Predicting performance of Grid based on Rough Set

Kun Gao[1], Zhongwei Chen[1], Meiqun Liu[2]
Computer Science and Information Technology College
Zhejiang Wanli University
No. 8, South Qian Hu Road, 315100, Ningbo, Zhejiang
P. R. China
Culture and Communication College
Zhejiang Wanli University
No. 8, South Qian Hu Road, 315100, Ningbo, Zhejiang
P. R. China

Abstract: In dynamic environment, the performance is restricted by various components, so we can not determine the contribution to performanc using traditional method. In this paper, we propose a novel method for predicting the performance in Grid Computing environment. We use the concept of Reduct in Rough Set theory and history record collected during a period of time to predict the applications runtime that the traditional methods can't obtain. We use the novel method in Data Ming Grid. The approach is based on frequencies of attributes appeared in discernibility matrix. The theoretical foundation of rough sets provides an intuitive solution to the problem of application run time estimation on Data Ming Grid. The results of the experiment show that the use of Rough Set theory can process uncertain problem in distributed and dynamic environment, and obtain better result than traditional methods.

Key-Words: Performance Evaluation, Distributed computing, soft computing

1 Introduction

Knowledge Grid is a software architecture for geographically distributed PDKD (Parallel and Distributed Knowledge Discovery) systems [1]. This architecture is built on top of a computational Grid and Data Grid that provides dependable, consistent, and pervasive access to high-end computational resources[2][3]. The Knowledge Grid uses the basic Grid services and defines a set of additional layers to implement the services of distributed knowledge discovery on world wide connected computers where each node can be a sequential or a parallel machine.

Grid computing is employing the resources of many computer nodes in a network to a certain question, usually to a scientific, technical, and commerce problem that requires much computer process power or access to mass data. In concept, Grid computing is a subset of distributed computing; On the other hand, in function, Grid computing is expansion and continuity to distributed computing. Grid emphasize coordination and cooperation between Grid resources[4][5].

It becomes the encouraging trend, because of the following reasons:

(1) Grid computing can effectively make use of the existing resources.

(2) It can condense a large amount of computing capability to solve the problem which can not be solved before grid.

(3) It will build widely distributed computing platform to integrate all kinds of resource including computation power resource, data resource, network resource and so on.

The research of scientist now focus on the resource allcation and task schedulling in Grid computing. It is the key component in Grid sytem. In order to workout the above problem, scientist must estimate the performance of Grid. In this paper, we propose a novel method for predicting the performance in Grid Computing environment. We use the concept of Reduct in Rough Set theory and history record collected during a period of time to predict the applications runtime that the traditional methods can't obtain. We use the novel method in Data Ming Grid. The approach is based on frequencies of attributes appeared in discernibility matrix. The theoretical foundation of rough sets provides an intuitive solution to the problem of application run time estimation on Data Ming Grid. The results of the experiment show that the use of Rough Set theory can process uncertain problem in distributed and dynamic environment, and obtain better result than traditional methods.

The rest of this paper is organized as followed: We introduce some related works and related Rough Set concept in section 2, 3; and then we propose a novel reduct algorithm in section 4; in section 5, we introduce the data mining framework, DMG. We conduct experiment to evaluate our approach in section 6. Finally in section 7, we conclude this paper.

2 Related Works

Early work in the parallel computing area proposed using similarity templates of application characteristics to identify similar tasks in a history. A similarity template is a set of attributes that we use to compare applications in order to determine if they are similar. Thus, for histories recorded from parallel computer workloads, one set of researchers selected the queue name as the characteristic to determine similarity [4]. They considered that applications assigned to the same queue were similar. In other work [5], researchers used several templates for the same history, including user, application name, number of nodes, and age.

Manually selecting similarity templates had the following limitations:

- Identifying the characteristics that best determine similarity isn't always possible.
- It's not generic: although a particular set of characteristics might be appropriate for one domain, it's not always applicable to other domains.

In [6][7], they proposed automated definition and search for templates and used genetic algorithms and greedy search techniques. They were able to obtain improved prediction accuracy using these techniques.

Recently, another effective approach to predict execution times on Grid is [8]. They investigate a use of sampling: in order to forecast the actual execution times of a given data mining algorithm on the whole dataset, they run the same algorithm on a small sample of the dataset. Many data mining algorithms demonstrate optimal scalability with respect to the size of the processed dataset, thus making the performance estimate possible and accurate enough. However, in order to derive an accurate performance model for a given algorithm, it should be important to perform an *off-line* training of the model, for different dataset characteristics and different parameter sets.

In this paper, we develop a rough sets based technique to address the problem of *automatically* selecting characteristics that best define similarity. In contrast to [6], our method determines a reduct as template, instead of using greedy and genetic algorithms. Rough sets provide an intuitively appropriate theory for identifying templates. The entire process of identifying similarity templates and matching tasks to similar tasks is based on rough sets theory, thereby providing an appropriate solution with a strong mathematical underpinning.

3 Related Concept on Rough Sets Theory

The theory of rough sets was introduced by Zdislaw Pawlak [17, 20] to deal with classification and analysis of data tables.Rough sets are particularly suitable for handling uncertainty in data. Uncertainty may be caused by missing or noisy data or due to ambiguity in the semantics of data. When handling such data, rough sets produce an inexact or "rough" classification. The concept of approximation space

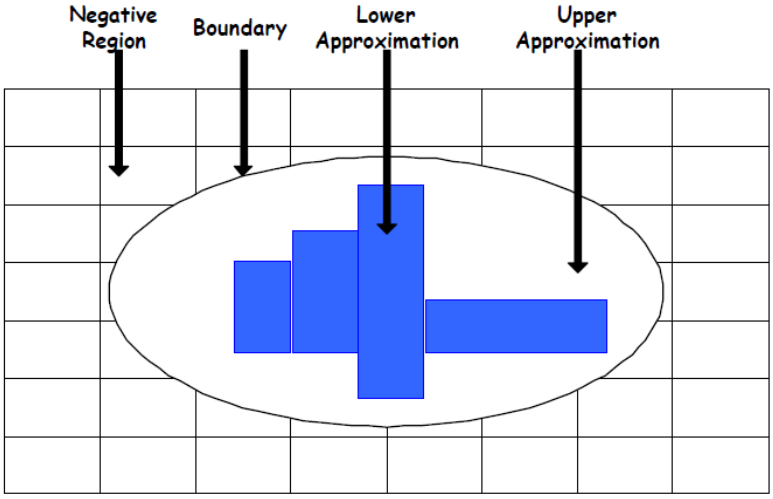

Figure 1 Rough Approximation Space

provides the boundaries for classifying objects. Rough sets use two concepts known as Upper Approximation Space and Lower Approximation Space as illustrated in figure 1. Its main idea is to maintain the ability of the same classification and derive classification rules through reducing knowledge.

The lower approximation of a concept (or class) consists of all objects that definitely belong to that concept and the upper approximation consists of all objects that possibly belong to the concept in question (i.e. objects beyond the upper approximation definitely do not belong to the class). The objects that fall between the upper and lower approximation spaces (which is also called the boundary region) are in the area of uncertainty or rough classification. Rough sets have been widely used in several application domains [17, 18, 19] for rule generation, attribute reduction and prediction. A distinctive feature of rough sets is that it operates using only the available data and does not require any additional assumptions such as grade of membership and prior probabilities. In this section we address the question of suitability of the theory of rough sets for identifying the characteristics that define similarity in application run-time estimation and develop a systematic method for applying the constructs of rough sets in this domain

Reduct is a very important aspect in rough sets theory. Reduct is an information system with minimal field sets, which remove the redundant data. The method for this idea is to search a certain fields that can represent original system wholly. So searching a reduct is to select some data with characteristic. Rough sets offers a set of method to find out all reduct. In this section, we introduce the principal concepts of rough sets theory related to our feature selection approach. The detail of the theory can be found in [9-13].

3.1 Information System

An information system is an ordered pair S=(U, A∪{d}), where U is a non-empty, finite set called the universe, A is a non-empty, finite set of conditional attributes, d is a decision attribute. A∩{d} =Φ. The elements of the universe are called objects or instances.

Information system contains knowledge about a set of objects in term of a predefined set of attributes. The set of objects is called concept in rough set theory. In order to represent or approximate these concepts, an equivalence relation is defined. The equivalence classes of the equivalence relation, which are the minimal blocks of the information system, can be used to approximate these concepts. Concept can be constructed from these blocks are called definable sets. As to undefinable sets, two definable sets, upper-approximation set and lower-approximation set are constructed to approximate the concept.

A simple IS is shown in Figure 2. The information system, two-dimensional table, this information system is composed of six records and two fields.

3.2 Indiscernibility Relation

An information system presents all the knowledge in related area. This two-dimensional table may be unnecessarily large because it may be superfluous in the two dimensions. The same or indiscernible records may be described several times, or some of the attributes may be redundant.

Let $P \subseteq A$, $xi, xj \in U$.

A binary relation IND called indiscernibility relation is defined as follow:

$IND(P) = \{(xi, xj) | (xi, xj) \in U \times U, a \in P, f(xi, a) = f(xj, a)\}$

Let $U/IND(P)$ denote the set of all equivalence classes of the relation IND(P).

U	a	b	c	d	e
1	1	0	2	2	0
2	0	1	1	1	2
3	2	0	0	1	1
4	1	1	0	2	2
5	1	0	2	0	1
6	2	2	0	1	1
7	2	1	1	1	2
8	0	1	1	0	1

Figure 2 Example Information System

3.3 Lower Approximation

Let $R \subseteq C$ and $X \subseteq U$. The lower approximation of X with respect to R is defined as follow:

$RX = \_\{Y \in U/R : Y \subseteq X\}$

RX is the set of all elements of U which can be with certain classified as elements of X, according to knowledge R.

3.4 Indiscernibility Relation

Let $S=(U,A\cup\{d\})$ be an information system, every subset $B\subseteq A$ defines an equivalence relation $IND(B)$,called an indiscernibility relation,defined as $IND(B)=\{(x,y)\in U\times U:a(x)=a(y)$ for every $a\in B\}$.

3.3 Positive Region

Given an information system:

$S=(U,A\cup\{d\})$

let $X\subseteq U$ be a set of records and $B\subseteq A$ be a selected set of fields. The lower approximation of X with respect to B is:

$B_*(X)=\{x\in U:[x]_B\subseteq X\}$.

The upper approximation of X with respect to B is:

$B^*(X)=\{x\in U: [x]_B\cap X\neq\Phi\}$.

The positive region of decision d with respect to B is:

$POS_B(d)=\cup\{B_*(X):X\in U/IND(d)\}$

The positive region of decision attribute with respect to B represents approximate quantity of B. Not all fields or records are necessary while describing approximate quantity of original IS, some are redundant. Reduct is the minimal set of fields describing approximate quantity.

3.4 Reduct

An attribute a is dispensable in $B\subseteq A$ if $POS_B(d)=POS_{B-\{a\}}(d)$. A reduct of B is a set of attributes $B'\subseteq B$ such that all attributes $a\in B-B'$ are dispensable, and $POS_B(d)=POS_{B'}(d)$.

A reduct consists of the minimal set of condition attributes that have the same discerning ability as the original IS. In other words, the reduct includes the most significant attributes. All reducts of a dataset can be found by constructing a kind of discernibility function from the dataset and simplifying it. Unfortunately, it has been shown that finding minimal reduct or all reducts are both NP-hard problems.

There are usually many reducts in an information system. In fact, one can show that the number of reducts of an information system may be up to $C^{|A|/2}_{|A|}$. In order to find reducts, discernibility matrix and discernibility function are introduced.

3.5 Discernibility Matrix

The discernibility matrix of an information system is a symmetric matrix:

$|U|\times|U|$

with entries c_{ij} defined as:

$\{a\in A|a(x_i)\neq a(x_j)\}$ if $d(x_i)\neq d(x_j)$, Φ otherwise.

A discernibility function can be constructed from discernibility matrix by or-ing all attributes in c_{ij} and then and-ing all of them together. After simplifying the discernibility function using absorption rule, the set of all prime implicants decides the set of all reducts of the IS.

4 A Novel Reduct Algrithm

The heuristic comes from the fact that intersection of a reduct and every items of discernibility matrix can not be empty. If there are any empty intersections between some item c_{ij} with some reduct, object i and object j would be indiscernible to the reduct. And this contradicts the definition that reduct is the minimal attribute set discerning all objects (assuming the dataset is consistent).

A straightforward algorithm can be constructed based on the heuristic. Let candidate reduct set $R=\Phi$. We examine every entry c_{ij} of discernibility matrix. If their intersection is empty, a random attribute from c_{ij} is picked and inserted in R; skip the entry otherwise. Repeat the procedure until all entries of discernibility matrix are examined. We get the reduct in R.

The algorithm is simple and straightforward. However, in most times what we get is not reduct itself but superset of reduct. For example, there are three entries in the matrix: $\{a_1, a_3\}$, $\{a_2, a_3\}$, $\{a_3\}$. According the algorithm, we get the reduct $\{a_1, a_2, a_3\}$ although it is obvious $\{a_3\}$ is the only reduct. This is because that our heuristic is a necessary but not sufficient condition for a reduct. The reduct must be a minimal one. The above algorithm does not consider this. In order to find reduct, especially shorter reduct in most times, we need more heuristics.

A simple yet powerful method is sort the discernibility matrix according $|c_{ij}|$. As we know, if there is only one element in c_{ij}, it must be a member of reduct. We can image that attributes in shorter and frequent $|c_{ij}|$ contribute more classification power to the reduct. After sorting, we can first pick up more powerful attributes, avoid situations like example mentioned above, and more likely get optimal or sub-optimal reduct.

The sort procedure is like this. First, all the same entries in discernibility matrix are merged and their frequency is recorded. Then the matrix is sorted according to the length of every entry. If two entries have the same length, more frequent entry takes precedence.

When generating the discernibility matrix, frequency of every individual attribute is also counted for later use. The frequencies is used in helping picking up attribute when it is need to pick up

one attribute from some entry to insert into reduct. The idea is that more frequent attribute is more likely the member of reduct. The counting process is weighted. Similarly, attributes appeared in shorter entry get higher weight. When a new entry c is computed, the frequency of corresponding attribute $f(a)$ are updated as $f(a)=f(a)+|A|/|c|$, for every $a \in c$; where $|A|$ is total attribute of information system. For example, let $f(a_1)=3$, $f(a_3)=4$, the system have 10 attributes in total, and the new entry is $\{a_1, a_3\}$. Then frequencies after this entry can be computed: $f(a_1)=3+10/2=8$; $f(a_3)=4+10/2=9$.

Input: an information system $(U, A \cup \{d\})$, where $A= \cup a_i$, $i=1,\ldots,n$.

Output: a reduct *Red*

1. $Red=\Phi$, count(a_i)=0, for $i=1,\ldots n$.
2. Generate discernibility matrix M and count frequency of every attribute count(a_i);
3. Merge and sort discernibility matrix M;
4. For every entry m in M do
5. If ($m \cap Red = = \Phi$)
6. select attribute a with maximal *count(a)* in m
7. $Red=Red \cup \{a\}$
8. Endif
9. EndFor
10. Return *Red*

Ficuture 3. A Heuristic Reduct Algorithm

Figure 3 is a heuristic reduct algorithm written in pseudo-code. In line 2, when a new entry c of M is computed, $count(a_i)$ is updated. $count(a_i):=count(a_i)+n/|c|$ for every $a_i \in |c|$. In line 3, Same entries are merged and M is sorted according the length and frequency of every entry. Line 4-9 traverses M and generates the reduct.

5 The Structure of the DMG

The DMG, Data Mining Grid, is an dynamic and distributed environment where data mining application is running. Its core component is task scheduling and resource allocation. These key questions can be solved through Rough Set theory.

Figure 4 describes the data ming system framework. It is mainly made up by following components:

5.1 DMGrid Client Node

In consideration of ease of use, the system adopts Browser/Server mode. Grid client exchanges information with Grid portal through Internet Explorer browser. Users submit the requirement of data mining and receive the final result at Grid client.

5.2 DMGrid Portal Node

It provides a single access way to distributed data mining application based grid. Users can make use of the whole grid resource transparently through the grid portal. This component is responsible for translating users' demand into the RSL language (Resource Specification Language) that can be recognized by grid, is used for grid resource discovery and grid resource allocation management. The final result is returned to grid portal first, and then returned to users by the portal.

5.3 DMGrid Resource Broker Node and DMGrid Tasks Allocation Broker Node

user's data mining requirement has driven grid resource discovery. According to users' demand condition, DMGrid resource broker looks for the resources which meet the condition in a large number of grid resources, including algorithms, computing capability and data resource. It is an important job that finds appropriate resource [14] [15]. As to any application based on grid, it is first to find appropriate resource, then allocate tasks and management them. It can be predicted that there may be many nodes which meet a condition. Resource broker is used for finding available resource among MDS (Meta Directory Service); mapping between data resource and computing resource, i.e., the task allocation broker is responsible for dispatching a certain task on a certain node.

5.4 Grid Node

The Grid nodes are made up of personal computer, high performance computer and cluster. Each node is installed GLOBUS, as grid middleware. They are the data carrier and the computation implementation entity.

The rationale of design and development the grid enabled data mining system is as follows:

- DMGrid adopts the standard, common and open grid service mode, follows OGSA norm, and offers unified support to the data mining applications.
- Based on Globus Toolkit and according to the existing networks system structure, DMGrid use the grid service to realize communication, operates each other and resource management.

• DMGrid is open, supports various data mining tools and algorithms, the extensibility is good.

• DMGrid is able to realize the improvement of performance by increasing network node, high performance computing node and cluster, the scalability is strong.

• DMGrid can deal with distributed huge volumes of high dimensional dataset, support heterogeneity data source.

• The main purpose to design and develop the DMGrid system is to improve the performance.

• Users carry out the data mining tasks in a transparent way; the concrete system structure, operation and characteristic in the grid environment is to be hidden.

• In the field of data mining, the security of the data and personal secrets are a sensitive topic. Data Ming Grid supports the choice of place that the data mining execute.

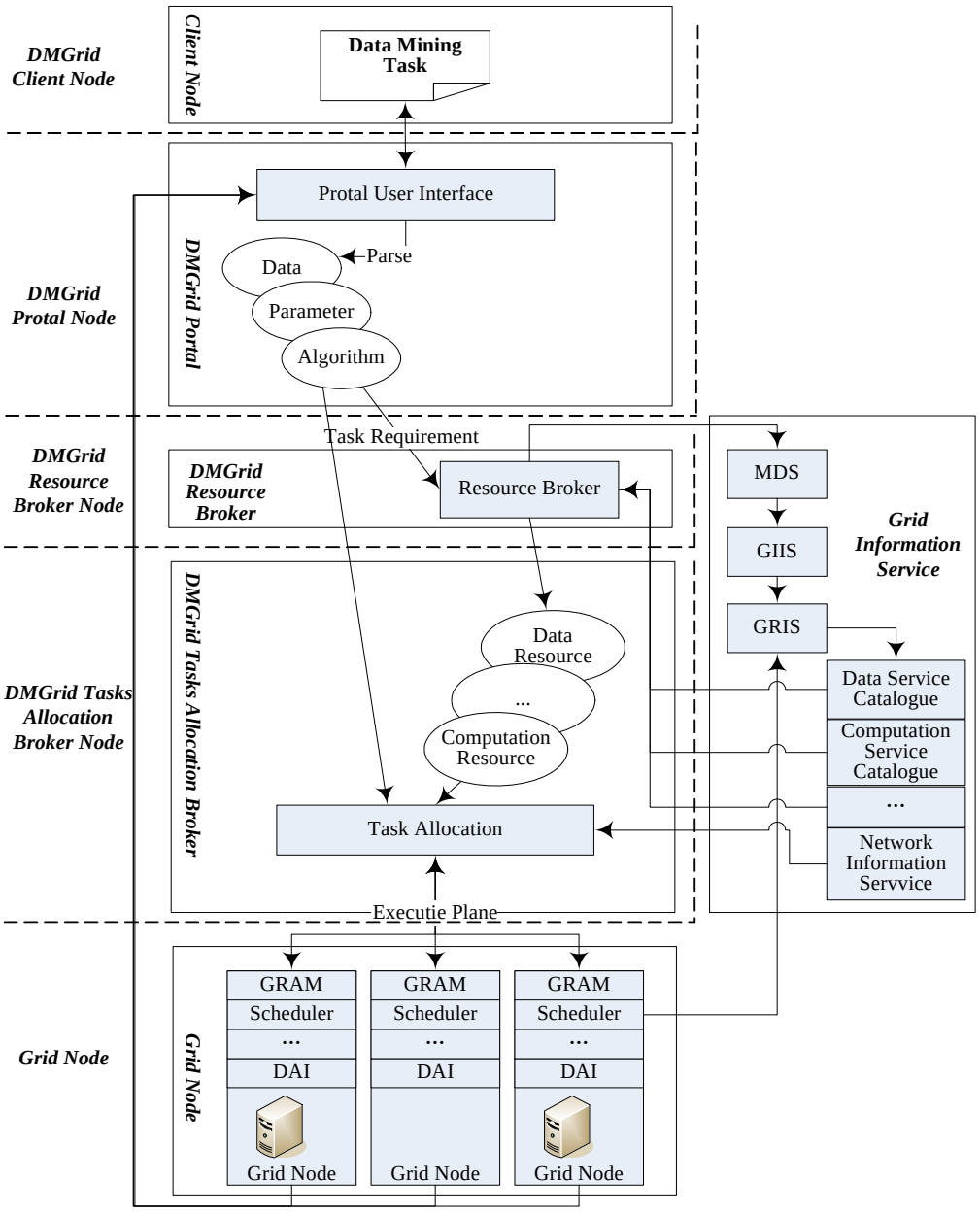

Figure 4. The framework of distributed data mining on grid

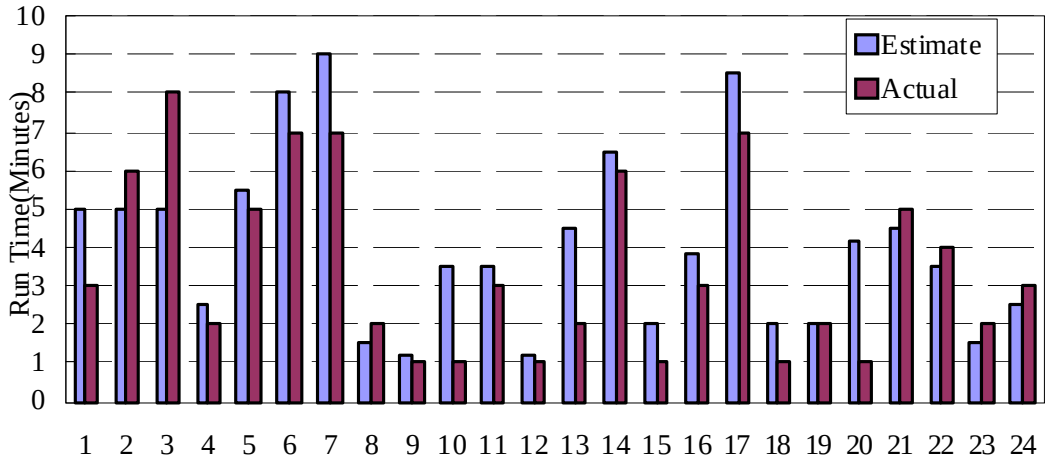

Figure 5 Experiments

6 Primary result of experiment

We contact the experiment in the Data Mining Grid, The simulated environment is composed of three machines which installed with GT3[16]. Each machine is interconnected by a switched fast Ethernet. Three distributed machines with different physical configurations and operating systems: a Pentium III running Windows 2000 with an 833-MHz processor and 512 Mbytes of memory; a Pentium 4 running Windows 2000 with a 2.0 GHz processor and 1Gbytes of memory; and a Sun Sparc station running Sun OS 5.8 with a 444-Mhz processor and 256 Mbytes of memory. For each data-mining job, we recorded the following information in the history: the algorithm, file name, file size, operating system, operating system version, IP address of the local host on which the job was run, processor speed, amount of memory, bandwidth, and start and end times. We used histories with 100 and 150 records, and as before, each experimental run consisted of 25 tests.

We differentiated the test case from the historical records by removing the runtime information. Thus, a test case consists of all the information specified except the recorded runtime. The runtime information recorded in the test case was the task's actual runtime. The idea was to determine an estimated runtime using our prediction technique and compare it with the task's actual runtime.

We compiled a history of data-mining tasks by running several data-mining algorithms and recording information about the tasks and environment. We executed several runs of data-mining jobs by varying the jobs' parameters such as the mining algorithm, the data sets and the sizes of the data sets. The algorithms we used were from the Weka package of data-mining algorithms[9]. We generated several data sets of sizes varying from 1 to 20 Mbytes.

In our experiment, the mean error was 0.23 minutes, and the mean error as a percentage of the actual runtimes was 7.6 percent. This shows that we accurately estimated the runtime for data-mining tasks on Grid. The reduct that our algorithm selected as a similarity template included the bandwidth, algorithm, file size, dimensionality, and available memory attribute. Figure 5 illustrates the actual and estimated runtimes from one of our experimental runs. Table 1 shows the condition attributes and corresponding reduct in each experiment. Figure 6 shows the number of condition attributes Vs. prediction performance. Because rough sets operate entirely on the basis of the condition attributes available in the history and require no external additional information, thus the more abundant the information correlating with performance, the more accurate the prediction is.

Table 1. Condition Attributes and Corresponding Reduct in Each Experiment

Experiment Number	Condition Attributes	Reduct
1	time, algorithm, parameter, disk cache, data size	algorithm, parameter, data size
2	operating system, time, algorithm, parameter, disk cache, data size, dimensionality, file name	algorithm, parameter, data size, dimensionality
3	CPU type, operating system, time, algorithm, parameter, disk cache, data size, dimensionality, file name, operating system version, CPU speed	algorithm, parameter, data size, dimensionality, CPU speed
4	memory type, CPU type, operating system, time, algorithm, parameter, disk cache, data size, dimensionality, file name, operating system version, CPU speed, available memory, disk type	algorithm, parameter, data size, dimensionality, CPU speed, available memory
5	IP, memory type, CPU type, operating system, time, algorithm, parameter, disk cache, data size, dimensionality, file name, operating system version, CPU speed, available memory, disk type, bandwidth, mainboard bus	algorithm, parameter, data size, dimensionality, CPU speed, available memory, bandwidth

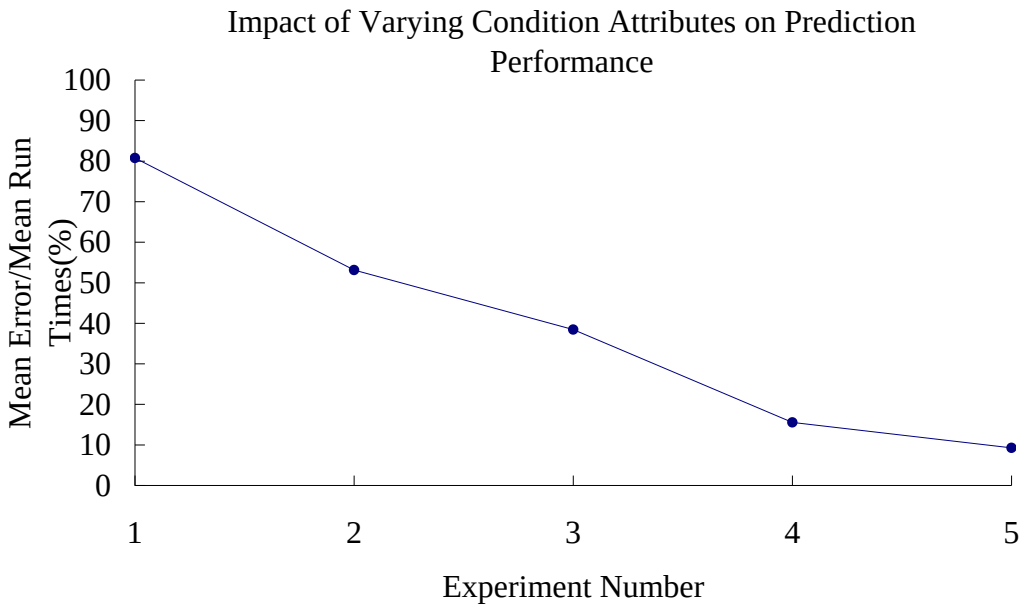

Figure. 6. The Number of Condition Attributes Vs. Prediction Performance

7 Conclusions and Future Works

We have presented a novel rough sets approach to estimating application run times. The approach is based on frequencies of attributes appeared in discernibility matrix. The theoretical foundation of rough sets provides an intuitive solution to the problem of application run time estimation on Data Mining Grid. Our hypothesis that rough sets are suitable for estimating application run time in Grid environment is validated by the experimental results, which demonstrate the good prediction accuracy of our approach. The estimation technique presented in this paper is generic and can be applied to others optimization problems.

References:

[1] M. Cannataro, D. Talia, P. Trunfio, KNOWLEDGE GRID: High Performance Knowledge Discovery Services on the Grid. *Proc. GRID 2001*, LNCS, Springer-Verlag, 2001.

[2] Foster I. and Kesselman C. (eds.) *The Grid: Blueprint for a Future Computing Inf.*, Morgan Kaufmann Publishers, 1999.

[3] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: towards an architecture for the distributed management and analysis of large scientific datasets. J. of Network and Comp. Appl, 2001.

[4] A.B. Downey , "Predicting Queue Times on Space-Sharing Parallel Computers,"*Proc. 11th Int'l ParallelProcessing Symp.* (IPPS 97), IEEE CS Press, 1997

[5] R. Gibbons , "A Historical Application Profiler for Use by Parallel Schedulers,"*Job Scheduling Strategies for Parallel Processing* , LNCS 1291, Springer-Verlag, 1997

[6] W. Smith , I. Foster, and V. Taylor , Predicting Application Runtimes Using Historical Information, Job Scheduling Strategies for Parallel Processing: IPPS/SPDP'98 Workshop, LNCS 1459, Springer-Verlag, pp.122-142,1998.

[7] W. Smith , V. Taylor, and I. Foster, Using Runtime Predictions to Estimate Queue Wait Times and Improve Scheduler Performance, Job Scheduling Strategies for Parallel Processing , LNCS 1659, D.G. Feitelson and L.Rudolph, eds., Springer-Verlag, pp. 202-229,1999.

[8] S. Orlando, P. Palmerini, R. Perego, and F. Silvestri, Scheduling high performance data mining tasks on a data grid environment. In Proceedings of Europar, 2002.

[9] X.Hu, Knowledge discovery in databases: An attribute-oriented rough set approach, Ph.D thesis, Regina university, 1995.

[10] J.Starzyk, D.E.Nelson, K.Sturtz, Reduct generation in information systems, Bulletin of international rough set society, volume 3, 1998.

[11] S.K.Pal, A.Skowron, Rough Fuzzy Hybridization-A new trend in decision-making, Springer, 1999.

[12] Witten,I,H., and Eibe,F., "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", Morgan Kauffman, 1999.

[13] Keyun Hu, lili Diao and Chunyi Shi: A Heuristic Optimal Reduct algorithm. 22nd Intl. Sym. on Intelligent Data Engineering and Automated Learning (IDEAL2000), Hong Kong, (2002)

[14] G. Allen, W. Benger, T. Goodale, H. Hege, G. Lanfermann, A. Merzky, T. Radke, E. Seidel, J. Shalf, The Cactus Code: A Problem Solving Environment for the Grid, Proceedings of the Ninth International Symposium on High Performance Distributed Computing (HPDC), Pittsburgh, USA, IEEE Press.

[15] K.Marzullo, M. Ogg, A. Ricciardi, A. Amoroso, F. Calkins, E. Rothfus, NILE: Wide-Area Computing for High Energy Physics, Proceedings of 7th ACM SIGOPS European Workshop, Connemara, Ireland, 2-4 Sept. 1996, ACM Press.

[16] Globus Toolkit, http://www.globus.org/ogsa/releases/alpha/

[17] Pawlak, Z., (1992), "Rough sets: Theoretical Aspects of Reasoning about Data", Kluwer Academic Publishers, London, UK.

[18] Slowinski, R., (1992), "Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory", (Eds.) R. Slowinski, Kluwer Academic Publishers.

[19] Slowinski, R., Stefanowski, J., (1993), "Special Issue on Rough Sets State of the Art and Perspectives", Foundations of Computing and Decision Sciences, Vol. 18, No. 3-4.

[20] Pawlak, Z., (1982), "Rough Sets", International Journal of Computer and Information Sciences, Issue. 11, pp. 413-433.