Usando Conhecimento de Aplicações de Mineração de Dados para a Otimização de sua Execução em Grades

Francisco Flávio de Souza, Jefferson Carvalho, Juliana Carvalho, Vasco Furtado

Universidade de Fortaleza - UNIFOR
Caixa Postal 1280 – Fortaleza – CE

fsouza@uol.com.br, vasco@unifor.br,
jeffersoncarvalho@gmail.com, julianacarvalho@fortalnet.com.br

Abstract. *In this paper, we introduce 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 an ontology to describe some characteristics of a grid. This declarative feature of the architecture enables its dynamic optimization layer to decide how to transform procedures of DM applications in terms of grid jobs composed of tasks, by using heuristics based on DM and Grid knowledge, and to submit them to the Grid layer, aiming at resulting in efficient load balancing.*

Resumo. *Neste artigo, introduzimos a arquitetura SMARTBASEG e apresentamos os resultados iniciais de alguns experimentos. Nossa proposta considera que o domínio de mineração de dados (MD) pode ser representado em termos de uma ontologia contendo a definição dos principais conceitos envolvidos em seus algoritmos. Nós também utilizamos uma ontologia para descrever algumas características de uma grade. Essa propriedade declarativa da arquitetura possibilita à sua camada de otimização dinâmica, através de heurísticas baseadas no conhecimento sobre MD e grades, decidir como transformar procedimentos de MD em jobs compostos de tarefas, e submetê-los à grade, visando um balanceamento de carga eficiente.*

1. Introdução

A descoberta de conhecimento em bases de dados é o processo de exploração dessas bases com a finalidade de extrair informação útil ainda desconhecida. Ela envolve a manipulação de grandes quantidades de dados e requer alta capacidade de processamento. A sua atividade mais importante é a de mineração de dados (MD), que pode ser vista como composta das funções de classificação, formação de conceitos, análise de séries temporais, análise de importância de atributos e associação [1].

Grades Computacionais (ou simplesmente grades) aparecem como uma alternativa eficiente para a demanda de processamento de MD (vide alguns exemplos em [2]). Entretanto, apesar do fato de que alguns algoritmos de MD sejam naturalmente adequados a funcionar em uma grade, como é o caso de alguns algoritmos genéticos e de segmentação de dados, a maioria deles necessita de adaptação para serem executados eficientemente [3]. Algoritmos de árvores de decisão (TDIDT, do inglês "Top Down Induction Decision Tree") - uns dos mais populares algoritmos de classificação - são um exemplo disso. Eles são compostos de fases iterativas de exploração da base de dados, e

tal característica pode levá-los a problemas de balanceamento de carga [4] e de escalonamento, que irão provocar, ao final da execução desses algoritmos, baixos resultados em termos de desempenho.

A nossa motivação, neste trabalho, é apresentar uma arquitetura baseada em conhecimento para a otimização de aplicações de MD, chamada SMARTBASEG, que age como uma camada intermediária entre as aplicações (implementações de algoritmos) de MD e o *middleware* de uma grade. Internamente, SMARTBASEG se utiliza de uma ontologia sobre as características de aplicações de MD e de grades. Além disso, ela possui uma base de regras que implementa um conjunto de heurísticas de otimização de tarefas. Durante a execução de uma aplicação de MD, a base de regras consulta a ontologia acerca das características dessa aplicação e da grade onde ela está sendo executada com a finalidade de compor as tarefas a serem submetidas à grade, primando pelo balanceamento de carga.

2. Estado da Arte

Os trabalhos relacionados com o tema podem ser divididos em duas áreas: mineração de dados paralela em máquinas multiprocessadas (SMP, do inglês "**S**ymmetric **M**ulti**p**rocessing") e mineração de dados paralela em grades. No primeiro grupo, pretende-se desenvolver algoritmos paralelos que façam uso de máquinas multiprocessadas para a obtenção de uma maior eficiência em MD[5], [6]. Esses trabalhos, no entanto, quase sempre, partem do princípio de que os recursos computacionais a serem utilizados estarão disponíveis. Um ramo particular de MD paralela refere-se à geração em paralelo de árvores de decisão (TDIDT). Alguns trabalhos nessa área podem ser encontrados em [7, 8, 9]. Basicamente, algoritmos de TDIDT são úteis para a função de classificação, e sua implementação em paralelo é essencial para lidar com grandes quantidades de dados. Árvores de decisão são construídas em dois passos: criação de nós e poda. O primeiro deles é o que consome mais recursos do ponto de vista computacional [10]. De acordo com [7], o paralelismo em algoritmos TDIDT pode ser alcançado pelo processamento dos dados em paralelo dentro de um nó (em nível de registros e/ou de atributos) e pela construção dos nós da árvore de decisão (que não têm dependência entre si) em paralelo. Entretanto, implementações de algoritmos TDIDT em paralelo são complexas e, segundo [11], não garantem eficiência, pois a forma da árvore de decisão, normalmente, é irregular, o que faz com que a capacidade de processamento usada para cada nó varie. Além disso, é somente em tempo de execução do algoritmo que a forma da árvore é determinada; portanto, alocações estáticas de poder de processamento e de dados, muito provavelmente, irão resultar em problemas de balanceamento de carga.

Em se tratando de grade, as soluções de paralelização apresentadas ainda não são, por si só, suficientes para o aumento de eficiência em MD [12]. Questões de *overhead* causadas por processos realizados em máquinas centralizadoras e de balanceamento de carga podem ser cruciais. Por outro lado, alguns trabalhos buscam tornar a atividade de MD mais eficiente através da paralelização de acessos aos bancos de dados que estão sendo minerados. Essas soluções são tipicamente encontradas em sistemas gerenciadores de bancos de dados, inclusive, já disponíveis em produtos comercializados [13]. Esses trabalhos, no entanto, não consideram o fato de que o aumento de eficiência em MD não se resume, na maioria dos casos, em melhorar o acesso a dados. De fato, a complexidade de um procedimento (rotina ou parte de uma aplicação) está ligada à sua lógica própria de manipulação dos dados. Um procedimento

que realiza uma ordenação de um arquivo exige mais recurso do que um outro que faça uma simples leitura desse mesmo arquivo. Portanto, o desconhecimento por parte de um escalonador sobre o tipo e a complexidade do procedimento inviabiliza uma escolha inteligente dos recursos da grade. Um outro subgrupo de trabalhos visa propor a implementação de heurísticas nos algoritmos e/ou em escalonadores para tratamento das situações próprias de um ambiente heterogêneo como uma grade [14], [15]. Basicamente, busca-se identificar a melhor configuração da grade para uma dada aplicação. Considera-se, principalmente, a granularidade das tarefas, o tamanho da base de dados e a quantidade de máquinas a serem usadas, o que permite decidir a melhor forma de escalonamento e distribuição das tarefas. A identificação dessas propriedades é possível pela criação de mecanismos de monitoramento da grade, além da definição de algoritmos de aprendizagem que determinem dinamicamente o estado das máquinas da grade [16]. A partir dessas informações, pode-se realizar um escalonamento inteligente das tarefas geradas por processos (execuções de aplicações) de MD.

3. Visão Geral de SMARTBASEG

Nossa proposta parte do princípio de que as aplicações de MD podem ser caracterizadas em termos de uma ontologia (explicitando conceitos envolvidos nos passos que compõem os seus algoritmos) e de que esta caracterização possibilita a uma camada de otimização decidir dinamicamente pela melhor forma de submeter uma aplicação de MD à grade. Nessa estratégia, uma base de regras (implementando heurísticas) é usada para a escolha apropriada de organizadores de tarefas e dos valores de seus parâmetros de entrada. Um organizador de tarefas é uma rotina específica de software responsável por um tipo de arranjo em um conjunto de procedimentos de MD, produzindo um conjunto de tarefas para grades. Portanto, ao final de uma fase de otimização, teremos um (ou mais) *job(s)* (conjunto(s) de tarefas), onde cada uma de suas tarefas irá corresponder a um (ou mais) procedimento(s) de MD, dependendo do tipo de arranjo realizado. Por exemplo, numa transação (cada submissão de um conjunto de procedimentos de MD feita por um processo a uma grade) contendo 20 procedimentos de MD, o resultado da otimização poderia ser um *job* contendo somente 5 tarefas (cada uma com 4 procedimentos), como também poderia ser 2 *jobs*, onde o 1º. teria 3 tarefas (cada uma com 2 procedimentos) e o 2º. teria 2 tarefas (cada uma com 6 procedimentos), e assim por diante. Decidimos, assim, propor uma arquitetura, chamada SMARTBASEG, que pode ser vista como uma camada de conhecimento entre aplicações de MD e uma grade que oferece o serviço de otimização na execução dessas aplicações e encapsula conceitos gerais (formas de paralelismo de algoritmos de MD) e específicos (o tipo de aplicação com que a grade pode trabalhar, a sua arquitetura e a API de seu *middleware*) de uma grade. O objetivo geral é tentar melhorar a qualidade no desenvolvimento (pelo aumento de produtividade em virtude do encapsulamento da grade e do reuso do código fornecido pela arquitetura) e, principalmente, na execução de aplicações de MD (pelo ganho de desempenho), independentemente da configuração da grade e da própria aplicação. Para isso, a arquitetura, atualmente implementada sobre OurGrid/MyGrid[17], fornece componentes que tornam possível essa “transparência” de grade (em tempo de desenvolvimento da aplicação) e também fazem acesso a um serviço de otimização de tarefas (em tempo de execução da aplicação).

Embora a arquitetura seja composta de várias camadas, neste trabalho, enfocaremos a camada de otimização de processos, que é a responsável por preparar os *jobs* a serem submetidos à grade, visando um melhor balanceamento de carga. Mais

precisamente, enfocaremos duas das fontes de informação de SMARTBASEG para o suporte à otimização de transações: a ontologia e a base de conhecimento (ou de regras). É através de heurísticas representadas na base de regras e de conceitos presentes na ontologia que o analisador/otimizador de transações, principal entidade da camada de otimização, decide como o(s) procedimento(s) de MD presentes em uma (ou mais) transação(ões) deve(m) ser enviado(s) à grade.

3.1. Ontologia

A ontologia usada por SMARTBASEG abrange os conceitos sobre as aplicações de MD, sendo alguns deles baseados em [18], e sobre grades. Por exemplo, a definição de que o algoritmo C4.5 é de classificação (portanto, supervisionado) e usa o método TDIDT, assim como o fato de que uma certa grade só trabalha com tarefas cujos procedimentos a serem executados sejam independentes entre si (ou seja, pertencem a aplicações "Bag-of-Tasks") estão explicitados na ontologia. Estes conceitos facilitarão a construção de regras na base de conhecimento para serem exploradas pela camada de otimização durante a execução das aplicações de MD. Para a integração entre os conceitos de MD e de grades, formando uma ontologia geral, SMARTBASEG acrescentou alguns conceitos próprios, dentre os quais, destacam-se o de processo e o de transação.

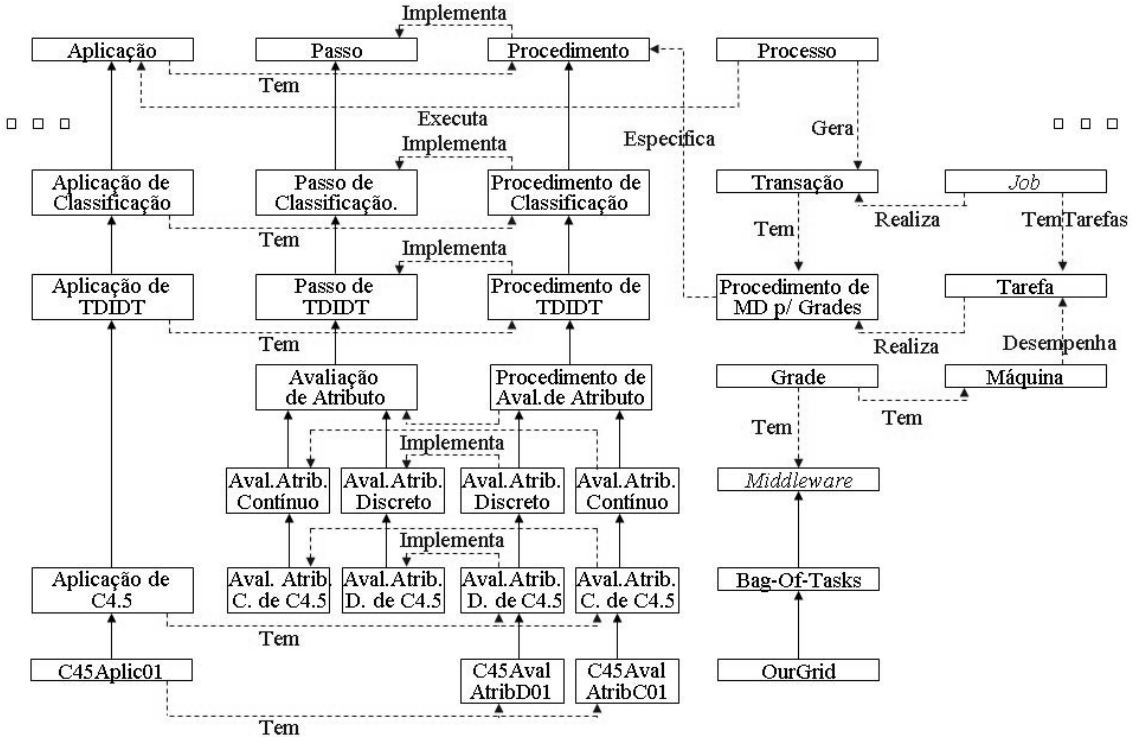

Figura 1. Integração entre conceitos de MD e de grades na ontologia de SMARTBASEG

Na figura 1, temos uma amostra da ontologia implementada, suas taxonomias (linhas cheias) e como seus conceitos se relacionam (linhas tracejadas). Em particular, exemplificamos a definição dos conceitos envolvidos em uma aplicação de MD, no caso, a C45Aplic01. Na ontologia, ela está descrita como sendo um *software* de implementação do algoritmo C4.5, que, por sua vez, é um algoritmo TDIDT e, portanto, pertencente à função de classificação. Vemos ainda que C45Aplic01 tem os procedimentos C45AvalAtribC01 e C45AvalAtribD01, que estão declarados como sendo, respectivamente, implementações de passos de TDIDT de avaliação de atributo

contínuo e de avaliação de atributo discreto, porém conforme a ótica do algoritmo C4.5. Por exemplo, o algoritmo SPRINT[11], embora tenha passos equivalentes, utiliza uma forma diferente de realizá-los. É importante destacar que o passo de avaliação de atributo está subdividido em dois (o de atributo contínuo e o de atributo discreto) em virtude de apresentarem diferentes complexidades, embora tenham a mesma essência. Ainda sobre a mesma figura, percebemos como os conceitos de processo e de transação realizam uma integração entre os conceitos de aplicação e de procedimento, próprios do domínio de MD, e os de *job* e de tarefa, inerentes ao domínio de grades. Finalmente, vemos que o conceito de grade é declarado como uma entidade que possui, dentre outras coisas, máquinas para o desempenho de tarefas e *middleware* para o seu próprio gerenciamento e escalonamento dessas tarefas, onde exemplificamos, dentre os que lidam com aplicações "Bag-of-Taks", o *OurGrid*.

3.2. Base de Conhecimento (ou de Regras)

Na base de conhecimento são representadas as regras a serem exploradas durante a execução de aplicações de MD a fim de que o analisador/otimizador de transações possa decidir qual organizador de tarefas irá utilizar.

```
Se (GradeEstaApta)/\~(ExisteCondicaoOtimizacao)
➔ executar Organizador_de_1_procedimento_por_tarefa;

Se (GradeEstaApta)/\(ExisteCondicaoOtimizaçao)/\(transacao.processo.aplicacao.algoritmo.metodo = 'TDIDT')/\
   (transacao.complexidadeProcedimentos() = 'SEMELHANTE')
➔ executar Organizador_de_N_procedimentos_semelhantes_por_tarefa;

Se (GradeEstaApta) /\ (ExisteCondicaoOtimizaçao)/\(transacao.processo.aplicacao.algoritmo.metodo = 'TDIDT')/\
   ~(transacao.complexidadeProcedimentos() = 'SEMELHANTE')
➔ executar Organizador_de_N_procedimentos_diferentes_por_tarefa;
```

Figura 2. Exemplo de regras na base de conhecimento de SMARTBASEG

Na figura 2, temos uma amostra da base de conhecimento heurístico de otimização, onde algumas das regras armazenadas provocam a execução de diferentes organizadores de tarefas, de acordo com a situação. Alguns conceitos como GradeEstaApta (ou seja, grade em funcionamento e com pelo menos uma máquina ligada) e ExisteCondiçãoOtimização (por exemplo, se os procedimentos são independentes e idempotentes e sua quantidade for maior que o número de máquinas) são atribuídos por regras não mostradas na figura. Dito isso, podemos ver que a 1ª. regra simplesmente organiza um procedimento por tarefa, já que não há condição de otimização, criando, portanto, um *job* que tem uma quantidade de tarefas igual ao de procedimentos de MD. Já a 2ª. regra considera que se pode otimizar os procedimentos de TDIDT de mesma complexidade, apenas distribuindo-os em igual quantidade por tarefa, criando assim um *job* que tem, no máximo, um número de tarefas igual ao de máquinas *on line* da grade. Finalmente, a 3ª. regra determina que, para procedimentos de complexidades diferentes, será necessário um organizador que faça uma distribuição equilibrada desses procedimentos entre as tarefas, objetivando o balanceamento de carga, ou seja, que as tarefas criadas exijam um esforço computacional parecido. Nesse último caso, o *job* tem, no máximo, um número de tarefas igual ao de máquinas *on line* da grade, assim como o gerado pela 2ª. regra.

4. Implementando uma aplicação de MD usando SMARTBASEG

Para uma validação de nossa proposta, foi implementado um algoritmo TDIDT, no caso, o C4.5 [19]. Este algoritmo executa sucessivos passos de divisão do conjunto de

treinamento inicial enquanto constrói a árvore de decisão. Depois de cada iteração, C4.5 escolhe um atributo para ser a raiz da árvore de decisão e divide o conjunto de treinamento por cada valor do atributo escolhido. O procedimento de escolha do melhor atributo para ser a raiz da árvore é o de maior custo no algoritmo e é potencialmente paralelizável, já que, para a sua realização, é necessária a execução de um outro procedimento (avaliação de atributo) para cada um dos atributos envolvidos. Todas essas avaliações podem ser feitas em paralelo, já que são independentes entre si.

A aplicação que implementou esse algoritmo (chamada C45Aplic01) usou interfaces Java que SMARTBASEG disponibilizou para a criação de seus componentes de MD para grade. Internamente, SMARTBASEG tem implementadas as ontologias de MD e grades (seção 3.1), geradas pela ferramenta Protégé [20], além de algumas heurísticas (seção 3.2), geradas pela ferramenta JEOPS [21], para melhor direcionar o uso da grade durante a execução de C45Aplic01. Uma das heurísticas usadas foi a de realizar a criação de tarefas baseando-se exclusivamente no número de máquinas disponíveis na grade. Neste caso, se um conjunto de transações de um mesmo processo tiver 50 avaliações de atributos, e a grade possuir somente 10 máquinas disponíveis, será gerado um *job* com 10 tarefas, cada uma delas contendo 5 avaliações de atributos. Outra heurística usada realiza a criação de tarefas levando-se em consideração o conhecimento sobre os procedimentos da aplicação de MD, pois eles podem ter complexidades diferentes. Neste caso, se um conjunto de transações de um mesmo processo tiver 50 avaliações de atributos (20 contínuos e 30 discretos), e a grade possuir somente 10 máquinas disponíveis, será gerado um *job* com 10 tarefas, cada uma delas contendo 2 avaliações de atributos contínuos e 3 de atributos discretos.

5. Avaliação Experimental

Nesta seção, nós exemplificamos como o uso de heurísticas definidas em SMARTBASEG influenciam o desempenho de aplicações de MD na grade. Para a avaliação das heurísticas, foram geradas sete bases de dados artificiais contendo 100 mil exemplos e 16 atributos cada uma. A 1ª. base de dados tem somente atributos discretos, e a 2ª., somente contínuos. As outras bases de dados têm atributos contínuos e discretos em diferentes proporções. A grade utilizada tem 10 máquinas (PC´s) de igual capacidade.

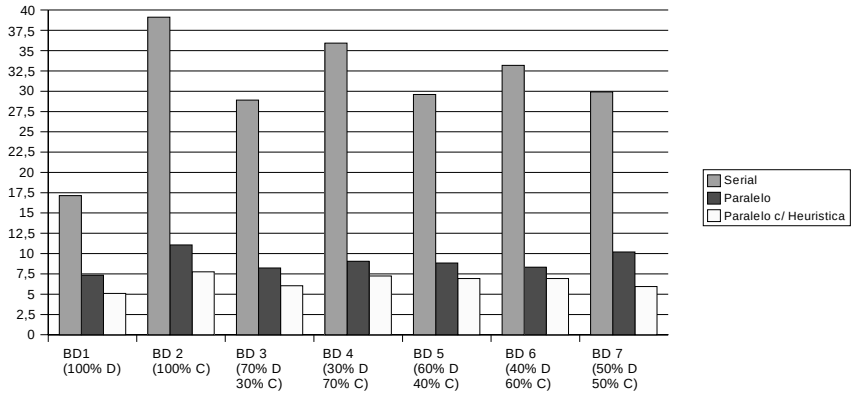

Figura 3. Tempos de execução de 3 implementações do algoritmo C4.5

Na figura 3, vemos o tempo de execução de implementações (aplicações) do algoritmo C4.5 minerando as sete diferentes bases de dados. Primeiramente, a aplicação com a versão serial do algoritmo foi executada em uma máquina local sem o uso da grade. Em seguida, a aplicação de C4.5 que usa a grade diretamente foi executada.

Finalmente, a aplicação C45Aplic01, que se utiliza dos componentes de MD para grade de SMARTBASEG e, conseqüentemente, do seu serviço de otimização (utilizando as heurísticas descritas anteriormente), foi executada. Neste último caso, para a 1ª. e 2ª. bases de dados, somente a heurística de balanceamento das tarefas foi aplicada enquanto que para as demais, o balanceamento de tarefas com a heurística de conhecimento da aplicação foi preferida.

Focando nossa análise somente nas duas versões de C4.5 para grade, já que a maioria das máquinas usadas eram de domínio administrativo local, inviabilizando uma comparação conclusiva com a versão serial, os resultados mostram que o uso de heurísticas melhora o desempenho do algoritmo C4.5. O balanceamento de tarefas com as heurísticas gerou maior ganho independentemente do percentual de atributos discretos e contínuos. Note que estes testes não são exaustivos, já que o que objetivamos é exemplificar a implantação de uma heurística que faça uso das ontologias e permita, assim, uma otimização na execução de aplicações de MD de forma totalmente transparente ao desenvolvedor e ao usuário dessas aplicações. A eficiência da camada de otimização é, portanto, dependente da qualidade do conhecimento representado quer seja através das ontologias, quer seja através das regras heurísticas.

6. Conclusão

Neste artigo, apresentamos a arquitetura SMARTBASEG, que parte do princípio de que as aplicações de MD podem ser caracterizadas em termos de uma ontologia, e que esta caracterização possibilita a definição de uma camada de otimização que pode decidir a melhor forma de submeter procedimentos dessas aplicações a uma grade. Nós descrevemos um exemplo de uso de heurísticas pela camada de otimização para tornar a aplicação de MD mais eficiente ao ser executada em uma grade. A principal contribuição deste trabalho é demonstrar que é possível fornecer um serviço de otimização de desempenho durante a execução de aplicações de MD em grades, a partir do uso efetivo de conhecimento sobre grades e, principalmente, sobre as aplicações sendo executadas, evitando que elas próprias tenham que fazê-lo e abstraindo-lhes também os conceitos de grade e de paralelismo.

Atualmente, nossas pesquisas visam identificar outros algoritmos cujas características peculiares possam suscitar novas heurísticas e/ou possibilidades de paralelização que sejam interessantes de implementar em SMARTBASEG..

Agradecimento

Este trabalho foi desenvolvido com a ajuda parcial da HP do Brasil

Referências Bibliográficas

1. Fayyad, U. M., Data mining and knowledge discovery: making sense out of data, IEEE Expert, pp. 20-25, outubro de 1996.
2. Hinke, H. T. e Novotny, J. Data Mining on NASA´s Information Power Grid. Ninth IEEE International Symposium on High Performance Distributed Computing, 2000.
3. Furtado, V., Data Mining in Grid. Technical Report 1, Projeto HP/UNIFOR: Mineração de Dados em Grade, 2004.
4. Mehta, M., Agrawal, R., e Rissanen, J., SLIQ: *a fast scalable classifier for data mining*. Proceedings of the Fifth International Conference on Knowledge Discovery in Databases and Data Mining, Montreal, Canadá, agosto de 1995.

5. Kargupta, H., Park, B., Hershbereger, D. e Johnson, E., Collective Data Mining: A New Perspective Toward Distributed Data Mining. Advances in Distributed and Parallel Knowledge Discovery, Eds: Hillol Kargupta and Philip Chan. MIT/AAAI Press, 1999.
6. Darlington, J., Guo, Y., Sutiwaraphun, J. e To, H. W., Parallel Induction Algorithms for Data Mining. Advances in Intelligent Data Analysis Reasoning about Data, Second International Symposium, IDA-97. Proceedings, pp. 437-445, 1997.
7. Hall, L. O., Chawla, N., e Bowyer, K. W., Combining Decision Trees Learned in Parallel. Fourth International Conf. on KDD and Data Mining. AAAI Press, 1998.
8. Tsai, Shu-Tzu e Yang, Chao-Tung,, Decision Tree Construction for Data Mining on Grid Computing, 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04), pp. 441-447, 2004.
9. Kubota,K., Nakase, A., Implementation and Performance Evaluation of Dynamic Scheduling for Parallel Decision Tree Generation, 15th International Parallel and Distributed Processing Symposium (IPDPS'01) Workshops, 2001.
10. Heath, D., Kasif, S. e Salzberg, S., "Committees of Decision Trees". B. Gorayska and J.L. Mey, eds., *Cognitive Technology: In Search of a Humane Interface,* pp. 305-317. 1996.
11. Shafer, J. C., Agrawal, R. e Mehta, M., *SPRINT: A scalable parallel classifier for data mining*. In Proceedings of the Twenty-Second International Conference on Very Large Databases, pp. 544-555, Bombaim, Índia, 1996. Morgan Kaufmann.
12. Silva, F. , Carvalho, S., Hruschka, E., A Scheduling Algorithm for Running Bag-of-Tasks Data Mining Applications on the Grid, Proceedings of the Euro-Par 2004, agosto-setembro de 2004.
13. The Astrogrid Project. Phase A Report. Database Technology and Data Mining, http://wiki.astrogrid.org/bin/view/Astrogrid/RbDataminingTechnologyReport, acessado em agosto de 2005.
14. Banino C., Beaumont O., Carter, L. Ferrante, J., Legrand, A., Robert, Y., Scheduling Strategies for Master-Slave Tasking on Heterogeneous Processor Platforms, In Applied Parallel Computing: Advanced Scientific Computing: 6th International Conference (PARA'02), volume 2367 of LNCS, pp. 423-432. Springer, 2002.
15. Orlando, S., Palmerini, P., Perego, R. e Silvestri, F., Scheduling high performance data mining tasks on a data Grid environment, Euro-Par, pp. 375-384, 2002.
16. Galstyan, A., Czajkowski, K., Lerman, K., Resource Allocation in the Grig Using Reinforcement Learning. AAMAS 2004, julho de 2004.
17. Cirne, W., Paranhos, D., Costa, L., Santos-Neto, E., Brasileiro, F., Sauve, J., da Silva, F. A. B., Osthoff, C., and Silveira, C., Running bag-of-tasks applications on computational grids: The MyGrid approach. Proceedings of the ICCP'2003 - International Conference on Parallel Processing, outubro de 2003.
18. Cannataro, M., Comito, C., A Data Mining Ontology for Grid Programming, First International Workshop on Semantics in Peer-to-Peer and Grid Computing, 2003.
19. Quinlan, J.R., C4.5: Programs for Machine Learning. Morgan Kauffmann, 1992.
20. Protégé. http://protege.stanford.edu, acessado em agosto de 2005.
21. JEOPS - The Java Embedded Object Production System, versão 2.1, http://www.di.ufpe.br/~jeops, acessado em agosto de 2005.