0.19
SDSC	0.439	0.447	0.05	0.06

Table 5. Comparison of prediction accuracy as well as prediction time of the improved Similarity Template and the improved IBL technique (Executions are performed on a PowerPC G4 550Mhz Mac with 512MB memory).

tational intelligence that can be exploited for performance predictions in the Grid. A generic toolkit will greatly facilitate the predictor evaluation/building process and we are developing such a toolkit called PROGRESS - a Prediction Framework for Grids, Resources, and Services. We will also extend our work to other Grid performance metrics such as resource response times and data transfer times.

6 Acknowledgments

We thank the support and help from the SC/Grid group in NIKHEF. We also want to express our gratitude to Peter van der Putten for his valuable suggestions and Dr. Nirav Kapadia for sending us his PhD thesis. We appreciate that the Parallel Workload Archive and Maui HPC Workload/Resource trace repository make the workloads publicly available.

References

[1] D. Aha. Feature weighting for lazy learning algorithms. In H. Liu and H. Motoda, editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Norwell MA, Kluwer., 1998.

[2] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[3] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[4] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[5] J. Cao, S. A. Jarvis, D. P. Spooner, J. D. Turner, D. J. Kerbyson, and G. R. Nudd. Performance prediction technology for agent-based resource management in grid environments. In *11th IEEE Heterogeneous Computing Workshop (in conjunction with IPDPS)*, 2002.

[6] L. Carrington, A. Snavely, X. Gao, and N. Wolter. A performance prediction framework for scientific applications. In *Workshop on Performance Modeling - ICCS*, 2003.

[7] S. H. Chen and P. P. W. (Eds.). *Computational Intelligence in Economics and Finance*. Springer-Verlag, 2003.

[8] The distributed asci supercomputer 2 (das2). `http://www.cs.vu.nl/das2/`.

[9] P. Dinda. *Resource Signal Prediction and Its Application to Real-time Scheduling Advisors*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2000.

[10] D. G. Feitelson. Workload modeling for performance evaluation. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 114–141. Springer Verlag, 2002.

[11] R. Gibbons. A historical application profiler for use by parallel schedulers. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 58–77. Springer Verlag, 1997.

[12] T. X. Hu. *Knowledge Discovery in Databases: An Attribute-Oriented Rough Set Approach*. PhD thesis, University of Regina, Canada, 1995.

[13] M. A. Iverson, F. Ozguner, and L. C. Potter. Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment. In *IPPS Eighth Heterogeneous Computing Workshop*, 1999.

[14] N. H. Kapadia. *On the Design of a Demand-Based Network-Computing System: The Purdue University Network-Computing Hubs*. PhD thesis, School of Electrical and Computer Engineering, Purdue University, 1999.

[15] N. H. Kapadia, J. A. B. Fortes, and C. E. Brodley. Predictive application-performance modeling in a computational grid environment. In *IEEE International Symposium for High Performance Distributed Computing (HPDC)*, 1999.

[16] R. Kohavi, P. Langley, and Y. Yun. The utility of feature weighting in nearest-neighbor algorithms. In *the Ninth European Conference on Machine Learning*, 1997.

[17] J. Komorowski, L. Polkowski, and A. Skowron. Rough sets: a tutorial. In S. Pal and A. Skowron, editors, *Rough-Fuzzy Hybridization: A New Method for Decision Making*. Springer Verlag, 1998.

[18] S. Krishnaswamy, A. Zaslasvky, and S. W. Loke. Predicting application run times using rough sets. In *Ninth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002)*, pages 455–462, 2002.

[19] Lhc computer grid project. `http://lcg.web.cern.ch.LCG/`.

[20] H. Li. An integrated algorithm for application run time predictions using workload traces. Technical Report TR-10-2004, Leiden Institute of Advanced Computer Science (LIACS), 2004. Available as `http://www.liacs.nl/~hli/professional/doc/algoRunTime.pdf`.

[21] H. Li, D. Groep, J. Templon, and L. Wolters. Predicting job start times on clusters. In *4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*, 2004.

[22] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowledge and Data Engineering, forthcoming*, 2004.

[23] Hpc workload/resource trace repository. `http://www.supercluster.org/research/traces`.

[24] Parallel workload archive. `http://www.cs.huji.ac.il/labs/parallel/workload/`.

[25] W. Smith, I. Foster, and V. Taylor. Predicting application run times using historical information. *Lecture Notes in Computer Science*, 1459:122–136, 1998.

[26] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

[27] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

[28] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, 2000.