Análise de Conglomerados Aplicada ao Reconhecimento de Padrões de Uso de Recursos Computacionais

Germano Capistrano Bezerra
Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

29 de janeiro de 2006

Agradecimentos

Agradeço ao Prof. Dr. Marcelo Finger pela confiança em mim depositada, pela orientação que recebi ao longo de todo o desenvolvimento deste trabalho e por sua solicitude ímpar, especialmente no momento de finalização desta dissertação.

Agradeço a meus pais, Aluísio e Aldênia, pelos ensinamentos de vida que sempre me deram, especialmente o de valorizar a educação como um dos elementos fundamentais para meu crescimento pessoal. Agradeço a meu pai pela leitura prévia deste trabalho e pelas diversas sugestões de melhoria que recebi.

Agradeço à minha esposa Débora, por seu incentivo e apoio. Também ajudou-me lendo e revisando esta dissertação, sendo fundamental para a melhoria na qualidade do texto final.

Agradeço aos diversos amigos que me incentivaram e ajudaram durante toda minha jornada.

Resumo

A computação em grades oportunistas tem por objetivo compartilhar recursos computacionais já existentes nas instituições para escalonamento de tarefas. A decisão de alocar uma máquina para realização de determinada tarefa pode ser facilitada se for possível a predição de ocorrência de ociosidade nessa máquina, considerando sua utilização pelo usuário original.

Este trabalho investiga a utilização de técnicas de análise de conglomerados para o reconhecimento de padrões de uso de recursos computacionais. Para tanto, serão identificados comportamentos prototípicos de utilização de um determinado recurso. Assumindo-se que a ocorrência desses padrões de comportamento é recorrente ao longo do tempo, é possível estabelecer um arcabouço para estimar se, em um dado momento, este recurso estará disponível ou não para compartilhamento com a rede.

Diversos conceitos envolvidos durante o processo de análise de conglomerados são discutidos e aplicados ao contexto do reconhecimento de padrões de uso de recursos computacionais. A implementação de um módulo de software possibilitou que fossem realizados alguns ensaios para validação das técnicas propostas.

Abstract

Opportunistic grid computing systems aim to share computer resources, already existent in institutions, in order to schedule computational applications. The choice of a machine to run some task will be aided by a resource idle prediction function.

In this work, cluster analysis will be applied to perform resource usage pattern recognition. Prototypic behaviors of resource usage will be identified. Based on the recurrence of these behaviors, a pattern recognition framework will estimate the sharing availability of a resource.

Some cluster analysis concepts will be discussed and applied to the computer resources usage pattern recognition framework. A software component was developed in order to perform the proposed techniques validation through simulations.

Conteúdo

Capítulo 1

Introdução

Os computadores têm desempenhado papel importante na sociedade moderna, tendo sido a figura central da revolução tecnológica ocorrida nas últimas décadas. Antes restritos a centros de pesquisas avançadas, eles passaram a fazer parte do cotidiano das pessoas, seja no meio acadêmico, corporativo ou familiar. A proliferação de seu uso fez com que fossem estruturadas redes com algumas centenas ou milhares de computadores interligados. Essas redes muitas vezes apresentam níveis elevados de ociosidade, uma vez que muitos computadores são utilizados plenamente apenas por períodos de tempo limitados. Por outro lado, otimizar o uso dos recursos, sejam quais forem suas naturezas, tem sido um desafio cada vez maior.

Computação em grade é uma tecnologia para que recursos computacionais distribuídos geograficamente possam ser compartilhados e integrados de maneira eficiente e disponibilizados aos seus clientes em forma de serviço [1]. Os recursos a serem utilizados para processar uma determinada tarefa são selecionados dinamicamente, em tempo de execução, respeitando os requisitos de níveis de serviço dos usuários originais dos recursos. De acordo com este paradigma, uma grade é uma rede de computadores na qual podem ser conectadas máquinas, com intenção de compartilhar de forma gerenciada alguns de seus recursos, socializando o seu uso pelos diversos processos a serem executados.

1.1 InteGrade

O projeto de pesquisa InteGrade [2], em desenvolvimento no IME-USP em parceria com outras instituições de ensino e pesquisa, visa o estudo e implementação de uma infra-estrutura para grades de computadores. O objetivo do InteGrade [3] é alavancar o poder de processamento dos recursos computacionais já existentes em instituições. Muitas vezes esses recursos ficam ociosos e o InteGrade fará uso da capacidade inativa das máquinas, constituindo dessa forma uma grade oportunista. Isto minimizará os custos de implantação de um ambiente tecnológico para processamento de aplicações computacionalmente complexas.

Um dos principais requisitos do InteGrade é que a sobrecarga causada por sua utilização seja quase imperceptível pelos usuários originais das máquinas que compõem a grade. Estes usuários terão prioridade na utilização dos recursos de suas máquinas. Como os recursos podem ser requisitados por seus usuários originais a qualquer momento, a tarefa de designar máquinas da grade para a execução de aplicações, possivelmente paralelas, poderá ser bastante complexa.

Para auxiliar o escalonamento de aplicações, a arquitetura do InteGrade prevê o desenvolvimento de um componente de software para reconhecimento de padrões de uso dos recursos computacionais que compõem a grade. O objetivo deste componente é estimar o estado futuro de disponibilidade de um determinado recurso. O serviço de escalonamento poderá fazer uma melhor distribuição de carga no sistema, atribuindo tarefas a recursos com maior probabilidade de ociosidade durante o tempo de execução das aplicações.

A princípio, nada ou muito pouco se sabe sobre como se comportarão os recursos da grade quanto à sua disponibilidade. Não haverá um conjunto de padrões de comportamento já previamente conhecidos. O InteGrade deverá aprender como cada uma das máquinas da grade se comporta. Isso será possível através de um processo de aprendizado constante, baseado no monitoramento contínuo da utilização dos recursos computacionais das máquinas, tais como memória RAM, memória SWAP, CPU e espaço em disco rígido.

1.2 Comportamentos Prototípicos

O monitoramento da utilização histórica dos recursos criará os elementos que servirão de base para o treinamento do sistema. Espera-se que o conhecimento de como os recursos foram utilizados no passado leve à identificação de padrões de comportamento que possivelmente se repetirão no futuro. Esses padrões representam os diferentes modos de operação das máquinas, seus comportamentos prototípicos.

Para resolver o problema de identificação de padrões do InteGrade, será necessária a aplicação de técnicas de aprendizado não-supervisionado, uma vez que comportamentos pré-classificados não estarão disponíveis para treinamento.

Uma das técnicas de aprendizado não-supervisionado existentes, tema central deste trabalho e que aborda o problema de dividir dados em subconjuntos disjuntos, é a análise de conglomerados (em inglês, *clustering*). Esta técnica será aplicada e testada quanto à sua adequação na solução do problema de reconhecimento de padrões de utilização dos recursos computacionais do InteGrade.

A definição de análise de conglomerados está intimamente ligada ao conceito de *cluster*, termo em inglês utilizado para designar um conglomerado de objetos. A técnica consiste em distribuir uma determinada massa de dados em *clusters*, de modo que os elementos de um determinado *cluster* se assemelhem mais entre si, de acordo com algum critério estabelecido, do que com elementos de outros *clusters*.

O monitoramento citado anteriormente permitirá a construção de uma massa de dados contendo objetos que representam a utilização histórica dos recursos

computacionais. Cada máquina que compõe a rede possivelmente é operada de vários modos distintos, dependendo das finalidades para as quais ela é utilizada.

Submetendo-se a massa de dados históricos ao processo de *clustering,* serão criados conglomerados de objetos semelhantes. Os objetos de um mesmo conglomerado provavelmente representam exemplos de um mesmo modo de operação. Dessa forma, espera-se que a partição gerada pelo algoritmo de *clustering* divida a massa de dados em conglomerados que representem os diversos comportamentos prototípicos de utilização dos recursos das máquinas.

Como um conglomerado representa um comportamento prototípico, é necessário estabelecer meios para se determinar como caracterizar esse comportamento. Tal caracterização indicará o modo de operação esperado para os elementos pertencentes ao conglomerado e será realizada por um objeto do mesmo tipo que os objetos que representam a utilização histórica.

Um objeto submetido ao *clustering* representa a operação da máquina, no passado, durante um determinado intervalo de tempo. O objeto que representa o comportamento prototípico, denominado objeto representativo do conglomerado, também representa a operação da máquina durante um intervalo de tempo de mesmo tamanho. Esse objeto representativo representa a utilização esperada dos recursos pelos elementos pertencentes ao respectivo conglomerado.

Há várias maneiras de se construir o objeto representativo. Normalmente segue-se uma das duas diretrizes: *(i)* cria-se um objeto com as informações de utilização média dos recursos apresentada pelos objetos originais que compõem o conglomerado ou *(ii)* escolhe-se o objeto do conglomerado que melhor represente os demais (de acordo com alguma métrica de avaliação). Neste trabalho, utilizou-se a primeira diretriz.

Para exemplificar um comportamento prototípico, sejam os objetos representados na Figura 1.1. Cada objeto é caracterizado com informações de utilização percentual de memória física, por um determinado intervalo de tempo, de uma determinada máquina. Considere-se que esses objetos foram construídos a partir do monitoramento de utilização histórica da máquina em questão e que foram submetidos, juntos com outros não demonstrados na figura, ao processo de *clustering.*

Esses quatro objetos (A, B, C e D) representam comportamentos semelhantes, com moderada utilização de memória no início do período seguida de elevados níveis de uso na segunda metade do período. Suponha-se então que esses objetos, e apenas esses, foram incorporados em um mesmo conglomerado após o *clustering* da massa de dados.

O objeto apresentado na Figura 1.2 possui as informações de utilização média do conglomerado formado pelos objetos A, B, C e D. É o chamado objeto representativo do conglomerado e caracteriza um dos comportamentos prototípicos de utilização de memória física da máquina.

Após o processo de *clustering* são formados alguns conglomerados. Cada conglomerado diz respeito a um determinado comportamento prototípico e possui seu próprio objeto representativo.

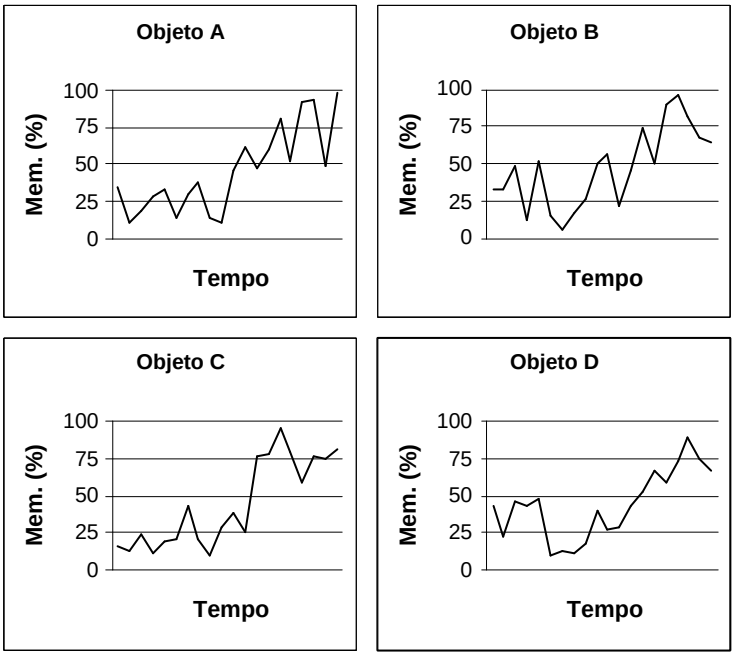

Figura 1.1: Objetos que representam a utilização de memória física em uma determinada máquina.

1.3 Reconhecimento de Padrões

Uma das principais aplicações de análise de conglomerados é a predição baseada em grupos [4]. Para que a predição seja possível, a massa de dados disponível é subdividida em *clusters* (ou conglomerados). A cada conglomerado constituído é atribuído um objeto representativo, que simboliza a caracterização média dos elementos do conglomerado.

Com a partição dos dados em *clusters*, o algoritmo de predição deve ser capaz de determinar a que *cluster* pertence um determinado elemento não classificado (que não foi submetido inicialmente ao *clustering*), com algumas características desconhecidas. Tal tarefa é realizada através da comparação entre as características conhecidas desse elemento e as características análogas dos diversos objetos representativos existentes.

A predição das características desconhecidas do objeto não classificado é feita com base nas características do objeto representativo do conglomerado ao qual aquele objeto foi atribuído.

O reconhecimento de padrões proposto neste trabalho é desenvolvido através desta técnica de predição baseada em grupos. A massa de dados obtida a partir do monitoramento de utilização histórica dos recursos é subdividida em *clusters*.

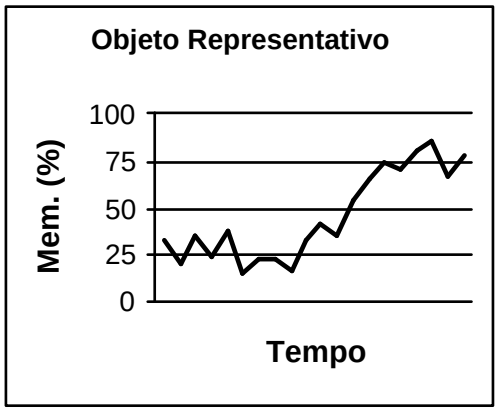

Figura 1.2: Objeto representativo do conglomerado formado pelos objetos A, B, C e D.

Como visto na seção anterior, esses *clusters* representam os comportamentos prototípicos de utilização de determinado recurso computacional.

Quando da chegada de requisição de uso de uma máquina, por parte do InteGrade, o algoritmo de predição deve estimar qual o modo corrente de operação (comportamento prototípico) dessa máquina com relação à utilização de seus recursos computacionais. Como os diversos comportamentos prototípicos são conhecidos, para que a predição seja possível é necessário que se conheça algumas medidas de utilização dos recursos no momento da requisição.

Identificado o modo de operação (no momento da requisição) de um recurso o algoritmo de predição assume que a utilização em um futuro próximo desse recurso seguirá o mesmo padrão do comportamento prototípico identificado. Assim, é possível estimar se o recurso estará disponível para compartilhamento com a grade. Essa predição é fundamental para auxiliar o escalonamento de aplicações, minimizando a migração de processos pela rede, já que os usuários originais das máquinas que formam a grade têm prioridade na utilização dos recursos.

Para ilustrar esse processo, considere-se que a partição dos dados de utilização histórica de memória física de uma máquina formou conglomerados caracterizados pelos objetos representativos apresentados na Figura 1.3. O *Cluster I* representa o mesmo comportamento prototípico apresentado na seção anterior.

Assim, o monitoramento de utilização histórica de memória permitiu que fossem identificados 3 comportamentos prototípicos. Suponha-se que essa máquina foi requisitada pelo InteGrade em um momento tal que a utilização recente de memória física é ilustrada na Figura 1.4.

Comparando o objeto que representa a utilização recente de memória fí-

sica com os comportamentos prototípicos existentes, é possível identificar que o *Cluster I* é o conglomerado cujo objeto representativo mais se assemelha ao objeto representando a operação atual.

Assim, o algoritmo de predição assume que a utilização de memória num futuro próximo dar-se-á conforme previsto nas características do objeto representativo do *Cluster I*. Assim, estima-se que a utilização de memória por parte do usuário original da máquina deverá ser intensa durante um período imediatamente posterior ao da requisição.

Os objetos não classificados, representando a utilização recente dos recursos das máquinas, sempre serão incompletos quando comparados aos objetos representativos dos comportamentos prototípicos. As informações desconhecidas desses objetos são exatamente aquelas que se deseja inferir. Se, entretanto, o objeto não classificado possuir uma quantidade muito pequena de informações conhecidas, a identificação de qual conglomerado a que ele pertence pode ser muito falha. Isso comprometeria a qualidade da predição proposta. Como será visto na Seção 3.2, esse fato é contornado com uma adequada representação dos elementos que serão submetidos ao *clustering*.

1.4 Objetivos

Este trabalho possui alguns objetivos básicos. Além de documentar diversos aspectos relevantes da teoria sobre análise de conglomerados, pretende-se verificar:

- A previsibilidade da ociosidade de recursos — O escalonamento de aplicações no InteGrade deve priorizar a alocação de tarefas para as máquinas cujos recursos apresentarem maior probabilidade de ociosidade durante o tempo de execução. A intenção é minimizar a possibilidade de ocorrência de migração de processos na rede. É importante, portanto, que seja verificada se essa ociosidade é previsível. A proposta para predição de ocisiosidade é baseada na observação dos dados passados de utilização das máquinas.

- A aplicabilidade de técnicas de *clustering* — As técnicas de análise de conglomerados podem ser utilizadas para implementar mecanismos de predição baseada em grupos. Considerando a existência de comportamentos prototípicos, esse trabalho visa estudar a aplicabilidade dos algoritmos de *clustering* para a predição de ociosidade de recursos computacionais. Isso sendo verificado, será possível prover o InteGrade de sofisticado modelo de monitoramento dos modos de operação das máquinas, otimizando o processo de escalonamento de aplicações.

Serão estudadas diversas possibilidades para a implementação do processo de análise de conglomerados aplicada ao reconhecimento de padrões.

1.5 Estrutura do Documento

O desenvolvimento deste trabalho está descrito conforme estrutura apresentada a seguir.

Contextualizando as técnicas de *clustering* como uma ferramenta de treinamento, o Capítulo 2 traz uma introdução sobre aprendizado computacional. São descritos brevemente alguns conceitos gerais e apresentados os dois modelos básicos de treinamento: supervisionado e não-supervisionado.

O Capitulo 3 apresenta o processo de análise de conglomerados propriamente dito, estabelecendo um elo entre a teoria e a aplicação realizada ao reconhecimento de padrões de uso de recursos computacionais no InteGrade.

Os ensaios realizados para validar a aplicação realizada com as técnicas de *clustering* são descritos na Capítulo 4. Além da metodologia utilizada nos ensaios, são descritas as métricas para interpretar os resultados obtidos. Esses resultados são discutidos ao final do capítulo.

Finalmente, o Capítulo 5 traz as conclusões finais do trabalho e apresenta sugestões de trabalhos futuros que podem ser desenvolvidos a partir do que aqui foi apresentado.

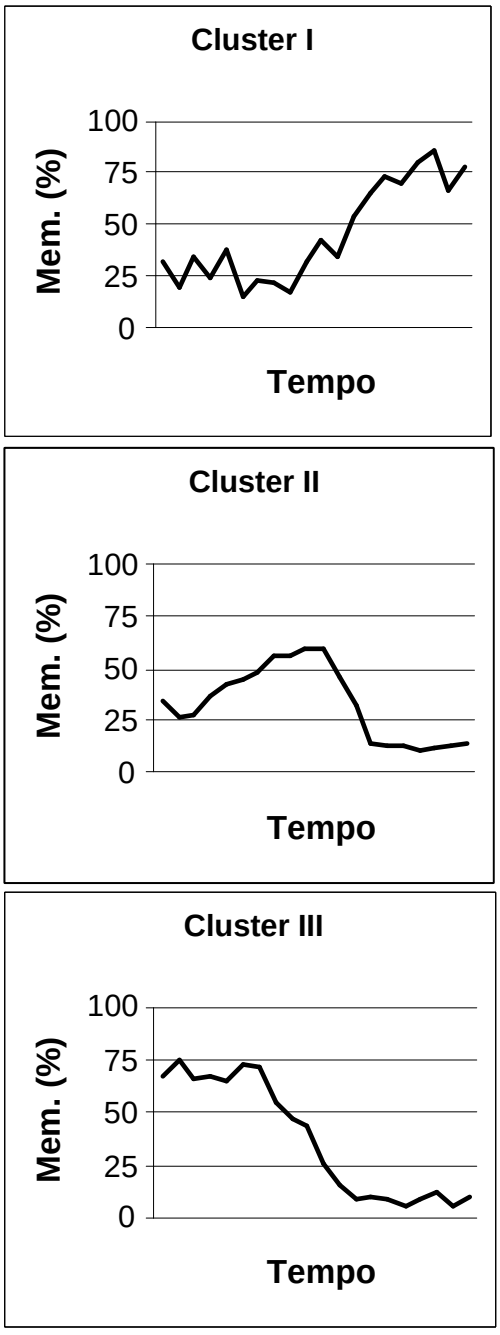

Figura 1.3: Partição dos dados históricos em três conglomerados, representando os modos de operação da máquina com relação à utilização de memória física.

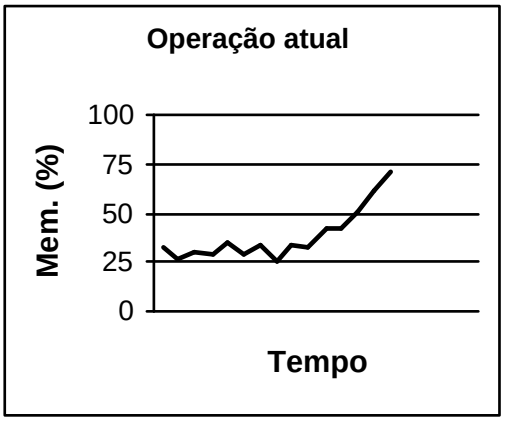

Figura 1.4: Monitoramento recente de utilização de memória física.

Capítulo 2

Aprendizado Computacional

2.1 Conceitos Gerais

Aprendizado computacional (ou automático) é um ramo da ciência da computação que começou a despertar a atenção dos pesquisadores a partir da metade do século passado, com o desenvolvimento das ciências cognitivas e da inteligência artificial. Com aplicação em diversos contextos, esse ramo de pesquisa tem sido bastante desenvolvido, trazendo o entendimento de que o aprendizado desempenha papel central nos sistemas que lidam com inteligência.

Pode-se definir aprendizado computacional como o estudo de mecanismos utilizados por sistemas inteligentes, visando melhorar seus desempenhos com o tempo. Esses mecanismos estão intimamente relacionados com a aquisição de conhecimento a partir da experiência em algum ambiente.

Pat Langley [5] propõe um arcabouço para a implementação do aprendizado computacional baseado em quatro termos básicos dessa definição:

- Conhecimento — organização dos dados em alguma estrutura interna obtida da experiência através da manipulação de dados por sistema ou intervenção manual.
- Experiência — observações realizadas em algum domínio do conhecimento, expressas como medição de alguma grandeza física.
- Ambiente — contexto em que se define o problema a ser analisado, podendo ser caracterizado através do objetivo do processo de aprendizado, da supervisão desse processo, da forma de aquisição dos dados de entrada, da regularidade e das configurações externas às quais está exposto.
- Desempenho — medida quantitativa de algum aspecto do comportamento de uma tarefa relacionada ao aprendizado desenvolvido. Mudanças capturadas por essa medida, na direção do objetivo estabelecido, caracterizam a melhoria no desempenho.

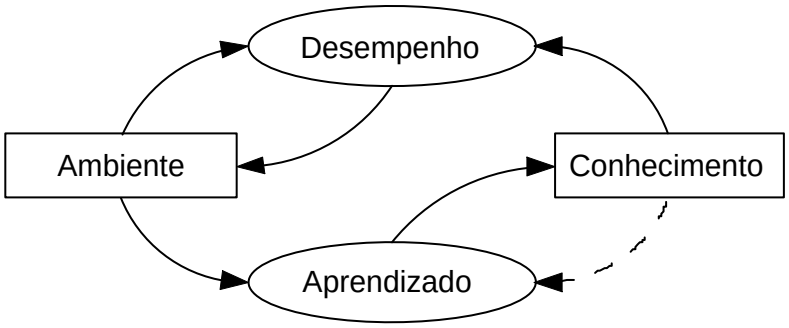

Figura 2.1: Interações entre aprendizado, desempenho, conhecimento e o ambiente, propostas por Pat Langley. A seta tracejada representa uma interação opcional.

A Figura 2.1 ilustra que o aprendizado não pode ser entendido de forma isolada. O ambiente define o contexto em que se dá o aprendizado e fornece as informações, obtidas através da experiência no próprio ambiente, para a construção do conhecimento. O conhecimento é utilizado para a melhoria de alguma métrica de desempenho, definida para o ambiente do aprendizado. O conhecimento pode inclusive ser utilizado para aprimorar o próprio processo de aprendizado, caracterizando o aprendizado constante.

2.1.1 Ambiente de Aprendizado

A caracterização do ambiente em que ocorre o processo de aprendizado depende de alguns fatores. O primeiro deles diz respeito à definição do objetivo de melhoria de desempenho do processo. Todo aprendizado é feito com a intenção de maximizar ou minimizar alguma medida de interesse do tema de estudo.

Outro fator importante é a utilização ou não de supervisão no processo. O aprendizado supervisionado indica a presença de algum tipo de tutor ou especialista auxiliando o início do processo de aprendizado.

Por exemplo, em trabalhos de classificação de dados realizados em ambientes supervisionados, há disponível um conjunto de objetos de treinamento pré-classificados. Já se o aprendizado computacional é aplicado na resolução de problemas, um exemplo de ambiente supervisionado de aprendizado é aquele que conta com o auxílio de um tutor para determinação de alguns resultados esperados. Em ambientes não supervisionados não há presença de tipo algum de tutor.

Também faz parte do ambiente a forma de aquisição dos dados, podendo esta ser tanto pontual como constante. No primeiro caso, todas as informações são obtidas simultaneamente, antes do início do procedimento de análise dos dados. No outro caso, ocorre uma obtenção gradativa da informação, com a aquisição de uma determinada quantidade por vez, ao longo do processo de

aprendizado.

Há também a possibilidade de um modo de operação híbrido. É feita uma coleta inicial de uma quantidade considerável de dados, que são processados logo em seguida. Com o decorrer do tempo, novos eventos geram mais dados que são coletados num procedimento constante.

Outro aspecto a ser considerado na caracterização do ambiente diz respeito à regularidade do mesmo: relevância das informações obtidas, presença de ruídos nos dados de entrada e a consistência do ambiente ao longo do tempo. O processo de aprendizado deve ser capaz de lidar com essas características, disponibilizando filtros de seleção da experiência adquirida.

2.1.2 Natureza da Representação da Experiência

A representação da experiência é fundamental para que o processo de aprendizado possa ser capaz de construir conhecimento a partir dela. A experiência, aqui definida como o conjunto de observações realizadas em algum domínio de competência, normalmente diz respeito à medição de alguma grandeza que pode ser:

- binária — que mede a presença ou ausência de alguma característica;
- atributos enumerados — similar às grandezas binárias, mas que permite mais de dois valores mutuamente exclusivos; ou
- numérica — valor inteiro ou real que representa alguma medida realizada. Essa é a forma de representação mais simples de ser usada. Normalmente, cada objeto analisado (ou medido) é descrito por um conjunto de m características. Assim, a representação desse objeto é feita através de um ponto no espaço vetorial de m dimensões.

2.2 Aprendizado Supervisionado

Os temas abordados por técnicas de aprendizado automático são tipicamente de duas modalidades: classificação de objetos e resolução de problemas. Independentemente de qual a aplicação em análise, foi visto que o mecanismo de aprendizado pode ser caracterizado quanto à ausência ou presença de supervisão do processo.

No aprendizado supervisionado, considerando-se para fins de ilustração sua aplicação à classificação de dados, um conjunto de objetos pré-classificados é disponibilizado por algum tutor ou especialista. Suponha-se ainda que esses objetos são descritos por um vetor de características, composto de valores numéricos.

Assim, supondo a existência de γ objetos pré-classificados, cada um definido por n características, a caracterização de um objeto pode ser feita através de um vetor $\bar{x} \in \mathbb{R}^m$. Cada característica dos objetos é definida por um escalar, $x_{\mathrm{ij}}, 1 \leq i \leq \gamma$ e $1 \leq j \leq m$. O valor x_{ij} representa, portanto, a característica j do objeto i.

Esses objetos pré-classificados compõem o denominado corpo de treinamento. A cada vetor $\bar{x}_i = \{x_{ij} | 1 \leq j \leq m\}$ é associada uma categoria Q_c, indicada no trabalho de pré-classificação. Sendo k o total de categorias existentes, tem-se que $1 \leq c \leq k$.

O trabalho de aprendizado reduz-se ao problema de desenvolvimento de uma função $g(\bar{x})$ que receba como parâmetro um vetor de m dimensões e retorne uma categoria Q_c, $1 \leq c \leq k$. Essa função é denominada função de classificação e deve atender aos requisitos de classificação do corpo de treinamento disponibilizado. Na prática, é definida uma função $\tilde{g}$, uma aproximação de g. No processo de definição dessa função aproximada, tem-se por objetivo minimizar o erro entre g e $\tilde{g}$, de acordo com alguma medida de erro definida. Isso pode ser melhor descrito pelo diagrama da Figura 2.2.

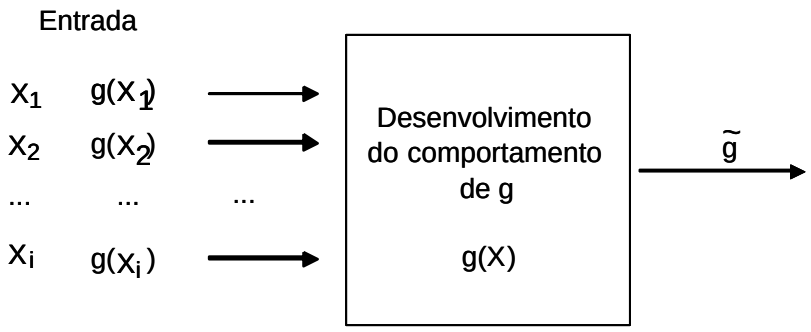

Figura 2.2: Aprendizado supervisionado faz uso de um conjunto de objetos pré-classificados para a construção de uma função aproximada de g.

Com esse procedimento foi obtido um classificador que permite a identificação da classe à qual pertence um objeto m-dimensional qualquer.

A evolução desse modelo pode seguir na direção do aprendizado constante, quando novos objetos classificados passam a aprimorar a medida de desempenho do classificador.

Para isso, é estabelecida uma retroalimentação ao processo de desenvolvimento do comportamento de g, conforme ilustrado na Figura 2.3. Dessa forma, a função $\tilde{g}$ fornece respostas ao próprio processo, que pode adaptar sua saída de forma a melhorar algum indicador de desempenho, normalmente uma medida de erro da função resultante. Cumpre-se o objetivo de otimizar a função de classificação $\tilde{g}$.

Como pode ser observado, a retroalimentação pode fazer uso de um tutor, com conhecimento sobre o domínio do conhecimento ou mesmo do ambiente em que ocorre o treinamento.

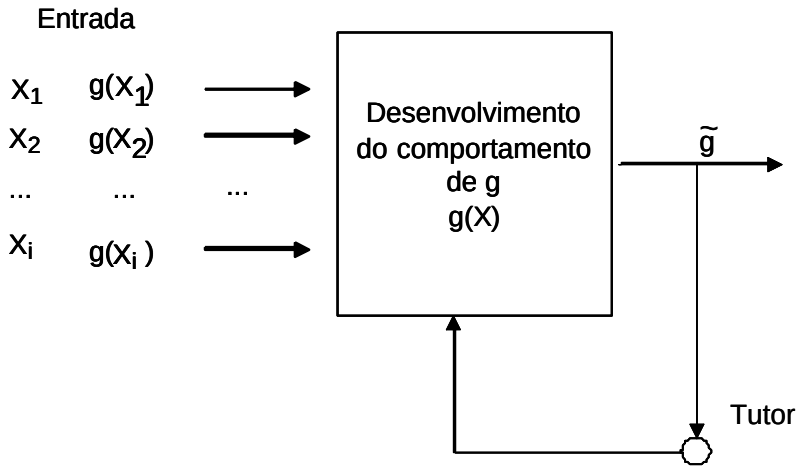

Figura 2.3: Retroalimentação do processo de desenvolvimento da função g promove o aprendizado constante.

2.3 Aprendizado Não-supervisionado

Pode-se referir ao processo de aprendizado não-supervisionado como aquele que se dá sem a existência de um conjunto de treinamento com exemplos pré-classificados ou sem a presença de tutores no domínio do conhecimento em que se dá o aprendizado.

Sem esse corpo de treino, é necessário que inferências sejam feitas sem conhecimento prévio dos objetos de estudo e sem um resultado esperado já definido. Na prática, um dado de entrada deve ser avaliado e classificado sem que, para isso, existam outros objetos de referência com os quais ele possa ser comparado.

Como não há uma massa de dados de exemplo contra a qual confrontar os novos eventos e dados de entrada, um sistema de aprendizado não-supervisionado faz uso de análises de redundância [6] [7] da informação recebida. Esse é um passo fundamental no processo da construção do conhecimento. Essa redundância pode ser obtida na aferição de algumas medidas na massa de dados, tais como a média, a variância e a co-variância de informações sobre os dados ou mesmo a forma como os objetos podem ser combinados ou aglomerados.

A primeira dessas medidas é a média. Alguns fenômenos podem ser analisados com a simples observação do valor médio de ocorrência de uma dada grandeza que os caracteriza, auxiliando no processo de aprendizado.

Por exemplo, mesmo sem o auxílio de técnicas de meteorologia, é possível fazer uma estimativa rudimentar da temperatura climática de um lugar inexplorado observando-se a temperatura desse ambiente ao longo de um determinado tempo e daí se extraindo a média das temperaturas obtidas.

Na maioria dos casos, como no próprio exemplo acima, o conhecimento da

média não pode ser considerado determinante no processo de aprendizado, mas pode ser um de seus valiosos recursos. O próprio cálculo da média está sujeito às diversas metodologias existentes, tais como a média aritmética, média harmônica e a média geométrica, entre outras.

Quando a simples observação da média de uma série de dados não for suficiente para se chegar a uma conclusão definitiva quanto à organização dos dados, pode-se fazer uso da medida de variância. A variância pode ser entendida como o grau de dispersão dos valores observados de uma dada grandeza.

Por exemplo, a média da cotação de um papel financeiro poderia induzir a um erro de avaliação se não fosse observada a variância de cotação desse papel. Mesmo com um bom desempenho médio, uma grande variância histórica dos preços pode ser indicativa de grandes variações potenciais (negativas ou positivas) da cotação desse papel, o que o torna extremamente arriscado.

Outra medida estatística importante é a co-variância, que estabelece a correlação entre os diversos conjuntos de objetos em análise. Um estudo de população de animais, por exemplo, pode valer-se da observação de uma alta correlação entre as quantidades de animais de duas espécies distintas para inferir a população de uma das espécies baseando-se no crescimento populacional da outra espécie.

Finalmente, outra linha de avaliação da ocorrência de redundância nos fenômenos observados é o entendimento de como os objetos se agrupam em classes de objetos com características semelhantes. Esse é o foco de uma das principais técnicas de aprendizado não-supervisionado, conhecida como análise de conglomerados, muitas vezes referenciada pelo termo inglês *clustering*. Essa técnica será discutida com maior detalhamento neste estudo.

2.3.1 Classificação e *Clustering*

A classificação [8] é um dos processos fundamentais na ciência, uma vez que os fatos e fenômenos devem ser ordenados antes de poder-se entendê-los e se desenvolverem princípios que expliquem sua ocorrência. Conceitualmente, entende-se por classificação o processo de ordenação de objetos por suas semelhanças.

Análise de conglomerados, ou *clustering* em inglês, é um tipo de classificação especial, feita sem o auxílio de um corpo de treinamento. Nesse processo, os dados são classificados em *clusters*, grupos de objetos com características semelhantes entre si. A própria definição de *cluster* não é algo muito preciso, mas é possível encontrar algumas alternativas tais como as apresentadas por Everitt [9]:

- "Um *cluster* é um conjunto de entidades que são semelhantes, e entidades de diferentes *clusters* não são semelhantes".
- "Um *cluster* é uma agregação de pontos no espaço amostral tal que a distância entre dois pontos quaisquer de um *cluster* é menor que a distância entre um ponto qualquer deste *cluster* e um ponto não presente neste".

- "Os *clusters* podem ser descritos como regiões conectadas de um espaço de dados multidimensional contendo uma relativamente alta densidade de pontos, separadas umas das outras por regiões contendo uma relativamente baixa densidade de pontos".

É imediata a percepção de que o conceito de *cluster* e, por conseqüência, as técnicas de análise de conglomerados estão baseadas em *(i)* como os objetos estão representados no espaço de dados, *(ii)* como eles podem ser comparados e *(iii)* na própria definição desse espaço.

2.3.2 Reconhecimento de Padrões no Projeto InteGrade

Conforme descrito no capítulo introdutório, o objetivo da aplicação de aprendizado computacional, no contexto do projeto InteGrade, é o reconhecimento de padrões, através da classificação dos modos de operação dos diferentes recursos computacionais de cada máquina que faz parte da rede que compõe a grade.

Não há, a princípio, informações sobre eventuais modos de operação existentes, não sendo possível a formação de um corpo de treinamento com objetos pré-classificados. A aplicação de métodos de análises de conglomerados é, portanto, apropriada.

O uso dos recursos computacionais de uma máquina, como por exemplo CPU e memória física, será monitorado e caracterizará os objetos que serão submetidos ao processo de *clustering*. Um objeto será composto de um vetor de valores reais, onde cada valor é a medida do comportamento da máquina, em relação ao uso de um recurso, num instante de tempo. O vetor resultante, um conjunto de valores instantâneos, representará o comportamento apresentado por esta máquina, num determinado intervalo de tempo.

A utilização de técnicas de análise de conglomerados para o reconhecimento de padrões no projeto InteGrade seguirá uma seqüência de passos básicos propostos por Milligan [10]. Essa abordagem não é obrigatória, mas seguir as etapas propostas é importante para que se consiga contemplar diversos aspectos relacionados ao processo. Muitas vezes pode-se ter a falsa impressão de que a implementação do método ou algoritmo escolhido representa toda a análise, mas trata-se apenas de um dos passos que devem ser seguidos.

De forma geral, um processo de *clustering* compreende:

1. definição dos elementos a serem considerados;
2. seleção das variáveis;
3. padronização e normalização das variáveis;
4. definição das medidas de similaridade ou distinção;
5. escolha do método de agrupamento;
6. número de conglomerados a serem considerados; e
7. testes e interpretação dos resultados.

A seqüência acima pode ser adaptada para as particularidades do ambiente em que se realiza o aprendizado, inclusive com a inclusão ou exclusão de etapas. No projeto InteGrade, como será visto no capítulo seguinte, foi utilizada a seqüência de Milligan com a inclusão de uma etapa relacionada ao reconhecimento de padrões propriamente dito. Nessa etapa, ocorrerá a classificação do comportamento corrente de uma máquina.

Capítulo 3

Análise de Conglomerados

Neste capítulo, é apresentada uma descrição detalhada dos passos citados ao final do capítulo anterior, incluindo a tarefa de reconhecimento de padrões inerente ao tema deste trabalho. Com isso, será realizada a fundamentação teórica da análise de conglomerados [4] [11] [12] [13] [14] [15] e será visto como essa técnica é aplicada ao contexto do projeto InteGrade.

Inicialmente, para justificar a aplicação de técnicas de *clustering*, será feita uma descrição do processo de reconhecimento de padrões proposto. Em seguida, será discutido como serão definidos os objetos que representarão a utilização histórica dos recursos. Os elementos que farão parte da massa de dados, as medidas representadas pelas variáveis que caracterizam esses elementos e as transformações aplicáveis a essas medidas são estudados na seqüência.

Caracterizados os objetos, o próximo passo da análise diz respeito a como esses objetos podem ser comparados entre si, estabelecendo uma medida de semelhança entre eles, fundamental para a partição dos dados em conglomerados. A forma como a partição dos dados é obtida é definida pelos algoritmos de *clustering* propriamente ditos e pelo número de conglomerados considerados, discutidos logo a seguir às medidas de semelhança.

Finalmente, é detalhado o processo de identificação de protótipos e proposto um método para interpretação dos resultados obtidos com a aplicação de *clustering* realizada.

3.1 Processo de Reconhecimento de Padrões

A aplicação das técnicas de análise de conglomerados, no contexto do projeto InteGrade, visa possibilitar o reconhecimento do modo de operação em que se encontra uma determinada máquina, quando a mesma é requisitada, pelo gerenciador da grade, a executar alguma aplicação distribuída. O reconhecimento do modo de operação, por sua vez, é utilizado para a predição de ociosidade dos recursos da máquina requisitada. Esse processo é conhecido como reconhecimento de padrões com uso de técnicas de *clustering* e será discutido nesta seção.

Inicialmente, será realizado um monitoramento da utilização histórica dos recursos computacionais da máquina em questão. Com esse monitoramento serão criados os objetos que representam a utilização dos recursos no passado. Esses objetos compõem a massa de dados original a ser consumida durante o processo de reconhecimento. Conforme descrito na Figura 3.1, essa massa de dados é submetida ao *clustering*. O objetivo é criar uma partição composta de conglomerados contendo os objetos originais, semelhantes entre si.

Os conglomerados representam comportamentos prototípicos, onde cada conglomerado possui um objeto representativo. Como cada objeto submetido ao *clustering* representa o comportamento de utilização de um recurso pela máquina durante um período de tempo, o elemento prototípico de um conglomerado de objetos contém o comportamento esperado para um objeto pertencente a esse conglomerado.

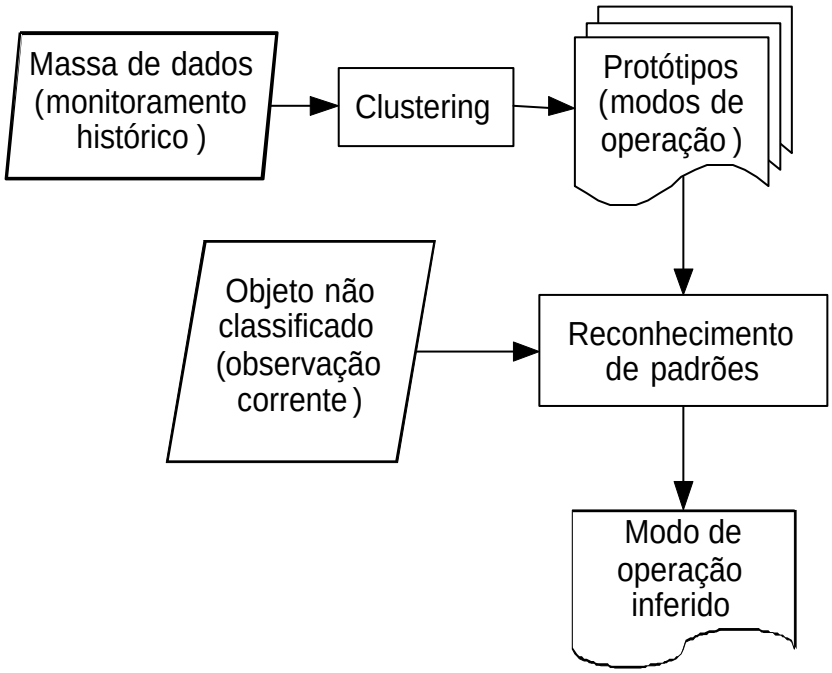

Figura 3.1: Reconhecimento de padrões de uso de recursos computacionais.

Os elementos prototípicos definem os modos de operação padrões. Quando o software gerenciador da grade faz uma requisição por qualquer recurso da máquina, é criado um objeto que representa a utilização dos recurso por um período de tempo imediatamente anterior à requisição.

Esse objeto, ainda não classificado, é submetido ao processo de reconhecimento de padrões. Quando um objeto desconhecido, não classificado, é apresen-

tado ao algoritmo de reconhecimento de padrões, determina-se qual elemento prototípico possui comportamento mais aproximado ao desse objeto não classificado. Esse elemento prototípico é considerado o modo de operação em que o recurso está sendo utilizado no momento da requisição da grade.

Com a identificação do elemento prototípico que mais se assemelha ao objeto não classificado, é possível assumir que o comportamento a ser apresentado pela máquina num futuro próximo seja semelhante ao apresentado pelo elemento prototípico reconhecido. Os valores das dimensões não conhecidas do objeto, que representam a utilização futura do recurso, são inferidos. Dessa forma, é possível realizar a predição de ociosidade de um recurso computacional.

3.2 Definição dos Elementos

A seleção dos elementos que serão considerados no processo de agrupamento é decisiva na determinação da estrutura dos conglomerados que serão criados, uma vez que a metodologia será aplicada sobre esses exemplos selecionados. Apesar disso, a pesquisa a respeito desse tópico de discussão ainda é bem limitada.

Uma técnica que pode ser empregada é a seleção aleatória de exemplos. Quando se pretende generalizar o resultado obtido para um grande número de elementos, pode-se tomar uma amostragem selecionada aleatoriamente da população de objetos disponíveis. Entretanto, esse método de seleção deve ser usado com bastante critério, pois traz o risco de a amostragem selecionada não ser representativa do domínio do conhecimento pesquisado.

A regra básica na escolha dos elementos a serem agrupados estabelece que eles devem representar de forma consistente o conjunto de objetos existentes. A proposta inicial para aplicação ao projeto InteGrade é a consideração de todos os dados coletados pelo processo de monitoramento. Assim, todas as informações obtidas na carga inicial do processo terão influência na formação dos *clusters*.

3.2.1 Objetos do InteGrade

O uso dos recursos computacionais de uma máquina, como por exemplo CPU e memória física, será monitorado e caracterizará os objetos que serão submetidos ao processo de *clustering*. Um objeto será composto de um vetor de valores reais, onde cada valor é a medida do comportamento da máquina, em relação ao uso de um recurso, num instante de tempo. O vetor resultante, um conjunto de valores instantâneos, representará o comportamento apresentado por esta máquina, num determinado intervalo de tempo.

As medidas a serem realizadas no processo de aprendizado dizem respeito à quantidade em uso dos recursos computacionais, como por exemplo CPU e memória física. Cada valor do vetor que caracteriza o objeto representa a medida de uso desse recurso num instante de tempo.

Apenas os objetos do mesmo tipo serão comparados entre si no processo de agrupamento.

O objetivo final da análise de conglomerados é realizar o reconhecimento de padrões, através da identificação de modos de operação prototípicos. É razoável admitir que esses modos de operação possam ser observados num período de um dia de utilização das máquinas. Assim, os objetos submetidos ao *clustering* no InteGrade representam dias de trabalho.

De fato, um objeto conterá informações de uso de um determinado tipo de recurso por um período de 48 horas. Esse período se inicia à meia-noite de um dia e se estende até a segunda meia-noite seguinte. Apesar de um dia de utilização da máquina (24 horas) ser um bom intervalo de tempo para a representação dos objetos, o modelo de 48 horas foi adotado para possibilitar a implementação de um método mais eficiente de reconhecimento de padrões.

O objeto não classificado que será apresentado ao algoritmo de reconhecimento representará um dia, ainda não totalmente conhecido, de utilização de recursos da máquina. Submetê-lo ao algoritmo de reconhecimento é o mecanismo proposto para a inferência da utilização ainda não conhecida.

Se o intervalo de tempo utilizado na representação dos objetos fosse apenas de 24 horas, a identificação do modo de operação realizada no início desse período, durante a madrugada por exemplo, não seria eficiente. Isso porque a caracterização do objeto seria muito pobre, contendo apenas algumas poucas informações sobre a utilização da máquina no presente. Uma comparação com os modos de operação prototípicos poderia levar a uma incorreta definição do modo corrente de operação da máquina.

Seja, por exemplo, o objeto descrito na Figura 3.2 representando a utilização recente de memória física da mesma máquina tomada como exemplo na Seção 1.3. Os comportamentos prototípicos dessa máquina estão descritos no capítulo introdutório, na Figura 1.3.

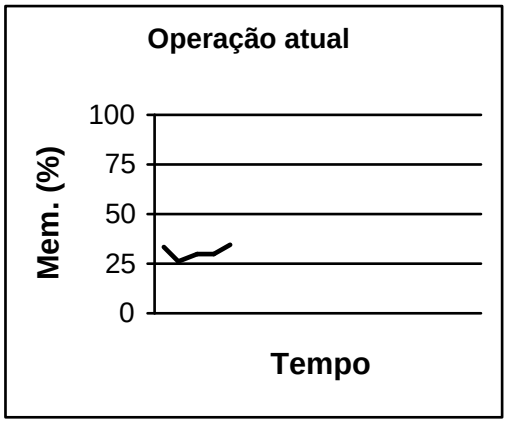

Figura 3.2: Objeto representando a utilização recente de memória física.

Se houver requisição por uso dessa máquina, no momento em que só sejam

conhecidas as informações representadas no gráfico, pouco pode ser afirmado a respeito do modo atual de operação da máquina, em relação ao uso de memória física.

Para evitar esse tipo de situação, cada novo dia continua sendo representado por um objeto diferente, mas o intervalo de representação é de 48 horas. A segunda metade de um objeto sempre coincide com a primeira metade do objeto que representa o dia seguinte.

No momento de uma requisição de uso de um recurso, toma-se, para representar o comportamento presente da máquina, o objeto que se encontra no segundo dia da amostragem de 48 horas. Assim, garante-se que haja pelo menos 24 horas de informações sobre a utilização do recurso.

Quando esses objetos forem comparados aos modos de operação prototípicos, sempre haverá uma quantidade de informação mínima sobre a utilização da máquina no presente, auxiliando o processo de predição da utilização futura. A maneira como será realizado o reconhecimento dos padrões de uso é detalhada posteriormente, na Seção 3.8.

3.2.2 Tipos Ideais

Além da utilização de objetos reais observados no ambiente em que se dá o treinamento, podem-se utilizar objetos artificais, denominados tipos ideais. Durante a pesquisa e modelagem do processo de agrupamento, podem surgir alguns tipos prováveis de serem encontrados nos dados disponíveis. Esses tipos são fabricados manualmente, como um vetor de características incluído na massa de dados. Esse tipo ideal é incluído na massa de dados e submetido ao processamento com os demais objetos.

Após a conclusão do processo de agrupamento, podem-se fazer verificações tais como: verificação se todos os tipos ideais ficaram em conglomerados distintos ou se surgiu algum conglomerado que não foi previsto durante a elaboração dos tipos ideais. Com os resultados obtidos, é possível fazer a remodelagem do processo.

Nas máquinas que compõem a grade gerenciada pelo InteGrade, há expectativa de serem encontrados alguns comportamentos típicos: dia normal de trabalho, dia atarefado e feriado são bons exemplos. A identificação desses tipos de comportamento pode ser feita pelos próprios usuários finais das máquinas, com o auxílio dos administradores da grade.

A existência desses tipos de comportamento deverá estar refletida nos *clusters* finais formados após a partição dos dados. O uso de elementos ideais, tipificados com esses comportamentos, é um recurso simples de ser utilizado e pode ser bastante útil. Por se tratar de etapa transparente ao restante da análise de conglomerados, para a implementação do uso de tipos ideais para validação dos *clusters* formados é suficiente que os usuários das máquinas auxiliem no processo de configuração inicial dos nós da grade.

3.2.3 Ruídos

Finalmente, pode-se também levar em consideração a presença de elementos atípicos nos dados de entrada. Isso se dá porque alguns objetos disponíveis para análise podem não pertencer a nenhum conglomerado apropriadamente, representando uma espécie de ruído de dados.

Alguns métodos disponíveis na literatura, como o método de Ward [16], são resistentes a ruídos. Em linhas bem gerais, esse método faz um agrupamento hierárquico dos elementos (vide Seção 3.6.2), de modo a minimizar a variância (erro) de atribuição de um elemento ao *cluster* a que pertence. Como os ruídos são elementos atípicos, a sua atribuição a um *cluster* natural será dificultada, dado o objetivo de otimização definido.

Uma forma alternativa é uma intervenção manual de retirada de eventuais ruídos e uma posterior análise comparativa de validação com os conglomerados obtidos. Na prática, inclusive no próprio InteGrade, isso nem sempre é possível, dado o volume de dados e a quantidade de dimensões dos objetos envolvidos.

3.3 Seleção das Variáveis

As variáveis escolhidas para caracterizar os objetos têm extrema importância na análise de conglomerados. Elas irão determinar como os objetos serão agrupados, já que a comparação entre os diversos objetos presentes na massa de dados é feita com base nas medidas dessas variáveis.

Apenas as variáveis que realmente têm importância para o agrupamento a ser realizado devem ser consideradas. Um especialista no domínio da informação sobre o qual está sendo desenvolvido o trabalho deve ser envolvido, pois a caracterização de um objeto muitas vezes depende de uma análise subjetiva. Ainda deve ser definido, também com a ajuda desse especialista, como se dará a medição da característica. Para ficarem apropriadas aos métodos de agrupamento, mesmo as grandezas qualitativas devem ser quantificadas.

O grau de correlação entre as variáveis escolhidas também pode ser levado em consideração, uma vez que variáveis bastante correlatas podem acabar dominando a caracterização de um objeto. Isso não necessariamente é ruim, mas o analista deve ter em mente o papel que cada característica deve ter na definição de um objeto.

Outro fenômeno que merece atenção nesse processo é a ocorrência de variáveis que distorcem a caracterização dos objetos (*masking variables*). O uso inadequado de variáveis que não possuam uma forte justificativa para serem consideradas pode levar a uma formação de conglomerados que não reflitam a realidade.

É possível, tomando como exemplo os objetos do InteGrade, que medições realizadas no período noturno não sejam tão relevantes quanto as demais e acabem por interferir indevidamente na caracterização de um objeto.

Muitos estudos práticos, documentados na literatura existente, já concluíram que a simples adição de uma ou duas variáveis como ruído leva a uma

Tabela 3.1: Utilização de memória física em uma determinada máquina.

Instante	*0:10*	*0:20*	*0:30*	*0:40*	*0:50*	*0:60*
Memória	35%	35%	38%	42%	45%	45%

considerável modificação na formação de conglomerados.

Será visto, em seção onde discutir-se-ão as medidas de semelhança entre os objetos, que o uso da distância euclidiana ponderada pode minimizar o efeito da eventual existência desse tipo de variáveis (as chamadas *masking variables*). A distância de Mahalanobis é um exemplo de distância euclidiana ponderada.

3.3.1 Utilização dos Recursos Computacionais

No projeto InteGrade, as variáveis selecionadas para a caracterização dos objetos serão todas as medidas realizadas sobre o uso de um determinado tipo de recurso computacional. Um objeto corresponderá ao comportamento da máquina em um período de tempo.

As medidas realizadas dizem respeito à utilização dos recursos num dado instante de tempo. Ainda, as medidas serão tomadas em intervalos de tempo constantes. Apenas para fins de ilustração, considere-se o exemplo da Tabela 3.1.

Foram realizadas medidas em intervalos de 10 minutos. Cada medida representa, em termos percentuais do total de memória existente, o nível de utilização de memória no instante em que foi realizada. Neste exemplo, foi construído um objeto que representa a utilização de memória física de uma máquina durante o período de 1 hora.

No InteGrade, poderão ser realizadas medidas no intervalo de tempo que se desejar e construir objetos que representem diferentes períodos de comportamento. O módulo de software desenvolvido poderá ser parametrizado de forma a trabalhar no modelo desejado. Para a realização desse estudo, foram construídos objetos que representam o comportamento de uma máquina num intervalo de tempo de 48 horas.

As medidas foram realizadas em intervalos de tempo de 5 minutos. No caso específico do uso de recursos de CPU, cada medida não representa a utilização instantânea e sim a média da utilização do recurso desde a última medida realizada, cinco minutos antes. Isso é util para lidar com eventuais picos instantâneos de utilização. Esses picos poderiam representar medidas distorcidas. A forma de coleta de dados de CPU favorece o cálculo dessa média, sem custo adicional algum ao processo de monitoramento.

3.3.2 Utilização *versus* Variação de Utilização

O comportamento das máquinas em relação ao uso de seus recursos pode ser descrito tanto por medidas de utilização propriamente dita quanto por variações

Tabela 3.2: Utilização de memória física em 4 períodos de uma hora.

Instante	*0:10*	*0:20*	*0:30*	*0:40*	*0:50*	*0:60*
I	90%	88%	90%	60%	60%	65%
II	82%	83%	80%	81%	78%	80%
III	50%	52%	53%	51%	52%	51%
IV	50%	49%	52%	25%	26%	24%

nas medidas de utilização.

Seja um objeto representado por um vetor $\bar{x} \in \mathbb{R}^m$ de m medidas $x_i, 1 \leq i \leq m$, que representam a utilização instantânea de um determinado recurso da máquina. Outra forma de representar esse objeto é feita através de um vetor $\bar{y} \in \mathbb{R}^{m-1}$ de $(m-1)$ variações de utilização. Dessa forma, $y_i = (x_{i+1} - x_i), 1 \leq i \leq m-1$.

As partições geradas pelas duas formas de se caracterizar as variáveis podem ser completamente diferentes uma da outra. Os ensaios realizados, como será visto adiante neste texto, levaram em consideração as duas medidas, que puderam ser comparadas quanto ao sucesso no reconhecimento de padrões dos objetos não classificados.

Para demonstrar que as partições podem ser completamente distintas, seja o exemplo ilustrado na Tabela 3.2. Para simplificação, sem perda de generalidade, foram construídos quatro objetos que representam a utilização percentual de memória física durante períodos de uma hora. Esses objetos estão representados graficamente na Figura 3.3.

Se cada objeto é representado por um vetor de medidas de utilização propriamente dita, é possível inferir que os objetos relativos aos períodos I e II sejam mais semelhantes entre si do que quando comparados com os relativos aos períodos III e IV. Os próprios objetos dos períodos III e IV são semelhantes entre si.

O primeiro conglomerado (períodos I e II) representa um padrão de comportamento de alta utilização de memória física. O segundo conglomerado (períodos III e IV) representa utilização moderada.

Se forem utilizadas as variações nas medidas, os objetos serão agrupados de forma completamente distinta. É possível observar que tanto o objeto do período I quanto o objeto do período IV apresentaram uma brusca variação de utilização na segunda metade do período de uma hora. Já os objetos dos períodos II e III representaram um comportamento mais uniforme quanto ao percentual de memória utilizado.

Com essa outra abordagem, a partição resultante é completamente diferente: objetos I e IV em um conglomerado e objetos II e III em outro conglomerado. Esse fenômeno foi investigado neste estudo.

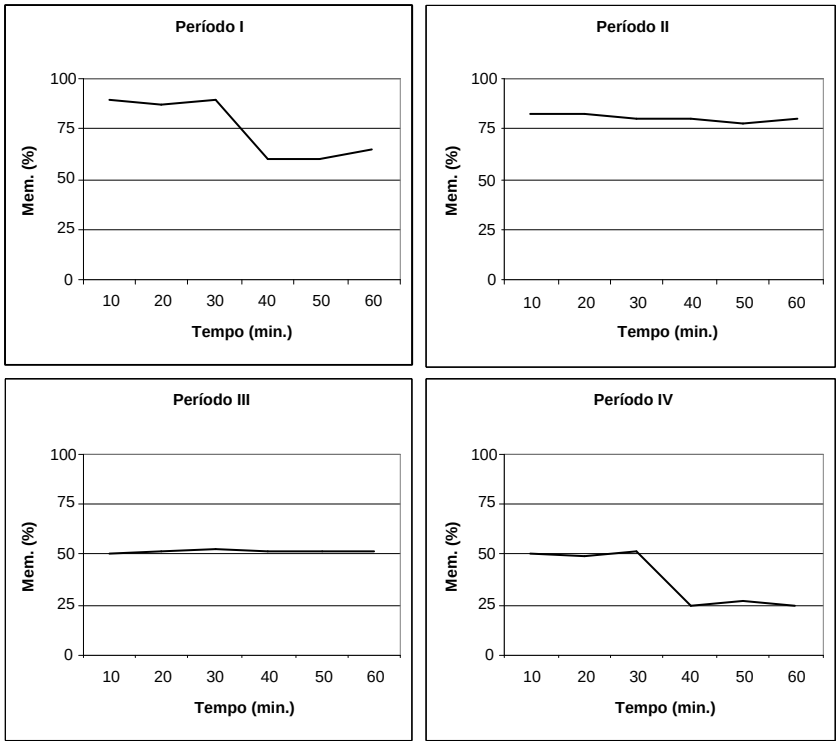

Figura 3.3: Utilização de memória física em 4 períodos distintos.

3.3.3 Dados Inexistentes

O processo de aprendizado deverá ser capaz de lidar com lacunas de informação referentes à ausência de medidas em um determinado espaço de tempo. No caso em estudo, essas lacunas podem ser causadas por problemas no armazenamento dos dados, falhas nos equipamentos ou algum problema no processo de monitoramento do uso dos recursos.

Jain e Dubes [11] enumeram formas para o tratamento de lacunas na informação:

1. Excluir todos os objetos que apresentarem lacunas de informação. Essa técnica leva a uma utilização pouco eficiente da massa de dados, a menos que a quantidade de objetos que apresentem dados inexistentes seja muito pequena.

2. Preencher a lacuna existente com a média das medidas de outros objetos, equivalente à informação inexistente. Assim, assume-se que para aquela medida, o objeto apresentou comportamento igual à média de outros ob-

jetos.

3. Modificar o método de cálculo das medidas de semelhança entre os objetos, de modo a considerar apenas as variáveis em que ambos os objetos em comparação apresentem valores disponíveis. Assim, no caso em que um objeto apresente lacuna de informação, essa lacuna não será levada em consideração na comparação com outros objetos.

4. De forma semelhante à anterior, o cálculo das medidas de semelhança é alterado. Na comparação de variáveis com lacunas, é utilizada uma média das diferenças entre todas as medidas disponíveis, nos outros objetos, para essa variável.

Pela natureza das informações coletadas no monitoramento de utilização de recursos, o tratamento utilizado neste trabalho foi simplesmente repetir a última medida disponível. Assume-se que o nível de utilização de um determinado recurso mantém-se constante desde a última medição realizada. Objetos que apresentarem um percentual de lacunas maior que um nível predeterminado, parametrizável, são descartados. Por exemplo, se o nível escolhido para o corte for 20%, objetos que apresentarem vetores com mais de 20% de medidas com lacunas são descartados do processo.

3.4 Padronização e Normalização das Variáveis

Em casos onde há grande diferença entre as magnitudes médias ou entre as variâncias das medidas das variáveis, muitos analistas são levados a crer que a padronização dos dados é mandatória. A maioria dos métodos de análise de conglomerados não assume, entretanto, a existência de dados padronizados.

A normalização ou a padronização, apesar de poderem ser aplicadas, nem sempre são necessárias. A existência de um conglomerado pode ter relação com o espaço de dados original e a padronização dos dados pode prejudicar o agrupamento dos dados.

Se a existência de conglomerados é assumida em um espaço de dados transformados, então a padronização poderá ser aplicada. Um novo questionamento diz respeito a como proceder à padronização. Milligan e Cooper [17] enumeram oito formas de fazer a medição das variáveis.

Sejam:

- x o valor medido de uma característica (utilização instantânea de um recurso)
- μ o valor médio, entre todos os objetos, da característica
- σ o desvio padrão das medidas, entre todos os objetos, da característica
- *max*, *min* e *rank* funções que retornam, respectivamente, o maior valor observado de uma característica, o menor valor medido de uma característica e a posição ordinal da medida x dentro da distribuição de medidas.

As medidas propostas são:

- $z_0 = x$ (sem padronização)
- $z_1 = \frac{x-\mu}{\sigma}$
- $z_2 = \frac{x}{\sigma}$
- $z_3 = \frac{x}{\max(x)}$
- $z_4 = \frac{x}{\max(x)-\min(x)}$
- $z_5 = \frac{x-\min(x)}{\max(x)-\min(x)}$
- $z_6 = \frac{x}{\sum x}$
- $z_7 = \text{rank}(x)$[1]

Cada uma dessas medidas pode ser aplicada em condições distintas, o que leva à necessidade de análise prévia e de realização de alguns testes para a determinação do método a ser utilizado. A princípio, o método utilizado na caracterização dos objetos do InteGrade é a própria medida, sem padronização ou normalização.

3.5 Medidas de Semelhança

A análise de conglomerados é baseada na construção de subconjuntos disjuntos (*clusters*) de uma massa de dados, onde os elementos de um mesmo subconjunto são mais semelhantes entre si do que quando comparados com elementos de outros subconjuntos. A determinação de como será medida a similaridade entre os elementos é, portanto, um passo fundamental no processo de aprendizado e corresponde à métrica na qual se acredita que os *clusters* existam.

Como os objetos que serão submetidos ao processo de partição dos dados em *clusters* são caracterizados por vetores de dados, a similaridade entre dois objetos corresponde a uma métrica que compare esses vetores de dados. Essa métrica dependerá dos tipos de dados que compõem esses vetores.

Muitas vezes, é mais simples medir a distinção entre os objetos, em vez da similaridade: a forma mais intuitiva de comparar dois vetores de dados é computar a distância entre eles. A distância entre esses vetores é de fato uma medida de dissimilaridade, conceito simetricamente oposto ao de similaridade.

Podem-se utilizar tanto medidas de similaridade como de dissimilaridade no processo de construção dos *clusters*: a decisão de qual abordagem deve ser utilizada depende de como os objetos são caracterizados e de qual é a melhor forma de compará-los. No contexto do projeto InteGrade, a métrica de comparação utilizada é a distância entre objetos.

[1] a função $\text{rank}(x)$ retorna a posição do valor x em uma lista ordenada de valores.

3.5.1 Distância entre Dois Vetores

Assumindo que os objetos são caracterizados por números reais, caso do processo de *clustering* no InteGrade, as formas mais difundidas de se calcular distâncias entre vetores em análise de conglomerados são métricas da família de distâncias de Minkowski, definida a seguir.

Sejam $\overline{x}$ e $\overline{y}$ dois vetores com m dimensões pertencentes ao conjunto de elementos $\mathbb{C}$, a serem submetidos aos algoritmos de *clustering*. Sejam ainda x_i e y_i $(1 \leq i \leq m)$ suas i-ésimas coordenadas, respectivamente. A métrica de Minkowski $d_p(\overline{x}, \overline{y})$, onde p é um parâmetro, é definida como:

$$d_p(\overline{x}, \overline{y}) = \sqrt[p]{\left(\sum_{i=1}^{m} (|x_i - y_i|)^p\right)} \tag{3.1}$$

Toda métrica, incluindo a de Minkowski, deve ter algumas propriedades básicas, importantes para sua utilização como medida de dissimilaridade entre elementos:

1. $d_p(\overline{x}, \overline{y}) \geq d_{\text{minimo}}, \forall \overline{x}, \overline{y} \in \mathbb{C}$
2. $d_p(\overline{x}, \overline{y}) = d_{\text{minimo}} \iff \overline{x} = \overline{y}$ (equivalência)
3. $d_p(\overline{x}, \overline{y}) = d_p(\overline{y}, \overline{x}), \forall \overline{x}, \overline{y} \in \mathbb{C}$ (simetria)
4. $d_p(\overline{x}, \overline{z}) \leq d_p(\overline{x}, \overline{y}) + d_p(\overline{y}, \overline{z}), \forall \overline{x}, \overline{y}, \overline{z} \in \mathbb{C}$ (desigualdade triangular)

No caso da métrica de Minkowski, $d_{\text{minimo}} = 0$.

O parâmetro p define a família de distâncias. Fazendo $p = 2$, tem-se a distância euclidiana:

$$d_{\text{Euclidiana}}(\overline{x}, \overline{y}) = d_2(\overline{x}, \overline{y}) = \sqrt{\left(\sum_{i=1}^{m} (x_i - y_i)^2\right)} \tag{3.2}$$

Fazendo $p = 1$, tem-se a métrica conhecida como distância de *Manhattan*:

$$d_{\text{Manhattan}}(\overline{x}, \overline{y}) = d_1(\overline{x}, \overline{y}) = \sum_{i=1}^{m} |x_i - y_i| \tag{3.3}$$

Finalmente, tomando-se $p = \infty$, a distância calculada representará o máximo da diferença entre os atributos que caracterizam os objetos:

$$d_\infty(\overline{x}, \overline{y}) = \max_{1 \leq i \leq m} |x_i - y_i| \tag{3.4}$$

É fácil observar que $d_\infty(\overline{x}, \overline{y}) \leq d_2(\overline{x}, \overline{y}) \leq d_1(\overline{x}, \overline{y})$. Cada uma dessas formas de se calcular a distância entre elementos pode ser aplicada, dependendo do contexto em que está sendo utilizada. A definição da distância a ser utilizada

muitas vezes depende da ajuda de um especialista do domínio do conhecimento em que está ocorrendo a análise de conglomerados.

Podem-se computar as métricas apresentadas até aqui de forma alternativa, introduzindo-se para isso um vetor $w_i (1 \leq i \leq m)$ com os pesos de cada um dos atributos no cálculo das distâncias entre os elementos. A generalização das distâncias de Minkowski, escrita em sua forma ponderada, é:

$$d_p(\overline{x}, \overline{y}) = \sqrt[p]{\left(\sum_{i=1}^{m} w_i (|x_i - y_i|)^p \right)} \tag{3.5}$$

Dessa forma, é possível definir uma escala de contribuição para cada variável. Usar um vetor w_i com os pesos dos atributos pode ser estratégia complementar, ou mesmo alternativa, à normalização de variáveis. Esse peso pode ser atribuído de formas distintas:

- subjetivamente — a escolha dos pesos é feita por um especialista no momento de escolha das variáveis que farão parte da descrição de um objeto;
- medidas estatísticas — por exemplo, pode-se utilizar uma escala de variância, onde o peso é inversamente proporcional à variância das medidas da característica.

A ponderação entre as variáveis através de medidas estatísticas do próprio conjunto de dados $\mathbb{C}$ é proposta na distância de Mahalanobis:

$$d_{\text{Mahalanobis}}(\overline{x}, \overline{y}) = \sqrt{(\overline{x} - \overline{y})\, \bar{C}\, (\overline{x} - \overline{y})^T} \tag{3.6}$$

Onde:

- $\overline{z} = \overline{x} - \overline{y}$ é um vetor de m elementos, com $z_i = x_i - y_i, 1 \leq i \leq m$
- $(\overline{x} - \overline{y})^T$ é o vetor $\overline{x} - \overline{y}$ transposto
- $\bar{C}$ é matriz de co-variâncias das variáveis x_i de $\overline{x} \in \mathbb{C}$. A co-variância entre as variáveis i e j é definida como $c_{\text{ij}} = \sum_{x \in \mathbb{C}} \frac{(x_i - \mu_i)(x_j - \mu_j)}{|\mathbb{C}|}$, onde $\mu_i = \sum_{x \in \mathbb{C}} \frac{x_i}{|\mathbb{C}|}$ e $|\mathbb{C}|$ reprenta o número de elementos de $\mathbb{C}$.

O uso da matriz de co-variâncias $\bar{C}$ leva em consideração o efeito de uma eventual presença de variáveis altamente correlacionadas: variáveis altamente correlacionadas têm sua influência diminuída para compensar o efeito da existência de informação redundante em um mesmo vetor de dados.

As quatro propriedades das distâncias de Minkowski citadas anteriormente também são observadas na distância de Mahalanobis.

Medidas de Similaridade

Além de medidas de dissimilaridade entre dois vetores $\overline{x}$ e $\overline{y}$, também são encontradas na literatura diversas medidas baseadas na similaridade entre elementos. Nesse tipo de abordagem, quanto maior o resultado da métrica, maior a semelhança entre os elementos.

Uma das formas mais comuns de se obter a similaridade entre objetos é utilizando-se o produto escalar normalizado $s(\overline{x}, \overline{y})$:

$$s(\overline{x}, \overline{y}) = \frac{\sum_{i=1}^{m} x_i y_i}{|\overline{x}|\,|\overline{y}|} \tag{3.7}$$

Onde $|\overline{x}|$ é o módulo de um vetor $\overline{x} \in \mathbb{C}$.

Essa medida é aplicável quando a utilização de variáveis normalizadas descreve corretamente o comportamento dos objetos pertencentes à massa de dados. Isso porque, matematicamente, o produto escalar normalizado $s(\overline{x}, \overline{y})$ significa o co-seno entre o ângulo formado pelos vetores $\overline{x}$ e $\overline{y}$. Por exemplo, se dois vetores $\overline{x}$ e $\overline{y}$ são tais que para todo $\exists k \in \mathbb{R}, x_i = k y_i, 1 \leq i \leq m \Rightarrow s(\overline{x}, \overline{y}) = 1$, que é a medida máxima possível.

Outra medida de similaridade comum, que ainda pode ser aplicada mesmo quando os vetores são descritos por variáveis de valores discretos, é a medida conhecida como distância de Tanimoto:

$$s_{\text{Tanimoto}}(\overline{x}, \overline{y}) = \frac{\sum_{i=1}^{m} x_i y_i}{|\overline{x}|^2 + |\overline{y}|^2 - \sum_{i=1}^{m} x_i y_i} \tag{3.8}$$

Após manipulação algébrica, $s_{\text{Tanimoto}}(\overline{x}, \overline{y})$ pode ser reescrita como:

$$s_{\text{Tanimoto}}(\overline{x}, \overline{y}) = \frac{1}{1 + \frac{\sum_{i=1}^{m} (x_i - y_i)^2}{\sum_{i=1}^{m} x_i y_i}} \tag{3.9}$$

Essa medida é inversamente proporcional ao quadrado da distância euclidiana entre os vetores divido pelo produto escalar desses mesmos vetores.

3.5.2 Distância entre Conjuntos de Vetores

Além de medidas de distância entre dois vetores de dados, muitos dos algoritmos para análise de conglomerados também fazem uso do conceito de similaridade (ou dissimilaridade) entre um vetor e um conjunto de vetores ou mesmo entre dois conjuntos de vetores.

Há duas diretrizes básicas que podem ser seguidas. Na primeira delas, é definido um vetor representativo do conjunto de dados e retorna-se ao problema da definição de distância entre dois vetores. Na segunda, todos os elementos que fazem parte do conjunto são levados em consideração no cômputo da distância.

Objeto Representativo

Há algumas maneiras de se escolher um vetor para representar os objetos que fazem parte de um *cluster*. A idéia é que esse vetor traduza um comportamento médio, ou mesmo esperado, das variáveis que caracterizam os elementos de um conjunto. Esse vetor pode ser um elemento do conjunto a ser representado ou mesmo um objeto criado artificialmente pelo processo de *clustering*.

São alguns exemplos de objetos representativos:

1. Ponto médio — é um objeto que não necessariamente pertence ao conjunto de elementos que compõem o *cluster*. Suas variáveis são definidas como: $z_i = \frac{1}{|\mathbb{C}|} \sum_{\overline{x} \in \mathbb{C}} x_i, 1 \leq i \leq m$. É a forma mais intuitiva de se calcular um objeto representativo e uma das formas mais difundidas na literatura. Como, entretanto, o vetor gerado pode não pertencer à massa de dados, pode-se obter um elemento representativo com medidas que não façam sentido no domínio válido para os elementos do conjunto. Isso é especialmente crítico quando as variáveis que descrevem os objetos são discretas.

2. Centro médio — é um objeto, pertencente ao conjunto de elementos $\mathbb{C}$, em que a somatória das distâncias aos demais elementos do conjunto é mínima. Assim, o centro médio $\overline{z} \in \mathbb{C}$ é tal que $\sum_{\overline{x} \in \mathbb{C}} d(\overline{z}, \overline{x}) \leq \sum_{\overline{x} \in \mathbb{C}} d(\overline{y}, \overline{x}), \forall y \in \mathbb{C}$, onde $d(\overline{z}, \overline{x})$ é a medida de distância entre dois vetores $\overline{x}$ e $\overline{z}$.

3. Centro mediano — é um objeto, pertencente ao conjunto de elementos $\mathbb{C}$, em que a mediana das distâncias aos outros elementos do conjunto é mínima. Assim, o centro mediano $\overline{z} \in \mathbb{C}$ é tal que $\text{mediana}(d(\overline{z}, \overline{x}) | x \in \mathbb{C}) \leq \text{mediana}(d(\overline{y}, \overline{x}) | x \in \mathbb{C}), \forall y \in \mathbb{C}$, onde $d(\overline{z}, \overline{x})$ é a medida de distância entre dois vetores $\overline{x}$ e $\overline{z}$ e a função mediana(A) retorna a mediana do conjunto A.

Com o uso de objetos representativos, a distância entre um vetor e um *cluster* é definida como a distância entre esse vetor e o vetor representativo do *cluster*. De forma semelhante, a distância entre dois *clusters* é definida como a distância entre os objetos representativos desses *clusters*.

O objeto representativo utilizado nesse trabalho foi o ponto médio. A escolha foi baseada no fato de ser a melhor forma de representar o comportamento prototípico. Além disso, não há efeito negativo causado pelo fato de que o ponto médio pode ser um objeto não pertencente ao respectivo conglomerado. A arquitetura proposta para o software desenvolvido, descrita no Apêndice 5.2, permite, entretanto, que sejam facilmente implementados e utilizados outros modelos.

Distância entre Elementos de Conjuntos

Apesar de o uso de um elemento representativo ser o instrumento mais intuitivo para o cômputo da distância entre conjuntos de vetores (ou mesmo entre

um vetor e um conjunto de vetores), é possível a utilização dos próprios elementos dos conjuntos para o cálculo dessa medida. Esse tipo de abordagem é bastante utilizado na prática e normalmente todos os elementos dos conjuntos são considerados no cálculo da distância.

Sejam:

- $\mathbb{C}$ o conjunto de todos os elementos que foram submetidos ao processo de partição de dados.
- $d(\overline{x}, \overline{y})$ a distância entre dois vetores $\overline{x}, \overline{y} \in \mathbb{C}$, calculada de acordo com um determinado modo.
- Γ_i e Γ_j dois conjuntos, com γ_i e γ_j elementos respectivamente, entre os quais se deseja calcular a distância. $\Gamma_i \subset \mathbb{C}$ e $\Gamma_j \subset \mathbb{C}$.

Alguns exemplos de cálculo de distância entre conjuntos $d_{\text{Cluster}}(\Gamma_i, \Gamma_j)$, levando-se em consideração todos os elementos desses conjuntos são:

1. Ligação simples (do inglês *single linkage*) — a distância entre os dois conjuntos é estabelecida como sendo a menor das distâncias entre elementos de conjuntos distintos. Assim, $d_{\text{Cluster}}(\Gamma_i, \Gamma_j) = \min\{d(\overline{x}, \overline{y}) | \overline{x} \in \Gamma_i; \overline{y} \in \Gamma_j\}$.
2. Ligação completa (do inglês *complete linkage*) — a distância entre os dois conjuntos é estabelecida como sendo a maior das distâncias entre elementos de conjuntos distintos. Assim, $d_{\text{Cluster}}(\Gamma_i, \Gamma_j) = \max\{d(\overline{x}, \overline{y}) | \overline{x} \in \Gamma_i; \overline{y} \in \Gamma_j\}$.
3. Dissimilaridade média — a distância entre os dois conjuntos é a média aritmética de todas distâncias entre elementos de conjuntos distintos. Assim, $d_{\text{Cluster}}(\Gamma_i, \Gamma_j) = \frac{1}{\gamma_i \gamma_j} \sum_{\overline{x} \in \Gamma_i} \sum_{\overline{y} \in \Gamma_j} d(\overline{x}, \overline{y})$.

Apesar de em alguns casos os resultados serem bem similares, a utilização de diferentes métodos para cálculo da distância entre *clusters* pode levar a partições bem diferentes umas das outras. A escolha do método mais adequado tem importância semelhante à escolha da métrica de distância entre dois elementos individuais.

A melhor medida a ser utilizada só pode ser definida basicamente de duas formas: tentativa e erro, submentendo-se a partição gerada a algum modelo de validação; ou através da ajuda de algum especialista no domínio do conhecimento. Como nem sempre é possível prever a existência de *clusters* naturais na massa de dados, acaba-se retornando ao caso da tentativa e erro na maioria das vezes.

Como será visto adiante, a ligação simples, por exemplo, é mais apropriada quando há presença de cadeias alongadas de vetores pelo espaço m-dimensional e quando os *clusters* são muito bem separados no espaço. Ruídos existentes entre os *clusters* podem acabar levando à união destes *clusters* na partição resultante, quando esse tipo de medida é utilizado.

Já a ligação completa é apropriada quando os *clusters* naturais são densos e com quantidade semelhante de objetos. Como também será visto, *clusters* com quantidade muito pequena de elementos poderiam acabar unificados na partição final dos dados.

Uma outra forma de se calcular a distância entre conjuntos de dados leva em consideração a distância entre os elementos representativos e também informações sobre o número de elementos no conjunto. Essa distância foi proposta por Ward [16] e é conhecida com a distância da mínima variância. Usando a mesma notação:

$$d_{\text{Cluster}}(\Gamma_i, \Gamma_j) = \sqrt{\frac{\gamma_i \gamma_j}{\gamma_i + \gamma_j}} d(\overline{z}_i, \overline{z}_j) \tag{3.10}$$

Onde $\overline{z}_i$ e $\overline{z}_j$ são, respectivamente, os elementos representativos de Γ_i e Γ_j. Ward propôs essa distância com o objetivo de identificar a forma de unir *clusters* de maneira que uma medida de erro de atribuição de um elemento a um *cluster* fosse minimizada. Por esse motivo, considerando os detalhes da medida de erro, essa distância é conhecida como de mínima variância.

3.6 Métodos de Agrupamento

O método de agrupamento é o núcleo da análise de conglomerados. Há quatro aspectos [10] que devem ser observados na escolha de um método:

- o método deve ser projetado de forma a recuperar os conglomerados que se espera estarem presentes na massa de dados;
- o método deve ser efetivo ao recuperar as estruturas para as quais foi projetado;
- o método deve ser capaz de lidar com erros presentes na massa de dados;
- deve ser possível o projeto de software que se adéqüe ao método.

Há uma grande variedade de algoritmos de *clustering* pesquisados e testados nos mais diversos tipos de aplicação. Esses algoritmos podem ser divididos em três categorias básicas, a saber:

- algoritmos seqüenciais: os elementos são apresentados ao algoritmo de forma seqüencial, um por vez. São algoritmos bem simples e o resultado final normalmente depende da ordem em que os objetos foram apresentados;
- algoritmos hierárquicos: são baseados na divisão ou na união de *clusters* a cada iteração do algoritmo. Podem utilizar matrizes ou grafos para o controle dos *clusters* que devem ser unificados ou divididos; e
- algoritmos de otimização: são estabelecidas partições iniciais dos dados, com posteriores rearranjos baseados na otimização de alguma medida de erro.

3.6.1 Algoritmos Seqüenciais

Os algoritmos seqüenciais são normalmente bem simples e computacionalmente pouco complexos. Há diversas variações desse tipo de algoritmo, que basicamente lida com a apresentação em série, uma ou mais vezes, de cada um dos objetos que são submetidos ao processo de agrupamento.

Definido um limiar de semelhança Θ para separação dos *clusters*, a maneira básica de implementar um algoritmo seqüencial consiste em:

1. Ordenar os objetos da massa de dados para apresentação ao algoritmo.
2. Criar um *cluster* inicial, cujo elemento unitário é o primeiro objeto da massa de dados.
3. Para cada objeto restante:
   (a) Apresentá-lo ao conjunto de *clusters* existente.
   (b) Se a distância ao *cluster* mais próximo desse objeto for menor que o limiar Θ, adicionar o objeto a este *cluster* mais próximo.
   (c) Caso contrário, criar um novo *cluster* contendo apenas esse objeto.

Pode-se definir, ainda, um número máximo de conglomerados a serem constituídos. Quando esse número for atingido, os objetos são adicionados sempre ao conglomerado mais próximo de cada um deles, mesmo que a distância encontrada seja maior que o limiar Θ definido.

É imediata a conclusão de que a partição resultante depende da ordem em que os objetos são apresentados ao algoritmo. Além disso, a determinação do limiar Θ que deve ser utilizado também é bastante relevante para a definição da partição que será gerada. A escolha do limiar Θ muitas vezes depende inclusive de uma análise manual da composição da massa de dados.

Apresentação de Objetos em Duas Fases

Outro aspecto a ser destacado é que objetos são adicionados aos *clusters* antes mesmo de a partição final (conjunto de todos os *clusters* existentes) estar completamente formada. Uma forma de contornar esse problema é implementar o algoritmo com a apresentação dos objetos em duas fases. Na primeira fase, apenas objetos que formem novos *clusters* são aproveitados. Numa segunda fase, os objetos que ainda não foram atribuídos a nenhum *cluster* são processados. O algoritmo em duas fases é assim construído:

1. Ordenar os objetos da massa de dados para apresentação ao algoritmo.
2. Criar um *cluster* inicial, cujo elemento unitário é o primeiro objeto da massa de dados.
3. Fase 1 — Para cada objeto restante:
   (a) Apresentá-lo ao conjunto de *clusters* existente.

(b) Se a distância ao *cluster* mais próximo desse objeto for maior que o limiar Θ, criar um novo *cluster* contendo apenas esse objeto.

(c) Caso contrário, armazenar o objeto para utilização na Fase 2.

4. Fase 2 — Para cada objeto que ainda não foi adicionado a nenhum *cluster*:

   (a) Apresentá-lo novamente ao conjunto de *clusters* existente.

   (b) Adicionar esse objeto ao *cluster* mais próximo.

Ainda há grande dependência da ordem em que os objetos são apresentados, bem como da escolha do valor para o limiar Θ.

Utilização de Dois Limiares de Semelhança

A escolha do limiar Θ é ao mesmo tempo decisiva e difícil. A escolha de valores muito grandes ou muito pequenos para esse limiar pode levar à formação de conglomerados que não façam sentido algum no contexto dos objetos considerados.

Uma maneira de lidar com esse problema é definir dois limiares distintos, entre os quais se forma uma região em que não é possível afirmar se um objeto pertence ou não a um *cluster*.

Sejam os dois limiares Θ_1 e Θ_2, tais que $\Theta_1 < \Theta_2$. Se a distância entre o objeto e o *cluster* mais próximo for menor que Θ_1, esse objeto é adicionado ao *cluster*. Se essa distância for maior que Θ_2, certamente trata-se de um outro *cluster*, que então é criado e composto inicialmente apenas desse objeto.

Se a distância for intermediária entre os dois limiares, nada é feito com o objeto. Cada objeto pode ser apresentado ao algoritmo em diversas iterações. A cada iteração, é criado um novo *cluster*, ao qual é adicionado o primeiro objeto da lista de elementos ainda não adicionados a *cluster* algum. Dessa forma, é garantido que todos os objetos tenham sido atribuídos a algum *cluster* na partição final dos dados.

Refinamentos

Como os algoritmos seqüenciais são sensíveis à ordem em que são apresentados os elementos a serem classificados, muitas vezes são feitos ajustes na distribuição final dos elementos. Esses ajustes compreendem:

- Fusão de conglomerados — ao final do processo de *clustering*, se dois conglomerados situarem-se a uma distância menor que o limiar Θ, esses conglomerados são fundidos em um só.

- Realocação de objetos — se a distância de um objeto a um outro *cluster* é menor que a distância desse objeto ao próprio *cluster* ao qual pertence, esse objeto é realocado ao *cluster* mais próximo.

Esses procedimentos de refinar a distribuição final podem ser feitos separadamente ou de forma combinada. Normalmente, primeiro se faz a fusão de conglomerados para então ser realizada a realocação de objetos.

3.6.2 Algoritmos Hierárquicos

Considere-se o seguinte:

- Seja uma partição P_k da massa de dados, composta de k conglomerados, subconjuntos dessa massa de dados. Sejam c_i^k esses subconjuntos. Assim, $P_k = \{c_i^k, 1 \leq i \leq k\}$.
- Seja P_l uma outra partição, contendo l conglomerados, com $l > k$. $P_l = \{c_j^l, 1 \leq j \leq l\}$.
- Se $\exists i, 1 \leq i \leq k$ tal que $c_j^l \subset c_i^k, \forall j, 1 \leq j \leq l$ então diz-se que a partição P_l é aninhada à partição P_k.

Os algoritmos hierárquicos baseiam-se na construção iterativa de partições aninhadas. São realizados até $(n-1)$ passos, onde n é o número de objetos existentes na massa de dados. As partições de passos adjacentes são tais que uma é aninhada à outra e a partição aninhada possui exatamente um conglomerado (subconjunto da massa de dados) a menos que a outra partição.

Existem duas classes de algoritmos hierárquicos: os aglomerativos e os divisivos. Nos algoritmos aglomerativos, a partição existente após um dado passo do algoritmo é aninhada à partição originada com o passo seguinte. Nos algoritmos divisivos ocorre o inverso: a partição resultante após um determinado passo é aninhada à partição originada com o passo anterior.

Assim, nos algoritmos hierárquicos aglomerativos, o processo de *clustering* é iniciado com uma partição contendo n conglomerados, subconjuntos unitários da massa de dados. Após o primeiro passo, a partição é composta de $n-1$ conglomerados, onde dois dos conglomerados da partição inicial foram unidos em um único conglomerado de dois elementos.

Esse processo se repete, com a união de dois conglomerados de uma partição a cada passo do algoritmo. Após o número máximo de n passos, tem-se uma partição com um único conglomerado de objetos, representando todos os objetos da massa de dados.

Se a cada passo, os dois conglomerados unificados são os mais semelhantes, de acordo com algum critério estabelecido, pode-se entender que os conglomerados da partição representam *clusters* de dados. A cada passo, a massa de dados é dividida em *clusters* contendo uma quantidade maior ou igual de elementos.

Nos algoritmos divisivos, o processo é inverso: inicialmente há uma partição contendo um único conglomerado. Após o primeiro passo, esse conglomerado é dividido em dois. Nos passos seguintes, um dos conglomerados é dividido em outros dois.

De forma semelhante, para que se obtenham *clusters* de dados, a divisão de um conglomerado em dois, nos algoritmos divisivos, é feita de modo que os dois conglomerados gerados sejam o mais semelhante possível um do outro.

Os algoritmos aglomerativos tornaram-se os mais utilizados em aplicações práticas: parte desse fenômeno se deve ao fato de que são computacionalmente menos complexos que os algoritmos divisivos.

Algoritmos Hierárquicos Aglomerativos

Seja $\bar{x}_i$, $1 \leq i \leq n$, cada um dos n objetos da massa de dados. O algoritmo hierárquico aglomerativo consiste em:

1. Criar uma partição inicial contendo n *clusters* c_i^n, $1 \leq i \leq n$, com c_i^n=$\{\bar{x}_i\}$.
2. A cada passo, considerar a partição P_k, com k *clusters* do passo anterior:
   (a) Comparar a distância dos k *clusters* entre si.
   (b) Unir os dois *clusters* c_i^k e c_j^k, $1 \leq i, j \leq k$ tais que a distância entre eles seja a menor das distâncias entre os diversos *clusters*. Têm-se assim $(k-1)$ clusters.

Uma das maneiras de se implementar esse algoritmo é utilizando-se de uma matriz denominada matriz de dissimilaridade (sem perda de generalidade, foi utilizada a distância como métrica de semelhança). Inicialmente, a matriz tem dimensões $n \times n$. Cada elemento a_{ij} da matriz contém o valor da distância entre os elementos $\bar{x}_i$ e $\bar{x}_j$, com $1 \leq i, j \leq k$. A matriz é, por construção, simétrica e tem a diagonal principal zerada.

Localiza-se a célula da matriz de menor valor. Os índices da célula de menor valor indicam quais são os elementos mais semelhantes entre si. Esses elementos, juntos, formarão um novo *cluster*. A matriz é redimensionada, tornando-se uma matriz $(n-1) \times (n-1)$, contendo as distâncias originais, exceto as distâncias envolvendo os elementos unidos em um único *cluster*. É calculada a distância entre esses *clusters* e os demais elementos remanescentes. Esses passos são repetidos até que se tenha uma matriz com as mesmas dimensões do número de *clusters* desejado (vide Seção 3.7).

Para fins de ilustração, seja a matriz de distâncias M, representando o estado do algortimo hierárquico aglomerativo num passo qualquer, contendo 5 *clusters*.

$$M = \begin{pmatrix} 0 & 0.8 & 1.3 & 5.6 & 3.9 \\ 0.8 & 0 & 1.9 & 3 & 1.4 \\ 1.3 & 1.9 & 0 & 0.6 & 1.7 \\ 5.6 & 3 & 0.6 & 0 & 6.3 \\ 3.9 & 1.4 & 1.7 & 6.3 & 0 \end{pmatrix}$$

Através da comparação dos valores da matriz M, conclui-se que os *clusters* 3 e 4 são os mais semelhantes entre si. Supondo que esteja sendo utilizada a ligação simples (vide Seção 3.5.2, é possível construir a matriz de distâncias M', do passo seguinte:

$$M' = \begin{pmatrix} 0 & 0.8 & 1.3 & 3.9 \\ 0.8 & 0 & 1.9 & 1.4 \\ 1.3 & 1.9 & 0 & 1.7 \\ 3.9 & 1.4 & 1.7 & 0 \end{pmatrix}$$

Ligação Simples e Ligação Completa

Como pode ser observado, quando o método da ligação simples é utilizado, é possível construir a matriz de distâncias de um passo apenas utilizando a matriz do passo anterior. Isso também é possível quando se utiliza a ligação completa. Isso ocorre porque só interessa a menor (ou maior) distância entre os objetos dos conglomerados e essa informação fica registrada na matriz do passo anterior.

Os métodos de ligação simples e ligação completa para se calcular a distância entre conglomerados de objetos são bastante utlilizados em algoritmos hierárquicos.

Uma conseqüência imediata da ligação simples é que, quando um objeto é adicionado a um conglomerado, a distância entre esse conglomerado e os demais elementos diminuirá ou permanecerá a mesma. Isso implica que grandes conglomerados tendem a crescer e a ser unificados, enquanto objetos isolados assim permanecerão até as últimas iterações do procedimento.

Um fenômeno que contribuiu bastante para a popularização desse método de ligação simples é o comportamento do algoritmo para casos em que haja distâncias muito similares na massa de dados. Se, num dado momento, ocorrem distâncias muito próximas umas das outras, será natural que os objetos sejam unificados em seqüência, o que realmente é esperado num agrupamento. Mas, se qualquer outro método for usado, isso não necessariamente irá ocorrer.

No caso da ligação completa, ocorre o inverso: sempre que um objeto é adicionado a um conglomerado, sua distância aos demais objetos será maior ou igual à original. Isso quer dizer que quanto mais um conglomerado crescer, menor a tendência de ele se unir aos demais conglomerados. O comportamento observado é que os objetos se juntem primeiro em pequenos conglomerados e só então os conglomerados são concatenados. Isso é particularmente útil quando se deseja a simples divisão de dados em conglomerados, em vez da busca por clusters naturais.

Alguns efeitos colaterais dessas abordagens:

- objetos intermediários entre conglomerados naturais podem fazê-los se unir em conglomerados antes do momento devido, principalmente no caso da ligação simples; e
- se os dados constituírem conglomerados de tamanhos muito distintos, a ligação composta pode unir os conglomerados pequenos antes de os grandes estarem completos.

Dendrograma

Os passos de unificação de *clusters* podem ser representados graficamente em uma estrutura denominada dendrograma.

O dendrograma é um diagrama em forma de árvore, bastante familiar aos taxonomistas. Essa estrutura indica quais os *clusters* unificados em cada passo e qual a distância em que essa unificação se deu. A forma geral de um dendrograma é apresentada com uma raiz ao topo e uma escala de distâncias indicando

em que nível dois conglomerados tornam-se um só. A Figura 3.4 exemplifica o agrupamento sucessivo dos objetos A, B, C, D, E e F.

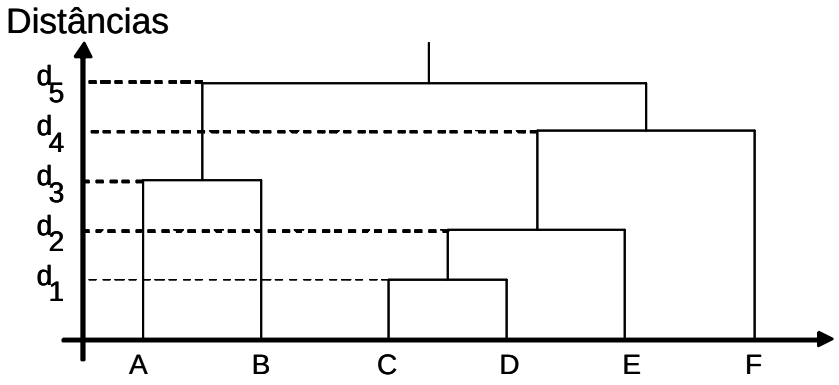

Figura 3.4: Dendrograma representando o processo de *clustering* de alguns objetos.

As distâncias d_1, d_2, d_3, d_4 e d_5 representam os níveis de distância em que os agrupamentos foram formados. Essa estrutura também pode ser utilizada para indicar qual a partição resultante do processo de *clustering*, permitindo que seja parametrizado o nível de distância entre os *clusters* ou o número de *clusters*. Dependendo do nível de agrupamento e da quantidade de conglomerados desejada, pode-se tomar um determinado valor para se estabelecer qual partição representa o resultado do processamento dos conglomerados. Por exemplo, se tomarmos como base um valor entre d_3 e d_4 na figura acima, haverá a presença de três conglomerados: {A, B}, {C, D, E} e {F}.

3.6.3 Algoritmos de Otimização

Outra família de algoritmos bastante pesquisada se baseia na otimização de alguma métrica que avalie a qualidade da partição gerada. A massa de dados normalmente é distribuída em conglomerados formando uma partição inicial. Após essa fase, são feitos rearranjos de forma a minimizar alguma medida de erro de alocação de objetos.

A motivação para esse tipo de algoritmo está no fato de que há um grande número de combinações possíveis para a realização da tarefa de dividir um conjunto de n objetos em k conglomerados. Cada combinação possível resulta numa determinada partição de dados distinta. Como visto anteriormente, as diversas técnicas de clustering objetivam fazer esse agrupamento de forma que objetos de um mesmo conglomerado sejam semelhantes entre si e que objetos de conglomerados distintos sejam menos semelhantes uns com os outros.

Para mapear esse objetivo, dada uma partição de dados qualquer, pode-se definir uma medida de erro entre a distribuição dos n objetos e a forma como eles foram particionados. Esse erro, de alguma forma, deve representar a semelhança entre objetos de um mesmo *cluster*. Assim, um erro pequeno indica que objetos de um mesmo *cluster* são semelhantes.

Seja a partição $P_i(n,k)$ de n objetos em k conglomerados. O erro associado a essa partição será expresso por $\varepsilon[P_i(n,c)]$. A idéia intuitiva é fazer com que se tome a partição P_i que possua o menor valor de erro possível. Essa partição será considerada ótima e, portanto, representa o agrupamento de objetos desejado.

Comparar o valor da medida de erro ε para cada uma das partições possíveis é impraticável, mesmo com a utilização de alto poder computacional. Um dos algoritmos de otimização mais difundidos é o algoritmo de k-centros. Esse algoritmo consiste em uma técnica de aproximação do procedimento de se comparar as medidas de erro ε possíveis. Isso é feito através da definição de uma vizinhança de partições para cada partição existente.

Tomando-se uma partição inicial, é feita uma busca, na vizinhança de partições, por aquela cuja medida de erro ε seja mínima. Essa partição é escolhida para uma nova iteração na busca da partição de erro mínimo. Segue-se esse procedimento até que o erro de cada uma das partições da vizinhança seja maior que o erro da partição atual. Assim, é estabelecido um critério de parada e identificada uma partição ótima, mesmo que a otimização eventualmente seja local.

K-Centros

Sejam n objetos, caracterizados por um vetor de m dimensões. Um objeto é representado por um vetor $\bar{x}_i \in \mathbb{R}^m$. Assim, o valor x_{ij} refere-se à característica j medida no objeto i, com $1 \leq i \leq n$ e $1 \leq j \leq m$.

Uma partição $P(n,k)$ da massa de dados equivale a dividir todos os n elementos em k subconjuntos disjuntos. Assim, a partição é tal que cada um de seus n objetos seja alocado em um único conglomerado entre os k possíveis.

Seja $\bar{b}_l \in \mathbb{R}^m$ o objeto representativo do *cluster* $c_l, 1 \leq l \leq k$. Assim, a variável b_{lj} é a medida representativa da característica j, para o *cluster* c_l. Seja ainda t_l o número de elementos do *cluster* c_l. Seja $d(\bar{x}_i, c_l)$ a medida de distância entre o objeto representado pelo vetor $\bar{x}_i$ e objeto representativo do cluster c_l.

Uma possível medida de erro para essa partição é a soma das distâncias entre os elementos e os seus respectivos *clusters* ao quadrado:

$$\varepsilon[P(n,k)] = \sum_{i=1}^{n} \left(d(\bar{x}_i, c_l(i))\right)^2 \tag{3.11}$$

$c_l(i)$ é o *cluster* ao qual pertence o objeto i.

O algoritmo consiste nos seguintes passos:

1. É assumida uma configuração inicial $P_0(n,k)$. Essa configuração inicial pode ser definida tanto aleatoriamente como através de algum critério

simples. Esse critério pode ser, por exemplo, a divisão dos objetos em conglomerados de tamanhos semelhantes formados pela ordenação desses objetos de acordo com os valores de uma determinada característica. São calculados os objetos representativos de cada *cluster*. Calcula-se, então, o erro $\varepsilon[P_0(n,k)]$ da partição $P_0(n,k)$.

2. Para o primeiro objeto da massa de dados ($i = 1$), é calculada a diferença no erro causada pelo remanejo desse primeiro objeto de seu conglomerado original $c_l(i), i = 1$, para cada um dos $k-1$ conglomerados restantes.

   A diferença no erro, causada pela transferência do objeto i do conglomerado $c_l(i)$ para $c_l, 1 \leq l \leq k$, é expressa por: $\Delta\varepsilon = \frac{t_l d(\bar{x}_i, c_l)^2}{t_l+1} - \frac{t_l(i) d(\bar{x}_i, c_l(i))^2}{t_l(i)-1}$. Se o valor mínimo de $\Delta\varepsilon$, considerando-se todas as combinações $c_l \neq c_l(i)$, for negativo, então o objeto i deve ser movido de $c_l(i)$ para o conglomerado c_l que faz $\Delta\varepsilon$ ser mínimo.

3. Os objetos representativos dos dois conglomerados são recalculados e o valor do erro ε é corrigido para a nova partição $P(n,k)$.

4. O passo 2 é repetido para todos os demais objetos, $i \neq 1$.

5. Se não tiver havido movimento algum desde a última iteração por todos os objetos, o algoritmo deve parar, tendo sido encontrado um mínimo local. Do contrário, uma nova iteração entre todos os elementos é iniciada, voltando-se ao passo 2.

3.7 Número de *Clusters*

Muitos métodos de agrupamento não levam em consideração o problema de determinação do número de conglomerados existentes na massa de dados. Ao contrário, nesses métodos um especialista deverá indicar *a priori* o número de conglomerados desejados.

Por outro lado, há diversos trabalhos de pesquisa disponíveis na literatura [18] [19] [20] [21] que investigam o tema de determinação do número de *clusters* naturais existentes em um conjunto de dados.

Muitas dessas pesquisas lidam com problemas de otimização. Dada uma massa de dados, diversas iterações do algoritmo de agrupamento são realizadas na tentativa de minimizar alguma medida de erro através da variação dos números de *clusters* utilizados para execução do algoritmo.

Há, entretanto, uma técnica simples, baseada em algoritmos seqüenciais, para a determinação do número ótimo de conglomerados a serem considerados. Para esse tipo de algoritmo, um dos parâmetros considerados é o limiar de semelhança entre os conglomerados. O número de conglomerados obtidos no processo de *clustering* é função do limiar de semelhança utilizado. A técnica consiste em:

1. Simular diversos valores para o limiar de semelhança e processar o algoritmo seqüencial uma vez para cada valor de limiar simulado.

2. A cada simulação, é determinado o número de conglomerados formados.

3. O número de conglomerados que mais for repetido, ao londo das diversas simulações é escolhido como o número ótimo de *clusters*.

Esta técnica foi explorada no contexto deste trabalho.

3.8 Reconhecimento do Modo de Operação

No processo de reconhecimento de padrões, há dois elementos fundamentais para a realização da predição de ociosidade proposta neste trabalho. O primeiro deles é o objeto não classificado que representa a utilização recente de um recurso, no momento da requisição de uso de uma determinada máquina. O segundo é o conjunto de comportamentos prototípicos dessa máquina, definidos pelo objetos representativos dos conglomerados da partição de dados gerada pelo *clustering*.

A ferramenta de comparação entre os objetos não classificados e os comportamentos prototípicos existentes é apresentada nessa seção.

3.8.1 Identificação de Protótipos

Seja a existência de k elementos prototípicos $\bar{\mu}^i \in \mathbb{R}^m, 1 \leq i \leq k$, representando a massa de dados obtida a partir do monitoramento histórico da utilização de um recurso computacional. Assim, $\bar{\mu}^i = \{\mu^i_j \in \mathbb{R}, 1 \leq j \leq m\}$.

Quando um objeto $\bar{x} = \{x_j, 1 \leq j \leq m\}$, não classificado, é apresentado ao algoritmo de reconhecimento de padrões, ele é comparado aos k elementos prototípicos. Como mencionado, esses objetos estão incompletos: algumas de suas dimensões são desconhecidas. Assim, $\exists q < m$ tal que os valores $x_j, q < j \leq m$, são desconhecidos. Eles representaram o comportamento futuro de utilização do recurso.

Seja o objeto $\bar{x}' = \{x'_j, 1 \leq j \leq q\}$ tal que $x'_j = x_j, 1 \leq j \leq q$. Esse objeto contém as dimensões conhecidas de $\bar{x}$. São construídos objetos $\bar{\mu}^{i'} = \{\mu^{i'}_j, 1 \leq j \leq q\}$, tais que $\mu^{i'}_j = \mu^i_j, 1 \leq j \leq q$, correspondentes aos elementos prototípicos fracionados no ponto q.

É determinado o elemento prototípico p, fracionado, tal que a distância $d(\bar{x}', \bar{\mu}^i)$ seja mínima. Assim $\bar{\mu}^p$ é o elemento prototípico reconhecido.

Assumindo que o comportamento de $\bar{x}$ é semelhante ao de $\bar{\mu}^p$, as dimensões desconhecidas de $\bar{x}$ podem ser inferidas de modo que $x_j = \mu^p_j, q < j \leq m$. O objeto $\bar{x}$ fica, então, completamente caracterizado:

$$x_j = \begin{cases} x_j & 1 \leq j \leq q \,(\text{conhecidas}) \\ \mu^p_j & q < j \leq m \,(\text{inferidas}) \end{cases} \tag{3.12}$$

3.8.2 Requisições do InteGrade

As requisições de recursos por parte do software gerenciador da grade têm por objetivo a utilização da capacidade ociosa do recurso na execução de alguma tarefa da grade. Para que a resposta quanto à disponibilidade possa ser enviada à grade, é necessário que o algoritmo de predição de comportamento receba, como parâmetros da solicitação, o nível mínimo de disponibilidade desejada e o tempo durante o qual os recursos serão destinados à grade.

Assim, a predição de comportamento futuro deve ser feita para um período de tempo especificado e deve ser baseada em um nível mínimo de disponibilidade que atenda à quantidade de recurso demandada.

Em termos do algoritmo de predição, deseja-se inferir o comportamento do objeto não classificado $\bar{x}$ durante um intervalo de tempo equivalente a κ dimensões desse objeto. É parâmetro ainda, o nível mínimo de disponibilidade desejada δ.

Utilizando o processo descrito na seção anterior, é possível verificar se os valores inferidos $x_j = \mu_j^p$, para o intervalo futuro $[q, q + \kappa]$, atendem ao nível mínimo de disponibilidade desejada.

Há duas opções para a resposta do algortimo de predição de ociosidade:

- sim, a máquina irá apresentar ociosidade suficiente para o nível mínimo de disponibilidade desejada, durante o intervalo de tempo requerido; ou
- não, os recursos não estarão ociosos, dados os parâmetros desejados.

3.9 Interpretação de Resultados

A interpretação dos resultados e o estabelecimento de medidas de avaliação de desempenho são as últimas etapas da análise de conglomerados. Alguns recursos comumente utilizados nessa fase são:

- participação de um analista especialista na área de pesquisa em que ocorre a aplicação da análise de conglomerados; e
- coleta de dados estatísticos sobre cada conglomerado existente na partição final dos objetos.

O que deve, entretanto, ser considerado de forma prioritária na interpretação dos resultados é que a avaliação de desempenho da aplicação das técnicas de *clustering* deve refletir uma comparação entre os resultados obtidos e os objetivos propostos. Os testes de validação realizados devem ser baseados em algum objetivo mensurável previamente estabelecido.

Na InteGrade, como o objetivo da análise de conglomerados é realizar inferência de comportamento de uso dos recursos, uma medida intuitiva e bastante representativa deste objetivo é o índice de acertos nas inferências realizadas. Em última instância, o algoritmo deverá responder "sim" ou "não" a uma requisição da grade. Medir quão eficientes foram as respostas dadas pelo algoritmo é, portanto, uma métrica de desempenho bastante adequada.

Essa medida é análoga à medida de quão precisa foi a atribuição de um objeto a um *cluster* [22]. Calcular um índice de acertos nas inferências realizadas, entretanto, estabele uma medida mais expressiva do desempenho apresentado, considerando objetivo de predição de ociosidade.

3.9.1 Matriz de Disponibilidade

O índice de acertos do algoritmo de predição mede qual o percentual de vezes em que a resposta dada pelo algoritmo correspondeu à realidade. Esse índice de acertos só pode ser medido após passado, e conhecido, o intervalo de tempo em que a ociosidade do recurso foi inferida.

O índice de acertos pode ser computado para cada conjunto de parâmetros passados nas requisições da grade:

- intervalo de tempo em que a ociosidade é inferida; e
- nível mínimo de disponibilidade desejada.

Dessa forma, pode ser construída uma matriz de índices de acertos de tal que:

1. Cada coluna contém os índices de acertos observados para um determinado intervalo de tempo utilizado como parâmetro; e
2. Cada linha contém os índices de acertos observados para um determinado nível de disponibilidade.

Essa estrutura é denominada, neste trabalho, “matriz de disponibilidade".

Capítulo 4

Ensaios

Foram realizados alguns ensaios para avaliar o uso de análise de conglomerados no processo de identificação de padrões de uso de recursos computacionais. Os ensaios foram construídos de forma a simular a utilização de um processo de *clustering* no contexto do InteGrade. Para tanto, foram utilizadas algumas máquinas da rede IME (rede de computadores existente no IME-USP).

Tomando a observação de utilização histórica dos recursos dessas máquinas, foram identificados, pelo processo de *clustering*, os comportamentos prototípicos existentes. Esses comportamentos serviram de entrada para o algoritmo de reconhecimento de padrões implementado. Com esse algoritmo, foi processada a predição de ociosidade para um conjunto de comportamentos inicialmente não submetidos ao processo de *clustering*.

Neste capítulo, os ensaios são descritos detalhadamente e é discutida a forma utilizada para validar a capacidade de predição da solução proposta.

4.1 Massa de Dados

Quatro máquinas da rede IME foram monitoradas, por aproximadamente dois anos, quanto ao seu comportamento em relação à utilização de recursos computacionais. Com esse monitoramento, foi possível construir uma massa de dados contendo as informações de utilização histórica desses recursos. Essa massa de dados foi submetida ao processo de *clustering*, resultando em sua partição em conglomerados de objetos semelhantes.

4.1.1 Máquinas

As máquinas utilizadas para fins de simulação do processo de reconhecimento de padrões do InteGrade foram quatro: *kama*, *oncoto*, *gsd* e *lingcomp*. Cada uma dessas máquinas tem aplicação e perfil de utilização distintos uma das outras.

- *kama* — máquina de uso geral, bastante utilizada pelos usuários da rede IME. Essa máquina atende a diversos *X-terminals* além de ser o principal

ponto de acesso à rede IME, partindo-se de localizações remotas. Apresenta comportamento bastante uniforme com relação ao uso dos recursos computacionais.

- *oncoto* — máquina pessoal de um dos usuários da rede IME. Esse tipo de máquina tem bastante potencial de utilização por parte da grade oportunista. Pertence a um usuário específico, com poucos acessos remotos. Possui alguns picos de utilização e momentos de grande disponibilidade de recursos, apresentando comportamento estável em relação ao uso de seus recursos.

- *gsd* — máquina utilizada por um grupo de pesquisa, o Grupo de Sistemas Distribuídos [23]. Esse grupo é composto por alunos de graduação, de pós-graduação e de professores que utilizam a máquina no desenvolvimento de seus projetos. Além disso, esta máquina é utilizada como servidor de páginas Web do grupo e como estação de trabalho de um dos professores. Tal qual a máquina *kama*, apresenta comportamento bastante variável em relação à utilização dos recursos computacionais.

- *lingcomp* — máquina utilizada por alunos de graduação e pós-graduação envolvidos com pesquisas em lingüística computacional. A utilização dos recursos desta máquina é relativamente uniforme, semelhante à apresentada pela máquina *oncoto*, apresentando, por outro lado, picos de utilização bem mais longos, uma vez que as aplicações de lingüística computacional são normamente grandes consumidoras de recursos.

4.1.2 Recursos

Foi monitorada a utilização de recursos (CPU, memória física, SWAP e disco rígido) dessas máquinas por um longo período de tempo, entre os anos de 2003 e 2005. A coleta de dados foi realizada com a manutenção de um serviço agendado em cada uma dessas máquinas. Esse serviço era executado por um arquivo com comandos do sistema operacional em lote. Esses comandos medem a utilização corrente de cada recurso.

Os ensaios realizados fizeram uso do monitoramento da utilização de capacidade de processamento (CPU) e de memória física. O período de coleta de dados considerado para cada uma das máquinas é mostrado na Tabela 4.1.

4.2 Metodologia

Na Seção 3.1, foi descrito o processo de reconhecimento de padrões, através do uso da análise de conglomerados. Foi visto que as entradas desse processo são:

- massa de dados com monitoramento histórico; e

- um objeto não classificado, representando a observação do comportamento corrente.

Tabela 4.1: Período da coleta dos dados utilizados na realização dos ensaios.

Máquina	*CPU*	*Memória física*
kama	Jan/05 a Out/05	Ago/03 a Fev/04
oncoto	Mai/05 a Nov/05	Mai/05 a Nov/05
gsd	Jan/05 a Out/05	Jan/05 a Out/05
lingcomp	Jan/05 a Jul/05	Jan/05 a Jul/05

A saída é o modo de operação inferido (comportamento prototípico), com o comportamento esperado para a utilização dos recursos computacionais num futuro próximo.

4.2.1 Descrição Geral

Os ensaios realizados consistem, basicamente, em simular diversas vezes o processo de reconhecimento de padrões para diversas combinações de parâmetros que definem o *clustering*. Com base nessas simulações, foram propostas e realizadas algumas medidas de desempenho, propiciando a comparação entre as diversas alternativas existentes de aplicação das técnicas aqui descritas. A metodologia utilizada para a realização destes ensaios é descrita a seguir.

As simulações são realizadas para cada máquina separadamente. O comportamento da máquina em análise numa simulação, com relação ao uso de um recurso específico (CPU ou memória física), é representado por um objeto integrante da massa de dados inicial. Todos os objetos desta massa contêm informações sobre o uso de um tipo de recurso durante o intervalo de tempo de 48 horas. Objetos que representam uso de recursos distintos são processados em simulações distintas.

Conforme visto anteriormente, o objetivo do processo de reconhecimento de padrões é inferir o comportamento da máquina num futuro próximo. Os ensaios, por outro lado, são realizados apenas com informações que representam o comportamento passado.

Para que o comportamento futuro seja fabricado artificialmente, alguns dos objetos do conjunto inicial de dados são mantidos desconhecidos do algoritmo na primeira fase dos ensaios. Para que nem todos os objetos sejam apresentados ao algoritmo no passo inicial, a massa de dados utilizada na simulação é dividida em duas partes disjuntas: corpo de treino e corpo de testes.

A escolha dos objetos que comporão cada uma destas partes é feita ao acaso, no processamento identificado como "Divisão da massa de dados", no fluxograma da Figura 4.1. Cada objeto da massa de dados é tomado com uma determinada probabilidade p para fazer parte do corpo de testes. Por conseqüência, cada objeto tem probabilidade de $(1 - p)$ de fazer parte do corpo de treino.

Nos ensaios realizados, foi utilizado $p = 0,2$. Dessa forma, aproximadamente 80% dos objetos fazem parte do corpo de treino e o restante dos objetos compõe o corpo de testes.

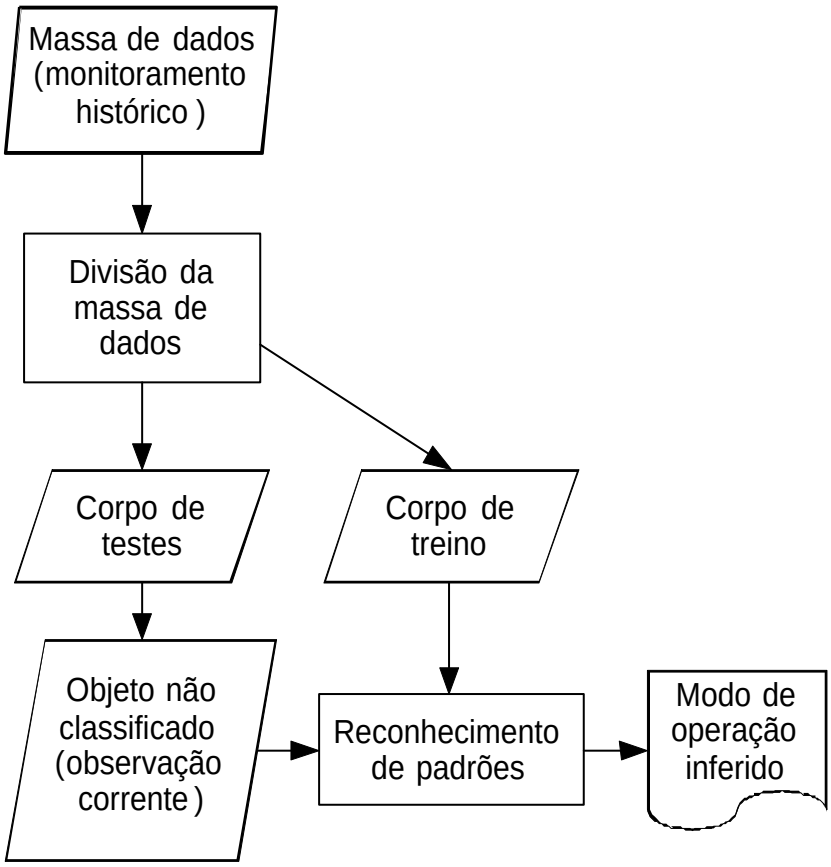

Figura 4.1: Fluxograma das informações tratadas nos ensaios.

É importante ressaltar que a definição de um corpo de treino não caracteriza o processo como um modelo de aprendizado supervisionado. Como citado na Seção 2.3.2, não há presença de objetos pré-classificados. O corpo de treino apenas assume o papel da massa de dados necessária à formação dos *clusters* que representam os modos padrões de operação. Após o processamento do corpo de treino, o software já é capaz de realizar a inferência de comportamento futuro.

A massa de testes não é utilizada no processo de *clustering*. Ela contém todos os objetos que foram mantidos desconhecidos no primeiro passo dos ensaios. Sua função é fornecer os objetos não classificados ao algoritmo de identificação de protótipos, que faz parte da etapa "Reconhecimento de Padrões" do fluxograma acima.

Em uma simulação, todos os objetos do corpo de testes são submetidos a este algoritmo. Assim, a predição de comportamento futuro é feita em objetos

que representam, na verdade, o comportamento no passado. Uma fração desses objetos é mantida desconhecida, tornando-os objetos cujo comportamento a partir de um determinado ponto possa ser previsto.

Assim, para possibilitar a realização de predição futura, o vetor de dados que caracteriza cada objeto do corpo de testes é dividido em dois, conforme indica o ponto t da Figura 4.2. Apenas a primeira parte do vetor fracionado é utilizada e o processo de reconhecimento ocorre como o previsto na Seção 3.8.1.

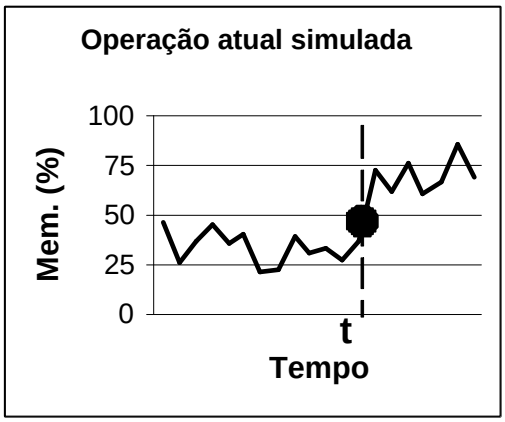

Figura 4.2: Objeto do corpo de testes é fracionado em um ponto t e submetido para reconhecimento de padrões.

O ensaio realizado simula o pedido de submissão de uma tarefa pelo InteGrade e o ponto onde o objeto do corpo de testes é fracionado simula o momento em que esta requisição foi feita. Assim, um objeto incompleto é submetido ao algoritmo de identificação de protótipos. Nestas simulações, sempre foi tomado o período de 24 horas anteriores ao ponto t para caracterizar esse objeto incompleto.

A resposta do algoritmo será o modo de operação esperado, representado pelo comportamento prototípico definido por um conglomerado. Como se trata, por outro lado, de um dado histórico que apenas foi fracionado, o comportamento manifestado após a requisição de uso é conhecido. Esse comportamento é confrontado contra o comportamento previsto pelo elemento prototípico.

É possível realizar uma validação da inferência realizada, com o estabelecimento de índices de acertos e medidas de desempenho para a predição realizada.

4.2.2 Simulações

Foi visto no Capítulo 2.3.2 que há diversas alternativas para a implementação das técnicas de *clustering*. Em um ensaio, no escopo deste trabalho, é executada uma simulação do processo de reconhecimento de padrões, utilizando uma

combinação específica dos parâmetros que definem uma determinada alternativa existente.

A Seção 3.9.1 estabelece que seja desenvolvida uma estrutura denominada matriz de disponibilidade. Haverá uma matriz de disponibilidade para cada simulação realizada. A matriz é da forma $D = \{d_{ij}; 1 \leq i \leq \theta, 1 \leq j \leq \lambda\}$. λ é a dimensão do conjunto dos tempos de disponibilidade que se deseja submeter para validação (e.g. 30 minutos, 1 hora, 4 horas). Analogamente, θ é a dimensão do conjunto de níveis de disponibilidade de um recurso, que se deseja testar. O nível de disponibilidade corresponde ao nível mínimo de disponibilidade desejada pela grade para utilização do recurso.

Cada d_{ij} é um número real, $0 \leq d_{ij} \leq 1$, que indica o índice de acertos da predição de ociosidade ou utilização de um recurso (aquele considerado na simulação) durante um determinado intervalo de tempo, respeitando um nível de disponibilidade requisitado pelo InteGrade.

Nos ensaios realizados neste trabalho, os tempos de disponibilidade que foram utilizados para validar a predição realizada são os elementos do conjunto Λ = {10, 20, 30, 60, 90, 120, 240}, cujos valores estão expressos em minutos. Já os níveis de disponibilidade utilizados foram os elementos do conjunto Θ = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}, cujos valores representam a fração total dos recursos.

Assim, para cada simulação é criada uma matriz de disponibilidade, de dimensões 9 x7. Por exemplo, em uma determinada simulação o elemento d_{45} representa o índice de acertos de predição de ociosidade ou utilização de um recurso, durante o intervalo de 90 minutos para um nível de disponibilidade de 40% da quantidade total de recurso existente na máquina.

Além da matriz de disponibilidade, são resultados obtidos em cada simulação:

- tempo de execução do processo de *clustering*, com o tempo gasto com a partição dos dados e definição do elemento representativo de cada c*luster*; e
- breve descrição de cada *cluster*, com a informação de quantos elementos da massa de dados (corpo de treino) pertencem ao *cluster* e com a identificação de seu elemento representativo. Esse elemento representativo descreve o comportamento prototípico representado pelo *cluster*.

Parâmetros das Simulações

São parâmetros de uma simulação:

- algoritmo a ser executado, tais como algoritmo seqüencial, algoritmo hierárquico aglomerativo ou algoritmo dos k-centros;
- recurso considerado: CPU ou memória física;
- janela de tempo que define o tamanho de um objeto e intervalo de tempo entre as variáveis de um objeto: os ensaios realizados nesse estudo conside-

raram uma janela fixa de 48 horas para um objeto e 5 minutos de intervalo de tempo entre as medidas representadas por variáveis adjacentes;

- arquivo contendo as informações do monitoramento histórico realizado;
- indicador booleano para identificar se o número de *clusters* será previamente fornecido ou não;
- número de *clusters* pré-determinado;
- indicador booleano para identificar se as variáveis que descrevem um objeto são as medidas de utilização propriamente ditas ou se são as variações das medidas;
- forma de se calcular o objeto representativo de um *cluster* (ponto médio, centro médio ou centro mediano);
- métrica utilizada para cálculo da semelhança entre dois objetos (euclidiana ou de Manhattan);
- métrica utilizada para o cálculo da distância entre conglomerados de objetos (objeto representativo, ligação simples ou completa, método de Ward); e
- percentual, aproximado, de objetos da massa de dados que será considerado para o corpo de testes: os ensaios realizados utilizam 20% como o valor para esse parâmetro.

Esses parâmetros são passados através de um arquivo de configuração carregado, no momento de início do ensaio, pelo módulo de software implementado.

Tipos de Simulação

Um conjunto de parâmetros de configuração define uma simulação. Para que as conclusões sobre o desempenho de uma determinada configuração sejam mais precisas, é possível que se determine, no arquivo de configuração citado, o número de simulações que devem ser realizadas com o mesmo conjunto de parâmetros. Tais simulações podem ser consideradas do mesmo tipo.

Sendo executadas várias simulações de um mesmo tipo, é possível que sejam realizadas métricas mais elaboradas para a avaliação de cada tipo de simulação. Essas métricas são baseadas nas medidas obtidas das matrizes de disponibilidade de cada simulação.

Neste estudo, foram realizadas 100 simulações de cada tipo.

4.2.3 Métricas de Desempenho

As métricas de desempenho são necessárias para que se possam comparar os diversos métodos utilizados nos ensaios. A escolha das métricas deve refletir os objetivos do processo proposto: melhores indicadores devem ocorrer em situações em que os resultados dos ensaios mais se aproximaram dos resultados esperados.

Índice de Acertos — Caixa Branca *versus* Caixa Preta

O índice de acertos é a métrica primária para avaliação de desempenho de um algoritmo. Esse índice é definido no contexto de uma única simulação e diz respeito ao percentual de acertos do algoritmo na previsão de ociosidade ou utilização de um recurso. A ociosiodade é medida por um determinado período de tempo e atendendo a um nível mínimo de disponibilidade desejada.

O percentual de acertos é calculado através da contagem dos elementos integrantes do corpo de testes que tiveram seu comportamento corretamente inferido pelo algoritmo. A razão entre o número destes elementos e o número total de elementos do corpo de testes resulta no percentual desejado.

A questão resume-se, então, em como indicar se um comportamento foi corretamente inferido ou não. Em caso afirmativo, considera-se que houve acerto na predição.

Seja um objeto $\bar{x} = \{x_i, 1 \leq i \leq m\}$, com m dimensões, que faz parte do corpo de testes. Cada uma dessas dimensões representa a utilização percentual de um recurso computacional num instante de tempo. Como visto anteriormente, antes de o objeto ser submetido ao processo de reconhecimento do modo de operação, é tomado ao acaso um ponto $q < m$ tal que os valores $x_i, q < i \leq m$, são mantidos ocultos ao algoritmo de reconhecimento. A dimensão q representa o local em que o objeto foi fracionado.

De acordo com a Seção 3.8, após o processamento deste objeto $\bar{x}$, o algoritmo de reconhecimento infere o comportamento futuro, a partir do ponto q, retornando um outro objeto $\bar{y} = \{y_i, 1 \leq i \leq m\}$, que é o comportamento esperado para a máquina em relação à utilização do recurso. Esse objeto $\bar{y}$ é tal que:

$$y_i = \begin{cases} x_i & 1 \leq i \leq q \\ \mu_i & q < i \leq m \end{cases}$$

Os valores $\mu_i, q < i \leq m$, são obtidos a partir do elemento prototípico e representam a inferência realizada.

Supondo que esteja sendo considerada a predição de comportamento para um intervalo de tempo equivalente a κ dimensões do vetor de dados e que o nível de disponibilidade desejada é δ. A resposta do algoritmo deve ser uma das duas opções:

- sim, a máquina irá apresentar ociosidade suficiente para o nível mínimo de disponibilidade desejada, durante o intervalo de tempo requerido; ou
- não, os recursos não estarão ociosos, dados os paramêtros desejados.

Em termos algébricos, se $\exists i, q < i \leq (q + \kappa)$ tal que $\mu_i + \delta > 1$ então a resposta do algoritmo é não. Do contrário, se $\mu_i + \delta \leq 1, \forall i, q < i \leq (q + \kappa)$ então a resposta é sim.

Como o comportamento real do vetor $\bar{x}$ é conhecido, é possível averiguar se de fato a máquina estava ociosa ou não. Se $\exists i, q < i \leq (q+\kappa)$ tal que $x_i + \delta > 1$ então a máquina não tinha recursos suficientes para atender a solicitação. Do contrário, se $x_i + \delta \leq 1, \forall i, q < i \leq (q + \kappa)$ então a máquina estava ociosa.

Se a resposta baseada no comportamento inferido, seja ela positiva ou negativa, coincidir com o que de fato ocorreu, então houve acerto. Com a definição da ferramenta para aferição de acertos, é possível que seja calculado o índice de acertos mencionado no início desta seção.

De acordo com o descrito até aqui, havendo indisponibilidade de recursos em algum momento da janela de tempo requerida, a resposta do algoritmo é não. Não há qualquer preocupação em realizar comparações instante a instante para verificar se o comportamento previsto coincide com o comportamento apresentado pelo objeto real. Essa abordagem é do tipo denominado, no contexto deste trabalho, *caixa preta*.

Se há interesse na verificação instante a instante, então deve ser utilizada uma outra abordagem, do tipo *caixa branca*. Nesta outra abordagem, o acerto só é computado se, a cada instante de tempo, a predição de ociosidade coincide com o que ocorreu de fato.

Em termos algébricos, o acerto só é computado se:

$$(\mu_i + \delta - 1)\,(x_i + \delta - 1) \geq 0, \forall i, q < i \leq (q + \kappa) \tag{4.1}$$

Se os conceitos discutidos na análise de conglomerados forem considerados de forma geral, o método mais preciso para medir o desempenho do algoritmo de inferência é o da *caixa branca*.

A janela de disponibilidade é um aspecto inerente ao reconhecimento de padrões que se deseja aplicar no contexto do InteGrade. Exatamente motivado por esta característica específica das requisições do InteGrade, o método da *caixa preta* foi proposto para medir o desempenho das técnicas avaliadas. Pois, recapitulando, a escolha das métricas deve refletir os objetivos do processo proposto.

De toda forma, nos ensaios realizados foram calculados os índices de acertos tanto do tipo *caixa branca* como do tipo *caixa preta*.

Construção da Matriz de Disponibilidade

Estabelecido o modelo para cálculo do índice de acertos, a construção da matriz de disponibilidade é imediata.

Em uma simulação, são verificados alguns valores para o nível mínimo de disponibilidade desejada δ. Sejam $\delta_i, 1 \leq i \leq \theta$, os $\theta \in \mathbb{N}$ valores testados. De forma análoga, são simulados diferentes intervalos de tempo κ em que se deseja verificar a ociosidade de um recurso. Sejam $\kappa_i, 1 \leq i \leq \lambda$, os $\lambda \in \mathbb{N}$ valores testados.

Para cada par $\langle \delta_i, \kappa_j \rangle, 1 \leq i \leq \theta, 1 \leq j \leq \lambda$, é calculado o índice de acertos d_{ij}, nos termos do que foi já foi descrito. Tomando todos os valores calculados, constrói-se a matriz de disponibilidade:

$$D_{\theta x \lambda} = (d_{\mathrm{ij}})\,, 1 \leq i \leq \theta, 1 \leq j \leq \lambda \tag{4.2}$$

Como há duas maneiras de se calcular o índice de acertos, *caixa branca* e *caixa preta*, são criadas duas matrizes, uma para cada tipo de índice, para toda simulação realizada.

Análise do Caso Médio

O resultado de cada simulação é descrito principalmente pela matriz de disponibilidade. Para consolidar a medida de desempenho apresentado em uma simulação, é intuitivo que se calcule um índice de acertos médio. Utilizando a média aritmética dos acertos, o índice de acertos médio Φ é dado por:

$$\Phi = \frac{\sum_{i=1}^{\theta} \sum_{j=1}^{\lambda} d_{\mathrm{ij}}}{\theta\lambda} \tag{4.3}$$

O índice Φ é calculado para cada simulação. Como cada tipo de simulação é executado várias vezes, calcula-se esse índice para instância de simulação realizada. Considerando que foram realizadas N simulações de cada tipo, são calculados N índices de acertos médio $\Phi_i, 1 \leq i \leq N$. O desempenho médio de um tipo de simulação, caracterizada por um conjunto de parâmetros de configuração específico, é definido por:

$$\Phi_{\mathrm{medio}} = \frac{\sum_{i=1}^{N} \Phi_i}{N}; \sigma_{\Phi} = \sqrt{\frac{\sum_{i=1}^{n} \left(\Phi_i - \Phi_{\mathrm{medio}}\right)^2}{N}} \tag{4.4}$$

Φ_{medio} é o desempenho médio propriamente dito e σ_{Φ} o desvio padrão dessa média.

Análise Prática do Pior Caso

Durante a realização dos ensaios, empiricamente concluiu-se que o índice de acertos médio camuflava o real desempenho de uma simulação.

Para grandes valores do nível mínimo de disponibilidade desejada δ, há maior probabilidade de as máquinas não estarem disponíveis. De forma análoga, para valores muito pequenos de δ, há maior probabilidade de as máquinas estarem ociosas o suficiente para atender às requisições.

Estas situações acabam por, muitas vezes, melhorar o índice de desempenho médio, mesmo que isso não se deva necessariamente ao poder de predição da técnica utilizada.

Para minimizar esse efeito, foi realizada uma análise prática do pior caso para cada simulação. Assim, um outro índice também foi utilizado. Neste caso, a média computada levou em consideração apenas o pior índice de acertos para cada intervalo de tempo constante da matriz de disponibilidade. O índice de pior caso é calculado como segue:

$$\Psi = \frac{\sum_{j=1}^{\lambda} \min\left\{d_{\mathrm{ij}}, 1 \leq i \leq \theta\right\}}{\lambda} \tag{4.5}$$

Na Equação 4.5, a função $\min\{\chi\}$ retorna o elemento de menor valor do conjunto $\chi \subset \mathbb{R}$.

De forma semelhante ao outro índice, o índice Ψ é calculado para cada simulação. Também foi calculado o desempenho médio de um tipo de simulação segundo a análise prática do pior caso. Analogamente ao caso anterior:

$$\Psi_{\text{medio}} = \frac{\sum_{i=1}^{N} \Psi_i}{N}; \sigma_\Psi = \sqrt{\frac{\sum_{i=1}^{n} (\Psi_i - \Psi_{\text{medio}})^2}{N}} \tag{4.6}$$

Ψ_{medio} é o desempenho médio propriamente dito e σ_Ψ o desvio padrão dessa média.

Desempenho Relativo

Algumas medidas de desempenho foram propostas para avaliar a eficácia das técnicas apresentadas. Foram apresentadas métricas que avaliam o desempenho absoluto, baseadas no índice de acertos do mecanismo de predição de comportamento futuro.

Assim, em cada simulação realizada, os resultados obtidos com a utilização de técnicas de *clustering* foram comparados com outras duas formas de realizar tal tarefa. Em uma delas, o comportamento esperado para a utilização de recursos por parte da máquina é igual à utilização média do recurso nas 24 horas anteriores. Na outra, simplesmente é assumido que o comportamento se manterá constante e igual ao observado no momento da requisição de recursos por parte do InteGrade.

Para cada simulação realizada, foi calculado o índice médio de acertos e o índice de pior caso, tanto para a predição realizada com os *clusters* quanto para a realizada com base na observação do passado recente.

Considerando que cada tipo de simulação foi executado N vezes, os seguintes índices foram calculados:

- Φ_i^{Clusters} e Ψ_i^{Clusters}, $1 \leq i \leq N$, com medidas de acerto para as predições realizadas com *clusters*;
- Φ_i^{24h} e Ψ_i^{24h}, $1 \leq i \leq N$, com medidas de acerto para as predições realizadas com base na observação média das 24 horas anteriores de utilização dos recursos; e
- Φ_i^{Ultimo} e Ψ_i^{Ultimo}, $1 \leq i \leq N$, com medidas de acerto para as predições realizadas com base na observação instantânea de utilização dos recursos, no momento em que foi realizada a requisição simulada.

Os índices médios Φ_{medio} e Ψ_{medio} foram calculados para cada modelo de inferência e comparados entre si.

Além da comparação entre os índices médios, foram criadas outras 4 séries,

definidas da seguinte forma:

$$\Phi_i^I = \Phi_i^{\text{Clusters}} - \Phi_i^{24h} \tag{4.7}$$

$$\Psi_i^I = \Psi_i^{\text{Clusters}} - \Psi_i^{24h} \tag{4.8}$$

$$\Phi_i^{\text{II}} = \Phi_i^{\text{Clusters}} - \Phi_i^{\text{Ultimo}} \tag{4.9}$$

$$\Psi_i^{\text{II}} = \Psi_i^{\text{Clusters}} - \Psi_i^{\text{Ultimo}} \tag{4.10}$$

Para todas as séries $1 \leq i \leq N$.

A análise realizada com cada uma dessas séries é bem semelhante. Para simplificar a explanação dessa análise, será tomada como exemplo, sem perda de generalidade, a série de medidas Φ_i^I.

As medidas Φ_i^I comparam, simulação a simulação, o desempenho da inferência utilizando *clusters* com o da inferência baseada nas 24 horas anteriores. Nas situações em que as medidas são positivas, a inferência com *clusters* foi mais bem sucedida. Nas situações em que as medidas são negativas, ocorreu o inverso.

São calculados Φ_{medio}^I e σ_{Φ^I}, conforme descrito na Equação 4.4. Se Φ_{medio}^I for positivo, significa que o desempenho médio da inferência com os *clusters* foi superior.

Assumindo, por questão de simplicidade, que Φ_i^I segue uma distribuição normal, pode-se construir a seguinte série: $\{\frac{\Phi_i^I - \Phi_{\text{medio}}^I}{\sigma_{\Phi^I}}\}_{1 \leq i \leq N}$. Por construção, a nova série também segue uma distribuição normal e ainda apresenta média zero e desvio padrão unitário. Uma distribuição normal ζ que possua média zero e desvio padrão unitário é dita distribuição normal padrão.

A probabilidade de $\zeta \leq k, k \in \mathbb{R}$ é dada pela função cumulativa normal padrão F:

$$F(k) = \int_{-\infty}^{k} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} dx \tag{4.11}$$

Como a distribuição normal padrão é simétrica, tem-se que $F(-k) = 1 - F(k)$. Assim:

$$\text{Prob}[\zeta > -k] = 1 - \text{Prob}[\zeta \leq -k] = 1 - F(-k) = F(k) \tag{4.12}$$

Como $\{\frac{\Phi_i^I - \Phi_{\text{medio}}^I}{\sigma_{\Phi^I}}\}_{1 \leq i \leq N}$ é uma distribuição normal padrão, então:

$$\text{Prob}[\frac{\Phi_i^I - \Phi_{\text{medio}}^I}{\sigma_{\Phi^I}} > -k] = F(k) \tag{4.13}$$

Tomando $k = \frac{\Phi_{\text{medio}}^I}{\sigma_{\Phi^I}}$, a Equação 4.13 é reescrita:

$$\text{Prob}[\frac{\Phi_i^I}{\sigma_{\Phi^I}} > 0] = F(\frac{\Phi_{\text{medio}}^I}{\sigma_{\Phi^I}}) \tag{4.14}$$

Como $\sigma_{\Phi^I} > 0$, então:

$$\text{Prob}[\Phi_i^I > 0] = F(\frac{\Phi_{\text{medio}}^I}{\sigma_{\Phi^I}}) \tag{4.15}$$

A medida $f_I = F(\frac{\Phi^I_{\text{medio}}}{\sigma_{\Phi^I}})$ indica a probabilidade de Φ^I_i ser positiva e, portanto, de o desempenho de uma inferência realizada com base nas técnicas de *clustering* ser superior àquele de uma inferência realizada com base na observação das 24 horas anteriores de utilização do recurso computacional.

De modo análogo, foram construídas as medidas seguintes:

$$g_I = F(\frac{\Psi^I_{\text{medio}}}{\sigma_{\Psi^I}}) \tag{4.16}$$

$$f_{\text{II}} = F(\frac{\Phi^{\text{II}}_{\text{medio}}}{\sigma_{\Phi^{\text{II}}}}) \tag{4.17}$$

$$g_{\text{II}} = F(\frac{\Psi^{\text{II}}_{\text{medio}}}{\sigma_{\Psi^{\text{II}}}}) \tag{4.18}$$

Essas medidas são os indicadores finais de quão eficaz é a aplicação das técnicas de *clustering* em relação a técnicas intuitivas mais simples.

4.3 Resultados Obtidos

Nesta seção, serão apresentados os resultados obtidos com os ensaios. Foi visto, na descrição da metodologia dos ensaios, que a medida primária de desempenho é o índice de acertos das predições. Com base nesse índice, são construídas as matrizes de disponibilidade. São executadas diversas simulações de um mesmo tipo e feitas algumas outras medidas para comparar o desempenho das variadas técnicas.

A Tabela 4.2 ilustra o conceito de matriz de disponibilidade. A matriz deste exemplo foi obtida com a simulação do algoritmo hierárquico aglomerativo aplicado à predição de disponibilidade de CPU na máquina *kama*. Os objetos submetidos ao *clustering* foram caracterizados pelas variações nas medidas de utilização. A métrica de semelhança utilizada foi a distância euclidiana e a distância entre conglomerados de objetos foi medida com base no elemento representativo (ponto médio). Os índices de acertos constantes da matriz foram calculados com base na análise de *caixa branca*.

As colunas da matriz trazem os índices para um dado intervalo de tempo de predição. As linhas, de forma análoga, dizem respeito ao nível mínimo de disponibilidade desejada. O índice de acertos médio foi de 0.7535 e o índice de acertos no pior caso foi de 0.7465.

Com a construção de uma matriz de disponibilidade para cada simulação é possível que sejam calculados os índices de acertos médio e de acertos no pior caso. A média e o desvio padrão desses índices são utilizados para comparar as técnicas entre si.

Nas seções seguintes, são discutidos os resultados dos ensaios realizados. Um conjunto mais detalhado de tabelas com os resultados realizados pode ser encontrado no Apêndice A.2.

Tabela 4.2: Matriz de disponibilidade (análise de caixa branca).

Nível	*10 min*	*20 min*	*30 min*	*60 min*	*90 min*	*120 min*	*240 min*
10%	1.0000	1.0000	1.0000	0.9839	0.9516	0.9194	0.8065
20%	1.0000	1.0000	0.9839	0.9516	0.9355	0.8871	0.8065
30%	1.0000	0.9677	0.9516	0.9355	0.8871	0.8710	0.7742
40%	0.9839	0.9355	0.9355	0.9355	0.8710	0.8548	0.7419
50%	0.9677	0.9355	0.9032	0.8387	0.7581	0.7581	0.6613
60%	0.9677	0.9516	0.9355	0.9032	0.8065	0.7581	0.6452
70%	0.9516	0.9194	0.9194	0.8871	0.7742	0.7097	0.5968
80%	0.9355	0.9194	0.8710	0.7903	0.7097	0.6613	0.6129
90%	0.9677	0.8548	0.8387	0.7419	0.6774	0.6290	0.5484

Tabela 4.3: Acertos (caso médio) na predição de ociosidade de memória física, usando algoritmo hierárquico aglomerativo.

		Caixa Branca		*Caixa Preta*	
Máquina	*Clusters*	*Medidas*	*Variações*	*Medidas*	*Variações*
		Φ_{medio}	Φ_{medio}	Φ_{medio}	Φ_{medio}
oncoto	*5*	0.960126	0.990351	0.969259	0.995402
oncoto	*10*	0.965654	0.99075	0.973867	0.995586
gsd	5	0.993931	0.99776	0.99652	0.99958
gsd	*10*	0.991643	0.996965	0.995004	0.999354
lingcomp	*5*	0.937909	0.988128	0.948466	0.993948
lingcomp	*10*	0.950504	0.987924	0.959636	0.995181
kama	*5*	0.832754	0.952467	0.868146	0.98202
kama	*10*	0.852695	0.953966	0.890966	0.982065

4.3.1 Caracterização dos Objetos

Foi discutido na Seção 3.3.2 que os objetos submetidos ao *clustering* podem ser caracterizados tanto através das medidas de utilização dos recursos computacionais quanto através da variação dessas medidas. Os ensaios realizados exploraram a diferença entre as duas abordagens. As tabelas seguintes resumem os resultados encontrados.

A Tabela 4.3 revela que, de acordo com a análise do caso médio (Φ), a utilização de variações nas medidas de utilização produz sistematicamente melhores resultados que a utilização das medidas de utilização propriamente ditas.

Os resultados foram produzidos com o uso do algoritmo hierárquico aglomerativo para a definição dos comportamentos prototípicos em relação à utilização de memória física nas máquinas. O número de *clusters* existentes foi pré-determinado e a distância entre os *clusters* foi medida através da distância

Tabela 4.4: Acertos (pior caso) na predição de ociosidade de memória física, usando algoritmo hierárquico aglomerativo.

		Caixa Branca		*Caixa Preta*	
Máquina	*Clusters*	*Medidas*	*Variações*	*Medidas*	*Variações*
		Ψ_{medio}	Ψ_{medio}	Ψ_{medio}	Ψ_{medio}
oncoto	*5*	0.898593	0.968791	0.91458	0.982925
oncoto	*10*	0.900914	0.973443	0.919108	0.983489
gsd	*5*	0.975052	0.984409	0.989062	0.995277
gsd	*10*	0.968703	0.985931	0.987549	0.996236
lingcomp	*5*	0.6655	0.928966	0.721604	0.971109
lingcomp	*10*	0.701933	0.940706	0.753068	0.970134
kama	*5*	0.565671	0.859186	0.604622	0.935198
kama	*10*	0.583286	0.858118	0.643271	0.939599

entre seus elementos representativos (ponto médio).

A Tabela 4.4 apresenta os resultados para o mesmo tipo de simulação, entretanto é realizada a análise prática do pior caso (Ψ).

O melhor desempenho das variações das medidas de utilização torna-se ainda mais acentuado quando é realizada a análise prática do pior caso. Esse desempenho superior é observado tanto no caso da análise de *caixa branca* quanto no caso da análise de *caixa preta*.

Considerando agora o uso do algoritmo seqüencial de duas fases para a definição dos comportamentos prototípicos em relação à utilização de CPU nas máquinas, foram observados os resultados descritos na tabela. O número de *clusters* foi pré-estabelecido em algumas simulações e, em outras, determinado pelo próprio algoritmo seqüencial, de acordo com o descrito na Seção 3.7.

A Tabela 4.5 apresenta os resultados deste tipo de simulação. O símbolo N/D apresentado nos resultados dos ensaios refere-se à utilização do algoritmo seqüencial sem informação prévia do número de *clusters*.

A utilização das variações das medidas mostrou-se, de fato, ser a melhor abordagem para caracterizar os objetos. Sistematicamente, como pode ser observado no Apêndice A.2, os resultados foram melhores quando os objetos da massa de dados foram construídos dessa forma.

Constatado o fato, as comparações entre os diversos métodos simulados, bem como as medidas de desempenho relativo f_I, g_I, f_{II} e g_{II}, foram realizadas utilizando *clusters* de objetos que representam as variações nas medidas de utilização dos recursos computacionais. Os resultados dessas comparações são apresentados nas seções seguintes.

4.3.2 Algoritmos Simulados

Os ensaios realizados propiciaram a comparação entre os diversos métodos de *clustering* aplicados ao processo de reconhecimento de padrões de comporta-

Tabela 4.5: Acertos (pior caso) na predição de ociosidade de CPU, usando algoritmo seqüencial.

		Caixa Branca		*Caixa Preta*	
	Clusters	*Medidas*	*Variações*	*Medidas*	*Variações*
		Ψ_{medio}	Ψ_{medio}	Ψ_{medio}	Ψ_{medio}
oncoto	*5*	0.919019	0.934209	0.9212	0.948284
oncoto	*N/D*	0.928537	0.936384	0.939722	0.949698
gsd	*5*	0.32417	0.519771	0.620481	0.759706
gsd	*N/D*	0.368289	0.527898	0.623511	0.771843
lingcomp	*5*	0.914633	0.940351	0.926325	0.956089
lingcomp	*N/D*	0.926971	0.936625	0.945242	0.951535
kama	*5*	0.611554	0.762186	0.738362	0.867337
kama	*N/D*	0.400905	0.764382	0.522849	0.865318

mento das máquinas em relação à utilização de seus recursos computacionais.

Conforme descrito na Seção 4.2.3, a análise de *caixa preta* foi proposta para que a métrica de avaliação média refletisse a existência, nas requisições do InteGrade, do parâmetro de intervalo de disponibilidade. As comparações realizadas nessa seção foram feitas baseadas na análise de caixa preta.

Memória Física

A Tabela 4.6 apresenta os resultados comparativos entre os diversos métodos de *clustering* aplicados à predição de ociosidade de memória física existente nas máquinas. O método identificado como "Ponto médio" é o algoritmo hierárquico aglomerativo, utilizando para o cálculo de semelhança entre dois *clusters* a distância entre os respectivos objetos representativos. Já no método de Ward a distância entre os *clusters* é dada pela distância de Ward.

Nesta tabela, percebe-se que o desempenho das diversas técnicas foi muito semelhante. Inclusive mostrou-se pouco sensível ao número de *clusters* pré-estabelecidos.

Fazendo-se a mesma comparação entre os métodos, mas considerando-se a análise prática do pior caso, os resultados obtidos são os constantes da Tabela 4.7. Novamente os resultados obtidos com a aplicação dos diversos métodos foi bem semelhante. Inclusive, diferentemente da análise anterior, o método da ligação simples apresentou desempenho semelhante ao apresentado pelos demais métodos.

CPU

De forma análoga ao que foi feito para o caso da memória física, a Tabela 4.8 apresenta os resultados comparativos entre os diversos métodos de *clustering* aplicados à predição de ociosidade da capacidade de CPU das máquinas.

Tabela 4.6: Análise comparativa (caso médio) da predição de ociosidade de memória física.

		Análise de caixa preta			
		Ponto médio	*Ligação simples*	*Ligação completa*	*Método de Ward*
	Clusters	Φ_{medio}	Φ_{medio}	Φ_{medio}	Φ_{medio}
oncoto	*5*	0.995402	0.994436	0.995475	0.9954
oncoto	*10*	0.995586	0.995843	0.995878	0.995691
gsd	*5*	0.99958	0.999261	0.999058	0.999023
gsd	*10*	0.999354	0.999254	0.999364	0.999283
lingcomp	*5*	0.993948	0.993932	0.992427	0.996064
lingcomp	*10*	0.995181	0.995932	0.996328	0.996109
kama	*5*	0.98202	0.981818	0.982287	0.982394
kama	*10*	0.982065	0.981871	0.979096	0.983518

Tabela 4.7: Análise comparativa (pior caso) da predição de ociosidade de memória física.

		Análise de caixa preta				
		Ponto médio	*Ligação simples*	*Ligação completa*	*Método de Ward*	*Seqüenc.*
	Clusters	Ψ_{medio}	Ψ_{medio}	Ψ_{medio}	Ψ_{medio}	Ψ_{medio}
oncoto	*5*	0.982925	0.982245	0.981444	0.983002	0.981443
oncoto	*10 - N/D*	0.983489	0.981055	0.982756	0.980292	0.983875
gsd	*5*	0.995277	0.995659	0.996971	0.995932	0.995996
gsd	*10 - N/D*	0.996236	0.995393	0.996927	0.996691	0.997218
lingcomp	*5*	0.971109	0.971509	0.966127	0.969386	0.973015
lingcomp	*10 - N/D*	0.970134	0.975613	0.973765	0.976233	0.978757
kama	*5*	0.935198	0.937987	0.935814	0.942713	0.937048
kama	*10 - N/D*	0.939599	0.940901	0.937913	0.939167	0.932845

Tabela 4.8: Análise comparativa (caso médio) da predição de ociosidade de CPU.

		Análise de caixa preta			
		Ponto médio	Ligação simples	*Ligação completa*	*Método de Ward*
	Clusters	Φ_{medio}	Φ_{medio}	Φ_{medio}	Φ_{medio}
oncoto	5	0.970883	0.97144	0.969461	0.968617
oncoto	*10*	0.972306	0.971294	0.972779	0.972627
gsd	*5*	0.928352	0.923854	0.932314	0.934991
gsd	*10*	0.93149	0.926905	0.940141	0.940205
lingcomp	*5*	0.9806	0.98078	0.980702	0.981423
lingcomp	*10*	0.983402	0.982382	0.982716	0.981092
kama	*5*	0.922115	0.924971	0.922608	0.925283
kama	*10*	0.92447	0.92287	0.924762	0.927239

Nesta tabela, percebe-se que o desempenho das diversas técnicas foi muito semelhante. Novamente o desempenho apresentado mostrou-se pouco sensível ao número de *clusters* pré-estabelecidos.

A Tabela 4.9 traz a mesma comparação dos métodos, mas considerando a análise prática do pior caso. Interessante observar que, novamente, todos os métodos apresentaram desempenho semelhante.

Os resultados obtidos na predição de ociosidade de recursos de memória física mostraram-se melhores que os obtidos na predição de ociosidade de recursos de CPU. Provavelmente, isto está relacionado com o fato de a utilização de memória nestas máquinas ser mais uniforme (com poucas variações de uso) do que a utilização de CPU.

Mesmo no caso da predição de ociosidade de CPU, os resultados para as máquinas *oncoto* e *lingcomp* foram superiores aos resultados das máquinas *kama* e *gsd*. Provavelmente este fenômeno também está relacionado com a uniformidade de utilização dos recursos (*oncoto* e *lingcomp* têm utilização de CPU mais uniforme).

4.3.3 Desempenho Relativo

Além de comparar o desempenho entre as diversas técnicas de *clustering*, os resultados obtidos com estas técnicas foram comparados com os resultados de duas outras formas de se realizar a predição de ociosidade de recursos.

Na primeira delas, a predição realizada é baseada na utilização média do recurso nas 24 horas anteriores ao momento da requisição de recursos. Na outra forma de se realizar a predição, é assumido que o comportamento de utilização se manterá constante e igual ao observado no momento da requisição de recursos.

Nesta seção, as comparações realizadas foram feitas com base nos índices de acertos calculados pelo método da *caixa preta* e foram realizadas análises

Tabela 4.9: Análise comparativa (pior caso) da predição de ociosidade de CPU.

		Análise de caixa preta				
		Ponto médio	*Ligação simples*	*Ligação completa*	*Método de Ward*	*Seqüenc.*
	Clusters	Ψ_{medio}	Ψ_{medio}	Ψ_{medio}	Ψ_{medio}	Ψ_{medio}
oncoto	*5*	0.941453	0.944857	0.942774	0.944029	0.948284
oncoto	*10 - N/D*	0.950787	0.944367	0.943363	0.941313	0.949698
gsd	*5*	0.753132	0.746277	0.75972	0.776045	0.759706
gsd	*10 - N/D*	0.768387	0.750356	0.78743	0.793925	0.771843
lingcomp	*5*	0.952368	0.952941	0.953129	0.952323	0.956089
lingcomp	*10 - N/D*	0.95121	0.958429	0.951665	0.952827	0.951535
kama	*5*	0.86881	0.866897	0.868524	0.868158	0.867337
kama	*10 - N/D*	0.867269	0.865742	0.866915	0.877257	0.865318

práticas do pior caso. Assim, para comparar os métodos, foram propostos e calculados os índices g_I e g_{II}, conforme descrito na Seção 4.2.3. Os resultados para a análise do caso médio foram semelhantes e podem ser verificados no Apêndice A.2.

Memória Física

A tabela 4.10 apresenta os resultados das técnicas de *clustering* comparados aos resultados obtidos com a predição realizada com base na observação das últimas 24 horas de utilização de memória física. Observa-se que a utilização do *clustering* agrega valor ao processo de predição de ociosidade de memória física.

Para todas as máquinas consideradas, a probabilidade de as técnicas de *clustering* apresentarem desempenho superior é bastante alta. Mesmo na máquina gsd, onde observaram-se os piores resultados relativos, essa probabilidade é sempre superior a 80%. Para as demais máquinas, essa probabilidade é próxima de 100%.

Quando, entretanto, o desempenho da predição de ociosidade é comparado ao da outra forma de predição (baseada na utilização instantânea do recurso), apenas a máquina *kama* apresenta índices que justifiquem a aplicação das técnicas de *clustering*. No caso desta máquina, a probabilidade de o desempenho do *clustering* ser superior oscila entre, aproximadamente, 70% e 80%. Isto é ilustrado na Tabela 4.11.

Para as demais máquinas, os melhores índices apresentados são pouco superiores a 50%, indicando que o uso do *clustering* é equivalente ao uso da informação de comportamento instantâneo no momento da requisição do InteGrade.

Tabela 4.10: Desempenho relativo da predição com *clusters* comparado ao da predição baseada nas 24 horas anteriores de utilização de memória física.

		Caixa preta - Pior caso				
		Ponto médio	*Ligação simples*	*Ligação completa*	*Método de Ward*	*Seqüenc.*
	Clusters	g_I	g_I	g_I	g_I	g_I
oncoto	*5*	0.98446	0.97910	0.96374	0.98584	0.96430
oncoto	*10 - N/D*	0.97306	0.97926	0.99057	0.97213	0.98773
gsd	*5*	0.80820	0.82619	0.87906	0.83202	0.81819
gsd	*10 - N/D*	0.85047	0.86372	0.84908	0.83077	0.88159
lingcomp	*5*	0.99809	0.99808	0.99405	0.99179	0.99881
lingcomp	*10 - N/D*	0.99696	0.99860	0.99838	0.99664	0.99547
kama	*5*	0.99926	0.99916	0.99984	0.99962	0.99940
kama	*10 - N/D*	0.99950	0.99976	0.99969	0.99928	0.99863

Tabela 4.11: Desempenho relativo da predição com *clusters* comparado ao da predição baseada na utilização de memória física no momento da requisição.

		Caixa preta - Pior caso				
		Ponto médio	*Ligação Simples*	*Ligação completa*	*Método de Ward*	*Seqüenc.*
	Clusters	g_{II}	g_{II}	g_{II}	g_{II}	g_{II}
oncoto	*5*	0.59186	0.59530	0.61679	0.59582	0.56357
oncoto	*10 - N/D*	0.53236	0.55269	0.56530	0.50389	0.62651
gsd	*5*	0.55164	0.60069	0.58012	0.62486	0.58605
gsd	*10 - N/D*	0.58210	0.58377	0.46423	0.57407	0.55850
lingcomp	*5*	0.36370	0.37543	0.25768	0.36966	0.40828
lingcomp	*10 - N/D*	0.358595	0.43610	0.41189	0.32929	0.56820
kama	*5*	0.74777	0.75079	0.79116	0.79110	0.72649
kama	*10 - N/D*	0.77344	0.75033	0.80028	0.78427	0.77926

Tabela 4.12: Desempenho relativo da predição com *clusters* comparado ao da predição baseada nas 24 horas anteriores de utilização de CPU.

		Caixa preta - Pior caso				
		Ponto médio	*Ligação simples*	*Ligação completa*	*Método de Ward*	*Seqüenc.*
	Clusters	g_I	g_I	g_I	g_I	g_I
oncoto	*5*	0.86403	0.84675	0.87328	0.86763	0.90652
oncoto	*10*	0.92384	0.90393	0.87839	0.88403	0.90689
gsd	*5*	0.82719	0.77693	0.93053	0.92977	0.86116
gsd	*10*	0.86635	0.81892	0.98912	0.98646	0.93372
lingcomp	*5*	0.89390	0.90283	0.90877	0.88937	0.91453
lingcomp	*10*	0.89643	0.92712	0.89345	0.90150	0.88189
kama	*5*	0.99993	0.99999	0.99999	0.99999	0.99999
kama	*10*	0.99999	0.99998	0.99999	0.99999	0.99998

CPU

Também no caso da predição de ociosidade de CPU, conforme é ilustrado na Tabela 4.12, o uso das técnicas de *clustering* é preferível ao uso da média de utilização do recurso nas 24 horas anteriores ao momento da requisição.

De forma análoga ao que foi desenvolvido para a predição de ociosidade de memória, a Tabela 4.13 apresenta os índices de desempenho relativo, obtidos da comparação do processo de *clustering* com a predição baseada na utilização de CPU no momento da requisição.

As máquinas *kama* e *gsd* apresentaram bons índices, superiores a 85%. As máquinas *lingcomp* e *oncoto* apresentaram índices mais modestos, com alguns métodos apresentando índices próximos a 70%.

4.4 Análise dos Resultados

Algumas conclusões gerais podem ser feitas a partir dos resultados dos ensaios:

- O uso de variações de medidas de utilização dos recursos para caracterização dos objetos históricos, a serem submetidos ao *clustering*, apresentou vantagem em relação a caracterizarem-se esses objetos com as próprias medidas absoluta de utilização de recursos.

- Os diversos métodos de *clustering*, de maneira geral, apresentaram bons índices de acertos para as predições propostas. Para uma determinada máquina, alguns métodos podem apresentar ligeira superioridade de desempenho em relação aos outros. Deve, portanto, ser realizada uma bateria de ensaios antes da definição de qual o método que irá constituir os comportamentos prototípicos da máquina.

Tabela 4.13: Desempenho relativo da predição com *clusters* comparado ao da predição baseada na utilização de CPU no momento da requisição.

		Caixa preta - Pior caso				
		Ponto médio	*Ligação simples*	*Ligação completa*	*Método de Ward*	*Seqüenc.*
	Clusters	g_{II}	g_{II}	g_{II}	g_{II}	g_{II}
oncoto	5	0.45309	0.52961	0.61069	0.48506	0.58566
oncoto	*10*	0.52305	0.63660	0.65218	0.47282	0.63395
gsd	*5*	0.89037	0.87303	0.92382	0.91938	0.90968
gsd	*10*	0.95438	0.93010	0.97367	0.99158	0.92211
lingcomp	*5*	0.55137	0.54600	0.55657	0.53929	0.56275
lingcomp	*10*	0.51766	0.74818	0.54102	0.61490	0.53041
kama	*5*	0.93442	0.87120	0.89881	0.90971	0.91174
kama	*10*	0.87818	0.88724	0.85890	0.94636	0.88827

- Empiricamente, pode-se afirmar que os índices de acertos observados são tão melhores quanto mais uniforme é a utilização do recurso computacional.
- O uso de *clustering* para a realização das predições de ociosidade é preferível ao uso da observação média das 24 horas anteriores ao momento da requisição.
- A utilização da informação de utilização instantânea do recurso, no momento da requisição, é muitas vezes equivalente, em termos de desempenho, ao uso de técnicas de *clustering*. Interessante observar que os casos em que o *clustering* é preferível são exatamente aqueles em que a utilização dos recursos é menos uniforme.

Resumindo: se a utilização de recursos é bem uniforme, aparentemente não há necessidade de utilização de técnicas de *clustering* para auxiliar no processo de predição, apesar de os índices de acerto para estas técnicas serem muito bons.

Entretanto, quando a utilização dos recursos for mais variada, mesmo que os índices de acerto das técnicas de *clustering* não sejam tão bons, são melhores que as técnicas mais simples de predição.

Capítulo 5

Conclusões e Comentários Finais

A computação em grades oportunistas é uma ferramenta que possibilita a alavancagem do poder de processamento de uma rede de computadores, através da utilização da capacidade ociosa dos recursos computacionais existentes em instituições diversas. Para otimizar o aproveitamento dos recursos ociosos, o projeto InteGrade prevê que seja desenvolvido um mecanismo a ser utilizado para a predição de ociosidade na rede.

O emprego de técnicas de análise de conglomerados é justificado para realizar a predição baseada em grupos. O agrupamento realizado permite que sejam identificados comportamentos típicos de utilização dos recursos. Sempre que recursos forem requisitados na grade, o modo de operação dos recursos no momento da solicitação é mapeado a um dos padrões, assumindo-se que esse comportamento irá se repetir.

5.1 InteGrade

Segundo descrição de projeto [24], o módulo LUPA (*Local User Pattern Analyser*) do InteGrade será o responsável por realizar análises dos padrões de uso das máquinas pelos usuários locais. As técnicas de aprendizado através da análise de conglomerados serão úteis para a identificação desses padrões.

O processo de predição de ociosidade de recursos aqui descrito, por sua vez, poderá ajudar o serviço de escalonamento de tarefas na grade a agendar a execução de aplicações de forma otimizada, sem comprometer a utilização da máquina pelos usuários locais.

Os algoritmos aqui propostos poderão ser incorporados à arquitetura do projeto InteGrade, através do módulo LUPA.

5.1.1 Configuração Automática

Há três etapas para a implementação do reconhecimento de padrões de utilização de recursos computacionais: coleta de dados, identificação dos modos padrões de operação e reconhecimento do modo corrente.

Inicialmente, é necessário que seja realizada coleta de dados históricos sobre a utilização das máquinas. Essa coleta é realizada através do monitoramento de utilização dos recursos.

Realizada a coleta, dá-se lugar ao treinamento propriamente dito. Diversos métodos podem ser testados, de forma análoga aos ensaios aqui descritos. Após os testes, são identificados os comportamentos padrões de utilização das máquinas.

O módulo LUPA deverá armazenar esses modelos de comportamentos e, sempre que houver requisição por recursos da máquina que o módulo gerencia, é reconhecido o modo de operação corrente e dada resposta quanto à disponibilidade dos recursos nos níveis solicitados.

As medidas de desempenho do sistema de predição devem continuar sendo realizadas com o funcionamento real do reconhecimento de padrões. Essas novas medidas devem ser comparadas com os resultados das simulações realizadas. O acompanhamento contínuo dessas medidas possibilitará, inclusive, a realização do controle estatístico da qualidade [25] para medir e aprimorar a qualidade do processo de reconhecimento de padrões.

A coleta de dados sobre a utilização dos recursos pode ser mantida, possibilitando que se realizem treinamentos periódicos *(i)* a intervalos pré-determinados (a cada mês por exemplo), *(ii)* quando os indicadores de desempenho do algoritmo de predição piorarem ou *(iii)* mesmo sob demanda dos usuários finais das máquinas. Esses treinamentos periódicos possibilitam ajustes na definição dos padrões de comportamento.

No momento em que uma máquina é inserida na grade oportunista criada pelo InteGrade, possivelmente ainda não foi realizado nenhum monitoramento de utilização histórica de seus recursos. Por outro lado, nos ensaios realizados, a predição baseada em *clustering* foi comparada a duas outras técnicas mais simples de predição: *(i)* assumir que o nível de utilização dos recursos no momento da requisição irá manter-se o mesmo durante o período de execução da aplicação do InteGrade ou *(ii)* estimar que o comportamento num futuro próximo reproduzirá a média apresentada nas 24 horas anteriores ao momento da requisição.

Os resultados obtidos com os ensaios realizados apontam que, enquanto não há dados suficientes para definição dos comportamentos prototípicos, pode-se realizar a predição de ociosidade com base em uma dessas duas técnicas mais simples. Isso poderá ser feito paralelamente à coleta inicial dos dados. Após um período de coleta, que pode variar de acordo com o perfil de uso da máquina, passa-se a utilizar os *clusters* identificados. O período mínimo de coleta, empiricamente observado, foi de aproximadamente 3 meses de informações sobre o uso da máquina.

5.2 Conclusões Gerais

O problema de reconhecimento de padrões de uso dos recursos computacionais mostrou-se tratável pelas técnicas de *clustering*. Os resultados obtidos mostraram que há algumas variações de desempenho das técnicas propostas mas, de maneira geral, a predição baseada em *clusters* pode agregar valor aos propósitos do InteGrade.

Como sugestão de trabalhos futuros, pode-se *(i)* implementar mais variações dos algoritmos e métodos apresentados (e.g. método Chameleon [26], medida de similaridade DPF [27]), enriquecendo o processo de configuração dos modos de operação das máquinas, *(ii)* bem como promover a integração, ao módulo LUPA do InteGrade, dos algoritmos implementados.

Apêndice A

Implementação realizada

A.1 Descrição da arquitetura do software implementado

Para implementar o reconhecimento de padrões proposto neste trabalho foi desenvolvido um módulo de software, na linguagem Java. O software implementado poderá ser reaproveitado ou melhorado, podendo inclusive ser integrado ou adaptado (linguagem e/ou padrões) ao software que implementa a grade oportunista do projeto InteGrade.

O diagrama de classes da Figura A.1 oferece uma breve descrição da arquitetura utilizada na implementação do software.

Considerando as diversas possibilidades de implementação das técnicas de *clustering*, a implementação do software utilizou fortemente o paradigma de orientação a objetos. A arquitetura proposta possibilitou a criação de um arcabouço em módulos que podem facilmente ser estendidos. Isso permite implementar os diversos aspectos das técnicas de *clustering*, possibilitando a inclusão de novos modelos em eventuais trabalhos futuros.

A classe **DataVector** é a representação de um objeto no contexto da análise de conglomerados. Ela possui atributos que representam todas as informações de utilização de recursos computacionais que caracterizam um objeto.

Objetos do tipo **Cluster** representam os *clusters* formados. O comportamento prototípico definido por um *cluster* é implementado por um objeto do tipo **DataVector**, atributo do objeto do tipo **Cluster** respectivo.

A classe **ClusteringManager** é responsável pelo controle da aplicação. Ela coordena o processo de partição dos dados originais, recebendo objetos do tipo **DataVector** e submetendo-os ao processo de partição em *clusters*. Além disso, funciona como ponto de acesso aos *clusters* formados e implementa a identificação de protótipos para os objetos que são apresentados ao algoritmo de predição.

A classe **ClusteringFactory** é uma fábrica de objetos. Ela constrói objetos de classes concretas que implementam alguma das classes abstratas ***Parser***, ***Algorithm***, ***DataDistance***, ***ClusterDataDistance*** e ***RepresentativeAlgo-***

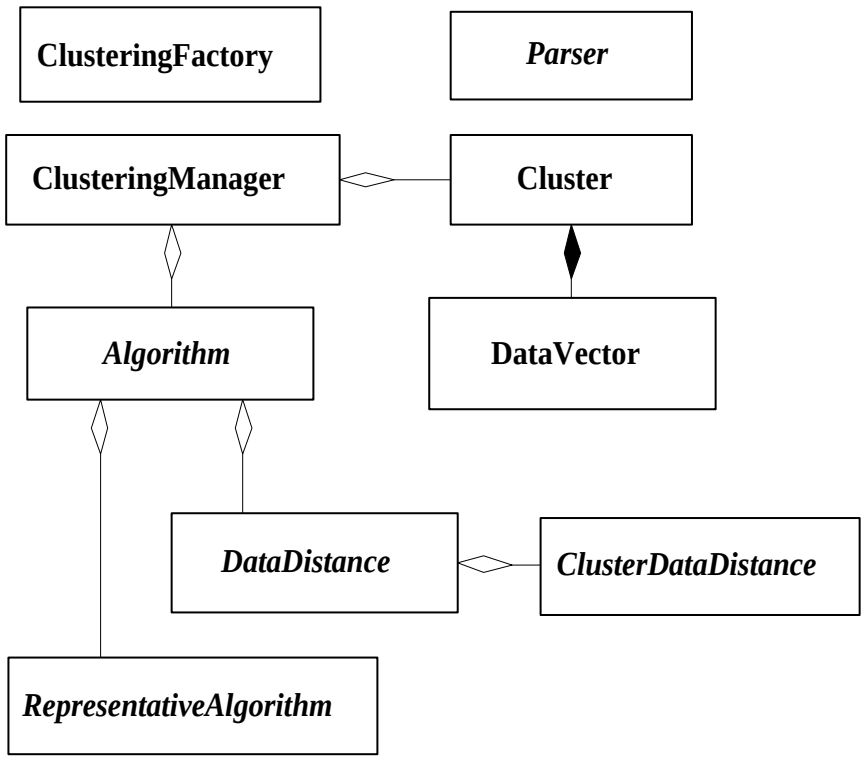

Figura A.1: Diagrama UML com as classes fundamentais do software implementado.

rithm.

A classe abstrata ***Parser*** possui a interface necessária à construção da massa de dados que será submetida ao algoritmo *clustering*. Suas subclasses realizam o tratamento de dados históricos obtidos a partir do monitoramento de utilização de um determinado recurso computacional.

O algoritmo de *clustering* propriamente dito é implementado por objetos do tipo ***Algorithm***. Cada técnica existente deve ser implementada por um subclasse apropriada.

As medidas de similaridade entre vetores de dados e as medidas de similaridade entre conjuntos de vetores de dados são implementadas, respectivamente, por subclasses das classes abstratas ***DataDistance*** e ***ClusterDataDistance***. Novamente, cada métrica existente é implementada por uma subclasse específica.

Finalmente, os diversos modos de se calcular o objeto representativo de um conglomerados são implementados por subclasses da classe abstrata ***RepresentativeAlgorithm***.

A.2 Ambiente de execução dos ensaios

O software implementado pode ser obtido na Internet, através da página:

- http://www.ime.usp.br/~germanob/dissertacao/.

Também é disponibilizado o arquivo com as configurações de um projeto do software Eclipse 3.0 [28], utilizado para o desenvolvimento do código implementado.

Na página citada anteriormente, pode-se obter também um exemplo do arquivo de configuração utilizado para a definição do tipo de simulação que será realizada pelo software implementado. Esse arquivo serve de entrada para o software e contém um conjunto de pares chave e valor, definindo cada uma das técnicas que será implementada na simulação. A fábrica de objetos descrita na seção anterior utiliza os parâmetros desse arquivo para construir os objetos necessários para a execução da simulação escolhida.

Nesta mesma página, são disponibilizados todos os arquivos que foram utilizados para realizar as simulações deste trabalho. Cada tipo de simulação é descrito por um arquivo de configuração distinto.

Apêndice B

Resultados das Simulações

Este apêndice traz os resultados mais detalhados das simulações realizadas. Esses resultados são apresentados em forma de tabela e são descritos pelos índices de acertos e pelos índices de desempenho relativo discutidos ao longo deste trabalho.

Tabela B.1: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e ponto médio como medida de similaridade entre os *clusters*.

		Memória Física – Ponto Médio – Análise do Caso Médio							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ
oncoto	*5*	0.9601	0.0201	0.9904	0.0078	0.9693	0.0168	0.9954	0.0053
oncoto	*10*	0.9657	0.0162	0.9908	0.0085	0.9739	0.0122	0.9956	0.0055
gsd	*5*	0.9939	0.0067	0.9978	0.0021	0.9965	0.0057	0.9996	0.0010
gsd	*10*	0.9916	0.0073	0.9970	0.0030	0.9950	0.0063	0.9994	0.0014
lingcomp	*5*	0.9379	0.0212	0.9881	0.0077	0.9485	0.0208	0.9939	0.0058
lingcomp	*10*	0.9505	0.0160	0.9879	0.0073	0.9596	0.0140	0.9952	0.0044
kama	*5*	0.8328	0.0250	0.9525	0.0160	0.8681	0.0221	0.9820	0.0075
kama	*10*	0.8527	0.0259	0.9540	0.0169	0.8910	0.0214	0.9821	0.0087

Tabela B.2: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e ponto médio como medida de similaridade entre os *clusters*.

		Memória Física – Ponto Médio – Análise Prática do Pior Caso							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ
oncoto	*5*	0.8986	0.0439	0.9688	0.0197	0.9146	0.0403	0.9829	0.0139
oncoto	*10*	0.9009	0.0389	0.9734	0.0171	0.9191	0.0319	0.9835	0.0140
gsd	*5*	0.9751	0.0179	0.9844	0.0104	0.9891	0.0122	0.9953	0.0077
gsd	*10*	0.9687	0.0196	0.9859	0.0128	0.9875	0.0131	0.9962	0.0062
lingcomp	*5*	0.6655	0.0789	0.9290	0.0308	0.7216	0.0747	0.9711	0.0193
lingcomp	*10*	0.7019	0.0719	0.9407	0.0307	0.7531	0.0706	0.9701	0.0229
kama	*5*	0.5657	0.0752	0.8592	0.0395	0.6046	0.0813	0.9352	0.0287
kama	*10*	0.5833	0.0864	0.8581	0.0406	0.6433	0.0920	0.9396	0.0232

Tabela B.3: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e ligação simples como medida de similaridade entre os *clusters*.

		Memória Física – Ligação Simples – Análise do Caso Médio							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ
oncoto	*5*	0.9594	0.0194	0.9893	0.0095	0.9677	0.0157	0.9944	0.0064
oncoto	*10*	0.9615	0.0177	0.9918	0.0087	0.9714	0.0132	0.9958	0.0052
gsd	*5*	0.9890	0.0092	0.9968	0.0034	0.9930	0.0079	0.9993	0.0017
gsd	*10*	0.9907	0.0081	0.9969	0.0028	0.9941	0.0074	0.9993	0.0016
lingcomp	*5*	0.9289	0.0231	0.9874	0.0081	0.9438	0.0204	0.9939	0.0071
lingcomp	*10*	0.9440	0.0206	0.9886	0.0084	0.9554	0.0182	0.9959	0.0051
kama	*5*	0.8141	0.0286	0.9543	0.0157	0.8575	0.0256	0.9818	0.0088
kama	*10*	0.8419	0.0254	0.9526	0.0170	0.8894	0.0198	0.9819	0.0093

Tabela B.4: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e ligação simples como medida de similaridade entre os *clusters*.

		Memória Física – Ligação Simples – Análise Prática do Pior Caso							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ
oncoto	*5*	0.8862	0.0430	0.9714	0.0199	0.9037	0.0411	0.9822	0.0139
oncoto	*10*	0.8883	0.0482	0.9701	0.0196	0.9150	0.0396	0.9811	0.0153
gsd	*5*	0.9726	0.0194	0.9824	0.0121	0.9897	0.0124	0.9957	0.0065
gsd	*10*	0.9729	0.0199	0.9830	0.0126	0.9861	0.0139	0.9954	0.0074
lingcomp	*5*	0.6249	0.0855	0.9379	0.0303	0.7124	0.0763	0.9715	0.0168
lingcomp	*10*	0.7262	0.0676	0.9498	0.0250	0.7888	0.0585	0.9756	0.0137
kama	*5*	0.5271	0.0810	0.8574	0.0335	0.5775	0.0819	0.9380	0.0229
kama	*10*	0.6118	0.0735	0.8586	0.0401	0.7276	0.0587	0.9409	0.0224

Tabela B.5: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e ligação completa como medida de similaridade entre os *clusters*.

		Memória Física – Ligação Completa – Análise do Caso Médio							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ
oncoto	*5*	0.9567	0.0194	0.9902	0.0082	0.9659	0.0171	0.9955	0.0051
oncoto	*10*	0.9632	0.0170	0.9889	0.0103	0.9732	0.0116	0.9959	0.0040
gsd	*5*	0.9930	0.0073	0.9972	0.0031	0.9959	0.0062	0.9991	0.0021
gsd	*10*	0.9916	0.0073	0.9972	0.0031	0.9955	0.0059	0.9994	0.0017
lingcomp	*5*	0.9418	0.0191	0.9838	0.0083	0.9522	0.0170	0.9924	0.0063
lingcomp	*10*	0.9521	0.0153	0.9897	0.0056	0.9617	0.0136	0.9963	0.0029
kama	*5*	0.8452	0.0260	0.9530	0.0153	0.8843	0.0227	0.9823	0.0094
kama	*10*	0.8693	0.0272	0.9537	0.0171	0.9085	0.0216	0.9791	0.0125

Tabela B.6: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e ligação completa como medida de similaridade entre os *clusters*.

		Memória Física – Ligação Completa – Análise Prática do Pior Caso							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ
oncoto	*5*	0.8869	0.0505	0.9697	0.0191	0.9058	0.0459	0.9814	0.0148
oncoto	*10*	0.9029	0.0413	0.9716	0.0195	0.9225	0.0345	0.9828	0.0150
gsd	*5*	0.9743	0.0197	0.9881	0.0103	0.9887	0.0144	0.9970	0.0052
gsd	*10*	0.9696	0.0192	0.9861	0.0107	0.9864	0.0147	0.9969	0.0055
lingcomp	*5*	0.6735	0.0701	0.9304	0.0291	0.7239	0.0666	0.9661	0.0186
lingcomp	*10*	0.6895	0.0690	0.9400	0.0264	0.7477	0.0658	0.9738	0.0171
kama	*5*	0.5752	0.0879	0.8537	0.0410	0.6474	0.0856	0.9358	0.0249
kama	*10*	0.6156	0.0777	0.8591	0.0380	0.7002	0.0714	0.9379	0.0208

Tabela B.7: Predição de ociosidade de memória física, utilizando algoritmo seqüencial com duas fases.

		Memória Física – Algoritmo Seqüencial – Análise Prática do Pior Caso							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ
oncoto	*5*	0.9042	0.0386	0.9680	0.0190	0.9197	0.0355	0.9814	0.0136
oncoto	*N/D*	0.9172	0.0392	0.9723	0.0161	0.9308	0.0350	0.9839	0.0127
gsd	*5*	0.9676	0.0236	0.9840	0.0133	0.9860	0.0143	0.9960	0.0080
gsd	*N/D*	0.9686	0.0206	0.9855	0.0121	0.9839	0.0149	0.9972	0.0051
lingcomp	*5*	0.6785	0.0789	0.9451	0.0299	0.7295	0.0734	0.9730	0.0196
lingcomp	*N/D*	0.7049	0.0733	0.9529	0.0248	0.7556	0.0650	0.9788	0.0167
kama	*5*	0.6160	0.0695	0.8573	0.0389	0.7084	0.0575	0.9370	0.0239
kama	*N/D*	0.6215	0.0757	0.8519	0.0392	0.7178	0.0633	0.9328	0.0243

Tabela B.8: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e o método de Ward como medida de similaridade entre os *clusters*.

		Memória Física – Método de Ward – Análise do Caso Médio							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ
oncoto	*5*	0.9579	0.0194	0.9908	0.0085	0.9687	0.0154	0.9954	0.0052
oncoto	*10*	0.9653	0.0179	0.9905	0.0092	0.9752	0.0132	0.9957	0.0049
gsd	*5*	0.9914	0.0081	0.9961	0.0037	0.9947	0.0071	0.9990	0.0019
gsd	*10*	0.9923	0.0074	0.9971	0.0028	0.9955	0.0068	0.9993	0.0014
lingcomp	*5*	0.9381	0.0194	0.9868	0.0082	0.9475	0.0175	0.9961	0.0035
lingcomp	*10*	0.9483	0.0172	0.9887	0.0079	0.9585	0.0155	0.9961	0.0036
kama	*5*	0.8616	0.0246	0.9537	0.0147	0.8976	0.0207	0.9824	0.0086
kama	*10*	0.8761	0.0215	0.9529	0.0146	0.9138	0.0166	0.9835	0.0080

Tabela B.9: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e o método de Ward como medida de similaridade entre os *clusters*.

		Memória Física – Método de Ward – Análise Prática do Pior Caso							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ
oncoto	*5*	0.8989	0.0392	0.9713	0.0193	0.9166	0.0373	0.9830	0.0144
oncoto	*10*	0.9043	0.0362	0.9708	0.0199	0.9231	0.0316	0.9803	0.0172
gsd	*5*	0.9717	0.0169	0.9848	0.0116	0.9867	0.0129	0.9959	0.0079
gsd	*10*	0.9725	0.0183	0.9842	0.0130	0.9883	0.0123	0.9967	0.0065
lingcomp	*5*	0.6965	0.0798	0.9367	0.0308	0.7395	0.0767	0.9694	0.0201
lingcomp	*10*	0.7082	0.0800	0.9469	0.0282	0.7650	0.0761	0.9762	0.0183
kama	*5*	0.6675	0.0636	0.8593	0.0408	0.7388	0.0661	0.9427	0.0214
kama	*10*	0.6914	0.0635	0.8592	0.0345	0.7739	0.0532	0.9392	0.0197

Tabela B.10: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e ponto médio como medida de similaridade entre os *clusters*.

		CPU – Ponto Médio – Análise do Caso Médio							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ
oncoto	*5*	0.9504	0.0291	0.9625	0.0199	0.9523	0.0278	0.9709	0.0154
oncoto	*10*	0.9526	0.0261	0.9663	0.0211	0.9551	0.0250	0.9723	0.0169
gsd	*5*	0.8198	0.0176	0.8680	0.0118	0.8884	0.0167	0.9284	0.0092
gsd	*10*	0.8381	0.0161	0.8674	0.0129	0.9104	0.0154	0.9315	0.0106
lingcomp	*5*	0.9476	0.0286	0.9695	0.0146	0.9616	0.0232	0.9806	0.0115
lingcomp	*10*	0.9541	0.0242	0.9721	0.0162	0.9690	0.0200	0.9834	0.0118
kama	*5*	0.7155	0.0356	0.8584	0.0192	0.7988	0.0282	0.9221	0.0140
kama	*10*	0.7661	0.0311	0.8651	0.0215	0.8391	0.0250	0.9245	0.0165

Tabela B.11: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e ponto médio como medida de similaridade entre os *clusters*.

		CPU – Ponto Médio – Análise Prática do Pior Caso							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ
oncoto	*5*	0.9207	0.0344	0.9300	0.0347	0.9227	0.0338	0.9415	0.0301
oncoto	*10*	0.9238	0.0387	0.9377	0.0270	0.9256	0.0376	0.9508	0.0232
gsd	*5*	0.3504	0.0667	0.5206	0.0523	0.5766	0.0524	0.7531	0.0393
gsd	*10*	0.3947	0.0553	0.5169	0.0563	0.6251	0.0554	0.7684	0.0414
lingcomp	*5*	0.9114	0.0369	0.9358	0.0267	0.9220	0.0352	0.9524	0.0232
lingcomp	*10*	0.9117	0.0298	0.9348	0.0240	0.9257	0.0291	0.9512	0.0202
kama	*5*	0.4814	0.0837	0.7663	0.0382	0.6594	0.0569	0.8688	0.0268
kama	*10*	0.5986	0.0572	0.7614	0.0340	0.7382	0.0390	0.8673	0.0291

Tabela B.12: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e ligação simples como medida de similaridade entre os *clusters*.

		CPU – Ligação Simples – Análise do Caso Médio							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Φ_{medio}	σ_{Φ}	Φ_{medio}	σ_{Φ}	Φ_{medio}	σ_{Φ}	Φ_{medio}	σ_{Φ}
oncoto	*5*	0.9054	0.0395	0.9647	0.0207	0.9096	0.0376	0.9714	0.0182
oncoto	*10*	0.9179	0.0392	0.9638	0.0202	0.9255	0.0368	0.9713	0.0165
gsd	*5*	0.8147	0.0169	0.8690	0.0134	0.8848	0.0160	0.9239	0.0095
gsd	*10*	0.8310	0.0173	0.8689	0.0127	0.8948	0.0172	0.9269	0.0108
lingcomp	*5*	0.9262	0.0338	0.9693	0.0171	0.9272	0.0336	0.9808	0.0122
lingcomp	*10*	0.9151	0.0390	0.9688	0.0153	0.9168	0.0392	0.9824	0.0112
kama	*5*	0.6590	0.0235	0.8628	0.0221	0.7566	0.0214	0.9250	0.0142
kama	*10*	0.6639	0.0388	0.8625	0.0238	0.7739	0.0307	0.9229	0.0173

Tabela B.13: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e ligação simples como medida de similaridade entre os *clusters*.

		CPU – Ligação Simples – Análise Prática do Pior Caso							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ
oncoto	*5*	0.8625	0.0630	0.9339	0.0314	0.8736	0.0564	0.9449	0.0267
oncoto	*10*	0.8814	0.0440	0.9324	0.0333	0.8946	0.0405	0.9444	0.0261
gsd	*5*	0.4072	0.0660	0.5142	0.0550	0.6054	0.0478	0.7463	0.0410
gsd	*10*	0.4477	0.0499	0.5175	0.0567	0.6035	0.0458	0.7504	0.0407
lingcomp	*5*	0.8690	0.0548	0.9388	0.0267	0.8710	0.0544	0.9529	0.0215
lingcomp	*10*	0.8796	0.0536	0.9377	0.0255	0.8877	0.0531	0.9584	0.0182
kama	*5*	0.3911	0.0951	0.7621	0.0358	0.5858	0.0776	0.8669	0.0292
kama	*10*	0.5145	0.0640	0.7624	0.0368	0.6813	0.0478	0.8657	0.0289

Tabela B.14: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e ligação completa como medida de similaridade entre os *clusters*.

		CPU – Ligação Completa – Análise do Caso Médio							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ
oncoto	*5*	0.9530	0.0274	0.9615	0.0240	0.9547	0.0262	0.9695	0.0182
oncoto	*10*	0.9529	0.0260	0.9660	0.0212	0.9556	0.0254	0.9728	0.0176
gsd	*5*	0.8238	0.0164	0.8664	0.0143	0.8931	0.0189	0.9323	0.0101
gsd	*10*	0.8415	0.0160	0.8684	0.0127	0.9057	0.0169	0.9401	0.0102
lingcomp	*5*	0.9502	0.0267	0.9681	0.0162	0.9649	0.0229	0.9807	0.0107
lingcomp	*10*	0.9524	0.0251	0.9720	0.0158	0.9693	0.0197	0.9827	0.0117
kama	*5*	0.7489	0.0329	0.8617	0.0210	0.8219	0.0297	0.9226	0.0150
kama	*10*	0.7747	0.0278	0.8647	0.0231	0.8492	0.0228	0.9248	0.0155

Tabela B.15: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e ligação completa como medida de similaridade entre os *clusters*.

		CPU – Ligação Completa – Análise Prática do Pior Caso							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ
oncoto	*5*	0.9192	0.0386	0.9309	0.0324	0.9213	0.0376	0.9428	0.0280
oncoto	*10*	0.9205	0.0426	0.9298	0.0316	0.9228	0.0413	0.9434	0.0265
gsd	*5*	0.3497	0.0719	0.5068	0.0486	0.5934	0.0539	0.7597	0.0375
gsd	*10*	0.4311	0.0562	0.5279	0.0506	0.6120	0.0574	0.7874	0.0332
lingcomp	*5*	0.9055	0.0344	0.9374	0.0269	0.9164	0.0329	0.9531	0.0240
lingcomp	*10*	0.9192	0.0346	0.9337	0.0304	0.9327	0.0306	0.9517	0.0216
kama	*5*	0.5771	0.0735	0.7645	0.0367	0.7264	0.0466	0.8685	0.0306
kama	*10*	0.6126	0.0629	0.7609	0.0362	0.7606	0.0390	0.8669	0.0287

Tabela B.16: Predição de ociosidade de CPU, utilizando algoritmo seqüencial com duas fases.

		CPU – Algoritmo Seqüencial – Análise Prática do Pior Caso							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ
oncoto	*5*	0.9190	0.0396	0.9342	0.0299	0.9212	0.0392	0.9483	0.0244
oncoto	*N/D*	0.9285	0.0382	0.9364	0.0326	0.9397	0.0342	0.9497	0.0274
gsd	*5*	0.3242	0.0588	0.5198	0.0521	0.6205	0.0680	0.7597	0.0407
gsd	*N/D*	0.3683	0.0837	0.5279	0.0513	0.6235	0.0739	0.7718	0.0394
lingcomp	*5*	0.9146	0.0311	0.9404	0.0249	0.9263	0.0316	0.9561	0.0207
lingcomp	*N/D*	0.9270	0.0286	0.9366	0.0255	0.9452	0.0220	0.9515	0.0204
kama	*5*	0.6116	0.0527	0.7622	0.0365	0.7384	0.0350	0.8673	0.0287
kama	*N/D*	0.4009	0.0972	0.7644	0.0363	0.5228	0.0958	0.8653	0.0298

Tabela B.17: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e o método de Ward como medida de similaridade entre os *clusters*.

		CPU – Método de Ward – Análise do Caso Médio							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ	Φ_{medio}	σ_Φ
oncoto	*5*	0.9538	0.0263	0.9614	0.0240	0.9569	0.0234	0.9686	0.0204
oncoto	*10*	0.9564	0.0248	0.9639	0.0197	0.9587	0.0232	0.9726	0.0166
gsd	*5*	0.8310	0.0147	0.8673	0.0112	0.9010	0.0162	0.9350	0.0095
gsd	*10*	0.8406	0.0155	0.8670	0.0120	0.9140	0.0137	0.9402	0.0088
lingcomp	*5*	0.9462	0.0215	0.9696	0.0156	0.9597	0.0173	0.9814	0.0119
lingcomp	*10*	0.9503	0.0230	0.9687	0.0163	0.9663	0.0189	0.9811	0.0117
kama	*5*	0.7640	0.0282	0.8570	0.0220	0.8325	0.0230	0.9253	0.0143
kama	*10*	0.7857	0.0290	0.8581	0.0228	0.8596	0.0234	0.9272	0.0157

Tabela B.18: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e o método de Ward como medida de similaridade entre os *clusters*.

		CPU – Método de Ward – Análise Prática do Pior Caso							
		Caixa Branca				*Caixa Preta*			
	Clusters	*Medidas*		*Variações*		*Medidas*		*Variações*	
		Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ	Ψ_{medio}	σ_Ψ
oncoto	*5*	0.9203	0.0390	0.9305	0.0320	0.9219	0.0382	0.9440	0.0271
oncoto	*10*	0.9142	0.0351	0.9266	0.0357	0.9193	0.0343	0.9413	0.0269
gsd	*5*	0.4405	0.0564	0.5137	0.0544	0.6924	0.0601	0.7760	0.0356
gsd	*10*	0.4413	0.0653	0.5265	0.0503	0.7253	0.0441	0.7939	0.0371
lingcomp	*5*	0.9057	0.0317	0.9346	0.0295	0.9155	0.0292	0.9523	0.0219
lingcomp	*10*	0.9027	0.0310	0.9335	0.0248	0.9201	0.0271	0.9528	0.0212
kama	*5*	0.6104	0.0551	0.7560	0.0390	0.7373	0.0419	0.8682	0.0235
kama	*10*	0.6584	0.0456	0.7652	0.0366	0.7865	0.0341	0.8773	0.0264

Tabela B.19: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e ponto médio como medida de similaridade entre os *clusters*.

		Memória Física – Ponto Médio – Desempenho Relativo (Caixa Preta)											
		Análise do Caso Médio						*Análise Prática do Pior Caso*					
	Clusters	*Clustering vs 24 horas*			*Clustering vs Último*			*Clustering vs 24 horas*			*Clustering vs Último*		
		Φ^{I}	$\sigma_{\Phi^{I}}$	f_{I}	Φ^{II}	$\sigma_{\Phi^{II}}$	f_{II}	Ψ^{I}	$\sigma_{\Psi^{I}}$	g_{I}	Ψ^{II}	$\sigma_{\Psi^{II}}$	g_{II}
oncoto	*5*	0.0280	0.0144	0.9736	0.0006	0.0016	0.6477	0.0919	0.0426	0.9845	0.0014	0.0058	0.5919
oncoto	*10*	0.0271	0.0122	0.9869	0.0003	0.0013	0.5900	0.0792	0.0411	0.9731	0.0005	0.0060	0.5324
gsd	*5*	0.0037	0.0045	0.7923	0.0003	0.0012	0.5969	0.0151	0.0173	0.8082	0.0004	0.0034	0.5516
gsd	*10*	0.0056	0.0068	0.7936	0.0002	0.0008	0.5908	0.0150	0.0144	0.8505	0.0005	0.0026	0.5821
lingcomp	*5*	0.0463	0.0200	0.9898	-0.0008	0.0018	0.3279	0.1701	0.0588	0.9981	-0.0065	0.0186	0.3637
lingcomp	*10*	0.0487	0.0205	0.9911	-0.0004	0.0031	0.4459	0.1688	0.0615	0.9970	-0.0070	0.0194	0.3586
kama	*5*	0.0878	0.0205	1.0000	0.0035	0.0032	0.8647	0.1985	0.0624	0.9993	0.0096	0.0144	0.7478
kama	*10*	0.0835	0.0212	1.0000	0.0036	0.0027	0.9141	0.2191	0.0666	0.9995	0.0094	0.0125	0.7734

Tabela B.20: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e ligação simples como medida de similaridade entre os *clusters*.

		Memória Física – Ligação Simples – Desempenho Relativo (Caixa Preta)											
		Análise do Caso Médio						*Análise Prática do Pior Caso*					
	Clusters	*Clustering vs 24 horas*			*Clustering vs Último*			*Clustering vs 24 horas*			*Clustering vs Último*		
		Φ^{I}	$\sigma_{\Phi^{I}}$	f_{I}	Φ^{II}	$\sigma_{\Phi^{II}}$	f_{II}	Ψ^{I}	$\sigma_{\Psi^{I}}$	g_{I}	Ψ^{II}	$\sigma_{\Psi^{II}}$	g_{II}
oncoto	*5*	0.0274	0.0123	0.9867	0.0003	0.0012	0.6101	0.0851	0.0418	0.9791	0.0012	0.0051	0.5953
oncoto	*10*	0.0295	0.0141	0.9815	0.0006	0.0013	0.6710	0.0867	0.0425	0.9793	0.0011	0.0080	0.5527
gsd	*5*	0.0035	0.0040	0.8102	0.0001	0.0003	0.6052	0.0146	0.0155	0.8262	0.0007	0.0028	0.6007
gsd	*10*	0.0044	0.0048	0.8197	0.0001	0.0004	0.6013	0.0160	0.0146	0.8637	0.0007	0.0033	0.5838
lingcomp	*5*	0.0474	0.0189	0.9939	0.0000	0.0030	0.4937	0.1808	0.0625	0.9981	-0.0043	0.0137	0.3754
lingcomp	*10*	0.0492	0.0208	0.9910	0.0007	0.0030	0.5935	0.1900	0.0636	0.9986	-0.0013	0.0080	0.4361
kama	*5*	0.0875	0.0205	1.0000	0.0033	0.0031	0.8616	0.2045	0.0651	0.9992	0.0086	0.0127	0.7508
kama	*10*	0.0888	0.0205	1.0000	0.0041	0.0037	0.8699	0.2104	0.0602	0.9998	0.0088	0.0130	0.7503

Tabela B.21: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e ligação completa como medida de similaridade entre os *clusters*.

		Memória Física – Ligação Completa – Desempenho Relativo (Caixa Preta)											
		Análise do Caso Médio						*Análise Prática do Pior Caso*					
	Clusters	*Clustering vs 24 horas*			*Clustering vs Último*			*Clustering vs 24 horas*			*Clustering vs Último*		
		Φ^{I}	$\sigma_{\Phi^{I}}$	f_{I}	Φ^{II}	$\sigma_{\Phi^{II}}$	f_{II}	Ψ^{I}	$\sigma_{\Psi^{I}}$	g_{I}	Ψ^{II}	$\sigma_{\Psi^{II}}$	g_{II}
oncoto	*5*	0.0306	0.0163	0.9701	0.0006	0.0015	0.6425	0.0794	0.0442	0.9637	0.0017	0.0057	0.6168
oncoto	*10*	0.0330	0.0143	0.9893	0.0006	0.0022	0.6121	0.0778	0.0331	0.9906	0.0010	0.0063	0.5653
gsd	*5*	0.0045	0.0055	0.7942	0.0002	0.0007	0.5866	0.0157	0.0134	0.8791	0.0004	0.0022	0.5801
gsd	*10*	0.0039	0.0052	0.7740	0.0001	0.0007	0.5612	0.0136	0.0132	0.8491	-0.0003	0.0032	0.4642
lingcomp	*5*	0.0486	0.0223	0.9852	-0.0011	0.0024	0.3208	0.1618	0.0643	0.9941	-0.0109	0.0168	0.2577
lingcomp	*10*	0.0451	0.0197	0.9890	0.0005	0.0029	0.5614	0.1817	0.0617	0.9984	-0.0037	0.0167	0.4119
kama	*5*	0.0870	0.0214	1.0000	0.0035	0.0029	0.8888	0.2021	0.0560	0.9998	0.0111	0.0138	0.7912
kama	*10*	0.0823	0.0225	0.9999	0.0032	0.0029	0.8696	0.2065	0.0602	0.9997	0.0121	0.0144	0.8003

Tabela B.22: Predição de ociosidade de memória física, utilizando algoritmo seqüencial com duas fases.

		Memória Física – Algoritmo Seqüencial *Análise Prática do Pior Caso (Caixa Preta)*					
	Clusters	*Clustering vs 24 horas*			*Clustering vs Último*		
		Ψ^{I}	$\sigma_{\Psi^{I}}$	g_{I}	Ψ^{II}	$\sigma_{\Psi^{II}}$	g_{II}
oncoto	*5*	0.0833	0.0462	0.9643	0.0012	0.0077	0.5636
oncoto	*N/D*	0.0813	0.0361	0.9877	0.0019	0.0058	0.6265
gsd	*5*	0.0112	0.0123	0.8182	0.0007	0.0033	0.5861
gsd	*N/D*	0.0125	0.0106	0.8816	0.0005	0.0031	0.5585
lingcomp	*5*	0.1732	0.0570	0.9988	-0.0030	0.0128	0.4083
lingcomp	*N/D*	0.1832	0.0702	0.9955	0.0023	0.0135	0.5682
kama	*5*	0.2101	0.0648	0.9994	0.0101	0.0167	0.7265
kama	*N/D*	0.2045	0.0682	0.9986	0.0104	0.0135	0.7793

Tabela B.23: Predição de ociosidade de memória física, utilizando algoritmo hierárquico aglomerativo e o método de Ward como medida de similaridade entre os *clusters*.

		Memória Física – Método de Ward – Desempenho Relativo (Caixa Preta)											
		Análise do Caso Médio						*Análise Prática do Pior Caso*					
	Clusters	*Clustering vs 24 horas*			*Clustering vs Último*			*Clustering vs 24 horas*			*Clustering vs Último*		
		Φ^{I}	$\sigma_{\Phi^{I}}$	f_{I}	Φ^{II}	$\sigma_{\Phi^{II}}$	f_{II}	Ψ^{I}	$\sigma_{\Psi^{I}}$	g_{I}	Ψ^{II}	$\sigma_{\Psi^{II}}$	g_{II}
oncoto	*5*	0.0250	0.0121	0.9808	0.0004	0.0013	0.6196	0.0788	0.0359	0.9858	0.0011	0.0046	0.5958
oncoto	*10*	0.0277	0.0133	0.9811	0.0003	0.0012	0.5832	0.0769	0.0402	0.9721	0.0001	0.0066	0.5039
gsd	*5*	0.0042	0.0051	0.7970	0.0001	0.0003	0.5970	0.0134	0.0139	0.8320	0.0009	0.0028	0.6249
gsd	*10*	0.0035	0.0051	0.7537	0.0001	0.0002	0.5974	0.0126	0.0132	0.8308	0.0005	0.0026	0.5741
lingcomp	*5*	0.0512	0.0181	0.9976	0.0004	0.0033	0.5495	0.1651	0.0688	0.9918	-0.0050	0.0149	0.3697
lingcomp	*10*	0.0508	0.0199	0.9947	0.0008	0.0042	0.5755	0.1728	0.0638	0.9966	-0.0066	0.0149	0.3293
kama	*5*	0.0803	0.0250	0.9993	0.0037	0.0029	0.8985	0.2130	0.0631	0.9996	0.0116	0.0143	0.7911
kama	*10*	0.0888	0.0217	1.0000	0.0042	0.0032	0.9053	0.1986	0.0623	0.9993	0.0092	0.0117	0.7843

Tabela B.24: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e ponto médio como medida de similaridade entre os *clusters*.

		CPU – Ponto Médio – Desempenho Relativo (Caixa Preta)											
		Análise do Caso Médio						*Análise Prática do Pior Caso*					
	Clusters	*Clustering vs 24 horas*			*Clustering vs Último*			*Clustering vs 24 horas*			*Clustering vs Último*		
		Φ^{I}	$\sigma_{\Phi^{I}}$	f_{I}	Φ^{II}	$\sigma_{\Phi^{II}}$	f_{II}	Ψ^{I}	$\sigma_{\Psi^{I}}$	g_{I}	Ψ^{II}	$\sigma_{\Psi^{II}}$	g_{II}
oncoto	*5*	0.0175	0.0157	0.8663	0.0005	0.0019	0.6049	0.0288	0.0263	0.8640	-0.0002	0.0019	0.4531
oncoto	*10*	0.0182	0.0168	0.8606	0.0018	0.0047	0.6482	0.0375	0.0262	0.9238	0.0003	0.0059	0.5231
gsd	*5*	0.0221	0.0147	0.9334	0.0066	0.0043	0.9385	0.0403	0.0427	0.8272	0.0279	0.0227	0.8904
gsd	*10*	0.0246	0.0134	0.9668	0.0091	0.0054	0.9548	0.0459	0.0414	0.8664	0.0379	0.0224	0.9544
lingcomp	*5*	0.0176	0.0144	0.8904	0.0001	0.0009	0.5242	0.0333	0.0267	0.8939	0.0005	0.0042	0.5514
lingcomp	*10*	0.0193	0.0158	0.8886	0.0002	0.0007	0.5898	0.0364	0.0289	0.8964	0.0001	0.0027	0.5177
kama	*5*	0.1211	0.0258	1.0000	0.0098	0.0033	0.9986	0.1742	0.0456	0.9999	0.0167	0.0111	0.9344
kama	*10*	0.1174	0.0249	1.0000	0.0099	0.0035	0.9974	0.1904	0.0422	1.0000	0.0150	0.0128	0.8782

Tabela B.25: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e ligação simples como medida de similaridade entre os *clusters*.

		CPU – Ligação Simples – Desempenho Relativo (Caixa Preta)											
		Análise do Caso Médio						*Análise Prática do Pior Caso*					
	Clusters	*Clustering vs 24 horas*			*Clustering vs Último*			*Clustering vs 24 horas*			*Clustering vs Último*		
		Φ^{I}	$\sigma_{\Phi^{I}}$	f_{I}	Φ^{II}	$\sigma_{\Phi^{II}}$	f_{II}	Ψ^{I}	$\sigma_{\Psi^{I}}$	g_{I}	Ψ^{II}	$\sigma_{\Psi^{II}}$	g_{II}
oncoto	*5*	0.0205	0.0169	0.8876	0.0008	0.0028	0.6099	0.0260	0.0254	0.8468	0.0004	0.0059	0.5296
oncoto	*10*	0.0187	0.0196	0.8301	0.0026	0.0050	0.7013	0.0303	0.0233	0.9039	0.0028	0.0080	0.6366
gsd	*5*	0.0195	0.0132	0.9299	0.0032	0.0027	0.8777	0.0325	0.0427	0.7769	0.0137	0.0120	0.8730
gsd	*10*	0.0221	0.0154	0.9239	0.0056	0.0033	0.9557	0.0389	0.0427	0.8189	0.0240	0.0162	0.9301
lingcomp	*5*	0.0190	0.0161	0.8822	0.0007	0.0022	0.6151	0.0313	0.0241	0.9028	0.0003	0.0023	0.5460
lingcomp	*10*	0.0220	0.0173	0.8985	0.0016	0.0024	0.7501	0.0437	0.0300	0.9271	0.0067	0.0100	0.7482
kama	*5*	0.1189	0.0209	1.0000	0.0109	0.0036	0.9987	0.1787	0.0402	1.0000	0.0180	0.0159	0.8712
kama	*10*	0.1153	0.0252	1.0000	0.0108	0.0043	0.9943	0.1714	0.0409	1.0000	0.0173	0.0143	0.8872

Tabela B.26: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e ligação completa como medida de similaridade entre os *clusters*.

		CPU – Ligação Completa – Desempenho Relativo (Caixa Preta)											
		Análise do Caso Médio						*Análise Prática do Pior Caso*					
	Clusters	*Clustering vs 24 horas*			*Clustering vs Último*			*Clustering vs 24 horas*			*Clustering vs Último*		
		Φ^{I}	$\sigma_{\Phi^{I}}$	f_{I}	Φ^{II}	$\sigma_{\Phi^{II}}$	f_{II}	Ψ^{I}	$\sigma_{\Psi^{I}}$	g_{I}	Ψ^{II}	$\sigma_{\Psi^{II}}$	g_{II}
oncoto	*5*	0.0202	0.0199	0.8454	0.0006	0.0021	0.6053	0.0284	0.0249	0.8733	0.0019	0.0068	0.6107
oncoto	*10*	0.0176	0.0176	0.8421	0.0024	0.0058	0.6607	0.0369	0.0316	0.8784	0.0027	0.0070	0.6522
gsd	*5*	0.0270	0.0163	0.9514	0.0102	0.0055	0.9684	0.0489	0.0331	0.9305	0.0394	0.0275	0.9238
gsd	*10*	0.0314	0.0140	0.9873	0.0165	0.0056	0.9983	0.0741	0.0323	0.9891	0.0595	0.0307	0.9737
lingcomp	*5*	0.0182	0.0158	0.8759	0.0001	0.0011	0.5343	0.0359	0.0269	0.9088	0.0004	0.0031	0.5566
lingcomp	*10*	0.0186	0.0157	0.8830	0.0001	0.0006	0.5605	0.0363	0.0291	0.8935	0.0003	0.0032	0.5410
kama	*5*	0.1144	0.0238	1.0000	0.0101	0.0038	0.9959	0.1818	0.0375	1.0000	0.0196	0.0154	0.8988
kama	*10*	0.1175	0.0271	1.0000	0.0099	0.0037	0.9961	0.1858	0.0393	1.0000	0.0154	0.0143	0.8589

Tabela B.27: Predição de ociosidade de CPU, utilizando algoritmo seqüencial com duas fases.

		CPU – Algoritmo Seqüencial *Análise Prática do Pior Caso (Caixa Preta)*					
	Clusters	*Clustering vs 24 horas*			*Clustering vs Último*		
		Ψ^{I}	$\sigma_{\Psi^{I}}$	g_{I}	Ψ^{II}	$\sigma_{\Psi^{II}}$	g_{II}
oncoto	*5*	0.0347	0.0263	0.9065	0.0019	0.0086	0.5857
oncoto	*N/D*	0.0357	0.0270	0.9069	0.0031	0.0091	0.6340
gsd	*5*	0.0450	0.0414	0.8612	0.0290	0.0217	0.9097
gsd	*N/D*	0.0562	0.0374	0.9337	0.0427	0.0301	0.9221
lingcomp	*5*	0.0335	0.0245	0.9145	0.0003	0.0022	0.5628
lingcomp	*N/D*	0.0306	0.0259	0.8819	0.0003	0.0042	0.5304
kama	*5*	0.1863	0.0409	1.0000	0.0186	0.0137	0.9117
kama	*N/D*	0.1762	0.0424	1.0000	0.0158	0.0130	0.8883

Tabela B.28: Predição de ociosidade de CPU, utilizando algoritmo hierárquico aglomerativo e o método de Ward como medida de similaridade entre os *clusters*.

		CPU – Método de Ward – Desempenho Relativo (Caixa Preta)											
		Análise do Caso Médio						*Análise Prática do Pior Caso*					
	Clusters	*Clustering vs 24 horas*			*Clustering vs Último*			*Clustering vs 24 horas*			*Clustering vs Último*		
		Φ^I	σ_{Φ^I}	f_{I}	Φ^{II}	$\sigma_{\Phi^{II}}$	f_{II}	Ψ^I	σ_{Ψ^I}	g_{I}	Ψ^{II}	$\sigma_{\Psi^{II}}$	g_{II}
oncoto	*5*	0.0199	0.0177	0.8703	0.0010	0.0034	0.6180	0.0321	0.0288	0.8676	-0.0003	0.0078	0.4851
oncoto	*10*	0.0240	0.0192	0.8935	0.0012	0.0026	0.6817	0.0290	0.0242	0.8840	-0.0006	0.0094	0.4728
gsd	*5*	0.0324	0.0137	0.9910	0.0132	0.0056	0.9907	0.0612	0.0415	0.9298	0.0454	0.0324	0.9194
gsd	*10*	0.0332	0.0130	0.9948	0.0207	0.0064	0.9993	0.0844	0.0382	0.9865	0.0667	0.0279	0.9916
lingcomp	*5*	0.0201	0.0170	0.8815	0.0004	0.0011	0.6391	0.0335	0.0273	0.8894	0.0003	0.0029	0.5393
lingcomp	*10*	0.0207	0.0153	0.9115	0.0003	0.0012	0.6028	0.0370	0.0286	0.9015	0.0013	0.0046	0.6149
kama	*5*	0.1210	0.0235	1.0000	0.0129	0.0053	0.9928	0.1845	0.0400	1.0000	0.0207	0.0154	0.9097
kama	*10*	0.1199	0.0267	1.0000	0.0133	0.0050	0.9963	0.1873	0.0432	1.0000	0.0236	0.0146	0.9464

Bibliografia

[1] Andrei Goldchleger, Fabio Kon & Alfredo Goldman vel Lejbman. *InteGrade: Rumo a um Sistema de Computação em Grade para Aproveitamento de Recursos Ociosos em Máquinas Compartilhadas.* MAC-IME-USP, São Paulo, 2002.

[2] *InteGrade: OO Grid Middleware.* http://integrade.incubadora.fapesp.br/. Último acesso em janeiro de 2006.

[3] Andrei Goldchleger, Fabio Kon, Alfredo Goldman vel Lejbman, Marcelo Finger & Germano Capistrano Bezerra. *InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines. Concurrency and Computation: Practice & Experience*, **16**, 449–459, 2004.

[4] Sergio Theodoridis & Konstantinos Koutroumbas. *Pattern Recognition.* Academic Press, 2003.

[5] Pat Langley. *Elements of Machine Learning.* Morgan Kaufmann Publishers, Inc., 1996.

[6] H. B. Barlow. *Unsupervised Learning.* Em G. Hinton & T. J. Sejnowski (Ed.), *Unsupervised Learning: Foundations of Neural Computation* (pp. 1–17). The MIT Press, 1999.

[7] W. J. Krzanowski & F. H. Marriot. *Multivariate Analysis Part 2: Classification, covariance structures and repeated measurements.* Arnold, 1995.

[8] R. R. Sokal. *Clustering and Classification: Background and Current Directions. In Proceedings of the Advanced Seminar on Classification and Clustering*, 1996.

[9] Brian Everitt, Sabine Landau & Morven Leese. *Cluster Analysis.* Arnold Publication, 2001.

[10] G. W. Milligan. *Clustering validation: Results and implications for applied analyses.* . Em P. Arabie, L. J. Hubert & G. De Soete (Ed.), *Clustering and Classification* (pp. 341–375). World Scientific Publishing, 1996.

[11] Anil Jain & Richard Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Inc., 1988.

[12] M. Anderberg. *Cluster Analysis for Applications.* Academic Press, Inc., 1973.

[13] A. D. Gordon. *Hierarchical Classification.* Em P. Arabie, L. J. Hubert & G. D. Soete (Ed.), *Clustering and Classification.* World Scientific Publishing, 1996.

[14] J. A. Hartigan. *Clustering Algorithms.* John Wiley & Sons, 1975.

[15] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, Inc., 1990.

[16] Joe H. Ward Jr. *Hierarchical Grouping to Optimize an Objective Function. Journal of the American Statistical Association*, **58**(301), 236–244. 1963.

[17] G. W. Milligan & M. C. Cooper. *A study of variable standardization. Journal of Classification*, **5**, 181–204. 1988.

[18] Stan Salvador & Philip Chan. *Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms.* Relatório Técnico CS-2003-18, Department of Computer Sciences, Florida Institute of Technology, 2003.

[19] Sandrine Dudoit & Jana Fridlyand. *A prediction-based resampling method for estimating the number of clusters in a dataset. Genome Biology*, **3**(7). 2002.

[20] Robert Tibshirani, Guenther Walther & Trevor Hastie. *Estimating the Number of Data Clusters via the Gap Statistic.* Relatório Técnico 208, Department of Statistics, Stanford University, 2000.

[21] Jeffrey Heer & Ed H. Chi. *Mining the Structure of User Activity using Cluster Stability. Proceedings of the Workshop on Web Analytics, SIAM Conference on Data Mining*, 2002.

[22] Robert Tibshirani, Guenther Walther, David Botstein & Patrick Brown. *Cluster validation by prediction strength.* Relatório Técnico 2001-21, Department of Biostatistics, Stanford University, 2001.

[23] *Distributed Systems Research Group.* http://gsd.ime.usp.br/. Último acesso em janeiro de 2006.

[24] *Projeto InteGrade 2. Descrição Detalhada do Projeto de Pesquisa. Chamada CTInfo/MCT/CNPq no. 011/2005.* 2005.

[25] Dorian Shainin & Peter D. Shainin. *Controle Estatístico do Processo.* Em J. M. Juran & F. M. Gryna (Ed.), *Controle da Qualidade: Métodos Estatísticos Clássicos Aplicados à Qualidade.* Makron Books do Brasil Editora Ltda, 1993.

[26] George Karypis, Eui-Hong Han & Vipin Kumar. *CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling.* IEEE Computer, **32**(8):68–75, 1999.

[27] Beltao Li, Edward Chang & Ching-Tung Wu. *DPF — A Perceptual Distance Function for Image Retrieval.* IEEE International Conference on Image Processing (ICIP), Rochester, 2002.

[28] *Eclipse downloads home.* http://www.eclipse.org/downloads/. Último acesso em janeiro de 2006.