A Knowledge-Based Architecture for Helping in the Optimization and Development of Data Mining Applications in Grids

Francisco Flávio de Souza, Leonardo Ayres, Vasco Furtado

Universidade de Fortaleza – UNIFOR, Mestrado em Informática Aplicada – MIA
Washington Soares, 1521, Fortaleza – Ce, Brazil
fsouza@uol.com.br, leoayresm@yahoo.com.br, vasco@unifor.com.br

Abstract

In this paper, we define the SMARTBASEG architecture and show the preliminary results we have obtained in some experiments. Our proposal considers that the domain of Data Mining (DM) can be represented in terms of an ontology containing the definition of the main concepts involved in its algorithms. We also use ontology to describe some characteristics of a computational grid. This declarative feature of the architecture enables its dynamic optimization layer to decide how to transform procedures of DM applications into Grid-adapted tasks and submit them to the Grid layer, aiming at resulting in efficient load balancing. To do that, the optimization layer uses a knowledge base that makes heuristics explicit based on DM and Grid knowledge. SMARTBASEG also offers components that facilitate the development of DM applications for Grids.

Introduction

Knowledge Discovery in Database (KDD) is the data exploration process aiming at extracting useful, yet unknown, information. It involves handling a great amount of data and often requires large amounts of computational resources. These two characteristics have a considerable impact on the attainment of satisfying results. The most important task in KDD is Data Mining (DM), which can be of five types: classification, concepts formation, analysis of temporal series, analysis of attributes importance, and association (Fayyad 1996).

Computational grids (or simply Grids) appear as an efficient alternative to the DM processing demand – see some examples in (Hinke and Novotny 2000). However, despite the fact that some DM algorithms are naturally adequate to work on Grids, such as some genetic and data segmentation algorithms, most of them need to be adapted for efficient execution (Furtado 2004). Top-Down Induction Decision Tree (TDIDT) algorithms – one of the most popular types of classification algorithms – represent an example of that. They are composed of iterative phases responsible for database exploration, and this can lead to load unbalancing (Mehta, Agrawal, and Rissanen 1996) and scheduling problems. Such issues have a negative impact on performance results at the end of the execution of these algorithms.

Our motivation in this work is to define a knowledge-based architecture for optimizing DM applications, called SMARTBASEG, which acts as an intermediate layer between DM applications (implementations of algorithms) and the Grid middleware. Internally, SMARTBASEG uses an ontology describing the characteristics of DM applications and of the Grid where these applications are executed. Besides that, it has a knowledge base (or rule base), which makes explicit a set of heuristics based on DM and Grid knowledge for the optimization of tasks. The architecture identifies a DM application during execution and explores its properties, as well as the Grid's static and dynamic conditions (such as the number of online machines) in order to compose load-balanced Grid tasks. Another goal is to indirectly make the scheduling performed by the Grid middleware more efficient by determining criteria in the selection of Grid machines (such as its processing and/or storage capability) to perform tasks.

SMARTBASEG also offers components to assist in the development of DM applications for Grids. These components encapsulate concepts of the Grid and of how these applications can obtain better performance when they are executed on it, enabling, this way, the development of DM applications with code reuse and the abstraction of Grid-related requirements.

Related Work

Related works can be divided in two areas: Parallel Data Mining on SMP and Parallel Data Mining on Grid. In the first group, it is intended to develop parallel algorithms that use distributed computational resources to enhance the efficiency of the DM processes (Darlington et al. 1997; Kargupta et al. 1999). However, these works almost always are based on the fact that multi-processed machines are available. A particular area of Parallel DM refers to the parallel generation of TDIDT. Some works in this area can

be found in (Hall, Chawla, and Bowyer 1998; Kubota, Nakase, and Oyanagi 2001; Tsai and Yang 2004). Basically, TDIDT algorithms are useful to the classification function and its parallel implementation is essential to deal with a great amount of data. Decision trees are built in two steps: creation and prune of nodes. The first one consumes more resources from the computational point of view (Heath, Kasif, and Salzberg 1996). According to (Hall, Chawla, and Bowyer 1998), parallelism in TDIDT algorithms can be reached by processing data in parallel inside a node (in the level of records and/or attributes) and/or by creating nodes (that are not inter-dependent) in the decision tree in parallel. However, implementations of TDIDT algorithms in parallel are complex and, according to (Shafer, Agrawal, and Mehta 1996), they do not guarantee efficiency because the format of a decision tree, normally, is irregular, which results in the variation of the processing capability used for each node. Besides that, it is only during the algorithm execution time that the format of the tree is defined; therefore, pre-allocations of processing capability and of data will probably result in load balancing problems.

Concerning Grids, known approaches to obtain parallelism are not enough to bring efficiency to the process (Silva, Carvalho, and Hruschka 2004). Overhead questions caused by processes performed in central machines and issues of load balancing can be crucial. On the other hand, some works aim at making the DM activity more efficient through parallel access to databases that are being mined. These solutions are typically found in database management systems, which are already available in commercial products (Page 2002). These works, however, do not consider the fact that the increase of efficiency in DM does not depend exclusively of the improvement of data access. In fact, the complexity of a procedure (a routine or part of an application) is connected to its own logic of data handling: a procedure that performs an ordering on a file requires more resources than another one that simply reads this same file. Therefore, the lack of knowledge by a scheduler about the processing type and complexity makes an intelligent choice of Grid resources unfeasible. Another sub-group of works, concerning DM proposals, suggests the implementation of heuristics in algorithms and/or in schedulers to deal with situations characteristic of a heterogeneous environment, such as a Grid (Orlando et al. 2002; Banino et al. 2004). Basically, it is intended to identify the best Grid configuration for a certain application. They mainly consider the granularity of tasks, the size of the database and the quantity of machines to be used, which enables the decision of the best way to schedule and distribute tasks. The identification of these properties is possible by the creation of Grid monitoring mechanisms, besides the definition of algorithms for the learning process that dynamically determines the state of the Grid machines (Galstyan, Czajkowski, and Lerman 2004). Based on this, it is possible to make an intelligent scheduling of mining tasks. These works have greatly influenced our proposal.

The SMARTBASEG Architecture

Our vision about the task scheduling and load balancing problems for DM procedures is not only concerned with the search for information about the Grid. Our proposal is based on the principle that DM applications can be characterized in terms of an ontology (that describes concepts involved in the steps that compose DM algorithms) and that this characterization enables the definition of an optimization layer in order to choose dynamically the best way to submit an application (using components provided by the architecture) to a Grid. In this strategy, there is a knowledge base (containing rules that explicit heuristics) that enables the choice of appropriate dynamic task organizers. A task organizer is a specific code responsible for the arrangement of the transactions of DM processes in jobs/tasks, which will be submitted to the Grid. Thus the optimization phase produces one (or more) job(s) or set(s) of tasks, where each task corresponds to one (or more) DM procedure(s). It is important to introduce the concept of process and transaction, which will be useful along the architecture description. A process is an application in execution, while a transaction is each interaction made by a process with the Grid in order to get a set of DM procedures executed by it. In this work, we propose an architecture that can be seen as a knowledge layer between the DM application and a Grid, in a way that, both in the development and in the use of DM algorithms, the Grid concept can be transparent to the developer and to the final user. The general goal is to obtain quality in the development and in the use of these algorithms, independently of the Grid configuration and of the algorithm implementation. The architecture provides components that achieve this “transparency” (during the application development) and also a service for tasks optimization (in the application execution time).

The proposed architecture, called SMARTBASEG, is currently implemented over OurGrid/MyGrid (Cirne 2003). It offers specific DM services, encapsulating the Grid concepts and possible strategies of how to better use it (e.g. parallelism forms). The architecture uses dynamic information about the Grid behavior and information about previous executions of DM applications on the Grid to decide how to best submit the jobs/tasks generated by the application procedures. For instance, some tasks of a job (that perform some application procedures) can be grouped, thus, generating a new job that has fewer tasks (where each new task is composed of two or more original ones) in order to produce a better load balancing in the Grid machines.

SMARTBASEG Overview

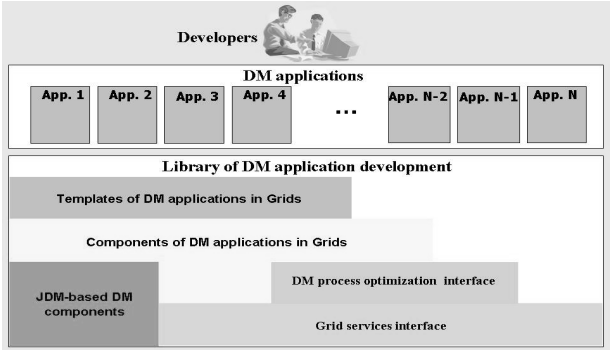

Figure 1. SMARTBASEG for the development of DM applications.

In figure 1, there is an overview of SMARTBASEG in the DM application developer's point of view. In this situation, the architecture makes available for the developer a set of templates, components, and interfaces, which make up layers organized in the following manner: the layer closest to the application is represented by templates of DM applications in Grids layer, in which skeletons for DM algorithms are available in order to facilitate the generation of applications by the developer. The components of DM applications in Grids layer encapsulates Grid concepts and also uses components that implement DM functionalities based on JDM – the *Sun* JSR 073 specification for DM (Hornik 2004) – and that make up the JDM-based DM components layer. They also determine which procedures will be sent to the grid in a parallel way. The DM processes optimization layer, in the application execution time, is responsible for optimizing the performance of an application being executed (process). Finally, the Grid services layer abstracts the Grid real implementation, thus, facilitating the portability of the other layers in the architecture. The developer of a DM application can access the layer he/she finds most appropriate, even though the ideal situation is to develop an application accessing the templates of DM applications in Grids layer because there are advantages in using already offered services, besides the code reuse and a better abstraction of the Grid concept.

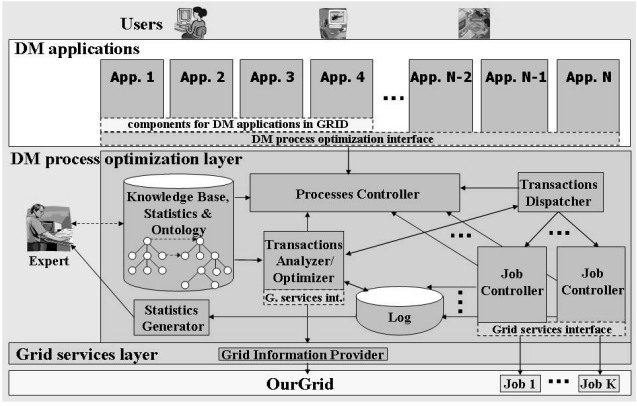

Figure 2. SMARTBASEG for the execution of DM applications.

In figure 2, we see the environment for the execution of DM applications proposed by the architecture, in which we focus on the Grid services layer and, mainly, on the optimization of DM process layer that will be described as follows.

Grid Services Layer. This software layer abstracts the Grid concepts (as well as its API) for the rest of the architecture and for the applications. The main services provided are: the access to the Grid information (obtained by the Grid information provider), such as its total number of on-line and off-line machines, the characteristics of the Grid scheduler, the beginning and finish times of a task in its initial, remote and final phases, the current status of a task (ready to execute, in execution, finished, not finished with failures, and canceled), the moment in which a job was added to the Grid and when it was finished, the current status of a job (ready to execute, in execution, finished, not finished with failures, and canceled); and the submission of jobs composed of tasks to the Grid.

DM Process Optimization Layer. This is the layer responsible for applying heuristics based on the knowledge of an expert to prepare jobs/tasks to be submitted to the Grid. It uses information about the characteristics of DM applications and of Grids. The main entities that compose it are:

- **Processes Controller**: Responsible for creating and maintaining the processes corresponding to applications (algorithms implementations) being executed on the optimization layer. It must also align the transactions arising from these processes and maintain them until they are finished (these transactions must come with an identification of the process that originated them). It's also an activity of the processes controller to load the concepts and statistics concerning finished processes and to maintain synthetic data about the execution of processes/transactions in progress (such as the total of transactions already submitted per process, the total execution time of the process up to the current moment, etc.);
- **Transaction Analyzer/Optimizer***:* This is an entity of the architecture responsible for deciding how the procedure(s) present in a transaction must be sent to the Grid to obtain better efficiency. To do that, it uses many heuristics represented in the knowledge base, concepts in the ontologies, and information maintained by other entities, such as: Grid conditions (how many machines exist, machines "capability", and "localization" of machines, etc.); results of previous executions of the application; results of transactions already executed in the current process of the application; and the nature of the application (the complexity of steps that make up the algorithm, the quantity of attributes and of examples, if the attributes are continuous or discrete, if the number of examples and of attributes decreases during the algorithm execution, etc.);

- **Log**: Place where the results of jobs (and their respective tasks) submitted to the Grid will be recorded, besides the results of decisions taken by the transactions analyzer/optimizer;
- **Job Controller**: Responsible for submitting to the Grid services layer jobs formatted by the transactions analyzer/optimizer. It keeps correspondence with the process transactions related to these jobs. When a job finishes, its controller records the information about it in the log and informs this fact to the processes controller;
- **Statistics Generator**: Responsible for making available synthetic information about the DM applications and the Grid to the experts, after processing the log records generated in consequence of the executions of DM procedures on Grid;
- **Statistics, Ontology and Knowledge Base**: Repository where are recorded the most important statistical information about the performance of DM applications. DM and Grid ontologies (maintaining conceptual information about the applications and about the Grid) are also represented. Finally, there is a knowledge base that represents heuristics in order to enable the transactions analyzer/optimizer to re-arrange jobs/tasks to be submitted to the Grid. These heuristics are based on information about the characteristics of procedures that are in the transactions as well as the concepts represented in the DM and Grid ontology;
- **Transactions Dispatcher**: Responsible for: verifying at the processes controller which transactions are recently arrived from processes, and forwarding the ones belonged to the process with the "highest" priority to the transactions analyzer/optimizer. When receiving the optimization result, it creates a job controller, which must be responsible for submitting jobs/tasks to the Grid.

Knowledge Representation

As mentioned previously, SMARTBASEG uses different sources (statistics information, ontology and Knowledge base) in order to enable the transactions analyzer/optimizer to perform its mission. For the intent of this paper, we will focus our explanation only on the ontology and the production rules.

Ontology. The ontology used by SMARTBASEG has concepts about DM applications (based in Cannataro and Comito 2003) and Grids. The definition that C4.5 is a supervised TDIDT classification algorithm is done in the DM ontology, as well as the characteristics of the steps that compose this algorithm. Moreover, Grid properties as the fact that a certain Grid only executes applications with independent and idempotent tasks (called Bag-of-Tasks applications) are made explicit in the ontology. These concepts facilitate the generation of rules in the knowledge base to be explored by the optimization layer during the execution of DM applications. Besides the Grid and DM concepts, in the ontology are also represented the concepts that were defined with the goal to integrate concepts from these two domains. The concept of process and transaction are examples of this.

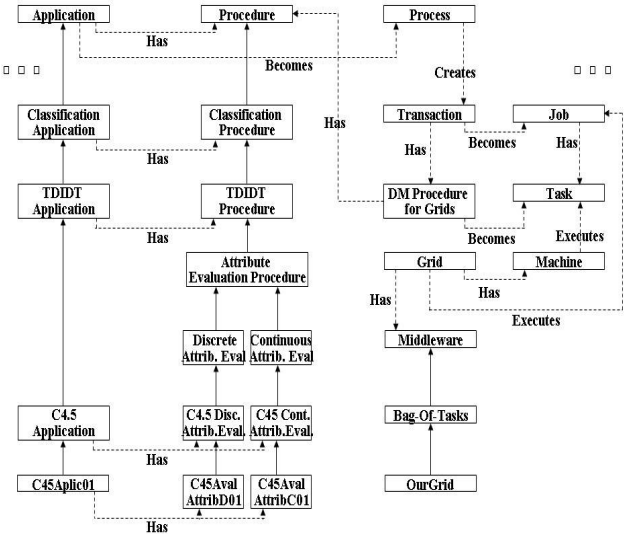

Figure 3. A part of SMARTBASEG's ontology describing DM and Grid concepts.

In Figure 3, we have a part of an implemented ontology, its taxonomies (solid lines) and how its concepts relate to each other (traced lines). Particularly, we exemplify the definition of the concepts involved in a DM application, in this case, the C45Aplic01. In the ontology, it is described as software that implements the C4.5 algorithm, which is a TDIDT algorithm and belongs to the classification function. We can see that the C45Aplic01 has the procedures C45AvalAtribC01 and C45AvalAtribD01, responsible for the continuous and discrete attribute evaluation, according to C4.5 algorithm. Basically, these procedures realize the same activity but as they have different computing complexity, they are represented particularly. On the same figure, it's possible to note that the concepts of process and transaction perform the integration between the concepts of application and procedures from the DM domain with the concepts of job and task from the Grid domain. Finally, we can see that the concept of Grid is declared as an entity that has machines to perform tasks and a middleware for its own management and the schedule of tasks. As an example, we have a grid middleware that deals with Bag-of-Tasks applications, the OurGrid.

Knowledge Base. In the knowledge base (or rule base), the rules to be explored during the execution of DM applications are represented in order to make the transactions analyzer/optimizer decide which kind of task organizer it will employ. Besides that, the transactions analyzer/optimizer can add conditions that will guide the scheduler (present in the middleware) of the Grid at the

moment to determine the machines that will execute the tasks.

```
...
IF (GridIsReady) AND NOT (ThereIsOptimizationCondition)
THEN organize_1_procedure_by_task;

IF (GridIsReady) AND (ThereIsOptimizationCondition) AND
   (ApplicationMethod = TDIDT) AND
   (TransactionsProceduresHasTheSameComplexity)
THEN organize_N_equivalent_procedures_by_task;

IF (GridIsReady) AND (ThereIsOptimizationCondition) AND
   (ApplicationMethod = TDIDT) AND NOT
   (TransactionsProceduresHasTheSameComplexity)
THEN organize_N_different_procedures_by_task;
...
```

Figure 4. Some rules in SmartBaseG's knowledge base.

In figure 4, we have a piece of the heuristic knowledge base of optimization, where pre-defined rules (represented as propositional rules to abstract the real ones) cause the execution of a specific task organizer given a certain situation. Some concepts such as GridIsReady (e.g., Grid is up and running with at least one machine online) and ThereIsOptimizationCondition (e.g. if the procedures are independent and idempotent and their number exceeds that of available Grid machines) are represented by rules not presented in the figure. With this information, we can see that the first rule simply organizes a procedure by task, since there is no optimization condition, creating a job with the same amount of tasks and DM procedures. The second rule determines that we can optimize the TDIDT procedures with the same complexities, just by dividing them equally. This creates a job that has a number of tasks equal to the number of online Grid machines. Finally, the third rule determines that procedures with different complexities need an organizer that balances the distribution of these procedures among tasks resulting in an egalitarian computational effort. There are many other rules not made explicit here, for example, a rule that organizes all kinds of DM procedures in just one task if it is determined that these procedures would have a very low computational cost, regardless of how many Grid machines are available. This prevents unnecessary allocation of resources, keeping them free to be assigned to other processes.

Implementing a DM Application Using SMARTBASEG

In order to make a preliminary validation of our proposal, it was implemented a TDIDT algorithm, in this case, the C4.5 (Quinlan 1992). This algorithm performs successive division steps of the initial training set while it builds a decision tree. After each iteration, the C4.5 chooses an attribute to be the root of the decision tree and divides the training set for each value of the chosen attribute. The procedure for choosing the best attribute to be the root of the tree is the most costly of the algorithm. Because of that, it is potentially parallelizable, since it is necessary for its realization the execution of a new procedure (called attribute evaluation) for each one of the involved attributes. All of these evaluations can be done in parallel, since they are independent from each other.

The algorithm used Java interfaces that SMARTBASEG made available. Internally, SMARTBASEG has the implementations of the DM and Grid ontology, which were generated through the Protégé tool (Noy et al. 2001), besides some heuristics, generated through the Jeops tool (Figueira and Ramalho 2000), to better direct the use of the Grid during the execution of the application that implements the C4.5 (C45App01). The implemented heuristics are the following.

In the task load balance heuristic, the number of tasks produced by the algorithm corresponds to the number of available machines in the Grid at a moment. The tasks can be grouped in order to improve load balancing between the machines. Suppose, for instance, that in a Grid BoT, as OurGrid, made up of 50 machines, 100 tasks of a DM algorithm must be executed. Typically, each task would be submitted for a machine and the others 50 would be in queue waiting the release of some machine for their processing. This situation is not desirable since it does not optimize the use of the resources, besides provoking a home machine overhead. To solve this, the optimization layer using the task load balance heuristic groups the tasks by two in accordance with the amount of machines. Thus, each machine process two tasks avoiding queue formation.

The other heuristic is named task balance with application knowledge. This heuristic is similar the previous one; however, it is enriched with knowledge about the application. In such a case, the task grouping is made not alone taking into consideration the amount of machines, but also the task type and mainly its cost. In TDIDT algorithms, internal calculations can vary strongly in function of the type of attribute that is participating of these calculations. For example, in continuous attributes, it is always necessary to make a sort of the attribute values before the calculation properly said. In discrete attributes this does not occur. Consequently the evaluation of a continuous attribute is much more costly than an evaluation of a discrete attribute. The heuristic basic idea is to homogeneously distribute the tasks in accordance with the type of attribute preventing excessive evaluations of continuous attributes in certain machines. Suppose for example, 100 tasks generated by an algorithm that is mining a database with five continuous and five discrete attributes in a Grid with 50 machines. This type of heuristic leads the optimizer to produce 50 tasks with 2 attribute evaluations each: one of continuous attribute and another of discrete attribute.

Experimental Evaluation

In this section, we exemplify how the use of heuristics defined in SMARTBASEG influenced the DM applications performance in a Grid.

For the heuristic evaluation, we generated seven artificial databases with 100k examples and 16 attributes. The first database has only discrete attributes and the second only continuous. The others are composed of continuous and discrete attributes in different proportions. Our intention is to exemplify situations where SMARTBASEG, having knowledge about the application, can improve the partitioning and the dispatch of task to the Grid, influencing the performance of the applications. In figure 5, we see the execution time of three applications, which are different versions of C4.5 algorithm, mining the seven different databases. First, a serial version of the algorithm was executed, in the local machine without use of the Grid. Then, the second version of the algorithm was executed in the Grid, with no heuristic. Finally, the third version of the algorithm, using the heuristics described previously, was executed in the same Grid. For the first and second database, only the task balance heuristic was applied while, for the other databases, the task balance with application knowledge heuristic was preferred.

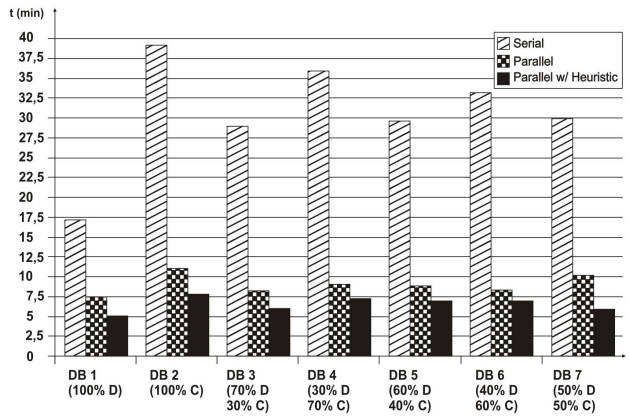

Figure 5. Execution time of the three versions of C4.5 algorithm using different approaches.

The results show that the use of heuristics improves the performance of the applications. The task balance with knowledge heuristic was the most interesting in terms of performance even if the proportion of discrete and continuous attributes varies. This example shows the major contribution of this work that is the possibility of representing in a knowledge base different heuristics to be used in each specific case. To render these heuristics explicit in the optimization layer enables the programmer of a DM application to be free of concerns related to the application performance in the Grid.

Conclusion and Future Work

In this article, we described the SMARTBASEG architecture, which is based on the principle that DM applications can be characterized in terms of ontology and that this characterization enables the definition of an optimization layer that can decide the best way to submit procedures of a DM application to a Grid. We described examples of heuristics used by the optimization layer to make the DM application more efficient when executed on a Grid. The main contribution of this work is to show that it is possible to provide an optimization service for better execution of DM applications on Grid. The approach uses knowledge about Grids and, mainly, about the DM applications being executed, avoiding that the developer of these applications be concerned with the particularities of the Grid and the means to get better performance on it. It is a multi-disciplinary approach, since it uses Artificial Intelligence techniques to help with the representation of heuristic knowledge concerning the preparation of tasks to be executed on a Grid.

Our research continues to focus on identifying interesting heuristics to be programmed into SMARTBASEG. Specifically, we are investigating the simultaneously use of multiple heuristics. We are also enriching the formalization of the ontology, including more features of DM algorithms and Grids. Another area of research is the implementation of the access layer to Grid services. This opens the possibility to define more heuristics that combine information about the application with dynamic characteristics of the Grid.

Acknowledgments

This work was partially developed in collaboration with HP Brazil P&D.

References

Banino, C., Beaumont, O., Carter, L., Ferrante, J., Legrand, A., and Robert, Y. 2004. Scheduling Strategies for Master-Slave Tasking on Heterogeneous Processor Platforms. IEEE Transactions on Parallel Distributed Systems 15(4): 319-330.

Cannataro, M., and Comito, C. 2003. A Data Mining Ontology for Grid Programming. In Proceedings of the First International Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGrid2003), 115-134. Budapest, Hungary. At http://www.isi.edu/~stefan/SemPGRID.

Cirne, W., Paranhos, D., Costa, L., Santos-Neto, E., Brasileiro, F., Sauve, J., da Silva, F. A. B., Osthoff, C., and Silveira, C. 2003. Running bag-of-tasks applications on computational grids: The MyGrid approach. In Proceedings of the International Conference on Parallel Processing (ICCP'2003), 407-418. Kaohsiung, Taiwan: IEEE Computer Society.

Darlington, J., Guo, Y., Sutiwaraphun, J. and To, H. W. 1997. In Proceedings of the Second International Symposium on Advances in Intelligent Data Analysis, Reasoning about Data, 437-445. London, UK: Springer-Verlag.

Fayyad, U. M. 1996. Data mining and knowledge discovery: making sense out of data. IEEE Intelligent Systems 11(5): 20-25.

Figueira Filho, C. S., and Ramalho, G. L. 2000. The Java Embedded Object Production System. In Proceedings of the International International Joint Conference,7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI, IBERAMIA-SBIA 2000, 53-62. Atibaia, SP, Brazil: Springer.

Furtado, V. 2004. Data Mining in Grid, Technical Report I, HP/UNIFOR Project, University of Fortaleza – UNIFOR, Fortaleza, CE, Brazil.

Galstyan, A., Czajkowski, K., and Lerman, K. 2004. Resource Allocation in the Grid Using Reinforcement Learning. In Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagents Systems (AAMAS'04), 1314-1315. New York, New York, USA: IEEE Computer Society.

Hall, L. O., Chawla, N., and Bowyer, K. W. 1998. Combining Decision Trees Learned in Parallel. In Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining. New York, New York, USA: AAAI Press.

Heath, D., Kasif, S. and Salzberg, S. 1996. Committees of Decision Trees. In Cognitive Technology: In Search of a Humane Interface, 305-317, B. Gorayska and J.L. Mey eds.: Elsevier.

Hinke, H. T., and Novotny, J. 2000. Data Mining on NASA´s Information Power Grid. In Proceedings of Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC'00), 292-293. Pittisburgh, Pennsylvania, USA: IEEE Computer Society.

Hornick, M. 2004. Java Specification Request 73: Java Data Mining (JDM), Version 1.0, Final Release: Java Community Process. At http://www.jcp.org/aboutJava/communityprocess/final/jsr073.

Kargupta, H., Park, B., Hershbereger, D. and Johnson, E. 1999. Collective Data Mining: A New Perspective Toward Distributed Data Mining. In Advances in Distributed and Parallel Knowledge Discovery, Hillol Kargupta and Philip Chan eds.: MIT/AAAI Press.

Kubota, K., Nakase, A., and Oyanagi, S. 2001. Implementation and Performance Evaluation of Dynamic Scheduling for Parallel Decision Tree Generation. In Proceedings of 15th International Parallel and Distributed Processing Symposium (IPDPS'01) Workshops. San Francisco, California, USA: IEEE Computer Society.

Mehta, M., Agrawal, R., and Rissanen, J. 1996. SLIQ: A fast scalable classifier for data mining. In Proceedings of the Fifth International Conference on Extending DataBase Technology (EDBT), 18-32. Avignon, France: Springer.

Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Fergenson, R. W., and Musen, M. A. 2001. Creating Semantic Web Contents with Protege-2000. IEEE Intelligent Systems 16(2): 60-71.

Orlando, S., Palmerini, P., Perego, R. and Silvestri, F. 2002. Scheduling high performance data mining tasks on a data Grid environment. In Proceedings of International Conference Euro-Par 2002, 375-384. Paderborn, Germany: Springer-Verlag.

Page, C. 2002. Phase A Report. Database and Data Mining, chapter 7. The Astrogrid Project. At http://wiki.astrogrid.org/bin/view/Astrogrid/RbDataminingTechnologyReport.

Quinlan, J.R ed. 1992. C4.5: Programs for Machine Learning. San Francisco, California, USA: Morgan Kauffmann.

Shafer, J. C., Agrawal, R. and Mehta, M. 1996. *SPRINT: A scalable parallel classifier for data mining*. In Proceedings of the Twenty-Second International Conference on Very Large Databases, 544-555. Bombain, Índia: Morgan Kaufmann.

Silva, F., Carvalho, S., and Hruschka, E. 2004. A Scheduling Algorithm for Running Bag-of-Tasks Data Mining Applications on the Grid. In Proceedings of the International Conference Euro-Par 2004. 254-262. Pisa, Italy: Springer-Verlag.

Tsai, S. T., and Yang, C. T. 2004. Decision Tree Construction for Data Mining on Grid Computing. In Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04), 441-447. Taipei, Taiwan: IEEE Computer Society.