Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

HPC-ICTM:
um Modelo de Alto Desempenho para Categorização de Áreas Geográficas

Rafael Krolow Santos Silva

Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Ciência da Computação

Orientador: Prof. Dr. César Augusto Fonticielha De Rose

Porto Alegre
2006

Dados Internacionais de Catalogação na Publicação (CIP)

S586h Silva, Rafael Krolow Santos
HPC-ICTM : um modelo de alto desempenho para categorização de áreas geográficas / Rafael Wrolow Santos Silva. – Porto Alegre, 2006.
127 f.
Diss. (Mestrado) – Fac. de Informática, PUCRS
Orientador: Prof. Dr. César Augusto Fonticiella De Rose
1. ICTM. 2. HPC-ICTM. 3. Modelo Categorizador. 4. Processamento de Informação Geográfica. 5. Geografia – Processamento de Dados. I. Título.
CDD 005.72
910.285

Ficha Catalográfica elaborada pelo Setor de Processamento Técnico da BC-PUCRS

Campus Central
Av. Ipiranga, 6681 – prédio 16 – CEP 90619-900
Porto Alegre – RS – Brasil
Fone: +55 (51) 3320-3544 – Fax: +55 (51) 3320-3548
Email: bceadm@pucrs.br
www.pucrs.br/biblioteca

Substitua esta folha pela ficha de homologacao

Dedico este trabalho aos meus pais, que me passaram todos os valores necessários para minha formação como ser humano. Exemplos puros de determinação, honestidade e respeito ao próximo.

Agradecimentos

- Ao Prof^{o}. César A. F. De Rose, pela oportunidade oferecida e pela orientação no trabalho desenvolvido.
- Ao Prof^{o}. e amigo Marilton Aguiar, pela orientação desde os tempos de graduação e pela atenção, dedicação e inúmeras oportunidades oferecidas em todos os momentos de minha vida acadêmica.
- À minha família, pelo amor incondicional e pelo constante apoio e incentivo. Em especial aos meus pais. Tenho consciência do esforço que fizeram para que eu atingisse meus objetivos. Jamais esquecerei disso e jamais deixarei de amá-los de todas as formas possíveis e imagináveis.
- À minha namorada Mônica. Fiel companheira que sempre esteve ao meu lado e suportou minhas ausências e todos os momentos de rebeldia.
- Aos amigos eternos, pelas palavras de conforto e incentivo nas horas mais complicadas e pelas horas de lazer que passamos juntos.
- Aos amigos e colegas do CPAD, pela excelente relação de trabalho e amizade. Foi um privilégio trabalhar num ambiente repleto de pessoas inteligentes e determinadas.
- À Hewlett Packard Brasil, pelo financiamento do trabalho.

Resumo

Este trabalho apresenta um modelo paralelo de alto desempenho para categorização de áreas geográficas, denominado HPC-ICTM. Trata-se de um modelo geral para análise de espaços de natureza geométrica, baseado em tesselações, que é capaz de produzir uma categorização confiável de conjunto de pontos de um dado espaço, de acordo com múltiplas características dos pontos, cada característica correspondendo a uma camada do modelo. Por exemplo, uma região geográfica pode ser analisada de acordo com a sua topografia, vegetação, demografia, dados econômicos etc, cada uma gerando uma subdivisão diferente da região. O modelo é definido como categorizador, pois é capaz de subdividir uma certa região em segmentos que apresentam características similares pertencendo à mesma classe, de acordo com o conjunto de variáveis observadas. Como os dados de entrada numéricos são usualmente suscetíveis a erros, o modelo utiliza a aritmética intervalar para se ter um controle automático de erros. A fim de realizar análises de grandes regiões geográficas e usar informações geograficamente distribuídas, utiliza-se duas plataformas de execução de aplicações paralelas, as máquinas agregadas e as grades computacionais. Apresenta-se também o modelo geral ICTM, sua implementação seqüencial e validação e a descrição da arquitetura do modelo paralelo para máquinas agregadas e grades computacionais, chamado HPC-ICTM. Um protótipo do sistema é validado com dados reais extraídos de imagens de satélite.

Palavras-chave: Modelo Categorizador. ICTM. HPC-ICTM.

Abstract

This work presents a high performance parallel model for geographic area categorization called HPC-ICTM. It is a general model for the analysis of areas of geometric nature, based in tesselations, that is capable to produce a reliable categorization of point sets for a given data space, in accordance with multiple characteristics of the points, each characteristic corresponding to a layer of the model. For example, a geographic region can be analyzed according to its topography, vegetation, demography, economic data etc., each one generating a different subdivision of the region. The model is defined as categorizer because it is capable to subdivide a certain region in segments that present similar characteristics belonging to the same class, in accordance with the set of observed variables. Because the input of numerical data is usually error prone, the model uses interval arithmetic to allow automatic error control. In order to make analysis of large geographic regions and to use geographically distributed information, two parallel execution platforms are utilized, clusters and computational grids. The general ICTM model, its sequential implementation and validation are presented, and the architecture of the parallel model for clusters and computationtal grids, called HPC-ICTM, is described. A prototype of the system is validated with real data extracted from satellite images.

Keywords: Categorizer Model. ICTM. HPC-ICTM.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ICTM	*Interval Categorizer Tesselation Model*	15
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico	15
GMFC	Grupo de Matemática e Fundamentos da Computação	15
ACI	Autômatos Celulares Intervalares	15
CT-PETRO	Fundo Setorial do Petróleo e Gás Natural	15
CT-INFO	Fundo Setorial para Tecnologia da Informação	15
FMC2	Fundamentos Matemáticos da Computação 2	15
Topo-ICTM	*Topography Interval Categorizer Tesselation Model*	15
CPAD	Centro de Pesquisa em Alto Desempenho	16
HPC-ICTM	*High Performance Computing Interval Categorizer Tesselation Model*	17
SIG	Sistema de Informação Geográfica	18
CAC	Cartografia Assistida por Computador	18
DEM	*Digital Elevation Model*	18
PC	*Personal Computer*	18
CADD	*Computer Aided Design and Drafting*	19
AM-FM	*Automated Mapping – Facility Management*	20
GIS	*Geographic Information System*	20
SGI	Sistema Geográfico de Informação	20

GPS	*Global Positioning System*	21
LIS	*Land Information Systems*	24
WAP	*Wireless Application Protocol*	25
SMP	*Symmetrical Multiprocessor*	29
LAN	*Local Area Network*	29
NORMA	*Non-remote Memory Access*	30
SCI	*Scalable Coherent Interface*	31
CRM	*Cluster Resource Manager*	31
DSM	*Distributed Shared Memory*	31
MPI	*Message Passing Interface*	31
PVM	*Parallel Virtual Machine*	31
SPMD	*Single-Program Multiple Data*	32
PBS	*Portable Batch System*	34
LSF	*Load Sharing Facility*	34
HTML	*HyperText Markup Language*	37
RPC	*Remote Procedure Call*	37
Bot	*Bag of Tasks*	38
UTM	*Universal Transversa de Mercator*	57
WQR	*WorkQueue with Replication*	74
jdf	*job description file*	85
SRTM	*Shuttle Radar Topography Mission*	90
GIPE	*Geographic Information Processing Environment*	109

Sumário

1 Introdução

A idéia de criação de um modelo para categorização de áreas geográficas originou-se da proposta de Coblentz et al. [1, 2], para uma metodologia de subdivisão confiável de áreas geológicas baseada na análise da monotonicidade da função que mapeia a declividade da área considerada. Neste modelo, a área total é dividida em regiões perfiladas lado a lado e, por isso, diz-se que a análise é unidimensional, pois percorre somente um sentido do modelo.

Na geofísica[1], uma subdivisão apropriada de uma área geográfica em segmentos é extremamente importante porque se possibilita extrapolar os resultados obtidos em algumas partes dos segmentos (onde uma pesquisa extensiva foi feita) para outras partes dentro do mesmo segmento e ter um bom entendimento destas partes que não foram totalmente analisadas.

Inspirados em algumas particularidades dos autômatos celulares, Aguiar e Costa [3] apresentaram um modelo que a partir da definição de uma malha que representa uma certa região, executa uma análise bidimensional do sinal da declividade do relevo desta região. Este modelo utiliza regras locais para a criação e categorização das sub-regiões, apresentando a situação relativa de cada sub-região com relação a área total, de acordo com os estados assumidos pelas células da malha. Mais precisamente, substituiu-se o modelo inicialmente proposto por um modelo bidimensional baseado em autômatos celulares.

Aguiar e Costa apresentaram também, a generalização deste modelo baseado em autômatos celulares [4]. Trata-se de um modelo baseado em tesselações, que é capaz de realizar uma análise de várias características associadas às células da malha, impondo pesos ou prioridades às características, resultando em categorizações diferentes, de acordo com essas prioridades. Este novo modelo é chamado de *Interval Categorizer Tesselation Model* (ICTM).

1.1 Motivação e Objetivo do Trabalho

Os modelos citados anteriormente foram desenvolvidos durante a realização de dois projetos aprovados junto ao CNPq, pelo Grupo de Matemática e Fundamentos da Computação (GMFC) da Universidade Católica de Pelotas (UCPel). O projeto ACI [5], no contexto do fundo setorial CT-PETRO[2], e uma extensão do mesmo no contexto do fundo setorial CT-INFO[3], denominado FMC2 [6].

Os principais resultados desses projetos foram [4, 7]: (i) a implementação seqüencial da versão Topo-ICTM do modelo categorizador, que considera apenas uma característica de uma região geográfica (bidimensional), qual seja, a declividade da função que mapeia o relevo da

[1] Estudo da terra usando medidas físicas tomadas na sua superfície.
[2] Fundo Setorial do Petróleo e Gás Natural.
[3] Fundo Setorial para Tecnologia da Informação.

região considerada; (ii) a extensão da versão simplificada Topo-ICTM para o modelo geral ICTM para análise de espaços de natureza geométrica, baseado em tesselações, que é capaz de produzir uma categorização confiável de conjunto de pontos de um dado espaço, de acordo com múltiplas características dos pontos, cada característica correspondendo a uma camada do modelo, possibilitando a realização de análises baseadas em várias características de um determinado espaço multi-dimensional; (iii) e a validação do modelo ICTM através da utilização de dados reais oriundos de imagens de satélite.

A partir desses resultados obtidos foi possível constatar a eficácia do modelo ICTM e observar seu desempenho. Para análises de grandes regiões geográficas o desempenho do modelo foi considerado insatisfatório, apresentando, em alguns casos, limitações que impediram seu uso. Além disso, a obtenção de informações geográficas é bastante complicada, principalmente devido ao alto custo dos aparelhos que as geram [8].

Para dar continuidade ao desenvolvimento do ICTM e superar as limitações encontradas até então, um novo projeto foi criado e submetido ao CNPq no final do ano de 2004 (obtenção dos recursos a partir do ano de 2005), para o programa de pesquisa e desenvolvimento para capacitação de pequenos grupos acadêmicos na área de Tecnologia da Informação. Uma das atividades previstas neste projeto era a paralelização do modelo ICTM para execução em máquinas agregadas (*clusters*) e em grades computacionais[4] (*computational grids*), com o intuito de obter melhor desempenho em análises de grandes regiões geográficas, reduzindo as limitações do modelo que impedem seu uso, e possibilitar a utilização de informações geograficamente distribuídas, facilitando o processo de obtenção dos dados geográficos (forte demanda das áreas relacionadas ao processamento e análise de dados geográficos).

O uso de informações geograficamente distribuídas, através de um ambiente de grade, pode ser bastante interessante. Diversos centros de pesquisa das áreas de geoprocessamento, sensoriamento remoto etc, poderiam montar um ambiente de grade e disponibilizar seus dados, de acordo com seus tópicos de interesse. Isso possibilitaria a criação de um repositório de dados distribuído que facilitaria o acesso às informações para cada centro de pesquisa. Além disso, o custo com a obtenção dos dados seria reduzido e seria estabelecida uma relação entre os membros da "comunidade" que poderia gerar diversos resultados, colaborando para o crescimento das áreas de conhecimento relacionadas.

Neste contexto, em um trabalho de cooperação entre o GMFC e o CPAD[5], se enquadra o presente trabalho. No entanto, é importante salientar que o projeto submetido ao CNPq não foi contemplado. Porém, tanto o GMFC quanto o CPAD continuam atualmente com as linhas de pesquisa que seriam inseridas no projeto, ou seja, o GMFC continua estudando e desenvolvendo o ICTM, inclusive com a submissão de novos projetos aos órgãos de fomento nacionais e até mesmo novos projetos com cooperação internacional para o próximo ano, e o CPAD desenvolve trabalhos de pesquisa tanto em máquinas agregadas quanto em grades computacionais.

A motivação para a realização deste trabalho está justamente relacionada ao envolvimento de dois grupos de pesquisa de Universidades diferentes, com focos de pesquisa distintos e atuais e com características inovadoras nas áreas de conhecimento relacionadas, o que favorece a publicação de artigos técnicos sobre o assunto.

[4] O termo "grade" será usado no decorrer do texto para referenciar grades computacionais.

[5] Centro de Pesquisa em Alto Desempenho - Parceria PUCRS/HP Brasil.

O objetivo principal do trabalho foi gerar o modelo de alto desempenho para categorização de áreas geográficas chamado de *High Performance Computing Interval Categorizer Tessellation Model* (HPC-ICTM), através da criação e definição eficaz de modelos paralelos do ICTM para execução em máquinas agregadas e em grades computacionais, possibilitando a utilização do mesmo para análise de grandes regiões geográficas e o uso de informações distribuídas geograficamente.

Sendo assim, este trabalho apresenta todo processo de geração e desenvolvimento do HPC-ICTM, através da descrição detalhada de todas as etapas deste processo. Entre elas, o estudo do modelo geral ICTM, a implementação seqüencial do mesmo e sua validação, a modelagem paralela do modelo para máquinas agregadas e grades computacionais, a definição do novo modelo HPC-ICTM e a implementação do seu protótipo, e a validação do modelo HPC-ICTM com dados reais oriundos de imagens de satélites.

No que diz respeito aos benefícios para a comunidade científica pode-se dizer que este trabalho contribui para o desenvolvimento das ciências ambientais. O modelo HPC-ICTM é inovador e apresenta resultados importantes que não podem ser obtidos nos *softwares* de processamento de dados geográficos atuais.

Além disso, o trabalho apresenta resultados interessantes para a área de processamento paralelo e distribuído. O uso de grandes volumes de dados em um ambiente amplamente distribuído, por exemplo, não é uma tarefa fácil. É necessário investigar muitos aspectos e superar muitos desafios para que se consiga resultados satisfatórios.

1.2 Organização do Restante do Documento

O restante do documento está organizado conforme o exposto a seguir.

O Capítulo 2 apresenta o estado da arte da análise de dados geográficos, destacando conceitos e características importantes como SIG's, Geoprocessamento, Geomática e Análise Digital de Terrenos. Além disso, apresenta-se também uma relação do modelo ICTM com os sistemas atuais para processamento de dados geográficos.

A seguir, no Capítulo 3, apresentam-se as plataformas de execução de aplicações paralelas adotadas no trabalho, as máquinas agregadas e as grades computacionais.

O modelo ICTM é detalhado no Capítulo 4, destacando a formalização do modelo, suas características, funcionamento, aplicabilidade, implementação seqüencial e validação através do uso de imagens de satélite.

O Capítulo 5 descreve todo processo de geração e desenvolvimento do HPC-ICTM, destacando a modelagem do ICTM para as duas plataformas utilizadas no trabalho e a implementação dos modelos paralelos criados.

O Capítulo 6 apresenta a validação do modelo HPC-ICTM através da utilização de dados reais extraídos de imagens de satélite.

Por fim, uma conclusão sobre o trabalho é apresentada no Capítulo 7, bem como alguns apontamentos para trabalhos futuros.

2 Estado da Arte da Análise de Dados Geográficos

2.1 Sistemas de Informações Geográficas

O emprego do conceito de computação para o processamento de dados geográficos remonta ao século passado, quando o Censo Americano utilizou cartões perfurados e uma máquina tabuladora para agilizar as atividades relativas ao censo de 1890, tendo finalizado após três anos. Isto resultou em grande avanço comparado ao censo anterior, de 1880, que demorou oito anos para ser efetuado convencionalmente.

Entretanto, a falta de ferramentas matemáticas adequadas para descrever quantitativamente a variação espacial era ressentida pela comunidade científica. Conforme Burrough [9], as principais etapas históricas dos Sistemas de Informações Geográficas (SIG's) são as seguintes:

- Surgimento dos primeiros modelos matemáticos (décadas de 30 e 40), juntamente com os métodos estatísticos para análise de séries temporais;
- Com a disponibilidade do computador digital (década de 60) ocorreu o desencadeamento da utilização de ferramentas computacionais adequadas para o mapeamento temático quantitativo e análise espacial;
- Surgimento do primeiro SIG em 1964 no Canadá (*Canada Geographic Information System*) por iniciativa do Dr. Roger Tomlinson, que embora tenha construído os módulos básicos do *software*, impulsionando o desenvolvimento de *hardware* e elaborado uma complexa base de dados, só publicou seus trabalhos uma década depois;
- Estabelecimento do desenvolvimento dos SIG's (final da década de 70) que favoreceu o surgimento da versão comercial dos primeiros sistemas no início da década de 80.

Os governos americano, canadense e alguns europeus apoiavam financeiramente iniciativas voltadas à Cartografia Assistida por Computador (CAC). E neste período passou-se a tornar disponíveis ao público bases de dados digitais, tais como os modelos digitais de elevação (DEM's – *Digital Elevation Models*).

Atualmente, percebe-se um crescimento acentuado das aplicações de SIG's, devido à disseminação do PC, além da introdução de tecnologia de relativo baixo custo e alto desempenho das estações de trabalho (*Workstations*).

Além do serviço de venda de mapas analógicos aos usuários, surgiu uma outra alternativa: o arrendamento de dados através do estabelecimento de contratos, definindo inclusive a freqüência das suas atualizações.

Com o surgimento dos Sistemas de Informação, associou-se à informação o conceito de valor agregado, que é obtido ao se reunir de forma ordenada conjuntos de dados que previamente

estavam não relacionados, e cuja combinação pode ser utilizada para a realização de tarefas adicionais.

Segundo Rodrigues [10, 11], *Geoprocessamento* é a tecnologia de coleta e tratamento de informações espaciais e de desenvolvimento de sistemas que as utilizam. Ainda, apresenta uma classificação dos Sistemas de Geoprocessamento como:

sistemas aplicativos: são conjuntos de programas que realizam operações associadas a atividades de projeto, análise, avaliação, planejamento etc, em áreas tais como transportes, mineração, hidrologia, urbanismo; geralmente, são voltados à representação espacial e à realização de operações sobre estas representações; são sistemas voltados à entrada de dados, saída de dados e realização de tarefas em projeto assistido por computador e mapeamento automatizado;

sistemas de informações: são *softwares* que desempenham a coleta, tratamento e apresentação de informações espaciais. De maneira mais geral, o SIG é o conjunto de *software*, *hardware*, procedimentos de entrada e saída dos dados, fluxos de dados do sistema, normas de codificação de dados, normas de operação, pessoal técnico etc que desempenham as funções de coleta, tratamento e apresentação de informações.

sistemas especialistas: sistemas computacionais que empregam o conhecimento na solução de problemas que normalmente demandariam a inteligência humana; emulam o desempenho de um especialista atuando em uma dada área do conhecimento.

Define-se informação geográfica como o conjunto de dados ou valores que podem ser apresentados em forma gráfica, numérica ou alfanumérica, e cujo significado contém associações ou relações de natureza espacial.

Entretanto, na literatura especializada, não há concordância no estabelecimento da classificação dos sistemas de geoprocessamento, e na maioria das vezes, apresentam-se múltiplas características com predominância de um conjunto particular de funções [11]. Por isso, faz-se necessário apresentar a diferenciação feita por Korte [12] entre CADD-CAM, AM-FM e GIS:

CADD: *Computer Aided Design and Drafting*, ou Projeto Assistido por Computador, é uma tecnologia normalmente empregada pelo CAM (*Computer Assisted Mapping*), ou Mapeamento Assistido por Computador, para a produção de mapas em substituição ao processo cartográfico tradicional.

Os dados são organizados em camadas (*layers*), empregados para organizar as feições do mapa por temas (*themes*). A utilização do CAM reduz o tempo de produção de mapas e possibilita a economia de recursos financeiros quando comparado aos processos cartográficos tradicionais. Assim, as atualizações se tornam mais simples e rápidas, uma vez que se modifica somente o elemento selecionado sem causar alteração nos demais.

Entretanto, CAM não é um sistema muito adequado para realizar análises; pois, as relações espaciais não são definidas na estrutura de dados, requerendo processamentos adicionais (mais demorados) para a inspeção de tais relações.

AM-FM: Mapeamento Automatizado (*Automated Mapping*) e Gerenciamento de Serviços de Utilidade Pública (*Facility Management*) baseiam-se também em tecnologia CADD.

Entretanto, a apresentação gráfica geralmente não é tão precisa e detalhada quanto em sistemas CAM porque sua ênfase está centrada no armazenamento, na análise e na emissão de relatórios.

As relações entre os componentes do sistema de utilidade pública são definidas como redes (*networks*) que são associadas à atributos. Todavia, relações espaciais não são definidas nestes sistemas.

GIS: Sistema de Informação Geográfica (*Geographic Information System*) é recomendado para a análise de dados geográficos.

Este tipo de sistema difere dos dois sistemas anteriormente apresentados porque define as relações espaciais entre todos os elementos dos dados. Esta relação espacial é conhecida como **topologia dos dados** e pode indicar um conjunto muito grande de informações, além da descrição da localização e geometria das feições cartográficas.

A Topologia também descreve como as feições lineares estão conectadas, como as áreas são limitadas e quais áreas são contíguas. Para definir a topologia do mapa, o SIG usa uma estrutura de dados espacial, empregando nodos (*nodes*), arcos (*lines*) e áreas (*polygons*).

O SIG também contém dados em atributos e dados geométricos espaciais que são associados com os elementos topológicos para representar informações descritivas.

Além disso, o SIG permite o acesso aos dados (espaciais e atributos) ao mesmo tempo e, por isso, o dado atributo pode ser relacionado com o dado espacial e vice-versa.

Outros autores apresentam definições que ajudam a compreender a complexidade funcional e estrutural de um SIG:

- Sistema Geográfico de Informação (SGI) constitui o tipo de estrutura mais importante em termos de viabilização do Geoprocessamento. O SGI é um conjunto de procedimentos computacionais que operando sobre bases de dados geocodificados ou, mais evoluidamente, sobre bancos de dados geográficos executa análise, reformulações e sínteses sobre os dados ambientais disponíveis [13].

- Sistemas de Informações Geográficas são modelos do mundo real úteis a um certo propósito; subsidiam o processo de observação (atividades de definição, mensuração e classificação), a atuação (atividades de operação, manutenção, gerenciamento, construção etc) e a análise do mundo real [14].

- SIG's são constituídos por uma série de processos de análise para focalizar o relacionamento de determinado fenômeno da realidade com sua localização espacial. Utilizam uma base de dados computadorizada que contém informação espacial, sobre a qual atuam uma série de operadores espaciais; baseia-se numa tecnologia de armazenamento, análise e tratamento de dados espaciais, não-espaciais e temporais e na geração de informações correlatas [15].

- SIG's integram numa única base de dados: informações espaciais provenientes de dados cartográficos, dados de censo e de cadastro urbano e rural, imagens de satélite, redes e modelos numéricos de terreno. Além disso, combinam as várias informações, através de algoritmos de manipulação, para gerar mapeamentos derivados; consultar, recuperar, visualizar e plotar o conteúdo da base de dados geocodificados [16].

A seguir, comentam-se as principais atividades envolvidas na utilização de um SIG.

2.1.1 Coleta de Dados

Um SIG permite a integração de dados que foram recolhidos em diferentes tempos, escalas, utilizando diferentes métodos de coleta de dados [10]. São fontes de dados: mapas (papel ou transparências), dados escritos, arquivos digitais contendo informações planialtimétricas e temáticas e informações armazenadas na memória do especialista. A integração destes dados de diferentes formatos, de tempos variados e em diferentes escalas, seria inviável economicamente e temporalmente sem a utilização do SIG.

Os dados podem ser importados para o SIG através da digitação dos dados textuais, ou da digitalização através de mesa *scanner* dos mapas existentes. Entretanto, estes métodos são limitados porque os mapas originais freqüentemente são antiquados, possuem erros de transcrição e podem não ter a escala apropriada.

Hoje, a coleta está cada vez mais sofisticada em função da maior diversidade do conhecimento humano e de tecnologias e equipamentos mais precisos, como: sensores remotos, fotogrametria e levantamentos a campo executados com GPS (*Global Positioning System* – Sistema de Posicionamento Global) e estações TOTAL.

A entrada de dados de má qualidade causa interpretações errôneas ou sem sentido da informação derivada de um SIG, por isso, recorrentemente se afirma que "um SIG é tão bom quanto as informações que contém".

O GPS provê dados exatos e atualizados instantaneamente, a um custo relativamente baixo. Ao se utilizar GPS, pode-se definir um dicionário de dados e recolher os atributos no campo ao mesmo tempo que se recolhem dados de posição. Desta forma, elimina-se os erros de transição (entrada de dados no sistema) e garante a atualização das informações da base de dados.

A fotogrametria obtém informações confiáveis sobre objetos e sobre o meio ambiente com o uso de processos de registro, medições e interpretações das imagens fotográficas e padrões de energia eletromagnética.

A necessidade contínua de se obter novos dados espaciais representa um dos maiores custos na utilização de um SIG. Com uma ferramenta de coleta de dados, o SIG simplifica a coleta de dados iniciais e também garante que a informação esteja sempre atualizada.

2.1.2 Tipos de Dados

Dados **cartográficos** e dados **não-gráficos** são os dois tipos de dados mais recorrentes em um SIG [17].

Os dados cartográficos são as informações geográficas oriundas de mapas armazenadas digitalmente. Cada entidade destes mapas é classificada como pontos, linhas e polígonos, estes últimos também são chamados de áreas ou regiões.

Um ponto representa uma característica que necessita somente uma localidade geográfica (por exemplo, latitude-longitude) para referenciá-la. Ou seja, um ponto pode representar a posição de estações meteorológicas, de poços e postes.

Uma linha (arco) é formada por uma série de pontos conectados unidimensionalmente, possuindo apenas comprimento (ausência de largura). Por exemplo, riachos, estradas e rastros de animais podem ser características representadas por linhas no SIG.

Um polígono é uma área cercada por linhas e a área compreendida pelo polígono possui comprimento e largura (bidimensional). Por exemplo, áreas com mesmo tipo de solo, regiões para plantação de arroz e banhados são representadas por polígonos.

Os dados não-gráficos consistem de informações descritivas sobre as características (pontos, linhas e áreas) armazenadas numa base de dados e referenciadas em um mapa.

Estas informações descritivas são nomeadas de atributos. Um atributo comum à todas as características é a situação geográfica, ao qual pode dar-se o nome de atributo `SITUAÇÃO`. Outros atributos dependem do tipo de característica e de que características são importantes para um propósito ou aplicação em particular. Por exemplo:

- uma parcela de terreno possui um proprietário, um tamanho e um uso;
- um poço de petróleo é de um determinado tipo e possui um índice de fluxo diário;
- uma estrada possui um nome, um tipo de superfície e pode possuir uma rota ou número de designação.

Cada uma destas características pode ser identificada especificamente num SIG ao dar-lhe um nome de atributo tal como `DONO`, `USO-TERRENO`, ou `NOME-ESTRADA`. O conjunto de valores assumido por cada atributo é chamado de **domínio**. Por exemplo, o domínio para o atributo `NOME-ESTRADA` são todos os nomes de estrada na área de interesse.

2.1.3 Estruturas de Dados

As estruturas de dados existentes em SIG's são a topologia e as camadas [12]. A estrutura de topologia refere-se à conexão das características das relações espaciais fundamentais. A topologia fornece a lógica que conecta pontos, linhas e polígonos. As camadas indicam apenas o modo que o SIG estratifica seus dados.

A informação topológica descreve a relação espacial entre as características e geralmente não é modificada pelo profissional que opera o SIG.

Ao fazer a descrição da posição de algum objeto, usualmente diz-se que está à esquerda, ao lado de, ou a determinada distância de um dado objeto. Esta definição não é precisa o suficiente para um SIG. Para se realizar uma análise espacial são requeridas as definições precisas fornecidas pela topologia.

A topologia define a relação posicional das características de acordo com as suas propriedades, por exemplo: as informações sobre que rótulos estão vinculados a cada característica, como os pontos estão ligados uns aos outros e, que pontos e linhas formam um polígono em particular.

Esta informação topológica, armazenada no SIG, permite que sejam efetuadas relações espaciais, tais como: a sobreposição de polígonos, o isolamento de polígonos, determinar se uma linha está dentro de um polígono e determinar a proximidade entre características. Um sistema SIG que realiza manipulações e análises não topológicas (como ocorre em sistemas CAD) são limitados.

Geralmente, um SIG permite a separação das informações de um mapa em categorias lógicas chamadas de camadas, temas, níveis de informação ou planos de informação (PI). Os planos de informação geralmente contém informações sobre um tipo de característica (como áreas de tipo de solo) ou sobre um pequeno grupo de características relacionadas (por exemplo, recursos de utilidade pública como telefone, gás e linhas de transmissão de energia).

Os dados de um mapa são separados logicamente em camadas para que assim possam ser manipulados e analisados espacialmente, isoladamente ou em combinação com outras camadas. Para se obter resultados analíticos significativos, os planos de informação devem estar referenciados geograficamente entre si por um sistema de coordenadas comum.

Estes planos podem ser combinados a fim de criar mapas compostos a partir da sobreposição destas, de maneira análoga à sobreposição de transparências em um retro-projetor. Durante esta análise criam-se novas sobreposições com a combinação matemática de sobreposições já existentes. Pode-se utilizar estas combinações para a criação de cenários alternativos.

2.1.4 Análise de Dados

A análise dos dados permite derivar informação a partir do conteúdo da base de dados do sistema. Esta análise de dados compreende [15]:

- Sobreposição espacial das características;
- Questões à base de dados;
- Reclassificação, combinação e eliminação de características;
- Cálculo de proximidade de características.

Por exemplo, o operador do SIG necessita saber que tipos de casas são construídas em um dado tipo de solo. Para responder a este tipo de pergunta o sistema utiliza dois conjuntos de dados: os tipos de solo e a localização geográfica das casas construídas na área solicitada.

Estes dados são fornecidos pelas duas camadas de informações e o SIG as combina para formar uma nova fonte de informações. A ligação entre as duas camadas de dados é a localização geográfica (latitude-longitude) de cada entidade.

Este exemplo é bastante simples e favorece ao operador visualizar a seleção das casas que estão construídas na área indicada. No entanto, na maioria dos casos manipula-se áreas muito grandes com dados bastante detalhados e, por isso, a tarefa pode ser extremamente difícil.

A visualização de dados compreende a criação de imagens e mapas, a visualização dos dados das características espaciais, a criação de produtos cartográficos e a combinação de todos estes elementos para a visualização na tela, ou para impressão das tabelas ou mapas, ou em arquivos para a utilização em outros programas.

2.2 Geomática

Segundo Gagnon e Bédard [18], durante a última década, a área de mapeamento tem sido incluída em um novo paradigma. No Canadá, como na maioria dos países, tem-se adotado o nome Geomática (*Geomatics*) para identificar este campo de atividade de mapeamento. A evolução do contexto de produção, otimização e gerenciamento de dados e informações espaciais tem sido a base da redefinição das atividades numa perspectiva global de planejamento em diferentes áreas.

Dessa forma, a Geomática representa a evolução do campo de atividades de levantamentos e mapeamento, congregando as disciplinas mais tradicionais (cartografia, geodésia, topografia) com as novas tecnologias (sensoriamento remoto, processamento digital de imagens, SIG, GPS e fotogrametria digital) e os novos campos de aplicação surgidos (entre eles a mineração, agricultura, meio ambiente, transportes, informática, turismo, saúde, telecomunicações etc).

A geomática engloba pelo menos quatro categorias técnicas relacionadas ao levantamento, mapeamento e planejamento [18]:

- Técnicas para coleta de informação espacial: cartografia, sensoriamento remoto, GPS, topografia convencional, fotogrametria, levantamentos de dados geográficos;
- Técnicas para armazenamento de informação espacial: banco de dados (orientados a objetos, relacional, hierárquico etc);
- Técnicas para tratamento e análise de informação espacial: modelagem de dados, geoestatística, aritmética, lógica, funções topológicas, redes;
- Técnicas para uso integrado de informação espacial: SIG's, LIS (Sistemas de Informação de Terrenos – *Land Information Systems*), AM/FM e CADD.

Para ressaltar algumas das principais aplicações na área de Geomática, assim como para o geoprocessamento, podemos citar os projetos de SIG's para [8]:

- elaboração de mapas urbanos básicos
- elaboração/atualização de mapas de arruamentos
- cadastro, mapeamento de cadastro urbano e rural
- fonte de apoio para trabalhos com GPS
- uso e ocupação do solo

- regularização dos limites de propriedades, demarcação de pequenas glebas
- previsão de safras, controle de pragas e agricultura de precisão
- estimativa de potencial econômico, projetos de desenvolvimento sustentável

A criação e a difusão do uso da internet, aliado ao avanço no desenvolvimento de redes de computadores cada vez mais rápidas, indicaram a possibilidade da aplicação do processamento distribuído aos SIG's.

As potenciais aplicações do processamento distribuído no geoprocessamento estão ligadas, na maioria das vezes, a sistemas de decisão on-line, por exemplo [19–22]:

1. Na logística:
   - a escolha da melhor rota a ser seguida pelos caminhões na distribuição de produtos.
   - mapas turísticos e de localização para viajantes, como o serviço prestado pelo *site* `http://www.apontador.com.br/`, que possibilita pesquisar o mapa de uma cidade (capitais) para localizar ruas, endereços, serviços e estabelecimentos mais próximos (como bancos, farmácias ou supermercados), para encontrar a melhor rota entre dois endereços.
   - a implantação de serviços WAP (*Wireless Application Protocol*), para usuários de internet por celular, para localização de endereços e roteirização. Se o celular for munido de um receptor GPS, as opções tornam-se ainda maiores: o usuário identifica exatamente o ponto onde se encontra e ainda usufruir de serviços de roteamento para descobrir a melhor maneira de chegar ao local desejado.
2. Na agricultura:
   - para o processamento de informações climáticas, para o suporte do trabalho de plantação e colheita.
   - para previsão de veranicos, geadas, secas prolongadas, excessos de água ou condições climáticas que favoreçam o surgimento de pragas e doenças. Reduzindo a possibilidade de prejuízos e perdas na produção de culturas climaticamente regionalizadas. O agricultor, armado de informações básicas atualizadas, tem melhores condições e maiores chances de sucesso no combate às adversidades climáticas.
   - para o manejo da água e do solo, se mal executado pela falta de informação ou se executado fora de época, pode ser danoso e impedir uma maior produtividade. O agricultor pode escolher as épocas adequadas para plantio e colheita, o momento e a quantidade certa de irrigar, o momento correto de tratar o solo e a forma de se proteger das pragas.
3. Na distribuição de energia:
   - como ferramenta de comunicação interna, no que tange aos sistemas interligados e mapas georreferenciados, permitindo a disseminação de informação de forma rápida, precisa e segura, com uma visão facilitada dos pontos críticos da rede.

- para a integração de mapas com relatórios detalhados.
- para a descrição do aplicativo em nível gerencial e operacional, o uso de ferramentas *web* e a integração aos demais sistemas da empresa sob a mesma plataforma.
- para a atualização do cadastro da rede elétrica, utilizando sistema construído a partir de tecnologias de geoprocessamento.
- para descrição do sistema de transmissão e geração, que é estruturado pela junção das plantas cartográficas, levantamentos aerofotogramétricos, dados das propriedades envolvidas e imagens digitalizadas dos processos.
- para precificação de postes, localização de consumidores inadimplentes e localização de agentes arrecadadores.

2.3 Análise Digital de Terrenos

O desenvolvimento e a aplicação de SIG's em análise de terrenos para as ciências ambientais, (real aplicabilidade do ICTM), foram motivadas pela visão do mundo em camadas [21], onde os processos biofísicos estão posicionados hierarquicamente como apresentado na Figura 1.

Figura 1 – Processos biofísicos em camadas.

Este tipo de divisão é bastante útil pois demonstra a complexidade dos processos individuais além de algumas dificuldades que são encontradas em delinear apropriadamente as escalas espaciais e temporais.

Muitos dos processos biofísicos mais importantes que ocorrem na superfície da terra (ou próximos) são influenciados pelo controle das interações e pelos níveis dos eventos passados e contemporâneos. Estes inter-relacionamentos são complexos e podem ser melhores entendidos usando a abordagem de modelagem de sistemas dinâmicos [23].

O trabalho de Phillips [24] apresenta exemplos de sistemas de populações interagindo em uma comunidade ecológica; e Schaffer [25] tratava a geomorfologia fluvial; ambos demonstram que a operação de processos chaves sobre diferentes escalas podem ser considerados independentemente uma em relação à outra.

Entretanto, não é aconselhável a utilização de modelos para uma dada escala originalmente desenvolvidos em outra escala.

A maioria das pesquisas hidrológicas, geomorfológicas e ecológicas são conduzidas na escalas global, nano e micro [21]. As meso e topo-escalas recebem pouca atenção apesar de serem realmente importantes pois muitas das soluções para problemas ambientais, como erosão acelerada do solo e poluição, requerem o gerenciamento de estratégias nestas escalas [26].

A influência da morfologia da superfície sobre a canalização na hidrologia e o impacto da declividade e do aspecto sobre a exposição ao sol representam os controles mais importantes operando na topo-escala.

Além disso, diversos estudos mostram como a forma da superfície do terreno pode afetar a migração lateral e o acúmulo da água, sedimentos e outros constituintes [21].

A crescente popularidade do trabalho nestas duas escalas intermediárias tem influenciado a disponibilidade de dados de elevação digitais contínuos e de alta-resolução e o desenvolvimento de novas ferramentas computadorizadas de análise de terrenos [20, 27, 28].

O modelo ICTM e conseqüentemente o modelo HPC-ICTM, introduzidos nesta dissertação, representam contribuições para a análise de dados geográficos nessas escalas intermediárias.

2.4 Considerações sobre o ICTM na Análise de Dados Geográficos

As seções anteriores deste capítulo exibem um apanhado geral do estado da arte dos temas Geoprocessamento, Geomática e Análise Digital de Terrenos, destacando a utilização dos SIG's na análise e processamento de dados geográficos. Mais precisamente, pode-se dizer que são nos SIG's que as informações geográficas são manipuladas e tratadas atualmente.

No entanto, é importante ressaltar que o ICTM não pode ser comparado a um SIG. Segundo Burrough [9], um SIG é um conjunto poderoso de ferramentas para coletar, armazenar, recuperar, transformar e visualizar dados sobre o mundo real. Já o ICTM é um modelo categorizador de regiões geográficas.

O ICTM pode ser considerado uma funcionalidade de um SIG, pois sua principal característica é a extração de uma informação geográfica, seja de uma imagem de satélite ou de um mapa, dando seqüência a geração de uma outra informação geográfica. Por exemplo, no caso de análise da topografia de uma determinada região, a partir da extração de dados altimétricos

de uma imagem de satélite que representa esta região, o modelo ICTM permite gerar de forma gráfica regiões que possuem características semelhantes, ou seja, trata-se da geração de outra informação espacial.

Uma outra questão importante é o aspecto inovador do modelo ICTM. Os SIG's atuais permitem gerar declividade, curvas de nível e outras informações referentes aos dados, por exemplo, de altitude de um terreno, mas o que é gerado no ICTM não é gerado em um SIG. Em outras palavras, os resultados de categorização produzidos pelo modelo não são produzidos pelos SIG's existentes.

No que diz respeito ao emprego do processamento paralelo e distribuído ao modelo ICTM, pode-se dizer que por tratar-se de uma "funcionalidade específica", não existem SIG's comerciais que processem seus algoritmos, modelos e técnicas em arquiteturas paralelas atuais. Quanto ao uso de informações geográficas distribuídas (processamento distribuído), pode-se dizer que a utilização de bases de dados geográficos distribuídas é o futuro da área de geoprocessamento. Espera-se que bancos de dados distribuídos permitindo interoperabilidade, o acesso de informações espaciais por SIG's distintos, sejam desenvolvidos em um futuro bastante próximo.

3 Plataformas Paralelas

Neste capítulo são abordadas duas plataformas de execução de aplicações paralelas – as máquinas agregadas e as grades computacionais – utilizadas para a geração do HPC-ICTM.

A Seção 3.1 versa sobre as máquinas agregadas, uma plataforma bastante difundida e utilizada no mundo inteiro. São apresentados alguns aspectos e conceitos importantes e relevantes para o trabalho, entre eles: os principais benefícios da plataforma, as características da arquitetura, os tipos de máquinas agregadas existentes, os paradigmas, bibliotecas e modelos de programação mais utilizados e os sistemas de gerenciamento de recursos de tais máquinas.

A Seção 3.2 apresenta as grades computacionais. Trata-se de uma nova plataforma que vem atraindo muitos pesquisadores devido às suas atraentes características e inúmeros desafios relacionados à sua utilização. Por ser um tema relativamente novo, apresentam-se alguns conceitos e as principais características sobre a plataforma, de acordo com alguns dos principais autores sobre o assunto.

3.1 Máquinas Agregadas

Desde o início da década de 90, devido à crescente facilidade de acesso aos componentes de computação de alto desempenho por parte dos pesquisadores e cientistas em geral, observa-se uma forte tendência na construção de máquinas paralelas, qual seja, a substituição de supercomputadores paralelos proprietários de alto custo por supercomputadores construídos através de PC's, estações de trabalho e multiprocessadores simétricos (SMP's – *Symmetrical Multiprocessors*), chamados de máquinas agregadas ou agregados (*clusters*) [29].

Devido à essa facilidade de construção, as máquinas agregadas tornaram-se a plataforma padrão para computação de alto desempenho e de larga escala. Atualmente, a construção de máquinas agregadas é uma prática bastante comum, como pode ser verificado na lista das 500 máquinas mais rápidas do mundo[1], onde 60.8% do conjunto total de supercomputadores paralelos são máquinas agregadas, ou seja, um total de 304 máquinas sendo que duas dessas figuram entre as 10 mais rápidas.

As máquinas agregadas são sistemas compostos por uma coleção de computadores interconectados (chamados de nós ou nodos) que trabalham em conjunto para execução de aplicações paralelas, como um único e integrado recurso computacional.

Cada nodo pode ser um sistema monoprocessado ou multiprocessado, ou seja, pode ser um simples computador pessoal, uma estação de trabalho ou um SMP, com mecanismos de entrada e saída e um sistema operacional. Os nodos podem estar dispostos em um mesmo gabinete ou fisicamente separados e conectados por uma rede local (LAN - *Local Area Network*) [30]. De

[1] Site TOP500: http://www.top500.org.

qualquer forma, as máquinas agregadas podem oferecer a noção de um único sistema operacional (imagem única) para os usuários e aplicações.

As principais vantagens com a utilização de máquinas agregadas são [31]:

- Relação custo $\times$ benefício: Ótima relação custo $\times$ benefício por serem construídos com "componentes de prateleira" (*hardware* produzido em larga escala e comum no mercado) que possuem custo reduzido;
- Configurabilidade: Alto grau de configurabilidade por tratar-se de uma arquitetura aberta, ou seja, a definição de seus componentes (nodos e rede) possibilita variadas configurações;
- Custo de manutenção: Custo reduzido devido à utilização de componentes produzidos em larga escala;
- Escalabilidade: Pode-se aumentar razoavelmente o número de nodos da máquina.

3.1.1 Arquitetura

Conforme foi comentado no início deste capítulo, as máquinas agregadas são compostas por uma coleção de computadores interconectados que operam em conjunto para solucionar um determinado problema. Sendo assim, as máquinas agregadas se enquadram em uma grande categoria de sistemas paralelos chamada de multicomputadores (sistemas com memória distribuída). Neste tipo de arquitetura, cada processador possui sua própria memória local e só pode endereçar esta memória. Devido à esta forma de acesso à memória, as máquinas agregadas são chamadas de máquinas NORMA (*Non-remote Memory Access*). A comunicação entre os processadores de um multicomputador é realizada através do paradigma de troca de mensagens.

É importante salientar que os nodos de uma máquina agregada podem ser multiprocessados e, por isso, podem ser enquadrados em uma outra categoria de sistemas paralelos, qual seja, os multiprocessadores (sistemas com memória compartilhada). No entanto, uma máquina agregada como um todo deve ser vista como um multicomputador.

A Figura 2 ilustra a arquitetura típica de uma máquina agregada. Observando a estrutura interna do nodo percebe-se que o adaptador de rede é fracamente acoplado ao processador, pois reside em um barramento de entrada e saída que é ligado ao barramento processador–memória através de um adaptador. Isso é uma conseqüência da utilização de uma estação de trabalho, PC, ou até mesmo de uma máquina multiprocessada como nodo, que possui vários níveis de barramento para suportar vários periféricos.

No que diz respeito aos componentes da arquitetura como um todo, pode-se dizer que os mais importantes são [32]:

- **Múltiplos nodos:** Múltiplos computadores de alto desempenho, podendo ser PC's, estações de trabalho ou SMP's;
- **Estado da arte em sistemas operacionais:** Sistemas operacionais baseados em *microkernel* ou com *kernel* otimizado, normalmente derivados do UNIX como Solaris e Linux;

- **Rede de interconexão:** Redes padrão como Ethernet, Fast-Ethernet e até mesmo Gigabit Ethernet ou redes de baixa latência tais como Myrinet [33] e SCI (*Scalable Coherent Interface*) [34];
- ***Middleware***: Sistemas de gerenciamento de recursos (CRM – *Cluster Resource Manager*) e monitoração, sistemas de arquivos paralelos e mecanismos de *hardware* como DSM (*Distributed Shared Memory*), por exemplo;
- **Ambientes de programação paralela:** Ambientes e ferramentas como compiladores MPI (*Message Passing Interface*) [35, 36] e PVM (*Parallel Virtual Machine*) [37], por exemplo;
- **Aplicações:** Aplicações seqüenciais, paralelas ou distribuídas.

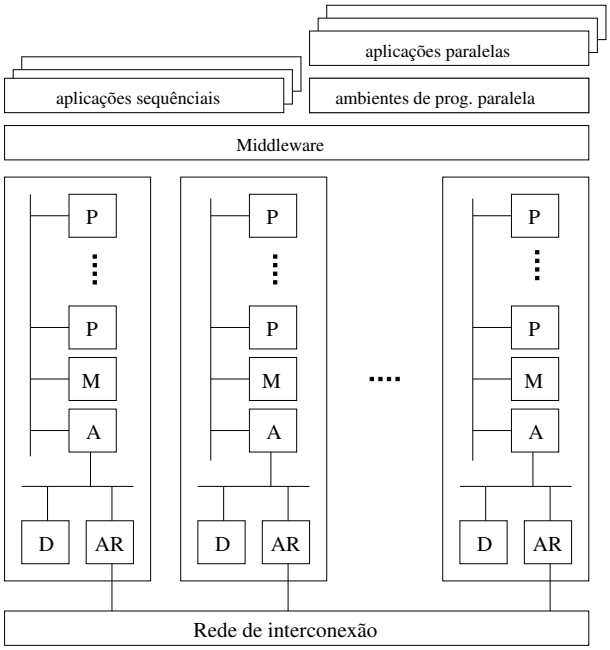

Figura 2 – Arquitetura de uma máquina agregada. Os componentes dos nodos são: Processador (P); Memória (M); Adaptador (A); Adaptador de Rede (AR).

3.1.2 Tipos de Máquinas Agregadas

Existem algumas variações e pontos de vista diferentes sobre os tipos de máquinas agregadas existentes por parte de alguns autores sobre assunto. Uma definição bastante encontrada na literatura, dada por Buyya [32], diz que existem basicamente quatro tipos de máquinas agregadas:

Alta Disponibilidade (*High Availability (HA) and Failover*): Estes modelos de máquinas agregadas são construídos para prover uma disponibilidade de serviços e recursos de forma ininterrupta através do uso da redundância implícita ao sistema. A idéia geral é que se um nodo da máquina vier a falhar (*failover*), aplicações ou serviços possam estar disponíveis em outro nodo. Esses tipos de máquinas agregadas são utilizadas para base de dados de missões críticas, correio, servidores de arquivos e aplicações.

Balanceamento de carga (*Load Balancing*): Este tipo de máquina agregada distribui o tráfego entrante ou requisições de recursos provenientes dos nodos que executam os mesmos programas entre as máquinas que compõem o agregado. Todos os nodos são responsáveis por controlar os pedidos. Se um nodo falhar, as requisições são redistribuídas entre os nodos disponíveis no momento. Este tipo de solução é normalmente utilizado em grandes servidores de *web* (*web farms*).

Combinação HA e *Load Balancing*: Como o próprio nome diz, combina as características dos dois tipos de máquinas agregadas citadas acima, aumentando assim a disponibilidade e escalabilidade de serviços e recursos. Este tipo de configuração de máquina agregada é bastante utilizada em servidores de *web*, *mail*, *news* ou *ftp*.

Processamento Distribuído ou Paralelo: Este modelo de máquina agregada aumenta a disponibilidade e desempenho para as aplicações, particularmente, as grandes tarefas computacionais. Uma grande tarefa computacional pode ser dividida em pequenas tarefas que são distribuídas ao redor das estações (nodos), como se fosse um supercomputador massivamente paralelo. Estas máquinas agregadas são usadas comumente para aplicações de computação científica, que possuem tarefas que exigem alto poder de processamento.

3.1.3 Paradigmas, Bibliotecas e Modelos de Programação

Apesar da existência de outros paradigmas, o paradigma de troca de mensagens é o mais utilizado para programação de máquinas agregadas [38].

Na programação paralela através deste paradigma de comunicação, uma aplicação consiste em uma coleção de processos autônomos; cada um deles possuindo sua própria memória local e comunicando-se com os demais processos através de funções de envio e recebimento de mensagens. Quando todos os processos executam o mesmo programa, atuando sobre partes diferentes dos dados, a aplicação é dita SPMD (*Single-Program Multiple Data*).

Estas funções, de envio e recebimento de mensagens, são organizadas em bibliotecas que criam uma abstração de máquina paralela. Entre as bibliotecas existentes, destacam-se as bibliotecas citadas anteriormente PVM e MPI.

No que diz respeito aos modelos de programação, não existe uma única classificação que contenha todos os modelos de programação paralela possíveis [39]. Vários autores apresentam suas classificações, mas nenhuma delas é exatamente a mesma. No entanto, pode-se destacar alguns modelos bastante difundidos [40]:

Mestre-Escravo (*Master-Slave*): Neste modelo o mestre executa as tarefas essenciais do programa paralelo e divide o resto das tarefas para os processos escravos (Figura 3). Os

escravos processam as tarefas e retornam os resultados para o mestre. Uma desvantagem deste método é a distribuição centralizada das tarefas, fazendo com que o mestre se torne o gargalo do sistema. Uma vantagem é a boa tolerância a falhas do modelo. Se qualquer um dos escravos falharem, o modelo continua funcionando. O problema só acontece no caso de falha do mestre.

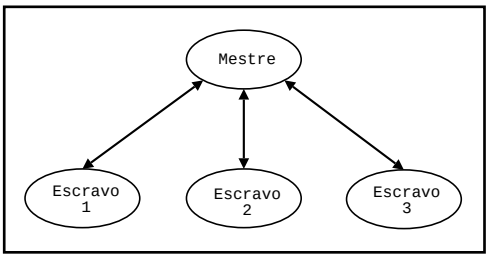

Figura 3 – Modelo mestre-escravo.

Divisão e Conquista (*Divide and Conquer*): Neste modelo um processo "pai" divide as tarefas em tarefas menores e atribui as mesmas aos processos "filhos". Pode-se pensar em uma estrutura baseada em árvore onde a integração dos resultados é feita de forma recursiva (Figura 4). Uma desvantagem inerente ao modelo é a dificuldade de obter um balanceamento de carga adequado na divisão de tarefas.

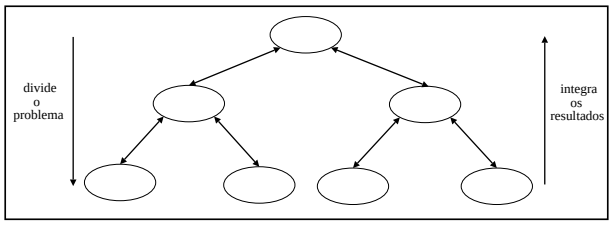

Figura 4 – Modelo divisão e conquista.

Pipeline: Este modelo se caracteriza pela formação de um *"pipeline virtual"* com um fluxo contínuo de dados (Figura 5). Ocorre a sobreposição da comunicação com a computação, pois os processos recebem mensagens dos processos anteriores ao mesmo tempo que estão processando dados previamente recebidos. Uma desvantagem desse modelo é a não tolerância a falhas. Se um processo cai, o *"pipeline virtual"* é quebrado.

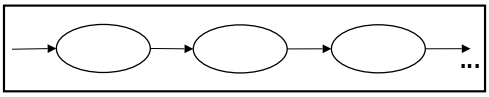

Figura 5 – Modelo *pipeline*.

Fases Paralelas (*Phase Parallel*): Neste modelo existem duas fases bem definidas, a computação e a interação (comunicação e sincronização). Portanto, os processos estão sempre

processando ou comunicando (Figura 6). Este modelo deve ser aplicado a problemas que se adaptem, devido às suas características, a essa abordagem de utilização de "fases". Isso porque o modelo apresenta alguns problemas devido ao sincronismo necessário entre os processos. Um deles é a ociosidade dos processos que acabam antes suas tarefas e precisam esperar os demais. O outro, é o *overhead* (alto tráfego na rede) de comunicação durante a fase de interação.

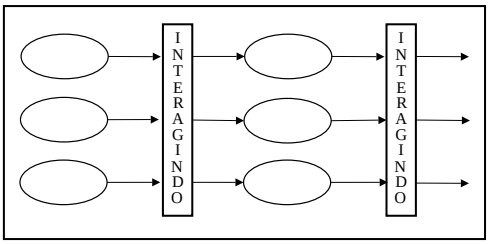

Figura 6 – Modelo fases paralelas.

Modelo Híbrido (*Hybrid Model*): Os limites entre os modelos citados acima podem as vezes serem nebulosos. Para algumas aplicações é necessário unir elementos de diferentes modelos. Este modelo caracteriza-se então, pela utilização de conceitos de vários modelos de programação paralela.

3.1.4 Sistemas de Gerenciamento de Recursos

O sistema de gerenciamento de recursos (CRM) é um dos *softwares* mais importantes de uma máquina agregada. Ele é responsável por controlar o acesso aos nodos da máquina agregada. Esse controle é baseado na especificação de direitos de acesso, políticas de escalonamento e configurações do ambiente de execução [41].

Além disso, um CRM deve prover mecanismos e serviços que permitam aos administradores e usuários interagir com o ambiente de computação, tornando possível a execução de requisições para alocação e liberação de recursos, a obtenção de informações sobre a fila de alocação, sobre direitos de acesso e sobre o agregado em geral, bem como a submissão de aplicações.

Alguns exemplos de CRM's conhecidos e utilizados no mundo inteiro são o PBS (*Portable Batch System*) [42] e o LSF (*Load Sharing Facility*) [43]. O CPAD utiliza um gerenciador de recursos de pequeno porte desenvolvido no próprio centro especificamente para atender suas necessidades, chamado CRONO [44].

3.2 Grades Computacionais

O termo "grade" surgiu na década de 90 para denotar uma proposta de infra-estrutura de computação distribuída [45]. Desde então, este termo tem despertado muito interesse na comunidade acadêmica e no ambiente empresarial, devido à constante necessidade de alto poder

computacional para aplicações científicas e comerciais de larga escala [46]. Entretanto, ainda existem muitas dúvidas e indefinições relacionadas ao termo. Mais precisamente, ainda não existe uma definição para o termo ou para as condições básicas que um sistema deve satisfazer e as características que deve possuir para ser considerado uma grade [47].

A visão original estabelece uma metáfora entre a rede elétrica e a grade computacional. A rede elétrica disponibiliza energia elétrica sob demanda e esconde do usuário detalhes como a origem da energia e a complexidade da malha de transmissão e distribuição. Ou seja, um equipamento elétrico é simplesmente conectado na tomada para que ele receba energia. A grade computacional, portanto, seria uma rede na qual o indivíduo se conecta para obter poder computacional (ciclos, armazenamento, *softwares*, periféricos etc). Um sistema que forneça poder computacional sob demanda e transparentemente, abstraindo toda a complexidade do processo e os problemas inerentes ao mesmo, é certamente desejável [48, 49].

Essa visão pode levar a uma comparação entre a grade computacional e a internet. Por exemplo, quando uma pessoa usa *home banking*, seu computador *desktop*, uma série de roteadores e os computadores do seu banco se agregam sob demanda para lhe fornecer um serviço de forma transparente, através da internet. Dessa forma, a internet se comportaria exatamente como uma grade. Entretanto, isso é apenas uma visão. É preciso pensar em grades como plataformas de execução de aplicações paralelas para que se possa oferecer um conceito mais útil.

Algumas definições mais formais são encontradas na literatura. Uma delas define uma grade computacional como um meio de compartilhamento seguro, coordenado e flexível entre recursos, indivíduos e instituições, através da utilização da internet [45]. Outra, que talvez seja a mais adequada, diz que uma grade computacional é um meio de compartilhar recursos, tais como computadores pessoais, supercomputadores, sistemas de armazenamento, dispositivos especializados e *softwares* que estão geograficamente distribuídos, para resolver problemas computacionais de larga escala na ciência, engenharia e comércio [47, 49–53].

Um outro conceito bastante interessante é o de Organizações Virtuais (VO's - *Virtual Organizations*). Uma organização virtual nada mais é do que um conjunto de indivíduos de diversas organizações (empresas, Universidades, institutos...) e seus recursos, podendo estar distantes geograficamente, que serão compartilhados de forma estruturada e organizada por tempos determinados ou indeterminados [45].

É interessante ressaltar também, a idéia da tecnologia chamada de computação par-a-par (*peer-to-peer computing*) que, assim como a computação em grade, tem o objetivo de utilizar recursos compartilhados em larga escala [54]. Os conceitos de ambas tecnologias costumam ser confundidos. A principal diferença entre elas é que a computação par-a-par está relacionada à descentralização para permitir uma maior escalabilidade do sistema (evitando dependência sobre pontos centralizados) e a comunicação direta entre os pares do sistema (que fazem o papel de servidor e cliente). Uma discussão mais detalhada sobre as duas tecnologias é dada por Foster e Iamnitichi [55].

3.2.1 Características da Plataforma

Conforme comentado anteriormente, ainda não existe um consenso sobre quais as condições necessárias (características) para um sistema ser considerado uma grade. Como exemplo pode-

se pensar nas diferenças entre grades e máquinas agregadas. De maneira geral, as grades são mais distribuídas, diversas e complexas que uma máquina agregada ou outra plataforma de execução de aplicações paralelas. Os aspectos que evidenciam esta distribuição, diversidade e complexidade são [46]:

Heterogeneidade: Os componentes das grades são heterogêneos, ou seja, possuem características diferentes como arquitetura, *softwares* instalados e etc.

Alta dispersão geográfica: As grades podem ter escala mundial. Mais precisamente, seus componentes podem estar geograficamente distribuídos (em qualquer parte do mundo).

Compartilhamento: Os recursos utilizados por uma aplicação podem ser compartilhados com outras aplicações, não existindo a possibilidade de se criar uma "partição", por exemplo, para um determinado conjunto de tarefas de uma aplicação.

Múltiplos domínios administrativos: Tem a ver com o segundo aspecto citado (alta dispersão geográfica), ou seja, as grades podem agregar recursos de várias instituições.

Controle distribuído: Está relacionado ao aspecto anterior. Não existe uma única entidade que tenha poder sobre toda a grade.

Além dos aspectos expostos acima, Baker, Buyya e Laforenza acrescentam os seguintes [52]:

Autonomia: Os gerentes de recursos locais possuem políticas de acesso que devem ser respeitadas.

Escalabilidade: A grade pode crescer bastante, passando de um sistema com poucos recursos para um sistema com milhões de recursos.

Adaptabilidade: Os recursos de uma grade podem cair a qualquer momento devido à alguma falha no sistema. Os gerentes de aplicações e recursos devem prover mecanismos para tratar esse comportamento dinâmico e usar os recursos e serviços disponíveis eficientemente.

Outras características são abordadas por Németh e Sunderam [47]. Eles dizem que uma grade é um ambiente que oferece de forma **transparente** aos usuários, com **acesso restrito**, um **pool virtual de recursos**.

Pode-se fazer então, uma síntese de alguns requisitos necessários para um sistema ser considerado uma grade. Porém, é necessário ressaltar que a ausência de um dos requisitos citados não deve automaticamente desqualificar uma determinada plataforma como grade. São eles: heterogeneidade, alta dispersão geográfica, compartilhamento, múltiplos domínios administrativos, controle distribuído, autonomia, escalabilidade, adaptabilidade, *pool* virtual de recursos, acesso restrito aos recursos e transparência.

3.2.2 Tipos de Grades e Aplicações

Aplicações para grades computacionais podem ser encontradas em muitas áreas tais como física, astronomia, bioinformática, entre outras.

Reinefeld e Schintke apresentam três tipos de grades computacionais e, portanto, três tipos de aplicações que podem se beneficiar das grades [49]:

1. **Grades de informações baseadas em HTML (*HyperText Markup Language*):** As grades de informações, desde sua criação em 1990, tornaram-se um dos maiores sucessos na área da tecnologia da computação. Uma das razões para esse sucesso é o conceito de *hyperlink*, que é o mecanismo para referenciar outras páginas da *Web*. Através dos *hyperlinks* é bastante fácil a navegação pela *Web*;

2. **Grades de recursos:** Provêem mecanismos para coordenar o uso de recursos como computadores, dados, aplicações e instrumentos especiais de laboratório;

3. **Grades de serviços:** Fornecem serviços e aplicações independentes de suas localidades, implementação ou plataforma de *hardware*. Esses serviços são construídos sob recursos disponíveis nas grades de recursos. O que difere essas grades das grades de serviços é o nível de abstração que é disponibilizado para os usuários.

Outros tipos de aplicações podem ser encontrados na literatura. Foster apresentou cinco categorias de aplicações [51]:

1. **Supercomputação distribuída:** Nesta categoria se enquadram os problemas que precisam de muito poder de processamento, memória e armazenamento de dados e que não conseguem esses recursos em uma única máquina. Como exemplo de aplicação pode-se citar o *Distributed Interactive Simulation*;

2. **Computação de alto fluxo:** Grandes quantidades de tarefas independentes são escalonadas para muitos recursos (muitas vezes máquinas ociosas). Diferentemente da categoria de supercomputação distribuída, esse tipo de aplicação envolve poucas dependências (ou nenhuma) entre as tarefas. Um exemplo bastante conhecido é o projeto SETI@home [56, 57];

3. **Computação por demanda:** Esta categoria aborda as aplicações que se baseiam na utilização dinâmica de recursos por um curto período de tempo. Esses recursos podem ser *softwares*, repositórios de dados, dispositivos especializados como microscópios e telescópios, entre outros. Como exemplo tem-se o NetSolve [58, 59], que é um sistema baseado em Chamadas Remotas de Procedimento (RPC - *Remote Procedure Call*) cujo objetivo é a utilização de componentes de *software* e *hardware* remotos;

4. **Computação com uso intensivo de dados:** Este tipo de aplicação realiza uma síntese de novas informações de muitos dados de muitas fontes envolvendo o processamento de grandes quantidades de dados em bancos de dados distribuídos [60]. Foster cita como exemplo os sistemas de energia, onde vários *terabytes* de dados são gerados a cada dia, e são necessários formas de consultar estes dados armazenados de forma distribuída;

5. **Computação colaborativa:** Compreende as aplicações que apresentam trabalhos colaborativos entre múltiplos participantes através de ambientes compartilhados virtualmente, permitindo que esses participantes possam colaborar em tempo-real, desconsiderando a distância geográfica. Como exemplo tem-se o NICE, que usa computação colaborativa, permitindo que crianças criem e mantenham mundos virtuais [61].

Os tipos de aplicações apresentados acima, são baseadas em classes de aplicações para grades computacionais. O desenvolvimento de aplicações para grades computacionais requer uma atenção especial. Alguns métodos de programação de aplicações paralelas tradicionais não são apropriados para programação de aplicações para grades, pois envolvem um volume de comunicação entre as tarefas relativamente alto. Sendo assim, precisa-se de uma classe de aplicações alternativa que seja adequada ao comportamento de uma grade, ou seja, que observe principalmente a característica de distribuição geográfica dos componentes de uma grade.

A maioria dos sistemas de grades existentes baseiam-se nas seguintes classes de aplicações:

Parameter sweep: Essas aplicações possuem somente uma tarefa, que é executada várias vezes. A cada nova execução pode-se alterar os parâmetros da aplicação [62];

Bot - *Bag of Tasks*: Essas aplicações também são conhecidas como “saco de trabalho”. Trata-se de aplicações que possuem várias tarefas independentes uma das outras [63];

Workflow: São aplicações cujo as tarefas possuem uma determinada relação de ordem, seja por dependência de dados ou de controle das mesmas [64].

Entre as classes de aplicações citadas acima, as mais comumente utilizadas são as aplicações *parameter sweep* e Bot, pois vários problemas podem ser mapeados facilmente para essas aplicações. Pode-se citar alguns cenários como: *data-mining*, buscas massivas [65] (como quebra de chaves), simulações Monte Carlo, aplicações em bioinformática [66], processamento de imagens [67, 68] e cálculo de fractais (como o Mandelbrot).

4 O Modelo ICTM

Conforme comentado no Capítulo 1, o ICTM originou-se da proposta de Coblentz et al. [2], para uma metodologia de subdivisão confiável de áreas geológicas baseada na análise da monotonicidade da função que mapeia sua declividade. Esta proposta foi aperfeiçoada e acabou gerando um modelo baseado em autômatos celulares [3], que a partir da definição de uma malha que representa uma certa região, executa uma análise bidimensional do sinal da declividade do relevo desta região. Este modelo utiliza regras locais para criação e categorização das sub-regiões, apresentando a situação relativa de cada sub-região com relação a área total, de acordo com os estados assumidos pelas células da malha.

A generalização deste modelo baseado em autômatos celulares, foi proposta em [4]. Trata-se do objeto do presente trabalho, o modelo ICTM, que utiliza o conceito de tesselações e não mais de autômatos celulares. Mais precisamente, pretendía-se utilizar o conceito de autômatos celulares para embasar o processo de categorização proposto. Isso permitiria não apenas uma análise estática das características do espaço em um momento dado, mas também possibilitaria o avanço na direção de uma simulação dos aspectos dinâmicos desse espaço, em um intervalo de tempo.

Entretanto, a complexidade dos processos de modelagem e análise que foram sendo encontrados, e a possibilidade de extensão do modelo com mecanismos de descoberta de conhecimentos (fora do escopo deste trabalho), levou à escolha da modelagem baseada em tesselações, que se limita a uma análise estática do espaço.

As tesselações podem ser consideradas como a estrutura estática subjacente aos autômatos celulares [69]. Ou seja, assim como o autômato celular, as tesselações são malhas de células idênticas e discretas, onde cada uma destas células têm seu estado determinado localmente a partir dos estados da sua vizinhança de células.

Portanto, as evoluções destes estados, dadas pelas regras de transição de estados dos autômatos celulares, não estão presentes nas tesselações, que se limitam à geração de um único valor de estado fixo, para suas células.

Essa restrição conceitual da idéia acabou possibilitando que fosse tentada a utilização imediata do modelo em uma área aplicada bem determinada, qual seja, a do processamento de imagens geográficas, mais especificamente, a análise digital de terrenos. Isso tornou-se real com a aprovação dos projetos ACI e FMC2 citados no Capítulo 1.

As seções seguintes apresentam a formalização, as características, o funcionamento do processo de categorização, algumas considerações sobre implementação e validação, e a aplicabilidade do modelo ICTM. É importante ressaltar que a formalização do modelo foi definida por Aguiar e Costa [4] e foi reproduzida neste documento para facilitar o entendimento do mesmo. No entanto, um exemplo hipotético é apresentado neste trabalho, em conjunto com a formalização, com o intuito de simplificar ainda mais a compreensão das definições matemáticas do ICTM.

4.1 Modelagem Inicial: Topo-ICTM

A primeira versão do ICTM, denominada Topo-ICTM, surgiu a partir do estudo do trabalho descrito em [2], que apresentou um método baseado em uma análise unidimensional para subdividir áreas geofísicas em sub-regiões de monotonicidade, considerando apenas uma direção.

O modelo Topo-ICTM é um ICTM bidimensional de uma camada, ou seja, trata-se de um modelo baseado em tesselações que executa uma análise bidimensional da declividade, usando regrais locais para a criação e categorização das sub-regiões, de acordo com a situação relativa de cada sub-região à área total, a partir dos estados assumidos pelas células. O Topo-ICTM permite a análise da variação do sinal da declividade da função que mapeia a topografia (por isso o nome **Topo**-ICTM) de uma dada região geográfica, subdividindo esta região em sub-regiões que apresentam o mesmo comportamento com respeito à declividade do relevo.

Cada região é dita pertencer a uma dada categoria de declividade de acordo com o sinal (positivo, negativo, nulo) da declividade da função relevo[1]. Uma aplicação imediata é na Geofísica, onde uma subdivisão apropriada de áreas geográficas em segmentos apresentando características similares é freqüentemente conveniente [2].

Além disso, a análise pode ser facilmente refinada, pela repetição do procedimento para focar uma sub-região de uma certa categoria de declividade, ou mudando os parâmetros de entrada (número de células da tesselação, etc.), ou considerando um raio de vizinhança maior, por exemplo.

4.1.1 Formalização do Modelo Topo-ICTM

Esta seção apresenta a formalização do modelo categorizador intervalar baseado em tesselações para a categorização de regiões geográficas baseado na declividade topográfica, formalizado nos termos de operações sobre matrizes, chamado de Topo-ICTM. A Figura 7 exibe esquematicamente as etapas de funcionamento do modelo.

Matriz absoluta e matriz relativa

Os dados de entrada do modelo são extraídos a partir de imagens de satélite da região topográfica a ser analisada, onde as alturas são dadas nos pontos referenciados pelas coordenadas de latitude e longitude. Esta região geográfica é representada por uma tesselação regular que é determinada pela subdivisão da área total em subáreas retangulares suficientemente pequenas, cada uma representada por uma célula da tesselação. Esta subdivisão é feita de acordo com o tamanho da célula estabelecida pelo analista geofísico e está diretamente associada ao grau de refinamento da tesselação.

Definição 4.1.1 *Uma tesselação é uma matriz M con n_r linhas e n_c colunas. A entrada na x-ésima linha e na y-ésima coluna é chamada de célula xy da tesselação M.*

[1] Neste trabalho, utiliza-se o termo "declividade" para significar o sinal da função de declividade.

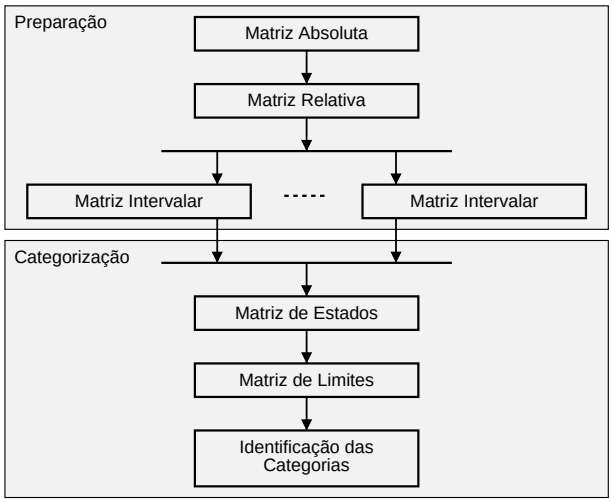

Figura 7 – Etapas do processo de categorização.

Na análise topográfica, freqüentemente se tem abundância de dados, a maioria destes são geofisicamente irrelevantes. Então, toma-se, para cada subdivisão, o valor médio das alturas nos pontos providos pelas imagens de satélites, que são as entradas da matriz espectral da tesselação M:

Definição 4.1.2 *A matriz espectral de uma tesselação M é a matriz* $n_r \times n_c$ $M^{abs.spec} = [m_{xy}^{abs.spec}]$*, onde a entrada* $m_{xy}^{abs.spec}$ *é o valor absoluto da altura média dos pontos representados pela célula xy da tesselação M.*

Daqui pra frente será utilizado um exemplo hipotético (matriz com 8 linhas e 16 colunas) para demonstrar a formalização do modelo. A Tabela 1 exibe um exemplo de matriz absoluta onde cada célula possui um único valor.

Tabela 1 – Exemplo de uma matriz absoluta.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	2	2	2	2	2	3	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	3	2	2	2
2	1	1	1	1	1	1	1	1	2	2	2	2	3	2	2	2
3	1	1	1	1	1	1	1	1	2	2	2	2	3	2	2	2
4	1	1	1	1	1	1	1	2	2	2	2	2	3	2	2	2
5	1	1	1	1	1	1	1	2	2	2	2	2	3	2	2	2
6	1	1	1	1	1	1	2	2	2	2	2	2	3	2	2	2
7	1	1	1	1	1	1	2	2	2	2	2	2	3	2	2	2

A fim de simplificar os dados da matriz espectral $M^{abs.spec}$, os valores absolutos de suas entradas são normalizados, dididindo-os pelo maior de seus valores m_{max}. A Tabela 2 apresenta a matriz relativa resultante.

Definição 4.1.3 *A matriz espectral relativa $M^{rel.spec}$ é definida como a matriz $n_r \times n_c$ dada por $M^{rel.spec} = \frac{M^{abs.spec}}{M_{max}}$.*

Tabela 2 – Exemplo de uma matriz relativa.

	0 . . . 5	6	7	8 . . . 11	12	13 . . . 15
0	0.33	0.33	0.67	0.67	1.00	0.67
1	0.33	0.33	0.67	0.67	1.00	0.67
2	0.33	0.33	0.33	0.67	1.00	0.67
3	0.33	0.33	0.33	0.67	1.00	0.67
4	0.33	0.33	0.67	0.67	1.00	0.67
5	0.33	0.33	0.67	0.67	1.00	0.67
6	0.33	0.67	0.67	0.67	1.00	0.67
7	0.33	0.67	0.67	0.67	1.00	0.67

Matrizes espectrais intervalares

As alturas são medidas bastante exatas em um modelo digital de elevação (DEM), então os únicos erros nos valores m_{xy} vêm da discretização da área em termos do conjunto discreto das células da tesselação. É desejável conhecer os valores da função que mapeia o relevo $h_{\xi\upsilon}$ para todo ξ e υ, mas somente os valores $h_{xy} = m_{xy}^{rel.spec} = \frac{m_{xy}^{abs.spec}}{m_{max}}$ para 11,..., $1n_r$,..., n_c1,..., n_cn_r, determinados pela divisão da região em n_rn_c células, são utilizados efetivamente nos cálculos.

Seguindo a abordagem apresentada em [2], baseada na matemática intervalar [70,71], utilizam-se intervalos pra controlar os erros associados aos valores das células. Para cada $\xi\upsilon$, significando que $\xi\upsilon$ pertence ao mesmo segmento de área que xy.

Para um y fixo, quando $\xi > x$, o ponto xy é ainda o mais próximo até que seja alcançado o ponto médio $x_{mid}y = \frac{(x+(x+1))}{2}y$ entre xy e $(x+1)y$. É razoável assumir que o maior erro de aproximação possível $|m_{xy}^{rel.spec} - h_{\xi y}|$ para tais pontos é obtido quando a distância entre xy e ξy é a maior, ou seja, quando $\xi y = x_{mid}y$. Neste caso, o erro de aproximação é igual a $|h_{x_{midy}} - m_{xy}^{rel.spec}|$.

Lema 4.1.1 *Para um y fixo, se $\xi > x$, então o erro de aproximação ϵ é limitado por*

$$0.50.|m_{(x+1)y}^{rel.spec} - m_{xy}^{rel.spec}|.$$

Prova: Se os pontos xy e $(x+1)y$ pertencem ao mesmo segmento de área, então a dependência de $n_{\xi y}$ sobre ξy seria razoavelmente suave para $\xi \in [x,(x+1)]$. Portanto, em um intervalo pequeno $[x,(x+1)]$, pode-se, com razoável exatidão, ignorar os termos quadráticos e de mais alta ordem na expansão de $h_{(\xi+\Delta\xi)y}$ e assim, aproximar $h_{\xi y}$ por uma função linear. Para uma função linear $\xi \mapsto h_{\xi y}$, a diferença $h_{x_{midy}} - m_{xy}^{rel.spec}$ é igual à metade da diferença $m_{(x+1)y}^{rel.spec} - m_{xy}^{rel.spec}$. Por outro lado, se os pontos xy e $(x+1)y$ pertencem a segmentos diferentes, então a dependência $h_{\xi y}$ deveria apresentar alguma não suavidade, e é razoável esperar que a diferença $m_{(x+1)y}^{rel.spec} - m_{xy}^{rel.spec}$ é muito maior que o erro de aproximação. Em ambos casos, o erro de aproximação ϵ é limitado por $0.50.|m_{(x+1)y}^{rel.spec} - m_{xy}^{rel.spec}|$.

Lema 4.1.2 *Para um y fixo, se $\xi < x$, então o erro de aproximação ϵ é limitado por*

$$0.50.|m_{xy}^{rel.spec} - m_{(x-1)y}^{rel.spec}|.$$

Proposição 4.1.1 *Para o erro de aproximação ϵ_x,*

$$\epsilon_x \leq \Delta_x = 0.5.min(|m_{xy}^{rel.spec} - m_{(x-1)y}^{rel.spec}|, |m_{(x+1)y}^{rel.spec} - m_{xy}^{rel.spec}|).$$

Prova: Segue dos Lemas 4.1.1 e 4.1.2.

Como um resultado, considerado um dado y, juntamente com os valores centrais $m_{xy}^{rel.spec}$, para cada x, têm-se os intervalos $m_{xy}^{x[]}$ contendo todos os valores possíveis de $h_{\xi y}$, for $x - \frac{1}{2} \leq \xi \leq x + \frac{1}{2}$.

Corolário 4.1.1 *Considerando um y fixo, para cada x, se $x - \frac{1}{2} \leq \xi \leq x + \frac{1}{2}$, então $h_{\xi y} \in m_{xy}^{x[]} = [m_{xy}^{x^-}, m_{xy}^{x^+}]$, onde $m_{xy}^{x^-} = m_{xy}^{rel.spec} - \Delta_x$ e $m_{xy}^{x^+} = m_{xy}^{rel.spec} + \Delta_x$.*

Usando uma argumentação análoga, considerando um x fixo, é possível concluir que:

Proposição 4.1.2 *Para o erro de aproximação ϵ_y,*

$$\epsilon_y \leq \Delta_y = 0.5.min(|m_{xy}^{rel.spec} - m_{x(y-1)}^{rel.spec}|, |m_{x(y+1)}^{rel.spec} - m_{xy}^{rel.spec}|).$$

Corolário 4.1.2 *Considerando um x fixo, para cada y, se $y - \frac{1}{2} \leq \upsilon \leq y + \frac{1}{2}$, $h_{x\upsilon} \in m_{xy}^{y[]} = [m_{xy}^{y^-}, m_{xy}^{y^+}]$, onde $m_{xy}^{y^-} = m_{xy}^{rel.spec} - \Delta_y$ e $m_{xy}^{y^+} = m_{xy}^{rel.spec} + \Delta_y$.*

Definição 4.1.4 *Se $m_{xy}^{x^{+-}} = m_{xy}^{rel.spec} + -\Delta_i$ e $m_{xy}^{y^{+-}} = m_{xy}^{rel.spec} + -\Delta_j$, então as matrizes espectrais intervalares $M^{x[]}$ e $M^{y[]}$, associadas à matriz espectral relativa $M^{rel.spec}$, são definidas pelas matrizes $n_r \times n_c$ intervalares*

$$M^{x[]} = [m_{xy}^{x[]}] = [[m_{xy}^{x^-}, m_{xy}^{x^+}]], M^{y[]} = [m_{xy}^{y[]}] = [[m_{xy}^{y^-}, m_{xy}^{y^+}]].$$

As Tabelas 3 e 4 apresentam as matrizes intervalares para o exemplo hipotético.

Tabela 3 – Exemplo de uma matriz intervalar no eixo X.

	0 . . . 5	6	7	8 . . . 11	12	13 . . . 15
0	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[0.83,1.17]	[0.67,0.67]
1	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[0.83,1.17]	[0.67,0.67]
2	[0.33,0.33]	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[0.83,1.17]	[0.67,0.67]
3	[0.33,0.33]	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[0.83,1.17]	[0.67,0.67]
4	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[0.83,1.17]	[0.67,0.67]
5	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[0.83,1.17]	[0.67,0.67]
6	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[0.67,0.67]	[0.83,1.17]	[0.67,0.67]
7	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[0.67,0.67]	[0.83,1.17]	[0.67,0.67]

Tabela 4 – Exemplo de uma matriz intervalar no eixo Y.

	0 . . . 5	6	7	8 . . . 11	12	13 . . . 15
0	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[1.00,1.00]	[0.67,0.67]
1	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[1.00,1.00]	[0.67,0.67]
2	[0.33,0.33]	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[1.00,1.00]	[0.67,0.67]
3	[0.33,0.33]	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[1.00,1.00]	[0.67,0.67]
4	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[1.00,1.00]	[0.67,0.67]
5	[0.33,0.33]	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[1.00,1.00]	[0.67,0.67]
6	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[0.67,0.67]	[1.00,1.00]	[0.67,0.67]
7	[0.33,0.33]	[0.67,0.67]	[0.67,0.67]	[0.67,0.67]	[1.00,1.00]	[0.67,0.67]

Matriz de estados e os registradores de monotonicidade

Prossegue-se com a categorização da declividade inspirada no trabalho de Coblentz et al. [2]. Assume-se inicialmente que as funções de aproximação do relevo induzidas pelo modelo baseado em tesselações são funções lineares por partes. Modela-se todo processo como um tipo de problema de satisfação de restrições, onde o modelo baseado em tesselação está a cargo de encontrar uma função de aproximação linear, também por partes, do relevo (e o conjunto de pontos limites entre as regiões resultantes) que satisfaz as restrições impostas pela matriz espectral intervalar. Para simplificar o espaço de soluções, toma-se uma abordagem qualitativa das funções de aproximação do relevo, agrupando-as em classes de equivalência de acordo com o sinal de sua declividade (positiva, negativa, nula), assim o modelo constrói uma solução qualitativa simples para o problema de satisfação de restrições, isto é, a classe das funções de aproximação compatíveis com as restrições da matriz espectral intervalar. Procede-se da seguinte forma:

Proposição 4.1.3 *Sejam $M^{x[]}$ e $M^{y[]}$ as matrizes espectrais intervalares. Para um dado xy, se:*

1. $m^{x^+}_{xy} \geq m^{x^-}_{(x+1)y}$*, então existe uma função de aproximação do relevo não crescente entre* xy *e* $(x+1)y$ *(direção oeste-leste);*

2. $m^{x^-}_{(x-1)y} \leq m^{x^+}_{xy}$*, então existe uma função de aproximação do relevo não decrescente entre* $(x-1)y$ *e* xy *(direção oeste-leste);*

3. $m^{y^+}_{xy} \geq m^{y^-}_{x(y+1)}$*, então existe uma função de aproximação do relevo não crescente entre* xy *e* $x(y+1)$ *(direção norte-sul);*

4. $m^{y^-}_{x(y-1)} \leq m^{y^+}_{xy}$*, então existe uma função de aproximação do relevo não decrescente entre* $x(y-1)$ *e* xy *(direção norte-sul).*

Prova: Apresenta-se um esquema da prova. No item 1, toma-se, por exemplo, $\mu_{xy} = m^{x^+}_{xy}$, $\mu_{(x+1)y} = m^{x^-}_{(x+1)y}$ e se usa uma interpolação linear para definir os valores μ_{ky} para $x < k < x+1$. As provas de 2 a 4 são similares.

Para cada célula, são definidos quatro registradores direcionados de declividade – *reg.e (leste), reg.w (oeste), reg.s (sul) e reg.n (norte)* – indicando os sinais de declividade admissí-

veis da função que aproxima a função do relevo em uma destas direções, levando em consideração os valores das suas células vizinhas. A análise da declividade é feita de acordo com a Proposição 4.1.3.

Definição 4.1.5 *Um registrador de declividade da célula xy é uma tupla reg=(reg.e, reg.w, reg.s, reg.n), cujos valores dos registradores direcionados de declividade são dados por:*

1. *Para células que não estão na borda da tesselação:*

$$reg.e = \begin{cases} 0 & \textit{se a condição do item 1 da Proposição 4.1.3 é verdadeira;} \\ 1 & \textit{caso contrário.} \end{cases}$$

$$reg.w = \begin{cases} 0 & \textit{se a condição do item 2 da Proposição 4.1.3 é verdadeira;} \\ 1 & \textit{caso contrário.} \end{cases}$$

$$reg.s = \begin{cases} 0 & \textit{se a condição do item 3 da Proposição 4.1.3 é verdadeira;} \\ 1 & \textit{caso contrário.} \end{cases}$$

$$reg.n = \begin{cases} 0 & \textit{se a condição do item 4 da Proposição 4.1.3 é verdadeira;} \\ 1 & \textit{caso contrário.} \end{cases}$$

2. *Para as células das bordas leste, oeste, sul e norte: reg.e = 0, reg.w = 0, reg.s = 0, reg.n = 0, respectivamente. Os demais registradores direcionados de declividade são determinados usando 1.*

Observa-se que, para cada célula, existe uma, e somente uma, possibilidade de atribuição de valores aos registradores de declividade.

Definição 4.1.6 *A matriz dos registradores de declividade é definida como uma matriz $n_r \times n_c$ denotada por $M^{reg} = [m_{xy}^{reg}]$, onde a entrada na x-ésima linha e na y-ésima coluna é o valor do registrador de declividade da célula correspondente.*

Corolário 4.1.3 *Considerando a direção oeste-leste, uma função de aproximação do relevo m_{xy} é:*

1. *estritamente crescente entre xy e $(x+1)y$ se $m_{xy}^{reg.e} = 1$ (neste caso, $m_{(x+1)y}^{reg.w} = 0$);*
2. *estritamente decrescente entre xy e $(x+1)y$ se $m_{(x+1)y}^{reg.w} = 1$ (neste caso, $m_{xy}^{reg.e} = 0$);*
3. *constante entre xy e $(x+1)y$ se $m_{xy}^{reg.e} = 0$ e $m_{(x+1)y}^{reg.w} = 0$.*

Resultados similares do Corolário 4.1.3 valem também para a direção norte-sul.

Definição 4.1.7 *Sejam $W_{reg.e} = 1, W_{reg.s} = 2, W_{reg.w} = 4$ e $W_{reg,n} = 8$ os pesos a serem associados aos registradores de declividade. A matriz de estados é definida como uma matriz $n_r \times nc$ dada por $M^{state} = [m_{xy}^{state}]$, cuja entrada na x-ésima linha e na y-ésima coluna é o valor correspondente ao do estado da célula calculado como o valor da codificação binária dos registradores de declividade correspondentes, da seguinte forma*

$$m_{xy}^{state} = W_{reg.e} \times m_{xy}^{reg.e} + W_{reg.s} \times m_{xy}^{reg.s} + W_{reg.w} \times m_{xy}^{reg.w} + W_{reg.n} \times m_{xy}^{reg.n}.$$

Assim, para um dado xy, a célula correspondente pode assumir um, e apenas um, dos estados apresentados na Figura 8, representado pelo valor $m_{xy}^{state} = 0..15$. A Tabela 5 ilustra a matriz de estados para o exemplo hipotético.

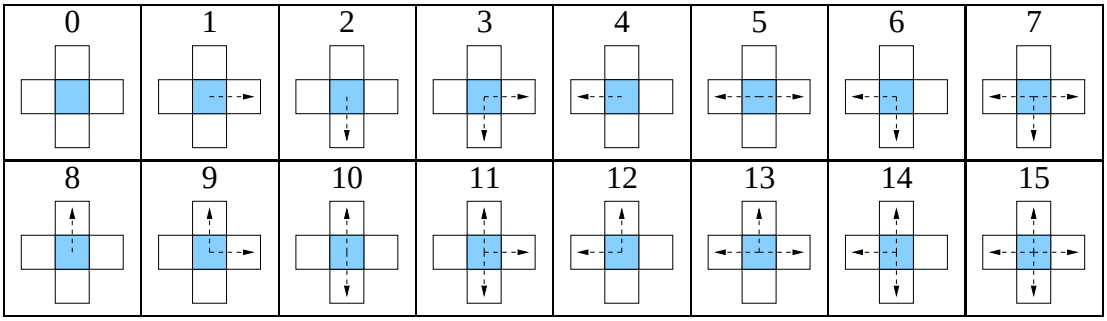

Figura 8 – Esquema de todos os valores possíveis dos estados da célula.

Tabela 5 – Exemplo de uma matriz de estados.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	1	0	0	0	0	1	0	4	0	0
1	0	0	0	0	0	0	1	0	0	0	0	1	0	4	0	0
2	0	0	0	0	0	0	0	9	0	0	0	1	0	4	0	0
3	0	0	0	0	0	0	0	3	0	0	0	1	0	4	0	0
4	0	0	0	0	0	0	1	0	0	0	0	1	0	4	0	0
5	0	0	0	0	0	0	3	0	0	0	0	1	0	4	0	0
6	0	0	0	0	0	1	0	0	0	0	0	1	0	4	0	0
7	0	0	0	0	0	1	0	0	0	0	0	1	0	4	0	0

Matriz de limites e as sub-regiões de declividade constante

Uma célula é definida como limítrofe quando a função do relevo muda sua declividade, apresentando pontos críticos (máximo, mínimo ou pontos de inflexão). Para se identificar tais células limítrofes, utiliza-se um registrador de limite associado àquela célula. As células da borda de toda região são consideradas células limítrofes.

Definição 4.1.8 *A matriz de limites é definida como a matriz $n_r \times n_c$ denotada por $M^{limit} = [m_{xy}^{limit}]$, onde a entrada na x-ésima linha e na y-ésima coluna é determinada como $m_{xy}^{limit} = 0$, se uma das condições listadas na Tabela 6 é verificada; 1, caso contrário.*

Tabela 6 – Condições das células xy não limítrofes.

Id	Condições
1	$m^{reg.e}_{(x-1)y} = m^{reg.e}_{xy} = 1$
2	$m^{reg.w}_{xy} = m^{reg.w}_{(x+1)y} = 1$
3	$m^{reg.e}_{(x-1)y} = m^{reg.e}_{xy} = m^{reg.w}_{xy} = m^{reg.w}_{(x+1)y} = 0$
4	$m^{reg.s}_{x(y-1)} = m^{reg.s}_{xy} = 1$
5	$m^{reg.n}_{xy} = m^{reg.n}_{x(y+1)} = 1$
6	$m^{reg.s}_{x(y-1)} = m^{reg.s}_{xy} = m^{reg.n}_{xy} = m^{reg.n}_{x(y+1)} = 0$

Analisando a matriz de limites é fácil de detectar a existência de configurações de relevo conhecidas (por exemplo, os esquemas mostrados na Figura 9). A presença de células limítrofes permitem a subdivisão da área total em categorias de declividade.

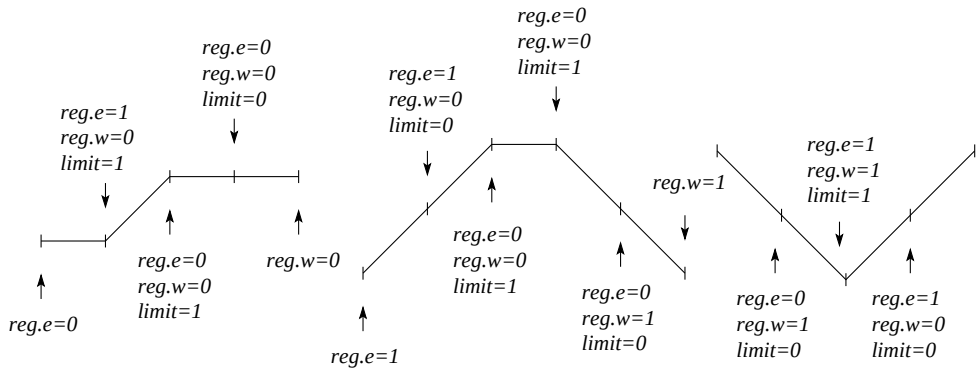

Figura 9 – Esquemas das células limítrofes.

Definição 4.1.9 *A sub-região de declividade constante associada à célula não limítrofe xy, denotada por SR_{xy}, é definida indutivamente como segue:*

1. $xy \in SR_{xy}$*;*
2. *Se $x'y' \in SR_{xy}$, então todas suas células vizinhas não limítrofes também pertencem a SR_{xy}.*

Observe que $SR_{xy} = SR_{x'y'}$ se, e somente se, $x'y' \in SR_{xy}$ (resp., $xy \in SR_{x'y'}$). Como resultado final, toda a área torna-se dividida em categorias de declividade bem definidas.

A Definição 4.1.9 induz a um algoritmo recursivo similar aos comumente utilizados para preenchimento de polígonos. A Tabela 7 apresenta a matriz de limites referente ao exemplo hipotético desenvolvido até o momento, destacando as categorias encontradas na "região" analisada.

Tabela 7 – Exemplo de uma matriz de limites.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	1	1	0	0	0	1	1	1	0	1
2	1	0	0	0	0	0	0	1	1	0	0	1	1	1	0	1
3	1	0	0	0	0	0	0	1	1	0	0	1	1	1	0	1
4	1	0	0	0	0	0	1	1	0	0	0	1	1	1	0	1
5	1	0	0	0	0	0	1	1	0	0	0	1	1	1	0	1
6	1	0	0	0	0	1	1	0	0	0	0	1	1	1	0	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

4.1.2 Considerações sobre o Modelo Topo-ICTM

O Topo-ICTM considera apenas uma característica, neste caso, a declividade da função que mapeia o relevo da região considerada, fazendo uma análise bidimensional, conforme a latitude e a longitude.

A dimensão $n_r \times n_c$ da tesselação pode ser arbitrária ou escolhida de acordo com um critério específico estabelecido pela aplicação. De qualquer forma, a categorização obtida pode ser refinada definindo-se outra dimensão para a tesselação ou tomando cada sub-região como uma nova área a ser analisada separadamente.

A análise pode ser feita até ser alcançado um número conveniente de subdivisões, caracterizando, assim, o dinamismo do modelo. A formalização usando matrizes de registradores permite que a informação armazenada nestes seja facilmente acessada a qualquer momento do processamento, devido a indexação dos elementos nas matrizes.

4.2 O modelo Geral: ICTM

Esta seção apresenta a definição do ICTM em sua forma mais geral, qual seja, um modelo geral, baseado em tesselações, capaz de produzir uma categorização confiável de sub-regiões de um espaço de características geométricas, podendo analisar múltiplas características conhecidas em suficientemente muitos pontos.

A categorização determinada por cada característica é efetuada em uma camada do modelo, gerando diferentes subdivisões da região analisada. Por exemplo, uma região pode ser analisada conforme sua topografia, vegetação, demografia, dados econômicos, etc.

O conjunto de pontos analisados pode pertencer a um espaço multi-dimensional, determinando, assim, o caráter multi-dimensional de cada camada.

Uma categorização global pode ser alcançada, a partir da categorização de cada camada, mediante um procedimento de projeção. Esta categorização global determinará uma subdivisão mais confiável e com maior significância, combinando as análises efetuadas para cada característica.

Este tipo de projeção permite análises bastante interessantes sobre a dependência mútua

destas características.

4.2.1 ICTM **Multi-camada**

Cada característica da aplicação está representada em uma camada do modelo ICTM. Assim, devido às características paralelas, as subdivisões em cada camada também ocorrerão de forma independente.

O primeiro passo em direção à expansão do modelo ICTM é a definição do conceito de tesselações com multi-camadas. Na Figura 10, cada camada bidimensional é representada por um rótulo alfanumérico.

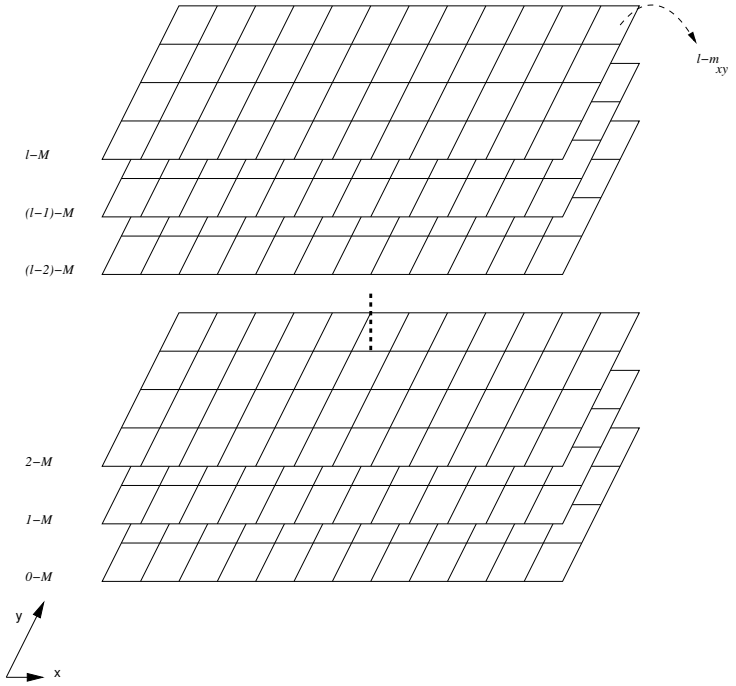

Figura 10 – Visão das multi-camadas bidimensionais do ICTM.

Definição 4.2.1 *Sejam M uma tesselação $n_c \times n_r$ e $l \in \mathbb{N}$, uma tesselação multi-camada $L - M$ é a estrutura (M, l). A entrada na l-ésima camada, x-ésima linha e na y-ésima coluna é denotada por $l - m_{xy}$.*

Definição 4.2.2 (Tesselação Multi-camada Rotulada) *Sejam M uma tesselação $n_c \times n_r$ e $l \in \sum^*$, uma tesselação multi-camada $L - M$ é a estrutura (M, l). A entrada na camada rotulada por l, x-ésima linha e na y-ésima coluna é denotada por $l - m_{xy}$.*

O rótulo permite uma identificação mais intuitiva da característica que está sendo analisada. Entretanto, a utilização de rótulos pode descaracterizar a idéia intuitiva de camadas (uma camada sobre a outra) e passar a apresentar cada camada como um quadro de visualização distinto do outro.

A camada de número 0 de uma tesselação M, denotada por $0-M$ ou rotulada como $\pi-M$, é considerada a camada base da projeção da demais camadas.

Assim, todas as células da tesselação ganham um novo indexador, que é a referência à camada desejada.

As Definições 4.1.2, 4.1.3, 4.1.4, 4.1.5, 4.1.6 e 4.1.8 e a Tabela 6, apresentadas na Seção 4.1.1, são reescritas com o identificador da camada a qual pertencem e são apresentadas a seguir.

Definição 4.2.3 *A matriz espectral da camada l de uma tesselação M é a matriz $n_r \times n_c$ $l - M^{abs.spec} = [l - m_{xy}^{abs.spec}]$, onde a entrada $l - m_{xy}^{abs.spec}$ é o valor absoluto da altura média dos pontos representados pela célula xy na camada l da tesselação M.*

Definição 4.2.4 *A matriz espectral relativa da camada l denotada por $l - M_{rel.spec}$ é definida como a matriz $n_r \times n_c$ dada por $l - M^{rel.spec} = \frac{l-M^{abs.spec}}{l-m_{xy}}$.*

Definição 4.2.5 *Se $l - m_{xy}^{x^{+-}} = l - m_{xy}^{rel.spec} + -\Delta_i$ e $l - m_{xy}^{y^{+-}} = l - m_{xy}^{rel.spec} + -\Delta_j$, então as matrizes espectrais intervalares da camada $l, l - M^{x^{[]}}$ e $l - M^{y^{[]}}$, associadas à matriz espectral relativa $l - M^{rel.spec}$, são definidas pelas matrizes $n_r \times n_c$ intervalares*

$$l - M^{x^{[]}} = [l - m_{xy}^{x^{[]}}] = [[l - m_{xy}^{x^-}, l - m_{xy}^{x^+}]], l - M^{y^{[]}} = [l - m_{xy}^{y^{[]}}] = [[l - m_{xy}^{y^-}, l - m_{xy}^{y^+}]].$$

Definição 4.2.6 *A matriz dos registradores de declividade da camada l é definida como uma matriz $n_r \times n_c$ denotada por $l - M^{reg} = [l - m_{xy}^{reg}]$, onde a entrada na x-ésima linha e na y-ésima coluna da camada l é o valor do registrador de declividade da célula correspondente.*

Definição 4.2.7 *A matriz de estados da camada l é definida como uma matriz $n_r \times nc$ dada por $l - M^{state} = [l - m_{xy}^{state}]$, cuja entrada na x-ésima linha e na y-ésima coluna da camada é o valor correspondente ao do estado da célula, calculado como o valor da codificação binária dos registradores de declividade correspondentes, da seguinte forma*

$$l - m_{xy}^{state} = W_{reg.e}\, X\, l - m_{xy}^{reg.e} + W_{reg.s} \times l - m_{xy}^{reg.s} + W_{reg.w} \times l - m_{xy}^{reg.w} + W_{reg.n} \times l - m_{xy}^{reg.n}.$$

Tabela 8 – Condições das células xy não limítrofes da camada l.

Id	Condições
1	$l - m_{(x-1)y}^{reg.e} = l - m_{xy}^{reg.e} = 1$
2	$l - m_{xy}^{reg.w} = l - m_{(x+1)y}^{reg.w} = 1$
3	$l - m_{(x-1)y}^{reg.e} = l - m_{xy}^{reg.e} = l - m_{xy}^{reg.w} = l - m_{(x+1)y}^{reg.w} = 0$
4	$l - m_{x(y-1)}^{reg.s} = l - m_{xy}^{reg.s} = 1$
5	$l - m_{xy}^{reg.n} = l - m_{x(y+1)}^{reg.n} = 1$
6	$l - m_{x(y-1)}^{reg.s} = l - m_{xy}^{reg.s} = l - m_{xy}^{reg.n} = l - m_{x(y+1)}^{reg.n} = 0$

Definição 4.2.8 *A matriz de limites da camada l é definida como a matriz $n_r \times n_c$ denotada por $l - M^{limit} = [l - m_{xy}^{limit}]$, onde a entrada na x-ésima linha e na y-ésima coluna da camada l é determinada como $l - m_{xy}^{limit} = 0$, se uma das condições listadas na Tabela 8 é verificada; 1, caso contrário.*

Na próxima seção apresenta-se a reestruturação do modelo ICTM para a multi-dimensionalidade da tesselação.

4.2.2 ICTM Multi-dimensional

Até o momento o modelo ICTM manipula tesselações bidimensionais, fato esse que permite sua aplicação em uma gama considerável de situações práticas. Nesta seção, apresenta-se a generalização do número de dimensões da tesselação. Assim, possibilita-se a análise de características que necessitam mais de duas coordenadas para serem definidas ou referenciadas. A Figura 11 apresenta as várias camadas de mesma dimensão.

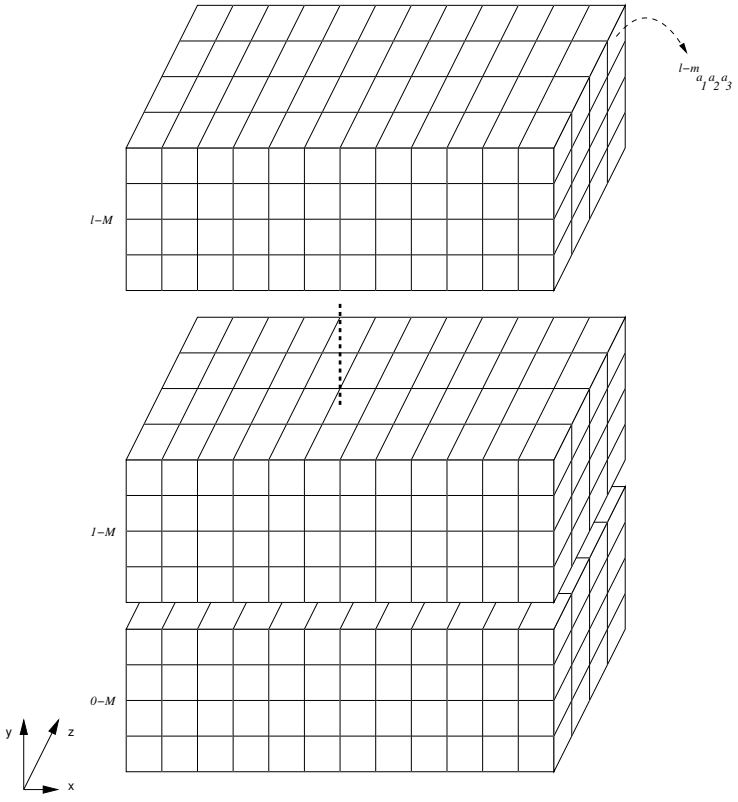

Figura 11 – Visão das multi-camadas multi-dimensionais do ICTM ($n = 3$).

Definição 4.2.9 *Uma tesselação de dimensão n é uma matriz M com a_1 x a_2 x a_3 x ...x a_n células.*

Definição 4.2.10 *Sejam M uma tesselação n-dimensional e $l \in N$. Uma tesselação multi-camada $L - M$ é a estrutura (M, l) e uma entrada na l da tesselação M é denotada por $l - m_{a1...an}$.*

Uma das alterações do modelo é a referência à célula (Definição 4.2.10). Neste caso, optou-se por referenciar uma célula pela sua coordenada no sistema multi-dimensional. Portanto, a célula $l - m_{2,3,4,5}$ está posicionada: a duas unidades da origem no eixo a_1, a três unidades da origem no eixo a_2, a quatro unidades da origem no eixo a_3 e a cinco unidades da origem no eixo a_4.

Definição 4.2.11 (Tesselação Rotulada) *Sejam M uma tesselação n-dimensional a_1 x...x a_n e $l \in \sum^*$. Uma tesselação multi-camada $L - M$ é a estrutura (M, l) e uma entrada na camada rotulada por l da tesselação M é denotada por $l - m_{a_1...a_n}$.*

Definição 4.2.12 *A matriz espectral da camada l de uma tesselação M n-dimensional é a matriz a_1x...x a_n, denotada por $l - M^{abs.spec=[l-m^{abs.spec}_{a_1...a_n}]}$, onde a entrada $l - m^{abs.spec}_{a_1...a_n}$ é o valor absoluto da altura média dos pontos representados pela célula $a_1...a_n$ na camada l da tesselação M.*

Definição 4.2.13 *A matriz espectral relativa da camada l denotada por $l - M_{rel.spec}$ é definida como a matriz a_1 x...xa_n dada por $l - M^{rel.spec} = \frac{l-M^{abs.spec}}{l-m_{max}}$.*

As proposições 4.1.1 e 4.1.2 da Seção 4.1.1 são reescritas na seguinte proposição:

Proposição 4.2.1 *O erro de aproximação ϵ_{ai}, em uma tesselação multi-dimensional $\epsilon_{a_i} \leq \Delta_{a_i} = 0.5.min\ (|l - m^{rel.spec}_{a_1...a_i...a_n} - l - m^{rel.spec}_{a_1...a_{i-1}...a_n}|, |l - m^{rel.spec}_{a_1...a_{i+1}...a_n} - l - m^{rel.spec}_{a_1...a_i...a_n}|)$, para $i = l...n$.*

Definição 4.2.14 *Se $l - m^{a^{i+-}}_{a_1...a_n} = l - m^{rel.spec}_{a_1...a_n} + - \Delta_{a_i}$, então as matrizes espectrais intervalares da camada $l, l - M^{a^{[]}_n}$, associadas à matriz espectral relativa $l - M^{rel.spec}$, são definidas pelas matrizes a_1 x...x a_n intervalares*

$$l - M^{a^{[]}_i} = [l - m^{a^{[]}_i}_{a_1...a_n}] = [[l - m^{a^-_i}_{a_1...a_n}, l - m^{a^+_i}_{a_1...an}]], para\ i = 1...n.$$

No caso multi-dimensional existe um ϵ (erro de aproximação) para cada eixo do sistema de coordenadas, logo tem-se uma matriz espectral intervalar para cada eixo. Ou seja, se o sistema tem n-dimensões, então existirão n matrizes espectrais intervalares indicando os erros de aproximação para cada camada do modelo.

A seguir, apresenta-se a re-leitura da Proposição 4.1.3 da Seção 4.1.1.

Proposição 4.2.2 *Sejam $M^{a^{[]}_i}$ (para $i = 1...n$) as matrizes intervalares. Para um dado $a_1...a_n$, se:*

1. *$l - m^{a^+_i}_{a_1...a_i...an} \geq l - m^{a^-_i}_{a_1...a_{i+1}...an}$, então existe uma função de aproximação do relevo não crescente entre $a_1...a_i...a_n$ e $a_1...a_{i+1}...a_n$ (direção $a_i - a_{i+1}$);*

2. $l - m^{a_i^-}_{a_1 \ldots a_{i-1} \ldots an} \leq l - m^{a_i^+}_{a_1 \ldots a_i \ldots an}$, *então existe uma função de aproximação do relevo não decrescente entre* $a_1 \ldots a_{i-1} \ldots a_n$ *e* $a_1 \ldots a_i \ldots a_n$ *(direção* $a_{i-1} - a_i$*).*

Os registradores de declividade sofrem uma pequena modificação em sua definição (Definição 4.1.5 - Seção 4.1.1).

Definição 4.2.15 *Um registrador de declividade da célula* $a_1 \ldots a_n$ *é uma tupla*

$$reg = (reg.a_1^-, reg.a_1^+, reg.a_2^-, reg.a_2^+, ..., reg.a_n^-, reg.a_n^+),$$

cujos valores dos registradores direcionados de declividade são dados por:

1. *Para células que não estão na borda da tesselação:*

$$reg.a_i^- = \begin{cases} 0 & \textit{se a condição do item 1 da Proposição 4.2.2 é verdadeira;} \\ 1 & \textit{caso contrário.} \end{cases}$$

$$reg.a_i^+ = \begin{cases} 0 & \textit{se a condição do item 2 da Proposição 4.2.2 é verdadeira;} \\ 1 & \textit{caso contrário.} \end{cases}$$

2. *Para as células cuja(s) face(s)* $a_i^\pm$ *está(ão) na borda, então* $reg.a_i^{+-} = 0^2$*. Os demais registradores direcionados de declividade são determinados usando o item 1 dessa definição.*

Observa-se que os índices $a_i^\pm$ são sensíveis ao contexto onde são empregados, ou seja, quando eles aparecem na forma $reg.a_i^\pm$ (Definição 4.2.15), eles indicam o registrador de declividade no eixo da coordenada a_i na direção que decresce (cresce) o seu valor. A Figura 12 apresenta a distribuição dos a_i para o caso tridimensional.

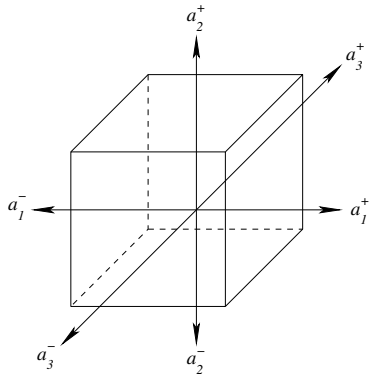

Figura 12 – Notação do sentido dos registradores $a_i^\pm$.

Quando se apresentarem $l - m^{a_i^\pm}_{a_1 \ldots a_n}$ indicam os limites inferior (a_i^-) e superior (a_i^+) do intervalo do erro de aproximação da célula $a_1 \ldots a_n$, na coordenada a_i.

Definição 4.2.16 *A matriz dos registradores de declividade da camada l é definida como uma matriz n-dimensional $a_1...a_n$ denotada por $l - M^{reg} = [l - m^{reg}_{a_1...a_n}]$.*

No caso multi-dimensional, o corolário 4.1.3 da Seção 4.1.1 é re-apresentado como segue:

Corolário 4.2.1 *Considerando a direção $a_i - a_{i+1}$, uma função de aproximação do relevo $l - m_{a_1...a_n}$ é:*

1. *estritamente crescente entre $a_1...a_i...a_n$ e $a_1...a_{i+1}...a_n$ se $l - m^{reg.a_i^+}_{a_1...a_n} = 1$ (neste caso, $l - m^{reg.a_i^-}_{a_1...a_n} = 0$);*

2. *estritamente decrescente entre $a_1...a_i...a_n$ e $a_1...a_{i+1}...a_n$ se $l - m^{reg.a_i^-}_{a_1...a_{i+1}...a_n} = 1$ (neste caso, $l - m^{reg.a_i^+}_{a_1...a_i...a_n} = 0$);*

3. *constante entre $a_1...a_i...a_n$ e $a_1...a_{i+1}...a_n$ se $l - m^{reg.a_i^+}_{a_1...a_i...a_n} = 0$ e $l - m^{reg.a_i^-}_{a_1...a_{i+1}...a_n} = 0$.*

O peso de cada registrador deve ser determinado de modo a não configurarem valores conflitantes para o registrador de estado da célula. Ou seja, cada estado da célula deve ser alcançado somente a partir de uma combinação de pesos dos registradores de declividade. A Definição 4.2.17 caracteriza algoritmicamente a determinação dos pesos dos registradores de declividade.

Definição 4.2.17 *Sejam os pesos $w_{reg.a_i^-} = 2^{(2xi)-2}$ e $w_{reg.a_i^+} = 2^{(2xi)-1}$ (para $i = 1...n$) que serão associados aos registradores de declividade direcionados. A matriz n-dimensional de estados da camada l é definida como uma matriz $a_1...a_n$ dada por $l - M^{state} = [l - m^{state}_{a_1...a_n}]$, cuja entrada na coordenada $a_1...a_n$ da camada l é o valor correspondente ao do estado da célula, calculado com o valor da codificação binária dos registradores de declividade correspondentes, da seguinte forma*

$$l - m^{state}_{a_1...a_n} = \sum_{i=1}^{n} w_{reg.a_i^-} \times l - m^{reg.a_i^-}_{a_1...a_n} + w_{reg.a_i^+} \times l - m^{reg.a_i^+}_{a_1...a_n}.$$

Definição 4.2.18 *A matriz de limites da camada l é definida como a matriz a_1 x...x a_n denotada por $l - m^{limit} = [l - m^{limit}_{a_1...a_n}]$, onde a entrada $a_1...a_n$ da camada l é determinada como $l - m^{limit}_{a_1...a_n} = 0$, se uma das condições listadas na Tabela 9 é verificada; 1, caso contrário.*

Tabela 9 – Condições das células $a_1...a_n$ não limítrofes da camada l, para $i = 1...n$.

Id	Condições
(3 x i) -2	$l - m^{a_i^+}_{a_1...a_{i-1}...a_n} = l - m^{a_i^+}_{a_1...a_i...a_n} = 1$
(3 x i) - 1	$l - m^{a_i^-}_{a_1...a_i...a_n} = l - m^{a_i^-}_{a_1...a_{i+1}...a_n} = 1$
3 x i	$l - m^{a_i^+}_{a_1...a_{i-1}...a_n} = l - m^{a_i^+}_{a_1...a_i...a_n} = l - m^{a_i^-}_{a_1...a_i...a_n} = l - m^{a_i^-}_{a_1...a_{i+1}...a_n} = 0$

A Definição 4.1.9 de sub-região da Seção 4.1.1 deve ser re-apresentada da seguinte maneira:

Definição 4.2.19 *A sub-região de declividade constante associada à célula não limítrofe $a_1...a_n$, denotada por $l - SR_{a_1...a_n}$, é definida indutivamente como segue:*

1. $a_1...a_n \in l - SR_{a_1...a_n}$;
2. *Se $a_1^{'}...a_n^{'} \in l - SR_{a_1...a_n}$, então todas suas células vizinhas não limítrofes também pertencem a $l - SR_{a_1...a_n}$.*

4.2.3 Projeção das Camadas do Modelo

A camada $\pi - M^{limit}_{a_1...a_n}$ ou $0 - M^{limit}_{a_1...a_n}$ foi reservada para ser a projeção das células limites das camadas superiores. Esta projeção tem o intuito de identificar algumas informações interessantes para o analista, como:

- as células que são limites em todas as camadas;
- a projeção de todas as sub-regiões;
- o grau de certeza de uma determinada célula ser limite, etc.

Para este tipo de análise inicial (pois é a mais imediata) tem-se dois tipos de projeções, quando as camadas tem as mesmas ou diferentes participações. Estes dois tipos de projeções são apresentados a seguir.

Projeção tipo 1

Cada camada participa igualmente na determinação da projeção das sub-regiões limites. Assim, o método trivial de projeção pode obter, na camada base, todas as sub-regiões geradas individualmente em cada camada.

Definição 4.2.20 *Seja uma tesselação M com l-camadas e n-dimensional. A projeção $\pi - M^{limit}_{a_1...a_n}$ das l camadas sobre a camada base, indicando todas as células limites, é dada por*

$$\pi - M^{limit}_{a_1...a_n} = 0 - M^{limit}_{a_1...a_n} = \bigvee_{i=1}^{l} i - M^{limit}_{a_1...a_n}, \forall a_1...a_n \in M.$$

Neste caso, basta a célula ser limítrofe em apenas uma das camadas e ela será projetada como limítrofe na camada $\pi - M^{limit}_{a_1...a_n}$.

Projeção tipo 2

Cada camada pode apresentar participações diferentes na determinação da projeção das sub-regiões limites. Para isso, introduz-se a noção da pesificação das camadas. Assim, cada camada tem seu grau de participação na projeção final, normalmente definido pelo especialista.

Ainda, as camadas podem possuir pesos que não são complementares, ou seja, a sua soma pode não ser 1, ou 100%. Portanto, os pesos das camadas devem ser normalizados.

Definição 4.2.21 *Cada camada l de uma tesselação n-dimensional M tem um peso associado, denotado por $w_i = \frac{x_i 1}{x_i 2}$, para $i = 1, ..., l$.*

Definição 4.2.22 *Sejam $w_i = \frac{x_i 1}{x_i 2}$, para $i = 1, ..., l$. os pesos associados às camadas da tesselação M. A normalização destes pesos, denotada por $\overline{w_i}$ é dada por:*

$$\overline{w_i} = \frac{x_{i1} \times \pi_{j \neq i}^{l} x_{j2}}{\sum_{i=1}^{l} (x_{i1} \times \pi_{j \neq i}^{l} x_{j2})}$$

Definição 4.2.23 *Seja uma tesselação M com l-camadas e n-dimensional e os pesos normalizados $\overline{w_i}$ associados a cada camada. A projeção $\pi - M_{a_1 ... a_n}^{limit}$ das l camadas sobre a camada base, indicando todas as células limites, é dada por*

$$\pi - M_{a_1 ... a_n}^{limit} = 0 - M_{a_1 ... a_n}^{limit} = \sum_{i=1}^{l} i - M_{a_1 ... a_n}^{limit} \times \overline{w_i}, \forall a_i ... a_n \in M.$$

Neste segundo tipo de projeção, as células marcadas como limite na camada $\pi - M$ também indicarão o grau de certeza desta marcação. Os valores do registrador de limite $\pi - M_{a_1 ... a_n}^{limit}$ assumirão valores entre 0 e 1. Fazendo uma analogia com a representação gráfica de uma célula limite, se $\pi - M_{a_1 ... a_n}^{limit} = 1$ então ela poderia ser representada pela cor preta. Se $\pi - M_{a_1 ... a_n}^{limit} = 0$, ela poderia ser branca e os demais valores seriam representados como tons de cinza.

A Figura 13 ilustra uma projeção (genérica) das categorizações de três camadas bidimensionais na camada base.

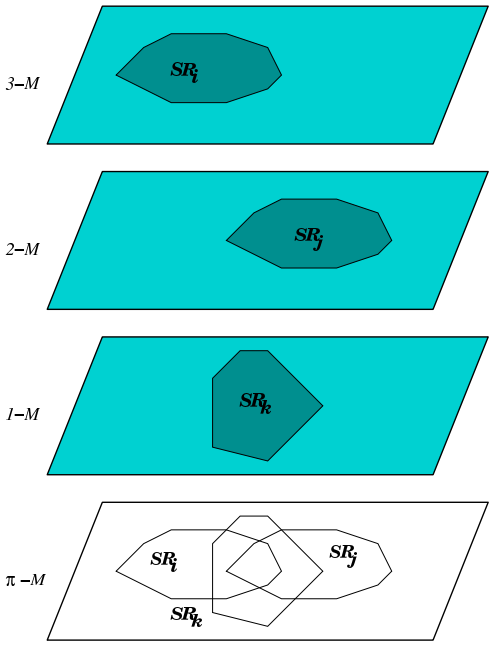

Figura 13 – Exemplo de projeção das categorizações na camada base.

4.3 Implementação Seqüencial e Validação do ICTM

A implementação das versões seqüenciais do ICTM, descritas anteriormente, foi realizada através das linguagens C e C++ para o ambiente *Linux*.

Primeiramente foi implementada a versão Topo-ICTM [7]. Esta primeira implementação contém somente as funções do ICTM que compõem o processo de categorização propriamente dito.

Posteriormente foi implementada a versão correspondente ao modelo geral ICTM [4]. As únicas diferenças existentes nessa nova versão estão relacionadas ao suporte necessário para as novas características e funcionalidades do modelo: (i) *modelo multi-camada;* (ii) *modelo multi-dimensional;* (iii) e *algoritmos de projeção.*

A fim de avaliar o comportamento e validar o modelo ICTM, foram realizados alguns testes. Os resultados destes testes são apresentados a seguir, com base nos modelos de elevação digital (DEM) de resolução 1000m e 500m do quadrante com coordenadas:

- Canto superior Esquerdo: $X = 427559m$, $Y = 6637852m$;
- Canto inferior Direito: $X = 480339m$, $Y = 6614507m$.

Estas são coordenadas UTM (*Universal Transversa de Mercator*), Fuso 22S (Zona UTM 22, hemisfério sul) e Datum SAD69 (Datum da América do Sul).

A Figura 14 representa o DEM de resolução 1000m horizontalmente e 400m verticalmente, compreendendo 24 linhas e 53 colunas, obtido a partir da digitalização de cartas topográficas na escala 1:1.000.000.

A Figura 15 representa o DEM de resolução 500m horizontalmente e 400m verticalmente, compreendendo 49 linhas e 106 colunas, obtido a partir da interpolação de cartas topográficas digitalizadas na escala 1:1.000.000.

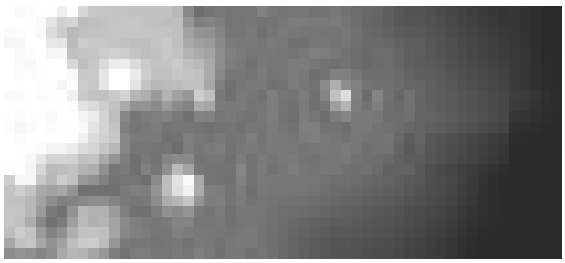

Figura 14 – Imagem do DEM de resolução 1000m.

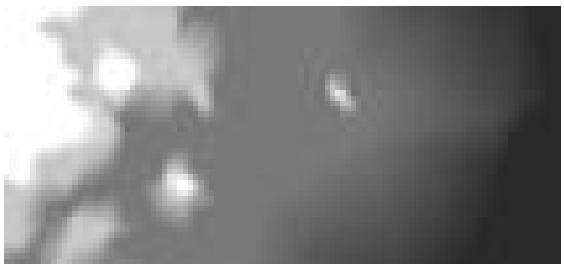

Figura 15 – Imagem do DEM de resolução 500m.

Os resultados são apresentados nas Tabelas 10 (DEM 1000m) e 11 (DEM 500m) e nas figuras do Apêndice A que apresentam uma outra forma de representação de resultados (mapa com marcações dos limites de células) do ICTM. Pode-se observar claramente que no ICTM o número de categorias é inversamente proporcional ao raio da vizinhança.

Além disso, pode-se perceber que para o DEM 500m, cujo ponto congrega uma área 4 vezes menor que o ponto correspondente no DEM 1000m, o número de categorias também seguiu

Tabela 10 – Resultados para o quadrante no DEM de 1000m.

Raio de Vizinhança	Número de Categorias
1	76 (veja Figura 40)
5	36 (veja Figura 41)
10	22 (veja Figura 42)
20	18 (veja Figura 43)

Tabela 11 – Resultados para o quadrante no DEM de 500m.

Raio de Vizinhança	Número de Categorias
1	230 (veja Figura 44)
5	143 (veja Figura 45)
10	125 (veja Figura 46)
20	108 (veja Figura 47)

aproximadamente este fator. Neste caso, o DEM com resolução de 500m, especificamente desta região, não apresentou novas regiões a serem categorizadas, apenas detalhou as regiões encontradas. No entanto, em regiões com maiores variações de declividade possivelmente este fator não seja mais verificado.

Ainda, em áreas mais planas os raios de vizinhança maiores foram boas aproximações para as categorias. Por exemplo, observe a região A na Figura 40, suas representações com raios maiores (Figuras 41, 42 e 43) indicam aproximações razoáveis para este tipo de declividade. Entretanto, para regiões com maiores variações de declividades, os melhores resultados (mais detalhados) foram alcançados com raios menores.

Regiões com maiores concentrações de categorias (por exemplo, a região B na Figura 40) são indicadoras que uma análise mais refinada deve ser feita (veja a representação no DEM de 500m de resolução correspondente na Figura 44). O refinamento do modelo é dado por dois aspectos:

- pela resolução espacial do modelo digital de elevação;
- pelo raio da vizinhança da célula.

Assim, regiões com várias células limites podem ser melhores estudadas com o aumento da resolução dos dados de altimetria, ou com a redução do raio de vizinhança.

No ICTM, o estado de uma célula em relação aos seus vizinhos, em termos de declividade, pode ser verificado instantaneamente, contrastando com as análises usuais presentes nos SIG's (conforme as Figuras 16 e 17).

Na informação gerada por contornos (curvas de nível) a informação é global, permitindo a visualização do estado da célula em função de células distantes. No ICTM, a informação é pontual, associando várias propriedades no mesmo ponto.

Também percebe-se que o tamanho da área, a resolução do DEM referente à esta área e o grau de variação da declividade são as propriedades mais importantes na determinação de um raio de vizinhança apropriado para uma categorização significativa.

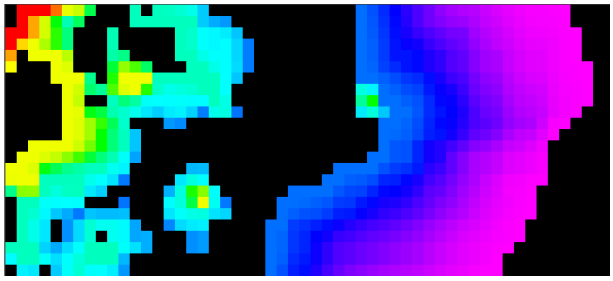

Figura 16 – Linhas de contorno no DEM de resolução 1000m.

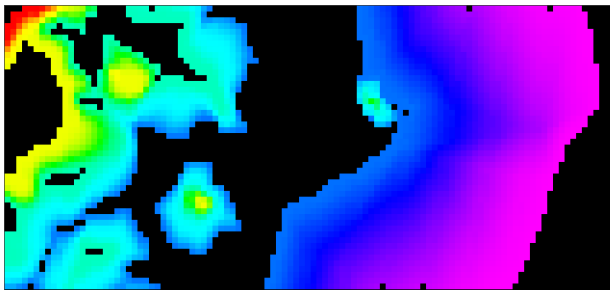

Figura 17 – Linhas de contorno no DEM de resolução 500m.

Estes resultados indicam que regiões com menor variação de declividade são receptivas ao ICTM com raios maiores. Ao contrário, regiões com grandes variações de declividade sugerem raios menores. A Figura 18 indica as classes conforme o grau de declividade do modelo de elevação digital com resolução de 1000m.

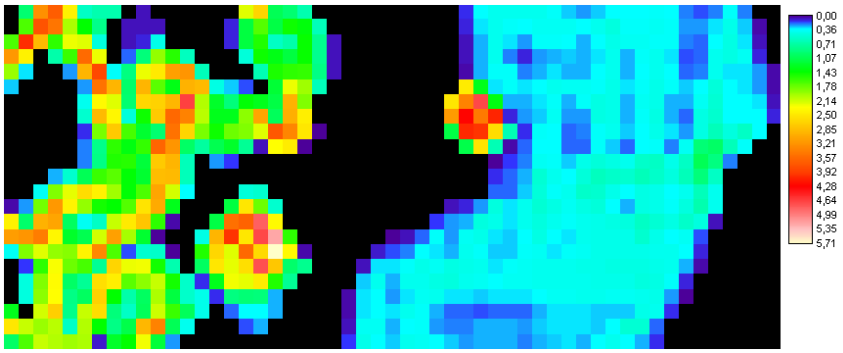

Figura 18 – Graus de declividade do DEM de resolução 1000m.

Evidentemente, regiões pequenas comportam apenas raios menores, pois raios de vizinhança maiores tendem a suavizar uma área muito grande ao redor da célula em questão.

4.3.1 Análise de Desempenho

Para demonstrar o comportamento do modelo ICTM e avaliar o desempenho da versão seqüencial implementada, realizou-se alguns testes com base em dois quadrantes da imagem obtida com o satélite LANDSAT da região em torno da Lagoa Pequena, Pelotas, Rio Grande do Sul, considerando a análise de propriedades da região contidas nos quadrantes. A Figura 19 apresenta a área que compreende os quadrantes em questão, com coordenadas UTM, Fuso 22S (Zona UTM 22, hemisfério sul) e Datum SAD69 (Datum da América do Sul). Os quadrantes são representados em malhas de células, conforme o exposto a seguir:

- Quadrante A : Matriz com 577 linhas e 817 colunas (471409 células);
- Quadrante B: Matriz com 1309 linhas e 1765 colunas (2310385 células).

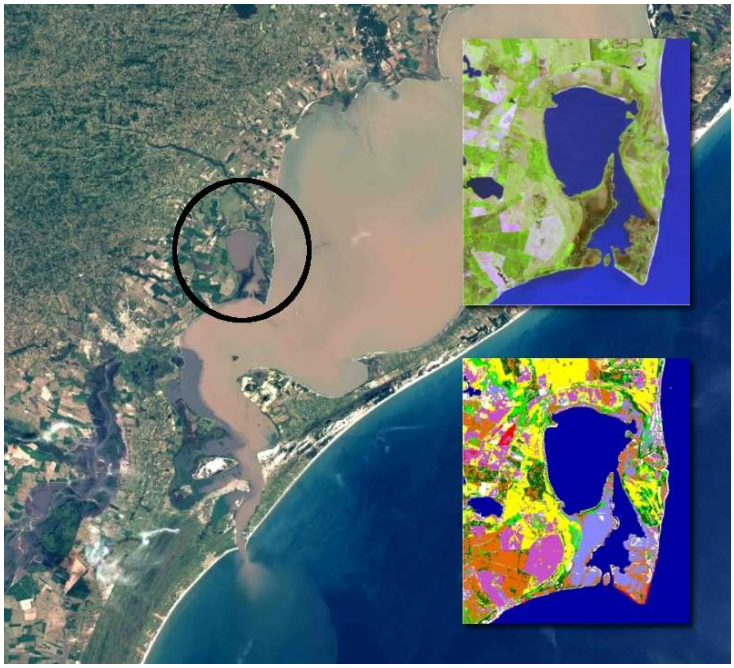

Figura 19 – Mapa de uso do solo da região em torno da Lagoa Pequena (azul claro: ilhas, azul escuro: água, amarelo: futuras colheitas (safras), violeta: transição, verde claro: floresta de riparia, verde escuro: floresta de restinga, vermelho: praia de lagoa, branco: sem classificação).

Os testes foram realizados em três máquinas rodando o sistema operacional Linux Debian, mas com configurações diferentes. São elas:

- Máquina 1: Dois processadores Intel PIII 1GHz, com 512MB de memória principal e 256KB de *cache*;

- Máquina 2: Um processador Intel P4 1.6GHz, com 256MB de memória principal e 256KB de *cache*;
- Máquina 3: Um processador Intel P4 2.8GHz, com 1GB de memória principal e 1024KB de *cache*.

A Tabela 12 apresenta os resultados obtidos. Observe que quanto maior o número de células a serem processadas, maior é a quantidade necessária de memória e capacidade de processamento para que o modelo execute naturalmente. Para o quadrante A, com nc variando de 1 a 5, a máquina 3 sempre foi mais rápida. Isso porque ela apresenta a melhor configuração (mais memória, processador mais rápido e mais *cache*) disponível, da mesma forma que a máquina 2 (segunda melhor configuração) levou vantagem sobre a máquina 1. No entanto, com nc 10 e 15 no mesmo quadrante, a máquina 2 teve que fazer acessos aos disco (*swap*) para conseguir executar o modelo, ou seja, a quantidade de memória livre disponível não foi suficiente. No quadrante B este fenômeno se repete com nc variando principalmente de 3 a 5. Já com nc 10 e 15, nenhuma máquina foi capaz de executar o modelo. A quantidade de memória necessária para alocar a estrutura de dados que suportasse o conjunto de dados de entrada foi superior ao limite máximo de memória que pode ser alocada nas máquinas. Sendo assim, o sistema operacional abortou a execução do modelo.

Tabela 12 – Comportamento do ICTM seqüencial (nc é o número de propriedades/camadas analisadas e $tcel$ é o número total de células que são analisadas durante o processamento).

Quadrante	**nc**	**tcel**	**Máquina 1**	**Máquina 2**	**Máquina 3**
$A_{(577\times817)}$	1	471409	5.483s	4.362s	1.884s
	2	942818	10.644s	8.690s	3.637s
	3	1414227	15.282s	13.040s	5.681s
	4	1885636	20.534s	18.109s	7.601s
	5	2357045	25.606s	20.994s	9.464s
	10	4714090	52.943s	*Swap*	19.494s
	15	7071135	*Swap*	*Swap*	27.709s
$B_{(1309\times1765)}$	1	2310385	25.276s	21.289s	8.450s
	2	4620770	53.612s	*Swap*	16.857s
	3	6931155	*Swap*	*Swap*	26.711s
	4	9241540	*Swap*	*Swap*	34.877s
	5	11551925	*Swap*	*Swap*	45.693s
	10	23103850	*Abortado*	*Abortado*	*Abortado*
	15	34655775	*Abortado*	*Abortado*	*Abortado*

Portanto, no que diz respeito ao desempenho do modelo, pode-se dizer que os resultados obtidos foram insatisfatórios para análises de grandes regiões. Mais precisamente, a medida que se aumenta o volume de dados de entrada, a quantidade de cálculo realizado pelo modelo e a quantidade necessária de memória livre na máquina que suporte este volume de dados também aumenta. Sendo assim, a execução do ICTM em máquinas convencionais, no caso de análise de grandes regiões, pode se tornar inviável.

4.4 Aplicabilidade do ICTM

Dentre as aplicações do modelo ICTM, na análise de superfícies topográficas e a batimetria destas superfícies, pode-se destacar os seguintes trabalhos futuros: (i) cálculos de volume e caracterização do perfil para dragagem; (ii) caracterização de unidades geológicas e geofísicas; (iii) utilização das propriedades de declividade e orientação em modelos hidrológicos, análise de transporte de sedimentos, balanços de energia, identificação de padrões de vegetação; (iv) diferenças micro-climáticas; (v) caracterização da profundidade efetiva do solo; (vi) caracterização do regime de nutrientes do solo; (vii) análise do risco de erosão (análise de transporte de sedimentos); (viii) necessidade de práticas especiais de conservação de solo e etc.

Outra aplicação do ICTM poderia ser a análise das variações de crescimento das árvores dentro de uma classe topográfica homogênea. Este tipo de análise é freqüentemente utilizada para identificar as variações na disponibilidade de suprimento de água, ar e nutriente do solo, as quais estão intimamente relacionadas com as propriedades do solo, com especial destaque para a textura do solo.

Além disso, pode-se utilizar futuramente o conceito de autômatos celulares para estender o tipo de análise categorizadora do modelo. Isso permitiria não apenas uma análise estática das características do espaço em um momento dado, mas também possibilitaria o avanço na direção de uma simulação dos aspectos dinâmicos desse espaço, em um intervalo de tempo. Assim, permitiria-se a modelagem de processos dinâmicos mais complexos.

Uma vez adicionada esta característica dinâmica no modelo, as seguintes análises poderiam ser efetuadas: (i) simulação da dispersão de óleo no oceano; (ii) simulação da dispersão de sedimentos no estuário; (iii) simulação da dispersão de contaminantes químicos em águas subterrâneas; (iv) simulação do comportamento espacial sócio-econômico, relacionados com as características físicas do meio urbano; (v) determinação do zoneamento agroclimático, atualmente fornecido em intervalos de 10 dias, sendo que os dados meteorológicos são diários; (vi) levantamentos geoquímicos, complementando a análise geoestatística e etc.

A aplicabilidade do ICTM foi definida em conjunto com uma profissional especialista na área de sensoriamento remoto do GMFC.

5 O modelo HPC-ICTM

Este capítulo aborda a proposta para a dissertação de mestrado, ou seja, a definição do modelo HPC-ICTM. O objetivo principal é permitir que o modelo ICTM possa ser aplicado para análises de grandes regiões geográficas e que informações geográficas distribuídas possam ser utilizadas, reduzindo o problema de baixo desempenho devido à necessidade de muita memória e de grande capacidade de processamento na análise de grandes regiões e a dificuldade no acesso aos dados geográficos.

Para atingir este objetivo utiliza-se as plataformas de execução de aplicações paralelas apresentadas no Capítulo 3. O uso de máquinas agregadas reduzirá a necessidade por memória e capacidade de processamento em uma única máquina e aumentará o desempenho do modelo para análise de grandes regiões geográficas. Com o uso das grades computacionais será possível executar o modelo para análises de grandes regiões (independente da obtenção de desempenho) e utilizar informações geograficamente distribuídas, o que é uma forte demanda das áreas relacionadas à análise e processamento de dados geográficos devido à dificuldade existente atualmente para obtenção de tais informações.

A seguir apresenta-se a modelagem paralela do ICTM para as duas plataformas em questão.

5.1 Modelagem Paralela do ICTM

O primeiro passo para definição dos modelos paralelos do ICTM para as plataformas de máquinas agregadas e grades computacionais foi verificar as maneiras possíveis de se paralelizar o modelo, ou seja, as formas de particionar o ICTM.

A seguir, de acordo com o estudo das plataformas paralelas adotadas e com a identificação das formas de particionamento do modelo ICTM, partiu-se para o relacionamento das maneiras de particionar o modelo com as características das máquinas agregadas e das grades computacionais. A idéia principal era verificar quais das formas de particionamento poderiam ser implementadas em cada uma das plataformas utilizadas.

O próximo e último passo, antes da definição dos modelos paralelos do ICTM para as duas plataformas paralelas (máquinas agregadas e grades), foi definir as ferramentas/tecnologias que seriam empregadas nas futuras implementações paralelas do modelo.

As seções abaixo apresentam estas três etapas essenciais para a modelagem paralela do modelo, bem como a definição dos modelos paralelos propriamente ditos.

5.1.1 Particionamento do ICTM

A partir da análise detalhada do modelo ICTM foi possível identificar quatro maneiras de paralelizá-lo. Cada uma dessas formas de particionamento são descritas nas seções seguintes, começando pela mais intuitiva e simples até a mais sofisticada e complicada.

Paralelização de camadas

Esta é a forma mais simples e direta de paralelizar o ICTM. Cada processo paralelo calcula uma determinada camada do modelo, ou seja, cada propriedade da região geográfica é analisada individualmente.

Dessa forma, o processo de categorização do modelo é aplicado para cada camada (propriedade geográfica) em questão. A Figura 20 ilustra essa forma de particionamento do modelo ICTM. Observe que a quantidade de trabalho enviada para cada processo paralelo é grande (granulosidade grossa), pois cada um deve aplicar todas as funções do modelo sobre a propriedade recebida.

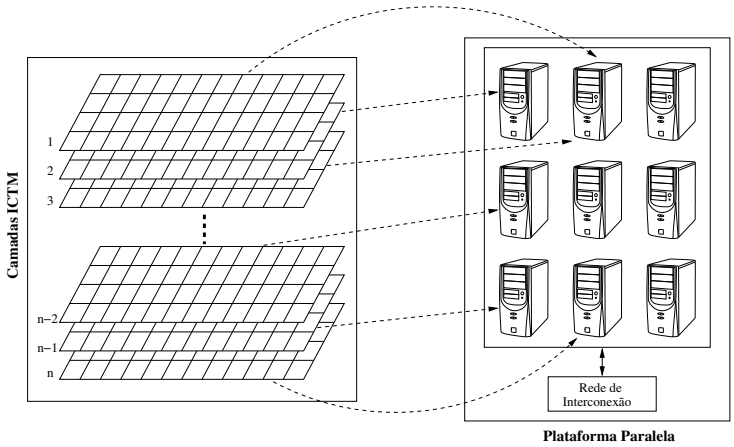

Figura 20 – Particionamento do ICTM em camadas.

Quanto a implementação desta forma de particionamento, pode-se dizer que as tarefas são independentes uma das outras. Ou seja, não é necessário que exista comunicação entre os processos responsáveis pelas categorizações de cada uma das camadas do modelo. Cada processo precisa receber apenas os dados de entrada referentes a propriedade geográfica a ser analisada e os parâmetros de entrada para o modelo.

No que se refere a aplicabilidade, esta forma de particionamento deve ser utilizada para análises de grandes regiões geográficas com mais de uma propriedade (camada).

Paralelização de funções

Esta forma de paralelizar o ICTM também é bastante intuitiva. Cada processo paralelo calcula uma função independente do modelo. Apesar do ICTM apresentar um processo de

categorização que segue etapas seqüenciais, algumas funções do modelo podem ser processadas em paralelo. São elas:

- Funções referentes a criação e ao cálculo das matrizes intervalares (veja Seção 4.1.1, Definição 4.1.4);
- Funções referentes a criação e ao cálculo das matrizes de monotonicidade (veja Seção 4.1.1, Definição 4.1.6).

A Figura 21 ilustra esta forma de particionamento do modelo ICTM, considerando a paralelização das matrizes intervalares do modelo. Observe a necessidade de ser ter um "controle" seqüencial, mesmo se tratando de uma modelagem paralela. Isso deve acontecer devido à característica do próprio processo de categorização do ICTM, ou seja, existe uma seqüência de passos que precisa ser respeitada.

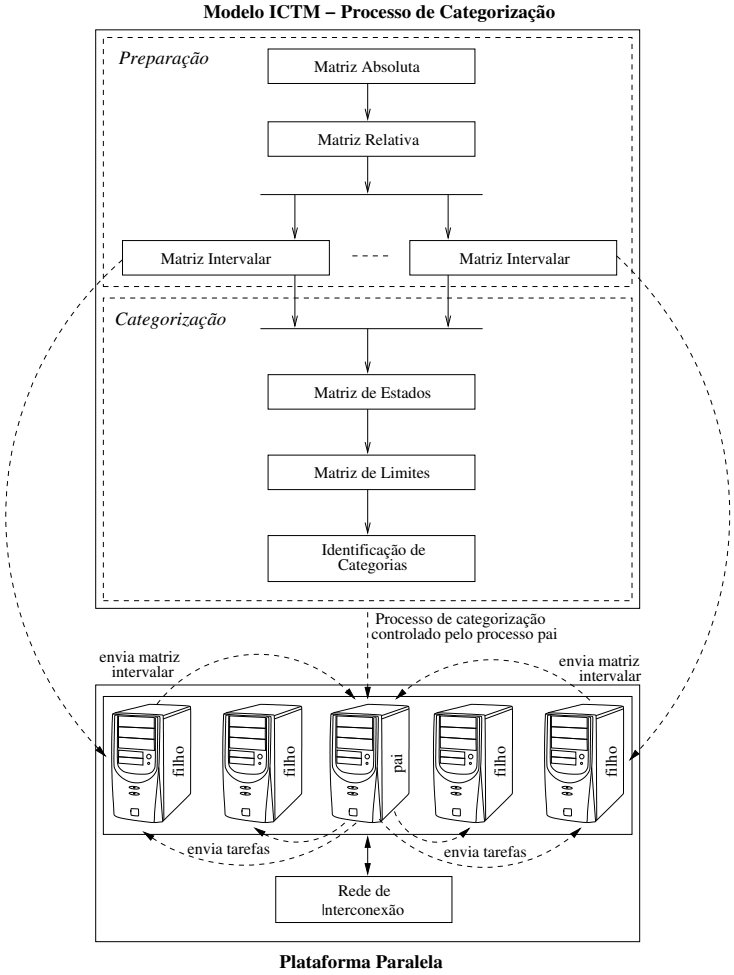

Figura 21 – Particionamento do ICTM em funções.

Sendo assim, para implementar uma versão paralela do modelo baseada na paralelização de funções, é necessário utilizar um modelo de programação que permita a comunicação en-

tre o processo coordenador (responsável pela divisão das tarefas) e os demais processos que efetivamente farão os cálculos.

Note também, que a quantidade de trabalho enviado para cada processo paralelo não é tão grande, ao contrário da forma de particionamento anterior onde cada processo calculava todas as funções do modelo ICTM. Neste caso, o trabalho realizado pelos processos paralelos é a nível de funções do modelo, exigindo uma maior comunicação entre os processos. Mais precisamente, cada processo calcula determinadas funções paralelas do modelo para todas as células da malha que representa a região analisada. Pode-se dizer então, que a quantidade de trabalho enviado para cada processo paralelo é média (granulosidade média).

No que diz respeito a aplicabilidade, esta forma de particionamento deve ser utilizada para análises de regiões geográficas muito grandes e complexas, onde a quebra do processo de categorização do modelo possa acelerar a análise.

Paralelização de domínios

Esta forma de paralelização do ICTM já é um pouco mais sofisticada. Cada processo paralelo calcula parte da região que será analisada. Mais precisamente, a malha (tesselação) que compreende a região geográfica é particionada em blocos de células que são calculados em paralelo.

Esta decomposição do trabalho total (malha da região) só é possível devido ao formalismo utilizado ser inerentemente paralelo. Ou seja, cada célula da tesselação só utiliza informações de seus vizinhos (à direita, esquerda, acima e abaixo) para determinar seu estado.

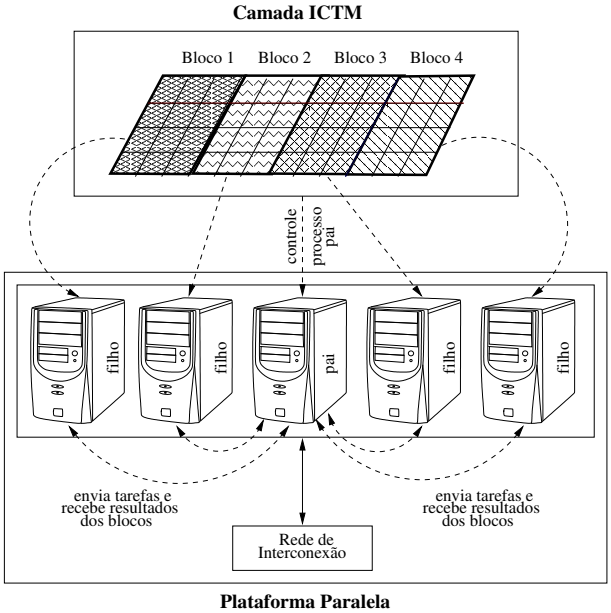

Figura 22 – Particionamento do ICTM em domínios.

Esta abordagem exige muito cuidado do programador e apresenta, em sua definição, problemas claros de desempenho devido à quantidade de comunicação necessária entre os processos.

Conforme foi comentado anteriormente, cada célula precisa de informações de suas células vizinhas. Sendo assim, para o cálculo das células das bordas dos blocos, criados pela decomposição da malha total, é necessário que informações de células contidas em outros blocos rodando em processadores diferentes sejam obtidas. A Figura 22 exibe esta forma de particionamento do modelo ICTM, considerando a existência de uma única malha (camada única) dividida em quatro blocos de células.

Embora a quantidade de células que cada processo paralelo recebe para o processamento dos cálculos seja pequena, a quantidade de trabalho não é tão pequena assim. Isso porque para cada bloco de células o processo paralelo deve aplicar todas as funções do modelo, sendo necessário comunicar com os outros processos paralelos, conforme mencionado anteriormente. Sendo assim, pode-se dizer que a quantidade de trabalho de cada processo paralelo também é média neste caso (granulosidade média).

Quanto a aplicabilidade, esta forma de particionamento deve ser utilizada em análises de regiões geográficas representadas em malhas com muitos pontos. É preciso que a quantidade de trabalho seja muito grande para compensar o volume de comunicação inerente à abordagem.

Paralelização de células

Esta forma de paralelizar o ICTM é a menos intuitiva e mais sofisticada de todas. Cada processo paralelo calcula células individuais do modelo. Assim, cada um dos estados das células da malha da região geográfica analisada é obtido individualmente. A Figura 23 ilustra esta forma de particionar o modelo ICTM.

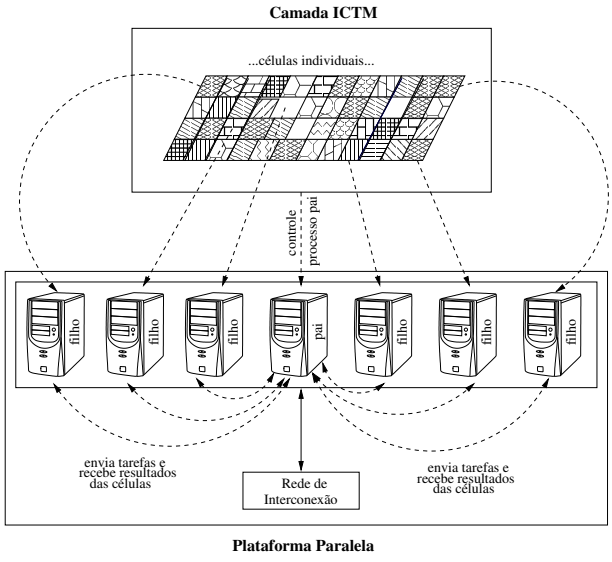

Figura 23 – Particionamento do ICTM em células.

Este tipo de paralelização só é possível pelo mesmo motivo citado na seção anterior, ou seja, o formalismo é inerentemente paralelo e cada célula utiliza somente informações de suas células vizinhas. No entanto, o mesmo problema comentado na paralelização de domínios

ocorre, e com muito mais freqüência. Os valores das células distribuídas entre os processos podem ser necessários para outras células de outros processos distintos.

Nesta abordagem, a quantidade de trabalho enviado para cada processo paralelo é bastante pequena (granulosidade fina), embora cada um execute todas as funções do modelo sobre uma determinada célula. Assim, a comunicação entre os processos torna-se muito mais freqüente.

Portanto, é preciso avaliar se o custo de comunicação inerente a esta forma de paralelização não a inviabilizará. Ou seja, a aplicabilidade desta forma de particionamento é um pouco restrita, pois é necessário que a malha seja suficientemente extensa e complicada para que os resultados sejam satisfatórios.

5.1.2 Relação: Particionamento $\times$ Plataformas Paralelas

De acordo com o estudo dos conceitos e características das plataformas paralelas utilizadas no trabalho e com a definição das formas de particionamento do modelo ICTM, foi possível identificar quais as formas de paralelização do modelo podem ser implementadas nas máquinas agregadas e nas grades computacionais.

Nas máquinas agregadas, todas as maneiras de paralelizar o modelo podem ser implementadas. Mais precisamente, não existe nenhum problema ou limitação, tanto nas formas de particionamento como na própria plataforma, que impeça este procedimento.

Para o caso da forma de particionamento em camadas, as seguintes características são evidentes:

- Grande quantidade de trabalho enviado aos processos paralelos, ou seja, granulosidade grossa;
- Necessidade de um processo responsável pelo envio dos parâmetros do modelo e das tarefas (camadas) com seus conjuntos de dados de entrada para os processos que efetivamente farão os cálculos para cada camada;
- Nenhuma necessidade de comunicação entre os processos que executam as tarefas, pois eles precisam apenas receber os dados de entrada referentes a cada camada e os parâmetros do modelo do processo responsável pelo envio dessas informações;

Baseado nestas características, pode-se fazer uma análise das possíveis forma de se implementar esta forma de particionamento, de acordo com os modelos de programação paralela apresentados na Seção 3.1.3.

Um possível modelo de programação paralela para esta forma de particionamento é o modelo “mestre-escravo” exibido na Figura 3. Dessa forma, o processo mestre ficaria responsável pelo envio dos parâmetros do modelo, das tarefas e dos dados de entrada aos processos escravos que, por sua vez, ficariam responsáveis pelas categorizações das propriedades analisadas. Após finalizar suas tarefas, os processos escravos comunicariam com o processo mestre para solicitar mais trabalho.

O modelo de programação “divisão e conquista” exibido na Figura 4 poderia também ser empregado para esta forma de particionamento. Assim, o processo topo da estrutura (árvore)

ficaria responsável pela carga de todos os conjuntos de dados de entrada, pela leitura dos parâmetros do modelo e pelo processamento de uma das camadas. Além disso, antes de iniciar seu processamento, o processo topo enviaria os parâmetros, os dados e as demais tarefas (camadas a serem processadas) para seus processos filhos. Estes, por sua vez, repetiriam o mesmo procedimento do processo topo, ou seja, receberiam os dados da camada e os parâmetros do modelo e processariam mais camadas, enviando a seus filhos o restante das tarefas com seus respectivos dados e os parâmetros do modelo. Observe que o uso deste modelo de programação introduz um custo de comunicação bastante alto se comparado com o modelo "mestre-escravo", pois cada processo precisa receber dados que não são necessários para execução de sua tarefa. Ainda, a integração de resultados inerente ao modelo não seria necessária, pois cada processo geraria sua própria saída. O número de processos paralelos teria que ser igual ao número de camadas analisadas, pois só assim cada processo executaria uma única tarefa. Caso contrário, um coordenador divisor de tarefas (certamente o processo topo) teria que existir para que os processos pudessem solicitar mais trabalho.

Da mesma forma que o modelo anterior, o modelo de programação *"pipeline"* exibido na Figura 5 poderia ser utilizado. O primeiro processo do *pipeline* executaria a primeira camada e enviaria as próximas camadas com seus respectivos dados e os parâmetros do modelo para o segundo processo do *pipeline*. Este, executaria a segunda camada e repetiria o procedimento de envio das informações realizado pelo primeiro processo. No entanto, percebe-se também que o custo de comunicação é grande neste modelo, pois cada processo recebe informações desnecessárias para o processamento de sua tarefa, apenas para poder repassá-las aos demais processos.

Quanto ao modelo de programação "fases paralelas" exibido na Figura 6, esta forma de particionamento não apresenta o comportamento típico deste modelo. Mais precisamente, não existe a divisão clara das etapas de computação e comunicação e, por isso, este modelo não se adapta a esta maneira de paralelizar o modelo ICTM.

Para o caso da forma de particionamento em funções, tem-se as seguintes características:

- Quantidade média de trabalho enviado aos processos paralelos, ou seja, granulosidade média;
- Necessidade de um processo coordenador do método de categorização do ICTM;
- Necessidade de um processo responsável pelo envio dos parâmetros do modelo e dos conjuntos de dados de entrada para os processos que efetivamente farão os cálculos de cada função paralela;
- Maior necessidade de comunicação entre os processos que realizam os cálculos das funções e o processo coordenador.

Relacionando esta forma de particionamento com os modelos de programação paralela apresentados neste documento, de acordo com as características citadas, pode-se dizer que o único modelo que se adapta a esta maneira de paralelizar o ICTM é o modelo "mestre-escravo". Isso porque o mestre pode assumir o papel do coordenador e o papel do processo responsável pelo

envio dos parâmetros e dados geográficos aos demais processos escravos ao mesmo tempo. Assim, os processos escravos calculam as funções paralelas e retornam os resultados ao processo mestre que se encarrega de manter a seqüência de passos do método de categorização do ICTM e de enviar novas tarefas aos escravos.

Para o caso da forma de particionamento em domínios, as seguintes características estão presentes:

- Quantidade média de trabalho enviado aos processos paralelos, ou seja, granulosidade média;
- Necessidade de um processo responsável pelo envio dos parâmetros do modelo e dos conjuntos de dados de entrada para os processos que efetivamente farão os cálculos de cada bloco de células;
- Grande necessidade de comunicação entre os processos que realizam os cálculos dos blocos de células criados devido à necessidade de um determinado processo obter os valores das células da borda do bloco de um outro processo.

Analisando os modelos de programação paralela, percebe-se que todos eles não se adaptam perfeitamente com a idéia presente nesta forma de particionamento. Existem duas etapas bem distintas nesta maneira de paralelizar o ICTM, quais sejam, uma etapa de processamento e a outra de comunicação. Na etapa de processamento, as funções do modelo são executadas coluna a coluna, célula por célula até atingir a coluna de células da borda do bloco. Neste momento, é necessário comunicar com um outro processo para obter os valores das células (vizinhas) que fazem parte da borda de outro bloco que está sendo processado pelo mesmo. Ou seja, inicia-se a fase de comunicação entre os processos que só finaliza após todos os processos terminarem de receber as informações desejadas. Após o término da fase de comunicação, entra-se novamente na fase de processamento ou computação e esta alternância entre as fases prossegue até o final do processamento das tarefas. Além disso, é necessário que exista um processo que faça a distribuição dos blocos e o envio de dados e parâmetros para os processos que realmente vão calcular os blocos.

Sendo assim, o modelo “fases paralelas” surge como ideal, pois ele oferece justamente a idéia de alternância entre duas fases distintas: computação e comunicação. No entanto, a questão referente a distribuição de tarefas e envio dos dados geográficos e parâmetros do ICTM fica prejudicada, devido à inexistência da figura de um processo coordenador neste modelo.

No modelo “mestre-escravo”, não é possível fazer com que os processos escravos se comuniquem. Da mesma forma, no modelo “divisão e conquista” e *“pipeline”* os processos não se comunicam simultaneamente, ou seja, não é possível fazer com que todos os processos se comuniquem ao mesmo tempo.

Sendo assim, é necessário mesclar características dos modelos (modelo híbrido) para se chegar a uma solução. Uma alternativa seria a utilização de características do modelo “mestre-escravo” com características do modelo “fases paralelas”. Neste modelo híbrido, o processo mestre seria o responsável pela divisão das tarefas (blocos de células) e envio dos dados de entrada e parâmetros do modelo aos processos escravos. Estes, executariam as funções do modelo, durante uma fase de computação, para todas as células do bloco até atingirem as células

da borda do bloco, momento em que entram na fase de comunicação. Após finalizar o cálculo dos blocos, os processos escravos enviam os resultados para o processo mestre e solicitam mais trabalho.

Para o caso da forma de particionamento em células, as seguintes características aparecem:

- Quantidade pequena de trabalho enviado aos processos paralelos, ou seja, granulosidade fina;
- Necessidade de um processo responsável pelo envio dos parâmetros do modelo e dos conjuntos de dados de entrada para os processos que efetivamente farão os cálculos de cada célula da malha;
- Necessidade de comunicação intensa entre os processos que realizam os cálculos das células devido à necessidade de um determinado processo obter os valores das células vizinhas que estão sendo analisadas por outros processos.

Quanto a relação com os modelos de programação paralela, esta forma de particionamento apresenta o mesmo comportamento da paralelização em domínios. Em outras palavras, um modelo híbrido que combine características entre o modelo "mestre-escravo" e o modelo "fases paralelas" é uma alternativa possível.

Entretanto, a paralelização de células pode gerar um custo de comunicação muito alto devido à necessidade constante de comunicação entre os processos, conforme comentado na seção anterior. Por isso, esta forma de particionamento não foi modelada e considerada a partir deste ponto.

Já no caso das grades computacionais, devido às suas características (veja Seção 3.2.1), a forma de particionamento que melhor se adapta é a de paralelização em camadas, pois neste caso, não existe comunicação entre os processos que executam as tarefas.

Qualquer aplicação paralela que necessite de muita comunicação entre as tarefas, não se enquadra no contexto de grades devido, principalmente, à característica de alta dispersão geográfica da plataforma.

No entanto, é possível utilizar as outras formas de particionamento do modelo em um ambiente de grade. Para isso, é preciso que o ambiente possua mecanismos que suportem a comunicação entre as tarefas. Neste trabalho, somente a forma de particionamento em camadas será considerada para a plataforma de grades computacionais.

5.1.3 Definição de Tecnologias

As máquinas agregadas já estão bastante difundidas e seus conceitos e características são bem conhecidos, o que facilita o processo de desenvolvimento de aplicações paralelas para esta plataforma. Sendo assim, optou-se pela biblioteca padrão de comunicação paralela MPI [35, 36].

Já as grades computacionais possuem características importantes que precisam ser levadas em conta na hora de se criar uma aplicação paralela. Dentre as características que foram citadas (Capítulo 3, Seção 3.2.1), uma das mais importantes é a alta dispersão geográfica que as grades

podem ter. Ou seja, conforme mencionado anteriormente, as aplicações que possuem tarefas que produzem muita comunicação, não são adequadas para esta plataforma.

Uma classe de aplicações que se adapta facilmente ao ambiente de grades são as chamadas aplicações "saco de trabalho" ou Bot (*Bag-of-Tasks*), pois elas caracterizam-se por possuir tarefas independentes que não se comunicam entre si [63].

De forma análoga, o ICTM multi-camada pode ser executado como uma aplicação Bot. Para cada camada do modelo, o processo de categorização é aplicado separadamente, sem comunicação entre as camadas (forma de particionamento em camadas).

Assim, devido ao fato da forma de particionamento em camadas do ICTM apresentar o comportamento típico de uma aplicação Bot, adotou-se o ambiente de grade OurGrid [72].

As seções seguintes apresentam um detalhamento das duas tecnologias adotadas.

MPI - *Message Passing Interface*

O padrão MPI foi proposto por um Fórum aberto (MPI-Fórum) constituído de pesquisadores, empresas, usuários e vendedores que definiram a sintaxe, a semântica e o conjunto de rotinas padronizadas para ambientes de troca de mensagens, tendo como objetivos principais aspectos como eficiência, funcionalidade e portabilidade.

A primeira versão do padrão MPI (MPI-1) foi introduzida pelo MPI-Fórum em maio de 1994 e atualizada em junho de 1995. A segunda versão do padrão MPI (MPI-2) começou a ser discutida logo a seguir e foi finalizada em julho de 1997, oferecendo algumas extensões para o padrão original. Entre as novidades mais interessantes pode-se citar o gerenciamento dinâmico de processos, contando com a possibilidade de criação de novos processos em tempo de execução [73].

O padrão MPI define uma biblioteca de troca de mensagens que deixa a cargo dos usuários a definição da forma pela qual os processos são criados e controlados. Atualmente, o padrão MPI é amplamente aceito e existem implementações da biblioteca MPI para os mais variados tipos de sistemas paralelos. Além disso, a biblioteca MPI está disponível para as principais linguagens usadas para programação paralela: Fortran 77, Fortran 90, ANSI C e C++.

Como exemplo de implementação pode-se citar a versão MPICH [74]. Esta implementação foi desenvolvida paralelamente à especificação do padrão MPI (MPI-1), numa ação conjunta do *Argonne National Laboratory* com a *Mississippi State University* e a IBM. Este desenvolvimento paralelo da MPICH com o MPI, possibilitou que os membros do MPI-Fórum pudessem avaliar a viabilidade da implementação, o que acabou acontecendo e fortalecendo o projeto. Como principal característica da MPICH, destaca-se a alta portabilidade.

OurGrid

O OurGrid é um sistema de grades computacionais desenvolvido pela Universidade de Campina Grande (UFCG) do estado da Paraíba, Brasil, em parceria com a HP Brasil (*Hewlett Packard* Brasil). A primeira versão do ambiente OurGrid foi lançada em dezembro de 2004 e, desde então, outras versões foram produzidas buscando o aperfeiçoamento do *software*. Atualmente, a versão disponível no site do projeto[1], lançada em outubro deste ano, é a 3.2.1.

[1] Projeto OurGrid: http://www.ourgrid.org.

Mais precisamente, o OurGrid é um ambiente de execução global baseado no conceito de *sites* para aplicações paralelas modeladas principalmente como Bot. Além disso, a última versão do ambiente permite a utilização de aplicações paralelas MPI. No entanto, o ambiente ainda apresenta algumas falhas no que diz respeito ao uso deste recurso.

Os principais componentes do ambiente OurGrid são:

- *Site:* Instituições, centros de pesquisa ou laboratórios que possuem os recursos que fazem parte da grade;
- *MyGrid:* Escalonador dos usuários da grade (vinculados a um determinado *site*). A partir deste escalonador é que um usuário obtém os recursos desejados [75];
- *Peer:* Gerente de recursos dos *sites* responsável pela obtenção dos mesmos, ou seja, todas as requisições feitas pelos usuários, através do MyGrid, são encaminhas para este componente que se encarrega de encontrar recursos disponíveis, seja no próprio domínio administrativo ou em outros domínios;
- *User Agent: Software* que roda dentro das máquinas ociosas de cada *site* e que fornece uma interface uniforme para execução de aplicações e submissão de arquivos;
- *SWAN:* Mecanismo que permite que os *User Agents* executem em uma *sandbox*, evitando o acesso à áreas não permitidas do sistema e limitando as possibilidades de ocorrer danos no recurso.

Algumas abstrações foram criadas para facilitar o entendimento do funcionamento do ambiente. São elas:

- *Home Machine:* Máquina local do usuário onde é executado o MyGrid;
- *Peer Machine:* Máquina servidora (com acesso externo - internet) de um determinado *site* onde é executado o *Peer* OurGrid;
- *Grid Machine:* Máquina da grade (recurso).

A Figura 24 ilustra a grade do ambiente OurGrid, destacando a utilização de todos os componentes citados acima, com exceção do mecanismo SWAN (que pode ser executado em conjunto com os *User Agents* - UA).

O processo de execução de uma aplicação paralela na grade OurGrid é bastante simples. O usuário, vinculado a um determinado *site*, deve instalar em uma máquina local (*home machine*) os componentes do MyGrid responsáveis pelo escalonamento local das aplicações e pela interface com as máquinas remotas (*grid machines*) pertencentes à grade. Além disso, o usuário precisa conhecer o endereço de um *Peer* qualquer, esteja ele instalado em um máquina servidora (*peer machine*) no mesmo *site* do usuário, ou em um outro *site* qualquer. É através deste *Peer* que os endereços das máquinas disponíveis na grade serão obtidos. Isso porque os recursos ociosos de um determinado domínio, se anunciam para o *Peer* local através dos *User Agents*, ou seja, o *Peer* mantém uma lista das máquinas disponíveis que é constantemente atualizada. Depois de realizada a comunicação MyGrid - *Peer*, os recursos são fornecidos ao escalonador do

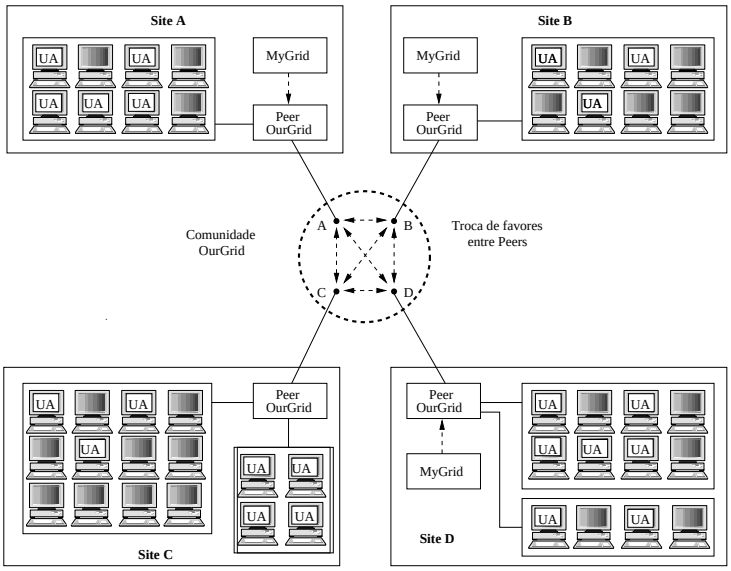

Figura 24 – A grade do ambiente OurGrid.

usuário e, a partir desse momento, a submissão da aplicação já pode ser realizada diretamente pelo escalonador aos recursos, através da comunicação MyGrid - *User Agent*.

Nos casos em que o *Peer* não conseguir todos os recursos que o usuário deseja, ele pode buscá-los na comunidade, através da comunicação com outros *Peers* que a compõem. A forma pela qual o pedido de um *Peer* remoto é atendido é baseada no modelo de **troca de favores** [72]. Basicamente, um *Peer* irá favorecer àqueles que mais lhe cederam recursos anteriormente.

Quanto aos algoritmos de escalonamento, o MyGrid utiliza dois: WQR (*WorkQueue with Replication*) e *Storage Affinity*.

O WQR funciona da seguinte forma: cada uma das tarefas que compõem a aplicação do usuário é distribuída para uma das máquinas da grade cedidas ao usuário. Se sobrarem máquinas para as quais não foram atribuídas tarefas, ou após todas as tarefas existentes já terem sido escalonadas e algumas máquinas já terem completado as suas tarefas, as tarefas ainda não completadas são replicadas nessas máquinas disponíveis. Quando uma das instâncias da tarefa finalizar seu processamento, as demais são canceladas.

O *Storage Affinity* também trabalha com réplicas, da mesma forma que o WQR. A sua diferença é que, para decidir quais tarefas são enviadas para quais recursos, ele leva em conta a existência dos arquivos utilizados pelas tarefas nos recursos, e vai escalonar a tarefa preferencialmente naqueles recursos que irão exigir uma menor transferência de arquivos. Para realizar o armazenamento dos arquivos, o usuário determina explicitamente na execução das tarefas quais devem ser armazenados e quais serão utilizados apenas durante a execução da tarefa.

5.1.4 Modelos Paralelos do ICTM

Depois de identificar as formas de particionamento do ICTM, de relacionar estas formas com as características das plataformas de execução de aplicações paralelas e de definir as tecnologias

que seriam utilizadas para implementação de cada forma de paralelizar o modelo, partiu-se para a definição dos modelos paralelos do ICTM.

É importante ressaltar, que a forma de particionamento em células não foi considerada devido aos possíveis problemas de desempenho citados anteriormente (Seção 5.1.1).

Execução em máquinas agregadas

Para a paralelização das formas de particionamento em camadas e funções, optou-se pela utilização do mesmo modelo de programação paralela, qual seja, o modelo “mestre-escravo”.

Esta escolha deve-se ao fato de que as duas formas de particionamento possuem algumas características semelhantes, entre elas: a necessidade de um processo responsável pela distribuição das tarefas e envio dos dados geográficos e parâmetros do modelo aos demais processos que efetivamente executam as operações do modelo; e a necessidade de comunicação apenas entre o processo “coordenador” e os processos “trabalhadores”. A única diferença entre as formas de paralelização está relacionada a granulosidade do problema, ou seja, a forma de particionamento em camadas apresenta uma granulosidade grossa e a forma de particionamento em funções uma granulosidade média. No entanto, esta diferença não impede o uso do mesmo modelo de programação.

Sendo assim, as tarefas (camadas e funções paralelas) são divididas pelo mestre e enviadas aos escravos. Estes, realizam os cálculos necessários e comunicam ao mestre que a tarefa foi finalizada, enviando os resultados se necessário. Para o caso do particionamento em camadas, cada processo escravo gera sua própria saída, ou seja, os resultados não são enviados ao mestre, o processo escravo apenas comunica que já finalizou e solicita mais trabalho. Já no caso do particionamento em funções, cada processo escravo calcula a função paralela, envia os resultados e solicita mais trabalho ao mestre. Isso se repete até o mestre comunicar aos escravos que as tarefas terminaram. A Figura 25 apresenta o modelo em questão.

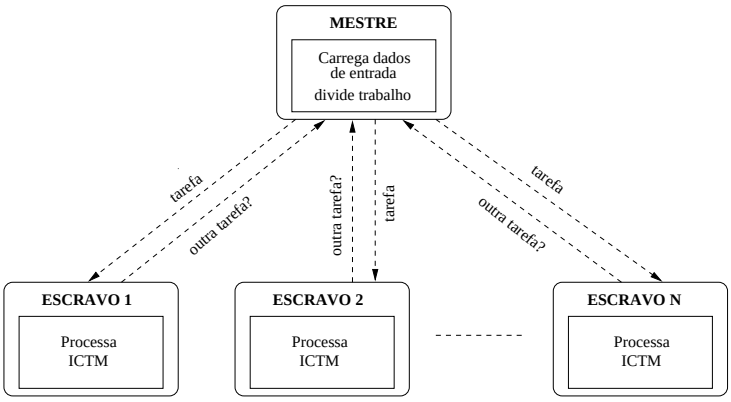

Figura 25 – Modelo paralelo do ICTM para máquinas agregadas: execução das formas de particionamento em camadas e funções.

Observe que o número de tarefas é fixo em ambos os casos, ou seja, no caso das camadas o número de tarefas é igual ao número de camadas analisadas, e no caso das funções o número de

tarefas é igual ao número de funções paralelizáveis do modelo.

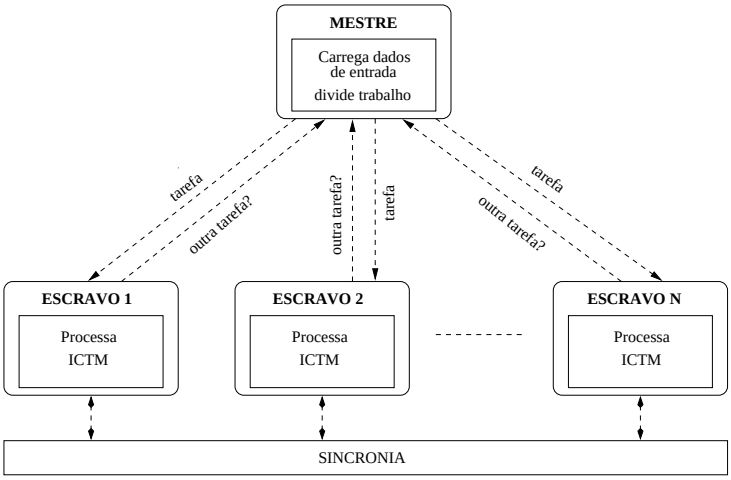

Figura 26 – Modelo paralelo do ICTM para máquinas agregadas: execução da forma de particionamento em domínios.

Para a paralelização da forma de particionamento em domínios, foi utilizado um modelo híbrido que combina características do modelo mestre-escravo com características do modelo fases-paralelas (Figura 26).

Conforme mencionado na Seção 5.1.2, esta é uma alternativa que se adapta perfeitamente com esta maneira de paralelizar o modelo ICTM, visto que os modelos de programação paralela utilizados individualmente não solucionam o problema como um todo.

Os blocos de células são divididos pelo mestre e enviados aos escravos. Estes, realizam os cálculos necessários até um determinado ponto (células da borda que necessitam de valores das células da borda de outros blocos), comunicam com outros processos para buscar informações necessárias e continuam calculando. Quando terminam, retornam os resultados ao mestre e solicitam mais trabalho. Esse processo é repetido até o mestre informar que não existe mais trabalho.

Execução em grades computacionais

Para realizar a modelagem do ICTM para grades computacionais foi necessário estudar como a forma de particionamento em camadas do modelo seria adaptada a uma aplicação Bot, e como essa aplicação Bot seria descrita em um serviço no ambiente OurGrid.

No OurGrid, um serviço (*job*) é descrito em tarefas que possuem três fases distintas. São elas:

- *Init:* Esta fase é responsável pelo envio das informações (aplicação e dados) da máquina do usuário (*home machine*) para a máquina remota (*grid machine*);
- *Remote:* Esta fase é responsável pela execução da aplicação na máquina remota;

- *Final:* Esta fase é responsável pelo envio dos resultados da máquina remota para a máquina do usuário.

De acordo com esta definição, o ICTM multi-camada pode ser adaptado facilmente, ou seja, basta criar um *job* com n tarefas, onde n é o número de camadas que serão analisadas, colocando em cada uma das fases as seguintes informações: na fase *put*, os dados referentes a cada camada e o código seqüencial do ICTM; na fase *remote*, a instrução para que seja executado o modelo para os dados de entrada fornecidos; e na fase *final*, a instrução para que o usuário receba os resultados da camada analisada. A Figura 27 mostra este modelo de execução do ICTM no ambiente OurGrid.

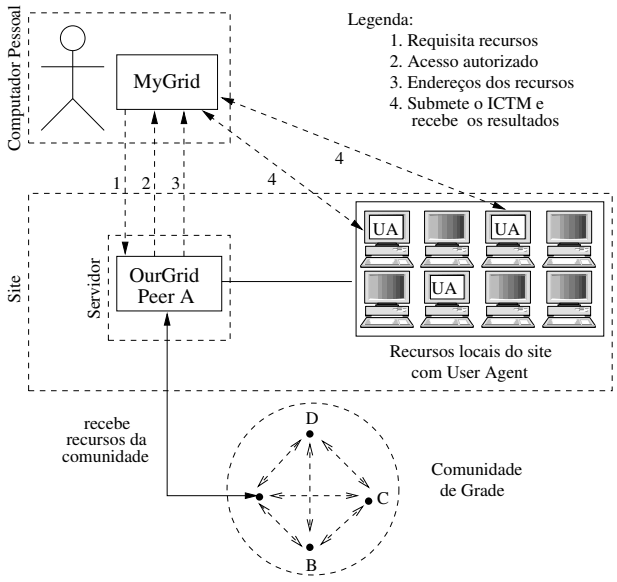

Figura 27 – Modelo paralelo do ICTM para grades: execução da forma de particionamento em camadas.

O processo de execução do modelo ICTM no ambiente OurGrid é exatamente o mesmo que foi descrito na Seção 5.1.3. De acordo com a figura, o primeiro passo é a requisição de recursos por parte do escalonador do usuário (MyGrid) ao *Peer* OurGrid. O *Peer* recebe o pedido e se encarrega de buscar os recursos necessários. Essa busca por recursos é feita de forma automática, ou seja, os recursos ociosos locais se anunciam ao *Peer* (através dos *User Agents*) que mantém uma lista de máquinas disponíveis. Se o pedido não for totalmente atendido localmente, o *Peer* inicia a comunicação com outros *Peers* da comunidade a fim de buscar novos recursos ociosos em outros *sites*. De posse dos endereços dos recursos, o *Peer* envia os mesmos para o escalonador do usuário e, a partir deste momento, a submissão do *job* ICTM já pode ser realizada diretamente para as máquinas da grade, ou seja, a comunicação entre escalonador e recurso é feita de forma direta.

5.2 Implementação dos Modelos Paralelos

A implementação dos modelos paralelos do ICTM para máquinas agregadas e grades computacionais foi realizada através das linguagens C e C++, seguindo o padrão das versões seqüenciais, da biblioteca de programação paralela MPI e do ambiente de grade OurGrid. O sistema operacional usado foi o Linux, mantendo também o padrão anterior.

Foram implementadas três versões paralelas para máquinas agregadas, uma para cada forma de particionamento, de acordo com os modelos paralelos definidos. Para a plataforma de grades computacionais, realizou-se apenas a adaptação do ICTM multi-camada para o ambiente OurGrid.

As seções seguintes apresentam os detalhes de implementação para cada plataforma paralela.

5.2.1 Versões para Máquinas Agregadas

A primeira versão paralela implementada foi a do modelo mestre-escravo para a forma de particionamento em camadas.

Nesta implementação, a comunicação entre os processos ocorre em dois momentos distintos: (i) quando o processo mestre envia aos processos escravos as tarefas (camadas) a serem realizadas com os respectivos dados de entrada e os parâmetros necessários para o método de categorização; (ii) e quando um processo escravo avisa o processo mestre que a sua tarefa já foi finalizada e solicita mais trabalho.

É importante ressaltar que cada processo escravo executa todas as funções do modelo para uma determinada propriedade. Ao finalizar os cálculos, o próprio processo escravo gera uma saída, ou seja, produz a categorização para a camada em questão.

A quantidade de processos paralelos pode variar de 2 até $n + 1$, onde n é o número de camadas analisadas. Ou seja, o número mínimo de processos é 2, um mestre e um escravo, e o número máximo é $n + 1$, um mestre e n escravos. O comportamento do processo mestre e dos processos escravos é apresentado nos Algoritmos 1 e 2, respectivamente.

No Algoritmo 1, o primeiro passo é o envio dos parâmetros e das camadas/propriedades do modelo para cada um dos processos paralelos existentes (linhas 1-6). Logo a seguir, o processo mestre entra em um laço de repetição (linhas 7-19) esperando o recebimento das mensagens dos processos escravos, que avisam que uma camada foi processada e solicitam mais trabalho. O processo mestre então, recebe a mensagem de um processo escravo (linha 8) e se existirem mais tarefas – camadas a serem analisadas – o processo mestre envia mais trabalho a este escravo (linhas 10-14). Se não existirem mais tarefas, o processo mestre envia uma mensagem de final ao escravo em questão (linhas 15-18).

Algoritmo 1: Processo mestre no particionamento em camadas.

Entrada: tarefas = número de camadas.
Entrada: camada = 1, contador de camada a ser enviada.
Entrada: camadaenv = 0, contador de camada enviada.
Entrada: camadaproc = 0, contador de camada processada.
Entrada: sinal = -9, sinal de parada.

```
1  para cada Processo Escravo faça
2      Envia parâmetros do modelo;
3      Envia uma camada a um escravo E1 (escravo que recebeu a primeira tarefa);
4      camada = camada + 1;
5      camadaenv = camadaenv + 1;
6  fim para cada
7  enquanto camadaproc < tarefas faça
8      Recebe aviso de um escravo E2: camada x processada;
9      camadaproc = camadaproc + 1;
10     se camadaenv < tarefas então
11         camadaenv = camadaenv + 1;
12         camada = camadaenv;
13         Envia uma camada ao escravo E2 (escravo que já processou uma tarefa)
14     fim se
15     senão
16         camada = sinal;
17         Envia mensagem de final para o escravo E2 (conteúdo de "camada");
18     fim se
19 fim enqto
```

No Algoritmo 2, o primeiro passo é o recebimento das mensagens enviadas pelo processo mestre com os parâmetros do modelo e a primeira tarefa a ser executada (linhas 1 e 2). Depois disso, o processo escravo entra em um laço de repetição (linhas 3-7) onde ele executa o modelo ICTM para a primeira camada recebida (linha 4), avisa o mestre que a tarefa foi processada e solicita mais trabalho (linha 5). Este procedimento se repete até a recepção de um sinal de parada, ou seja, o processo mestre envia uma tarefa com um código indicativo de término de processamento.

Algoritmo 2: Processo escravo no particionamento em camadas.

Entrada: sinal = -9, sinal de parada.

```
1  Recebe parâmetros do modelo;
2  Recebe tarefa (camada) do mestre;
3  enquanto camada != sinal faça
4      Executa o modelo ICTM para a camada recebida;
5      Avisa o mestre e solicita mais trabalho;
6      Recebe tarefa (camada) do mestre;
7  fim enqto
```

A segunda versão paralela implementada foi a do modelo mestre-escravo para a forma de particionamento em funções.

Nesta implementação, a comunicação entre os processos ocorre quando: (i) o processo mestre envia aos processos escravos as tarefas (funções paralelas do modelo) a serem realizadas com os respectivos dados de entrada e os parâmetros necessários para o método de categorização; (ii) e um processo escravo avisa o processo mestre que a sua tarefa já foi finalizada e solicita mais trabalho.

Ao contrário da versão anterior, onde o mestre apenas divide o trabalho e envia aos escravos, nesta versão o mestre calcula determinadas funções do modelo, quais sejam, as funções não paralelizáveis. Sendo assim, o mestre inicia o processo de categorização da propriedade em questão e no momento em que identifica as funções paralelas, ele atribui as mesmas para diferentes processos escravos, aguardando os resultados. Este procedimento se repete até não existirem mais funções paralelizáveis. O resultado final é produzido pelo processo mestre que executa por último a função responsável pela definição das categorias (geração da matriz de limites).

É importante destacar a necessidade de se controlar os passos seqüenciais do processo de categorização do ICTM. Para isso, foram criadas etapas que dividiram este processo (considerando o caso bidimensional):

- Primeira Etapa: Cálculo das funções referentes a geração das matrizes intervalares simples. São geradas 2 matrizes para cada eixo do modelo (X e Y);
- Segunda Etapa: Cálculo das funções referentes a geração das matrizes intervalares com raio de vizinhança maior do que 1. São geradas 4 matrizes, de acordo com o modelo de vizinhança adotado (à direita, esquerda, acima e abaixo);
- Terceira Etapa: Cálculo das funções referentes a geração das matrizes de monotonicidade. São geradas 4 matrizes para as 4 direções (à esquerda, direita, acima e abaixo) do modelo de vizinhança adotado.

Tabela 13 – Divisão das funções paralelizáveis do modelo.

Número	**Etapa**	**Descrição da Função**
1	Primeira	Matriz Intervalar eixo X
2	Primeira	Matriz Intervalar eixo Y
3	Segunda	Matriz Intervalar eixo X com raio à direita
4	Segunda	Matriz Intervalar eixo X com raio à esquerda
5	Segunda	Matriz Intervalar eixo Y com raio abaixo
6	Segunda	Matriz Intervalar eixo Y com raio acima
7	Terceira	Matriz de Monotonicidade à direita
8	Terceira	Matriz de Monotonicidade à esquerda
9	Terceira	Matriz de Monotonicidade abaixo
10	Terceira	Matriz de Monotonicidade acima

Sendo assim, uma etapa só pode ser iniciada após a conclusão da etapa anterior. No caso da primeira etapa, é preciso que a matriz absoluta e a matriz relativa do modelo já tenham sido calculadas. A segunda etapa só pode ser executada após o término da primeira e a terceira após o término da segunda. Com isso, o número máximo de processos paralelos em tempo de execução é 4 devido ao número máximo de funções paralelas em uma etapa. Além disso, cada função paralela do modelo foi numerada de 1 a 10 (Tabela 13). Este procedimento facilitou a programação desta versão do modelo.

Quanto ao número total de processos paralelos, embora existam 10 funções independentes, a variação é de 2 até 5. Isso porque o número máximo de funções paralelas que podem ser executadas ao mesmo tempo é 4. Assim, o número mínimo de processos é 2, um mestre e um escravo, e o número máximo é 5, um mestre e 4 escravos. O Algoritmo 3 apresenta o comportamento do processo mestre e o Algoritmo 4 o comportamento dos processos escravos.

Algoritmo 3: Processo mestre no particionamento em funções.

```
Entrada: func = 0, contador de função a ser enviada.
Entrada: etapa = 0, contador de etapas.
Entrada: nescravo = 0, contador de escravos receptores de tarefas.
Entrada: sinal = -9, sinal de parada.
1  Calcula matriz absoluta e matriz relativa;
2  para cada Processo Escravo faça
3  |  Envia parâmetros do modelo;
4  fim para cada
5  enquanto func < 10 faça
6  |  se etapa = 0 então
7  |  |  para cada Processo Escravo faça
8  |  |  |  nescravo = nescravo + 1;
9  |  |  |  se nescravo <= 2 então
10 |  |  |  |  func = func + 1;
11 |  |  |  |  Envia função e dados necessários para escravo;
12 |  |  |  |  se nescravo = 2 então
13 |  |  |  |  |  etapa = etapa + 1;
14 |  |  |  |  |  Interrompe laço e vai para próxima etapa;
15 |  |  |  |  fim se
16 |  |  |  fim se
17 |  |  fim para cada
18 |  |  Recebe resultados da etapa dos escravos (número de escravos = nescravos);
19 |  |  nescravos = 0;
20 |  fim se
21 |  Repete linhas 3-17 para as outras etapas, considerando 4 funções paralelas;
22 fim enqto
23 func = sinal;
24 Envia mensagem de final para os escravos (conteúdo de "func");
25 Calcula matriz de estados e de limites;
26 Gera resultado final;
```

No Algoritmo 3, o primeiro passo é a execução das primeiras funções do modelo ICTM, o cálculo da matriz absoluta e da matriz relativa (linha 1). A seguir, os parâmetros do modelo são enviados a todos os processos escravos (linhas 2-4). Depois disso, o processo mestre entra em um laço de repetição (linhas 5-22) para o envio das tarefas (funções paralelas) aos processos escravos e o recebimento dos resultados, de acordo com a divisão por etapas mencionada anteriormente. Na primeira etapa (linhas 6-20), apenas as funções 1 e 2 são enviadas (linha 11) como tarefas ao(s) escravo(s). Observe que nesse ponto, apenas 3 processos estarão executando. Os demais, que por ventura podem existir, estarão aguardando por tarefas. Isso acontece devido ao processo de categorização do ICTM obedecer uma seqüência de passos. Depois de enviar as duas funções da primeira etapa, o processo mestre aguarda o recebimento dos resultados (linha 18) desta etapa e passa para a próxima. Este laço de repetição é repetido para as outras duas etapas (linha 21), considerando o envio de 4 funções paralelas em cada etapa. Após receber os resultados provenientes do processamento das funções paralelas, o processo mestre envia uma mensagem de final aos escravos (linha 24), executa as últimas duas funções do modelo (linha 25), o cálculo da matriz de estados e de limites, e gera o resultado final (linha 26).

No Algoritmo 4, o primeiro passo é o recebimento das mensagens enviadas pelo processo mestre com os parâmetros do modelo e a primeira tarefa a ser executada (linhas 1 e 2). Depois disso, o processo escravo entra em um laço de repetição (linhas 3-8) onde são recebidos, quando necessário, os dados para o processamento da função em questão (linha 4), é executado o modelo ICTM para a primeira função paralela recebida (linha 5), e o processo mestre é avisado sobre o processamento da tarefa e a necessidade de mais trabalho (linha 6). Este procedimento se repete até a recepção de um sinal de parada, ou seja, o processo mestre envia uma tarefa com um código indicativo de término de processamento, semelhante ao fim de execução dos processos escravos na versão anterior.

Algoritmo 4: Processo escravo no particionamento em funções.

```
   Entrada: sinal = -9, sinal de parada.
1  Recebe parâmetros do modelo;
2  Recebe tarefa (função - func) do mestre;
3  enquanto func != sinal faça
4  |   Recebe dados, quando necessário, para cálculo da função;
5  |   Calcula função recebida;
6  |   Envia resultados ao mestre;
7  |   Recebe tarefa (função - func) do mestre;
8  fim enqto
```

A terceira e última versão paralela implementada para a plataforma de máquinas agregadas foi a de um modelo híbrido, que combina características do modelo mestre-escravo com o modelo fases-paralelas, para a forma de particionamento em domínios.

Nesta implementação, a comunicação entre os processos ocorre em três momentos: (i) quando o processo mestre envia aos processos escravos as tarefas (blocos de células) a serem realizadas com os respectivos dados de entrada e os parâmetros necessários para o método de categorização; (ii) quando os processos escravos param de calcular e se comunicam com o

intuito de obter as informações necessárias (valores das células das bordas dos blocos) para o término do processamento da tarefa; (iii) e quando um processo escravo avisa o processo mestre que a sua tarefa já foi finalizada, envia os resultados e solicita mais trabalho.

Nesta versão, o processo mestre tem a responsabilidade de dividir a malha total que representa a região geográfica em blocos de células, de acordo com uma definição inicial sobre a quantidade de blocos que deve ser criada. Além disso, o processo mestre deve enviar os blocos aos escravos, obter os resultados e enviar novos blocos se for o caso. Já os processos escravos, por sua vez, executam todas as funções do modelo ICTM para cada bloco. Para o término do processamento de cada uma das funções, cada processo escravo sempre irá se comunicar com os demais, a fim de obter os valores das células vizinhas que estão nas bordas de outros blocos. Ao final do processamento de todos os blocos (tarefas), o processo mestre gera o resultado final, ou seja, a categorização da propriedade analisada.

Quanto ao número total de processos paralelos, a variação é de 2 até $n + 1$, onde n agora é o número de blocos que serão criados. Ou seja, o número mínimo de processos é 2, um mestre e um escravo, e o número máximo é $n + 1$, um mestre e n escravos. O Algoritmo 5 apresenta o comportamento do processo mestre e o Algoritmo 6 o comportamento dos processos escravos.

Algoritmo 5: Processo mestre no particionamento em domínios.

```
   Entrada: tarefas = número de blocos.
   Entrada: bloco = 1, contador de bloco a ser enviado.
   Entrada: blocoenv = 0, contador de bloco enviado.
   Entrada: blocoproc = 0, contador de bloco processado.
   Entrada: sinal = -9, sinal de parada.
 1 para cada Processo Escravo faça
 2 |   Envia parâmetros do modelo;
 3 |   Envia um bloco a um escravo E1 (escravo que recebeu a primeira
   |   tarefa);
 4 |   bloco = bloco + 1;
 5 |   blocoenv = blocoenv + 1;
 6 fim para cada
 7 enquanto blocoproc < tarefas faça
 8 |   Recebe aviso de um escravo E2: bloco x processado;
 9 |   blocoproc = blocoproc + 1;
10 |   se blocoenv < tarefas então
11 |   |   blocoenv = blocoenv + 1;
12 |   |   bloco = blocoenv;
13 |   |   Envia um bloco a um escravo E2 (escravo que já processou uma
   |   |   tarefa)
14 |   fim se
15 |   senão
16 |   |   bloco = sinal;
17 |   |   Envia mensagem de final para o escravo E2 (conteúdo de "bloco");
18 |   fim se
19 fim enqto
```

O comportamento do processo mestre nesta implementação é bastante semelhante ao comportamento do processo mestre da versão referente ao particionamento em camadas apresentada no Algoritmo 1. O primeiro passo é o envio dos parâmetros e das tarefas (blocos de células) para cada um dos processos paralelos existentes (linhas 1-6). Logo a seguir, o processo mestre entra em um laço de repetição (linhas 7-19) esperando o recebimento das mensagens dos processos escravos, avisando que o bloco foi processado e solicitando mais trabalho. O processo mestre então, recebe a mensagem de um processo escravo (linha 8) e se existirem mais tarefas – blocos a serem analisados – o processo mestre envia mais trabalho a este escravo (linhas 10-14). Se não existirem mais tarefas, o processo mestre envia uma mensagem de final ao escravo em questão (linhas 15-18).

Algoritmo 6: Processo escravo no particionamento em domínios.

```
   Entrada: sinal = -9, sinal de parada.
 1 Recebe parâmetros do modelo;
 2 Recebe tarefa (bloco) do mestre;
 3 enquanto bloco != sinal faça
 4 |   para cada Coluna de Células do Bloco faça
 5 |   |   se A Coluna não for da Borda então
 6 |   |   |   Executa funções do modelo ICTM;
 7 |   |   fim se
 8 |   |   senão
 9 |   |   |   Comunica com escravos vizinhos para obter valores das
   |   |   |   células vizinhas;
10 |   |   |   Finaliza cálculo do bloco;
11 |   |   fim se
12 |   fim para cada
13 |   Envia resultados ao mestre;
14 |   Recebe tarefa (bloco) do mestre;
15 fim enqto
```

Quanto ao comportamento dos processos escravos apresentado no Algoritmo 6, o primeiro passo é o recebimento das mensagens enviadas pelo processo mestre com os parâmetros do modelo e da primeira tarefa (bloco) a ser executada (linhas 1 e 2). A seguir, o trabalho dos escravos é dividido em duas fases: fase de *cálculo* e fase de *comunicação*. Primeiramente, inicia-se a fase de cálculo onde o processo escravo entra em um laço de repetição (linhas 3-15). Neste laço, executa-se o modelo ICTM para cada coluna de células do bloco, verificando se a mesma não faz parte da borda do bloco (linhas 4-12). Se a coluna fizer parte da borda do bloco, o processamento é interrompido e inicia-se a fase de comunicação entre os processos (linhas 8 e 9), com o objetivo de trocar informações referentes aos valores das células das bordas dos blocos. Após a troca de informações (realizada até um processo receber todos os valores de uma coluna de outro processo), a fase de cálculo é reiniciada e o processamento do bloco é finalizado (linha 10). Por fim, o escravo envia os resultados ao mestre e aguarda o recebimento de outra tarefa (linhas 13 e 14).

Este procedimento de alternância entre fases distintas caracteriza o modelo fases-paralelas. O modelo utilizado é chamado de híbrido, pois baseia-se também no uso de um processo coordenador (mestre) que divide o trabalho total e recebe os resultados de cada processo trabalhador (escravos), ou seja, utiliza-se também o modelo de programação mestre-escravo.

5.2.2 Adaptação para a Grade OurGrid

De acordo com o exposto na Seção 5.1.4, o processo de adaptação do modelo ICTM multicamada para o ambiente de grade OurGrid foi bastante simples.

A tarefa principal foi adaptar o modelo para um serviço (*job*) no OurGrid. Isso é feito através da definição de um arquivo de descrição de serviço (*jdf - job description file*). A seguir apresenta-se um exemplo de *jdf* para um *job* chamada "*SimpleJob*" (vem junto com o pacote OurGrid) que possui duas tarefas para calcular o fatorial de um determinado número:

```
-------------------------------------------------------------
job :
label   : SimpleJob

task :
init    : put Fat.class Fat.class
remote  : nice java -cp . Fat 3 261147332 6819792792357414911
          output-$JOB.$TASK
final   : get output-$JOB.$TASK output-$JOB.$TASK

task :
init    : put Fat.class Fat.class
remote  : nice java -cp . Fat 261147332 522294661
          6819792792357414911 output-$JOB.$TASK
final   : get output-$JOB.$TASK output-$JOB.$TASK
-------------------------------------------------------------
```

Observe a existência de duas cláusulas, *job* e *task*. A cláusula *job* possui um atributo chamada *label*, que define um rótulo para o *job* (no caso *SimpleJob*). Já a cláusula *task* possui três atributos – *init*, *remote* e *final* – que definem as fases de uma tarefa (veja Seção 5.1.4).

Depois de definido o *jdf* basta executar o mesmo na grade para que as tarefas sejam processadas. O que é feito no *job SimpleJob* é o seguinte: (i) transferência do código java (*Fat.class*) para a máquina remota; (ii) execução do código java transferido anteriormente e colocação do resultado do processamento em um arquivo (*output*); (iii) e retorno dos resultados armazenados no arquivo da máquina remota para a máquina local.

Voltando ao contexto do ICTM, pode-se criar um *jdf* que reproduza naturalmente as informações do modelo. Para o caso referente a categorização de três propriedades (camadas) diferentes de uma determinada região geográfica, o *jdf* correspondente seria estruturado da seguinte forma:

```
-------------------------------------------------------------
job :
label   : ICTM

task :
init    : put executavel_ictm executavel_ictm
          put entrada_camada1 entrada_camada1
remote  : executavel_ictm entrada_camada1 >  output-$TASK
final   : get output-$TASK output-$TASK

task :
init    : put executavel_ictm executavel_ictm
          put entrada_camada2 entrada_camada2
remote  : executavel_ictm entrada_camada2 >  output-$TASK
final   : get output-$TASK output-$TASK

task :
init    : put executavel_ictm executavel_ictm
          put entrada_camada3 entrada_camada3
remote  : executavel_ictm entrada_camada3 >  output-$TASK
final   : get output-$TASK output-$TASK
-------------------------------------------------------------
```

Com este *jdf*, o que seria feito é o seguinte: (i) transferência do executável do código seqüencial do ICTM e do arquivo de entrada da propriedade (camada) a ser analisada para a máquina remota; (ii) execução do programa para o conjunto de dados fornecido e colocação dos resultados no arquivo de saída; (iii) e envio do arquivo de saída com os resultados da máquina remota para a máquina local.

Entretanto, é importante ressaltar que para arquivos de entrada muito grandes o processo de envio dos mesmos pode ser muito lento. Portanto, o ideal seria adotar a idéia de que as informações das propriedades que serão analisadas são disponibilizadas localmente pelos provedores de recursos (interessados em fornecer seus dados). Com isso, a etapa de envio do arquivo de entrada seria desnecessária, o que eliminaria o problema em questão. Quanto ao recebimento da saída, para arquivos de entrada muito grandes o arquivo de saída também pode ser um problema. É interessante que os arquivos, dependendo do tamanho, sejam compactados localmente antes de serem enviados a máquina destino. Este procedimento pode reduzir bastante o tempo de execução da aplicação.

5.3 Definição do Protótipo

Com o intuito de reunir todas as funcionalidades desenvolvidas em um único sistema, definiu-se um protótipo do modelo HPC-ICTM. Através deste sistema, pode-se executar o modelo localmente, em uma estação de trabalho, em uma máquina agregada ou em um ambiente

de grade. A Figura 28 apresenta os módulos do protótipo.

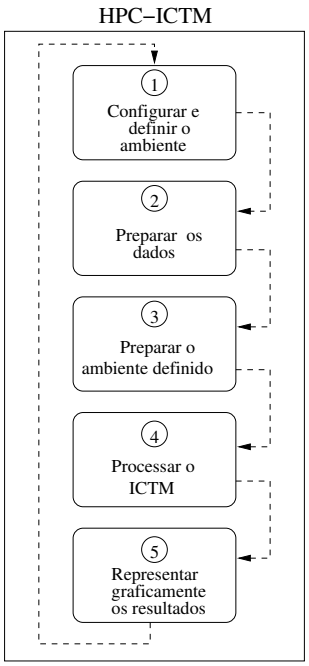

Figura 28 – Módulos do protótipo do modelo HPC-ICTM.

O primeiro passo para utilizar o protótipo é atualizar os arquivos de configuração das plataformas paralelas, chamados de `cluster-conf` e `grade-conf`. Nestes arquivos estão as definições necessárias para o uso de ambas plataformas. No arquivo `cluster-conf` as seguintes informações são solicitadas: nome da máquina agregada, quantidade de nodos necessários, arquitetura dos nodos (32/64 *bits*), rede de interconexão e tempo de alocação dos nodos. No caso do arquivo `grade-conf` as informações necessárias são: localização do arquivo de descrição da grade (*gdf - grid description file*), localização do arquivo de descrição do serviço (*jdf*) e utilização da interface de visualização da execução (sim/não).

É importante destacar que estes arquivos foram criados com base nas tecnologias utilizadas no CPAD, ou seja, considerou-se o uso do CRM desenvolvido no próprio centro (CRONO), da implementação MPICH do MPI e do ambiente de grade OurGrid. Todas elas instaladas e configuradas adequadamente. Além disso, considera-se o uso de sistemas de arquivos compartilhados. No caso de mudança de alguma tecnologia específica, é preciso modificar os arquivos de configuração e alguns trechos de código de alguns módulos do protótipo.

Além destes dois arquivos de configuração das plataformas, existe um terceiro arquivo de configuração chamado de `modelo-conf`. Neste arquivo são definidas as questões referentes a forma de particionamento desejada, dimensão da região a ser analisada e quantidade de camadas que serão processadas.

Após a realização da atualização dos arquivos de configuração, já é possível rodar o protótipo. Para isso, é preciso disparar um *script* que executa cada módulo do sistema de forma

ordenada, passando como parâmetro o arquivo contendo os dados geográficos que serão analisados.

O primeiro módulo a ser executado é o módulo 1. Ele é responsável pela configuração e definição do ambiente de execução. Ou seja, neste módulo é que serão configuradas as plataformas de execução, de acordo com a escolha realizada pelo usuário. O usuário pode optar por executar o modelo na máquina local, em uma máquina agregada, ou em uma grade computacional.

Neste ponto, o programa pergunta ao usuário aonde ele quer executar o modelo ICTM e oferece as três opções para escolha: local - máquina agregada - grade. De acordo com a escolha do usuário, o próprio sistema configura o ambiente de execução com base nos arquivos de configuração de plataformas citados anteriormente. Se a escolha for "local", nenhuma configuração precisa ser realizada.

A seguir, o sistema executa automaticamente o módulo 2. Neste módulo é feita a preparação dos dados da região geográfica que será analisada. De acordo com a plataforma escolhida, o arquivo de entrada fornecido será preparado: conversão do formato de dados da imagem obtida para o formato de entrada do modelo; e disponibilização adequada dos dados para a plataforma escolhida. Para realizar esta tarefa, o sistema verifica as informações inseridas no arquivo `modelo-conf`. De posse dessas informações, o sistema sabe qual forma de particionamento será empregada e em qual plataforma isso irá acontecer e então, prepara os dados para execução.

Depois disso, também de forma automática, o sistema executa o módulo 3. Este módulo é responsável por disponibilizar os dados de entrada formatados no padrão de entrada do modelo e o arquivo fonte do modelo (de acordo com a plataforma escolhida e a forma de particionamento configurada) em um diretório pré-definido que será utilizado para execução do HPC-ICTM. No caso da utilização de máquinas agregadas, por exemplo, três versões foram implementadas e, por isso, existem três arquivos fontes diferentes.

Continuando o processo automático realizado pelo *script* criado, o sistema executa o módulo 4. Ele é responsável por executar o modelo, a partir do diretório pré-definido, na plataforma escolhida e gerar os resultados desejados. Estes resultados são arquivos texto que contém os mapas de limites que definem as categorizações.

Já o módulo 5 é uma etapa extra que produz uma visualização gráfica para representar a região analisada, destacando as categorias encontradas. Este módulo não é executado de forma automática pelo sistema. É preciso que o usuário execute manualmente o programa que gera a representação da região de forma gráfica, a partir do arquivo de entrada que representa a mesma. Este programa pode ser executado exclusivamente para obtenção da representação gráfica, mas pode também ser executado para obtenção de todos os resultados do modelo. Entretanto, este procedimento está restrito a uma única máquina, ou seja, o processamento de grandes regiões pode não ser possível. No caso de se executar todo o modelo, o programa ainda exibe informações sobre todas as matrizes geradas para todas as células da malha.

No que diz respeito a implementação dos módulos do protótipo, utilizou-se as linguagens C e C++ e programação em *shel-script* (*bash script*) para o ambiente *Linux*. Além disso, foi utilizada a biblioteca OpenGL [76] para visualização gráfica dos resultados.

A Figura 29 apresenta a interface do protótipo. Ela é composta por três janelas: (i) a janela de visualização, que apresenta as informações visuais do relevo, sub-áreas e limites; (ii) a janela do console, que apresenta informações das células e da malha e onde são feitas as configurações

para execução do modelo; (iii) e a janela do cardápio de opções (à direita) com as operações disponíveis no sistema (informações de todas as matrizes calculadas).

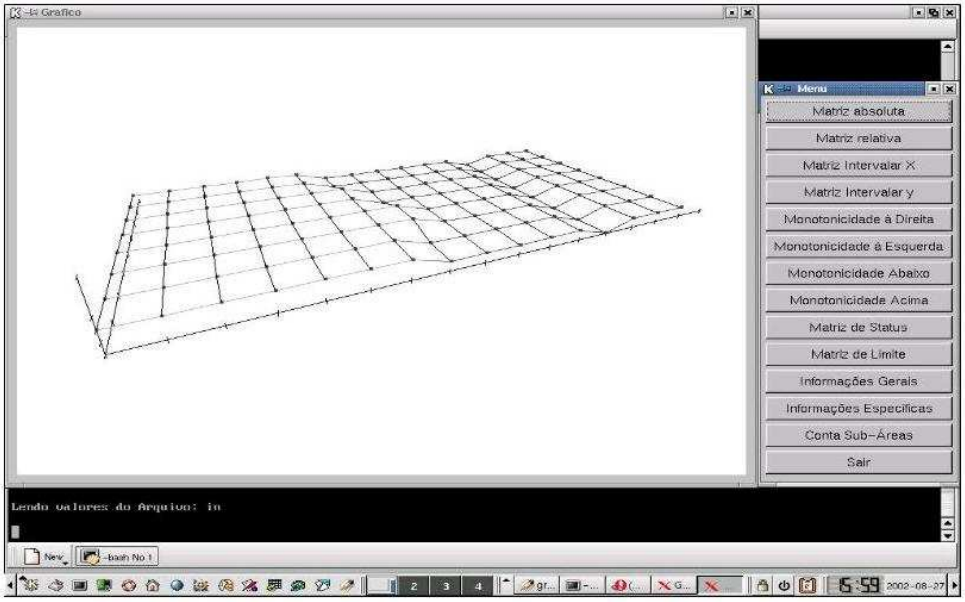

Figura 29 – Interface do protótipo.

6 Validação do HPC-ICTM

Neste capítulo apresenta-se um conjunto de testes realizado com o objetivo específico de validar o modelo HPC-ICTM, ou seja, avaliar o comportamento e analisar o desempenho dos modelos paralelos do ICTM (que compõem o HPC-ICTM) e suas respectivas implementações, para as duas plataformas de execução de aplicações paralelas utilizadas no trabalho.

Primeiramente definiu-se quais seriam as imagens de satélite utilizadas nos testes: localização das regiões geográficas, quantidade de quadrantes a serem analisados e dimensão dos quadrantes. A seguir, partiu-se para a definição da metodologia que seria empregada para realização dos testes: organização dos testes, quantidade de propriedades para cada quadrante, valor do raio de vizinhança em cada teste e forma de comparação dos resultados. Por fim, foi definido o ambiente de teste que seria empregado para cada uma das plataformas e efetuados todos os testes propriamente ditos.

As seções seguintes apresentam os resultados obtidos em cada uma destas etapas, bem como os resultados dos próprios testes e uma análise de desempenho para cada modelo paralelo do HPC-ICTM.

6.1 Conjunto de Dados Geográficos

Para realizar a validação do modelo HPC-ICTM foram utilizadas imagens do satélite LANDSAT[1] e do radar SRTM[2] (*Shuttle Radar Topography Mission*). É importante ressaltar que todos os dados geográficos extraídos das imagens citadas são dados altimétricos, ou seja, referentes a topografia das regiões escolhidas. Sendo assim, para a realização de testes com análise de mais de uma camada (propriedade da região) foram utilizados os mesmos dados de topografia, o que não interfere na obtenção dos resultados desejados visto que a dimensão de cada camada é exatamente a mesma.

A região escolhida foi a da cidade de Pelotas, Rio Grande do Sul, Brasil. Mais precisamente, a região analisada foi dividida em 5 quadrantes em torno da Lagoa Pequena, conforme exibido na Figura 19 utilizada nos testes seqüenciais para o modelo ICTM. As Figuras 30 e 31 são imagens do satélite LANDSAT que exibem de forma geral a área que compreende a cidade de Pelotas e, conseqüentemente, os quadrantes utilizados.

Os quadrantes possuem coordenadas UTM, Fuso 22S (Zona UTM 22, hemisfério sul) e Datum SAD69 (Datum da América do Sul). Eles são representados em malhas de células, conforme o exposto a seguir:

- Quadrante A: Matriz com 241 linhas e 241 colunas (58081 células);

[1] Satélite LANDSAT: http://www.cdbrasil.cnpm.embrapa.br/txt/landsat.htm
[2] Radar SRTM: http://srtm.usgs.gov/

Figura 30 – Cidade de Pelotas.

Figura 31 – Região Sul de Pelotas.

- Quadrante B: Matriz com 577 linhas e 817 colunas (471409 células);
- Quadrante C: Matriz com 1309 linhas e 1765 colunas (2310385 células);
- Quadrante D: Matriz com 1739 linhas e 2164 colunas (3763196 células);
- Quadrante E: Matriz com 4022 linhas e 4670 colunas (18782740 células).

Note que os quadrantes foram escolhidos com uma boa variação do número de células contidas em cada malha. O menor quadrante (quadrante A) é relativamente pequeno, contendo um pouco mais de 58 mil células. Já o maior quadrante (quadrante E) contém mais de 18 milhões de células.

6.2 Metodologia Empregada

Com o intuito de verificar o comportamento e o desempenho de todos os modelos paralelos do ICTM, foram realizados três testes diferentes para a plataforma de máquinas agregadas e dois testes para a plataforma de grades computacionais. Além disso, foi realizado um teste da versão seqüencial do modelo ICTM para todos os quadrantes mencionados na seção anterior, a fim de descobrir o melhor tempo de referência a ser utilizado nas comparações com as versões paralelas.

Os testes para máquinas agregadas foram:

1. Avaliação da versão referente a paralelização de camadas do modelo ICTM. Para este teste foram obtidos os resultados paralelos para os quadrantes A, B e C. Para cada quadrante, considerou-se a análise de 3, 5 e 7 camadas do modelo e o valor do raio de vizinhança de células igual a 1.

2. Avaliação da versão referente a paralelização de funções do modelo ICTM. Neste caso foram obtidos os resultados paralelos para os quadrantes C e D, considerando a análise de uma única camada do modelo e diferentes valores para o raio de vizinhança de células. Os tempos seqüenciais foram obtidos novamente devido à variação do valor do raio.

3. Avaliação da versão referente a paralelização de domínios do modelo ICTM. Neste teste foram obtidos os resultados paralelos para o quadrante E, considerando a análise de uma única camada do modelo, a decomposição da malha que representa a região em diferentes quantidades de blocos e o valor do raio de vizinhança de células igual a 1.

Os testes para grades computacionais foram:

1. Avaliação do ICTM multi-camada em um ambiente de grade real, levando em conta o uso de dados geográficos centralizados. Foram obtidos os resultados paralelos para todos os quadrantes criados. Para cada quadrante, considerou-se a análise de 3, 5 e 7 camadas do modelo.

2. Avaliação da utilização de dados geográficos distribuídos. Foram obtidos os resultados paralelos para os quadrantes C e D, considerando a análise de 3, 5 e 7 camadas do modelo. Neste caso, os dados de cada camada foram armazenados localmente nas máquinas que executaram o modelo.

Para todos os 5 testes mencionados foi realizada uma comparação entre os tempos seqüenciais e os tempos obtidos com a versão paralela do modelo, apresentando todos os resultados encontrados e uma análise de desempenho sobre os mesmos.

6.3 Ambientes de Teste

Os ambientes de teste foram definidos de acordo com a metodologia descrita na seção anterior.

Para os testes seqüenciais foi utilizado o seguinte conjunto de máquinas do CPAD, cada uma com o sistema operacional Linux (distribuição Debian) e os compiladores GNU (`gcc 3.3.5`) e Intel (`icc 8.0`, apenas para máquinas de 64 *bits*):

- `HP E800 Server`: Dois processadores Intel PIII de 1GHz, com 512MB de memória principal e 256KB de *cache* (referida como `E800 Server`);
- *Estação de Trabalho*: Um processador Intel Pentium 4 de 1.6GHz, com 512MB de memória principal e 256KB de *cache* (referida como *Workstation*);
- `HP Compaq dc5000 MT`: Um processador Intel Pentium 4 de 2.8GHz, com 1GB de memória principal e 1024KB de *cache* (referida como `P4`);
- `HP Integrity rx2600 IA64`: Dois processadores Itanium de 64 *bits* de 1.5GHz, com 2GB de memória principal e 1024KB de *cache* (referida como `IA64-dual`).

Para a realização dos testes referentes as versões paralelas do ICTM para execução em máquinas agregadas, foi utilizada a principal máquina agregada do CPAD, chamada de Amazônia. Este multicomputador heterogêneo é composto por 4 tipos diferentes de nodos:

- 8 nodos `HP-E60`, cada um com 2 processadores Pentium III de 550MHz, com 256MB de memória principal e 512KB de *cache* (referidos como `E60`);
- 8 nodos `HP-E800`, cada um com 2 processadores Pentium III de 1GHz, com 256MB de memória principal e 256KB de *cache* (referidos como `E800`);
- 8 nodos `HP Compaq dc5000 MT`, cada um com 1 processador Pentium 4 de 2.8GHz, com 1GB de memória principal e 1024KB de *cache* (referidos como `P4`);
- 5 nodos `HP Integrity rx2600 IA64`, cada um com 2 processadores Itanium de 64 *bits* de 1.5GHz, com 2GB de memória principal e 1024KB de *cache* (referidos como `IA64-dual`);

Além disso, a Amazônia possui dois tipos de rede de interconexão. Uma rede de interconexão *full-duplex* primária Myrinet com latência de 5 microsegundos e vazão de 1.28Gbits/s. E uma rede de interconexão secundária Fast-Ethernet padrão com uma vazão de 100Mbits/s. No entanto, somente a rede Fast-Ethernet interliga todas as máquinas do agregado. A rede Myrinet não está disponível para as máquinas de 64 *bits*. Sendo assim, utilizou-se em todos os testes apenas a rede de interconexão padrão.

No que diz respeito aos *softwares* instalados nos nodos da Amazônia, tem-se a seguinte configuração: implementação `MPICH` da biblioteca MPI; sistema operacional Linux (distribuição Debian); compilador GNU (`gcc 3.3.5`) para todos os nodos; e compilador Intel (`icc 8.0`) para os nodos de 64 *bits*.

Quanto aos testes referentes a plataforma de grades computacionais, utilizou-se o ambiente real[3] de grade do projeto OurGrid que integra vários centros de pesquisa de norte a sul do Brasil, entre eles o CPAD. Mais precisamente, os recursos utilizados foram:

- Uma estação de trabalho do CPAD para instalação e configuração do MyGrid (máquina `acai.cpad.pucrs.br`);
- Um servidor do CPAD – máquina com acesso a internet – para instalação do *Peer* OurGrid do centro (máquina `marfim.cpad.pucrs.br`);
- Os nodos da máquina agregada Amazônia como recursos locais (nodos `amazonia01 - amazonia29`);
- Uma estação de trabalho do LSD (Laboratório de Sistemas Distribuídos) da UFCG - Universidade Federal de Campina Grande na Paraíba, com o MyGrid instalado e configurado (máquina `paru.lsd.ufcg.edu.br`);
- Um servidor do LSD – máquina com acesso a internet – com o *Peer* OurGrid instalado (máquina `dragão.lsd.ufcg.edu.br`).

[3]Status da grade atual: http://pauastatus.lsd.ufcg.edu.br/

Estes recursos foram utilizados de forma específica de acordo com os testes definidos. Esta forma de uso é detalhada na Seção 6.4.2.

6.4 Resultados Obtidos

O primeiro teste realizado está relacionado a obtenção dos tempos seqüenciais para cada um dos quadrantes gerados.

O objetivo principal foi escolher os melhores tempos seqüenciais para cada um dos quadrantes e adotá-los como referência para as comparações com as versões paralelas. A Tabela 14 apresenta os resultados obtidos, considerando o ambiente de teste descrito na seção anterior, os números de camadas para cada quadrante utilizados nos demais testes e o mesmo valor para o raio de vizinhança de células ($raio = 1$, uma única célula vizinha influencia na determinação do estado da célula em questão).

Tabela 14 – Tempos seqüenciais para os 5 quadrantes gerados (nc é o número de camadas analisadas).

Quadrante	nc	E800 Server	*Workstation*	P4	IA64-dual
$A_{(241\times241)}$	1	0.632s	0.593s	0.257s	0.452s
	3	1.838s	1.627s	0.885s	1.119s
	5	3.024s	2.663s	1.449s	1.827s
	7	4.471s	3.763s	2.804s	2.516s
$B_{(577\times817)}$	1	5.483s	4.132s	1.884s	3.234s
	3	15.282s	12.373s	5.681s	7.188s
	5	25.606s	20.514s	9.464s	11.979s
	7	37.045s	28.682s	14.247s	16.818s
$C_{(1309\times1765)}$	1	25.276s	19.971s	8.450s	12.098s
	3	*Swap*	*Swap*	26.711s	33.652s
	5	*Abortado*	*Swap*	45.693s	56.226s
	7	*Abortado*	*Abortado*	*Abortado*	79.568s
$D_{(1739\times2164)}$	1	42.282s	33.242s	15.734s	21.340s
	3	*Abortado*	*Swap*	44.982s	58.135s
	5	*Abortado*	*Abortado*	*Abortado*	116.649s
	7	*Abortado*	*Abortado*	*Abortado*	*Abortado*
$E_{(4022\times4670)}$	1	*Abortado*	*Abortado*	*Swap*	100.364s
	3	*Abortado*	*Abortado*	*Abortado*	*Abortado*
	5	*Abortado*	*Abortado*	*Abortado*	*Abortado*
	7	*Abortado*	*Abortado*	*Abortado*	*Abortado*

Observe que quanto maior o volume de dados de entrada, ou seja, quanto maior for a região geográfica e o número de camadas analisadas, maior é a necessidade por poder computacional. Em outras palavras, a medida que se aumenta o tamanho da malha que representa a região analisada, maior é a necessidade por memória RAM e por capacidade de processamento. Para os quadrantes C, D e E, com mais de uma camada, somente as máquinas com mais de 512MB

de memória conseguiram executar o modelo. Em alguns casos, a aplicação foi abortada pelo sistema operacional devido à quantidade de memória necessária para suportar o volume de dados de entrada ser maior do que a quantidade de memória livre na máquina. Em outros casos, foi necessário a utilização do disco (*swap*) para armazenar os dados, o que acabou prejudicando o desempenho da aplicação.

Além disso, embora a máquina `IA64-dual` com processadores de 1.5GHz tenha o dobro da memória RAM da máquina `P4` com processador de 2.8GHz e o compilador Intel utilizado para as máquinas de 64 *bits* apresente uma melhor otimização, os tempos obtidos para todos os quadrantes foram sempre melhores na máquina `P4`. Ou seja, a capacidade de processamento também é muito importante para execução do ICTM.

Enfim, estes resultados comprovam a necessidade de se paralelizar o modelo ICTM, a fim de acelerar o processo de categorização realizado pelo mesmo e de permitir que sua aplicação se torne viável para análises de grandes regiões.

Quanto a escolha dos tempos de referência, optou-se por utilizar os tempos obtidos na máquina `IA64-dual`, pois foi a máquina capaz de processar o maior número de testes, embora a máquina `P4` tenha apresentado resultados um pouco melhores em alguns casos.

6.4.1 Máquinas Agregadas

Conforme descrito na Seção 6.2, três testes foram propostos para validação dos modelos paralelos do ICTM em uma máquina agregada. As seções abaixo apresentam cada um deles.

Paralelização de camadas

Esta seção apresenta os resultados obtidos para a forma de particionamento em camadas do modelo ICTM, de acordo com o modelo paralelo exibido na Figura 25.

A Tabela 15 mostra os tempos de execução paralela, os *speedups* e a eficiência para cada quadrante analisado, de acordo com o número de camadas processadas. Considera-se a regra $np = nc+1$, onde np é o número de processos paralelos e nc é o número de camadas analisadas. Utilizando-se esta regra, um processo (mestre) fica responsável pelo envio dos parâmetros do modelo e das tarefas aos demais processos (escravos) que efetivamente executam o modelo ICTM. Além disso, o valor do raio de vizinhança de células utilizado foi igual a 1.

Quanto aos nodos da Amazônia utilizados neste teste, foram escolhidos conjuntos homogêneos de nodos (`E800` e `IA64-dual`) com configurações bem diferentes para analisar o comportamento da versão implementada.

Observe que aumentando o conjunto de dados de entrada ou o número de camadas em cada quadrante, o tempo de execução seqüencial também aumenta porque, neste caso, as camadas são processadas em uma única máquina, sendo as saídas geradas em um mesmo arquivo texto no diretório corrente.

No entanto, o aumento do número de camadas (3, 5 e 7) não apresenta diferenças significativas nos tempos de execução paralela. Isso aconteceu porque cada camada foi analisada por um processador diferente rodando em um nodo distinto, que se encarregou de executar o modelo e gerar a saída correspondente a sua camada. Este comportamento, demonstrado graficamente

Tabela 15 – Resultados para a forma de particionamento em camadas ($np = nc + 1$ para os tempos paralelos; Ts é o tempo seqüencial de referência; Tp é o tempo de execução paralela; Sp e Ep são os valores de *speedup* e eficiência, respectivamente).

Quadrante	nc	Ts	E800			IA64-dual		
			Tp	Sp	Ep	Tp	Sp	Ep
$A_{(241\times241)}$	3	1.119s	2.085s	0.54	13%	1.711s	0.65	16%
	5	1.827s	2.285s	0.80	13%	2.378s	0.77	13%
	7	2.516s	2.840s	0.86	11%	2.630s	0.96	12%
$B_{(577\times817)}$	3	7.188s	7.119s	1.00	25%	4.013s	1.79	45%
	5	11.979s	7.437s	1.61	27%	4.557s	2.63	44%
	7	16.818s	8.055s	2.09	26%	5.568s	3.02	38%
$C_{(1309\times1765)}$	3	33.652s	28.621s	1.18	29%	13.564s	2.48	62%
	5	56.226s	30.627s	1.84	31%	14.014s	4.01	67%
	7	79.568s	31.005s	2.57	32%	14.433s	5.51	69%

através da análise do quadrante C na Figura 32, comprova a relevância dessa implementação paralela do ICTM.

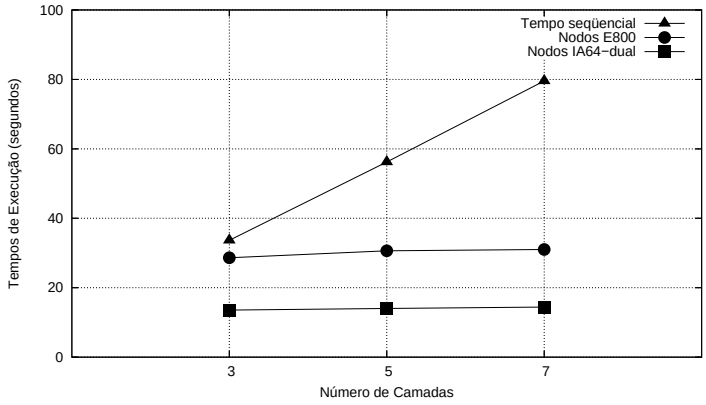

Figura 32 – Comparação entre a versão seqüencial e a versão paralela para o quadrante C.

Além disso, percebe-se que para regiões relativamente pequenas (quadrante A) a versão paralela não é a melhor opção. Portanto, pode-se dizer que esta versão paralela deve ser aplicada quando a região geográfica for grande e quando o número de camadas for maior do que 1, pois nestes casos, o uso da versão seqüencial pode se tornar inviável.

A Figura 33 mostra, de uma outra maneira, o comportamento do ICTM para esta forma de particionamento, considerando a análise do quadrante B com 7 camadas, nodos `E800` e `IA64-dual` e diferentes números de processos paralelos.

Note que a decisão de utilizar os tempos seqüenciais obtidos na máquina `IA64-dual` teve um grande impacto sobre os *speedups* dos nodos `E800` da Amazônia, sendo o ganho de desempenho observado somente com mais de quatro processadores. Não só o melhor desempenho

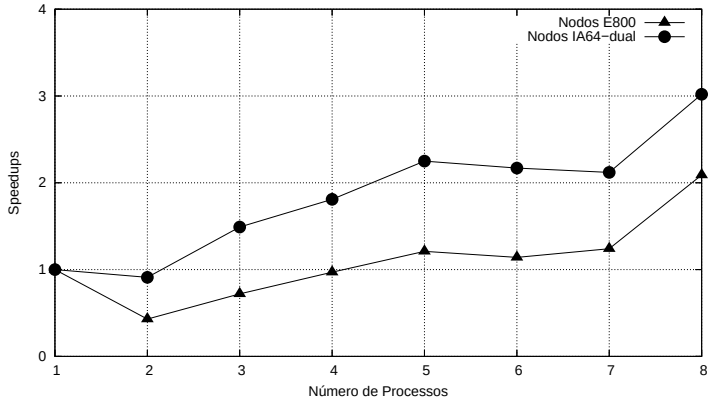

Figura 33 – Comportamento paralelo do ICTM para o quadrante $B_{(577\times817)}$ com 7 camadas.

dos processadores de 64 *bits* e a grande quantidade de memória disponível nestes nodos contribuíram para este resultado, mas também a melhor otimização do compilador Intel usado nas máquinas Itanium.

Entretanto, percebe-se claramente que o tempo de execução paralela tende a diminuir a medida que se aumenta o número de processos paralelos, até alcançar a regra $np = nc + 1$ utilizada para obtenção dos tempos expostos na Tabela 15. Ou seja, o melhor tempo paralelo é obtido quando cada camada é analisada individualmente por um processo diferente rodando em uma máquina distinta, com todo o poder computacional da mesma dedicado exclusivamente para o processo de categorização do ICTM.

De forma geral, pode-se dizer que quanto maior o número de camadas e a dimensão da região geográfica, mais necessário se torna a utilização desta versão paralela.

Paralelização de funções

Esta seção apresenta os resultados obtidos para a forma de particionamento em funções do modelo ICTM, de acordo com o modelo paralelo exibido na Figura 25.

A Tabela 16 mostra os tempos de execução seqüencial, os tempos de execução paralela, os *speedups* e a eficiência para cada quadrante analisado, de acordo com o valor do raio de vizinhança empregado. Considera-se a regra $np = nf + 1$, onde np é o número de processos paralelos e nf é o número total de funções paralelas do modelo que podem ser executadas simultaneamente. Utilizando-se esta regra, um processo (mestre) fica responsável pelo envio dos parâmetros do modelo e das tarefas aos demais processos (escravos) que efetivamente executam o modelo ICTM. Neste caso, como somente quatro funções podem ser executadas ao mesmo tempo no ICTM, o valor de nf será sempre 4 e, conseqüentemente, o valor de np sempre 5.

Quanto aos nodos da Amazônia utilizados neste teste, foram escolhidos os melhores conjuntos homogêneos de nodos (`P4` e `IA64-dual`), ou seja, as máquinas com maior poder computacional. Esta escolha deve-se ao fato de que o modelo paralelo em questão exige que a máquina paralela possua uma boa configuração para que seja possível obter desempenho.

Note que quanto maior for a dimensão da malha que representa a região a ser analisada e

Tabela 16 – Resultados para a forma de particionamento em funções ($np = 5$ para os tempos paralelos; Ts é o tempo seqüencial de acordo com o raio utilizado; Tp é o tempo de execução paralela; Sp e Ep são os valores de *speedup* e eficiência, respectivamente).

Quadrante	**raio**	**P4**				**IA64-dual**			
		Ts	**Tp**	**Sp**	**Ep**	**Ts**	**Tp**	**Sp**	**Ep**
$C_{(1309\times1765)}$	10	11.784s	9.945s	1.18	24%	11.637s	9.645s	1.21	24%
	20	15.721s	13.002s	1.20	24%	12.347s	10.134s	1.22	24%
	30	24.249s	20.278s	1.20	24%	13.069s	10.715s	1.22	24%
	40	27.731s	22.678s	1.22	24%	20.735s	14.967s	1.39	28%
	50	31.952s	23.765s	1.34	27%	34.118s	24.122s	1.42	28%
	60	36.730s	25.456s	1.44	29%	38.816s	27.356s	1.42	28%
	70	41.918s	28.330s	1.48	30%	43.287s	30.506s	1.42	28%
	100	75.210s	39.423s	1.91	38%	79.325s	44.331s	1.79	36%
	200	248.321s	110.098s	2.25	45%	256.567s	115.123s	2.23	45%
$D_{(1739\times2164)}$	10	18.216s	12.110s	1.50	30%	18.118s	12.002s	1.51	30%
	20	24.439s	16.315s	1.50	30%	20.277s	13.221s	1.53	31%
	30	37.578s	25.080s	1.50	30%	22.340s	14.620s	1.53	31%
	40	43.946s	26.648s	1.66	33%	47.858s	29.345s	1.63	33%
	50	51.121s	28.950s	1.77	35%	55.387s	31.210s	1.77	35%
	60	59.166s	30.008s	1.97	39%	62.655s	33.001s	1.90	38%
	70	68.224s	33.987s	2.01	40%	69.668s	35.301s	1.97	39%
	100	155.345s	62.345s	2.51	50%	159.333s	67.441s	2.36	47%
	200	399.178s	132.234s	3.02	60%	405.212s	136.200s	2.98	60%

quanto maior for o valor do raio de vizinhança de células, mais complexo se torna o processo de categorização do ICTM. Isso acontece devido ao método de determinação do estado de uma célula, ou seja, ao invés de se considerar apenas uma célula vizinha nas quatro direções (à direita, esquerda, acima e abaixo) considera-se n células, onde n é o valor do raio adotado. O gráfico exibido na Figura 34 ilustra este comportamento de forma mais clara.

Percebe-se também que as máquinas `P4` obtiveram melhores resultados para raios com valores mais elevados. Já as máquinas `IA64-dual` apresentaram resultados melhores com raios menores. Isso se deve ao fato dos processadores do conjunto de nodos `P4` serem superiores aos processadores dos nodos `IA64-dual`.

A Figura 35 mostra, de uma outra maneira, o comportamento do ICTM para esta forma de particionamento, considerando a análise do quadrante D com o valor do raio igual a 200, nodos `P4` e `IA64-dual` e diferentes números de processos paralelos. Analisando este gráfico verifica-se facilmente que os melhores resultados para esta versão são obtidos quando a regra $np = nf + 1$ é utilizada. Assim, explora-se o paralelismo da melhor forma possível.

É importante ressaltar também, que esta versão deve ser aplicada somente em grandes regiões geográficas que sejam muito complexas (valores de raio grandes). Do contrário, a versão seqüencial pode apresentar resultados melhores.

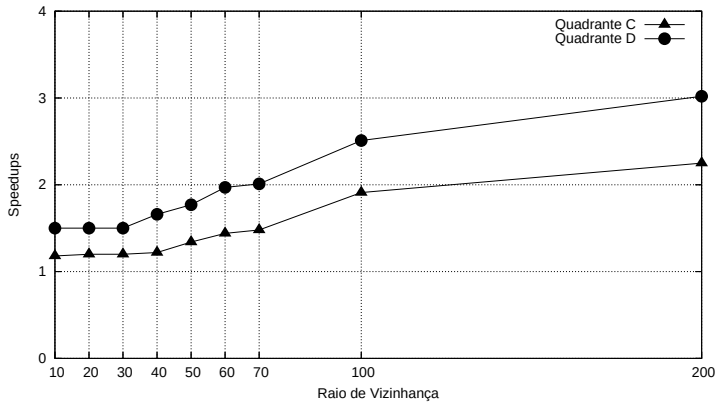

Figura 34 – Comportamento paralelo do ICTM para os quadrantes $C_{(1309\times1765)}$ e $D_{(1739\times2164)}$ com diferentes valores de raio nas máquinas P4.

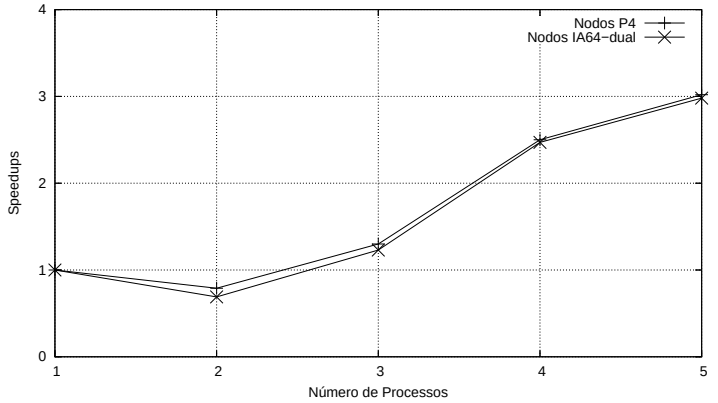

Figura 35 – Comportamento paralelo do ICTM para o quadrante $D_{(1739\times2164)}$ com $raio = 200$.

Paralelização de domínios

Esta seção apresenta os resultados obtidos para a forma de particionamento em domínios do modelo ICTM, de acordo com o modelo paralelo exibido na Figura 26.

A Tabela 17 mostra os tempos de execução paralela e os *speedups* para o quadrante analisado, de acordo com a quantidade de blocos utilizada em cada teste. Considera-se a regra $np = nb + 1$, onde np é o número de processos paralelos e nb é o número total de blocos criados. Utilizando-se esta regra, um processo (mestre) fica responsável pelo envio dos parâmetros do modelo e das tarefas aos demais processos (escravos) que efetivamente executam o modelo ICTM.

Foram usados em conjunto os nodos `E60`, `E800` e `P4` da Amazônia devido à necessidade de um maior número de processos paralelos. Ao todo são 40 processadores divididos em 24 nodos.

Os blocos são constituídos por colunas de células, ou seja, a malha é “cortada” verticalmente

Tabela 17 – Resultados para a forma de particionamento em domínios ($np = nb+1$ para os tempos paralelos; Ts é o tempo seqüencial de referência; Tp é o tempo de execução paralela; Sp é o *speedup* obtido).

Quadrante	**Ts**	**nb**	**Amazônia - 32 *bits***	
			Tp	**Sp**
$E_{(4022\times4670)}$	100.364s	3	105.332s	0.95
		7	70.450s	1.42
		15	50.340s	1.99
		31	41.455s	2.42
		39	33.211s	3.02
		63	69.420s	1.45
		79	115.367s	0.87

de acordo com o número de blocos definidos. Se o total de colunas dividido pelo número de blocos definido não for um número exato, o restante de colunas é dividido uma a uma entre os blocos, a fim de balancear da melhor forma possível a carga de trabalho para cada processo paralelo.

Observe que a quantidade de blocos criados é determinante para o desempenho desta versão paralela do ICTM. Alguns aspectos precisam ser levados em conta tais como quantidade de cálculo por bloco, número de processos em cada nodo (divisão de memória/processador) e volume de comunicação entre os blocos. Com 3 blocos, a quantidade de cálculo necessária para processar cada bloco foi muito grande e o desempenho obtido foi insatisfatório. Já com 79 blocos, a comunicação entre processos e a divisão de um mesmo nodo por 2 processos prejudicaram o desempenho desta versão.

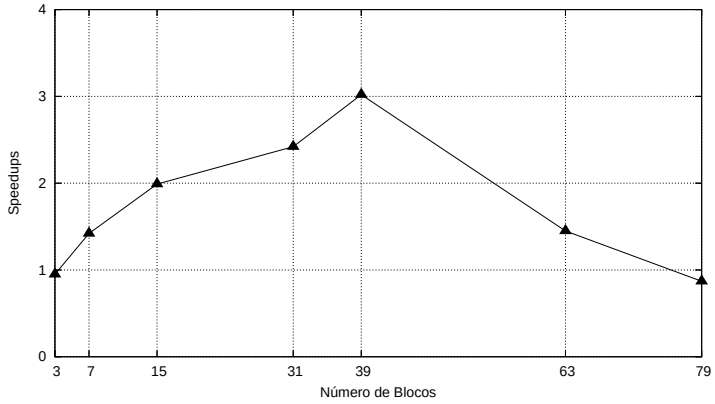

Figura 36 – Comportamento paralelo do ICTM para o quadrante $E_{(4022\times4670)}$ com diferentes quantidades de blocos.

O melhor resultado obtido foi com 39 blocos. Neste caso, os 40 processadores disponíveis foram utilizados, sendo que 39 processos calcularam um único bloco e o processo restante

controlou a divisão de tarefas. A Figura 36 mostra o comportamento geral do ICTM, de acordo com a variação de blocos de células criados.

Quanto a aplicabilidade desta versão, ela deve ser utilizada para análise de regiões extremamente grandes como o quadrante E que está representado em uma malha com mais de 18 milhões de células. Além disso, se o raio de vizinhança for maior do que 1, é preciso que durante a comunicação entre os processos que calculam os blocos uma transferência maior de dados seja realizada, o que pode aumentar muito o tempo de execução da aplicação como um todo.

6.4.2 Grades Computacionais

De acordo com a metodologia definida anteriormente, dois testes foram propostos para a validação do modelo paralelo do ICTM na plataforma de grades computacionais. As seções seguintes abordam estes testes.

Dados geográficos centralizados

Esta seção apresenta os resultados obtidos para a forma de particionamento em camadas do modelo ICTM de acordo com o modelo paralelo exibido na Figura 27, considerando que os dados geográficos estão centralizados na máquina do usuário.

Foram obtidos os tempos de execução na grade OurGrid para todos os quadrantes criados. Para isso, utilizou-se os recursos citados na Seção 6.3 de acordo com a Figura 37.

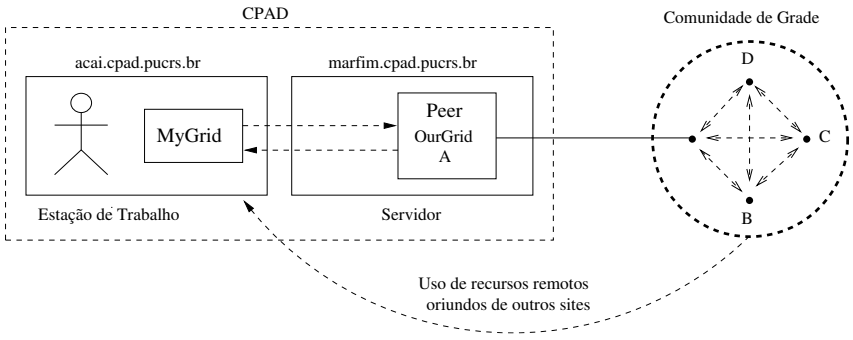

Figura 37 – Uso da grade OurGrid para execução do modelo com dados geográficos centralizados.

Nesta abordagem, os *jobs* são disparados a partir de uma estação de trabalho (máquina `acai.cpad.pucrs.br`) com o MyGrid instalado e configurado. O MyGrid requisita os recursos ao *Peer* local instalado em um servidor do *site* (máquina `marfim.cpad.pucrs.br`) que se encarrega de obter os recursos. Estes recursos são obtidos através da comunicação do *Peer* local com outros *Peers* da comunidade de grade. Mais precisamente, os *jobs* são configurados para não utilizarem os recursos locais, somente os recursos da comunidade. Isso é feito através da definição de um outro atributo para a cláusula *job* (veja Seção 5.2.2), chamado de *requirements*:

$$requirements : (site \,!= cpad).$$

O *jdf* utilizado para execução do modelo ICTM no quadrante A com 3 camadas é exibido a seguir. Note que além da opção de não utilizar os recursos locais, outras duas opções são fornecidas no atributo *requirements*: sistema operacional e quantidade de memória nas máquinas. Além disso, percebe-se claramente que os dados geográficos estão armazenados localmente e serão transferidos através do comando *put*.

```
-----------------------------------------------------------------
job :
label : ICTM-3C-LAGOAPEQUENA_RADAR
requirements: ( site != cpad and os == linux and mem >= 200 )

task :
init  : put ictm32 ictm32
        put 1-lagoapequena_radar.ictm 1-lagoapequena_radar.ictm
remote: ./ictm32  1-lagoapequena_radar.ictm > out3clp-$JOB.$TASK
final : get out3clp-$JOB.$TASK out3clp-$JOB.$TASK

task :
init :  put ictm32 ictm32
        put 1-lagoapequena_radar.ictm 1-lagoapequena_radar.ictm
remote: ./ictm32  1-lagoapequena_radar.ictm > out3clp-$JOB.$TASK
final : get out3clp-$JOB.$TASK out3clp-$JOB.$TASK

task :
init  : put ictm32 ictm32
        put 1-lagoapequena_radar.ictm 1-lagoapequena_radar.ictm
remote: ./ictm32  1-lagoapequena_radar.ictm > out3clp-$JOB.$TASK
final : get out3clp-$JOB.$TASK out3clp-$JOB.$TASK
-----------------------------------------------------------------
```

O objetivo principal com a utilização desta abordagem foi utilizar um ambiente real de grade para validar o modelo, com todas as características envolvidas como a alta dispersão geográfica dos recursos, por exemplo. A Tabela 18 mostra todos os resultados encontrados, considerando o uso dos algoritmos de escalonamento WQR e *Storage Affinity*.

Observe que os tempos obtidos são muito maiores do que os tempos seqüenciais. Isso acontece principalmente devido à transferência dos arquivos de dados de cada camada da máquina local para a máquina remota e devido à transferência dos arquivos com os resultados obtidos da máquina remota para a máquina local.

Além disso, os algoritmos de escalonamento utilizados trabalham com réplicas que são criadas para cada tarefa do *job*, de acordo com a configuração fornecida ao MyGrid (nos testes realizados foram 3 réplicas). Quando uma réplica finaliza seu processamento, as demais precisam ser canceladas. Este procedimento também consome muito tempo, pois é preciso matar

Tabela 18 – Resultados do ICTM na grade OurGrid (dados centralizados).

Quadrante	nc	Ts	Grade OurGrid	
			WQR	S. Affinity
$A_{(241\times241)}$ Arquivo: 540K	1	0.452s	18.789s	16.066s
	3	1.119s	24.512s	21.569s
	5	1.827s	55.319s	49.156s
	7	2.516s	59.441s	55.345s
$B_{(577\times817)}$ Arquivo: 4.8M	1	3.234s	81.921s	72.844s
	3	7.188s	89.666s	82.898s
	5	11.979s	100.701s	86.737s
	7	16.818s	116.201s	91.345s
$C_{(1309\times1765)}$ Arquivo: 21M	1	12.098s	204.400s	192.382s
	3	33.652s	222.556s	210.293s
	5	56.226s	232.875s	216.667s
	7	79.568s	250.213s	225.312s
$D_{(1739\times2164)}$ Arquivo: 38M	1	21.340s	274.231s	264.112s
	3	58.135s	290.389s	278.236s
	5	116.649s	313.222s	284.561s
	7	-	325.667s	296.212s
$E_{(4022\times4670)}$ Arquivo: 178M	1	100.364s	470.476s	456.980s
	3	-	496.557s	465.180s
	5	-	503.112s	477.213s
	7	-	525.987s	503.098s

todas as réplicas criadas para cada uma das tarefas disparadas. As diferenças nos tempos de execução entre os dois algoritmos utilizados são mínimas. O algoritmo *Storage Affinity* levou vantagem em todos os casos de teste, pois ele considera se os dados já estão na máquina na hora de transferir os mesmos.

Enfim, a utilização deste modelo de execução do ICTM fica restrita a uma condição, qual seja, a impossibilidade de utilizar a versão seqüencial em máquinas convencionais ou as versões paralelas para máquinas agregadas devido ao tamanho da região a ser analisada.

Seria interessante validar este modelo em um ambiente de grade real cujo seus membros fossem interligados através de uma rede de interconexão dedicada de alta velocidade. Só assim seria possível a obtenção de desempenho.

Dados geográficos distribuídos

Esta seção apresenta os resultados obtidos para a forma de particionamento em camadas do modelo ICTM de acordo com o modelo paralelo exibido na Figura 27, considerando que os dados geográficos estão distribuídos pela grade, ou seja, estão armazenados nas máquinas onde serão processados.

Foram obtidos os tempos de execução na grade OurGrid para os quadrantes C e D. Para

isso utilizou-se os recursos citados na Seção 6.3 de acordo com a Figura 38.

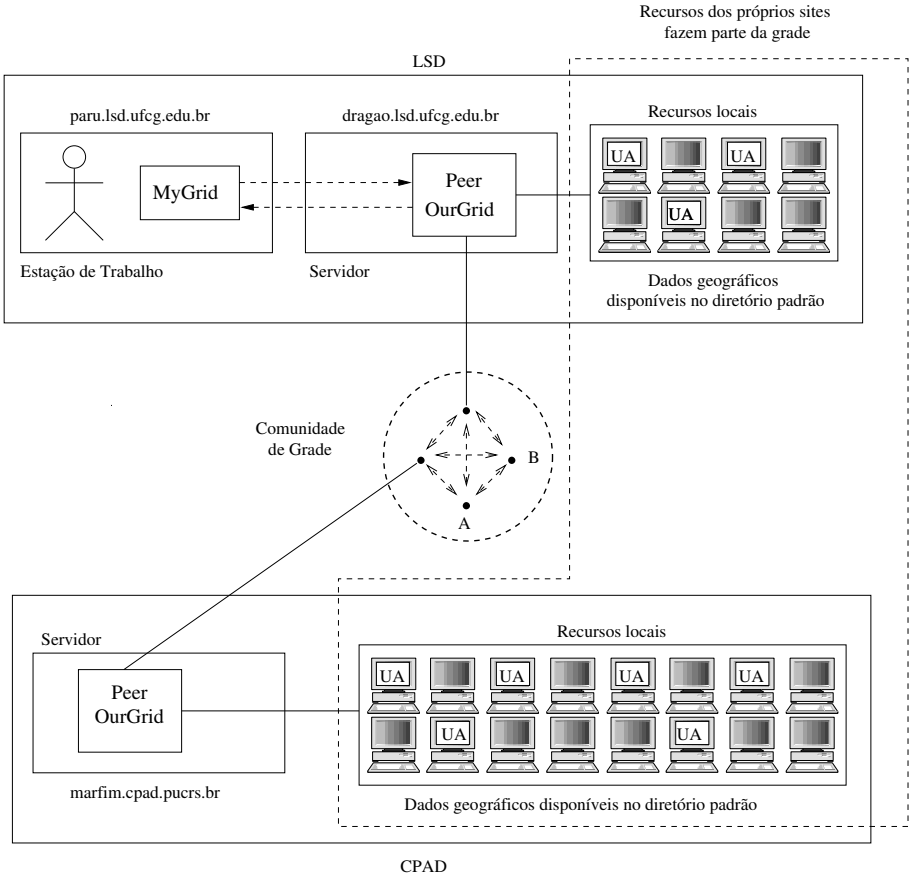

Figura 38 – Uso da grade OurGrid para execução do modelo com dados geográficos distribuídos.

Da mesma forma que a abordagem exibida na seção anterior (dados centralizados), os *jobs* são disparados a partir de uma estação de trabalho com o MyGrid instalado e configurado (neste caso, máquina `paru.lsd.ufcg.edu.br`). O MyGrid requisita os recursos ao *Peer* local instalado em um servidor do *site* (neste caso, máquina `dragão.lsd.ufcg.edu.br`) que se encarrega de obter os recursos. Estes recursos são obtidos localmente e através da comunicação do *Peer* local com outros *Peers* da comunidade de grade.

Nesta abordagem, os *jobs* são configurados para utilizarem os recursos locais e os recursos da comunidade. Isso é feito através da definição do atributo *requirements*. No entanto, para que seja possível utilizar dois sites pré-definidos ao mesmo tempo, é necessário que o ICTM seja adaptado em mais de um *job*. Por exemplo, considerando o mesmo exemplo da seção anterior, ou seja a análise do quadrante A com 3 camadas, sendo que dois conjuntos de dados referentes a duas primeiras camadas estão armazenados localmente na máquina `paru.lsd.ufcg.edu.br` e o outro conjunto referente a terceira camada está armazenado remotamente na máquina `amazonia01.cpad.pucrs.br`, tem-se os seguintes *jobs*:

```
------------------------------------------------------------------
   JOB 1
------------------------------------------------------------------
job :
label : ICTM-3C-LAGOAPEQUENA_RADAR-1
requirements: ( site == lsd and os == linux and mem >= 200 )

task :
init  : put ictm32 ictm32
remote: ./ictm32
        /home/user/1-lagoapequena_radar.ictm > out3clp-$JOB.$TASK
final : get out3clp-$JOB.$TASK out3clp-$JOB.$TASK

task :
init :  put ictm32 ictm32
        remote: ./ictm32
        /home/user/1-lagoapequena_radar.ictm > out3clp-$JOB.$TASK
final : get out3clp-$JOB.$TASK out3clp-$JOB.$TASK
------------------------------------------------------------------

------------------------------------------------------------------
   JOB 2
------------------------------------------------------------------
job :
label : ICTM-3C-LAGOAPEQUENA_RADAR-2
requirements: ( site == cpad and os == linux and mem >= 200 )

task :
init  : put ictm32 ictm32
        remote: ./ictm32
        /home/user/1-lagoapequena_radar.ictm > out3clp-$JOB.$TASK
final : get out3clp-$JOB.$TASK out3clp-$JOB.$TASK
------------------------------------------------------------------
```

O *job* 1 é responsável pela execução das duas primeiras camadas do modelo que estão armazenadas localmente no LSD. Já o *job* 2 é responsável pela execução da terceira camada que está armazenada localmente no CPAD. Para que isso seja possível, basta que o usuário conheça a real localização do dado geográfico na máquina remota (*hostname* e *path*), além do sistema de arquivos ser compartilhado. No caso da existência de mais camadas distribuídas em diferentes *sites*, basta criar novos *jobs* para cada localização diferente.

A Tabela 19 mostra o teste realizado para esta abordagem, considerando os quadrantes C e D com 3, 5 e 7 camadas e diferentes formas de distribuição das camadas entre os *sites*, ou seja, em cada teste uma determinada quantidade de propriedades (arquivos com os dados geográficos de entrada) está armazenada localmente e o restante remotamente.

Tabela 19 – Resultados do ICTM na grade OurGrid, considerando o uso de dados distribuídos (nc é o número de camadas e Ts é o tempo seqüencial de referência).

Quadrante	nc	Ts	Distribuição		Grade OurGrid
			LSD	CPAD	
$C_{(1309\times1765)}$	3	33.652s	2	1	21.915s
	5	56.226s	3	2	23.011s
	7	79.568s	4	3	24.110s
$D_{(1739\times2164)}$	3	58.135s	1	2	47.760s
	5	116.649s	2	3	50.234s
	7	-	3	4	53.314s

Como esperado, o tempo seqüencial aumenta a medida que se aumenta o número de camadas da região a ser analisada, pois as mesmas são processadas em seqüência em uma mesma máquina. Com o uso do ambiente de grade isso não ocorre, ou seja, não existe muita diferença nos tempos obtidos na grade, mesmo com o aumento do número de camadas (quantidade de trabalho). Este comportamento do ICTM multi-camada na grade é semelhante ao comportamento obtido com a versão para máquinas agregadas desta mesma forma de particionamento (camadas). O gráfico da Figura 39 ilustra este comportamento com base no quadrante C, considerando os tempos da versão paralela para máquinas agregadas obtidos nos nodos `IA64-dual`.

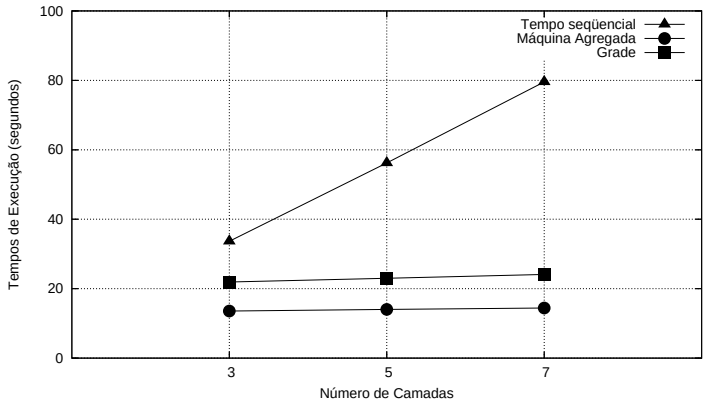

Figura 39 – Comparação entre a versão seqüencial, a versão paralela para máquinas agregadas e a versão para grades.

Observe que a diferença entre os tempos obtidos na grade OurGrid e os tempos obtidos na máquina agregada Amazônia está relacionada à transferência dos arquivos com os resultados da máquina remota para a máquina local. Já a diferença entre os tempos obtidos na grade e os tempos seqüenciais dizem respeito a não transferência dos arquivos de entrada para cada camada da máquina de onde foi disparado os *jobs* para as máquinas que processaram os mesmos.

Sendo assim, o uso do ambiente de grade nessas condições pode gerar ganho de desempenho para o processamento do modelo ICTM em grandes regiões geográficas com diferentes

propriedades. Além disso, o uso de informações distribuídas para o processamento de dados geográficos é uma demanda bastante forte das áreas de geoprocessamento, geomática e etc. A tecnologia existente por trás dos mecanismos/aparelhos que geram este tipo de informação faz com que a obtenção dos dados se torne bastante difícil, devido principalmente ao alto custo dos mesmos.

6.5 Considerações sobre o Modelo HPC-ICTM

De acordo com o que foi exposto nas seções anteriores, percebe-se que o ICTM pode ser utilizado para análises de regiões com uma ou várias propriedades geográficas, ou seja, pode-se analisar em uma mesma área a topografia, a vegetação e etc. Neste tipo de análise os dados de entrada são pontuais. Em outras palavras, a malha que representa a região geográfica possui valores absolutos referentes a cada sub-região compreendida em uma célula, gerando uma matriz absoluta de valores.

No entanto, existe outra forma de analisar regiões geográficas no ICTM. Trata-se da análise de imagens intervalares[4]. Para cada pixel da imagem há uma dúvida em decidir qual é o seu valor, pois ele pode variar entre um intervalo. Neste caso, os dados originais são intervalares, diferente do caso anterior, onde os originais são pontuais. Então, a matriz absoluta é de intervalos, um para cada célula. Neste caso, fatia-se cada intervalo, em um número arbitrário de fatias, sendo cada fatia uma camada no ICTM. Agora sim, a fatia é novamente um dado pontual.

Para este tipo de análise, onde um número bastante grande de camadas é gerado, a forma de particionamento em camadas adaptada para ambas plataformas utilizadas no trabalho pode ser bastante útil, dependendo da dimensão da imagem intervalar.

No que se refere a aplicabilidade de cada versão paralela implementada pode-se dizer que:

- O particionamento em camadas para máquinas agregadas deve ser utilizado na análise de grandes regiões geográficas que apresentam diferentes propriedades. Quanto maior for o número de camadas, maior será o ganho obtido com a versão paralela.

- O particionamento em funções para máquinas agregadas deve ser utilizado na análise de grandes regiões geográficas que apresentam um nível de complexidade grande, seja pelo tamanho do raio de vizinhança das células ou pelos próprios valores absolutos de entrada.

- O particionamento em domínios para máquinas agregadas deve ser utilizado para regiões representadas em malhas com suficientemente muitos pontos. Quanto maior for a dimensão da malha e até mesmo a complexidade dos valores de entrada, melhor será o ganho obtido com esta versão.

- O particionamento em camadas para grades computacionais deve ser utilizado quando as propriedades estão armazenadas localmente nos nodos que farão seu processamento, ou seja, aconselha-se o uso de grades principalmente quando os dados forem distribuídos, pois a transferência de arquivos muito grandes pela internet torna o processamento do

[4] Análise não inserida no trabalho devido à falta de imagens intervalares reais para a realização dos testes.

modelo muito lento. Por outro lado, se a rede que interliga os recursos da grade for dedicada e de alta desempenho, pode ser que a transferência de arquivos não tenha tanto impacto no tempo de execução da aplicação.

Enfim, é preciso verificar se as características da região geográfica se enquadram em cada uma das versões paralelas do HPC-ICTM. Caso contrário, a versão seqüencial pode ser a melhor opção.

7 Conclusão e Trabalhos Futuros

Este documento apresentou uma descrição detalhada do desenvolvimento do modelo HPC-ICTM. Este modelo permite a realização de categorizações em grandes regiões geográficas e a utilização de informações geograficamente distribuídas. Mais precisamente, é possível analisar grandes espaços de natureza geométrica e produzir categorizações confiáveis sobre o conjunto de pontos deste espaço, de acordo com as características dos mesmos.

Foram utilizadas duas plataformas de execução de aplicações paralelas, as máquinas agregadas e as grades computacionais. As máquinas agregadas já são bastante conhecidas e difundidas e, por isso, os resultados obtidos para esta plataforma são, de certa forma, previsíveis. Já as grades computacionais apresentam muitas características inovadoras e muitos desafios que impedem sua maior proliferação. Este trabalho contribuiu neste sentido, apresentando os resultados da aplicação do modelo ICTM em um ambiente de grade.

A definição dos modelos paralelos do ICTM foi eficaz, permitindo que análises de grandes regiões possam ser realizadas. No entanto, algumas particularidades quanto a aplicação de cada modelo paralelo foi mencionada, sendo necessário respeitá-las para que o modelo HPC-ICTM possa ser executado com sucesso.

Durante o desenvolvimento do trabalho, cinco artigos foram produzidos e encaminhados para conferências, periódicos e *journals* de grande relevância nas áreas relacionadas. Destes cinco trabalhos, quatro já foram aceitos e alguns já foram publicados. O Apêndice B apresenta a relação completa destes artigos, destacando os eventos para onde foram enviados os trabalhos, bem como a classificação e a edição de cada publicação.

No que diz respeito a continuidade do trabalho, conforme comentado no Capítulo 1, o ICTM continua sendo desenvolvido pelo GMFC da UCPel. Para o próximo ano, todo o trabalho apresentado neste documento terá prosseguimento. Entre as principais atividades pode-se destacar:

- Adaptação do ICTM para trabalhar com o formalismo de autômatos celulares;
- Realização de simulações de aspectos dinâmicos de uma dado espaço geográfico, a partir da utilização do formalismo de autômatos celulares;
- Implementação do mecanismo de extração de fatos no modelo;
- Adaptação do HPC-ICTM para trabalhar com autômatos celulares;
- Utilização do protótipo do HPC-ICTM como base para criação do ambiente GIPE (*Geographic Information Processing Environment*) para integrar todas as funcionalidades do modelo ICTM.
- Criação do ambiente HPC-GIPE, com todas as funcionalidades do ICTM e as funcionalidades do HPC-ICTM descritas neste documento;

- Geração de um banco de dados geográfico a partir dos resultados obtidos com os modelos implementados;
- Implementação de alguma forma de particionamento identificada neste trabalho que produza comunicação entre as tarefas, em um ambiente de grade.

Além dessas linhas de trabalho podem surgir muitas outras, pois o ICTM aborda várias áreas da Ciência da Computação que a cada passo de desenvolvimento do modelo podem motivar novos rumos e caminhos a seguir.

Quanto a realização deste trabalho, acredita-se que os resultados alcançados foram satisfatórios. Além disso, a cooperação entre dois grupos de pesquisa diferentes de Universidades distintas foi bastante interessante, pois permitiu uma interação com profissionais de diferentes áreas e com idéias e tópicos de interesse diferenciados.

Referências

[1] COBLENTZ, D. et al. Towards Reliable Sub-Division of Geological Areas: Interval Approach. In: REZNIK, L.; KREINOVICH, V. (Ed.). *Soft Computing in Measurements and Information Acquisition*. Berlin-Heidelberg: Springer-Verlag, 2003. p. 223–233.

[2] COBLENTZ, D. et al. Towards Reliable Sub-Division of Geological Areas: Interval Approach. In: *Proc. of the 19th International Meeting of the North American Fuzzy Information Processing Society (NAFIPS'2000)*. Atlanta: [s.n.], 2000. p. 368–372.

[3] AGUIAR, M. S.; COSTA, A. C. R. Autômatos Celulares para Análise da Monotonicidade da Declividade de Áreas Geológicas. In: *III Workshop Brasileiro de GeoInformática*. Rio de Janeiro: Porto Alegre – Sociedade Brasileira de Computação, 2001. p. 87–94.

[4] AGUIAR, M. S.; COSTA, A. C. R. *Um Modelo Categorizador Intervalar n-Dimensional com l-Camadas Baseado em Tesselações*. Tese (Doutorado) — PPGCC/UFRGS, Porto Alegre, 2004.

[5] DIMURO, G. P. et al. *Projeto ACI: Autômatos Celulares Intervalares com Aplicações em Topografia*. Pelotas: ESIN/UCPEL. Disponível em: <http://gmc.ucpel.tche.br/aci>. Acesso em: 03 mar 2005.

[6] DIMURO, G. P. et al. *Projeto FMC2: Fundamentos Matemáticos da Computação: Modelos e Aplicações de Computações Intervalares*. Pelotas: ESIN/UCPEL. Disponível em: <http://gmc.ucpel.tche.br/fmc2>. Acesso em: 03 mar 2005.

[7] SILVA, R. K. S. et al. Topo-ICTM: Uma Ferramenta para Categorização Topográfica Baseada no Modelo de Tesselação Intervalar Bidimensional. In: *XIX Conferência Latinoamericana de Informática (CLEI'2003)*. La Paz: Universidad Mayor de San Andrés, 2003. p. 1–10.

[8] COSTA, F. A. *Introdução ao Sensoriamento Remoto*. Pelotas: ESIN/UCPEL, Palestras do Projeto FMC. Disponível em <http://gmc.ucpel.tche.br/fmc2/papers.htm>. Acesso em: 08 abr 2005.

[9] BURROUGH, P. *Principles of Geography Information Systems for Land Resources Assessment*. Oxford: Clarendon Press, 1989.

[10] RODRIGUES, M. Geoprocessamento. In: *V Encontro Nacional de Engenheiros Cartógrafos*. UNESP: Presidente Prudente, 1988. p. 144–160.

[11] RODRIGUES, M. Introdução ao Geoprocessamento. In: *I Simpósio Brasileiro de Geoprocessamento*. São Paulo: EDUSP, 1990. p. 1–26.

[12] KORTE, G. *The GIS book*. Santa Fé: On World Press, 1994.

[13] SILVA, J.; SOUZA, M. *Análise Ambiental*. Rio de Janeiro: Editora da UFRJ, 1987.

[14] RODRIGUES, M.; QUINTANILHA, J. A. A seleção de software SIG para gestão urbana. In: *XV Congresso Brasileiro de Cartografia*. São Paulo: SBC, 1991. p. 513–519.

[15] TEIXEIRA, A. L.; MORETTI, E.; CHRISTOFOLETTI, A. *Introdução aos sistemas de informação geográfica*. Rio Claro: Ed. do Autor, 1992.

[16] CAMARA, G. Anatomia de sistemas de informações geográficas: visão atual e perpectivas de evolução. In: ASSAD, E.; SANO, E. (Ed.). *Sistema de informações geográficas: aplicações na agricultura*. Brasília: EMBRAPA, 1993.

[17] MENEGUETTE, A. *Introdução ao Geoprocessamento*. Presidente Prudente: UNESP. Disponivel em: <http://www2.prudente.unesp.br/arlete/hp_arlete/courseware/intgeo.htm>. Acesso em: 09 abr 2005.

[18] GAGNON, P.; BÉDARD, Y. From Surveying to Geomatics – Evaluation of education needs to adapt to a new paradigm (A Canadian Perspective). *Geomatica*, v. 50, n. 3, p. 269–286, 1996.

[19] SILVA, J. X. D. *Geoprocessamento para Análise Ambiental*. Rio de Janeiro: Sermograf, 2001. 228 p.

[20] BURROUGH, P. A.; MCDONNELL, R. A. *Principles of Geographical Information Systems*. Oxford, UK: Oxford University Press, 1998.

[21] MACKEY, B. The Role of GIS and Environmental Modeling in the Conservation of Biodiversity. In: *Proc. of the 3th International Conference on Integrating GIS and Environmental Modelling*. Santa Fe: [s.n.], 1996.

[22] GRAYSON, R. B.; BLOSCHL, G.; MOORE, I. D. Distributed Parameter Hydrologic Modeling Using Vector Ulevation Data: THALES and TAPES-C. In: SINGH, V. P. (Ed.). *Computer Models of Watershed Hydrology*. Boca Raton, FL: CRC, 1994. p. 669–696.

[23] KIRKBY, M. J. et al. Scaling up processes and models. *Journal of Soil and Water Conservation*, v. 51, n. 5, p. 391–396, 1996.

[24] PHILLIPS, J. D. Sediment storage, sediment yield, and time scales in sediment denudation studies. *Geographical Analysis*, v. 18, p. 161–167, 1986.

[25] SCHAFFER, W. M. Ecological abstraction: the consequences of reduced dimensionality in ecological models. *Ecological Monographs*, v. 5, p. 383–401, 1981.

[26] MOORE, I. D.; HUTCHINSON, M. F. Spatial extension of hydrologic process modelling. In: *Proc. of the International Hydrology and Water Resources Symposium*. Canberra: Institute of Australian Engineers, 1991. p. 803–808.

[27] WILSON, J. P. GIS-based land surface/subsurface models: new potential for new models. In: *Proc. of the 3th International Conference on Integrating GIS and Environmental Modelling*. Santa Fe: [s.n.], 1996.

[28] WILSON, J. P.; BURROUGH, P. A. Dynamic modelling, geostatics and fuzzy classification: new sneakers for a new geography? *Association of American Geographers*, v. 89, p. 736–746, 1999.

[29] BAKER, M.; BUYYA, R. Cluster Computing: The Commodity Supercomputer. *Software: Practice and Experience*, v. 29, n. 6, p. 551–576, 1999.

[30] DE ROSE, C. A. F.; NAVAUX, P. O. A. *Arquiteturas Paralelas*. Porto Alegre: Instituto de Informática da UFRGS, Editora Sagra Luzzatto, 2003. (Série Livros Didáticos, 15).

[31] DE ROSE, C. A. F.; NAVAUX, P. O. A. Fundamentos de Processamento de Alto Desempenho. In: *Anais da 4ª Escola Regional de Alto Desempenho (ERAD'2004)*. Pelotas: [s.n.], 2004. p. 41–66.

[32] BUYYA, R. *High Performance Cluster Computing: Architectures and Systems*. Upper Saddle River, New Jersey: Prentice Hall PTR, 1999. v. 1, 849 p.

[33] BODEN, N. J. et al. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, v. 15, n. 1, p. 29–36, 1995.

[34] IBEL, M. et al. High-Performance Cluster Computing Using Scalable Coherent Interface. In: *Proceedings of 7th International Workshop on SCI-based High-Performance Low-Cost Computing*. [S.l.: s.n.], 1997. p. 45–54.

[35] WALKER, D. W. The Design of a Standard Message Passing Interface for Distributed Concurrent Computers. *Parallel Computing*, v. 20, n. 4, p. 657–673, 1994.

[36] GROPP, W.; LUSK, E.; SKJELLUM, A. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. Cambridge, Mass.: MIT Press, 1999. 371 p.

[37] GEIST, A. et al. *PVM Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*. Cambridge, Mass.: MIT Press, 1994. 176 p.

[38] STERLING, T. An Introduction to PC Clusters for High Performance Computing. *The International Journal of High Performance Computing Applications*, v. 15, n. 2, p. 92–101, 2001.

[39] BUYYA, R. *High Performance Cluster Computing: Programming and Applications*. Upper Saddle River, New Jersey: Prentice Hall PTR, 1999. v. 2, 664 p.

[40] WILKINSON, B.; ALLEN, M. *Parallel Programing: techniques and applications using networked workstations and parallel computers*. Upper Saddle River, New Jersey: Prentice-Hall, 1999. 431 p.

[41] BAKER, M.; FOX, G.; YAU, H. *Cluster Computing Review*. [S.l.], 1995.

[42] BAYUCAN, A. et al. Portable Batch System Administration Guide. *Veridian System*, 2000.

[43] ZHOU, S. et al. Utopia: a Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems. *Software: Practice and Experience*, v. 23, n. 12, p. 1305–1336, 1993.

[44] NETTO, M. A. S.; DE ROSE, C. A. F. CRONO: A Configurable and Easy to Maintain Resource Manager Optimized for Small and Mid-Size GNU/Linux Cluster. In: *Proceedings of the International Conference on Parallel Processing*. Kaohsiung: IEEE Computer Society Press, 2003. p. 555–562.

[45] FOSTER, I.; KESSELMAN, C.; TUECKE, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *The International Journal of Supercomputer Applications*, v. 15, n. 3, p. 200–222, 2001.

[46] CIRNE, W. Grids Computacionais: Arquitetura, Tecnologias e Aplicações. In: *Anais da 3ª Escola Regional de Alto Desempenho (ERAD'2003)*. Santa Maria: [s.n.], 2003. p. 103–134.

[47] NEMETH, Z.; SUNDERAM, V. Characterizing Grids: Attributes, Definitions and Formalisms. *Journal of Grid Computing*, v. 1, p. 9–23, 2003.

[48] CHETTY, M.; BUYYA, R. Weaving Computational Grids: How Analogous are They with Electrical Grids? *Computing in Science and Engineering*, v. 4, n. 4, p. 61–71, 2002.

[49] REINEFELD, A.; SCHINTKE, F. Concepts and Technologies for a Worldwide Grid Infrastructure. In: *Euro-Par Parallel Processing*. Paderborn: Springer, 2002. p. 62–71.

[50] CATLETT, C. Standards for Grid Computing: Global Grid Forum. *Journal of Grid Computing*, v. 1, p. 3–7, 2003.

[51] FOSTER, I.; KESSELMAN, C. *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco: Morgan-Kaufman Publishers, 1999. 677 p.

[52] BAKER, M.; BUYYA, R.; LAFORENZA, D. Grids and Grid Technologies for Wide-area Distributed Computing. *Software: Practice and Experience*, v. 32, n. 15, p. 1437–1466, 2002.

[53] SHIRTS, M.; PANDE, V. Screen Savers of the World, Unite! *Science*, v. 290, n. 8, p. 1903–1904, 2000.

[54] MILOJICIC, D. S. et al. Peer-to-Peer Computing. *Technical Report HPL-2002-57, HP Lab*, 2002.

[55] FOSTER, I.; IAMNITCHI, A. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In: KAASHOEK, F. M.; STOICA, I. (Ed.). *International Workshop on Peer-to-Peer Systems (IPTPS'2003)*. Berkeley: Springer, 2003. p. 118–128.

[56] ANDERSON, D. J. C.; KORPELA, E. SETI@home: An Experiment in Public-Resource Computing. *Communication of the ACM*, v. 45, n. 11, p. 56–61, 2002.

[57] ANDERSON, D. P. et al. SETI@home: Internet Distributed Computing for SETI. *Bioastronomy 99: A New Era in Bioastronomy, G. Lemarchand and K. Meech, eds., ASP Conference Series No. 213 (Astronomical Society of the Pacific: San Francisco)*, p. 511–518, 2000.

[58] CASANOVA, H.; DONGARRA, J. NetSolve: A Network Server for Solving Computational Science Problems. *International Journal of Supercomputing Applications and High Performance Computing*, v. 11, n. 3, 1997.

[59] ARNOLD, D. et al. *User's Guide to NetSolve V1.4.1*. Knoxville, TN, 2002.

[60] AVERY, P. Data Grids: a new computational infrastructure for data-intensive science. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, v. 360, n. 15, p. 1191–1209, 2003.

[61] ROUSSOS, M. et al. NICE: Combining Constructionism, Narrative and Collaboration in a Virtual Learning Environment. *SIGGRAPH Comput. Graph.*, ACM Press, v. 31, n. 3, p. 62–63, 1997.

[62] CASANOVA, H. et al. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In: *Proceedings of the 9th Heterogeneous Computing Workshop*. Cancun: IEEE Computer Society Press, 2000. p. 349–363.

[63] CIRNE, W. Running Bag-of-Tasks Applications on Computational Grids: The Mygrid Approach. In: *Proceedings of the 2003 International Conference on Parallel Processing*. Kaohsiung: IEEE Computer Society Press, 2003. p. 407–416.

[64] COOPER, K. et al. New Grid Scheduling and Rescheduling Methods in the GrADS Project. In: *Proceedings of the 18th International Parallel and Distributed Processing Symposium*. Santa Fe: IEEE Computer Society Press, 2004. p. 199–206.

[65] ABRAMSON, D.; GIDDY, J.; KOTLER, L. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? In: *International Parallel & Distributed Processing Symposium (IPDPS'2000)*. Cancun: IEEE CS Press, 2000. p. 520–528.

[66] STILES, J. R. et al. Monte Carlo simulation of neuromuscular transmitter release using MCell, a general simulator of cellular physiological processes. *Computational Neuroscience*, p. 279–284, 1998.

[67] SMALLEN, S.; CASANOVA, H.; BERMAN, F. Applying scheduling and tuning to online parallel tomography. In: *Proceedings of the Conference on Supercomputing*. Denver: ACM/IEEE, 2001. p. 12–12.

[68] SMALLEN, S. et al. Combining Workstations and Supercomputers to Suport Grid Applications: The Parallel Tomography Experience. In: *Proceedings of the 9th Heterogeneous Computing Workshop (HCW'2000)*. Cancun: [s.n.], 2000. p. 241–252.

[69] WOLFRAM, S. *Cellular automata and complexity: collected papers*. Reading: Addison-Wesley, 1994.

[70] MOORE, R. E. *Methods and Applications of Interval Analysis*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1979. 201 p.

[71] OLIVEIRA, P. W.; DIVERIO, T. A.; CLAUDIO, D. M. *Fundamentos da Matemática Intervalar*. Porto Alegre: Instituto de Informática da UFRGS: SAGRA-Luzzato, 1997.

[72] ANDRADE, N. et al. OurGrid: An approach to easily assemble grids with equitable resource sharing. In: *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*. Seattle: Springer Verlag, 2003. p. 61–86.

[73] Message Passing Interface Forum MPIF. *MPI-2: Extensions to the Message-Passing Interface*. Knoxville, 1996.

[74] GROPP, W. et al. High-performance, portable implementation of the MPI Message Passing Interface Standard. *Parallel Computing*, v. 22, n. 6, p. 789–828, 1996.

[75] CIRNE, W. et al. Scheduling in Bag-of-Task Grids: The PAUÁ Case. In: *Proceedings of the 16th SBAC-PAD'04 - Symposium on Computer Architecture and High Performance Computing 2004*. Foz do Iguaçu: IEEE, 2004. p. 124–131.

[76] SWEET, M.; WRIGHT, R. S. *OpenGL SuperBible*. Indianapolis, Indiana, USA: Waite Group Press, 2000. 696 p.

Apêndice A – Exemplos de categorizações

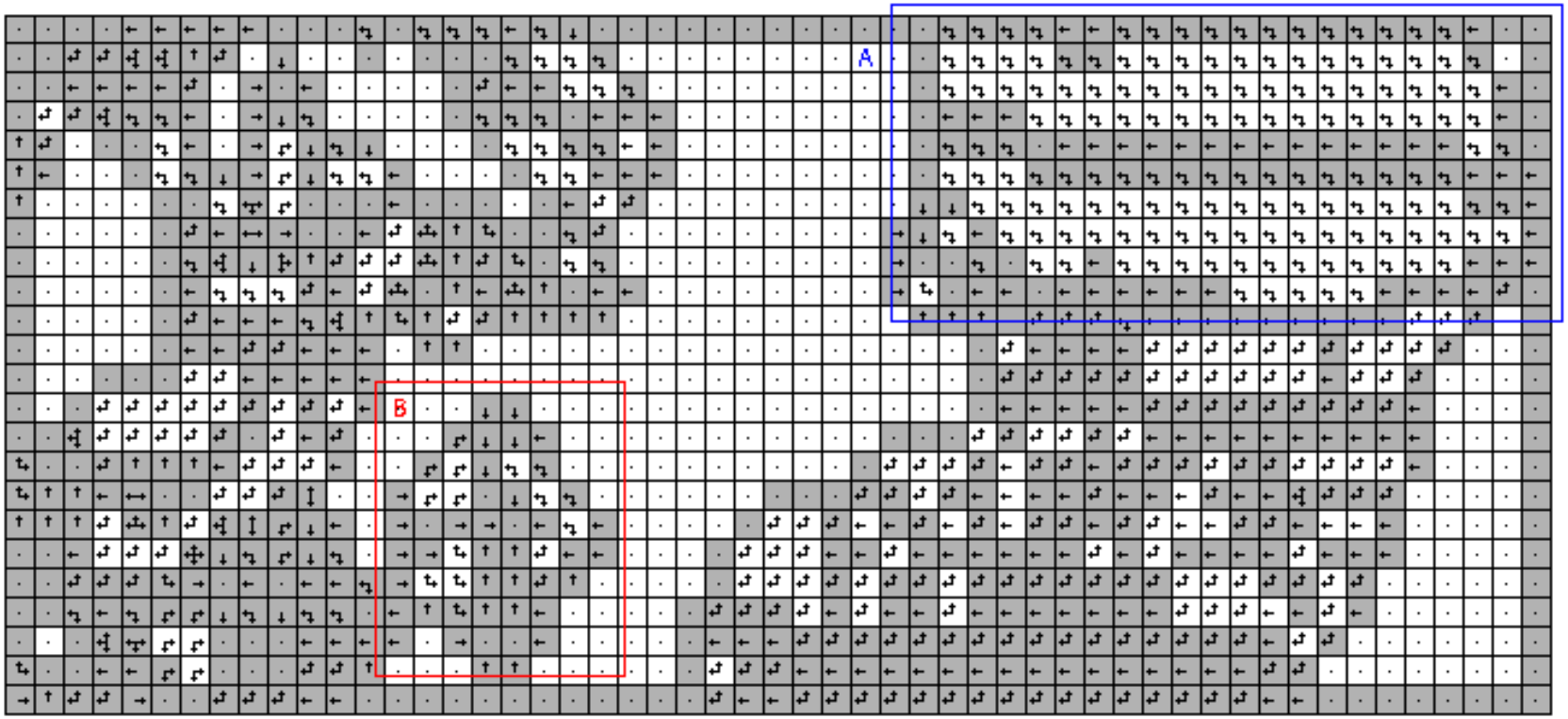

Figura 40 – DEM1000m-raio 1.

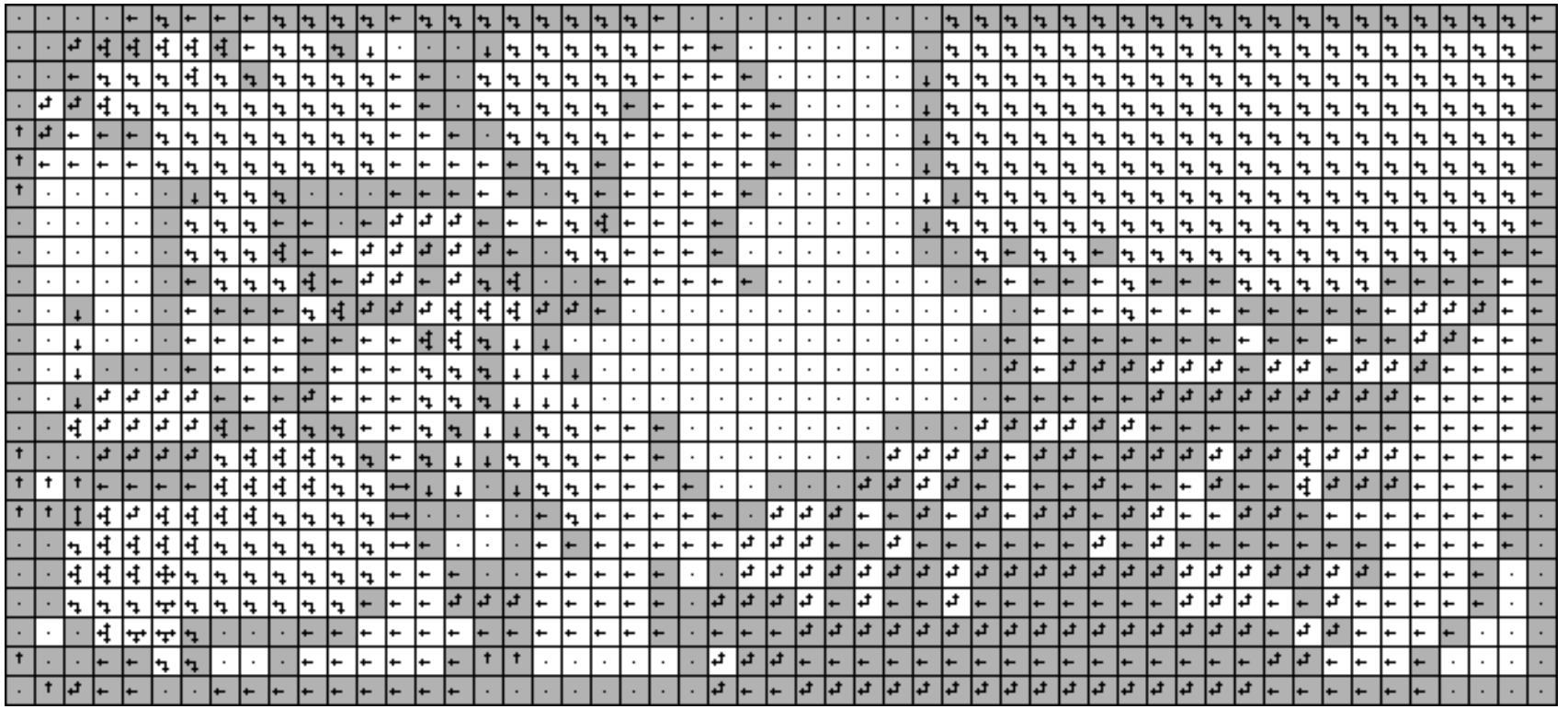

Figura 41 – DEM1000m-raio 5.

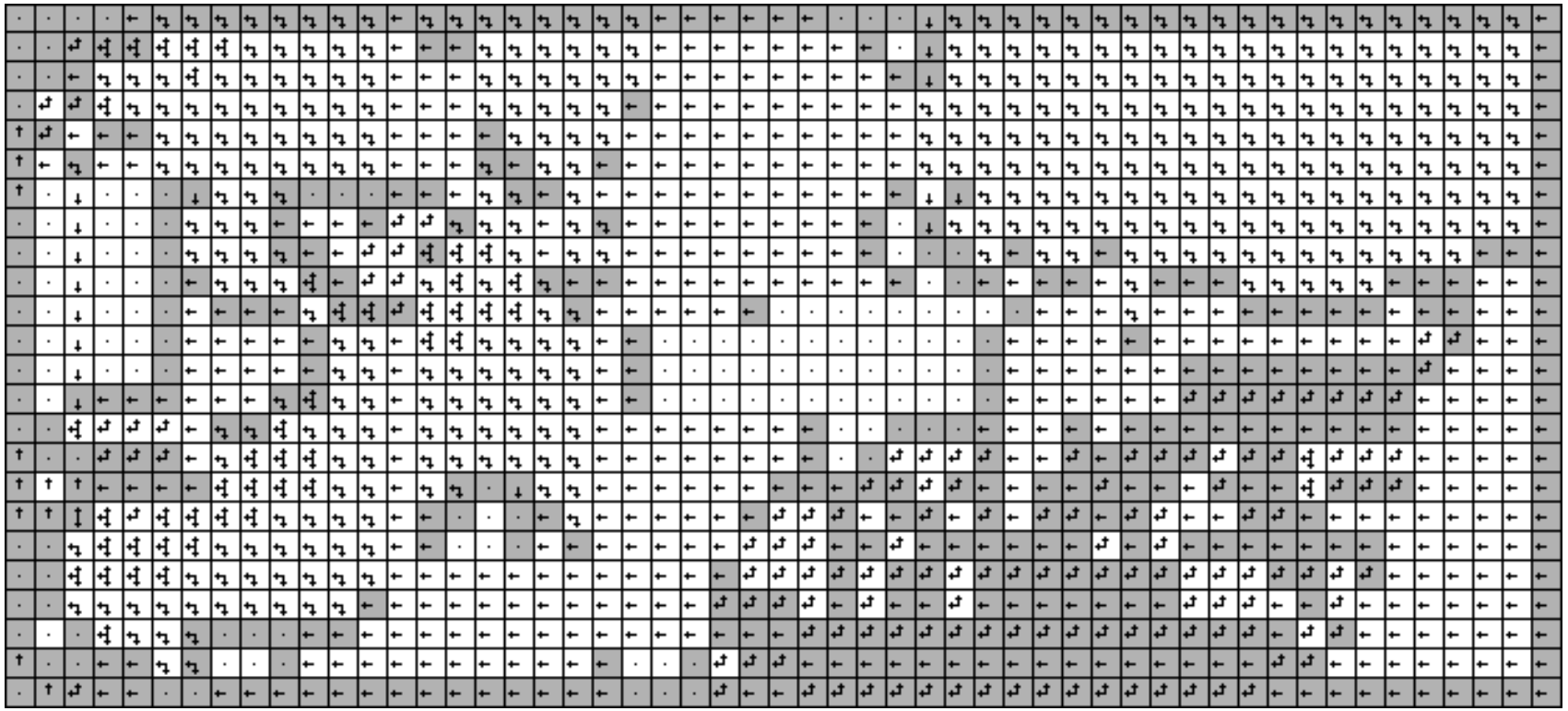

Figura 42 – DEM1000m-raio 10

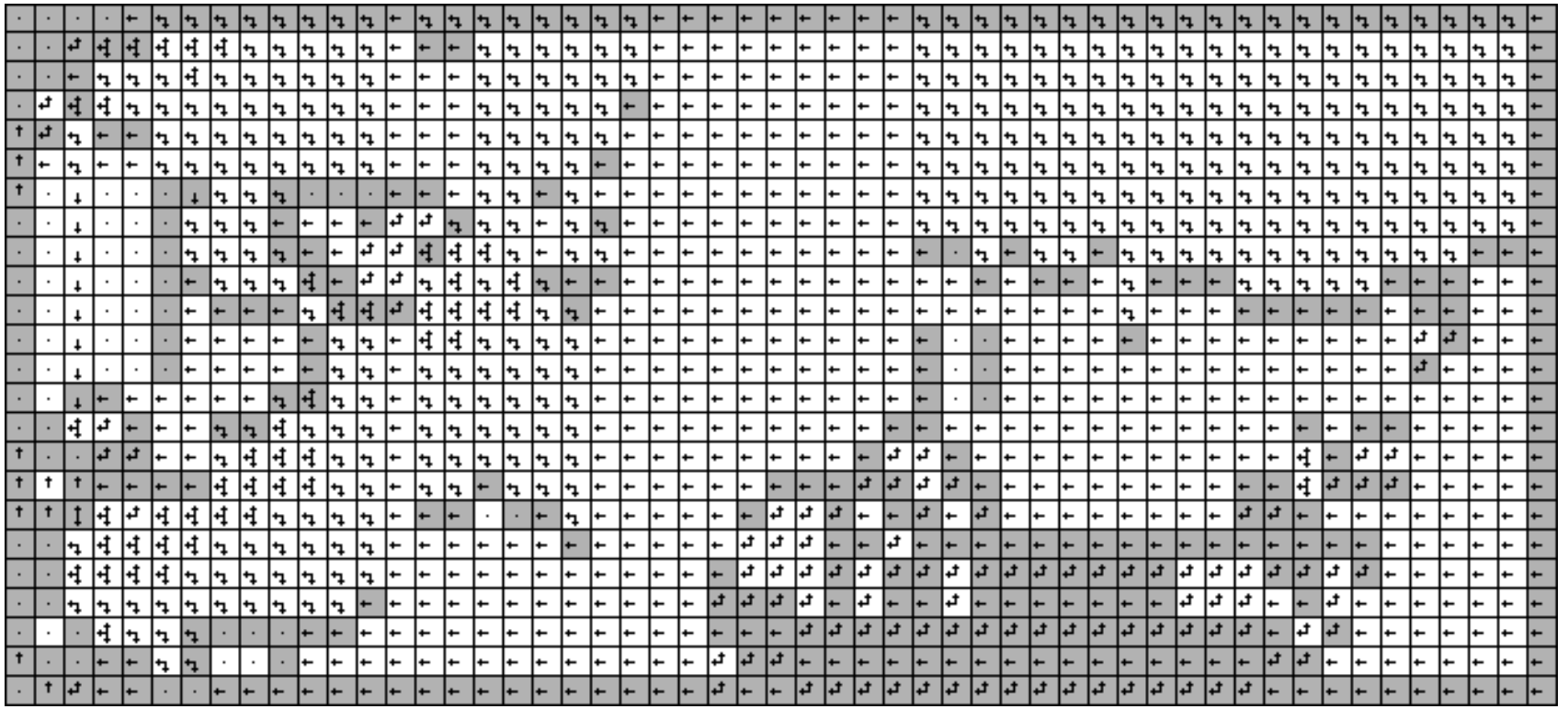

Figura 43 – DEM1000m-raio 20.

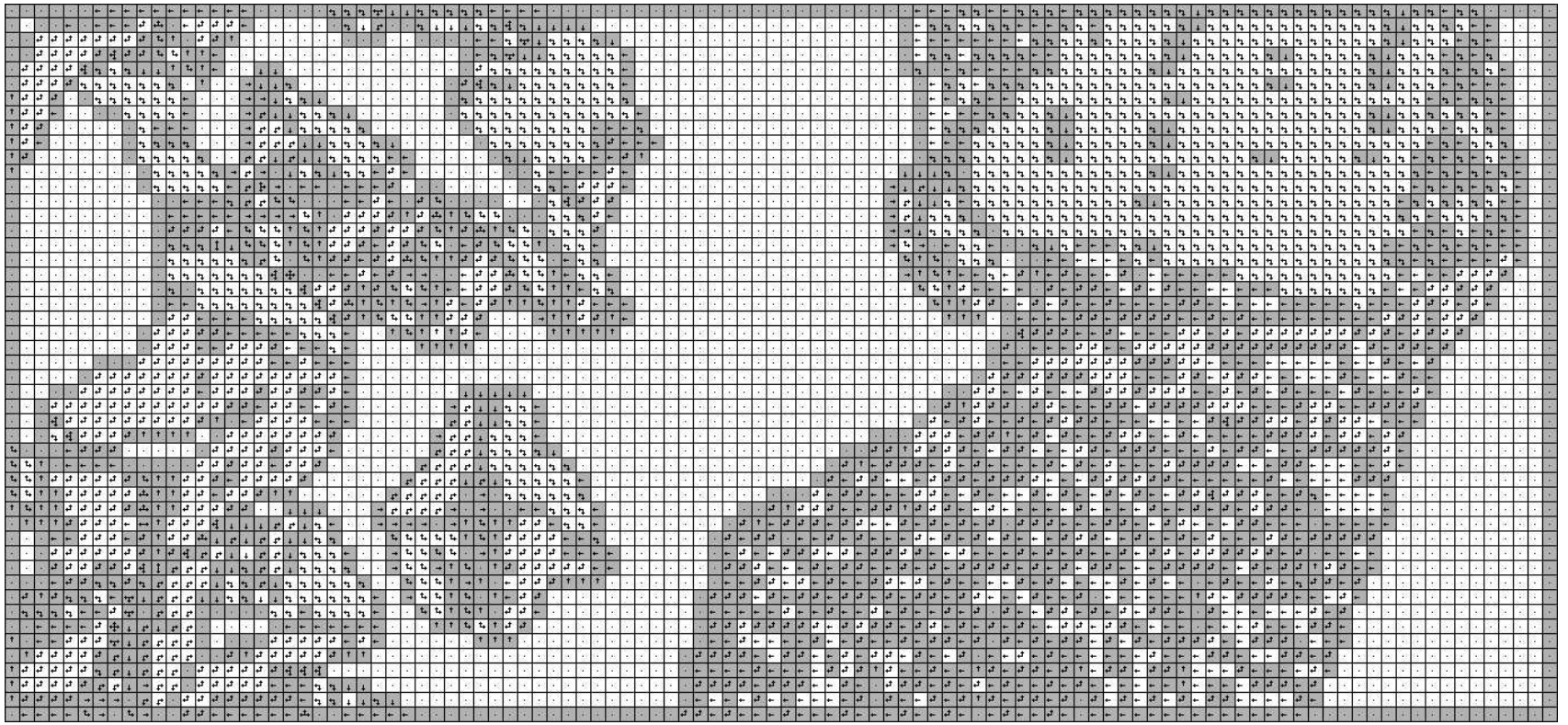

Figura 44 – DEM500m-raio 1.

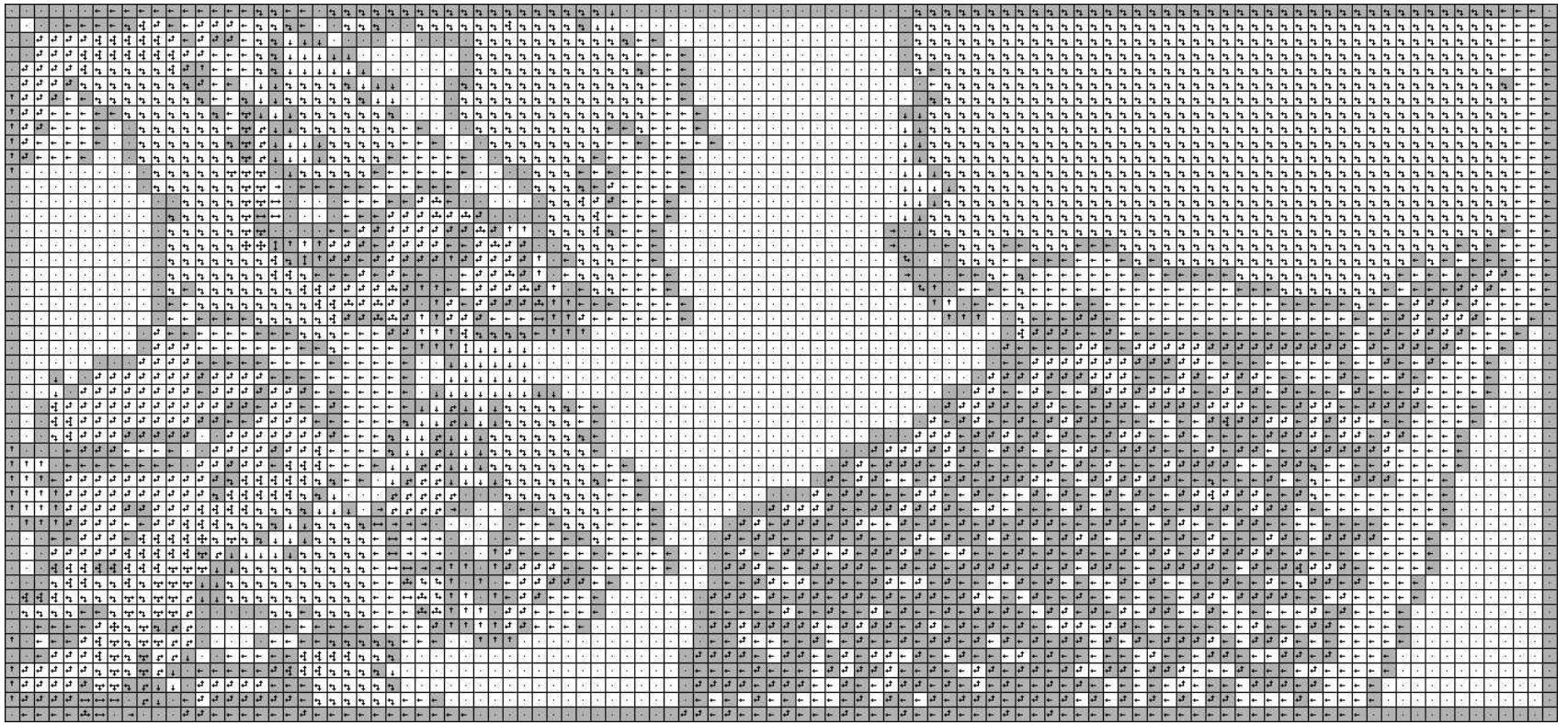

Figura 45 – DEM500m-raio 5.

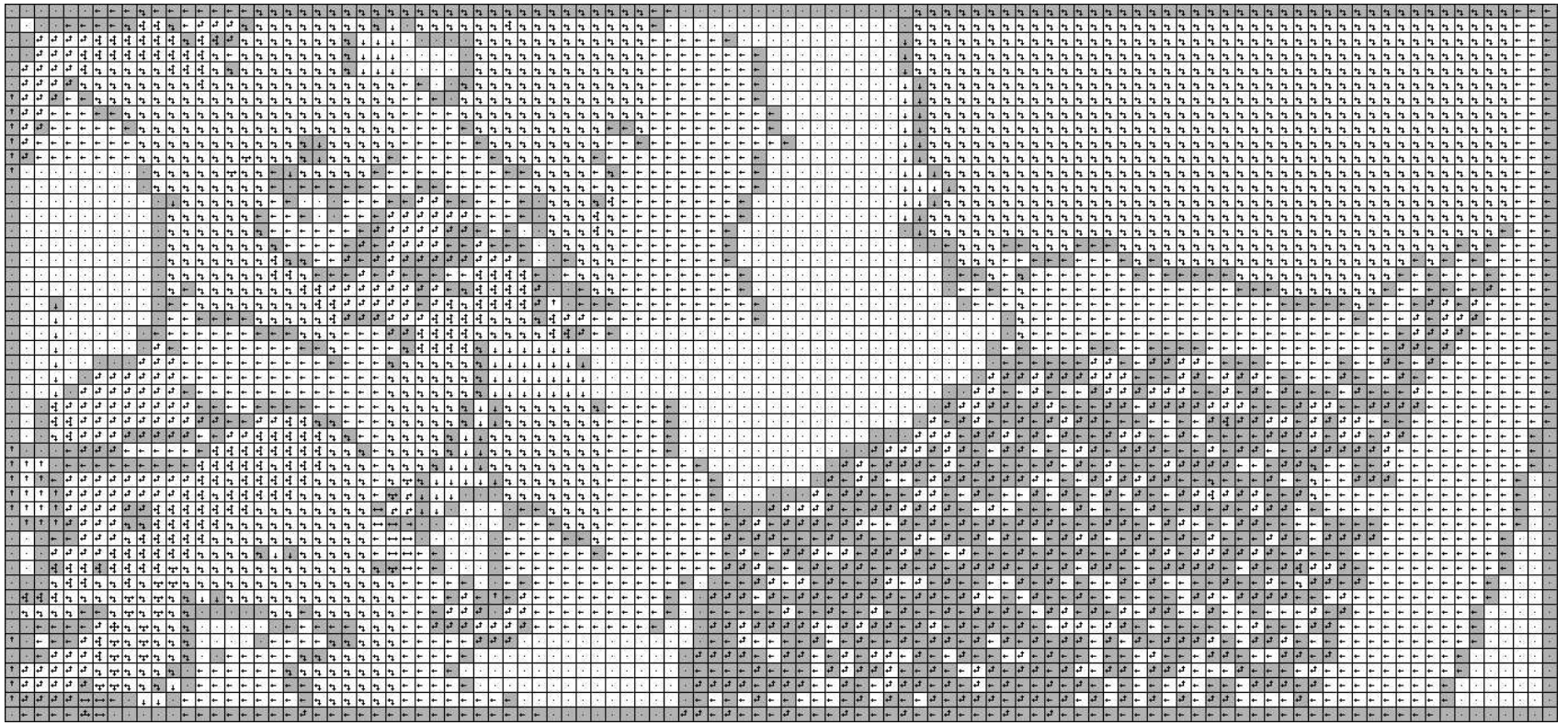

Figura 46 – DEM500m-raio 10.

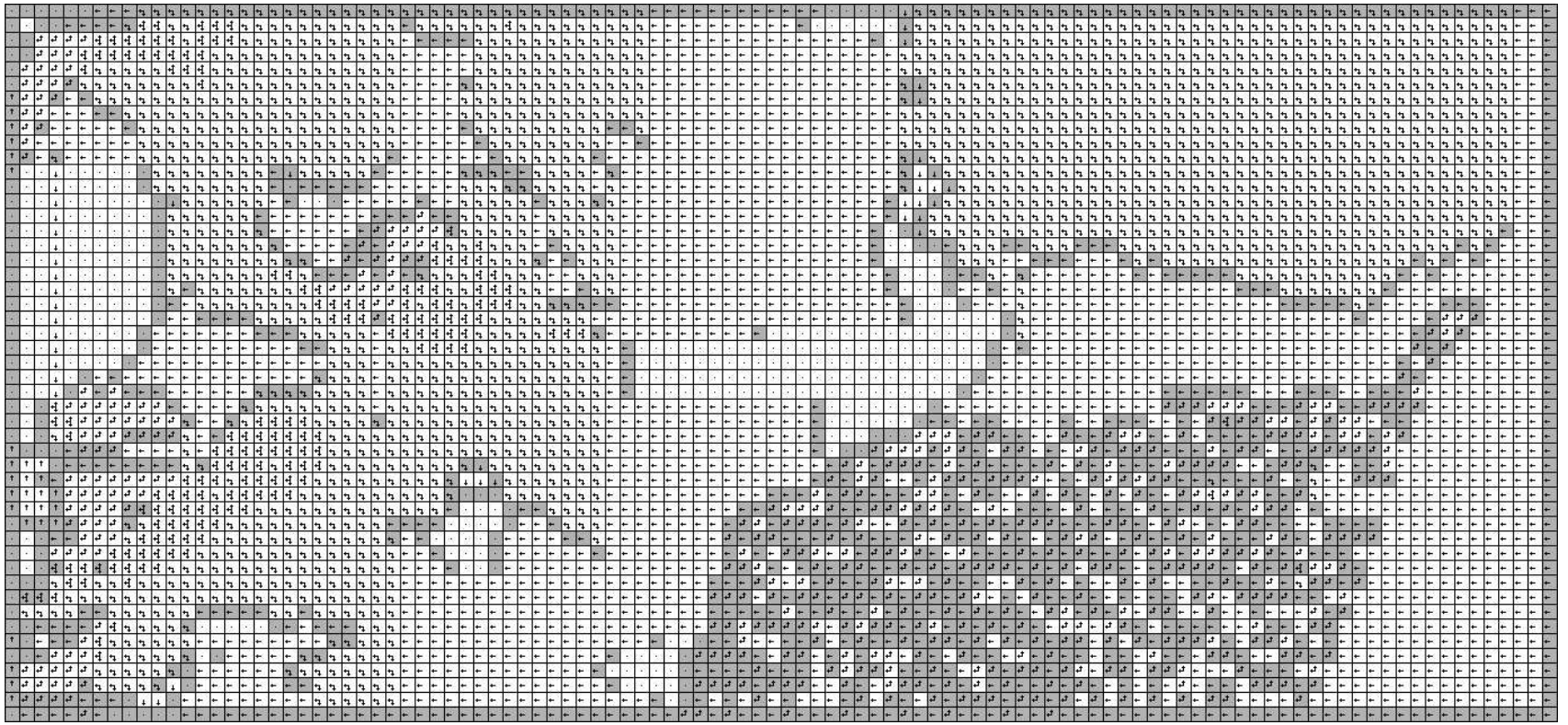

Figura 47 – DEM500m-raio 20.

Apêndice B – Matéria encaminhada para publicação

Durante o desenvolvimento das atividades propostas para realização da dissertação de mestrado, cinco artigos relacionados à este trabalho foram encaminhados para publicação em eventos de destaque nas áreas envolvidas. A Tabela 20 apresenta a relação destes artigos.

Tabela 20 – Publicações relacionadas à dissertação no período 2004/2005.

Trabalho	Publicação	Edição	Notificação	Classificação
HPC-ICTM	PARA 2004	Anais do PARA	Aceito	CIA
	LNCS 2005	Springer LNCS	Aceito	CIA
	ERAD 2006	Anais da Escola	Aceito	S/C
	Reliable Computing 2005	Springer Verlag	Aguardando	CIA
Alocação Transparente "*Grid Computing*"	eScience 2005	IEEE	Aceito	CIB

O primeiro artigo desenvolvido foi referente ao trabalho intitulado "*HPC-ICTM: the Interval Categorizer Tesselation-Based Model for High Performance Computing*". Este artigo foi enviado no final do ano de 2004 para o *PARA'2004 -- Workshop on State-of-the-Art in Scientific Computing*, em Lyngby na Dinamarca. O trabalho foi aceito, apresentado e publicado nos anais da conferência.

O *PARA'2004* recomendou o trabalho, a partir da seleção dos melhores artigos da conferência no ano, para *LNCS -- Lecture Notes in Computer Science* no ano de 2005, apenas sendo necessário incluir mais algum conteúdo original. Sendo assim, o segundo artigo foi elaborado e enviado para o *post-proceedings* do *PARA'2004*, sendo aceito e publicado logo a seguir pela *Springer LNCS*.

Após a publicação em *LNCS*, o trabalho foi novamente recomendado para o *Journal Reliable Computing 2005*, bastando incluir mais algum tipo de conteúdo original. Este novo artigo (terceiro) foi enviado e até o presente momento encontra-se em fase de avaliação.

O quarto artigo desenvolvido foi referente ao trabalho intitulado "*HPC-ICTM: um Modelo de Alto Desempenho para Categorização de Áreas Geográficas*". Este artigo foi enviado em outubro de 2005 à *ERAD 2006 - Escola Regional de Alto Desempenho*, para o fórum de pós-graduação. O trabalho foi aceito e será publicado nos anais da escola no início do ano de 2006.

Os quatro trabalhos mencionados acima estão diretamente ligados à esta dissertação de mestrado. Já o último artigo desenvolvido, intitulado "*Transparent Resource Allocation to*

Exploit Idle Cluster Nodes in Computational Grids", não está diretamente ligado à dissertação. O trabalho propõe uma estratégia para alocação de recursos ociosos de máquinas agregadas na execução de aplicações *Bot* em grades computacionais, o que acabou servindo de base para a adaptação do modelo ICTM para a plataforma de grades.

Este artigo foi enviado para o *Workshop* do *eScience 2005 -- 1st IEEE International Conference on eScience and Grid Technologies*. O trabalho foi aceito e será apresentado em dezembro deste ano.