**Universidade de Brasília**
**Instituto de Ciências Exatas**
**Departamento de Ciência da Computação**

# Agentes Racionais Baseados no Modelo Belief-Desire-Intention para o Sistema Multiagente MASE

Cássio Giorgio Couto Coelho

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Célia Ghedini Ralha

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Prof.ª Dr.ª Alba Cristina Magalhães Alves de Melo

Banca examinadora composta por:

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora) — CIC/UnB
Prof. Dr. Felipe Rech Meneguzzi — PUCRS
Prof. Dr. George Luiz Medeiros Teodoro — CIC/UnB

**CIP — Catalogação Internacional na Publicação**

Coelho, Cássio Giorgio Couto.

Agentes Racionais Baseados no Modelo Belief-Desire-Intention para o Sistema Multiagente MASE / Cássio Giorgio Couto Coelho. Brasília : UnB, 2014.
124 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2014.

1. Bioma Cerrado, 2. Distrito Federal, 3. Gestão Ambiental, 4. Inteligência Artificial Distribuída, 5. MASE-BDI, 6. Modelo Baseado em Agentes, 7. Raciocínio Prático, 8. RIDE, 9. Uso e Cobertura da Terra

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

**Universidade de Brasília**
**Instituto de Ciências Exatas**
**Departamento de Ciência da Computação**

# Agentes Racionais Baseados no Modelo Belief-Desire-Intention para o Sistema Multiagente MASE

Cássio Giorgio Couto Coelho

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora)
CIC/UnB

Prof. Dr. Felipe Rech Meneguzzi
PUCRS

Prof. Dr. George Luiz Medeiros Teodoro
CIC/UnB

Prof.ª Dr.ª Alba Cristina Magalhães Alves de Melo
Coordenadora do Mestrado em Informática

Brasília, 15 de agosto de 2014

# Dedicatória

Dedico este trabalho à minha mãe, grande motivadora da minha vida. Sem você eu não seria.

# Agradecimentos

O sucesso dessa pesquisa não seria possível sem uma série de pessoas na minha vida, ajudando das mais diversas formas, direta ou indiretamente. Em primeiro lugar agradeço à minha mãe, ao meu pai e ao meu irmão, por terem me apoiado em todo o tipo de dificuldade nesses últimos anos. Agradeço também a meu companheiro Felipe, que esteve ao meu lado pelos últimos sete anos da minha vida me provendo companheirismo e estabilidade emocional, e a sua família, por ter me acolhido em seu núcleo como um filho querido.

Agradeço também à minha orientadora, Célia, que em muitos momentos me trouxe de volta à terra para que eu pudesse concluir esse projeto, e pode conduzir o meu caminho nesta caminhada. Agradeço aos professores George, Alexandre e Bruno por terem oferecido ideias, insumos e recursos valiosos que engrandeceram essa pesquisa de forma imensurável. Um obrigado especial também à Carolina Abreu, grande companheira nessa longa jornada, que foi parte essencial para que essa projeto tenha crescido tanto e continue avançando. Agradeço também ao professor Ravi Passos, que se mostrou grande amigo e conselheiro durante esse mestrado, ao professor Sérgio Cortes, por ter confiado em mim e incentivado minha conquista, e ao professor Ralha, que me ajudou nos últimos momentos de escrita desse trabalho.

Por fim, agradeço a todos os amigos que motivaram essa conclusão: Lira, Fernanda, Rafael, Gustavo, André Z., Léo, Bruno, André M., Rodrigo, Jaque, Ítalo, Diego M., Diego F., colegas da ABD, a amizade de vocês é inestimável. Muito obrigado!

*"Aut inveniam viam aut faciam."*

***Hannibal***

# Resumo

MASE, acrônimo para *Multi-Agent System for Enviromental Simulation*, foi uma aplicação desenvolvida para a investigação da dinâmica do uso e conversão do solo em cenários ambientais, e apresentou bons resultados utilizando o modelo Cerrado-DF. Como forma de aumentar o domínio dessa ferramenta, este trabalho explorou o modelo de cognição baseado em *Belief-Desire-Intention* por meio do *framework* JADEX. Para isso, a arquitetura do MASE foi reformulada e seu código foi refatorado, tanto para que os agentes representassem melhor o raciocínio humano quanto para que a aplicação possuísse melhor desempenho de tempo na execução das simulações. A evolução dessas características trouxe o sucessor do MASE, que foi validado nesse trabalho por meio de dois estudos de caso. Os resultados gerados com essa nova proposta foram comparados com os obtidos no MASE, testando assim a flexibilidade da ferramenta e a melhoria do desempenho do sistema.

**Palavras-chave:** Bioma Cerrado, Distrito Federal, Gestão Ambiental, Inteligência Artificial Distribuída, MASE-BDI, Modelo Baseado em Agentes, Raciocínio Prático, RIDE, Uso e Cobertura da Terra

# Abstract

MASE, acronym to Multi-Agent System for Enviromental Simulation, was an application developed for land usage and cover change dynamics investigation, using different environmental scenarios, and good results with the Cerrado-DF model were obtained with its usage. To increase the domain of MASE, this work explored the Belief-Desire-Intention cognition model using the JADEX framework. This objective was obtained by MASE architecture reformulation, with code refactoring, so the agents could better represent human rationality, as the system time performance could be enhanced. The evolution of this features brought MASE's sucessor: MASE-BDI, which was validated by two case studies. The generated results were compared with the ones obtained in the past with MASE, so the MASE-BDI flexibility could be tested, as performance enhance could be proved as well.

**Keywords:** Agent-Based Model, Cerrado Biome, Distributed Artificial Intelligence, Environmental Management, Federal District, Land Use Cover Change, MASE-BDI, Practical Reasoning, RIDE

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

A ação humana é um dos principais fatores de transformação do meio ambiente, tanto em intensidade quanto em extensão e complexidade. A dinâmica antrópica não é motivada apenas por eventos físicos ou biológicos, como erupções vulcânicas ou crescimento populacional de presas, mas também pela interação da sociedade em que se contextua. Economia, política e cultura são alguns dos elementos relacionados ao uso antrópico dos recursos naturais [8].

Utilizando conceitos da álgebra, podemos definir a sociedade como um conjunto de elementos humanos. Tal como um conjunto, possui uma unidade que reúne seus elementos, mas também um heterogeneidade intrínseca conferida pelos seus próprios elementos. Dessa forma, o homem, com suas características únicas e contexto próprio, age e pensa diferentemente de outros indivíduos, por mais que sejam todos humanos em uma mesma sociedade. Assim, dentro da problemática da organização da exploração em espaços naturais, há a grande dificuldade de tratar a ação antrópica do ponto de vista de sociedade e meio ambiente e, ao mesmo tempo, de indíviduo e localidade [9].

Nesse contexto, como é possível compreender a dinâmica ambiental utilizando-se simultaneamente os conceitos de grupo e indivíduo que residem no mesmo ambiente? Tal questionamento é de extrema importância na criação de políticas públicas que visam a longevidade dos recursos naturais. Sistemas computacionais utilizam diversas ciências em busca de aplicações que auxiliem gestores ambientais na atividades de identificação de zonas de risco ambiental, melhoramento de produção agropecuária e coordenação do crescimento urbano, por exemplo [10]. Essas aplicações, por sua vez, também utilizam diferentes abordagens dentro da Ciência da Computação para cumprir seus objetivos.

A criação, o aperfeiçoamento e o uso de diferentes aplicações computacionais são interessantes do ponto de vista da manipulação da informação. A contemplação de diversos ângulos do mesmo problema utilizando diferentes ferramentas, considerando o foco em características distintas do ambiente, é importante para a obtenção de informações mais

detalhadas e completas. [8, 11, 12]

## 1.1 Caracterização do Problema

MASE, acrônimo para *Multi-Agent System for Environmental Simulation*, é um software que foi desenvolvido para simular modelos ambientais de uso da terra utilizando a abordagem multiagente. O estudo de caso do Cerrado-DF (Capítulo 3), utilizado para validar a implementação do MASE, mostrou resultados bastante satisfatórios em simulações que consideram diferentes tipos de agentes transformadores com variações de subtipo – agricultor e pecuarista. Os dois tipos apresentaram comportamentos pré-determinados baseados em suas regras, enquanto os subtipos foram utilizados para diferenciar a exploração de espaço por atividades de produtores individuais da exploração por empreendimentos (por exemplo, cooperativas e produtores de larga escala) [13–15].

Não obstante, projetar um tipo de agente com regras e comportamentos fixos não é suficiente para expressar a individualidade das entidades no processo de tomada de decisão integrado, pois suas ações não são adaptativas ao contexto ambiental que se insere [2, 16]. Por mais que uma série de comportamentos pré-definidos possam ser adequados para um grupo de agentes em um contexto, deve-se lembrar que a sociedade é um conjunto de indíviduos ou entidades individuais, cujas ações singulares não são representadas como se apresenta no atual modelo do MASE. Por mais que os indivíduos possuam tarefas e objetivos comuns, suas intenções podem se modificar de acordo com o as mudanças do ambiente em que estão inseridos, conforme suas percepções e conforme outros valores individualmente importantes [2, 5]. Torna-se necessário que um modelo de racionalidade mais consistente, extensível e individualizado seja definido para que o MASE atenda aos requisitos citados.

Além disso, considerando-se a perspectiva de desempenho do sistema, percebe-se que o MASE produz resultados em intervalos de tempo não ideais [17, 53]. Por mais que as próprias características da simulação influenciem no gasto de recursos computacionais, como imagens muito grandes ou regras muito complexas, faz-se necessário que a própria aplicação seja otimizada para conseguir usar estratégias que reduzam o tempo de execução, resultando em uma maior agilidade do processo iterativo de validação, adequação de atributos do modelo e nova execução.

## 1.2 Justificativa

A Ecologia é definida como a ciência da interação entre os organismos vivos e os ambientes que se relacionam [18]. Trata-se de um campo de estudo multidisciplinar, uma

vez que todos os atributos que concernem aos elementos ecológicos podem influenciar em seus processos. Por exemplo, são de interesse da Ecologia tópicos relacionados ao relevo do espaço natural e do tipo de espécies que nele residem. Essas características, por serem muitas e de grande variedade, justificam em parte a complexidade dos problemas ecológicos. As interações, sejam elas entre organismos ou variáveis do ambiente, e as consequências que elas geram explicam também a sofisticação dos questionamentos em Ecologia [19, 20].

A Ciência da Computação oferece artefatos e soluções que podem ser integradas ao processo metodológico dos ecólogos e de outros profissionais de áreas correlatas, para auxiliar na análise e prospecção de soluções em ambientes dinâmicos e complexos. Inclusive, vários desses artefatos já são usados por subáreas da Ecologia, para expandirem seus domínios e suas possibilidades, ou renovarem seus processos, substituindo práticas tradicionais por usos mais avançados e adequados ao tratamento do grande volume de informações relacionadas [10].

LUCC (acrônimo para *Land Use Cover Change*) é a subárea de Ecologia de Paisagens que lida diretamente com os processos de uso e conversão da terra, necessitando de sistemas computacionais que auxiliem na compreensão dos processos ambientais [21]. Sistemas de Informações Georreferenciadas (GIS), Sistemas de Gerência em Bancos de Dados (SGBD), simuladores e sistemas de inferência são algumas das ferramentas integradas aos processos investigatórios a respeito da dinâmica da terra. O MASE, sendo uma sistema para a simulação ambiental, se encaixa nessa lista de aplicações que auxiliam o entendimento do impacto de uma atividade exploratória de uso da terra, ou a validação de uma política pública, por exemplo [15].

A construção e integração de um módulo de racionalidade mais sofisticado não é apenas uma extensão do MASE, mas sim um aperfeiçoamento necessário para o estudo do ambiente sob uma perspectiva diferente. A abordagem multiagente na simulação traz a metáfora de entidades transformando um ambiente. Essas entidades, em ação durante a simulação, devem ser capazes de resolver problemas em tempo de execução de forma a representar o aumento da fronteira agrícola em uma determinada área, por exemplo. O agregado formado pela exploração de cada agente individual é uma característica emergente da decisão de vários agentes, refletindo o modelo de cognição utilizado. O estudo de modelos de racionalidade e uso do raciocínio prático no MASE é necessário uma vez que quão melhor os agentes representarem o raciocínio humano, melhor serão justificadas suas explorações e movimentações, e mais similares com a realidade serão as ações tomadas na simulação.

Como o MASE é um simulador de modelos operacionais (apresentados na Seção 2.4.1)[22], deve-se também dar atenção à desempenho do sistema. Os modelos operacio-

nais possuem atributos que podem ser modificados pelo usuário ou pela própria aplicação para produzir conjuntos de resultados distintos. Essa modificação, ou calibragem, leva em consideração a qualidade dos resultados. Por exemplo, deseja-se testar a eficiência de uma política pública de exploração do espaço por meio de simulações. Para isso, pode-se explicitamente determinar um conjunto de regras pertinentes a essa política no MASE, e dentre as variáveis internas da simulação, inclui-se a área de atuação da política pública e a quantidade de agentes atuando em todo o espaço. É de interessante do usuário testar várias combinações de área de atuação e quantidade de agentes, pois ele pode obter pontuações diferentes nos resultados que comprovam ou refutam a política pública testada. Sendo assim, percebe-se que em simulações utilizando modelos operacionais, é intríseca uma etapa iterativa de validação baseada em retroalimentação (*feedback*) dos resultados. Com objetivo de agilizar essa etapa, é importante então que o tempo de execução das simulações não tenha uma duração na grandeza de horas (conforme apresentado no Capítulo 5, Tabela 5.1.3). Assim, é possível uma melhor interação do usuário com a simulação.

## 1.3 Questão de Pesquisa

O MASE possui abordagem de Sistemas Multiagentes (SMA) para a simulação de uso e conversão da terra, com o objetivo de auxiliar a tomada de decisão em Gestão Ambiental. O MASE apresentou resultados muito bons utilizando o modelo Cerrado-DF, que estuda a conversão do solo no Bioma Cerrado na área do Distrito Federal entre os anos de 2002 e 2008 [13]. No entanto, a tomada de decisão em seu sistema é concentrada nos poucos agentes que apresentam a racionalidade avançada, além do tempo de execução das simulações ser muito extenso, o que dificulta o processo iterativo de obtenção de resultados e melhoria do modelo simulado [17].

A questão de pesquisa que esse trabalho propõe comprovar é: a inclusão de um modelo de racionalidade baseado em Crenças, Desejos e Intenções pode trazer um maior poder de representatividade do raciocínio humano aos agentes do MASE?

Considera-se, neste trabalho, que representação do raciocínio humano se traduz na capacidade de cada agente conseguir tomar a melhor ação ao enfrentar um problema específico no contexto que se insere.

Consequentemente, a inclusão do modelo de raciocínio BDI pode trazer um aumento do consumo de recursos computacionais. Dessa forma, pode-se questionar: uma nova arquitetura modularizada com acoplamento de componentes poderia trazer uma melhoria no desempenho computacional do MASE?

Deve-se salientar que a nova arquitetura não somente tem como propósito o aumento da escalabilidade dos componentes, mas também a inclusão do modelo BDI aos agentes.

Procura-se então, com esse trabalho uma melhor abstração da cognição humana e organização da modelo arquitetural, que reflita de forma mais adequada cenários de simulação em LUCC.

## 1.4 Objetivos

O objetivo geral deste trabalho é o aperfeiçoamento do simulador MASE, expandindo a racionalidade de seus agentes com a inclusão do modelo cognitivo baseado em Crenças, Desejos e Intenções. Para alcançar essa meta, define-se como objetivos específicos o seguinte:

- Realizar o levantamento do estado arte em ferramentas correlatas em Sistemas Multiagentes para a simulação;
- Realizar estudos em modelos e técnicas de distribuição de carga em sistemas paralelos;
- Definir um modelo arquitetural adequado para o MASE, direcionado a melhor distribuição da tomada de decisão entre os agentes, considerando o modelo de racionalidade adotado.
- Refatorar o código atual do MASE, visando tanto a expansão da racionalidade dos agentes quanto o rebalanceamento de carga do sistema;
- Definir um novo estudo de caso com maior abrangência geográfica, para validar o novo modelo arquitetural com a distribuição de carga.
- Realizar experimentações com os estudos de caso em diferentes cenários.
- Analisar os resultados obtidos, comparando com experimentos passados como forma de validar a melhoria representatividade do raciocínio humano e desempenho do sistema.

## 1.5 Contribuições

A principal contribuição desse trabalho foi a definição e a implementação do modelo de raciocínio dos agentes baseado em BDI, bem como a melhoria de desempenho do MASE. Como consequência da inclusão do modelo citado no MASE, houve o aumento da escalabilidade dos agentes durante a simulação. Uma nova arquitetura foi definida e desenvolvida, resultando no MASE-BDI. As publicações relacionadas a esse trabalho foram: [23], [15], [24], [25], [26].

## 1.6 Metodologia e Estrutura do Documento

A metodologia adotada neste trabalho de mestrado inclui várias etapas. Primeiramente, um estudo em Inteligência Artificial (IA) foi realizado, com o objetivo de se definir qual modelo de racionalidade seria mais adequado à realidade das entidades do MASE. Paralelamente, uma investigação sobre aplicações correlatas foi realizada, para que se pudesse levantar os requisitos para uma aplicação computacional utilizando conceitos no domínio ambiental. Adicionalmente, buscou-se na Computação Paralela um modelo correlato de representação de entidades em um ambiente computacional distribuído, para que se pudesse entender de que forma a carga do MASE poderia ser equalizada. Em seguida, frameworks de SMA foram pesquisados com o fim de buscar as ferramentas adequadas que cumprissem os requisitos levantados.

Foi escolhido como modelo de racionalidade o BDIq e refatorado o código do MASE já considerando esse novo paradigma de cognição e tomada de ações e decisões dos agentes. Dois estudos de caso foram definidos, um já consolidado em outros trabalhos do MASE [15, 17] e outro com mais características e extensão geográfica, e simulados na nova proposta. Os resultados da simulação foram então avaliados segundo metodologia empírica sugerida por Smajgl et al.[27], e também comparados com resultados anteriores do MASE, a fim de responder a questão de pesquisa.

Este documento está dividido em 6 capítulos, que descrevem o percurso metodológico e prático da pesquisa, descrito nos parágrafos anteriores. No Capítulo 2 é feito um levantamento teórico sobre os Sistemas Multiagentes, o modelo de Racionalidade Belief-Desire-Intention (BDI) e as atividades em Gestão Ambiental. Conceitos da área de Computação Paralela também são reunidos nesse capítulo, uma vez que SMA é uma área de interseção entre Inteligência Artificial e Sistemas Distribuídos, e utiliza várias técnicas de paralelismo e administração de recursos para a coordenação dos agentes. Por fim, são apresentadas aplicações correlatas ao MASE. Esses conhecimentos foram pré-requisitos para a construção do módulo de racionalidade e refatoração do código do MASE.

No Capítulo 3 é realizado um histórico da proposta do MASE, e introduzida a nova proposta, e são mostradas sua nova arquitetura e adaptações realizadas para a implementação.

No Capítulo 4 é tratado o caso de uso utilizado, e são apresentadas as variáveis e as regras dos dois modelos utilizados nesse trabalho, Cerrado-DF e RIDE (Região de Integração do DF e Entorno).

No Capítulo 5 são mostrados os experimentos realizados nos diferentes cenários, tanto na proposta antiga do MASE quanto na nova, considerando a ampliação da racionalidade e refatoração da estrutura. Os resultados de acerto do modelo e variação de performance são analisados e discutidos.

Por fim, no Capítulo 6 é concluído o documento, revisitando-se a hipótese para a sua validação e indicando-se os trabalhos futuros a serem feitos.

# Capítulo 2

# Fundamentação Teórica

Programação Orientada a Agentes (*Agent-Oriented Programming*, AOP) é um paradigma de software recente, porém utilizado amplamente em diversos campos de desenvolvimento em pesquisa e em aplicações comerciais. AOP promove a divisão de tarefas de um sistema em múltiplas partes que podem ser executadas individualmente e paralelamente por componentes chamados agentes [28]. Os agentes, entidades autonômas, proativas e sociais, são intrinsecamente modulares, podendo compor um conjunto heterogêneo e de extensão variável, definido de acordo com modelos de raciocínio e conhecimento [29–31].

AOP reúne conceitos de Inteligência Artificial, Sistemas Distribuídos e Programação Paralela [28], podendo ainda relacionar-se com áreas externas à Computação, dependendo do contexto do uso do SMA em questão. O MASE, que simula a conversão e uso da terra, possui relações diretas com a Ecologia e a Gestão Ambiental, que por sua vez possui interligação com outros domínios do conhecimento. Neste capítulo, os conhecimentos diretamente relacionados são expostos como a base de desenvolvimento do MASE-BDI, proposta de extensão do MASE. Também ainda é apresentado o formalismo necessário para a representação do raciocínio na proposta nova, assim como algumas ferramentas desenvolvidas para a implementação de agentes inteligentes.

## 2.1 Sistemas Multiagentes

Inteligência Artificial (IA) é a área da Ciência da Computação que aborda questões a respeito da automação do raciocínio [32]. Adicionalmente a essa definição, IA engloba sistemas que utilizam de raciocínio para resolver problemas diversos [16], ou seja, apresentam características similares à inteligência humana em sua execução. A exemplificar, Visão Computacional, uma subárea da IA, utiliza de métodos e processos para reconhecer padrões em mídias visuais ou em captura de imagens para realizar algum tipo de processamento.

Tabela 2.1: PAGE de um agente táxi automatizado.

| | |
|---|---|
| Percepções (*Perceptions*) | Vídeo<br>Acelerômetros<br>Medidores<br>Sensores do Motor<br>Volante |
| Ações (*Actions*) | Dirigir<br>Acelerar<br>Frear<br>Buzinar<br>Falar |
| Objetivos (*Goals*) | Chegar ao destino com segurança<br>Maximizar lucros<br>Obedecer leis de trânsito<br>Manter os passageiros confortáveis |
| Ambiente (*Environment*) | Ruas<br>Parques<br>Engarrafamentos |

É bastante comum a interseção de IA com outras áreas da Computação. Tomando novamente Visão Computacional como exemplo, IA mescla suas funcionalidades com os métodos de Processamento de Imagens para reconhecer padrões em fotos. Inteligência Artificial Distribuída (IAD) possui muitas características encontradas em Sistemas Distribuídos, e é subárea de IA que trata dos SMA, tópico base deste trabalho.

### 2.1.1 Agentes

Um agente é um elemento computacional construído para interagir em um ambiente perceptível, programável para ter ações autonômas e atingir objetivos definidos [2]. Essas características podem ser representadas organizadamente pelo PAGE (acrônimo para *Perceptions*, *Actions*, *Goals* e *Environment*) do agente. A Tabela 2.1 mostra um exemplo de caracterização de agentes pelo PAGE, utilizando uma adaptação de um agente táxi automatizado [16]. Nesse exemplo, o agente simula um táxi automatizado trafegando por ruas com passageiros, levando-os aos seus destinos com segurança como objetivo primário, maximizando lucros por meio da adoção de caminhos mais eficientes, obedecendo as leis de trânsito e mantendo o conforto dos passageiros como outros objetivos. Possui diversas percepções, como as vias por meio de captura de vídeo, acelerômetros para controle de velocidade, medidores para checar níveis de gasolina ou de chuva, e assim por diante. Suas ações são as de um táxi normal, dirigir, acelerar, frear e buzinar, mas pode ter algumas outras para se comunicar com o passageiro, como falar por onde estão passando.

As percepções dos agentes são limitadas ao ambiente em que eles estão inseridos providos pelo o que os seus sensores capturam [16]. Isso significa que o nível de detalhamento do tipo de estímulos externos que o agente recebe afeta diretamente a qualidade da tomada

de decisão. As ações, por conseguinte, são derivadas das percepções, sendo a própria tomada de decisão em si, além das estratégias para se alcançar o objetivo lançado ao agente.

### 2.1.2 Ambiente

De forma computacional e física, o ambiente do agente é importante ser definido, pois serve como base do agente, denominada plataforma, onde ele percebe as características externas e executa suas ações em busca de completar seu objetivo. Segundo Russel et al [16], o ambiente pode ser classificado dentro de cinco características, determinando seu tipo. A saber:

1. Acessível: diz respeito a quanto do ambiente o agente consegue acessar, podendo ser totalmente ou parcialmente acessível;
2. Determinístico: quanto aos fatores de influência do ambiente, diz respeito se a repetição deles a partir de um mesmo espaço inicial e uma mesma sequência gera o espaço subsequente;
3. Episódico: refere-se à passagem de tempo do ambiente; se esta pode ser segmentada ou não;
4. Estático: Descreve a dinamicidade do ambiente; se ele não se altera enquanto os agentes agem então ele é estático;
5. Discreto: mostra a parametrização do ambiente, ou seja, se seus atributos podem ser classificados a fim de serem representados por meio de elementos discretos. Por exemplo, a temperatura de um ambiente real é um valor contínuo. No entanto, para fins de representação, ele pode ser discretizado em um conjunto de números inteiros.

Um ambiente acessível, determinístico, episódico, estático e discreto é mais fácil de ser descrito e implementado que outros com alguma das características contrárias. A Tabela 2.2 mostra os atributos para o ambiente do táxi automatizado. Como ele trafega em ruas que podem ter dois sentidos, além de ter calçadas e outras áreas não apropriadas para carros, o ambiente é parcialmente acessível. O número de variáveis do trânsito o classifica como não determinístico, além de não episódico e dinâmico. Também é contínuo, devido ao detalhamento e posicionamento das áreas de tráfego.

### 2.1.3 Tipos de Agente

Dentre diversas representações possíveis, agentes podem ser descritos por funções [16]. Elas são mapeamentos de percepções em sequências de ações, com possíveis fluxos de

Tabela 2.2: Classificação do Ambiente para o táxi automatizado

| Característica | Valor |
|---|---|
| Acessível | Parcialmente |
| Determinístico | Não |
| Episódico | Não |
| Estático | Não |
| Discreto | Não |

```
FuncaoAgente(Percepção)
entrada: percepção
saída    : ação
estático: memória
memória ← AtualizarMemoria(memória, percepção);
ação ← EscolherMelhorAcao(memória);
memória ← AtualizarMemoria(memória, ação);
retorna ação;
```

**Algoritmo 1:** Função de mapeamento ação-percepção de um agente. (adaptado de [16])

decisão dependentes do nível de racionalidade do agente. A função encontrada no Algoritmo 1 mostra uma simples representação da lógica do mapeamento. O agente, que possui uma memória interna para registro de estado, recebe uma percepção externa, a armazena e procura a melhor ação a ser feita de acordo com o parâmetro recebido. Uma vez executada essa ação, a memória é novamente atualizada, dessa vez com a ação que foi tomada. Dessa forma, o agente registra o que foi feito, influenciando a próxima execução da sua função.

A capacidade de um agente inteligente é limitada pelo seu conhecimento, seus recursos computacionais e sua percepção [33]. De acordo com esse conjunto de atributos, agentes podem ser classificados em quatro categorias [16]:

1. Agentes Reflexivos Simples: sua racionalidade é a mais simples de todas as categorias. Suas ações são respostas mapeadas diretamente sobre as percepções recebidas, num fluxo condicional. Assemelha-se ao pensamento instintivo, como os reflexos do corpo, como fechar os olhos quando uma luz muito forte é acesa perto deles (Algoritmo 2). Não existe memória estática para registro de estados, apenas condicionais para a ação escolhida a partir da percepção;

2. Agentes Reflexivos com Registro de Estados: são aqueles que ainda possuem ações mapeadas diretamente para percepções num fluxo condicional, mas consideram o estado atual do agente em sua resposta. Sua função de busca de melhor ação

considera uma máquina de estado interna para o retorno (Algoritmo 3). O estado futuro é guardado de acordo com a ação e o estado anterior;

3. Agentes Orientados a Objetivos: mais evoluídos em racionalidade, os agentes desse tipo ponderam, dentre o universo de ações a serem tomadas, qual a melhor em busca de completar um ou vários objetivos. Além de poder considerar estados, sua função de busca de ação considera como as ações podem modificar o ambiente externo (Algoritmo 4). A ação é eleita dentre um possível vetor delas. Além disso, os objetivos podem ser atualizados de acordo com as mudanças do ambiente, os objetivos alcançados e o estado do próprio agente;

4. Agentes Orientados a Utilidade: além de considerar objetivos e ponderar sobre melhor ação a tomar, esse tipo de agente pode considerar outras variáveis alheias ao ambiente que está, chamada de utilidades. Sua resposta pode ser a ação mais econômica, por exemplo, ou pode ter alguma outra influência no raciocínio do agente (Algoritmo 5). Bastante similar à estrutura dos orientados a objetivos, com a exceção da utilidade do agente, que influencia a ação eleita e a atualização dos objetivos.

```
FuncaoAgenteReflexivoSimples(Percepção)
entrada: percepção
saída    : ação
se percepção == percepção_1 então ao← ação_1;
;
senão se percepção == percepção_2 então ao← ação_2;
;
...
senão
 |  ação ← ação_n
retorna ação;
```

**Algoritmo 2:** Função de mapeamento ação-percepção de um agente reflexivo simples.

### 2.1.4 Coordenação de Agentes

Alguns problemas complexos podem ser refatorados em questões mais simples, que inclusive podem ser resolvidos paralelamente. SMA podem oferecer suas funcionalidades de forma expressiva e eficiente na resolução desses problemas [2, 34]. Algumas das propriedades de SMA são: a distribuição de atividades, a descentralização dos dados e a comunicação assíncrona [28]. Enquanto essas propriedades conferem maior facilidade na criação de agentes, torna-se necessário algum tipo de coordenação dos componentes para

```
FuncaoAgenteReflexivoComRegistroDeEstados(Percepção)
entrada: percepção
saída    : ação
estático: estado
ação ← EscolherAcaoConformeEstadoEPercepcao(estado, percepção);
estado ← AtualizarEstado(ação, estado);
retorna ação;
```

**Algoritmo 3:** Função de mapeamento ação-percepção de um agente reflexivo com registro de estado.

```
FuncaoAgenteOrientadoAObjetivos(Percepção)
entrada: percepção
saída    : ação
estático: estado, objetivos
ação ← ElegerMelhorAcao(estado, percepção, objetivos);
estado ← AtualizarEstado(ação, estado);
objetivos ← AtualizarObjetivos(estado, objetivos, percepção);
retorna ação;
```

**Algoritmo 4:** Função de mapeamento ação-percepção de um agente orientado a objetivos.

que conflitos sejam resolvidos em tempo de execução, choques de agentes com ações sobre o mesmo domínio sejam evitados e objetivos de um grupo de agentes sejam alcançados [35]. Uma estratégia de coordenação de acesso deve ser conceituada no SMA para que os agentes executem suas tarefas devidamente.

Um SMA não possui naturalmente um controle global explícito. Cabe então ao projetista escolher uma estratégia de coordenação a ser utilizada que melhor se adeque ao problema que se precisa resolver. A estruturação de agentes é uma solução amplamente utilizada [35] e consiste em formar vínculos entre agentes gerentes e subordinados. Um gerente global pode ser escolhido, e nesse caso a estrutura é vista como uma hierarquia, ou um grupo de agentes é responsável pela coordenação, formando uma federação [36]. Agentes subordinados a esses gerentes então passam a ser administrados no caso de conflitos ou escalonamento de recursos.

É possível também que o planejamento de um grupo de agentes seja deliberado por um gerente ou um grupo. Nesse caso, o plano é montado pelos gerentes e objetivos são delegados aos subordinados, evitando conflitos e melhorando o uso de recursos compartilhados [35]. É importante lembrar que os agentes são entidades autonômas, então sua dependência de outros agentes coordenadores ou gestores não deve ferir seu arbítrio e sua capacidade de resolver problemas [2].

```
FuncaoAgenteOrientadoAObjetivos(Percepção)
entrada: percepção
saída    : ação
estático: estado, objetivos, utilidade
ação ← ElegerMelhorAcao(estado, percepção, objetivos, utilidade);
estado ← AtualizarEstado(ação, estado);
objetivos ← AtualizarObjetivos(estado, objetivos, percepção, utilidade);
retorna ação;
```

**Algoritmo 5:** Função de mapeamento ação-percepção de um agente orientado a utilidades.

### 2.1.5 Protocolos de Interação

Mecanismos e protocolos de interação com linguagem de comunicação, através de coordenação dos diversos agentes, seja com uso de uma abordagem de cooperação ou competição são essenciais na tecnologia de SMA. Na literatura de SMA existem várias abordagens para definir protocolos de interação [2, 34]. Também existem ferramentas de desenvolvimento de SMA, os quais possuem protocolos básicos de comunicação e interação entre os agentes, como no *Java Agent Development Framework - JADE* [28, 37, 38]. No entanto, a complexidade do controle de agentes em um SMA pode aumentar muito, tendo em vista as características de distribuição do processamento e as diversas tecnologias que podem ser utilizadas na solução. Não apenas as várias instâncias de agentes em uma plataforma aumenta a carga computacional, mas também a troca de mensagens em excesso acarreta um gasto de processamento e memória a ser considerado, com a finalidade de se encontrar um equilíbrio ou até formas alternativas de comunicação.

No contexto da utilização de um SMA, qualquer projeto de SMA que possua características de raciocínio, distribuição e diversas tecnologias embutidas, deve utilizar mecanismos de coordenação, seja com cooperação e planejamento sofisticados ou com competição e negociação. Diversas estratégias de protocolo de interação têm sido apresentadas, como forma de obter alto desempenho em sistemas com a mínima intervenção de especialistas humanos. Como parte da integração do SMA no processo de gestão ambiental, é importante a pesquisa desses aspectos, permitindo que a utilização do sistema em si não acarrete em grandes custos de operação e manutenção, mas que esteja inserido de maneira harmônica no trabalho cotidiano do gestor ambiental, no contexto desse trabalho.

Resumidamente, um SMA pode ter seus agentes apresentando as seguintes características[2]:

1. Sociabilidade: os agentes devem ser capazes de interagir e se comunicar;
2. Mobilidade: podem migrar de ambientes quando solicitados;

3. Veracidade: comprometem-se com a comunicação de informações verdadeiras;
4. Benevolência: agentes que não possuem objetivos conflitantes e estejam disponíveis devem tentar prover informações a solicitantes, ajudando-os a cumprir seus objetivos;
5. Racionalidade: devem cumprir seus objetivos, e tentar não cometer ações que os distanciem do que lhes é delegado;
6. Aprendizado e Adaptação: os agentes devem atualizar seus próprios objetivos de acordo com a dinamicidade do ambiente e do conjunto de agentes.

Dessa forma, os agentes podem interagir e trabalhar em torno de um objetivo comum.

### 2.1.6 Ferramentas para o Desenvolvimento de SMA

É possível desenvolver SMA escrevendo sua estrutura inteira em código manualmente. Isso não é necessário, no entanto, visto que já existem diversas ferramentas para o apoio ao desenvolvimento de SMA. Elas já possuem padrões, classes, estruturas para comunicação e outros artefatos prontos para serem utilizados, possibilitando a criação de redes de agentes inteligentes em aplicações diversas, permitindo ao projetista focar na resolução de seu problema e não em criar itens de infraestrutura. Comerciais ou livres, elas podem possuir diferentes abordagens, oferecendo maior flexibilidade e variedade para o usuário. A Tabela 2.1 resume comparativamente as ferramentas exploradas.

#### ABLE

ABLE, acrônimo para *Agent Building and Learning Environment*, é um projeto que foi proposto e disponibilizado pela *MBI T. J. Watson Research Center*. Trata-se de um *framework* Java, com uma biblioteca integrável e uma ferramenta de produtividade para a construção de agentes inteligentes, utilizando aprendizado automatizado e raciocínio. ABLE também provê interface gráfica para criação e configuração de *AbleBeans*, objetos padrões para diversos usos, como criação automática de agentes e regras. A biblioteca integrada proporciona capacidades de comunicação com base de dados, além de escala e inferência de regras, utilizando lógica booleana ou difusa (*fuzzy*), além de possuir técnicas de aprendizado de máquina, como redes neurais, classificadores bayesianos e árvores de decisão. ABLE também disponibiliza uma plataforma para implantação e execução de agentes, inclusive através de ambiente distribuído [39]. Programadores podem extender os artefatos disponíveis para criar soluções personalizadas. A ferramente ABLE pode ser obtida no site[1] da *IBM alphaWorks*.

[1] http://researcher.watson.ibm.com/researcher/view_group.php?id=979

### JACK

*JACK Intelligent Agents*[2] é um ambiente de desenvolvimento comercial orientado a agentes desevolvido pelo *AOS group*. JACK foi desenvolvido para prover extensões a linguagem Java, inserindo elemento orientados a objeto, e foi projetado como um conjunto de componentes fortemente tipados com baixo custo computacional, provendo alta performance. Apesar do JACK possuir abordagem BDI consolidada e interface gráfica complexa, é possível executá-lo em máquinas sem grande capacidade computacional [40].

### JADE

JADE[3], acrônimo para *Java Agent Development Framework*, é um conjunto de ferramentas destinado ao desenvolvimento de SMA, que implementa a arquitetura proposta pela FIPA para a definição do protocolo de interação no sistema. FIPA é a organização conectada com o Instituto de Engenheiros Elétricos e Eletrônicos (*Institue of Electrical and Electronic Engineers*, IEE), e é responsável pela especificação padrão para o desenvolvimento de tecnologias baseadas em agentes inteligentes. JADE é um *middleware* para desenvolvimento e execução de agentes inteligentes em Java, e foi desenvolvido pela Telecom Italia Labs (TILAB). Toda a infraestrutura necessária para a criação, comunicação e administração de agentes é disponibilizada em classes prontas para serem usadas ou extendidas e personalizadas. JADE também provê uma plataforma para execução dos agentes desenvolvidos, que já provê serviços de páginas amarelas e brancas. A plataforma também possui suporte para ambientes distribuídos, desde que cada máquina pertecente ao cenário possua uma plataforma JADE ativa. Assim, os agentes podem migrar de máquina e comunicar-se inter-plataformas em tempo de execução. São também ferramentas do JADE um módulo de administração e *debbugging* de agentes, facilitando o trabalho do programador e do projetista no desenvolvimento do SMA. Outras características do JADE são descritas na Seção A.1.

### JADEX

JADEX[4] é um *framework* de desenvolvimento integrado ao JADE, que extende suas capacidades adicionando uma camada abstrata sobre as funcionalidades originais. Além de ser *middleware*, JADEX também disponibiliza aos seus agentes funcionalidade de raciocínio direcionado a resolução de objetivos por meio de planos de ação. De forma similar ao JADE, JADEX provê uma Interface de Programação Java (API) integrável ao SMA,

[2] http://aosgrp.com/products/jack/
[3] http://jade.tilab.com/
[4] http://www.activecomponents.org/bin/view/About/Features

definindo mecanismos de envio de mensagem, administração de agentes e controle da plataforma prontos para o programador utilizar, tanto construindo classes em Java quanto definindo agentes e capacidades em XML. A adição da camada de raciocínio inclui outros objetos específicos para a utilização do modelo BDI na construção da lógica de atuação dos agentes. Esses objetos são melhor caracterizados na Seção A.2.

### Jason

Jason[5] é um interpretador de código-aberto que provê extensão ao AgentSpeak. Ele habilita aos usuários a construção de SMA para ambientes considerados muito complexos, devida a quantidade e relacionamento das características presentes. Jason é facilmente personalizável, e é adequado para a implementação de planejamento reativo de acordo com a arquitetura BDI. É tipicamente usado na simulação de robôs interagindo entre si e com o ambiente. As mensagens enviadas entre os agentes seguem o padrão FIPA de interação e por esse motivo podem comunicar-se com agentes em plataformas JADE e JADEX [41].

### SeSam

*Shell for Simulated Agent System*, SeSam[6], desenvolvido pela Universidade de Würzburg na Alemanha, traz um ambiente genérico para experimentação com simulação baseada em agentes. Essa ferramenta foca na construção de agentes em Java de forma simplificada, porém permitindo a produção de sistema complexos, em que podem ser incluídas dependências entre os elementos de forma dinâmica. Ela inclui um ambiente de modelagem visual, um ambiente flexível de desenvolvimento Java, um módulo para definir situações na simulação e um sistema de análise integrado. Simulações com diferentes parâmetros iniciais podem ser distribuídos, e as mensagens entre os agentes podem ser enviadas entre as plataformas, inclusive as em JADE e JADEX, uma vez que o SeSam utiliza o padrão de FIPA na comunicação de seus agentes.

### SimAgent

SimAgent[7], fruto do trabalho de Aaron Sloman na Universidade de *Birmingham*, tem como objetivo criar e testar diferentes arquiteturas de agentes, inclusive aquelas em que o ambiente há subsistemas paralelos concorrentes a objetos e agentes. SimAgent pode ser executado como uma ferramenta de simulação independente, ou associado a um sistema interno de um robô, contanto que ele possua poder computacional suficiente para executar

[5] http://jason.sourceforge.net
[6] http://www.simsesam.de/
[7] http://www.cs.bham.ac.uk/research/projects/poplog/packages/simagent.html

as rotinas necessárias. Originalmente, esse processo foi desenvolvido em uma pesquisa que explorava o comportamento similar ao humano em agentes, e acabou por ser usado para outros propósitos, como jogos e simulações. Em essas aplicações distintas do foco inicial, a variabilidade proposta pelos diferentes agentes atraiu os usos diversificados. Por exemplo, algumas simulações utilizavam agentes com sensores e percepções, enquanto outros somente eram utilizados para comunicação. Diferentes tipos de raciocínio foram explorados nesse ambiente com agentes diversificados. SimAgent utiliza a linguagem de programação Pop-11 e o software Poplog para o seu desenvolvimento.

#### ZEUS

Zeus[8], é composto por três componentes funcionais: uma biblioteca com as estruturas e funcionalidades necessárias para a configuração dos agentes; uma ferramenta de construção de agentes, para facilitar o processo de programação; e uma ferramenta de visualização, para a observação e depuração dos agentes em execução. ZEUS foi desenvolvido pela British Telecomunnications e foi construído em Java, mesma linguagem utilizada para a construção dos seus agentes [42].

## 2.2 Modelo de Racionalidade BDI

O agente pode possuir percepções, comportamentos, registros de estados e objetivos. No entanto de que forma ele pode relacionar esses atributos para decidir que ação tomar? Funções de caracterização da racionalidade das quatro categorias de agentes foram apresentadas na Seção 2.1.3. As características do ambiente em que se encontram os agentes foram expressadas na Seção 2.1.2. Nesta Seção, é introduzido modelo de raciocínio prático BDI que demonstra como um agente decide qual será a próxima ação a ser executada, por meio de um processo interno de deliberação e consideração dos fins que se intenta obter.

O modelo BDI tem origem em estudos em Filosofia, sendo primeiramente apresentado em [43]. Em seu estudo, Bratman propõe a organização do raciocínio prático em crenças, que são fatos que o indivíduo toma como verdadeiros em um determinado momento, desejos, que são estados almejados, e intenções, que são atitudes que podem ser tomadas. O processo de internalização desses parâmetros tem como resultado a próxima ação a ser executada, e é dividido em duas etapas de deliberação dos desejos e consideração das consequências das ações possíveis, descrito com mais detalhes na Seção 2.2.1.

O modelo BDI é o mais conhecido e aplicado em agentes inteligentes [44]. As crenças do agente não são apenas fruto de suas percepções. Elas também podem ser inferências

[8] http://sourceforge.net/projects/zeusagent/

| Ferramenta | Propósito geral | Tipo de Tecnologia | Modelos de Racionalidade | Outras funcionalidades | Licença |
|---|---|---|---|---|---|
| ABLE -Agent Building and Learning Environment | Construção de agentes inteligentes utilizando raciocínio automatizado e aprendizagem de máquina. | É um framework Java com biblioteca integrável e ferramenta para criação de agentes inclusas. | Diversos, como lógica booleana, lógica fuzzy, redes neurais, classificadores bayesianos e árvores de decisão. | Integração com bases de dados, plataforma para execução em ambientes distribuídos. | Código livre para usos acadêmicos ou comerciais. |
| JACK Intelligent Agents | Contrução de aplicações baseadas em SMA de baixo custo computacional e grande portabilidade. | Inteiramente escrito em Java, consiste de um suíte de ferramentas destinadas ao projeto e construção de SMA em Java. | Primariamente BDI. | Diversas ferramentas para aferição de desempenho, segurança e integração com bibliotecas diversas. | Comercial. |
| JADE -Java Agent Development Framework | Construção de aplicações SMA diversas. | Framework escrito em Java para a construção de SMA em qualquer linguagem capaz de ser integrada a Java. | Nenhum modelo explicitamente implementado. | Comunicação padronizada de acordo com FIPA; Interface gráfica e agentes de suporte; Integração com o IDE Eclipse; Plataforma de execução em ambientes distribuídos | Código livre (GNU LGPL 2) |
| JADEX Active Components | Construção de aplicações SMA diversas, utilizando paradigmas de programação SCA (Service Component Architecture) e SOA (Service Oriented Architecture). | Framework escrito em Java, extensão do framework JADE. | Primariamente BDI. | Todas as funcionalidade do JADE, além de camadas de rede e segurança disponibilizadas ao usuário. | Código livre (GNU LGPL 2) |
| Jason | Construção de aplicações SMA diversas. | Interpretador de aplicações utilizando a linguagem AgentSpeak(L), com componentes escritos em Java. | Primariamente BDI. | Comunicação padronizada de acordo com FIPA; Interface gráfica. | Código livre (GNU LGPL) |
| SimAgent | Construção de aplicações SMA em simulação, explorando diferentes arquiteturas e complexidades de agentes. | Construído em Pop-11, suporta aplicações escritas Prolog, Lisp e Poplog. | Nenhum modelo explicitamente implementado. | A linguagem de programação própria do SimAgent, Poplog, permite a programação procedural, orientada a objetos e orientada a eventos. Independente do paradigma escolhido pelo programador, é possível a implementação de regras e outros tipos de controle de agentes. | Código livre (GPL), MIT/XFREE86 para as biblioteca de Poplog. |
| SeSam | Construção de aplicações SMA diversas. | Construído em Java, permite a criação de aplicações em Java. | Nenhum modelo explicitamente implementado. | Comunicação padronizada de acordo com FIPA; Interface gráfica. | Código livre (GNU LGLP) |
| ZEUS | Construção de aplicações SMA diversas em ambientes distribuídos. | Construído em Java, permite a criação de aplicações em Java. | Nenhum modelo explicitamente implementado. | Geração de código por configuração visual; | Còdigo livre |

Figura 2.1: Resumo comparativo das ferramentas apresentadas.

a respeito de outros fatos, ou outros valores que são importantes. Elas não são necessariamente verdadeiras para outros agentes ou verdadeiras em geral, mas o são para o determinado agente em um determinado instante de tempo de execução. O agente, como um componente computacional, possui uma parcela de processamento e memória finitos, dessa forma exigindo adequações em sua lógica de raciocínio para que não se torne um fardo na aplicação. Dessa maneira, as crenças devem ter taxas de atualização razoáveis para que não consumam demasiados ciclos de computação.

Os desejos do agente são abstrações que levam às suas metas podendo ser contraditórios, no entanto somente um desejo pode ser selecionado e cumprido por vez. Esse grupo válido compõe um objetivo do agente, que guiam as intenções do agentes para alcançar um estado do ambiente desejado, uma tarefa a ser cumprida ou uma informação a ser obtida.

As intenções são as possibilidades que o agente possui para agir. Elas são guiadas pelos objetivos, mas consideram as crenças para poderem ser executadas. A partir da tríade crenças (fatos considerados verdadeiros), desejos (traduzidos em objetivos posteriormente), e intenções (campo de atitudes disponíveis), o agente pode montar planos de ações. Ações são atos possíveis a serem executados, que quando estruturados formam planos. A etapa de montagem de escolha do objetivo atual e planejamento das ações, tendo em vista a tríade citada, é a deliberação do agente.

A execução do plano pode trazer o alcance do objetivo ou o seu fracasso. Isso se deve a atualização do ambiente ou reconsideração das crenças do agente. Durante o percurso do plano, certas ações podem se tornar inócuas para a o sucesso do objetivo, ou impossíveis de serem concluídas. Isso causa uma outra etapa de deliberação por parte do agente, em que novas intenções podem ser consideradas para uma nova tentativa de realização dos desejos, ou a desistência do objetivo por completo.

### 2.2.1 Sistemas Intencionais e Raciocínio Prático

A Figura 2.2 mostra diversos modelos arquiteturais em Inteligência Artificial para a representação do raciocínio humano. Esses modelos, representados na parte mais externa da Figura, são decorrentes de teorias desenvolvidas por autores dos mais variados campos do co-nhecimento, reunidos em disciplinas, representadas no círculo interno da imagem. É interessante perceber que comumente uma teoria deriva várias arquiteturas, e há arquiteturas que são derivadas de mais de uma teoria. Wooldridge e Jennings realizaram em [45] um levantamento sobre arquiteturas e linguagens de agentes, e utilizaram esse conhecimento para afirmar que a atividade humana, sob o ponto de vista do senso comum, estabelece uma relação de intenção e causalidade. Citando o exemplo por eles exposto no trabalho em questão:

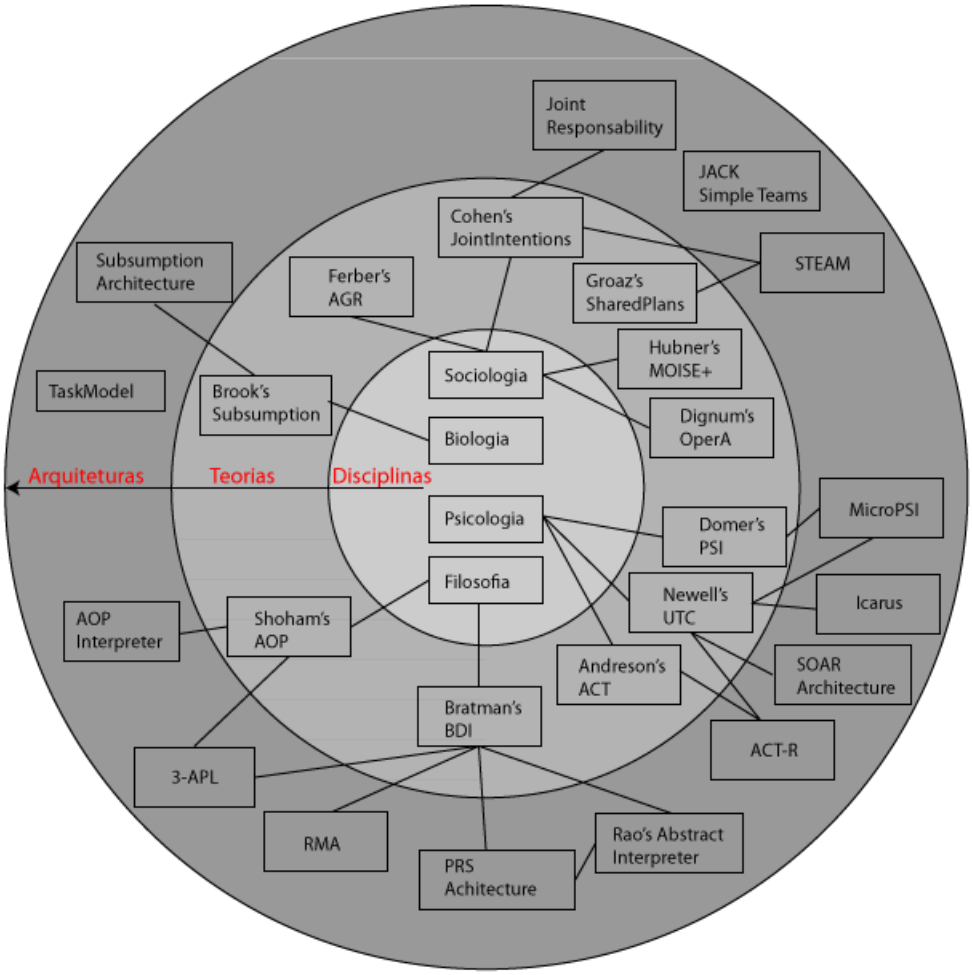



Figura 2.2: Arquiteturas propostas para Sistemas Multiagentes. Adaptado de Braubach et al. [1]

> Janine **pegou** seu guarda-chuva porque ela **acreditou** que ia chover.
> Michael **trabalhou muito** porque ele **queria** obter um Doutorado.

Temos que ações humanas são explicadas por crenças e desejos internos, que motivaram uma intenção. A respeito dessa relação, Daniel Dennett propõe estratégias de predição de ações baseada em posturas [46].

1. Postura Física: Propõe resultados de uma ação baseada em leis físicas. Por exemplo, temos o fato de pólos iguais de imãs se repelirem. Dessa forma, se o pólo norte de um imã for posicionado perto do pólo norte de outro, é possível predizer que eles se repelirão. Essa predição é tem como base argumentativa uma lei física no campo de Eletromagnetismo.

2. Postura de Formação ou Projeto [9]: Diz respeito do propósito intrínseco de um objeto ou ser, ou seja, baseia a predição de uma ação na constituição de um elemento. Exemplificando, uma pessoa conecta uma lâmpada em um soquete. Prediz-se, então, que ela acenderá. Esse tipo de predição não depende de conhecimentos em como a lâmpada foi feita, quais são os mecanismos que fazem com que ela acenda nem as leis físicas que fazem com que ela funcione, mas sim o propósito que ela criada. No entanto, essa predição toma asserções que podem invalidar o resultado caso sejam falsas. Elas são: o objeto ou ser foi feito para o propósito específico e ele funciona normalmente. Uma lâmpada queimada ou de brinquedo são enganosas em suas conclusões se o julgamento não leva em consideração essas características adicionais.

3. Postura Intencional: Considera-se uma entidade capaz de ação e que suas intenções são ações que satisfaçam suas crenças e desejos. Ao determinar-se o conjunto de crenças válidas e os desejos atuais dessa entidade, é possível prever qual a próxima ação a ser feita pela entidade examinando as intenções que satisfazem os desejos possíveis a serem alcançados. Essa predição possui um domínio mais amplo que as outras duas, porém pode causar indeterminações caso: i) não existem crenças válidas; ii) existem crenças falsas (decorrentes de mal-julgamentos ou má-percepção da entidade); iii) existem desejos conflitantes possíveis. Para ilustrar essa postura, considere uma entidade inteligente que joga pôquer. Suas crenças são compostas por: as suas cartas, as cartas sobre a mesa, o jogador J e as apostas atuais. A entidade também crê que J não tem um bom jogo, por deliberação interna. O desejo da entidade atual é ganhar a partida e não perder dinheiro. Suas intenções possíveis são: desistir da partida (*fold*), aumentar a aposta (*raise*) e permanecer no jogo (*stay*). Sendo assim, é possível prever que a próxima ação é permanecer no jogo. Se J aumentar a aposta, a entidade pode deliberar a respeito da suas crenças ou possuir desejos diferentes, e assim sua ação se adaptará à nova realidade. É importante que a percepção é um item importante no processo decisório, pois ela que provê um mecanismo de captura de eventos ocorridos no ambiente. Sua deliberação interna faz parte do processo de atualização de crenças e desejos.

Os Sistemas Intencionais são aqueles compostos por uma ou mais entidades cujas ações podem ser preditas pela reunião de suas crenças, desejos e sagacidade racional [47]. Percebe-se que o domínio da predição é nas ações de uma entidade, e que é essa entidade

[9] Dennett utiliza o termo em inglês *Design Stance* em seu trabalho[46]. A tradução do termo *design* para português possui diversos sentidos, portanto utilizou-se os termos formação e projeto, pois trazem a semântica de amâgo de um objeto, ser ou entidade palpável.

é capaz de agir, ou seja, afetar o ambiente em que se encontra em busca da completude do raciocínio. O Sistema Intencional em si pode possuir ordens [46], listadas como:

1. Sistema Intencional de Primeira Ordem: São aqueles em que as entidades possuem crenças sobre o ambiente em que se insere e desejos internos, e com base neles suas ações podem ser preditas por suas intenções.

2. Sistemas Intencionais de Segunda Ordem: Além de possuir as caracteristícas de um sistema de Primeira Ordem, possui desejos e crenças a respeito de desejos e crenças, podendo ter intenções de segunda ordem. Isso não se aplica somente a seus desejos e crenças próprios, mas de qualquer entidade presente em seu ambiente.

Os Sistemas Intencionais de Segunda Ordem, por mais que sejam mais complexos, possuem um nível de alcance maior sobre o sistema, uma vez que suas possibilidades não se limitam aos desejos internos, crenças e percepções do ambiente. Entidades capazes desse alcance podem controlar outras entidades, além de possuir maior controle de si mesmas.

O conceito de Raciocínio Prático [47] se baseia na proposta de Dennett no que tange à Postura Intencional e as ordens dos Sistemas Intencionais, e inclui com algumas especificidades. O conjunto de desejos afins é composto em um objetivo, e ações tomadas são direcionadas a realização de metas. Adicionalmente, possíveis objetivos conflitantes são considerados durante a etapa de deliberação. Dessa forma, a entidade assume um deles e tenta cumpri-lo. No caso de sucesso ou de falha, uma nova deliberação é feita.

Sendo assim, o Raciocínio Prático utiliza ações possíveis em seu processo cognitivo, ao contrário do Raciocínio teórico, que utiliza fatos para chegar a uma conclusão. O modelo BDI é uma aplicação de Raciocínio Prático que utiliza a tríade Crenças, Desejos e Intenções na deliberação de planos de ações em busca do alcance de um objetivo, que caminha em busca da solução de um problema [2].

O processo cognitivo prático pode ser subdividido em duas etapas[2]:

1. Deliberação: Face a suas crenças válidas, a entidade decide quais desejos almeja conquistar na situação que se encontra. O processo deliberativo é variado, e pode ser definido por uma máquina de estados interna ou por um objetivo coletivo que o grupo o qual a entidade pertence busca alcançar, por exemplo. O resultado dessa etapa é o conjunto de intenções atuais ligadas a realização do desejo escolhido

2. Raciocínio meios-fim: Estabelecido o desejo ou conjunto de desejos a serem realizados, a entidade então organiza suas intenções a fim de completá-los. Ela, por meio de um planejador interno, toma o resultado da etapa anterior (objetivos e intenções atuais), a ações possíveis, percepções capturadas do ambiente e suas crenças. O resultado dessa etapa é o plano de ações (Figura 2.3).

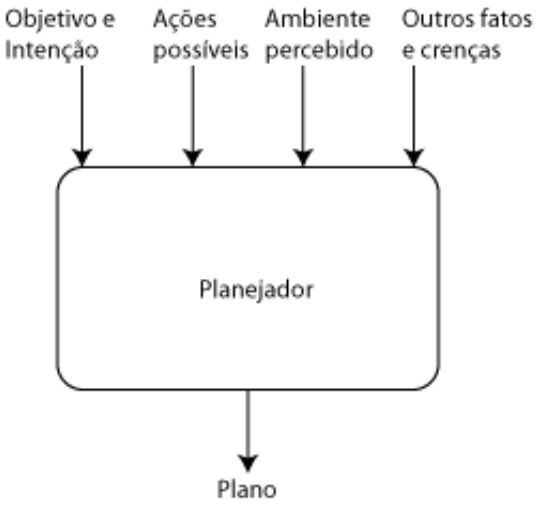



Figura 2.3: O módulo planejador de um racionador prático. Esse módulo é responsável pela etapa meios-fim do raciocínio.

Várias arquiteturas utilizando o modelo BDI foram propostas por vários autores [48]. A Figura 2.4, adaptada de [2] mostra de forma geral o núcleo de uma entidade inteligente que segue o modelo PRS. Bratman propõe em 1987 o *Procedural Reasoning System* (PRS) [44], cuja arquitetura simples foi muito bem estabelecida, sendo incluída em vários sistemas posteriores . As ações possíveis são agrupadas em planos e listadas em biblioteca para uso em tempo de execução, representando efetivamente uma intenção de uma entidade. Agentes de um SMA que implementam essa arquitetura proposta são denominados agentes deliberativos [49].

### 2.2.2 Formalizando a Representação do Planejamento

A busca pela conquista de um objetivo sem planejamento é ineficiente, uma vez que as consequências das ações tomadas podem não apenas gastar recursos desnecessariamente, mas também atrasar ou impossibilitar o alcance de uma meta. Por mais que planejar uma sequência de ações possa ser complexo e apresentar um certo custo, é uma atividade que é mais eficiente que a tomada imprudente de atitudes. No entanto, de acordo com o contexto geral, com a dinamicidade do ambiente e com o universo de ações possíveis à entidade em questão, os planos criados podem falhar em atingir o objetivo. Ainda assim, eles podem ser reaproveitados em uma nova tentativa ou em uma atividade de reflexão e correção dos erros percebidos, processo de grande importância quando o foco da entidade é o aprendizado [50].

A formalização da representação do planejamento é o essencial para que entidades inteligentes possam construir planos de ação de forma automatizada, baseada em objetivos,

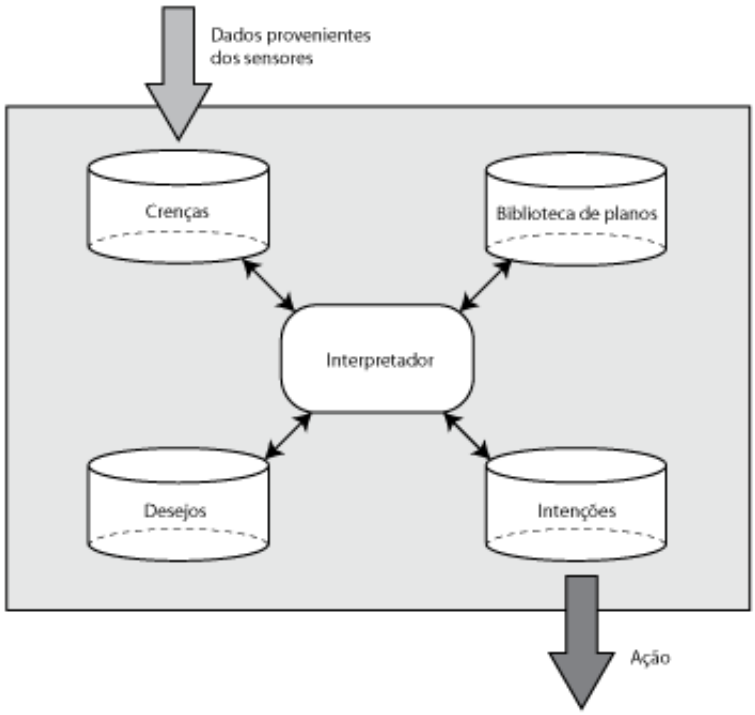



Figura 2.4: Uma forma geral de se representar o modelo genérico da arquitetura PRS. Adaptado de [2]

parâmetros internos e percepções coletadas do ambiente. Também são importantes na criação dos planos as pré-condições necessárias para as ações, as consequências das ações tomadas, a probabilidade de sucesso de uma ação e a situação atual. Dentre diversos formalismos e sistemas de representação do raciocínio automatizado, STRIPS (acrônimo para *Stanford Research Institute Problem Solver*) se destaca pela sua flexibilidade de aplicação em diferentes contextos da IA, inclusive SMA [51]. Nesse tipo de representação são considerados um estado inicial $I$, que descreve de que forma o ambiente se encontra inicialmente, e um conjunto de operadores $O$, que corresponde a ações que modificam o estado em que o ambiente se encontra. Cada operador é uma tupla composta pela ação $a$, por suas pré-condições $pre$, que descrevem que aspectos do ambientes devem ser verdadeiros para sua aplicação, e suas listas de adição $add$ e de remoção $del$, que são respectivamente os aspectos do ambiente que são adicionados e removidos quando a ação é executada [52].

A flexibilidade de STRIPS se traduz em adicionar, remover ou adequar aspectos nos operadores, e por mais que não seja o tipo de formalismo mais recente, ele ainda é muito utilizado e extendido. Por exemplo, Oates et al. adaptaram o espaço de operadores $O$ com a inclusão de $prob$, a probabilidade da ação ter efeito [50]. Isso satisfaz a descrição de operadores que possuem influência de fatores que não podem ser controlados diretamente, ou seja, possuem chances de ocorrer e direcionar o resultado da ação. Dessa forma, sistemas em que ações tem probabilidade de sucesso podem ser formalizados utilizando

STRIPS sem a supressão dessa característica.

Redes de Tarefas Hierárquicas (*Hierarchical Tasks Networks*, HTN), são definidas como uma forma de representação mais natural de relacionamento de ações em grafos direcionados de dependência para a automação do planejamento [53]. As tarefas em HTN são adaptações dos operadores $O$ de STRIPS, em que os espaço de pré-condições e consequências consideram o encadeamento de ações tanto atômicas quanto compostas. Ações atômicas são definidas como $a$ na proposta original de STRIPS, enquanto as ações compostas são listas parcialmente ordenadas de ações atômicas, para que seja garantida a sua execução sequencial. Uma aplicação para HTN é a representação do comprometimento de ações, ou seja, a garantia da execução de uma ação por parte de agentes e seus desdobramentos. Meneguzzi et al. propõem, em SMA, uma formalização de comprometimento social de ações e objetivos dos agentes na atividade de planejamento utilizando HTN [54]. A motivação do trabalho de Meneguzzi et al. é a proposta de formalizar o planejamento de situações em que a conquista de um ou vários objetivos depende da cooperação de vários agentes e suas interações. O compromisso assumido por um agente em suas ações é crucial para o planejamento por parte de outros agentes presentes.

## 2.3 Computação Paralela

Computação Paralela é uma disciplina da Ciência da Computação que estuda as aplicações computacionais capazes de serem multiprocessadas cooperativamente em busca da solução de um problema [55]. Atualmente, com a popularização de *hardware* baseado em vários núcleos de processamento, como processadores e placas de vídeo, tal área se tornou mais relevante em projeto de *software*[56]. Uma vez que várias tarefas são conjuntamente executadas ao mesmo tempo, recursos comuns como memória, rede e os próprios processadores devem ser tratados diferentemente, evitando conflitos entre as linhas de execução que levem a soluções equivocadas ou travamento da aplicação, por exemplo.

Historicamente, computadores com múltiplos processadores eram utilizados majoritamente para pesquisa e uso não-doméstico. Simulações científicas, previsão climática e renderização de modelos tridimensionais eram alguns dos destinos desses computadores. No entanto, a demanda do mercado e a pesquisa em produção de *hardware* com capacidades de multiprocessamento motivaram a popularização de CPU multicores e GPU em várias aplicações pessoais e domésticas, como telefones, computadores, *videogames* e televisões [55]. Com isso, a importância da disciplina de Computação Paralela aumentou por conferir técnicas utilizadas em vários tipos de aplicação moderna.

### 2.3.1 O Modelo Tarefa-Canal

Dentre os modelos de programas paralelos, destaca-se o tarefa-canal por se tratar de um modelo de alto-nível, que se preocupa em descrever os programas considerando comunicação entre os componentes, mas sem considerar compartilhamento de recursos [3]. Dentro de desenvolvimento de sistemas operacionais, esse modelo é relevante pois se assemelha como processos se comportam em uma fila [57]. Também podemos fazer um paralelismo com SMA, uma vez que agentes em execução podem ser simplificados a esse modelo.

Uma tarefa é qualquer programa sequencial que possa ser executado em uma máquina clássica de arquitetura de von Neumann [55], em conjunto com sua memória local e suas portas de entrada e saída de dados [58]. A memória local possui as instruções a serem executadas, enquanto as portas servem de entrada e saída para as mensagens entre diversas tarefas. Um canal é uma fila de mensagens que liga a saída de uma tarefa à entrada de outra. Sendo assim, as mensagens formam uma lista encadeada a medida que vão chegando, sendo lida em um esquema "a primeira a chegar é a primeira a ser atendida" (conhecida como *First in, First out*, ou FIFO) [57]. A Figura 2.5 ilustra o modelo, mostrando como é formado o grafo de tarefas e canais.

Tarefas, além de executar suas próprias instruções, podem realizar quatro operações básicas, e a partir delas realizar operações mais complexas. Elas são: enviar mensagens, receber mensagens, iniciar outras tarefas e terminar sua execução. A comunicação entre os sistemas podem ser mão única ou mão dupla, dependendo do vínculo que existe entre as tarefas. Suas execuções são assíncronas, uma vez que cada tarefa pode iniciar em tempos diferentes e ter instruções próprias. O envio de mensagens, similarmente, é assíncrono e não bloqueia a execução. Ao contrário, o recebimento bloqueia o fluxo de execução da tarefa até que haja alguma mensagem disponível no canal [55].

Propriedades pertinentes a esse modelo são desempenho, independência da arquitetura e modularidade. O desempenho é conquistada pelo fato das tarefas serem efetivamente máquinas de von Neumann virtuais, ou seja, por mais que cada uma possua um fluxo de instruções sequencial, elas podem ser executadas paralelamente, assim resolvendo múltiplas partes do problema ao mesmo tempo. A independência da arquitetura é possível graças ao mecanismo bloqueante de recebimento de mensagens. Se uma tarefa depende de outra e esta chegou ao ponto de dependência antes daquela, ela é bloqueada até que uma mensagem seja recebida. Assim, não importa a quantidade de processadores do computador que executa a aplicação, ela atingirá sua resposta independentemente. Por fim, a modularidade é alcançada pela natureza intrínseca das tarefas. Cada tarefa executa um código próprio e apenas percebe as entradas e saídas de outras tarefas. Dessa forma, modificar um código de uma tarefa não afeta o código das outras, tornando o sistema

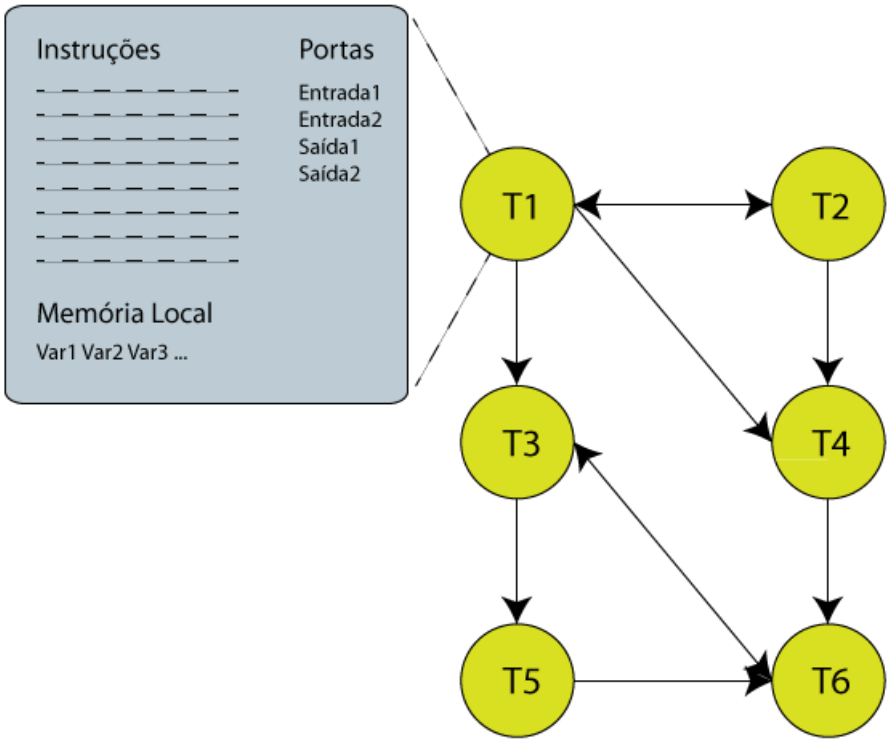



Figura 2.5: O modelo de tarefa-canal. Cada círculo significa uma tarefa e as setas significam um canal, em que o sentido mostra a origem e o destino das mensagens. A tarefa T1 foi mostrada mais especificamente no retângulo. Adaptado de [3]

reusável e modular [55].

### 2.3.2 Memória Compartilhada e Condição de Corrida

Por mais que um processador possua registradores para a execução de instruções, é impossível realizar a computação de aplicações sem memória. Na memória são feitas operações de armazenamento de valores e posição de linha de execução dos programas, sendo que esse procedimento é essencial para a troca de contexto nos processadores, permitindo assim ambiente multiprogramados [57].

Em programas paralelos, podemos ter tarefas com memória essencialmente local como sugere o modelo de tarefa-canal. No entanto, como deve ser feito o manejo de uma coleção de recursos comuns a várias tarefas simultâneas? A fim de se manter fiel ao modelo citado, pode-se criar uma tarefa que centraliza esses recursos para si e atende a requisições de outras tarefas quando necessário, recebendo e enviando mensagens. No entanto, essa tarefa pode ser um ponto de contenção do sistema, como uma retenção de processamento quando várias tarefas precisam de um recurso e esperam a resposta dessa tarefa centralizadora de recursos. Outra solução para o problema de gerenciar

tarefas simultâneas seria utilizar uma cópia das variáveis comuns em todas as tarefas e atualizá-las periodicamente. Não obstante, para recursos comuns muito grandes, seria um desperdício de memória manter várias cópias. Além disso, como uma tarefa poderia saber se a sua cópia está atualizada com as outras? Isso exigiria um canal de atualização conectado a todos os recursos de todas as tarefas, tornando o modelo demasiadamente complexo. Logo, não é uma resposta ideal para o problema [57].

Uma solução mais simples seria a adaptação do modelo de tarefa-canal para que a aplicação pudesse ter um espaço de memória compartilhado entre todas as tarefas [3]. Além das vantagens de não existir um controle central onde poderia haver uma retenção e existir apenas uma cópia dos recursos, economizando memória, é muito mais simples programar uma variável comum acessável por todas as tarefas pertinentes. No entanto, a administração de escritas e leituras dessas variáveis é comprometido pela assincronicidade das tarefas, uma vez que nenhuma conhece em que ponto a outra está e os acessos são aleatórios. Esse fenômeno é chamado de condição de corrida [57].

Não apenas memória compartilhada está sujeita a condições de corrida, mas qualquer recurso compartilhado sujeito à escrita por dois ou mais processos ou tarefas estão. Um exemplo simples de condição de corrida é ilustrado na Figura 2.6, em que possuímos a variável x na memória compartilhada, e é modificada pela Tarefa T1 e pela Tarefa T2 em pontos diferentes de seus códigos. Dessa forma, é impossível prever qual ser o valor de x impresso nas instruções seguintes.

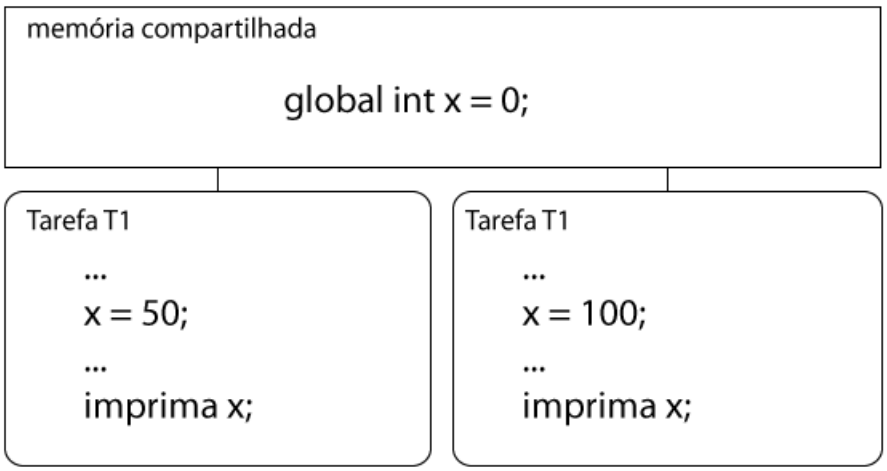



Figura 2.6: Um exemplo de condição de corrida.

### *Deadlocks*

Em ambiente multiprogramado as tarefas paralelas nem sempre estão sendo executadas. Isso porque com a limitação de quantidades de processadores executadas. Isso

porque com a limitação de quantidades de processadores disponíveis na máquina, as tarefas podem ser processadas por um intervalo de tempo, suspensas e colocadas em uma fila para serem continuadas no futuro, enquanto a próxima tarefa dessa fila é executada no processador. Essa situação é chamada de troca de contexto, e o intervalo de tempo é denominado *quantum* [57] Há potencialmente uma situação de *deadlock* quando um recurso é obtido por uma Tarefa A e uma Tarefa B também o necessita e não o obtém. Por exemplo, a tarefa A requisita acesso exclusivo a uma área de memória compartilhada M1 e a obtém. Seu *quantum* termina e ela é suspensa, sem liberar a memória alocada. A tarefa B precisa da mesma posição de memória e não a obtém, uma vez que a tarefa A ainda a está exclusivamente alocando. Sendo assim, a tarefa B não pode prosseguir sua execução até que a tarefa A libere o recurso necessário. No entanto, assim que A liberar M1, B, em uma oportunidade futura, pode utilizar o recurso. Portanto, nessa situação não há *deadlock*.

Considera-se agora que B requisita acesso exclusivo a uma área de memória compartilhada M2. Ela obtém M2 e ocorre seu *quantum*. A seguir, ocorre o descrito no parágrafo anterior, a tarefa A requisita e obtem exclusivamente M1 e seu *quantum* ocorre. B, ainda com M2 alocado, requisita M1 e não obtem e seu *quantum* ocorre novamente. Em seguida, A requisita M2 e não obtém, pois B está com acesso exclusivo. Essa situação então é efetivamente um *deadlock*. Nenhuma das tarefas pode continuar por uma estar em posse do recurso que a outra precisa, e nunca esse recurso será liberado por elas estarem sem andamento em sua execução.

Coffman et al. (1971 apud Tanenbaum) mostram que quatro condições são necessárias para que um *deadlock* ocorra. Ao identificá-las em uma situação, um potencial *deadlock* pode ocorrer.

1. Exclusão mútua: Cada recurso é obtido exclusivamente por um processo.
2. Múltipla obtenção de recursos(*Hold and Wait*): Um processo que já obtem um recurso exclusivamente pode requisitar e obter mais deles em diferentes momentos.
3. Recursos não-preemptivos: Não é possível retirar a força um recurso obtido exclusivamente.
4. Espera circular: É preciso haver uma espera de duas ou mais tarefas por recursos a serem liberados, com cada tarefa travando um recurso esperado por outra.

Tal situação pode ser evitada com o remanejamento dos recursos, refatorando o código das tarefas e utilizando técnicas que invalidem ao menos uma das condições citadas.

A exclusão mútua pode ser evitada adequando o código para que as tarefas possam acessar ao mesmo tempo um mesmo recurso. No entanto, isso pode ser proibitivo para

certas situações em que é fisicamente impossível o acesso mútuo, ou o prejuízo causado pela remoção de um recurso no meio da execução é inevitável. Por exemplo, é impossível remover o acesso exclusivo a uma gravadora de cds enquanto um disco estiver sendo escrito por uma tarefa.

Algumas estruturas são propostas para transformar o ato de múltipla obtenção de recursos em uma instrução exclusiva. A área do código em que as obtenções são feitas, denominada área crítica, é isolada por alguma variável ou chamada que faz com que uma tarefa entre sozinha nesse trecho crítico. Se outras tarefas encontrarem essas chamadas, elas esperam até que a tarefa saia da área crítica ou seja destruída. O importante é que todos os recursos sejam obtidos e liberados de uma só vez por apenas um processo. Independente do mecanismo escolhido para essa solução, é importante notar que as tarefas não podem tomar muito tempo com os recursos retidos, caso contrário pode-se criar um "gargalo"no sistema, em que outras tarefas são impedidas de executar para esperar pela liberação da área crítica. A escolha do próximo a entrar na área crítica também precisa ser feita de forma justa, sob a pena de outras tarefas ficarem esperando eternamente por um recurso sem estarem segurando nenhum outro. Esse fenômeno é conhecido como *starvation*.

Quanto a recursos não-preemptivos, a tomada de recursos a força é a solução encontrada para resolver um *deadlock*. No entanto, de forma parecida a resolução da exclusão mútua, pode ser impossível ou os resultados podem ser inadmissíveis.

Por fim, soluciona-se a espera circular por análise do grafo de dependência de recursos. Utilizando-se um algoritmo justo de distribuição de recursos, pode-se distribuir sequencialmente os recursos necessários pela tarefas, inclusive com o auxílio de mecanismos que evitam múltipla obtenção de recursos. A vantagem é que quando se libera os recursos de todas tarefas travadas e se redistribui justamente é que algumas delas podem voltar a executar paralelamente mesmo na zona crítica, uma vez que pode haver conjuntos de recursos disjuntos utilizados por tarefas diferentes. A despeito disso, no pior dos casos em que não há disjunção entre os recursos, o sistema se reduz momentaneamente a uma execução sequencial, mais lenta, porém segura.

## 2.4 Gestão Ambiental

A Gestão Ambiental, conjunto de processos de avaliação, análise e tomada de decisão no contexto ambiental, baseia-se no conceito de ecologia de paisagens [59]. Ecologia de paisagens é o estudo da estrutura, função e dinâmica de áreas heterogêneas compostas pela interação dos ecossistemas, e é motivada pela pesquisa das interações entre a sociedade humana e seu espaço de vida, natural ou antropizado, considerando os fatores culturais

dessas ações [60, 61]. Essa área de estudo, por mais que recente, estabelece-se de grande importância pela variedade de interações e consequências decorrentes entre homem e meio. Dentro desses estudos, são fontes de conhecimentos uma ampla gama de recursos, como imagens obtidas por sensoriamento remoto, dados de GIS, estudos em censo e crescimento populacional e simulações computacionais. Por causa disso, trata-se de uma área de conhecimento intrínseca ao crescimento do poder computacional, visto que o volume de dados a ser processado é muito grande [62].

Conceitos como *cobertura da terra*, *uso da terra* e *conversão da terra*, pertinentes a esse tema, devem ser diferenciados. *Cobertura da terra*, expressão derivada de *land coverage*, refere-se ao atributos da terra, tudo que está sobre ela ou imediatamente abaixo. Exemplos desses atributos são topografia, massa aquáticas e vegetação. Quanto ao *uso da terra*, proveniente da expressão *land usage*, trata-se do conjunto de atividades e práticas no espaço em questão. Exemplos desse conjunto são atividades agropecuárias, extração mineral e silvicultura. Esse conceito é intríseco ao de *conversão da terra*, de *land conversion*, que diz respeito à transição de usos da terra e sua dinâmica. Modelos podem ter sua regras criadas a partir de um conjunto de possíveis coberturas da terra em um espaço, assim como os usos e como eles são convertidos. A diferenciação conceitual afeta o entendimento da pesquisa e como regras de modelagem podem ser criadas a partir dos diferentes significados [21].

*Land use cover change* (LUCC) é a área em Gestão Ambiental e Ecologia de Paisagens que se interessa em determinar a dinâmica da terra. Transições entre os diferentes estados da terra possuem processos intrísecos bastante complexos e interligados quanto a seus atributos. Dessa forma, é dada especial atenção ao entendimento dos processos de transformação da cobertura da terra, e como um conjunto de variáveis pode interagir com os mais diversos usos no ato da conversão da terra [20].

Percebe-se então que é de grande importância o papel da sociedade na transformação do meio ambiente, uma vez que atividades socioeconômicas e atos políticos determinam o tipo de atividade desenvolvida em um espaço, consequentemente influenciando na dinâmica da transformação da terra. Faz-se necessário o estudo integrado entre as relações sociais, econômicas e gestoras de uma sociedade com o ambiente em que está inserido para o completo entendimento da terra para atividade de Gestão Ambiental [21]. No entanto, tais relações possuem grande complexidade, o que leva à necessidade da criação de modelos, equilibrados entre simples e complexos, para a obtenção de informações e resultados para a gestão. As diferentes metodologias de modelagem do meio ambiente são apresentados a seguir.

### 2.4.1 Modelo e Simulação

Modelagem é a atividade de consignação do conhecimento teórico e prático em um modelo que represente um fenômeno real, enquanto um modelo é um construto abstrato que reúne os atributos necessários para representar um fenômeno real [63]. Os próprios atributos descritivos do modelo limitam sua capacidade representativa, e cabe ao especialista considerar o equilíbrio entre simplicidade (para aumentar o entendimento) e completude (para capturar o máximo de variáveis que influenciem o fenômeno, de forma explicíta ou não) [63]. Cabe ressaltar que, para a prática da Gestão Ambiental e para este trabalho, o modelo aqui referido é o modelo científico, ou seja, aquele que considera uma metodologia científica na sua concepção e validação [22].

Dentre os vários tipos de modelos, Novaes [22] faz um agrupamento conforme o uso:

1. Modelo descritivo - visa a caracterização, de forma geral, do fenômeno;
2. Modelo exploratório - busca, por meio dos atributos e subfenômenos capturados, fazer uma descrição lógica entre os elementos e suas relações;
3. Modelo preditivo - por meio da captura de eventos temporais de um fenômeno busca fazer predições sobre futuros eventos no conjunto, por meio de algum tipo de inferência ou mecanismo de classificação (como um teorema);
4. Modelo operacional ou de controle - uso de atributos variáveis pelo especialista para a reprodução do fenômeno. Também pode usar dessa variação para a tentativa de reproduzir eventos hipotéticos.

Quanto a natureza do modelo, ele pode ser de três tipos [64]:

1. Modelos mentais - heurísticos, associados a intuitividade do fenômeno;
2. Modelos físicos - são representados de forma análoga ao fenômeno;
3. Modelos simbólicos - são aqueles descritos por uma linguagem, representando conceitualmente o fênomeno.

As simulações, a partir de um modelo, tentam reconstituir o fenômeno de forma controlada, em determinado período de tempo [65]. As simulações computacionais são aplicações que descrevem um conjunto de instruções em um determinado período temporal, processadas a partir de entradas de dados e devolvendo resultados após um determinado número de execuções.

### Modelos Baseados em Indivíduos

Os Modelos Baseados em Indivíduos (MBI), também conhecidos como Modelagem Orientada a Agentes, é um técnica para a descrição de fenômenos naturais complexos bastante popular. Um MBI descreve um sistema dinâmico representado como uma coleção de comunicações, objetos concorrentes denominados indivíduos, agentes e outras entidades. Por mais que cada objeto individual possua um modelo interno descritivo finito, sua composição com outros objetos e suas interações são capazes de criar situações complexas [66]. Devido a demanda de validação de políticas públicas e de práticas de manejo ambiental, MBI se tornou uma técnica investigatória popular em LUCC, reunindo vários conceitos de Ecologia de Paisagem e outras disciplinas, como a Economia, Sociologia e a Computação [66].

Chli e de Wilde [67] indicam que MBI é um método para o estudo de sistema com duas propriedas aparentes: o sistema é composto por agentes interativos e; o sistema apresenta propriedades emergentes, ou seja, propriedades que emergem da interação dos agentes entre si e entre o ambiente. Análise matemática é quase sempre pouco eficiente nesse contexto, por experiências consolidadas em MBI e suas consequências geradas dinamicamente. Sendo assim, apenas métodos práticos de análise são passíveis de resultados coerentes.

Tesfatsion [68] cita quatro objetivos específicos pertinentes aos pesquisadores em MBI . São eles:

- Empirismo: Explicações causais são de interesse dos pesquisadores para fenômenos ocorridos da interação de agentes em ambientes específicos. Em particular, é questionado se tipos particulares de interação produzem consequências distintas, comparando experimentos com interações padrões em um MBI. Por exemplo, pode ser hipotetizado se ao se modificar um comportamento de agente para que ele aja de forma específica sobre um determinado tipo de aspecto do ambiente, resultados distintos serão percebidos.

- Normatividade: Procura-se conhecer o estado de equilíbrio do cenário em que um MBI trabalha após serem inseridas políticas públicas projetadas, ou seja, se, após a inserção de um conjunto de regras, a sociedade, o ambiente e outras entidades chegarão a um estado considerado desejável pelos pesquisadores.

- Uso de Heurísticas: Em face a um conjunto de variáveis iniciais, de que forma os agentes de um MBI alcançarão um objetivo geral desejável? A obtenção de uma heurística ótima é fruto de exaustiva experimentação e adaptação das regras, e pode ter soluções não previstas dependendo da complexidade do sistema estudado.

- Uso de Metodologias: O caráter empírico dos MBI não excluem o uso de metodologias de trabalho, ao contrário, exigem igual rigor científico em suas iterações de etapa e experimentações. A metodologia aplicada em um estudo de um MBI não é apenas parte do processo investigativo, mas um dos produtos do trabalho, uma vez que a sua conquista pode ser aplicada em outros cenários a serem propostos.

A maior desvantagem percebida em MBI em comparação a Modelagem Matemática e Resolução se refere a validação do modelo, uma vez que não são aplicáveis teoremas que assegurem a robustez dos resultados obtidos. Para ser alcançado um conjunto confiável de resultados, um número grande de experimentos e ajustes deve ser praticado, e metodologias específicas devem ser aplicadas para a avaliação e análise dos dados obtidos. Allan [69] cita uma série de desafios a serem enfrentados em MBI em ordem de extender as capacidades de MBI a outras áreas, popularizando o método fora de LUCC em outras ciências tecnológicas.

- Resolver dificuldades encontradas na execução de aplicações muito grandes ou complexas;
- Melhorar o desempenho das aplicações que utilizam MBI, seja por distribuição ou por aplicação de algoritmos paralelos eficientes;
- Propor novos métodos mais gerais de validação e verificação de resultados, especialmente os de larga-escala;
- Utilizar representações padrões para os modelos, para que código computacionais gerados sejam equivalentes em linguagens diferentes;
- Prever novos usos em diferentes áreas científicas para MBI;
- Construir projetos computacionais colaborativos em MBI para que as melhores práticas sejam aproveitadas por um maior número de aplicações.

A própria terminologia sugere que a modelagem baseada em individualidade se mistura com simulação, especialmente a em SMA. Outros aspectos reenforçam essa união de conceitos, como a presença de um ambiente dinâmico, as consequências emergentes das interações entre os agentes e a racionalidade, quando presente, se utilizando de raciocínio prático para a resolução de problemas.

Apesar de modelos matemáticos e teoremas não se aplicarem de forma eficiente na avaliação de um MBI, há formas de se mensurar a sua eficiência por meio de outras métricas. Pontius et al. [70] propõem a mensuração da Figura de Mérito para modelos em simulação.

A Figura de Mérito é representada por uma pontuação de 0 a 100, e é considerada como muito boa se tiver um valor igual a 50 ou maior, ou seja, é um resultado de um modelo acurado. Ela pode ser utilizada em cenários de simulação ambiental em que se possui a representação do estado inicial real em que o modelo se baseia (imagem A), a representação do estado final real (imagem B), e o resultado que MBI previu após uma simulação ter sido executada (imagem C). Primeiramente, são comparadas as imagens A e B, as representações reais, para se mapear onde houve conversão da terra e permanência de estado, gerando uma imagem D. A imagem D é então comparada com a imagem C, produzida sinteticamente pelo simulador, a fim de se conhecer onde o MBI acertou e errou suas predições. A partir disso, quatro tipos de situações são produzidas: Q1 é quantidade de conversão da terra que foi predita corretamente; Q2 é quantidade de permanência de estado predita corretamente; Q3 é a quantidade de permanência predita erroneamente; e Q4 é a quantidade de conversão predita erroneamente. Esse método de comparação é definido como *three-map comparison*, uma vez que utiliza três mapas para medir quantos erros e acertos efetivos foram realizados.

Uma vez definidas as variáveis Q1, Q2, Q3 e Q4, a Figura de Mérito é calculada como a Q1*100/(Q1+Q3+Q4). Dessa forma, se Q3 e Q4 forem nulos, ou seja, o MBI for ideal e consegue prever todas as conversões e permanências do ambiente, a Figura de Mérito é máxima e igual a 100.

### 2.4.2 Ferramentas baseadas em MBI

A título de comparação, são revisados vários *softwares* e aplicações computacionais que se utilizam em algum nível de modelagem baseada em individualidade. Uma análise mais completa pode ser encontrada na literatura, em [71], [72], [69], [73], [74], [75] e [76].

#### Ascape

Ascape [10] [77] é uma biblioteca aberta para desenvolvimento e exploração de modelos baseados em agentes de forma geral, desenvolvida por Miles Parker em *Brookings Institute Center on Social and Economics Dynamics*. Ascape suporta o projeto de modelos complexos, graças ao apoio por ferramentas direcionadas ao usuário para aqueles que não tem conhecimento profundo em programação aproveitarem toda a potencialidade dos modelos dinâmicos gerados. Essa biblioteca foi desenvolvida em Java e direcionada para a execução em ambientes que suportam a máquina virtual Java, além dos modelos gerados possuírem a possibilidade de serem gravados em vídeo quando executados, ou exportados para a publicação na Web.

[10] http://ascape.sourceforge.net/

A representação espacial do modelo em Ascape é uma distribuição em grid do ambiente formando células. Regras abstratas definidas pelo usuário controlam o comportamento dos agentes que ocupam os espaços. Essa biblioteca possui um fórum de discussão e manuais técnicos para o auxílio no processo de geração do modelo. É notável que grande parte dos modelos criados no Ascape são pertinentes a modelagem em Economia e Mercadologia.

### Cormas: COmmon-Pool and Multi-Agent System

Cormas[11], desenvolvido pela *French Agricultural Research Center for International Development* (CIRAD), é um SMA focado em simulação para o manejo de recursos naturais. Cormas utiliza a plataforma *VisualWorks* para o desenvolvimento, execução e depuração de seus modelos, além de possuir uma biblioteca com vários artefatos prontos e exemplos de situações comuns encontradas em MBI. São definidas três entidades genéricas principais para serem usadas pelo usuário, extensíveis utilizando a linguagem de programação Smalltalk, a saber:

- Agentes Espaciais: unidades de células decorrentes da divisão do ambiente por um grid. Suas ações são mapeadas a eventos externos e internos fixos, como por exemplo, ocupação física de sua representação por outro agente (evento externo) ou mudança de estado (evento interno).
- Agentes Comunicantes:entidades exclusivamente utilizadas para a comunicação por mensagens entre agentes.
- Agentes Espaciais-Comunicantes: uma mescla dos dois agentes anteriores, com capacidades adicionais de movimentação pelo espaço.

Cormas trabalha com passos de simulação, em que cada passos seus agentes realizam suas ações de forma sequencial e atualizam seus estados. Não há nenhum aspecto de raciocínio nos agentes além de suas ações mapeadas em eventos.

### Eco Lab

Eco Lab [12] [78] é uma biblioteca para programação de agentes de uso geral, escrita em C++ e TCL/Tk. Ela provê uma série de funcionalidades interessantes à execução, como visualização individualizada de agentes, exportação de agentes entre plataformas, , suporte a ambientes distribuídos e ferramentas de fácil inicialização do sistema. Originalmente desenvolvido para simular modelos particulares em Ecologia, Eco Lab ampliou seu domínio mostrando que modelos gerais poderiam ser executados com suas funcionalidades. Essa

[11] http://cormas.cirad.fr/
[12] http://ecolab.sourceforge.net/

biblioteca foi escrita em C++ e Objective-C, desenvolvida e mantida pela *Russell Standish* na Universidade de Sydney, Austrália. Eco Lab é de código livre e existem documentações diversas para o auxílio de seu uso.

### MASON: Multi-Agent Simulation of Neighbourhoods

MASON[13] é uma biblioteca de código livre e aberto, para modelagem e desenvolvimento de agentes em Java, facilmente integrável com outras bibliotecas, inclusive em outras linguagens [79]. Ela foi produzida com o esforço conjunto do Departamento de Ciência da Computação e do Departamento de Complexidade Social, ambos pertencentes ao Centro Universitário *George Mason*, na cidade de *Fairfax*, Virgínia, Estados Unidos. MASON tem como intenção a melhoria de performance e espaço de outra ferramenta, RePast, com o desafio de incluir modelos com muitos agentes e interações diversas. O principal foco dos modelos gerados é a simulação em Robótica de Enxame, área de Automâtos Eletrônicos que lida com uma quantidade grande de robôs em um ambiente trabalhando conjuntamente para conquistar um objetivo comum, utilizando técnicas de aprendizado de máquina, comunicação de percepções e deliberação em conjunto. MASON está atualmente em sua décima-sétima edição e possui vários módulos computacionais opcionais que oferecem funcionalidades como visualização 2D e 3D, geração de vídeos em Quicktime e produção de estatísticas da simulação.

### MASS: Multi-agent Simulation Suite

MASS[14] [80] consiste de quatro aplicações conjuntas oferencendo soluções para diferentes aspectos em modelagem. O *Functional Agent Based Language for Simulation* (FABLES) é um linguagem de programação fácil de usar para a implementação dos modelos. *Charting Package*(CP) é um pacote de gráficos dinâmicos para a visualização das estatísticas da simulação. *Model Exploration Module* (MEME) é um módulo que habilita a avaliação e análise dos resultados obtidos. Por fim, *Participatory Extension* (PET) é um ambiente integrado em formato de aplicação Web, disponível para modelagem, administração e implantação de MBI. Todas as quatro funcionalidades são livres para uso, porém limitadas em suas funcionalidades - ferramentas mais específicas do MASS podem ser compradas e automaticamente integradas. Os modelos do MASS são tipicamente direcionados a fenômenos em Ciências Sociais, especialmente aqueles ligados à Economia e à Sociologia.

[13] http://cs.gmu.edu/~eclab/projects/mason/
[14] http://mass.aitia.ai/

### NetLogo

NetLogo[15], um ambiente programável para modelagem e simulação de fenômenos ambientais e sociais, é uma das ferramentas para modelos baseados em agentes mais popular da atualidade. Produzida originalmente por Wilensky [81], ela vem sendo desenvolvida pelo Centro de Aprendizado Conectado e Modelagem Computacional da *Northwestern University*, em *Evanston*, Illinois, Estados Unidos. Usuários podem dar instruções a milhares de agentes independentes, todos agindo concorrentemente, e explorar tanto os comportamentos em nível individual quanto os padrões emergentes gerados da ação de vários agentes em um mesmo domínio. O NetLogo possui uma interface gráfica para construção de modelos e outras ferramentas integradas, além da possibilidade de criação e publicação de extensões pelos próprios usuários. A ferramenta oficialmente oferece mapeamento direto de eventos e ações dos agentes, similar ao Cormas, mas extensões de usuário podem adicionar formas mais complexas de raciocínio, como o modelo BDI e o aprendizado por redes neurais. A linguagem de programação do NetLogo é baseada em Logo, um dialeto de Lisp, que é de fácil uso e permite a prototipação rápida de modelos. Seu código é livre e aberto, e existe extensiva documentação a respeito do ambiente, incluindo fóruns de discussão e uma grande comunidade ativa de usuários.

### RePast: Recursive Porous Agent Simulation Toolkit

RePast é uma ferramenta de modelagem para a criação de agentes em Ciências Sociais [82]. Com uma biblioteca para criar agentes, executá-los, visualizá-los e coletar dados, ele utiliza uma abordagem de "enxame", similar ao MASON, em que os agentes são pequenas entidades de um conjunto maior em busca de um objetivo comum. Essa ferramente é livre e de código aberto, e foi desenvolvida pelo *Argonne National Laboratory*, operado pela Universidade de Chicago, nos Estados Unidos. Sua versão mais atual, a Repast Simphony, possui ferramentas visuais internas para o desenvolvimento do modelo, execução, conexão com banco de dados, registro de ocorrências e obtenção de resultados [83]. Sua implementação é em Java, mas os modelos podem ser criados tanto visualmente quanto em linha de código.

### TerraME

TerraME[16] é um conjunto de ferramentas livres que oferece suporte a múltiplos paradigmas em um ambiente multiescalar na modelagem das interações homem-ambiente e suas decorrências[84]. Ele permite a composição de abordagens, como automatos celulares

[15] http://ccl.northwestern.edu/netlogo/
[16] http://www.terrame.org/

com SMA, sendo bastante flexível, dando liberdade ao usuário para a criação de modelos que possuem aspectos mais fortes em determinados campos do conhecimento. TerraME também conta com ferramentas de integração com Dados Georreferenciados e utiliza Lua em sua programação, uma linguagem simples, porém bastante extensível e abrangente. O usuário pode informar as características comportamentais, temporais e espaciais do modelo separadamente, facilitando projetos de larga-escala, um dos grandes desafios para MBI. Na especificação dos comportamentos, são permitidos apenas mapeamentos diretos de eventos-ação, tornando todos os agentes da simulação exclusivamente reativos.

TerraMe se destaca por ter sido desenvolvido no Brasil, pelo Instituto Nacional de Pesquisas Espaciais (INPE), em São José dos Campos, São Paulo. Ele é de código livre e aberto, e utilizado amplamente em pesquisas INPE relacionados a interações entre o homem e o ambiente natural e seus impactos.

# Capítulo 3

# Proposta de Solução

O MASE[1], proposto em 2012, possui um histórico de protótipos anterior a ele, um construído no CORMAS (Seção 2.4.2 e outro em JADE (Seções 2.1.6 e A.1). Em ambos protótipos utilizou-se um MBI denominado *ReasonBalance* que descrevia a exploração antrópica na reserva do Uacari, Amazônia. No *ReasonBalance* os agentes cumpriam o papel de agricultores explorando as áreas da reserva em diferentes cenários de consciência ambiental, trazendo diferentes níveis de impacto ambiental ao longo de um período de tempo predeterminado. O detalhamento desses protótipos e do *ReasonBalance*, além dos resultados obtidos e analisados se encontram em [85],[13] e [17].

A utilização do CORMAS, uma ferramenta bem estabelecida na criação e execução de MBI, e do JADE, um *framework* bem documentado e amplamente utilizado em pesquisas, trouxe conhecimentos tão importantes quanto os levantados na fundamentação teórica do MASE. Foi possível identificar que funcionalidades eram essenciais e desejáveis para o desenvolvimento da aplicação, além de listar quais as limitações poderiam ser melhoradas. A partir dessa análise, construiu-se o MASE, com sua primeira versão em 2012. A proposta do MASE é apresentada a seguir, na Seção 3.1.

Pontos de melhoria foram identificados no MASE, especialmente na racionalidade dos agentes e no desempenho de tempo das simulações. Esses tópicos são motivadores da reformulação da arquitetura do MASE, assim como a inclusão do modelo BDI. MASE-BDI, a aplicação sucessora do MASE, tem sua arquitetura e agentes apresentados na Seção 3.2.

## 3.1 Breve descrição do MASE

MASE é concebido como uma ferramenta de para simulações baseadas em MBI utilizando SMA para o entendimento e compreensão de cenários de uso e transformação

[1] http://mase.cic.unb.br/

de cenários ambientais. Diferentementemente das ferramentas levantadas na Seção 2.4.2, não é possível criar modelos diretamente nele, mas sim adicionar a ele entradas de regras, espaço e atributos espaciais por meio de arquivos descritores e imagens para que uma simulação seja executada e resultados possam ser obtidos (Figura 3.1). Não obstante, MASE apresenta uma interface gráfica para a visualização e interação com os descritores do modelo e com a própria simulação. Além disso, MASE apresenta uma arquitetura interna própria e consolidada de agentes em uma estrutura hierárquica, assim como um fluxo de funcionamento e regras internas próprias.

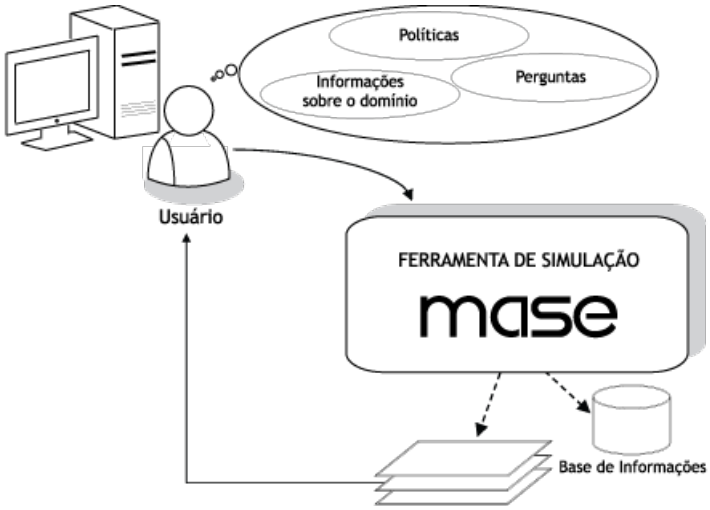



Figura 3.1: A Visão Geral da Simulação

MASE foi modelado a partir de requisitos definidos por especialistas em Ecologia e experiências realizadas com o CORMAS e com um protótipo em JADE. Sua estrutura então foi implementada em Java, utilizando o *framework* JADE (Seção A.1) para a instanciação de seus agentes. O *software* MATLAB[2] foi integrado à ferramenta inicialmente para suprir as necessidade de leitura, escrita, filtragem e classificação de imagens, mas por possuir uma licença proprietária, foi substituído pela biblioteca de manipulação de imagens IJ[3], escrita em Java e de código aberto e livre, assim fazendo com que o MASE pudesse ser livremente distribuído.

A arquitetura do MASE é definida em camadas, conforme mostra a Figura 3.2. A mais acima é a interface gráfica, que possibilita o controle do usuário com a aplicação. A ela está ligada a configuração, já citada em detalhes. Em seguida, as setas representam o interfaceamento entre os elementos. O Agente Janela é um *listener* e o responsável da interface gráfica, ou seja, ele percebe todas as mudanças inseridas pelo usuário (como o

[2] http://www.matlab.com
[3] http://http://rsbweb.nih.gov/ij/

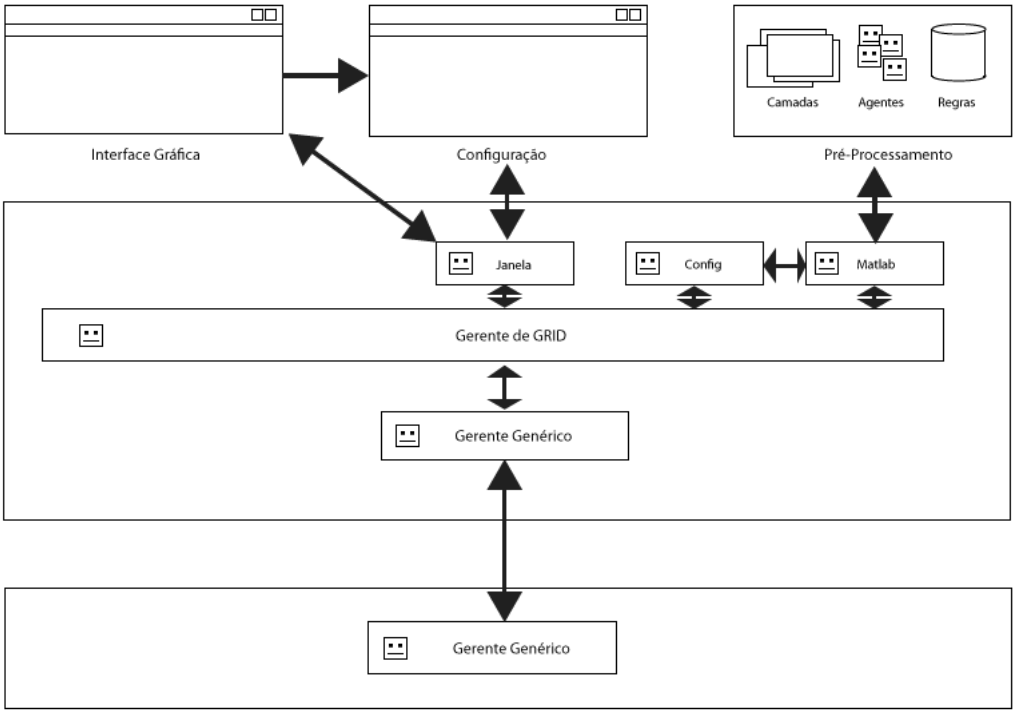



Figura 3.2: A arquitetura do MASE

salvar na janela de configuração, que ativa o Agente Configuração), promove um tratamento e informa ao Gerente de GRID qual ação tomar.

O Gerente de GRID, uma camada abaixo, é o centro da aplicação, e promove a conversão de todas as requisições, além de ativar agentes e enviar ordens. Os agentes de configuração e interfaceamento com o MATLAB ou IJ já foram visitados nesse estudo. Quanto ao Gerente Genérico, enquanto não especificado, é incapaz de realizar qualquer tarefa, como seu subordinado, o Agente Genérico, já na camada inferior. Eles podem ter inúmeras instâncias, desde que não existam gerentes sem subordinados e agentes sem superiores. Além disso, um Gerente Genérico não pode ser subordinado a outro gerente genérico, a ligação entre gerentes e agentes é uma hierarquia terminal.

As setas são todas bidirecionais (com exceção da ligação da interface gráfica a configuração). Isso representa o caminho de duas mãos entre os agentes, uma vez que as ordens e ativações são dos gerentes para os agentes, enquanto os avisos e informações são na direção contrária. Para facilidade de comunicação, os gerentes sempre tem os AID (*Agent Information Data*) dos seus subordinados, que contém seus endereços, e os agentes são instanciados já conhecendo o AID de seus superiores. Além disso, como Agente de Configuração, Agente MATLAB, Agente Janela e Gerente de GRID pertencem à Gerência Global, eles tem seus AID registrados no DF (*Directory Facilitator*).

O DF, por sua vez, é um mecanismo de páginas amarelas provido pelo JADE. Agentes que desejam ter seus serviços disponibilizados se registram nele para serem facilmente encontrados. No entanto, devido a essa facilidade de comunicação, um problema pode surgir com o grande volume de agentes instanciado: a sobrecarga de mensagens sendo trocadas. Para amenizar esse problema, as mensagens passaram a ter um caráter reduzido. Grande mensagens e objetos só são passados em casos de extrema necessidade. Em outros casos, o assunto da mensagem e o rementente já diz o suficiente para ser tratada, ou seja, mais sintética possível.

Detalhamentos mais profundos a respeito da arquitetura do MASE, com exemplos do fluxo de funcionamento podem ser encontrado em [17], [14] e [15].

## 3.2 MASE-BDI

Considerando-se novamente os Algoritmos 4 e 5, apresentados na Seção 2.1.3 e observa-se que, na função agente que eles definem, existem subfunções que consideram parâmetros como percepções, estado e objetivos. Sob a perspectiva do modelo BDI, estados e percepções podem ser simplificados em crenças: aquilo que o agente crê que é verdadeiro em um momento específico. Quanto aos objetivos, eles podem ser desejos formados em conjuntos válidos. A próxima ação, retorno dessa função, é derivada das intenções do agente. Sendo assim, faz sentido aplicar o modelo BDI a agentes orientados a objetivos e utilidades.

No entanto, analisando os agente do MASE, percebe-se que a maior parte da deliberação e tomada de decisões é feita pelos Gerentes. Isso causa um ponto de contenção no sistema, além de subutilização das capacidades outros agentes. Por isso, é proposta uma nova arquitetura, que divide melhor a tomada de decisão e distribui a carga computacional entre agentes da simulação.

Para a implementação do MASE-BDI utilizou-se Java com o *framework* JADEX, que adiciona uma camada abstrata ao JADE para a representação do modelo BDI e outras funcionalidades (Seções 2.1.6 e A.2). Para o processamento de imagens, continuou-se utilizando a biblioteca ImageJ, mas de uma forma mais modularizada. Em vez da criação dos objetos da biblioteca em pontos do código, criou-se uma classe separada que realiza as chamadas à biblioteca, e a utilização dela para que as operações com imagens fossem realizadas.

### 3.2.1 Arquitetura Geral

A arquitetura do MASE-BDI é mais sintética que a do seu predecessor, uma vez que possui menos tipos de agentes, porém mais concisa e melhor distribuída, por possuir a tomada de decisão descentralizada (Figura 3.3). A principal mudança é a inserção

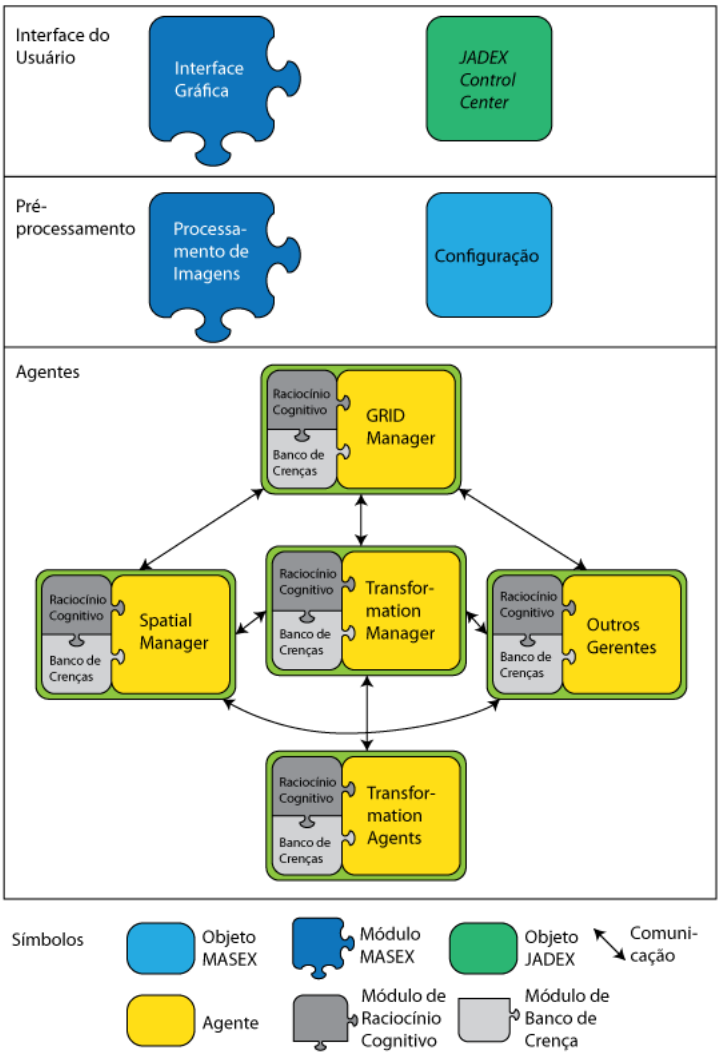



Figura 3.3: Arquitetura proposta para o MASE-BDI

explícita do módulo de racionalidade, no caso, do modelo BDI fornecido pelo JADEX no MASE-BDI. É representado por um módulo de raciocínio cognitivo e um banco de crenças. O módulo é o conjunto de mecanismos a serem usados para que uma ação seja tomada, e o banco é um conjunto de variáveis provenientes do objeto de configuração, da camada de pré-processamento e provenientes de eventos externos e outras percepções do agente.

De forma similar ao MASE, a arquitetura do MASEX três camadas, porém mudanças em seu conteúdo são visivelmente percebidas. A camada de interface com o usuário possui um módulo de interface gráfica e um objeto pertencente ao JADEX. A interface gráfica foi desenvolvida com a biblioteca Processing[4], construída em Java e facilmente utilizável,

[4] https://www.processing.org/

apenas para a demonstração da simplicidade de se desenvolver um módulo de visualização do usuário. O Processing é capaz de realizar chamadas explícitas a outras camadas do MASE-BDI, além de monitorar mudanças em variáveis específicas, dessa forma atualizando sua visualização para que o usuário possa interagir com a configuração do modelo e ver a simulação ser executada. Esse módulo pode ser completamente desativado, para o caso de inicialização e execução do MASE-BDI por console, em que se deseja apenas ver o resultado final. Utilizando-se as chamadas definidas na aplicação, também é possível integrar o MASE-BDI a outras interfaces gráficas. Quanto ao objeto *JADEX Control Center* (JCC), ele é utilizado principalmente para a depuração do modelo em modo desenvolvedor, não intencionado a ser usado pelo usuário final. Por meio dele é possível ver os agentes na plataforma JADEX, assim como outros agentes e componentes do *framework*, criar e destruir componentes manualmente, enviar mensagens de teste, dentre outras funcionalidades importantes para o desenvolvimento e teste do sistema.

A camada de pré-processamento inclui um módulo de processamento de imagens, que de forma similar ao MASE, é usado para o tratamento das imagens de entrada no MASE-BDI. A camada de agentes é a única que possui agentes implementados, ao contrário do MASE, que possuía agentes em todas as camadas. Além disso, o número de tipos de agentes é menor se comparado com a arquitetura predecessora (apresentada na Figura 3.2). A agentes gerentes possuem um novo tipo, *Others Managers*, criado com a intenção de criar uma solução genérica para o interfaceamento ativo com outras tecnologias, explicitamente deixando abertas as possibilidades de expansão do MASE-BDI. Os agentes de pré-processamento e interface gráfica (Janela) não existem mais por não possuírem real propósito, uma vez que suas atuações eram pontuais e determinadas pelo Gerente de GRID (GGRID no MASE, GRID Manager no MASE-BDI), e suas funções foram incorporadas pelas duas camadas anteriores. Os gerentes de célula também foram removidos, por apenas representar estados do ambiente individualmente e serem facilmente substituídos por uma matriz, administrada pelo Gerente Espacial (Spatial Manager). Os agentes de transformação (Transformation Agents) são mantidos, e a interação com o Gerente de GRID é instaurada para agilizar processos que requisitam comunicação *bottom-up* e não precisam passar pelo Gerente de Transformação.

### 3.2.2 Agentes do MASE-BDI

Existem cinco tipos de agentes no MASE-BDI (Figura 3.3), a saber: *GRID Manager* (GRIDM), *Spatial Manager* (SPM), *Transformation Manager* (TRM), *Transformation Agent* (TRA) e *Other Managers*. *Other Managers* é um tipo de gerente genérico, deixado sem funcionalidades, desejos, objetivos ou planos de ação intencionalmente a princípio,

com o propósito de ser o responsável por integrar o MASE-BDI à propostas futuras. Quanto aos outro quatro tipos de agentes, eles são descritos a seguir.

### GRIDM

GRIDM é cumpre o papel de agentes coordenador da simulação. Suas funções principais são instanciar gerentes, coordenar os passos (*steps*) da simulação, e terminar ou reiniciar a simulação. De fato, é o primeiro e único a ser instanciado junto com a plataforma JADEX, logo após o pré-processamento das variáveis do MASE-BDI, e se encarrega de construir a hierarquia do SMA.

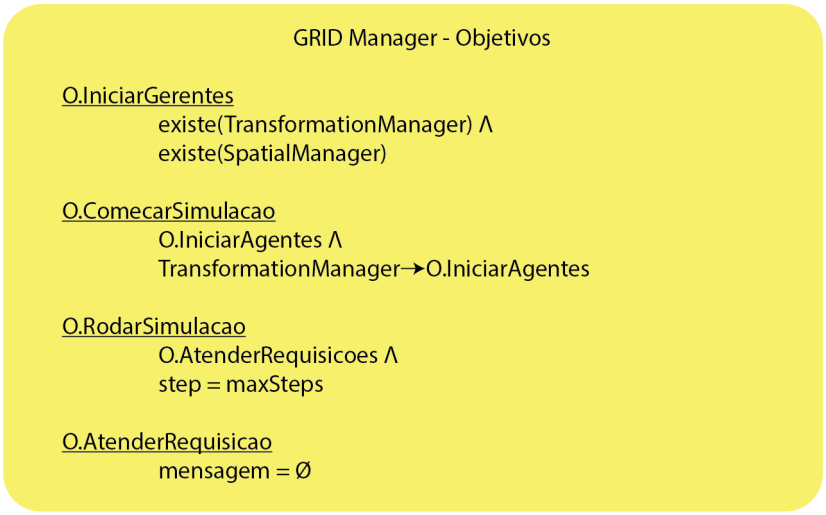



Figura 3.4: Objetivos do GRIDM. Adaptado de [4].

A Figura 3.4 mostra os quatro objetivos definidos para o GRIDM, em ordem de prioridade.O primeiro objetivo, *IniciarGerentes*, é alcançar o estado do ambiente em que existem os Gerentes TRM e SPM. O próximo objetivo, *ComecarSimulacao*, é alcançando quando o primeiro objetivo tem sucesso e o objetivo do TRM *IniciarAgentes* tem sucesso. De acordo com os tipos diferentes de objetivos do JADEX descritos na Seção A.2.3, os dois primeiros objetivos do GRIDM são do tipo "a ser alcançado" e a falha delas implica na falha da simulação inteira. Em seguida, o próximo objetivo do GRIDM é *RodarSimulacao*, que é alcançado quando o subobjetivo *AtenderRequisicao* tem sucesso e a quantidade de passos da simulação alcança o limite definido pelo usuário (por meio da variável *maxSteps*. Se esse objetivo falhar, ele é tentado novamente até ser alcançado, ou seja, é um objetivo a ser mantido, um estado de ambiente que o gerente deseja que seja permanecido. Por fim, seu último objetivo é *AtenderRequisicao*, que consiste em manter sua caixa de mensagens vazia. Quando uma nova mensagem chega, o objetivo falha, mas é tentado continuamente.

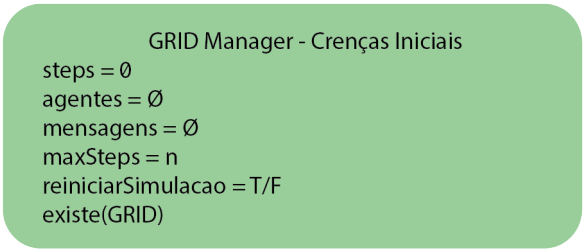



Figura 3.5: Crenças do GRIDM. Adaptado de [4].

As crenças iniciais do GRIDManager são mostradas na Figura 3.5. As crenças *maxSteps* e *reiniciarSimulacao* são definidas nas regras determinadas pelo usuário, e dizem respeito a quantidade de passos da simulação e se ao final o GRIDM deve reiniciar ou finalizar o cenário simulado. A crença *existe(GRID)* é pré-processada antes da plataforma iniciar e inserida no agente por objetos do MASE-BDI. As outras crenças, *steps*, *agentes* e *mensagens* são percebidas e atualizadas regularmente, e representam o passo atual em que a simulação se encontra, quais agentes estão ativos na plataforma e quantas mensagens existem na caixa do GRIDM.

Os planos de ação são representados na Figura 3.6. Não existe prioridades entre eles, mas todos exigem pré-condições para serem executados. O plano *InstanciarGerentes* é executado quando não existe nenhum dos dois gerentes ativos na plataforma. Suas ações adicionam os TRM e SPM à plataforma, e completam o objetivo *IniciarGerentes*. Quanto ao plano *AtenderRequisicoes*, sempre que uma ou mais mensagens são recebidas, ele é ativado para que possa receber informações e responder requisições, se necessárias. Esse plano é reativado enquanto a caixa de mensagens não estiver vazia, sempre eliminando a última mensagem recebida. O plano *Step* é ativado apenas se a plataforma já possui agentes, se todos eles trabalharam no passo de simulação atual e o passo final da simulação já não foi atingido. Ele aumenta a quantidade na variável *step*, permitindo que os agentes possam agir mais uma vez ciclicamente, até que se atinja a quantidade de passos definida pelo usuário. Os dois planos finais, *ReiniciarSimulacao* e *TerminarSimulacao*, exigem que a quantidade máxima de passos seja atingida, não haja mensagem na caixa de entrada do GRIDM e todos os agentes tenham trabalhado. No entanto, para que *ReiniciarSimulacao* seja ativado, a crença *reniciarSimulacao* deve ter o valor de verdade Verdadeiro (V). Dessa forma, todas as variáveis da simulação são removidas e os agentes são destruídos, para então serem restaurados no estado inicial. Caso *reniciarSimulacao* tenha valor de verdade Falso, *TerminarSimulacao* é ativado, a plataforma JADEX é encerrada e o MASE-BDI finaliza sua execução.

### TRM

O TRM é o agente gerente responsável por instanciar os TRA da simulação e resolver conflitos espaciais. Ao iniciar os agentes transformadores, ele indica onde devem se posicionar. Além disso, na ocasião em que dois agentes transformadores entram em conflito pelo mesmo espaço, o TRM se encarrega de assignar a cada um deles uma posição específica. Os objetivos do TRM estão especificados na Figura 3.7. O objetivo *IniciarAgentes* é satisfeito quando existem agentes e existem suas respectivas posições. Quanto ao objetivo de resolução de conflitos, *ResolverConflitos*, ele é alcançado os TRA conseguem trabalhar depois da resolução do conflito espacial entre eles. Por fim, o objetivo *AtenderRequisicao* se assemelha ao objetivo do GRIDM. É contínuo e procura manter sua caixa de mensagens vazia.

As crenças iniciais do TRM (Figura 3.8) em cinco variáveis, a saber: os agentes presentes na plataforma, as suas respectivas posições, as mensagens na caixa de entrada, a existência de conflito e a quantidade de agentes TRA. Essa última crença é definida pelo usuário, que decide quantos agentes devem ser instanciados na simulação. As demais crenças são atualizadas de acordo com mensagens recebidas e outros eventos percebidos no ambiente.

Os planos de ação do TRM são mostrados na Figura 3.9. Os dois primeiros, *InstanciarAgentes* e *ObterPosicoes*, são executados para cumprir o objetivo *IniciarAgentes*. Suas pré-condições são a ausência de agentes e posições, refletindo a situação das crenças iniciais do TRM, e intencionam obtê-los em suas ações. O plano *ResolverConflitos* é ativado sempre que é percebido pelo TRM que uma situação de conflito entre dois ou mais agentes existe. Nesse caso, suas ações são direcionadas a resolver este conflito existente. Por fim, o plano *AtenderRequisicao* se assemelha ao plano do GRIDM, utilizado para receber mensagens e tratá-las, atualizando crenças ou executando outras ações.

### SPM

O agente responsável pela administração dos recursos espaciais no MASE-BDI é o SPM. Suas percepções são direcionadas a mudanças do ambiente realizadas pelos TRA, além de buscar e fornecer novas posições ao TRM quando os agentes precisam ser instanciados ou conflitos necessitam ser resolvidos. Seus objetivos são descrito na Figura 3.10. O objetivo *AtualizarMatrizProximal* é ativado quando é percebida uma mudança no GRID por exploração do ambiente, e é conquistado quando a matriz proximal tem suas posições adequadas à nova realidade. Quanto a *DisponibilizarNovaArea*, é um objetivo que é ativado quando o TRM necessita de novas posições para o TRA, e é atingido quando a melhor posição x da matriz proximal é escolhida, com x não coincidente com nenhuma

área usada ou em uso. Por fim, o objetivo *AtenderRequisicao* é trivial e similar aos outros agentes gerentes.

As crenças iniciais de SPM são mostradas na Figura (Figura 3.11). As crenças *GRID, MatrizProximal* e *PoliticaPublica* são resultado do pré-processamento das imagens e regras inseridas pelo usuário. As crenças representam o estado do ambiente, a matriz composta pelas variáveis (que provém atração ou repulsão de exploração) e a política pública de exploração em vigência na área. As outras crenças, *Agentes, Mensagens* e *PosicoesUsadas* são processadas das percepções do agente dinamicamente.

Os planos de SPM são expressados na Figura 3.12, diretamente relacionados com os objetivos do gerente. O plano *AtualizaMatriz* é ativado sempre que alguma mudança no GRID é percebida, e ações são tomadas para que tanto a *MatrizProximal* seja atualizada de acordo, e a posição seja incluída na crença *PosicoesUsdas*, se já não estiver. O plano *BuscarNovaPosica* é executando quando algum agente está com o posicionamento indefinido, seja porque acabou de ser instanciado ou seja porque está em conflito. Em ambos casos, essa crença é fornecida pelo TRM, e ao ele é concedida uma nova posição do GRID, para que seja repassada para um TRA, que deixará de estar sem posição definida. Por fim, o plano *AtenderRequisicao* assemelha-se a dos outros agentes, e é definido para que eventos de mensagens sejam tratados.

### TRA

Os agentes TRA são os que agem sobre o GRID, modificando-o na simulação de acordo com regras pré-definidas de exploração inseridas pelo usuário. Apesar de possuírem apenas dois objetivos não-triviais (Figura 3.13), o conjuntos de suas ações simultâneas emerge consequências complexas. O objetivo *ExplorarPosicao* é o de maior prioridade, e é alcançado quando o agente executa trabalho no passo da simulação e seu potencial de exploração esgota, ou seja, ele executa o máximo de trabalho possível. Se esse objetivo falhar, uma vez que em com o tempo a área que ele se encontra se desgasta totalmente, ele adquire o objetivo *MoverSe*. Nesse objetivo, ele ativamente procura uma nova posição consultando o GRID ao seu redor e a MatrizProximal, que vai guiar a sua movimentação, fazendo o TRA se afastar de áreas proibidas ou pouco proveitosas, e se aproximar de áreas incentivadas por políticas públicas ou próximas de variáveis que facilitam seu trabalho. No processo de escolha da próxima posição pode haver conflito, caso haja mais agentes querendo se mover e almejando a mesma área escolhida. Uma vez que o agente percebe que existem outros agentes almejando alguma das posições possíveis, o objetivo *MoverSe* falha e os TRA envolvidos se encarregam de pedir ao TRM uma nova posição.

As crenças do TRA (Figura 3.14) são sua própria posição, sua vizinhança, sua caixa de mensagens e seu potencial de exploração, além de dois estados internos, se trabalhou

e se está em conflito com outros agentes. A posição é fornecida pelo TRM no momento da instância ou da resolução do conflito do TRA, e se modifica a cada vez que o objetivo *MoverSe* é completado. A vizinhança é uma percepção obtida, dependente da posição, e reúne atributos do espaço relativos ao GRID, a matriz proximal e a política públicas vigentes, além de outros agente que estejam próximos. O potencial de exploração é definido pelo usuário nas regras do modelo, e a caixa de mensagens é a fila de mensagens recebidas por outros agentes, importante para a detecção e resolução de conflitos.

Os planos do TRA são mostrados na Figura 3.15. Os planos *ExplorarArea* e *MovimentarSe* estão intrisecamente ligados aos dois primeiros objetivos do agentes, e inclusive estão representados em suas respectivas prioridades. O plano *ExplorarArea* pode falhar na sua pré-condição, em que o TRA tenta cumprir o objetivo de Explorar a posição que se encontra, mas o potencial do solo já está esgotado. Já o plano *MovimentarSe* busca uma nova área para a mudança, mas pode falhar se o agente se encontra isolado em uma vizinhança em que todas as posições estão degradadas, ou há mais algum agente almejando a posição que deseja. Em ambos os casos o TRA entra em conflito e adota o próximo plano, *PedirNovaPosicao*, em que pede e recebe uma nova posição do TRM. Caso ele consiga explorar a posição concedida, ele adota em caráter especial o plano *ReportarSucesso*, para que o TRM atualize sua crença que o TRA em questão não está mais conflitando.

Um esquema gráfico para fluxo de funcionamento do TRA é exibido na Figura 3.16. Nele, pode-se perceber que dois tipos de mensagem são recebidos, *Autorizar explorção* e *Enviar nova posição*. A primeira mensagem é enviada pelo TRM no momento que simulação começa, para que o TRA comece a trabalhar. A segunda, também enviada pelo TRM, é usada para resolver conflitos espaciais, e traz uma nova posição em seu conteúdo a ser utilizada pelo TRA. Ambas são recebidas pelo módulo recebedor de mensagens, que mapeia o evento específico ao propósito da mensagem. Esse evento é utilizado para escolher que tipo de situação o agente se encontra e assim fazê-lo deliberar sobre seu próximo objetivo e plano de ação. O plano escolhido vai para uma lista de ações a serem executadas no passo atual da simulação, e podem gerar eventos internos ou externos (mensagens). Por exemplo, se o plano executado for *ExplorarArea*, ele gera um evento interno de sucesso, que atualiza a crença *trabalhou*. Caso o plano seja *ReportarSucesso*, uma mensagem é enviada ao TRM para informá-lo que a posição fornecida é válida e pode ser utilizada no passo atual da simulação.

Concluindo essa seção, a Figura 3.17 mostra as diversas interações por meio de mensagens entre os agentes do MASE-BDI. GRIDM comunica-se com TRM e SPM para que iniciem seus trabalhos no começo da simulação (mensagem *Começar*). TRM interage com SPM por meio de mensagens para pedir posições para os TRA no início da simulação e na resolução dos conflitos. Os TRA se comunicam com TRM no caso de conflito, e mandam

mensagens para o GRIDM quando executam suas ações do passo, para que o GRIDM possa atualizar suas crenças.

GRID Manager - PLANOS

P.InstanciarGerentes
- Precondição
  - ¬ existe (Transformation Manager) Λ
  - ¬ existe (Spatial Manager)
- Lista de Remoção
  - Ø
- Lista de Adição
  - existe (Transformation Manager) Λ
  - existe (Spatial Manager)

P.AtenderRequisicao
- Precondição
  - Mensagens ≠ Ø
- Lista de Remoção
  - Mensagens → UltimaMensagem
- Lista de Adição
  - Ø

P.Step
- Precondição
  - existe (agentes) Λ
  - trabalharam (agentes) Λ
  - step ≠ max
- Lista de Remoção
  - Ø
- Lista de Adição
  - step = step + 1

P.Reiniciar Simulação
- Precondição
  - step = maxSteps Λ
  - mensagem ≠ 0 Λ
  - trabalharam (agentes) Λ
  - reiniciarSimulacao = T
- Lista de Remoção
  - step Λ agente Λ mensagens
  - Λ maxSteps Λ remove(GRID)
- Lista de Adição
  - step = 0 Λ
  - agentes = Ø Λ
  - mensagens = Ø Λ
  - maxSteps = n
  - cria(GRID)

P.TerminarSimulacao
- Precondição
  - step = maxSteps Λ
  - mensagem ≠ 0 Λ
  - trabalharam (agentes) Λ
  - reiniciarSimulacao = F
- Lista de Remoção
  - step Λ agente Λ mensagens
  - Λ maxSteps Λ remove(GRID)
- Lista de Adição
  - Ø

Figura 3.6: Planos de ação do GRIDM. Adaptado de [4].

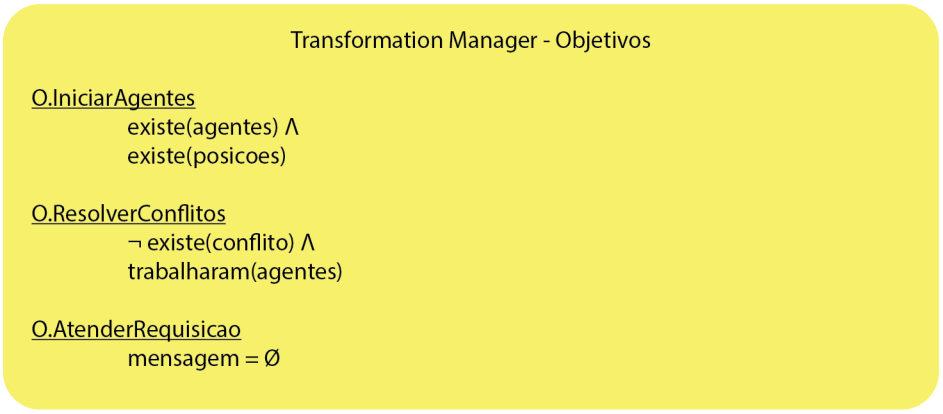



Figura 3.7: Objetivos do TRM. Adaptado de [4].

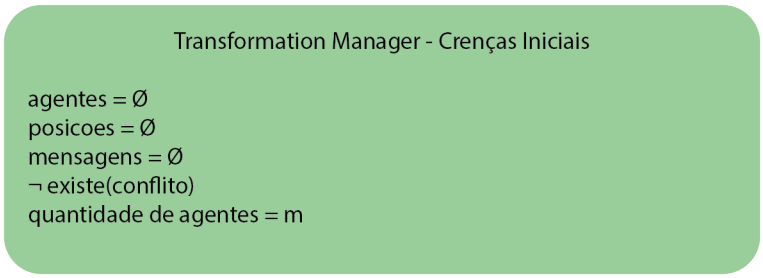



Figura 3.8: Crenças do TRM. Adaptado de [4].

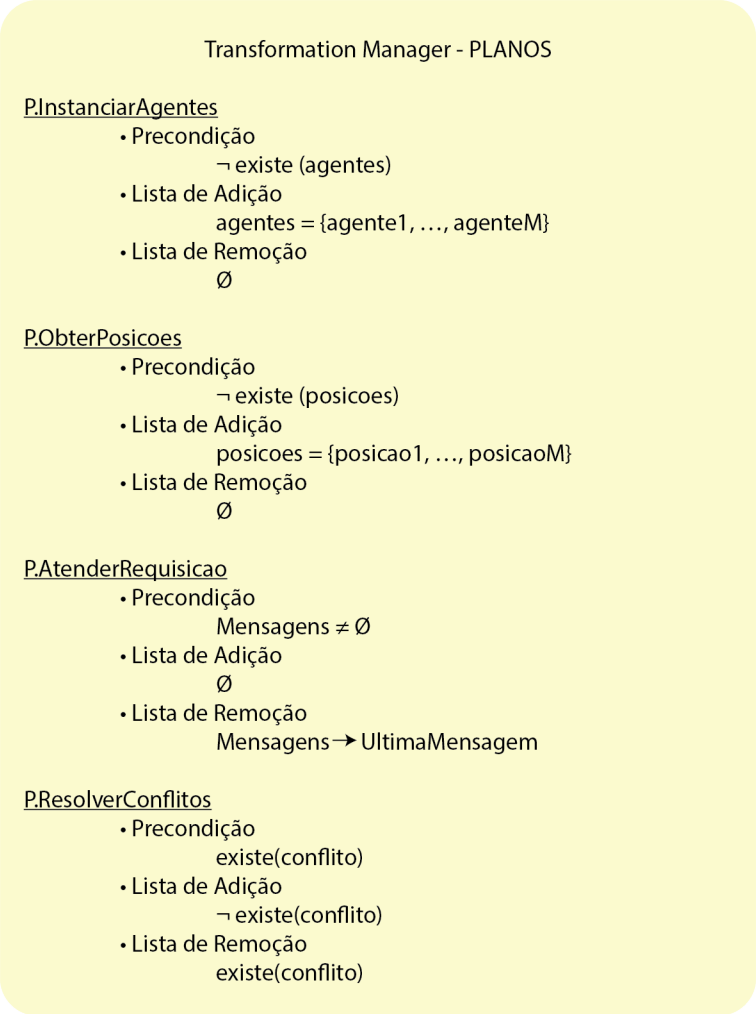



Figura 3.9: Planos de ação do TRM. Adaptado de [4].

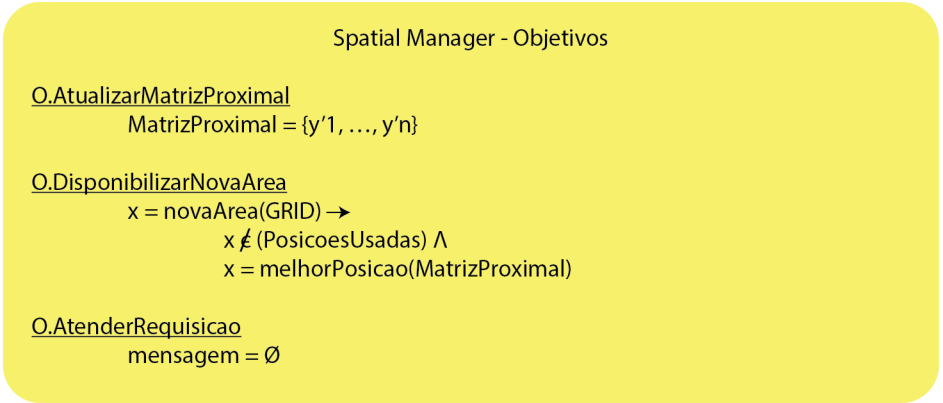



Figura 3.10: Objetivos do SPM. Adaptado de [4].

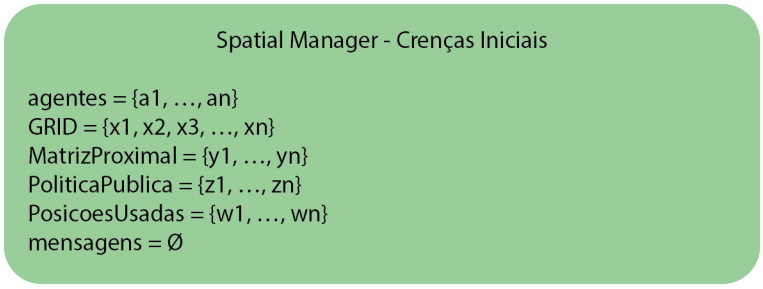



Figura 3.11: Crenças do SPM. Adaptado de [4].

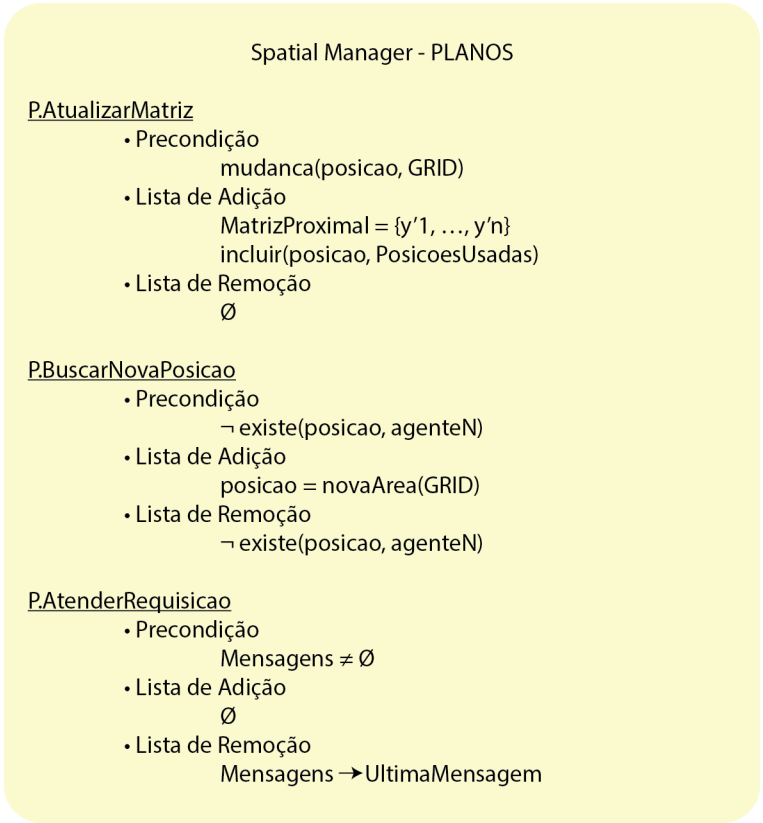



Figura 3.12: Planos de ação do SPM. Adaptado de [4].

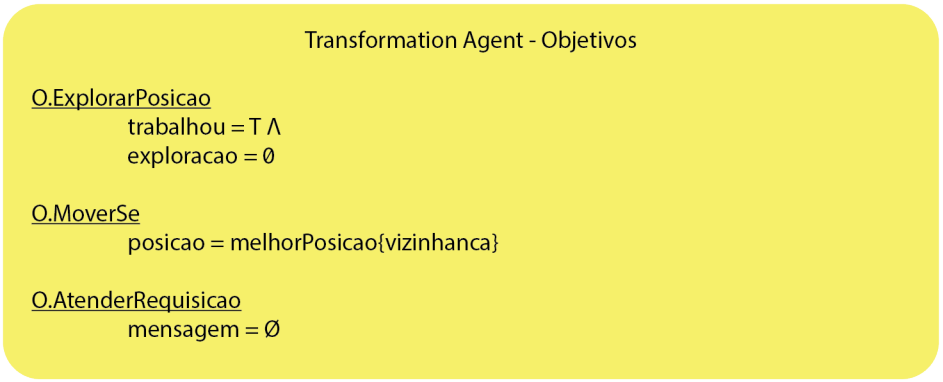



Figura 3.13: Objetivos do TRA. Adaptado de [4].

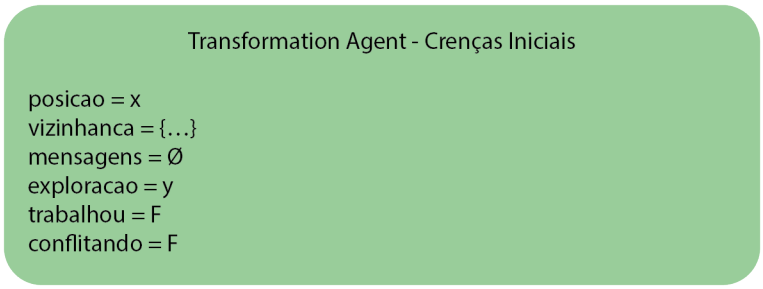



Figura 3.14: Crenças do TRA. Adaptado de [4].

Transformation Agent - PLANOS

P.ExplorarArea
- Precondição
  usoDaTerra(posicao) > 0
- Lista de Adição
  - trabalhou = T
  - usoDaTerra(posicao) = usoDaTerra(posicao) - exploracao
- Lista de Remoção
  - trabalhou = F

P.MovimentarSe
- Precondição
  usoDaTerra(posicao) ≤ 0
- Lista de Adição
  - posicao = melhorPosicao{vizinhanca}
- Lista de Remoção
  Ø

P.PedirNovaPosicao
- Precondição
  Conflitando = T
- Lista de Adição
  mensagem → TransformationManager
- Lista de Remoção
  Ø

P.AtenderRequisicao
- Precondição
  Mensagens ≠ Ø
- Lista de Adição
  Ø
- Lista de Remoção
  Mensagens → UltimaMensagem

P.ReportarSucesso
- Precondição
  trabalhou = T
- Lista de Adição
  mensagem → GRIDManager
- Lista de Remoção
  Ø

Figura 3.15: Planos de ação do TRA. Adaptado de [4].

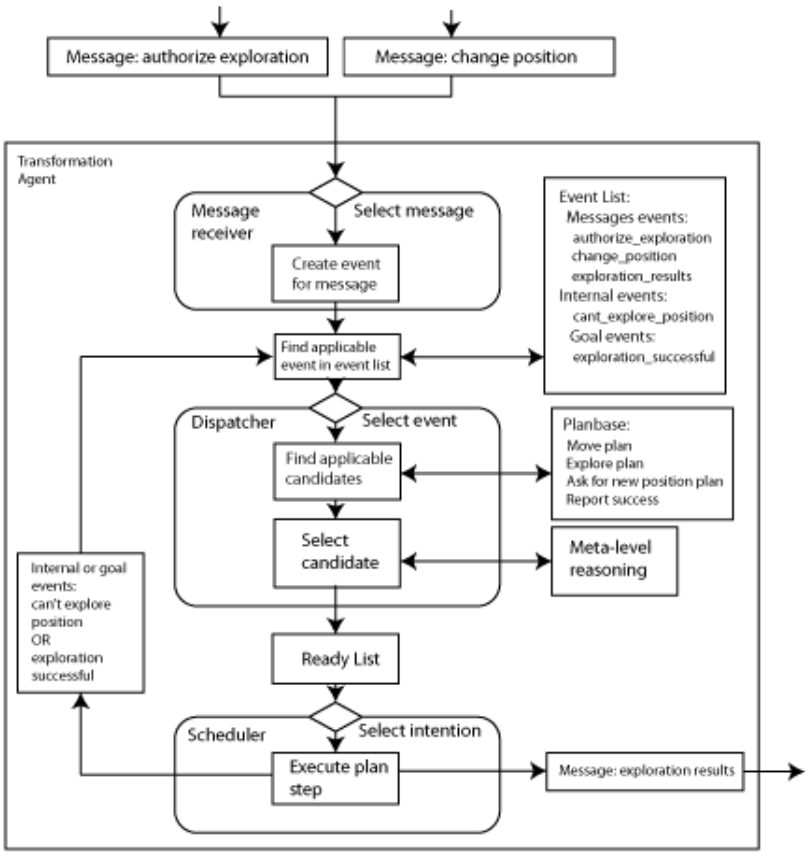



Figura 3.16: Fluxo de funcionamento do TRA. Adaptado de [5].

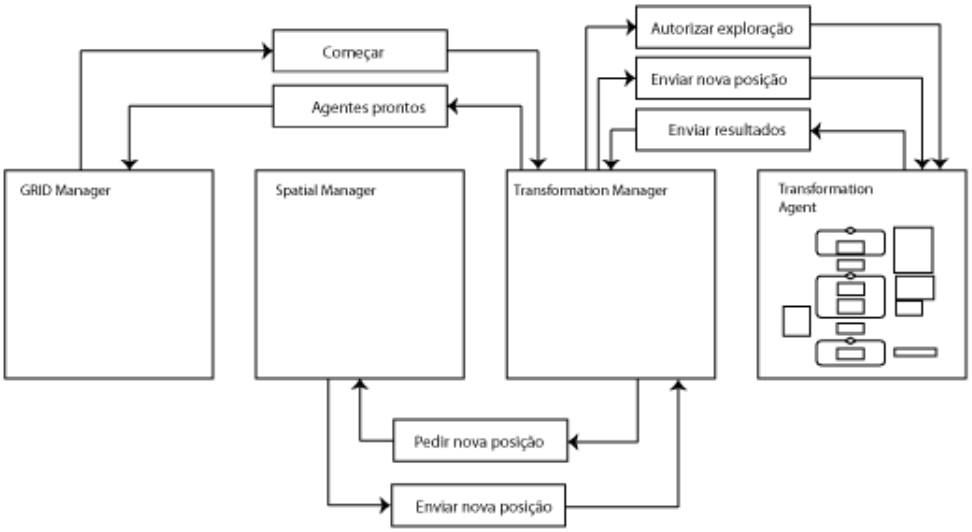



Figura 3.17: Mensagens enviadas entre as entidades do MASE-BDI.

# Capítulo 4

# Estudo de Caso

Apesar dos primeiros experimentos terem sido desenvolvidos com o modelo da Região Amazônica, outros biomas foram testados para prosseguimento dos estudos de simulação, tais como o Cerrado, incluindo o Estado do Distrito Federal e a Região da RIDE. Dessa forma, modelos definidos foram: Cerrado dentro dos limites do Distrito Federal e Cerrado considerando a região de entorno.

## 4.1 O Modelo Cerrado-DF

O bioma do Cerrado (Figura 4.1) é o segundo maior do Brasil, sendo considerado um *hotspot* de biodiversidade de grande relevância. No entanto, o esforço de conservação ainda é muito baixo em relação a Amazônia, por exemplo, dando vazão a altas taxas de exploração e desmatamento sem controle, além do problema natural das queimadas induzidas, que sem equilíbrio destroem mais ainda a cobertura vegetal [86]. Nesse raciocínio, a motivação para a caracterização desse bioma é evidente e urgente.

### 4.1.1 Variáveis Proximais

Para esse modelo utilizou-se os seguintes atributos do DF como camadas de simulação:

- Área remanescente: vegetação nativa - o cerrado em sua constituição natural. São englobadas aqui áreas de uso comum e áreas de conservação ambiental, destinadas a proteção e impassíveis de exploração;

- Área de uso antrópico: desmatamento - pode vir por cultivo agrícola, pastagens, área com influência urbana ou área degradada pela mineração. Também se encaixam nessa categoria, de forma especial, ruas, rodovias, ferrovias e edificações. Elas são tratadas de forma diferenciada porque o cultivo sobre elas é inexistente por razões

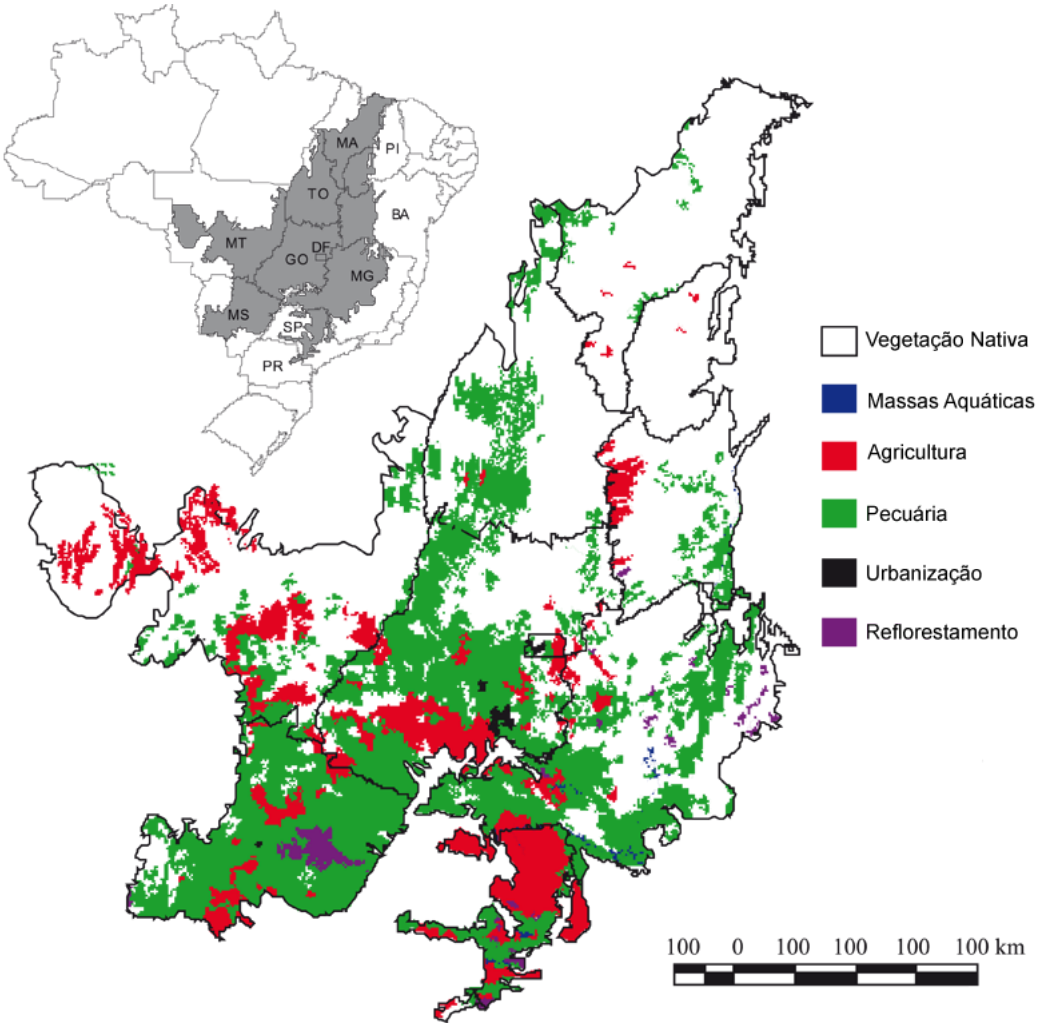



Figura 4.1: O Bioma do Cerrado [6]).

físicas (não faz sentido extração agrícola em cima de uma rua, por exemplo) ou por razões de organização de espaço (é improvável a produção agrícola no meio de um centro urbano, por exemplo).

- Hidrografia: corpo d'água - acumulação significativa de água, como lagos e outros reservatórios de água, naturais ou artificiais; curso d'água - rios, perenes ou não.

Esses atributos foram descritos por mapas provenientes dos órgãos públicos brasileiros INPE e IBAMA, cedidos pela colaboradora do projeto Carolina Abreu. Essas imagens foram capturadas pelos satélites LANDSAT e CBERS2, do INPE, e então classificadas e tratadas pelo PROBIO (Projeto do Desmatamento dos Biomas Brasileiros por Satélite), projeto mantido pelo IBAMA/DF. Os mapas de observação cedidos foram 2002 e 2008, em escala 1:50000 (Figura 4.2). Os aspectos relacionados a clima não foram considerados nessas camadas pela homogeneidade dessa característica, mas poderiam ser utlizados em outras simulações. Não se considerou queimadas também, apesar de relevante, pela

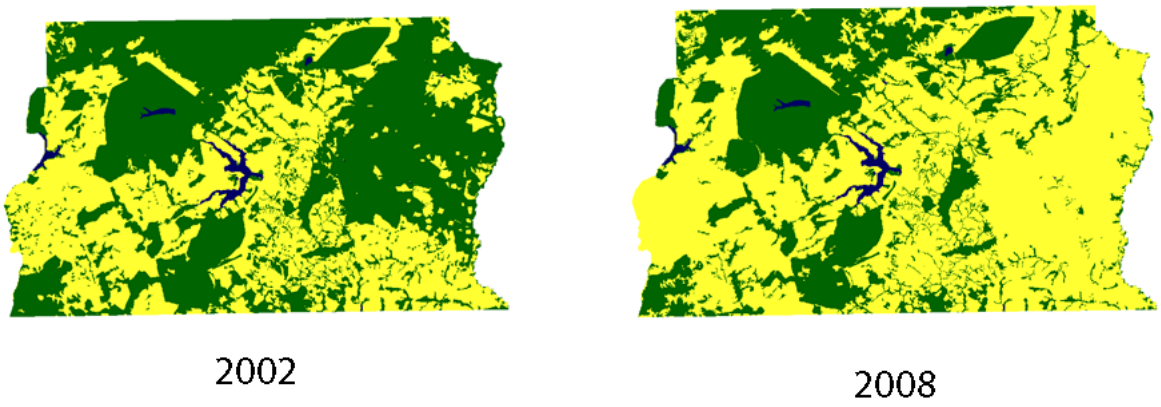



Figura 4.2: Imagens do DF cedidas nos tempos 2002 e 2008, reduzidos a 15 %original. Fonte: IBAMA/2009

questão da complexidade da dinâmica do fogo no meio ambiente - como é um agente de transformação muito rápido, o cálculo de influência sobre a vegetação se tornou inviável devida indisponibilidade de dados mais espcíficos que permitissem a adição dessa característica ao modelo.

Outro elemento inserido na caracterização dessa simulação foi o PDOT (Plano Diretor Ordenamento Territorial). O PDOT-DF é um conjunto de regras básicas de uso e ocupação do solo por categoria de uso (urbano, rural e de preservação ambiental) bem como critérios de controle do uso e da ocupação territoriais, de forma sistêmica, mediante estruturação das instituições governamentais em um Sistema de Planejamento Territorial e Urbano do DF (SISPLAN), nele incluída a participação popular. O PDOT-DF divide a região do DF em três polígonos:

- Zona Rural de uso diversificado (macrozona de consolidação do uso da terra para atividades agrossilvipastoris. Incentivo à agricultura e pecuária). Trata-se da área verde da Figura 4.3;

- Zona Rural de uso controlado (zona em que já existe agricultura e pecuária consolidada, mas que deve ser controlada (restrita) pelo impacto ao ambiente urbano, às fontes de abastecimento de água etc). Área azul da Figura 4.3;

- Outra (composta prioritariamente pela macrozona urbana e unidades de conservação. Área onde não pode haver atividades agrossilvipastoris). Região vermelha da Figura 4.3.

A intenção da inclusão do PDOT nessa simulação é a verificação da aplicação dessa política, como argumento para a sua evolução com base nos resultados obtidos. Ela

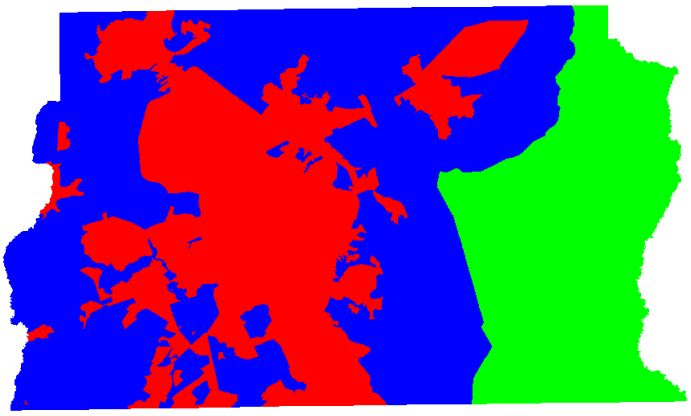


Figura 4.3: Os polígonos do PDOT. (IBAMA/2011)

é incluída como uma camada de simulação, então também tem serventia no teste da flexibilidade do software produzido.

As camadas consideradas, incluindo o PDOT, não só descrevem o espaço como também podem ser fatores de atração ou repulsão dos agentes. Todos os agricultores e pecuaristas são atraídos por rodovias, cursos e corpos d'água, são repelidos por edificações, ruas e unidades de conservação. Os agricultores são atraídos para áreas que já foram explorados pela pecuária anteriormente, e repelidos por áreas próximas a ruas e centros urbanos. Isso faz com que a coesão entre agentes e espaço seja reforçada, tornando o modelo mais próximo a realidade.

O cálculo para as camadas é feito segundo um filtro gaussiano, depois um negativo e por fim uma extração dos originais da imagem filtrada. As camadas atrativas são somadas, as repulsoras são subtraídas. Isso gera uma matriz de valores inteiros absolutos de probabilidades de um agente transformador se locomover pelo mapa. Após esse processo, as camadas do PDOT são consideradas como multiplicadores para os valores definidos na matriz. Áreas de uso diversificado tem seus valores aumentados em 10% para se tornarem mais atrativas. Uso controlado, permanecem constantes e outras são multiplicadas por zero, a fim de evitar que atraiam transformadores em áreas impossíveis, como centros urbanos. A matriz final é chamada Matriz Proximal e fica disponível para consulta por qualquer agente que requisite ao Gerente de Células ou ao Gerente de GRID.

### Regras

As regras de comportamento dos agentes, conforme apresentadas na Tabela 4.1 são comuns a todas as simulações realizadas nesse trabalho. Como regra geral aos gerentes, temos a instanciação de seus subordinados e o controle deles. Quanto aos agentes

transformadores, a regra geral é a competição do espaço, todos eles buscam aumentar o domínio de suas terras conquistadas. Mais particularmente, o Pecuarista tem prioridade sobre áreas de vegetação remanescente. Além disso, sob baixa probabilidade, tem possibilidade de ocupar uma área de agricultura se não sobrar nenhuma possibilidade de terra explorável ao seu redor. Podem voltar a áreas de pecuária em recuperação. Ele é atraído por terras próximas a cursos e corpos d'água, ferrovias e rodovias, e repelido por áreas urbanizadas, ruas e unidades de conservação. Já o Agricultor primeiro ocupa áreas que já foram exploradas por pecuária, podendo, se não restam opções, ir para áreas de vegetação nativa. Tende a voltar em áreas já agricultadas anteriormente, em uma cultura de rotação, mas não o fazem se há alguma área adjacente mais proveitosa. Seus fatores de atração e repulsão são similares ao do Pecuarista, por isso tendem a competir por áreas comuns.

O intervalo de tempo da simulação considera o Cerrado em 2002 como espaço inicial da simulação e 2008 como final. Considerando-se que a simulação é episódica, define-se que o espaço de tempo de um passo de simulação (*step*) é equivalente uma semana em tempo real. Assim, são necessários 365 *steps* de simulação para completar o intervalo de 2002 a 2008. Inicialmente, no começo da simulação, todos os espaços de mata bruta possuem um potencial de exploração de 1500. Os antropizados, 500. Espaços de variáveis (como rios) tem potencial nulo, uma vez que é impossível construir pastagens e culturas agrícolas sobre eles. Os agentes transformadores são instanciados em espaços antropizados aleatórios no começo da simulação e então exploram 500 unidades de potencial por *step* no espaço que estão, além de causar 150 unidades de potencial de impacto nas células imediatamente vizinhas abaixo, acima, à esquerda e à direita. Quando o potencial da célula se torna nulo, ele precisa mudar de espaço. Uma requisição é enviada ao gerente para que ele avalie novas áreas prováveis na região em que o agente se encontra.

Regras de instanciação e movimentação dos agentes transformadores serão apresentadas dentro da Seção 5.1, com experimentos e resultados, uma vez que variam nas diversas simulações feitas nesse trabalho.

### 4.1.2 Máquina de Estados Espacial

A presença de atividades exploratórias como agricultura e pecuária, além de outras formas de conversão do espaço, demanda uma máquina de estados de uso de cobertura do solo adequada ao contexto (Figura 4.4).

Nessa nova máquina, baseada no modelo apresentado em [87] (Figura 4.5, o estado inicial é Vegetação Remanescente. Não significa que a célula precisa estar nesse estado no começo da simulação, uma vez que as camadas de entrada vão definir em que ponto da matriz ela está. Ela transita para o próximo estado no momento que um agente

Tabela 4.1: Regras de funcionamento para os agentes

| **Agente** | **Regras** |
|---|---|
| Gerente de Célula (GC) | - Instancia células de acordo com a configuração.<br>- Mantém atualizada a matriz de probabilidades.<br>- Calcula os espaços mais prováveis de exploração, quando solicitada.<br>- Gerencia células quanto a sua exploração e regeneração. |
| Gerente de Transformação (GT) | - Instancia os tipos de agentes transformadores de acordo com a configuração.<br>- Solicita ao GC os espaços mais prováveis quando algum agente subordinado precisa se mudar.<br>- Faz a triagem dos espaços. Se são de vegetação nativa, destina os pecuarista primeiramente a eles.<br>- Gerencia exploração dos agentes. |
| Células (apenas no MASE) | - Resolve conflitos de dominação de espaço. Quando há vários agentes disputando um espaço, ele seleciona quem deve ir pra ele primeiro.<br>- Recebe ordens do GC<br>- Executa ordens do GC<br>- Envia estado para o GC |
| Agricultor | - Recebe ordens do GT<br>- Move-se de uma célula a outra<br>- Executa ações<br>- Envia seu estado e posição para GT |
| Pecuarista | - Recebe ordens do GT<br>- Move-se de uma célula a outra<br>- Executa ações<br>- Envia seu estado e posição para GT |

transformador age sobre ela, seja agricultando, explorando pecuária ou transformando em estado intangível. Os estados intermediários, Agricultura e Pecuária, são reversíveis e transponíveis. Por fim, os estados Urbanização e Área de Conservação são atingidos e finalizam a máquina, sendo recursivos até o restante da simulação.

### 4.1.3 O modelo RIDE

O modelo RIDE é uma extensão do Cerrado-DF, que considera a área do Distrito Federal e Entorno(Figura 4.6). Algumas variáveis proximais foram incluídas e o posicionamento dos agricultores e pecuaristas foi determinado por um atributo do espaço distinto. Não obstante, possui as mesmas regras de execução dos transformadores do espaço que o Cerrado-DF e a mesma classificação do ambiente. No entanto, o PDOT é removido, pois se aplica apenas ao Distrito Federal. Atributos do espaço como relevo e solo são adicionados para a melhor caracterização da exploração.

A área da RIDE consiste em 56.400 $km^2$, cobrindo todo o Distrito Federal, 18 municípios do estado de Goiás (Abadiânia, Água Fria de Goiás, Águas Lindas de Goiás, Alexânia, Cabeceiras, Cidade Ocidental, Cocalzinho de Goiás, Corumbá de Goiás, Cristalina, Formosa, Luziânia, Mimoso de Goiás, Novo Gama, Padre Bernardo, Pirenópolis, Planaltina, Santo Antônio do Descoberto, Valparaíso de Goiás e Vila Boa) e 3 municípios de de Minas Gerais (Buritis, Cabeceira Grande e Unaí). A RIDE é completamente

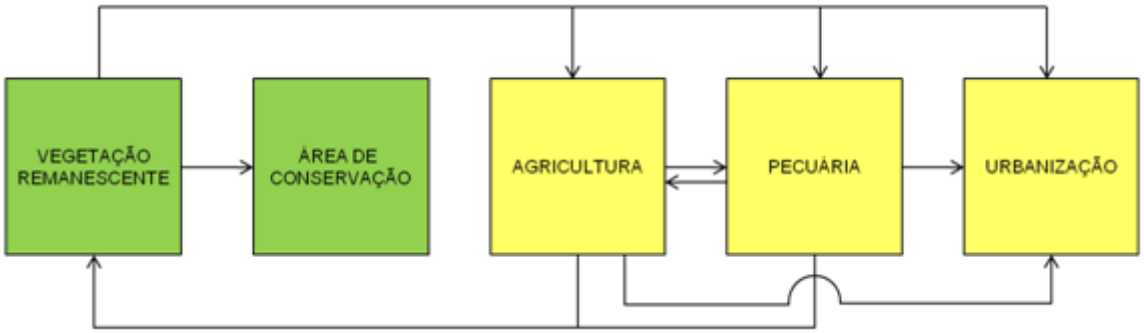



Figura 4.4: Máquina de Estados para o modelo Cerrado DF

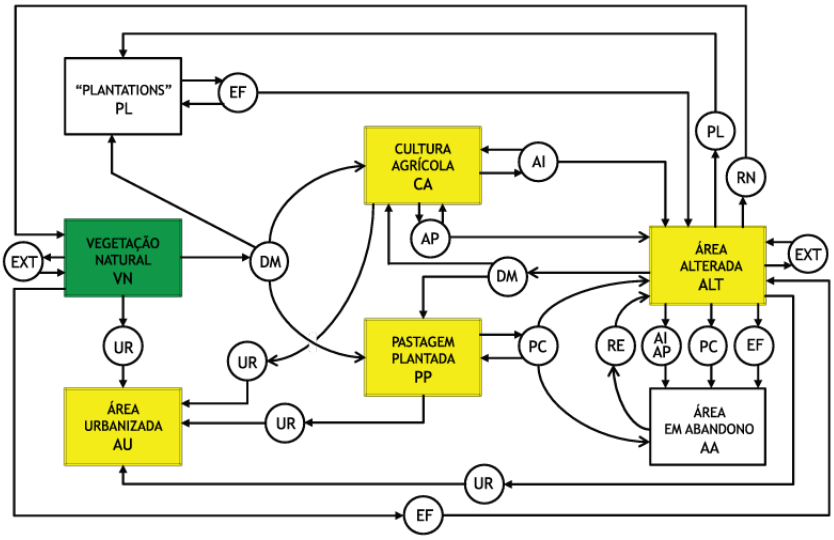



Figura 4.5: O modelo de transformação da terra proposto por Smith et al.

coberta pelo Cerrado, apresentando um área com um grande fator de impacto sobre o segundo maior bioma brasileiro.

Possuindo 26 áreas com diferentes níveis de proteção ambiental, 11 delas compreendendo ou interceptando áreas de reserva natural estrita ou áreas portegidas, a RIDE possui aproximadamente 12.5% de seu território sob algum tipo de proteção do meio ambiente, o que representa cerca de 8661.44 $km_2$. Mesmo com esse número significante de áreas protegidas, a região tem presenciado diversos conflitos ambientais e notável conversão do meio ambiente principalmente por influência antrópica. Construções em espaços ilegais, queimadas intencionais para abrir novos espaços de pasto, uso de agrotóxicos e má disposição de lixo e esgoto são alguns dos fatores que constribuíram para a degradação de áreas naturais na região do Cerrado. O processo de ocupação dos espaços no Distrito Federal tem gerado conflitos principalmente pela apropriação ilegal de áreas públicas e desobediência a leis de proteção ambiental[88]. Além de causar problemas aos residentes da RIDE, tais fatos pioraram tanto a qualidade de vida nessas áreas e suas adjacências,

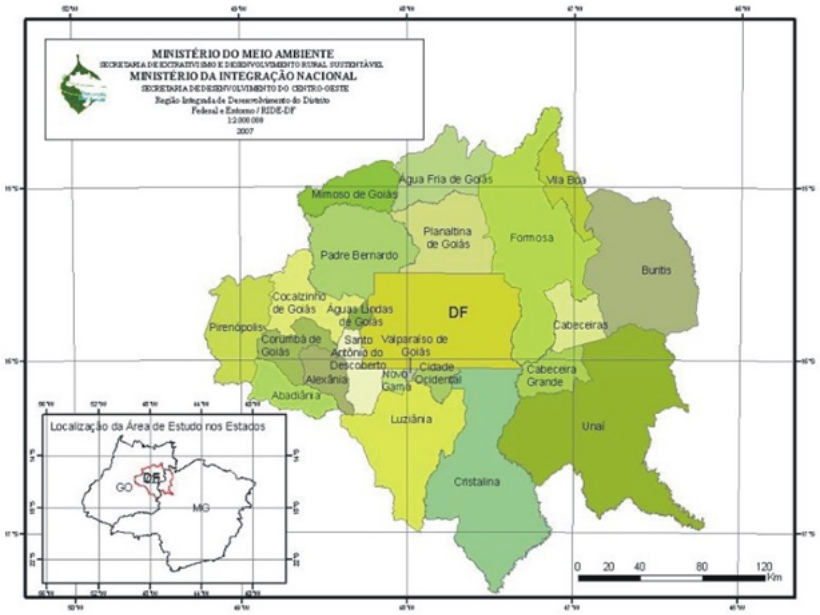



Figura 4.6: A região da RIDE. Fonte: Ministério do Meio Ambiente [7]

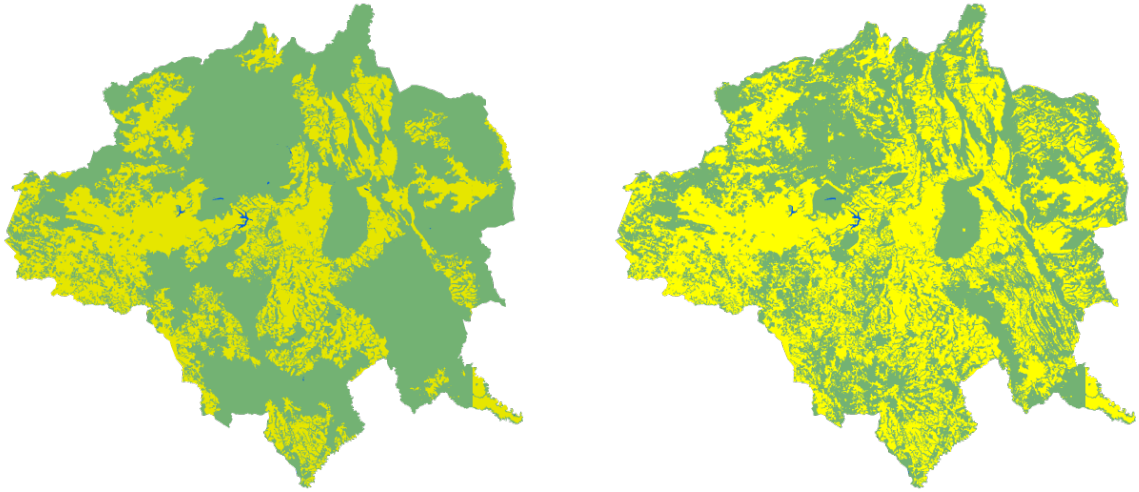
quanto contribuíram para destruição da cobertura vegetal original desses espaços.

A RIDE foi instaurada no Brasil pela Lei Complementar número 94, de Fevereiro de 1998, mas foi apenas reconhecida efetivamente pelo governo quando a Política Nacional pelo Desenvolvimento Regional foi aprovada em 2001[1]. O principal objetivo do governo era o desenvolver políticas regionais que poderia fortalecer a capacidade estratégica, monitoramento e práticas de análise em planos e programas direcionados à região. O texto da Política Nacional de Desenvolvimento Regional diz que a aplicação de planejamento integrado ao uso da terra nas áreas poderiam "reduzir desigualdades regionais e mostrar o verdadeiro potencial das regiões do país". No entanto, "mostrar o verdadeiro pontencial" desconsidera o impacto causado à sociedade e ao ambiente, visto que não foram associados estudos analisando os resultados da aplicação de políticas de exploração na RIDE quando foi criada.

### 4.1.4 Variáveis Proximais da RIDE

O estudo de caso da RIDE envolve dois marcos temporais, 2002 e 2008, similarmente ao Cerrado-DF. Eles são considerados espaços inicial e final da simulação, e foram novamente obtidas pelos satélites LANDSAT e CBERS2, do INPE, e então classificadas e tratadas pelo PROBIO, IBAMA/DF (Figura 4.7).

[1]Decreto número 7.469, de 4 de maio de 2011



(a) 2002 (b) 2008

Figura 4.7: As imagens obtidas pelo satélite LANDSAT da RIDE nos períodos de tempo 2002 e 2008.

O ambiente físico é representado por aspectos ambientais como o modelo do Cerrado-DF, porém considerou-se mais alguns atributos para uma caracterização mais rica e interessante do espaço, totalizando 11 variáveis proximais listadas a seguir:

1. Ferrovias;
2. Rodovias;
3. Rios;
4. Lagos e outros corpos aquáticos;
5. Unidades de Proteção Integral (IUCN - Categoria Ia);
6. Áreas protegidas com o uso de recursos de forma sustentável (IUCN - Categoria VI);
7. Relevo;
8. Solo;
9. Área Urbana;
10. Área Reflorestada;
11. Área Degradada;

De forma geral, é possível estimar o efeito de cada variável proximal sobre o ambiente do agente que nele se insere, assumindo a simplificação que elas são espacialmente independentes, ou seja, não afetam umas as outras. A relação pode ser positiva, negativa ou

Tabela 4.2: Relação de atração (+), repulsão (-) ou neutralidade das variáveis proximais sobre os tipos de agentes.

| Variável Proximal | Tipo de Agente | | | |
|---|---|---|---|---|
| | **Agricultor** | **Pecuarista** | **Urbanizador** | **Conservador** |
| **Ferrovias** | + | + | 0 | - |
| **Rodovias** | + | + | + | - |
| **Rios** | + | + | 0 | + |
| **Lagos** | + | + | 0 | + |
| **IUCN Ia** | - | - | 0 | + |
| **IUCN VI** | 0 | 0 | - | + |
| **Áreas Urbanas** | - | - | + | - |
| **Áreas Reflorestadas** | - | - | - | + |
| **Áreas Degradadas** | - | - | + | 0 |

Tabela 4.3: Relação de atração (+), repulsão (-) ou neutralidade do relevo sobre os tipos de agentes.

| Classes do Relevo | Tipo de Agente | | | |
|---|---|---|---|---|
| | **Agricultor** | **Pecuarista** | **Urbanizador** | **Conservador** |
| **Chapadas** | 0 | 0 | 0 | 0 |
| **Depressões** | - | - | - | + |
| **Elevados** | 0 | 0 | 0 | 0 |
| **Planaltos** | + | + | 0 | 0 |
| **Planícies** | 0 | 0 | 0 | 0 |

neutra, e o cálculo de sua influência é descrito na Seção 4.1.1. A Tabela 4.2 mostra qual é a influência das variáveis sobre os tipos de agentes, de acordo com a literatura e a opinião de profissionais na área. A Tabela 4.3 indica como as classes de relevo influenciam os tipos de agentes, e por fim, a Tabela 4.4 mostra com a influência dos tipos de solo.

### 4.1.5 Regras e Máquina de Estados da RIDE

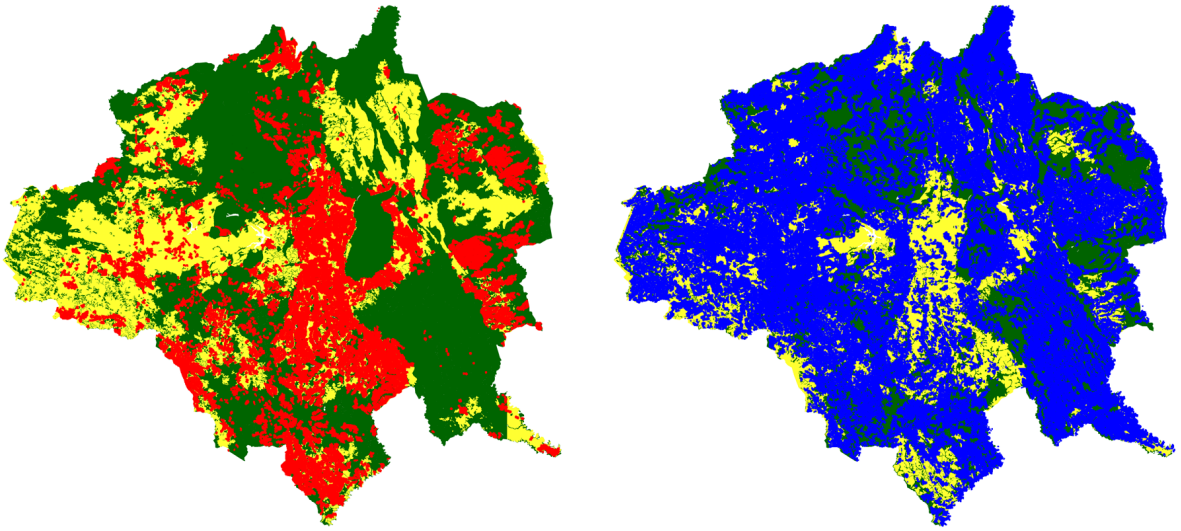
De forma geral, as regras do modelo Cerrado-DF (Seção 4.1.1) se aplicam de forma similar sobre a extensão da área do modelo da RIDE, e o mesmo é verdade para a máquina de conversão do uso da terra (Seção 4.1.2). Uma adição a esse modelo é o mapeamento dos espaço iniciais para os agricultores e pecuaristas (Figura 4.8). No começo da simulação, em vez de aleatoriamente iniciados em áreas antrópicas, são distribuídos em áreas marcadas pelo equipe do PROBIO. Agricultores e pecuaristas são respectivamente instanciados em áreas de agricultura (Figura 4.8a) e pecuária (Figura 4.8b).

### 4.1.6 Caracterização do Ambiente

O ambiente dos dois casos de uso, Cerrado-DF e RIDE, como sugere a classificação de [16], são comparados com a realidade de acordo com os atributos listados na Tabela 4.5.

Tabela 4.4: Relação de atração (+), repulsão (-) ou neutralidade do solo sobre os tipos de agentes.

| Classes de Solo | Tipo de Agente | | | |
|---|---|---|---|---|
| | **Agricultor** | **Pecuarista** | **Urbanizador** | **Conservador** |
| **Cd1-5** | - | - | 0 | + |
| **Id1-4** | 0 | 0 | 0 | 0 |
| **Lld1-12** | + | + | 0 | 0 |
| **Lvd-12** | + | + | 0 | 0 |
| **Lvde** | + | + | 0 | 0 |
| **Pd1-16** | 0 | 0 | 0 | 0 |
| **Pe 1-8** | 0 | 0 | 0 | 0 |
| **Rd1-6** | 0 | 0 | 0 | 0 |
| **Rde1-4** | 0 | 0 | 0 | 0 |
| **Tve1-6** | + | + | 0 | 0 |



(a) Lavouras (b) Pastos

Figura 4.8: Localizações das áreas de agricultura e pecuária na RIDE.

Tabela 4.5: Classificação do Ambiente

| **Real** | **Modelado** |
|---|---|
| Parcialmente observável | Parcialmente observável |
| Estocástico | Determinístico |
| Sequencial | Episódico |
| Dinâmico | Estático |
| Contínuo | Discreto |
| Multiagente | Multiagente |
| Híbrido | Híbrido |

O ambiente real é um SMA composto por seres humanos, sua ação antrópica e agentes naturais. Ele é, do ponto de vista do agente, parcialmente observável em sua maioria. Por mais que a ação antrópica possa ser influenciada por uma visão global por imagens de satélite, por exemplo, ela é pontual em relação ao volume de agentes envolvidos. É sequencial, uma vez que a continuidade temporal confere essa qualidade, e dinâmico, pela evolução das características do ambiente. É obviamente contínuo e multiagente, e os atores tem seus objetivos diversos: colaborar com o próximo, competir por espaço, aumentar lucros, entre outros. Dessa forma, a abordagem nesse ambiente é híbrida.

Já no modelo, várias características são aproximadas e simplificadas por limitação do software. Por mais que ainda seja multiagente, ele é discreto, estático e determinístico, pelas classes de célula e pelo comportamento dos agente, além da divisão do tempo, realizada por *steps*, que traz outro atributo: episódico. A simulação é multiagente e os agentes tem uma visão parcial sobre o ambiente (com exceção dos gerentes). Por fim, os agentes tanto competitivos (no caso dos transformadores, por exemplo, buscando novos espaços como objetivo principal) quanto colaborativos (no caso dos gerentes, por exemplo, coordenando a ocupação do espaço). Isso porque o mais relevante a ser respondido, o impacto ambiental sobre o Cerrado, é melhor representado pela competitividade de recursos dos agentes, trazendo resultados mais representativos.

# Capítulo 5

# Experimentos e Resultados

Utilizando o modelo Cerrado-DF e RIDE-DF, foram construídos vários ambientes de simulação. Foram averiguados qualidade do resultado e desempenho do sistema, em duas máquinas diferentes. Para os Cenários 2 e 3 foi utilizada a metodologia descrita em [70] (Seção 5.3) para a avaliação dos resultados. Os experimentos do MASE com a respectiva arquitetura são expostos a fim de comparação.

## 5.1 Experimentos no MASE

Os experimentos no MASE foram realizados na ferramenta durante período de dezembro de 2011 a abril de 2012. Os primeiros experimentos consideraram o Distrito Federal em 18 pedaços diferentes, sendo explorados pelo mesmo número de agentes à mesma taxa e com movimentação em vizinhanças adjacentes e determínisticas. O segundo experimento contemplou o Distrito Federal inteiro, mas ainda com o mesmo tipo de movimentação. O terceiro experimento foi feito com uma nova movimentação com abordagem probabilística e variação de comportamento nos agentes. Todos os experimentos foram rodados em no sistema operacional Windows 7, em plataforma Java 7 64-bit, num computador com processador Intel Core i5, 4GB de memória RAM.

### 5.1.1 Cenário 1 no MASE

Para diminuir a carga computacional da simulação, o mapa do Distrito Federal foi recortado em 18 pedaços (Figura 5.1) e cada um deles foi simulado com 40 agentes de cada tipo, sendo instanciados em posições aleatórias do GRID. A Área 2, como mostra a Figura 5.1, não possui nenhuma área em sua extensão, dessa forma não foi simulada.

A instanciação dos agentes transformadores nessa simulação foi aleatória pela área do recorte que não corresponda a nenhum espaço imediatamente acima de uma variável

proximal. Dessa forma, os agricultores e pecuaristas podem começar em qualquer lugar possível de se desenvolver a atividade econômica. Quanto a movimentação, ela funciona como demonstrado nos Algoritmos 6 e 7. O agente requisita novo espaço para o Gerente de Transformação, que requisita ao Gerente de Células os valores da Matriz Proximal da vizinhança de células do agente que precisa se mover. Essa vizinhança é um espaço de oito células imediatamente adjacentes à célula que o agente está (Figura 5.5). A célula com maior valor da matriz é escolhida para ser ocupada pelo agente, e ele é avisado a se mudar para lá. Caso nenhuma célula seja possível, o agente é realocado pelo Gerente de Transformação.

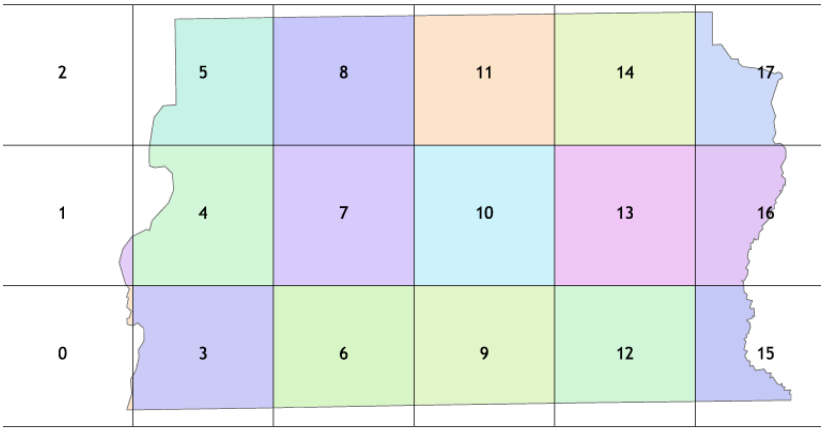
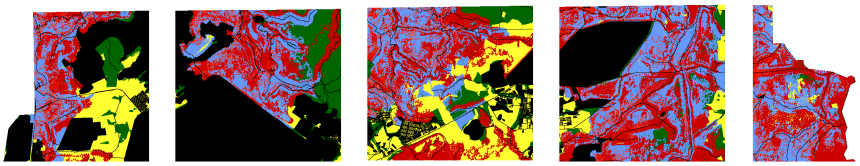



Figura 5.1: Divisão do Distrito Federal para os primeiros experimentos

```
RequisitarEMover()
estático: espacoAtual
RequisitarNovoEspaco(espacoAtual);
EsperarNovoEspaco();
espacoAtual ← ReceberNovoEspaco();
MoverSe(espacoAtual);
```

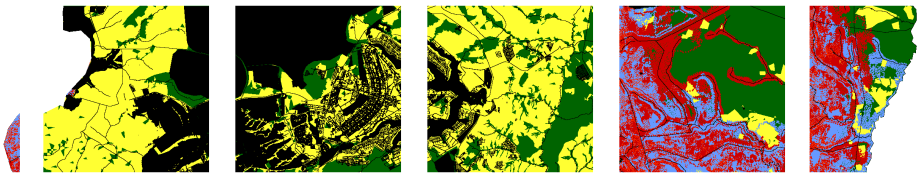
**Algoritmo 6:** Algoritmo de movimentação do agente transformador.

As Figuras 5.2, 5.3 e 5.4 mostram os resultados das simulações. Áreas em verde demonstram área bruta; amarelas, áreas antropizadas; vermelhas, fronteira agrícola; e



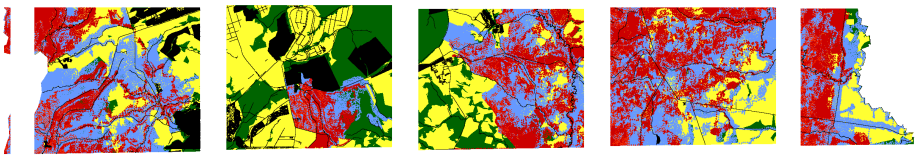
(a) Recorte 5 (b) Recorte 8 (c) Recorte 11 (d) Recorte 14 (e) Recorte 17

Figura 5.2: Recortes 5, 8, 11, 14 e 17 simulados



(a) Recorte 1 (b) Recorte 4 (c) Recorte 7 (d) Recorte 10 (e) Recorte 13 (f) Recorte 16

Figura 5.3: Recortes 1, 4, 7, 10, 13 e 16 simulados



(a) Recorte 0 (b) Recorte 3 (c) Recorte 6 (d) Recorte 9 (e) Recorte 12 (f) Recorte 15

Figura 5.4: Recortes 0, 3, 6, 9, 12 e 15 simulados

azuis, pecuária. Foi gerado um extrato contando o volume de exploração (ou seja, a soma das áreas antropizadas com fronteira agrícola e pecuária) dos recortes simulados e das respectivas áreas no espaço real de 2008. Essa contagem foi dividida pela área do recorte e multiplicada por 100 para obtenção de porcentagem dos mapas reais e dos simulados e apresentada na Tabela 5.1. Além disso, é mostrado o tempo decorrido para cada simulação.

### 5.1.2 Cenário 2 no MASE

O segundo cenário considera os agentes atuando no Distrito Federal como um todo, sem recortes. Sua instanciação ainda é aleatória, mas desde que em espaços antropizados (áreas amarelas). A movimentação é a mesma apresentada nos primeiros experimentos. Foram rodadas 15 simulações, a primeira com 10 agentes transformadores de cada tipo, a segunda com 20 de cada tipo e assim por diante, até 150.

A Figura 5.6 mostram visualmente o resultado dos experimentos 10, 70 e 150, com o mesmo padrão de cores do experimento 1. A Tabela 5.2 mostra o número de agentes de cada tipo utilizados na simulação e a porcentagem de acerto em comparação com o mapa do Cerrado DF em 2008.

### 5.1.3 Cenário 3 no MASE

O terceiro experimento foi realizado considerando uma nova movimentação sobre o mapa completo do Distrito Federal. No total, foram feitas 25 simulações, as 15 primeiras sem variação de comportamento explorativo. A movimentação passa a ser não-

```
ReceberRequisitacaoMovimentacao()
estático: espacoAntigo, espacosRequisitados, espacoNovo, agenteRequisitante
espacoAntigo, agenteRequisitante ← ReceberRequisicao();
espacosRequisitados ← RequisitarVizinhanca8();
espacoNovo ← -1;
para cada local espacoRequisitado ← espacosRequisitados faça
    se espacoRequisitado.valorProximal > espacoNovo.valor então espacoNovo ←
    espacoRequisitado;
    ;
fim
se espacoNovo == -1 então espacoNovo ← RealocarEspaco(espacoAntigo);
;
EnviarNovoEspaco(espacoNovo, agenteRequisitante);
```

**Algoritmo 7:** Algoritmo de escolha de novo espaço do Gerente de Transformação para os Cenários 1 e 2

determinística nessa etapa. O agente transformador continua com o mesmo algoritmo 6 de requisição para mover-se. No então, o Gerente de Células envia não mais as células imediatamente vizinhas à original (Figura 5.5), mas sim um conjunto próximo ao agente, chamada vizinhança radial (Figura 5.7). Para essa rodada de simulações, foi considerada a vizinhança de raio 3 (Figura 5.7c). A avaliação dos resultados foi considerando o cálculo da Figura de Mérito proposta por Pontius [70], pois apresenta uma forma qualitativa de se avaliar simulações em ambientes naturais, para comparação futura com o MASE-BDI.

Ainda na nova rotina de mudança de espaço, o Gerente de Transformação não considera o espaço mais provável para a escolha necessariamente. O conjunto de células tem seus valores de Matriz Proximal convertidos em um espaço probabilístico acumulado. Para isso, o valor da primeira célula é dividido pela soma total de todos os valores do espaço e multiplicado por 100. O valor da células seguintes são somados com as anteriores e novamente divididos pela soma total do conjunto e dividido por 100. Criam-se assim intervalos numéricos variando de 0 a 100. Um dado é lançado de 0 a 100, e o espaço que ele cair será para onde o agente vai se mudar. Caso todos os espaços do conjunto sejam proibidos para mudança, o agente é realocado para uma área nova (Algoritmo 8).

Os resultados das simulaçãoes 10, 70 e 150 são vistos graficamente na Figura 5.8. Já a Tabela 5.3 mostra a Figura de Mérito de cada simulação e o tempo decorrido.

Já a Tabela 5.3 mostra a Figura de Mérito de cada simulação e o tempo decorrido.

As 15 primeiras simulações não consideraram variação de comportamento de exploração, enquanto as outras 10 sim. Nessa etapa, o melhor resultado, o seja, com a melhor Figura de Mérito das 15 primeiras simulações, 90 agentes, foi tomado como o número de agentes de cada tipo base. Desse número, 5% passaram a explorar 3 vezes mais que o restante do grupo. Na seguinte, 10% passaram a ter essa variação, e assim por diante até

Tabela 5.1: Resultados do Cenário 1.

| Recorte | Real % | Simulado % | Tempo Decorrido |
|---|---|---|---|
| 0 | 82.9988 | 93.5458 | 00:37:17 |
| 1 | 100 | 99.9773 | 00:42:12 |
| 3 | 70.8781 | 75.1352 | 00:38:23 |
| 4 | 93.9469 | 0.21663 | 00:31:21 |
| 5 | 71.4401 | 66.4096 | 00:40:02 |
| 6 | 53.5666 | 19.4704 | 00:39:45 |
| 7 | 78.5527 | 0.23268 | 00:30:33 |
| 8 | 30.7595 | 80.5751 | 00:35:22 |
| 9 | 73.0284 | 60.7715 | 00:33:24 |
| 10 | 75.0388 | 1.9743 | 00:32:23 |
| 11 | 58.4736 | 71.792 | 00:36:57 |
| 12 | 87.2416 | 81.9157 | 00:35:56 |
| 13 | 92.3919 | 67.1786 | 00:32:21 |
| 14 | 82.6993 | 89.5014 | 00:41:12 |
| 15 | 91.9931 | 92.8291 | 00:40:01 |
| 16 | 95.3731 | 87.8817 | 00:34:07 |
| 17 | 94.3326 | 98.596 | 00:39:12 |

Tabela 5.2: Resultados do Cenário 2

| Quantidade de Agentes Transformadores de cada tipo | Porcentagem de acerto | Tempo Decorrido |
|---|---|---|
| 10 | 88.0783 | 00:12:34 |
| 20 | 89.6285 | 00:20:23 |
| 30 | 91.2394 | 00:40:10 |
| 40 | 92.5159 | 00:54:56 |
| 50 | 93.8924 | 1:23:45 |
| 60 | 95.5661 | 1:47:04 |
| 70 | 96.0693 | 2:37:45 |
| 80 | 96.8979 | 2:55:49 |
| 90 | 98.4636 | 3:46:22 |
| 100 | 99.9987 | 4:32:13 |
| 110 | 99.9080 | 5:25:12 |
| 120 | 98.9122 | 5:34:20 |
| 130 | 98.0519 | 6:00:12 |
| 140 | 97.1544 | 6:21:21 |
| 150 | 96.2322 | 6:34:12 |

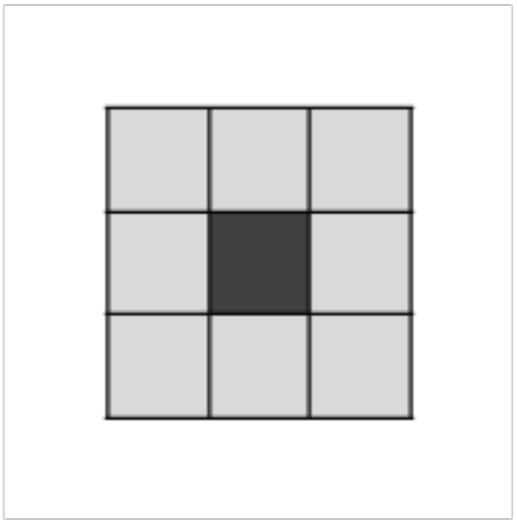
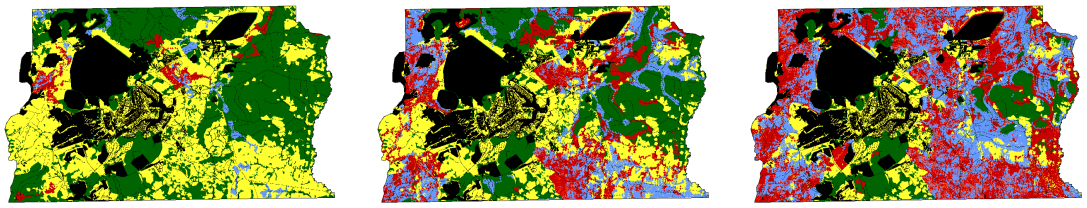


Figura 5.5: Representação da vizinhança da célula para movimentação nos cenários 1 e 2. A célula do meio é a onde o agente se encontra, enquanto as outras são a vizinhança.



(a) Simulação com 10 agentes de cada tipo

(b) Simulação com 70 agentes de cada tipo

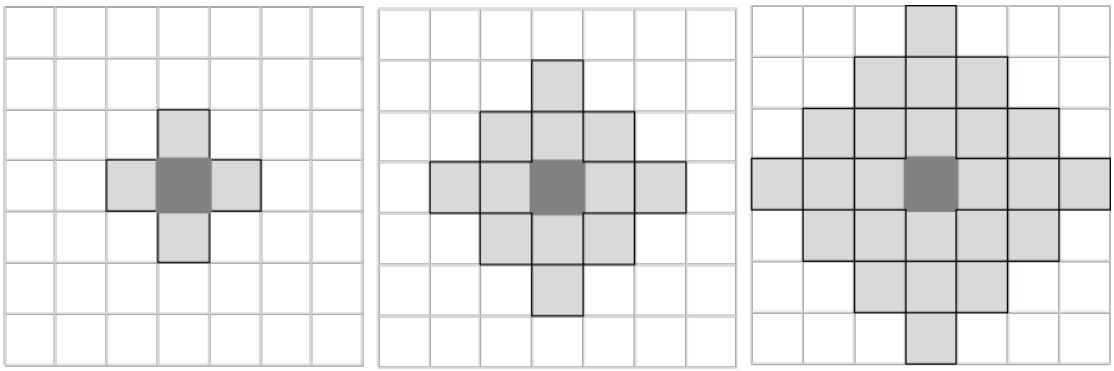
(c) Simulação com 150 agentes de cada tipo

Figura 5.6: Experimentos com 10, 70 e 150 agentes de cada tipo

50%. Os resultados gráficos das porcentagens 5, 25 e 50 são apresentados nas imagens 5.9. A Tabela 5.4 mostra a Figura de Mérito dos experimentos.

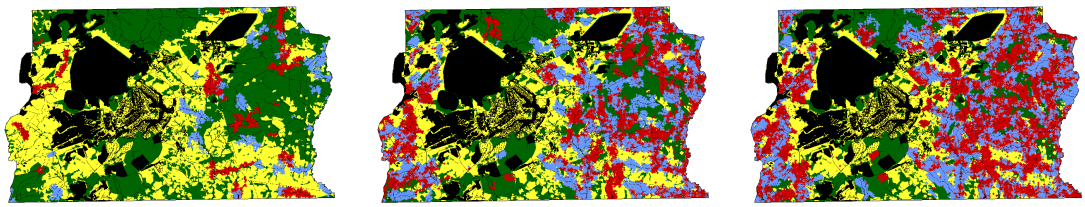
## 5.2 Experimentos no MASE-BDI

Dois experimentos foram realizados no MASE-BDI, entre maio e julho de 2014. O primeiro é o cenário 3 executado no MASE, utilizando as mesmas regras do Cerrado-DF e de movimentação probabilísticas dos agentes. Já o segundo experimento, denominado Cenário 4, foi feito com o modelo RIDE, a fim de se testar a escalabilidade do sistema utilizando imagens de alta resolução. Para ambos os experimentos, foi utilizada a avaliação de Figura de Mérito.



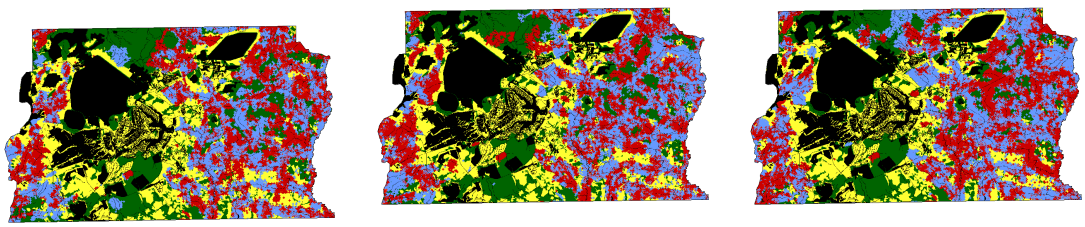
(a) Vizinhança de raio 1. (b) Vizinhança de raio 2. (c) Vizinhança de raio 3

Figura 5.7: Diferente tipos de vizinhança para o experimento 3.



(a) Simulação com 10 agentes de cada tipo (b) Simulação com 70 agentes de cada tipo (c) Simulação com 150 agentes de cada tipo

Figura 5.8: Experimentos com 10, 70 e 100 agentes de cada tipo



(a) Simulação com variação 5 (b) Simulação com variação 25 (c) Simulação com variação 50

Figura 5.9: Experimentos com variação de 5, 25, e 50%

```
ReceberRequisitacaoMovimentacao()
estático: espacoAntigo, espacosRequisitados, somaAcumulada, somaTotal,
          probabilidadesAcumuladas, espacoNovo, agenteRequisitante
espacoAntigo, agenteRequisitante ← ReceberRequisicao();
espacosRequisitados ← RequisitarVizinhancaRadial(3);
espacoNovo ← -1;
somaAcumulada ← 0 ;
somaTotal ← SomarVizinhanca(espacosRequisitados);
para local indice = 0, indice < espacosRequisitados.tamanho, indice++ faça
    local espacoRequisitado ← espacosRequisitados.indice ;
    somaAcumulada ← espacoRequisitado.valorProximal ;
    probabilidadesAcumuladas.indice ← somaAcumulada/somaTotal*100;
fim
se somaTotal == 0 então espacoNovo ← RealocarEspaco(espacoAntigo);
;
senão
    local dado ← NumeroAleatorio(0, 100) ;
    para local indice = 0, indice < probabilidadesAcumuladas.tamanho, indice++
    faça
        se dado < probabilidadesAcumulada.indice então
            espacoNovo ← espacosRequisitados.indice ;
            breakFor ;
        fim
    fim
fim
EnviarNovoEspaco(espacoNovo, agenteRequisitante);
```

**Algoritmo 8:** Algoritmo de escolha de novo espaço do Gerente de Transformação para o experimento 3.

### 5.2.1 Cenário 3 no MASE-BDI

Todos os experimentos do Cenário 3 do MASE-BDI foram executados na mesma máquina que fora utilizada anteriormesnte, para que a comparação fosse a mais justa possível. O resultados dos experimentos com 10, 30 e 70 agentes de cada tipo executados é exibido na Figura 5.10. A Tabela 5.5 mostra a Figura de Mérito de cada simulação e o tempo decorrido.

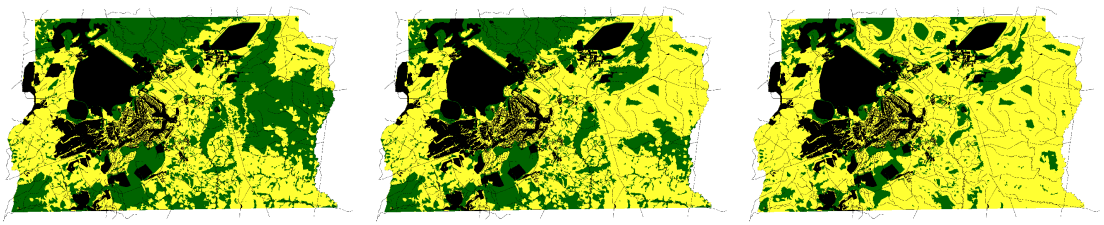
Percebeu-se que o melhor resultado obtido foi com 30 agentes de cada tipo. Dessa forma, utilizou-se a mesma metodologia do MASE e variou-se a quantidade de agentes de grupo de 5% a 50%, verificando-se sua Figura de Mérito e tempo decorrido. Os experimentos com variação de 5, 25 e 50 são mostrados na Figura 5.11. A Tabela 5.6 mostra a Figura de Mérito e o tempo decorrido desses experimentos.

Tabela 5.3: Resultados do Cenário 3

| Quantidade de Agentes Transformadores de cada tipo | Figura de Mérito | Tempo de Simulação |
|---|---|---|
| 10 | 13.14499263 | 0:14:23 |
| 20 | 23.38390255 | 0:17:25 |
| 30 | 30.96450862 | 0:26:20 |
| 40 | 33.16913627 | 0:35:11 |
| 50 | 40.20544277 | 0:59:19 |
| 60 | 41.76096015 | 1:24:24 |
| 70 | 43.83395872 | 1:49:05 |
| 80 | 48.33650662 | 2:34:13 |
| 90 | 52.93650662 | 3:00:45 |
| 100 | 50.76171565 | 3:24:19 |
| 110 | 50.37117590 | 4:03:04 |
| 120 | 50.84269911 | 5:12:51 |
| 130 | 52.40097387 | 6:23:41 |
| 140 | 51.95890604 | 6:46:13 |
| 150 | 51.42224730 | 7:45:12 |

Tabela 5.4: Resultados do Cenário 3 com variação

| Porcentagem de Agentes Transformadores | Figura de Mérito | Tempo de Simulação |
|---|---|---|
| 5 | 50.20587997 | 2:45:13 |
| 10 | 50.51588379 | 2:47:14 |
| 15 | 50.56158725 | 3:09:10 |
| 20 | 51.43231605 | 3:04:58 |
| 25 | 52.26612383 | 3:29:12 |
| 30 | 51.43839271 | 3:10:42 |
| 35 | 52.41535068 | 3:09:08 |
| 40 | 52.37892394 | 3:17:21 |
| 45 | 51.76890509 | 3:12:20 |
| 50 | 53.57247741 | 3:23:01 |

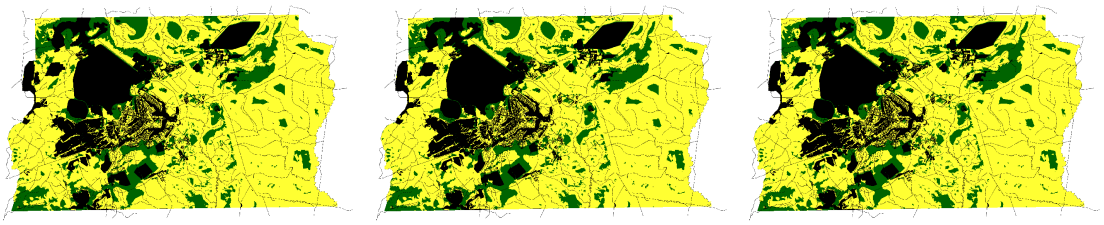


(a) Simulação com 10 agentes de cada tipo (b) Simulação com 30 agentes de cada tipo (c) Simulação com 70 agentes de cada tipo

Figura 5.10: Experimentos com 10, 30 e 70 agentes de cada tipo

Tabela 5.5: Resultados do Cenário 3 no MASE-BDI

| Quantidade de Agentes Transformadores de cada tipo | Figura de Mérito | Tempo de Simulação |
|---|---|---|
| 10 | 14.87036418 | 0:00:14 |
| 20 | 29.79210311 | 0:00:27 |
| 30 | 54.04066592 | 0:00:43 |
| 40 | 51.98603923 | 0:01:04 |
| 50 | 52.27247897 | 0:01:50 |
| 60 | 51.75358084 | 0:02:22 |
| 70 | 50.87789301 | 0:04:21 |



(a) Simulação com variação 5

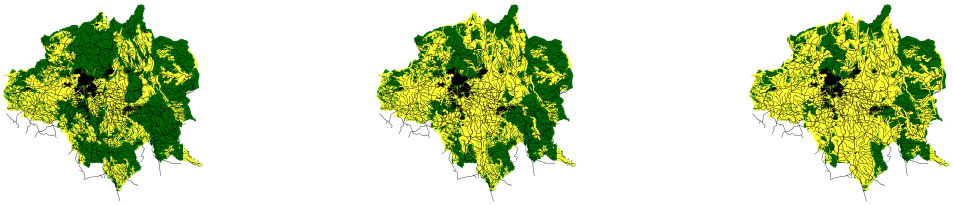
(b) Simulação com variação 25

(c) Simulação com variação 50

Figura 5.11: Experimentos com variação de 5, 25 e 50 %

Tabela 5.6: Resultados do Cenário 3 com variação no MASE-BDI

| Porcentagem de Agentes Transformadores | Figura de Mérito | Tempo de Simulação |
|---|---|---|
| 5 | 52.25424153 | 0:02:38 |
| 10 | 52.11245888 | 0:02:36 |
| 15 | 52.38607013 | 0:02:45 |
| 20 | 52.39183688 | 0:02:38 |
| 25 | 52.07170903 | 0:02:42 |
| 30 | 52.19144142 | 0:02:37 |
| 35 | 52.38957685 | 0:02:37 |
| 40 | 51.97350333 | 0:02:43 |
| 45 | 52.08681135 | 0:02:44 |
| 50 | 51.56417445 | 0:02:00 |



(a) Simulação com 10 agentes (b) Simulação com 220 agentes (c) Simulação com 450 agentes

Figura 5.12: Experimentos com 10, 220 e 450 agentes de cada tipo. %

### 5.2.2 Cenário 4 no MASE-BDI

O Cenário 4 utilizou-se do modelo RIDE, mas considerou a mesma regra de movimentação do Cenário 3 para os agentes, com uma escolha de próxima posição com vizinhança de raio 3. Para representar a área, utilizou-se imagens de 6146x3149 pixels, e iniciou-se a simulação com 10 agentes de cada tipo, com 15% deles variando a exploração como agentes de grupo. Os experimentos foram aumentando a quantidade de agentes de 10 em 10, até obter-se a saturação com 450 agentes de cada tipo. Saturação é o estado do ambiente em que não é mais possível continuar executando a simulação por não haver mais espaços disponíveis para os agentes. Para isso foi usada uma máquina com dois processadores Xeon E5-2680 e 32GB de memória, com o sistema operacional Linux CentOS 6.3 com o kernel 2.6.32 x86_64. A Figura 5.12 mostra os resultados das simulações com 10, 220 e 450 agentes de cada tipo. A Tabela 5.7 traz a Figura de Mérito e tempo dos 45 experimentos.

## 5.3 Análise de Dados

É perceptível a evolução do MASE diante da comparação dos resultados dos quatro cenários explorados. Um grande desafio enfrentado foi a capacidade computacional de executar modelos cada vez maiores. Os Cenários 1 e 2 mostram que a proposta original teve uma grande melhoria apenas pela troca de recortes pela imagem inteira do DF. No Cenário 1, podemos perceber que áreas muito pequenas, como a 0 e a 1 (Figuras 5.4a e 5.3a) possuem uma exploração excessiva, uma vez que o número de agentes é o mesmo para outras área maiores. Isso mostrou que não é adequado restringir os agentes em áreas sem ligação, eles precisam de um mínimo de espaço para expandir sua exploração e mostrar a variabilidade adequada dos resultados. No Cenário 2 percebe-se uma grande melhora exatamente por esse motivo. No entanto, o uso de uma movimentação determinística não faz sentido para a modelagem do comportamento humano, uma vez que suas decisões não são baseadas exclusivamente em variáveis determinísticas que em todas as ocasiões

Tabela 5.7: Resultados do Cenário 4 no MASE-BDI

| **Quantidade de Agentes Transformadores de cada tipo** | **Figura de Mérito** | **Tempo de Simulação** |
|---|---|---|
| 10 | 0.735625375 | 0:00:50 |
| 20 | 1.475833031 | 0:02:23 |
| 30 | 3.183221225 | 0:03:02 |
| 40 | 4.617038112 | 0:03:34 |
| 50 | 5.987528514 | 0:04:52 |
| 60 | 6.418717797 | 0:05:56 |
| 70 | 7.185681735 | 0:08:59 |
| 80 | 7.823407073 | 0:10:50 |
| 90 | 8.37163948 | 0:12:55 |
| 100 | 9.012987674 | 0:13:14 |
| 110 | 10.24056199 | 0:14:55 |
| 120 | 10.9094641 | 0:14:05 |
| 130 | 11.22076687 | 0:14:54 |
| 140 | 11.27593063 | 0:14:43 |
| 150 | 11.80540164 | 0:15:39 |
| 160 | 12.02960792 | 0:15:39 |
| 170 | 12.95510651 | 0:16:07 |
| 180 | 13.99640323 | 0:18:16 |
| 190 | 14.6523302 | 0:18:15 |
| 200 | 14.90007484 | 0:20:03 |
| 210 | 15.1320768 | 0:21:40 |
| 220 | 15.44207204 | 0:23:19 |
| 230 | 15.70143623 | 0:29:42 |
| 240 | 16.04385658 | 0:26:18 |
| 250 | 16.01430889 | 0:26:45 |
| 260 | 16.26962354 | 0:28:11 |
| 270 | 17.03407515 | 0:31:07 |
| 280 | 17.04101092 | 0:31:53 |
| 290 | 17.45558924 | 0:33:29 |
| 300 | 17.64497299 | 0:44:19 |
| 310 | 18.14588259 | 0:37:50 |
| 320 | 18.34851427 | 0:40:35 |
| 330 | 18.80741478 | 0:42:54 |
| 340 | 19.4214669 | 0:46:40 |
| 350 | 19.65010448 | 0:49:57 |
| 360 | 20.26060308 | 0:53:17 |
| 370 | 20.62662399 | 0:52:28 |
| 380 | 21.00228837 | 0:55:04 |
| 390 | 21.21090546 | 0:56:08 |
| 400 | 21.30928796 | 1:07:00 |
| 410 | 21.31812651 | 1:01:05 |
| 420 | 21.89350577 | 1:03:57 |
| 430 | 21.70729517 | 1:08:01 |
| 440 | 21.76417372 | 1:09:20 |
| 450 | 22.2950477 | 1:14:55 |

sempre dão a melhor resposta. Algoritmos em IA, que procuram a solução de problemas computacionais de forma inteligente, mostram limitações que são observadas no próprio raciocínio humano. Eles não são oniscentes, nem onispresentes, ou seja, ela não tem o domínio nem o conhecimento de todos os fatos possíveis [16]. No entanto, isso é a motivação para o uso de modelos de racionalidade, como o BDI, na análise e resolução de questões, uma vez que seres humanos utilizam de estratégias intelectuais e cognitivas para chegar uma resposta considerada satisfatória. Dessa forma, faz-se justificado o uso de um algoritmo de movimentação não determinístico.

Os Cenários 1 e 2 foram avaliados comparando apenas a imagem real (2008 observado) com a simulada no MASE (2008 observado). As áreas onde o estado permaneceu mata bruta e não foram modificados pelo agente adicionam uma pontuação injusta aos experimentos, pois o que se interessa conhecer é a eficiência com que os agentes transformadores realizam seus trabalhos.

O Cenário 3 no MASE mostra um comportamento com probabilidade no movimento dos agentes transformadores e dois níveis distintos de exploração, trazendo uma primeira tentativa de variação para se representar o comportamento observado nos agentes de forma mais variada. A Figura de Mérito (Seção 2.4.1) foi utilizada para a avaliação do MASE, uma vez que seus agentes trabalham para converter a terra, e a escolha tanto da sua movimentação quanto a quantidade de exploração é testada simultaneamente.

Para o caso do MASE sem variação de exploração (Tabela 5.3) observou-se que a partir de 80 agentes de a Figura de Mérito se aproxima de 50, atingindo sua melhor pontuação com 90 agentes. No Caso do MASE-BDI sem variação de exploração, a Figura de Mérito se aproximou de 50 a partir de 30 agentes de cada tipo, sendo inclusive o seu topo. A Figura 5.13 mostra a comparação entre o MASE e o MASE-BDI nesse Cenário. Ressalta-se que Para o MASE, 150 agentes de cada tipo saturaram o sistema, de forma que não havia posições suficientes para a simulação com 160. No caso do MASE-BDI, a saturação se encontra com menos agentes, 70. Observa-se que os agentes do MASE-BDI foram mais exploratórios que os do MASE, mas sem perder qualidade na Figura de Mérito  uma menor quantidade de agentes explorou no espaço predito corretamente. Isso demonstra uma maior representatividade do comportamento depredatório por parte dos agentes do MASE-BDI.

Ainda no Cenário 3, variou-se a quantidade de agentes de grupo, ou seja, aqueles que exploram mais que os agentes transformadores comuns, fornecendo a metáfora de cooperativas, grandes empresas e latifúndios agindo no espaço. Para essa variação, considerou-se a simulação com melhor pontuação (Simulação 90 no MASE e Simulação 30 no MASE-BDI) e variou-se a porcentagem de agentes de grupo. A Figura 5.14 demonstra a comparação das pontuações em ambos experimentos. É observável que ambas se mantiveram no

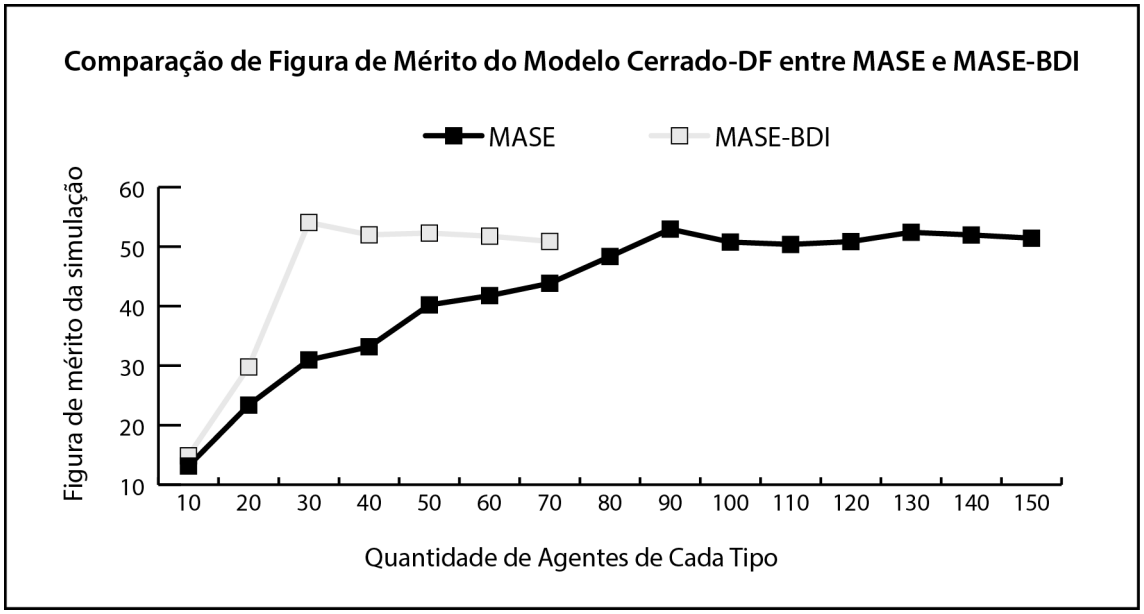



Figura 5.13: Comparação da Figura de Mérito entre MASE e MASE-BDI no Cenário 3

mesmo nível de Figura de Mérito, sem grandes divergências entre ambas. As Figuras 5.15 e 5.16 mostram a diferença de tempo de execução entre MASE e MASE-BDI sem variação e com variação, respectivamente. Observa-se uma discrepância muito grande entre ambas execuções, uma vez que o menor tempo de execução do MASE, na simulação 10 sem variação, foi maior que o maior tempo do MASE-BDI, na simulação 70 sem variação, o que mostra que a reformulação da arquitetura e refatoração do código foi trouxe uma grande melhoria para a proposta nova.

O Cenário 4, mostrado nos gráficos das Figuras 5.17 e 5.18, não obteve uma Figura de Mérito considerada muito boa, pois nenhum de suas pontuações se aproximou de 50. Não obstante, o objetivo desse experimento foi demonstrar a potencialidade o MASE-BDI, que pode executar uma quantidade de agentes muito grande em tempos de execução razoáveis. Por mais que tenha sido executado em uma máquina mais eficiente que a usada no Cenário 3, demonstra-se que o MASE-BDI é capaz de suportar muitas *threads* lógicas simultaneamente, o que abre espaço para experimentos maiores e com mais caracteríticas. Quanto ao modelo RIDE, percebe-se que seus resultados tendem a chegar a 30 em Figura de Mérito, e esse *feedback* é importante para que suas variáveis sejam ajustadas e o modelo seja novamente simulado.

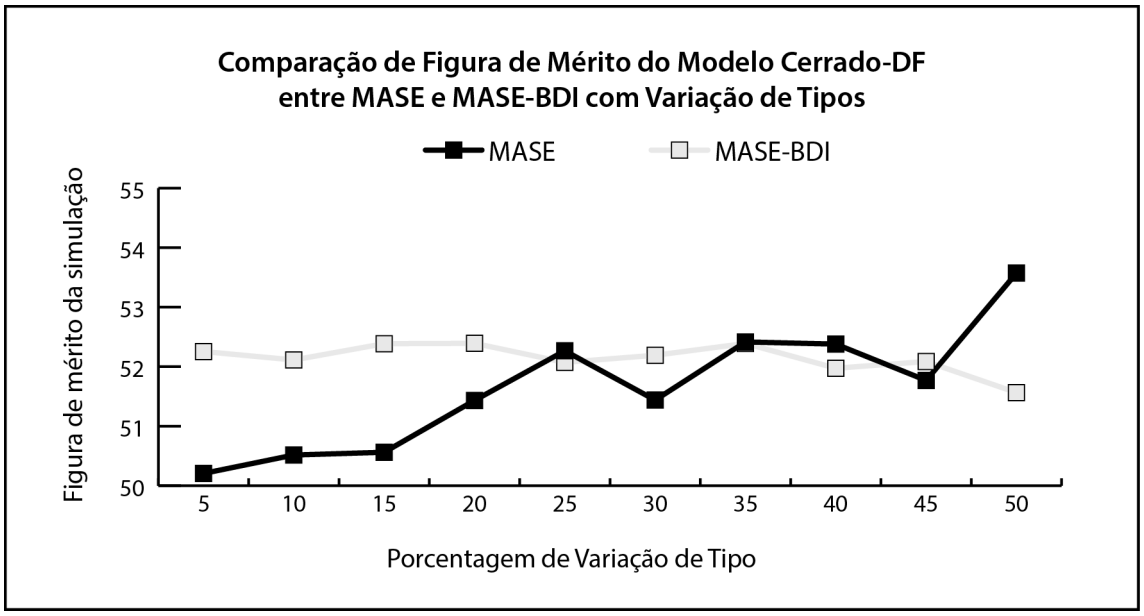



Figura 5.14: Comparação da Figura de Mérito entre MASE e MASE-BDI no Cenário 3 com variação de exploração

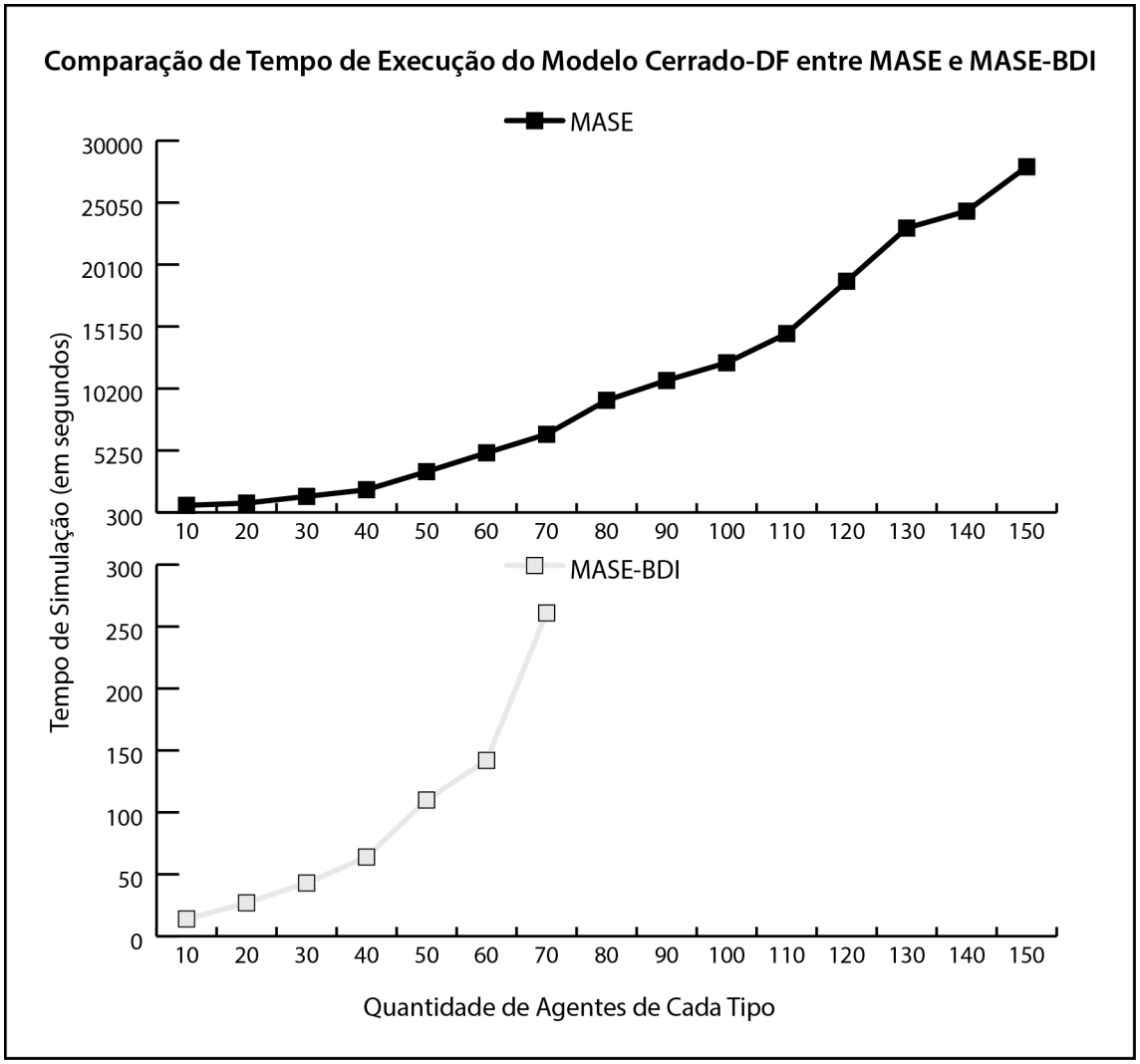



Figura 5.15: Comparação do tempo de execução do Cenário 3 no MASE e no MASE-BDI

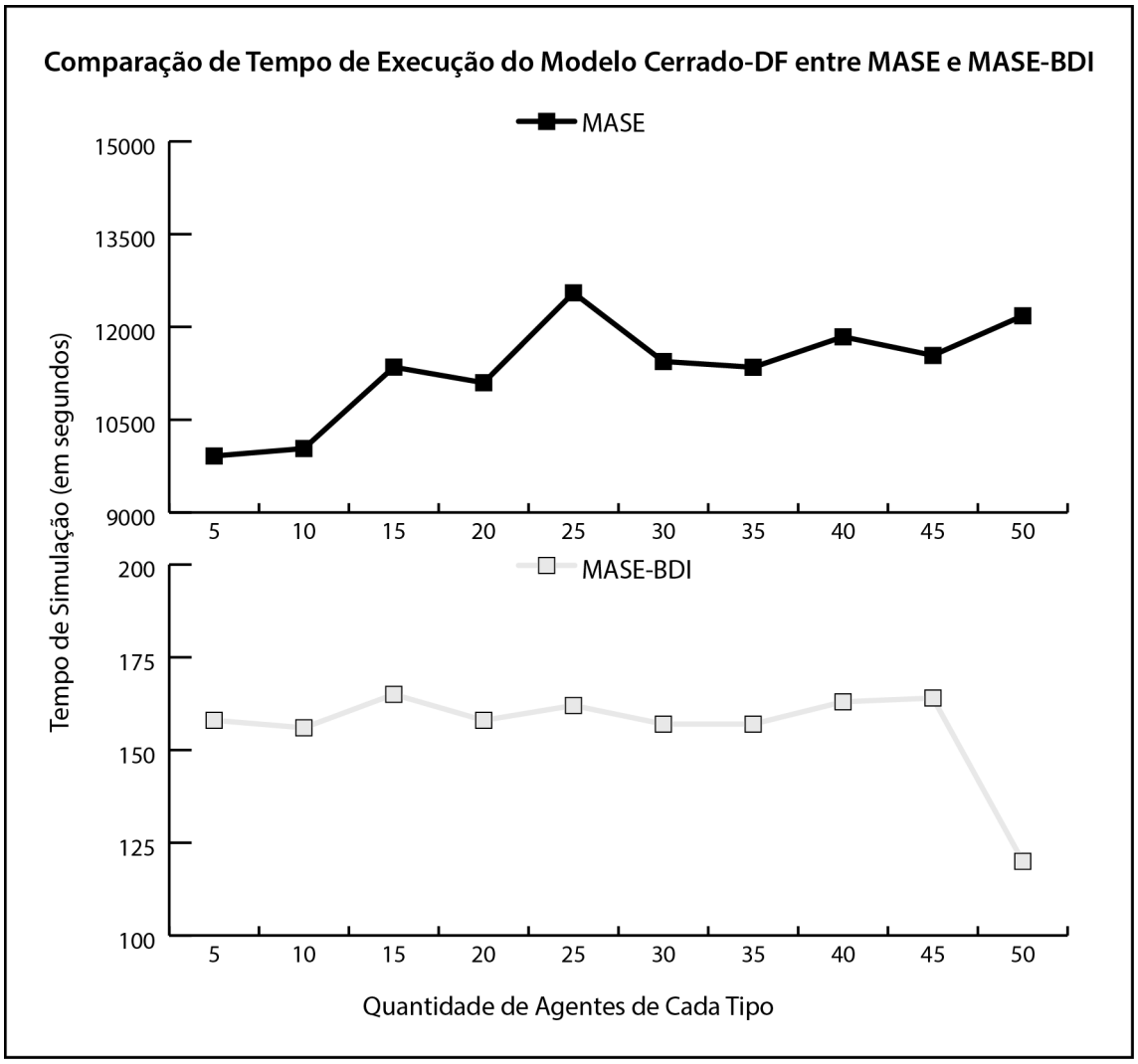



Figura 5.16: Comparação do tempo de execução do Cenário 3 no MASE e no MASE-BDI com variação na exploração

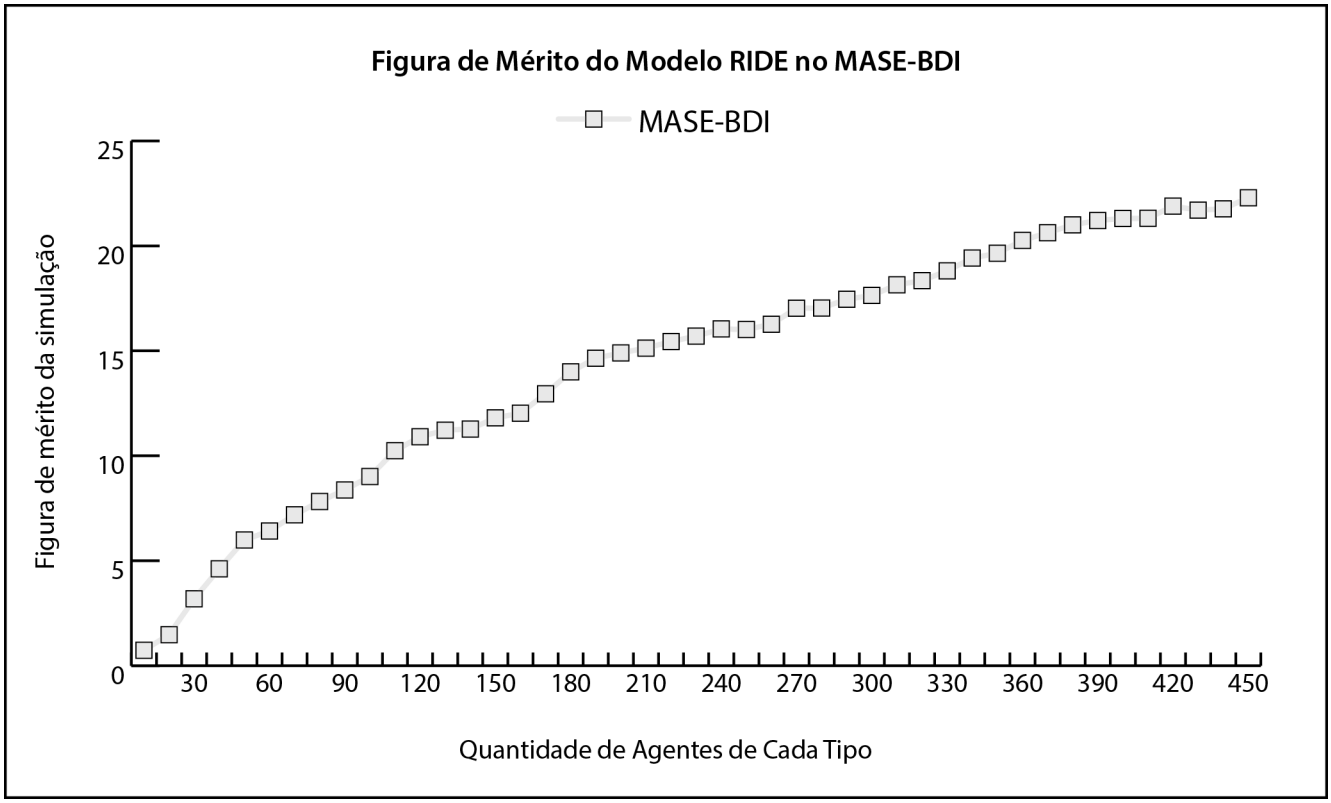



Figura 5.17: Figura de Mérito do modelo RIDE no MASE-BDI

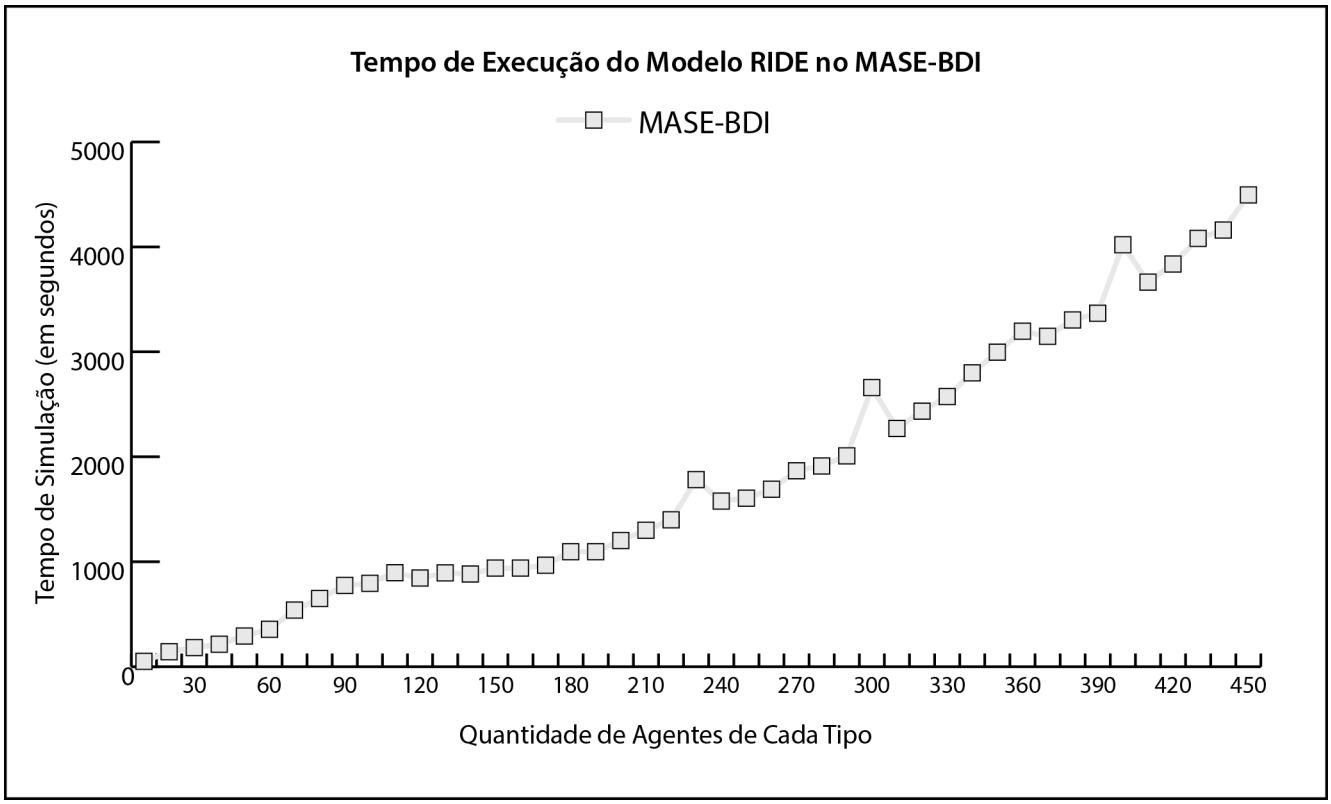



Figura 5.18: Tempo de execução do modelo RIDE no MASE-BDI

# Capítulo 6

# Conclusão

Neste Capítulo é revisitada a hipótese proposta na Seção 1.3, a fim de que seja validada. São apontados também os Trabalhos Futuros, decorrentes de nuances percebidas durante a pesquisa do MASE-BDI que podem ser exploradas proximamente.

## 6.1 Validação da Questão de Pesquisa

O presente trabalho mostrou a necessidade do aperfeiçoamento do MASE, uma vez que por mais que seus resultados com o modelo Cerrado-DF produzidos tenham sido bons, os agentes possuiam em grande parcela uma racionalidade limitada. Isso dificultava a simulação de modelos mais complexos, em que a tomada de decisão baseada na individualidade do produz comportamentos emergentes do grupo importantes para o entendimento do uso e da conversão da solo. Além disso, como visto na Seção 5.1.3, são apresentadas as Tabelas 5.3 e 5.4, que mostram um tempo de execução muito grande e inapropriado para o uso do MASE como ferramenta de MBI, em que as variáveis devem ser ajustadas de acordo com o *feedback* do resultado.

O modelo de racionalidade baseado em BDI foi descrito na Seção 2.2, para que seu funcionamento fosse compreendido e seus requisitos levantados para a reformulação da arquitetura, proposta na Seção 3.2.1. Os agentes do MASE-BDI foram então descritos na Seção 3.2.2, tendo suas crenças, objetivos e planos de ação mostrados, assim como suas interações, e então foi implementado utilizando o *framework* JADEX, que provê uma camada abstrata de BDI aos agentes, e foi baseado em JADE, *framework* utilizado no MASE.

Um novo modelo foi definido como extensão do Cerrado-DF, o RIDE (Capítulo 4). Os experimentos realizados com o MASE foram então comparados com os experimentos realizados com o MASE-BDI para que fossem ressaltadas as melhorias produzidas nesse trabalho, mostradas no Capítulo 5. Quanto a Figura de Mérito [70], a pontuação entre

as duas versões manteve-se relativamente estável sem grandes melhorias numéricas. No entanto, percebeu-se que os agentes do MASE-BDI apresentavam a racionalidade similar ao comporatmento humano, explorando o ambiente de forma mais degradativa, encenando a metáfora da ação antrópica sobre o meio ambiente. A diferença mais notável foi a de tempo de execução, que diminuiu bastante entre uma versão e outra. Enquanto o MASE apresentava simulações com grandezas de horas, o MASE-BDI conseguiu executar o Cenário estudado em questão de minutos. Ainda no Capítulo 5, foram mostrados os experimentos envolvendo o novo modelo, RIDE. Por mais que os resultados ainda não sejam considerados bons, ele mostra que o MASE-BDI é capaz de executar MBI mais complexos, com mais características, dando maior representatividade da racionalidade humana aos agentes. Os próximos passos nesse experimentos são a adequação das variáveis e novas simulações, e uma vez que o tempo de execução é menor, essa atividade tornou-se mais interessante e dinâmica para o usuário do MASE.

Com base nas análises realizadas, utilizando a arquitetura reformulada nos experimentos e considerando os resultados obtidos, é possível afirmar que a questão de pesquisa deste trabalho foi respondida e os objetivos citados na Seção 1.4 foram alcançados.

## 6.2 Trabalhos Futuros

A pesquisa com o MASE-BDI ainda possui muitas vertentes a serem exploradas. A seguir são listados alguns dos próximos passos e tendências nesse projeto:

- integração de GIS - no estudo de ferramentas para a criação e simulação de MBI, percebeu-se que grande parte delas possui integração mínima com dados georreferenciados. Essa característica é de grande relevância a elas, pois muitos atributos do modelo são diretamente ligados a variáveis dos dados, aumentando a flexibilidade dos modelos simulados no MASE-BDI. Além disso, atividades como escala dos mapas, comparação entre cenários e inclusão ou remoção de atributos do espaço são facilitadas se for possível para o *software* importar dados GIS.

- aumento da dinamicidade do ambiente - foi notado que por mais que o GRID da simulação e a Matriz Proximal do MASE-BDI são dinâmicos, as variáveis do espaço são estáticas, e não refletem a realidade ambiental. Rodovias são criadas todos os anos, enquantos lagos e rios sofrem secas e cheias com as variações do clima, por exemplo. De que forma isso poderia interagir com as simulações? A inclusão dessa nuance traria mais um controle ao projetista do MBI para execução no MASE-BDI, criando mais cenários de execução em busca de soluções para problemas de coordenação do uso da terra.

- exportação para ambiente Web - para facilitar a interação com o usuário, os resultados e a interface gráfica do MASE-BDI poderiam ser visualizáveis na Web. Dessa forma, não é necessário ao usuário realizar instalações e configurações complexas para interagir com o seu modelo e obter resultados e análise das simulações.
- inclusão de novos modelos de racionalidade - o BDI é o modelo mais usado em SMA [44], no entanto outras formas de raciocínio podem ser incluídas para dar atenção a outras características de agentes. A arquitetura do MASE-BDI define um Gerente Genérico para que outras tecnologias possam ser integradas mais facilmente.

Os trabalhos futuros possuem diversas vertentes, incluindo integração de dados de fontes GIS, interface Web e tratamento de ambientes dinâmicos, os quais demandam vários modelos abstratos, incluindo os de racionalidade dos agentes.

# Referências


[1] Alexander Pokahr and Lars Braubach. A survey of agent-oriented development tools. In *Multi-Agent Programming*, pages 289–329. Springer, 2009. xii, 21

[2] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, LTD, Chichester, England, 2nd edition, 2009. xii, 2, 9, 12, 13, 14, 23, 24, 25, 104

[3] M. J. Quinn. *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill, Dubuque, IA, 2004. xii, 27, 28, 29

[4] Charles F. Schmidt. Strips - an example of a linear planner. `http://www-rci.rutgers.edu/~cfs/472_html/Planning/STRIPS_472.html`. Acessado em: 14-07-2014. xii, 47, 48, 53, 54, 55, 56, 57, 58

[5] L. Braubach and A. Pokahr. Jadex active components framework: Bdi agents for disaster rescue coordination. In Marcin Paprzycki Mohammad Essaaidi, Maria Ganzha, editor, *Software Agents, Agent Systems and their Applications*, pages 57 – 84. IOS Press, 2011. xii, xiii, 2, 59, 104

[6] Edson Eyji Sano, Roberto Rosa, Jorge Luís Silva Brito, and Laerte Guimarães Ferreira. Mapeamento semidetalhado do uso da terra do Bioma Cerrado. *Pesquisa Agropecuária Brasileira*, 43(1):153–156, January 2008. xiii, 61

[7] Ministério do meio ambiente - região integrada do distrito federal. `http://www.mma.gov.br/`. Acessado em: 30-06-2014. xiii, 67

[8] A. Balmann. Farm-based modelling of regional structural change: A cellular automata approach. *European Review of Agricultural Economics*, 24(1):85–108, 1997. 1, 2

[9] M. Santos and M. L. S. Silveira. *O Brasil: território e sociedade no início do século XXI*. Record, Rio de Janeiro, 6 edition, 2004. 1

[10] C. Agarwal, G. M. Green, J. M. Grove, T. P. Evans, and C. M. Schweik. *A review and assessment of land-use change models: dynamics of space, time, and human choice*. United States Department of Agriculture - USDA, 2002. 1, 3

[11] J. S. Dean, G. J. Gumerman, J. M. Epstein, R. L. Axtell, A. C. Swedlund, M. T. Parker, and S. McCarroll. *Understanding Anasazi culture change through agent-based modeling*. Oxford University Press, Oxford, UK, 2000. 2

[12] P. H. Verburg. Simulating feedbacks in land use and land cover change models. *Landscape Ecology*, 21(8):1171–1183, 2006. 2

[13] C. G. Abreu, C. G. C. Coelho, C. G. Ralha, A. Zaghetto, and B. Macchiavello. Ferramenta de Simulação com Abordagem de Sistema Multiagente Híbrida para Gestão Ambiental, 2011. 2, 4, 41

[14] C. G. Abreu, C. G. C. Coelho, C. G. Ralha, A. Zaghetto, and B. Macchiavello. Ferramenta de Simulação com Abordagem de Sistema Multiagente Híbrida para Gestão Ambiental. *iSys: Revista Brasileira de Sistemas de Informação*, 4:1–22, 2011. 2, 44

[15] C. G. Abreu, C. G. C. Coelho, C. G. Ralha, A. Zaghetto, B. Macchiavello, and R. Machado. A multi-agent model system for land-use change simulation. *Environ. Model. Softw*, 42:11–37, 2013. 2, 3, 5, 6, 44

[16] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, USA, $2^{nd}$ edition, 2002. 2, 8, 9, 10, 11, 69, 84

[17] C. G. C. Coelho. Desenvolvimento de Ferramenta de Simulação Baseada em Sistemas Multiagente para Gestão Ambiental, 2012. 2, 4, 6, 41, 44

[18] Ecology definition. merriam-webster.com. `http://www.merriam-webster.com/dictionary/ecology`. Acessado em: 22-06-2014. 2

[19] C. S. Holling. Simplifying the complex: The paradigms of ecological function and structure. *European Journal of Operational Research*, 30(2):139–146, 1987. 3

[20] T. Houet, P. H. Verburg, and T. R. Loveland. Monitoring and modelling landscape dynamics. *Landscape Ecology*, 25:163–167, 2010. 3, 32

[21] E. F. Lambin, M. D. A. Rounsevell, and H. J. Geist. Are agricultural land-use models able to predict changes in land-use intensity? *Agribulture Ecosystems & Environment*, 82:321–331, 2000. 3, 32

[22] A. G. Novaes. *Modelos em planejamento urbano, regional e de transportes*. Edgard Blücher Ed., São Paulo, 1982. 3, 33

[23] Carolina G. Abreu, Cassio G. C. Coelho, Célia Ghedini Ralha, and Alexandre Zaghetto Bruno Macchiavello. Comparison of the mase system to different computational frameworks using a least cost pathway model. In *8th International Conference on Ecological Informatics*, volume 1, pages 1–3, 2012. 5

[24] Carolina G. Abreu, Cassio G. C. Coelho, Célia Ghedini Ralha, and Bruno Macchiavello. Mase: Multi-agent system for environmental simulation. In *IV Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais*, volume 1, pages 1–7. SBC, 2013. 5

[25] Carolina G. Abreu, Cassio G. C. Coelho, Célia Ghedini Ralha, and Bruno Macchiavello. The mase design experience. In *20th International Congress on Modelling and Simulation (MODSIM2013)*, volume 1, pages 2089–2095. MSSANZ, 2013. 5

[26] Carolina G. Abreu, Cassio G. C. Coelho, Célia Ghedini Ralha, and Bruno Macchiavello. A model and simulation framework for exploring potential impacts of land use policies: The brazilian cerrado case. In *47th Hawaii International Conference on System Sciences, HICSS 2014, Waikoloa, HI, USA, January 6-9, 2014*, pages 847–856, 2014. 5

[27] Alex Smajgl, Daniel G. Brown, Diego Valbuena, and Marco G.A. Huigen. Empirical characterisation of agent behaviours in socio-ecological systems. *Environmental Modelling Software*, 26(7):837 – 844, 2011. Acessado em: 22-06-2014. 6

[28] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley Series in Agent Technology, Sussex, England, 2007. 8, 12, 14, 101, 102, 103

[29] F. Bousquet, G. Trebuil, and B. Hardy, editors. *Companion modeling and multi-agent systems for integrated natural resource management in asia*. Metro Manila, Philippines: International Rice Research Institute, 2005. ISBN: 9712202089. 8

[30] F. Bousquet and C. Le Page. Multi-agent simulations and ecosystem management: a review. *Ecological Modelling*, 176(3-4):211–424, 2004. 8

[31] F. Bousquet, I. Bakam, H. Proton, and C. Le Page. Cormas: Common-pool resources and multi-agent systems. In *IEA/AIE '98: Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial In telligence and Expert Systems*, pages 826–837, London, UK, 1998. Springer-Verlag. 8

[32] G. F. Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison-Wesley, USA, $6^{th}$ edition, 2008. 8

[33] K. P. Sycara. The many faces of agents. *AI Magazine*, 19(2):11–12, 1998. 11

[34] G. Weiss, editor. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press, Cambridge, MA, USA, 1999. 12, 14

[35] N. R. Jennings, K. Sycara, and M. Wooldridge. A Roadmap of Agent Research and Development. *Int. Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):31, 1998. 13

[36] B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 4(19):45, 2004. 13

[37] Telecom Italia Lab (TILAB). Java Agent DEvelopment framework - JADE. Online. Disponível em: `http://jade.tilab.com`. Acesso em: 02/08/2012. 14, 100

[38] Telecom Italia Lab (TILAB). Java Agent DEvelopment framework - what is JADE? Online. Disponível em: `http://jade.tilab.com/description-technical.htm`. Acesso em: 02/08/2012. 14, 100

[39] J. P. Bigus, D. A. Schlosnagle, J. R. Pilgrim, W. N. Mills III, and Y. Diao. ABLE: A toolkit for building multiagent autonomic systems, 2002. 15

[40] Nick Howden, Ralph Rönnquist, Andrew Hodgson, and Andrew Lucas. JACK intelligent agents-summary of an agent infrastructure. *Management*, page 6, 2001. 16

[41] Rafael H. Bordini, Jomi Fred H??bner, and Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, Ltd, 2007. 17

[42] Hyacinth S. Nwana, Divine T. Ndumu, Lyndon C. Lee, and Jaron C. Collis. Zeus: A toolkit for building distributed multiagent systems, 1999. 18

[43] M. Bratman. *Intention, plans, and practical reason*. Harvard University Press, 1987. 18

[44] A. Georgeff, M. P. Lansky. Procedural knowledge. In *Proceedings of IEEE*, pages 1383–1398, 1987. 18, 24, 92

[45] M. J. Wooldridge and N. R. Jennings. Formalising the cooperative problem solving process. In *13th Int. Distributed Artificial Intelligence Workshop (IWDAI-94)*, pages 403–417, 1994. 20

[46] D.C. Dennett. *The Intentional Stance*. A Bradford book. A Bradford Book, 1989. 21, 22, 23

[47] Michael E. Bratman. *Intention, Plans, and Practical Reason*. Cambridge University Press, 1999. 22, 23

[48] Anand S. Rao and Michael P. Georgeff. Bdi agents: From theory to practice. In *Proceedings of the first international conference on multi-agent systems (ICMAS-95)*, pages 312–319, 1995. 24

[49] Michael Luck and Michael Wooldridge. The dmars architecture: A specification of the distributed multi-agent reasoning system. In *Autonomous Agents and Multi-Agent Systems*, pages 1–2, 2004. 24

[50] Tim Oates, Paul R. Cohen, and Probabilistic E Ects. Searching for planning operators with context-dependent and probabilistic effects, 1996. 24, 25

[51] T.D. Bui and W.J. Jamroga. Multi-agent planning with planning graph. In *Proceedings of eunite2003*, pages 558–564, 2003. 25

[52] Vladimir Lifschitz. On the semantics of strips. In Michael Georgeff, Lansky, and Amy, editors, *Reasoning about Actions and Plans*, pages 1–9. Morgan Kaufmann, San Mateo, CA, 1987. 25

[53] Pankaj R. Telang, Felipe Meneguzzi, and Munindar P. Singh. Hierarchical planning about goals and commitments. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 877–884, 2013. 26

[54] Felipe Meneguzzi, Pankaj R. Telang, and Munindar P. Singh. A first-order formalization of commitments and goals for planning. In *AAAI*. AAAI Press, 2013. 26

[55] Ian Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995. 26, 27, 28

[56] What is gpu computing? http://www.nvidia.com/object/what-is-gpu-computing.html. Acessado em: 22-06-2014. 26

[57] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2007. 27, 28, 29, 30

[58] R. Sen. Microsoft developer network: Develloping parallel programs. http://msdn.microsoft.com/en-us/library/cc983823.aspx. Acessado em: 23-06-2014. 27

[59] M. G. Turner, R. H. Gardner, and R. V. O. Neill. Ecological Dynamics at Broad Scales: Ecosystems and landscapes. *BioScience*, 45:29–35, 1995. 31

[60] Z. Naveh and A. Lieberman. *Landscape ecology: theory and application*. Springer-Verlag, New York, 1994. 32

[61] Richard T T Forman and Michel Godron. *Landscape ecology*. Wiley & Sons Ed, New York, 1986. 32

[62] M. G. Turner, R. H. Gardner, and R. V. O'Neill. *Landscape Ecology in theory and practice: pattern and process*. Springer-Verlag New York, Inc., 2001. 32

[63] A. Maria. Introduction to modeling and simulation. In *Proceedings of the 29th conference on Winter simulation - WSC'97*, pages 7–13, New York, USA, 1997. ACM Press. 33

[64] F. Neelamkavil. *Computer Simulation and Modelling*. Wiley, 1st edition, 1987. 33

[65] W. E. Grant, E. K. Pedersen, and S. L. Marín. *Ecology and Natural Resource Management: Systems Analysis and Simulation*. Wiley, 1997. 33

[66] Eric Bonabeau. Agent-based modeling: methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99 Suppl 3:7280–7, May 2002. 34

[67] Maria Chli and Philippe de Wilde. *Convergence and Knowledge Processing in Multi-Agent Systems*. Springer, advanced i edition, 2009. 34

[68] Leigh Tesfatsion. Agent-based computational economics: growing economies from the bottom up. *Artificial life*, 8(1):55–82, 2002. 34

[69] Rob Allan. Survey of Agent Based Modelling and Simulation Tools. *Engineering*, 501(October):57–72, 2009. 35, 36

[70] Robert Gilmore Pontius, Diana Huffaker, and Kevin Denman. Useful techniques of validation for spatially explicit land-change models. *Ecological Modelling*, 179(4):445–461, 2004. 35, 72, 75, 90

[71] Christian J. E. Castle and Andrew T. Crooks. Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations. Technical report, UCL Centre For Advanced Spatial Analysis, London, 2006. 36

[72] Cynthia Nikolai and Gregory Madey. Tools of the Trade : A Survey of Various Agent Based Modeling Platforms. *Journal of Artificial Societies and Social Simulation*, 12:2, 2009. 36

[73] Nigel Gilbert and Steven Bankes. Platforms and methods for agent-based modeling. *Proceedings of the National Academy of Sciences of the United States of America*, 99 Suppl 3:7197–7198, 2002. 36

[74] Tomas Salamon. *Design of Agent-Based Models : Developing Computer Simulations for a Better Understanding of Social Processes.* Academic series. Bruckner Publishing, Repin, Czech Republic, 9 2011. 36

[75] R Tobias and C Hofmann. Evaluation of free Java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation*, 7(1):1–26, 2004. 36

[76] S. F. Railsback, S. L. Lytinen, and S. K. Jackson. Agent-based Simulation Platforms: Review and Development Recommendations. *SIMULATION*, 82(9):609–623, September 2006. 36

[77] Miles T Parker. Ascape: Abstracting complexity. *Natural Resources and Environmental Issues*, 8(5):9, 2001. 36

[78] Russell K. Standish and Richard Leow. Ecolab: Agent based modeling for c++ programmers. *CoRR*, cs.MA/0401026, 2004. 37

[79] S. Luke, C Cioffi-Revilla, L Panait, and K Sullivan. MASON: A Multiagent Simulation Environment. *Simulation*, 81(7):517–527, 2005. 38

[80] László Gulyás, Sándor Bartha, Tamás Kozsik, Róbert Szalai, Attila Korompai, and Gábor Tatai. The Multi-Agent Simulation Suite (MASS) and the Functional Agent-Based Language of Simulation (FABLES) Extended Abstract for SwarmFest 2005, Turin, Italy, 2005. 38

[81] Uri Wilensky. NetLogo 5.0.5 User Manual. Evanston, IL, USA, 1999. Center for Connected Learning and Computer-Based Modeling, Northwestern University. 39

[82] Michael J. North, Nicholson T. Collier, and Jerry R. Vos. Experiences creating three implementations of the repast agent modeling toolkit, 2006. 39

[83] Michael J North, Nicholson T Collier, Jonathan Ozik, Eric R Tatara, Charles M Macal, Mark Bragen, and Pam Sydelko. Complex adaptive systems modeling with Repast Simphony. *Complex Adaptive Systems Modeling*, 1:3, 2013. 39

[84] Tiago Garcia de Senna Carneiro, Pedro Ribeiro de Andrade, Gilberto Câmara, Antônio Miguel Vieira Monteiro, and Rodrigo Reis Pereira. TerraME: An extensible toolbox for modeling natureâsociety interactions. *Environmental Modelling & Software*, 46:104–117, August 2013. 39

[85] Carlos Eduardo Marinelli, Cassio Giorgio Couto Coelho, Celia Ghedini Ralha, Alexandre Zaghetto, and Bruno Macchiavello. Modelo de simulação com uso de abordagem de sma para o zoneamento de unidades de conservação da amazônia. In *VI Simpósio Brasileiro de Sistemas de Informação*. 2010. 41

[86] E. E. Sano, R. Rosa, J. L. S. Brito, and L. G. Ferreira. Mapeamento semidetalhado do uso da terra do Bioma Cerrado. *Pesquisa Agropecuária Brasileira*, 43(1):153–156, January 2008. 60

[87] J. Smith, M. Winograd, G. Gallopin, and D. Pachico. Dynamics of the agricultural frontier in the Amazon and savannas of Brazil : analyzing the impact of policy and technology. *Environmental Modeling and Assesment*, 3:31–46, 1998. 64

[88] Roseli Senna Ganem and Zita de Moura Leal. *Parques do Distrito Federal*. Camara Legislativa do Distrito Federal, 2000. 66

[89] Oasis open sca. http://www.oasis-opencsa.org/. Acessado em: 14-07-2014. 103

[90] Jadex active components - features. http://www.activecomponents.org/bin/view/About/Features. Acessado em: 14-07-2014. 103, 104

[91] P. Diniz, S. Netto, and E. D. Silva. *Digital Signal Processing: System Analysis and Design*. Cambridge University Press, 2002. 107

[92] R. C. Gonzales and R. E. Woods. *Digital Image Processing*. Prentice Hall, Upper Saddle River, NJ, USA, 3rd edition, 2007. 107

[93] M. D. Abramoff, P. J. Magelhaes, and S. J. Ram. Image processing with ImageJ. *Biophotonics Int*, 11(7):36–42, 2004. 108

# Anexo A

# Tecnologias Utilizadas

O projeto do MASE contemplou o uso de várias tecnologias apoiadoras para o seu desenvolvimento. Elas são nesse apêndice apresentadas e contextualizadas na ferramenta.

## A.1 *Java Agent Development Framework*

Com o propósito de simplificar a implementação de SMA em uma linguagem fácil e orientada a objetos, o JADE é um *framework* criado pela *Telecom Italia SPA* e atualmente se encontra na versão 4.2 [37]. Ele provê uma série de bibliotecas escritas em Java, sob o princípio básico do uso de objetos e *threads* disponibilizados pela tecnologia, para definir agentes, plataformas, serviços, dentre outras funções e classes.

Além do uso direto e reduzido na montagem e utilização de sistemas multiagente, o JADE segue as especificações da FIPA (*Foundation for Intelligent Physical Agents*), o que significa uma maior coesão entre os agentes desenvolvidos e utilização de protocolo padronizado em suas comunicações. Essas, representadas por mensagens, são por sua vez flexíveis, apesar do cumprimento as ordens e recomendações da FIPA. Isso inclui o uso da comunicação na influência dos estados dos remententes e destinatários, além do transporte de informação serial, importante para a manutenção da confiabilidade dos dados. Essa mensagens, no âmbito do seu roteamento, podem causar problemas similares aos encontrados em redes de internet [38]. Isso, no entanto, é contornado pelo uso de pilhas de recebimento - estruturas de dados *LIFO* - que armazenam as mensagens em ordem de chegada para posterior tratamento do destinatário.

### A.1.1 Agentes

JADE provê a classe *jade.core.Agent* extensível ao programador para a construção de agentes. Conforme definido na Seção 2.1.1, agentes são elementos computacionais que

possuem a capacidade de interação em um ambiente. No contexto desse *framework*, um agente é um objeto que possui um ciclo de vida (Figura A.1) [28].

O ciclo de vida de um agente é essencial para a definição de suas funções mais básicas. Conforme sugere a Figura A.1, adaptada de [28], a chamada de sistema *create* faz com o que agente seja instanciado e vá para o estado Iniciado (*Started*). Nesse estado o agente já está na plataforma, mas não está executando, pois foi apenas aceito. Quando o método *invoke* é acionado no agente, ele então passa pro estado Ativado (*Activated*), em que ele efetivamente está executando. Dessa condição ele pode passar para os estados Em aguardo, Suspenso e Em trânsito (*Waiting, Suspended, Moving* respectivamente). Quando em aguardo, basta receber uma mensagem para voltar a execução. Se suspenso, algum outro agente deve ativamente chamar o método *resume* para voltar para o estado de execução. Caso esteja se movendo, automaticamente na chegada da plataforma o método *execute* será chamado. Quando destruído ou morto, o agente volta para o estado desconhecido. Esse estado, definido em paradigmas de programação como nulo (*null*), é uma área de memória em que objetos inutilizados são organizados e limpos posteriormente pela própria máquina virtual Java.

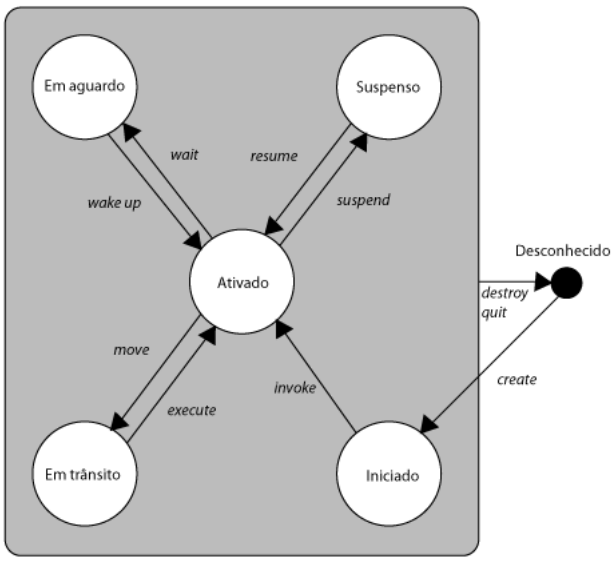



Figura A.1: Ciclo de vida de um agente.

O JADE já provê alguns agentes básicos rodando incondicionalmente quando a plataforma é iniciada. São utilizados dois deles nesse trabalho: o AMS (*Agent Management*) e o DF (*Directory Facilitator*). O AMS é responsável pelo gerenciamento dos agentes na plataforma, executando tarefas como aceitar novos agentes, destruí-los ou mudá-los de plataforma. No contexto do MASE, o AMS é acionado no momento em que os Gerentes

de Células e Transformação são instanciados pelo Gerente de GRID. Também quando esses gerentes precisam instanciar suas equipes, o AMS é responsável por aceitá-los na plataforma.

O DF provê o metasserviço de gerenciamento de serviços. Qualquer agente que queira implementar um serviço, ou seja, uma função que seja global à plataforma, deve enviar uma solicitação para o DF com o tipo do serviço e o nome. Assim, cria-se um mecanismo de páginas amarelas, em que outros agentes interessados em uma determinada funcionalidade podem consultar [28]. Um exemplo no MASE é o próprio Gerente de GRID. Ele gerencia os recursos da simulação, logo registra o serviço de gerência com o DF. Agentes interessados nesses recursos devem consultar esse serviço com o DF e enviar requisições para o Gerente de GRID. Isso tem a principal vantagem de manter a coerência do código da ferramenta. Os serviços são registrados em tempo de execução, e não anotados no código dos agentes que necessitam, prezando pela simplicidade da solução e facilitando a manutenção da aplicação.

### A.1.2 Comportamentos

Outras ferramentas para o desenvolvimento de agentes em JADE são os comportamentos (*Behaviours*). São um conjunto de classes que podem ser extendidas e personalizadas pelo programador para diferenciar os mais diversos tipos de ações que um agente pode ter, independente do nível de racionalidade (Seção 2.1.1).

A classe mais genérica, *jade.core.behaviours.Behaviour*, possui três métodos básicos: *Action, OnStart e OnEnd*. Necessariamente ligada a um agente, a instância da classe tem a chamada *OnStart* assim que seu agente entra no estado Ativado. Esse método é executado uma única vez e tem a utilidade de contextualizar a ação. *Action* é chamado consecutivamente e é a ação em si. Cabe ao programador codificar a rotina pertinente ao agente. Após a execução, *OnEnd* é o método chamado para liberar recursos tomados pelo *Behaviour*. Existem subclasses que não possuem fim (como *CyclicBehaviour*) ou que possuem são executadas apenas uma vez (como *OneShotBehaviour*). O seu uso deve ser adequado para cada propósito de aplicação.

Exemplificando no MASE, a classe *jade.core.behaviours.OneShotBehaviour* é executada pelo Agente de Configuração para o pré-processamento de regras. Uma vez que essa ação só precisa ser feita uma vez antes da simulação começar, o uso de um comportamento estritamente executado unicamente é coerente com a aplicação.

### A.1.3 Protocolos de Interação

Como mostrado na Seção 2.1.5, um SMA precisa de comunicação e interação entre seus agentes para que esses possam trabalhar em conjunto. FIPA define uma série de protocolos-padrão para a comunicação entre agentes, implementados em JADE pela classe *jade.lang.acl.ACLMessage*. Outros tipos de protocolos podem ser implementados pelo programador, mas pela simplicidade e objetividade optou-se por estender o domínio dessa classe no MASE.

As mensagens no JADE são objetos que podem transportar vários atributos textuais, numéricos e multi-valorados, desde que implementem a interface *java.io.Serializable*. Apenas agentes podem enviar mensagens, mas esse procedimento pode ser feito dentro de *Behaviours* para maior flexibilidade de programação. Mensagens também são capazes de mudar o estado Em aguardo de agente para Ativado, pois chamam o método *wake up*. Se várias mensagens chegarem para um mesmo agente, elas são organizadas em uma estrutura de dados tipo lista duplamente encadeada LIFO, de acordo com sua ordem de chegada [28].

As mensagens possuem performativas, que podem ser analogamente comparadas ao assunto de uma correspondência. O JADE já possui 21 performativas implementadas definidas pela FIPA [28]. O MASE traz mais 9 performativas específicas, utilizadas pelos agentes para otimização das ações a serem tomadas.

## A.2 JADEX

Similar o JADE, JADEX é um *framework* de desenvolvimento e aplicação de Sistemas Multiagentes. Sua principal característica é a de utilizar o modelo de programação orientado a serviços, em que são definidos componentes proativos que provem e requisitam funcionalidades uns dos outros. Esse modelo utiliza a arquitetura Componente-Serviço (*Service Component Architecture*, SCA), utilizado em soluções de empresas como a IBM e a Primeton [89], tipicamente para Aplicações Web ou Sistemas Distribuídos [90].

Os componentes proativos citados podem prover serviços passivamente ou executar ações com motivações diversas, reagindo conforme eventos percebidos do ambiente. Também são capazes de comunicar-se uns com outros e executar concorrente. Dessa forma, os componentes possuem características de agentes em um Sistema Multiagente, e podem ser programados para solucionar problemas utilizando simples ações reativas a deliberações e direcionamentos em busca de realizações de objetivos, podem implementar o modelo BDI em sua racionalidade [90].

A linguagem de programação padrão para os componentes do JADEX, assim como seus objetos pertinentes é o Java, apesar de que é possível descrevê-los utilizando XML (*eXtensible Markup Language*) ou BPMN(*Business Process Modeling Notation*) [90].

### A.2.1 O Modelo BDI no JADEX

Para esse trabalho, os componentes JADEX foram utilizados como agentes racionais em uma plataforma multiagente, em que seu modelo de racionalidade era o baseado em BDI. Um agente racional possui uma representação explícita do mundo em que se insere, assim como seus objetivos a serem conquistados; suas crenças são atualizadas conforme sua percepção sobre o ambiente e seu estado interno muda; e seus planos são agrupamentos de ações possíveis a serem realizadas, com diferentes estratégias e consequências. A esse conjunto de propriedades do agente é dado o nome de capacidades (*capabilities*), que do ponto de vista da implementação do agente, são arquivos que possuem objetivos, planos e crenças afins. O agente com racionalidade BDI também possui um módulo de raciocínio prático, composto por um elemento deliberador e um raciocinador meios-fim [2, 5]. Essa representação pode ser vista na Figura A.2, onde os componentes internos do agente são explicitados.

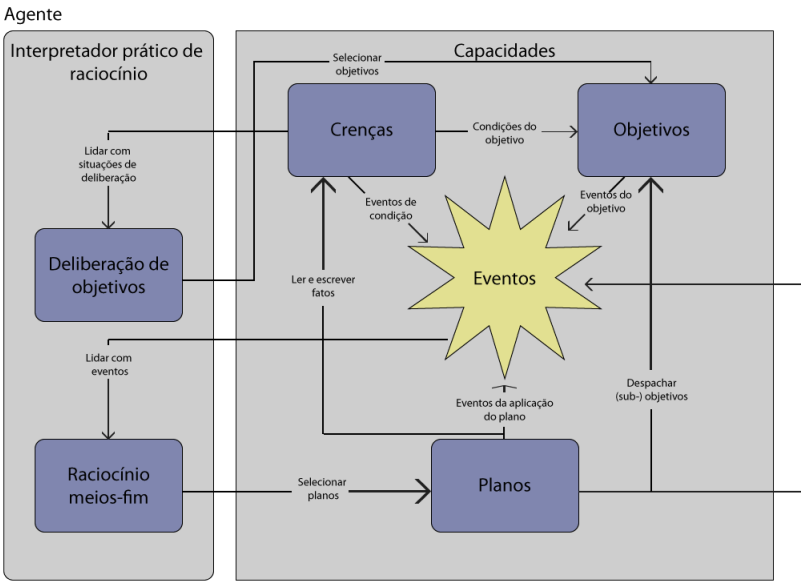



Figura A.2: Os componentes internos de um agente JADEX com raciocínio BDI. Adaptado de [5]

### A.2.2 Crenças

As crenças são um conjunto de variáveis que armazenam os fatos considerados verdadeiros pelo agente. Elas podem ser tipos primitivos Java (por exemplo, *int, byte, long*) ou classes (como listas, pontos de um plano cartesiano, imagens), e durante a execução, podem ser eliminadas do conjunto ou adicionadas a ele, de acordo com a percepção do ambiente quanto com a própria execução dos planos. O JADEX possui a Linguagem de Busca de Objetos (*Object-Query Language*, OQL), que facilita a busca de um fato em uma lista de crenças, e ainda pode utilizar a transição de estado das crenças, assim como sua adição ou remoção do conjunto, para ativar planos de ação ou objetivos.

### A.2.3 Objetivos

JADEX se utiliza da ideia que objetivos são concretos e representam um desejo momentâneo de um agente. A fim de sastifazer esse desejo, o agente tenta tomar ações que o levam a conquistar esse objetivo, até que ele seja concluído ou até que seja percebido a impossibilidade de completude ou o desinteresse em seu alcance.

Quatro tipos de objetivos são possíveis no JADEX, a saber:

1. Objetivo a Ser Realizado (*Perform Goal*): É um objetivo que não necessariamente traz algum resultado concreto, mas é algo que precisa ser realizado com sucesso, como uma rotina de atividades que precisa ser feita. Por exemplo, considera-se um agente que precisa informar a outros agentes que ele está encerrando um serviço de compra de bens. Ele pode adotar um objetivo o qual é o de avisar a todos os agentes sobre a finalização do seu serviço de aquisição;

2. Objetivo a Ser Alcançado (*Achieve Goal*): Trata-se de um objetivo que busca um estado específico no ambiente, em si próprio ou em qualquer variável abstrata presente. A ilustrar, considere um agente em um jogo de futebol. Seu objetivo pode ser ganhar a partida, ou seja, alcançar um estado em que seu time é vencedor do jogo.

3. Objetivo de Busca (*Query Goal*): Quando um agente necessita ser informado sobre um fato ou um evento, ele pode se comprometer com um objetivo de busca, que é realizado quando a informação desejada é obtida. A exemplificar, um agente precisa saber outro ainda está oferecendo o serviço de compra de bens em um determinado momento, e pode admitir um objetivo de busca para esse fim.

4. Objetivo de Manutenção (*Maintain Goal*): É o objetivo que demonstra o desejo do agente em manter constante um determinado estado do ambiente, de si próprio ou de qualquer variável abstrata ou concreta do mundo. Para mostrar como seria

o funcionamento desse objetivo, considera-se um agente que regula o conforto de pessoas em um cômodo. Seu objetivo pode ser o de manter a temperatura do ar condicionado do local a um nível que seja agradável a todos os presentes.

Sob a perspectiva da implementação, objetivos são objetos Java que possuem atributos e métodos próprios, e podem retornar resultados ao fim de sua execução. Eles podem possuir rotinas internas que são ativadas quando um evento ocorre, ou quando o objetivo atinge um estado de sucesso, falha ou desistência. Esses estados inclusive podem representar eventos internos ao agente, que delibera a respeito da próxima atitude baseado no resultado obtido. Os objetivos também podem ser insistidos pelo agente, caso eles falhem e uma nova tentativa de alcançá-los seja feita, ou eles sejam objetivos de manutenção.

### A.2.4 Planos de Ação

As ações a serem tomadas para a conquista de um objetivo são agrupadas em conjuntos ordenados denominados planos de ação. No JADEX, é permitido definir vários planos para um mesmo objetivo, cada um com um conjunto de ações distintas, conferindo um espaço de possibilidades maior para o agente atingir a meta desejada. Quando há mais de um plano para um objetivo, eles são selecionados pelo agente de acordo com as pré-condições que eles exigem, a prioridade de cada um ou o evento o qual cada um é mais adequado.

Por exemplo, considera-se novamente um agente que regula racionalmente o conforto de pessoas em um cômodo, controlando variáveis como temperatura, umidade, luminosidade e ruído. Considera-se também que ele adotou um objetivo de manutenção do conforto dos presente no cômodo. Ele possui vários planos de ação, dentre eles, o plano A, de abaixar a temperatura até o esfriar o cômodo, o plano B, de manter a temperatura mediana, ou o plano C, de aumentar a temperatura. O plano A é ativado quando não há pessoas enfermas ou de saúde frágil no recinto e a temperatura exterior está elevada, o plano B é ativado quando há pessoas enfermas ou de saúde frágil no recinto, e o plano C é escolhido quando não há pessoas enfermas ou de saúde frágil no recinto e a temperatura exterior está baixa. Uma pessoa que está se recuperando de uma cirurgia entra no recinto, e a temperatura exterior está baixa. Como o plano B é o único que as pré-condições são satisfeitas, ele é escolhido e executado. Em uma segunda situação, uma pessoa saudável entra no recinto. Os planos A e C podem ser ativados. O agente então pode adotar um sub-objetivo de busca para verificar qual se a temperatura está elevada ou baixa do lado de fora, e assim escolher entre os planos A e C.

### A.2.5 Eventos

Mudanças no ambiente percebidas pelos agentes, atualização de crenças ou objetivos, mensagens de outros agentes ou execução de planos podem gerar eventos. Eles são importantes na etapa de raciocínio meios-fim do agente, uma vez que ativam quais ações mais adequadas a serem tomadas diante do que foi ocorrido e percebido pelo agente. No JADEX, mensagens e mudanças no ambiente são considerados eventos externos, enquanto atualização de crenças e objetivos e eventos provocados por planos são eventos internos.

Na desenvolvimento do código do agente, é possível incluir um mapeamento de quais tipos de eventos que tem importância para o agente e quais podem ser ignorados. Uma vez que o Sistema Multiagente procura soluções para um problema de forma distribuída entre os agentes, é importante a distinção entre os eventos importantes ou não, uma vez que os agentes podem dividir o tratamento dos acontecimentos de forma coordenada e modular. Mais uma vez considerando o agente que regula o conforto dos residentes de um recinto: para ele, o evento de uma pessoa entrar o deixar o recinto é importante, pois cada pessoa tem necessidades que são pré-condições para a tomada de suas atitudes. No entanto, outros eventos, como o baixo nível de mantimentos no frigobar do recinto são ignorados por ele, uma vez que ele está comprometido em cuidar da temperatura, umidade, luminosidade e ruído. Um segundo agente que compromete-se com a qualidade e quantidade de alimentos disponíveis para as pessoas presentes, no entanto, consideraria esse evento importante.

## A.3 Processamento Digital de Imagens

O Processamento Digital de Imagens (PDI) é um ramo do Processamento Digital de Sinais, tendo por objeto de estudo a imagem [91]. Seu interesse e seus métodos são direcionados a dois principais objetivos: A melhoria da informação ilustrativa para uso humano ou o processamento para armazenamento, transmissão ou interpretação por um computador. Em relação as diversas aplicações, podem ser em três níveis: baixo, médio e alto. No primeiro, ambas entrada e saída são imagens; No segundo, a saída representa uma segmentação da imagem de entrada em alguma área de interesse para uma outra aplicação; No terceiro, alguma função cognitiva é aplicada, seja para realizar uma segmentação quanto para interpretá-la, utilizando funções dentro da área de Visão Computacional [92].

### A.3.1 Tratamento de Camadas

No MASE, o Agente de Processamento de Imagens (API) é responsável pelo pré-processamento das imagens submetidas pelo usuário. Elas podem ser de três tipos: mapa de estado inicial, variáveis proximais e políticas de uso do solo. O formato de entrada é sempre BMP, e a saída é uma matriz específica para casa caso. O API também é responsável pela geração do extrato de simulação, explicado na Seção A.3.2.

A biblioteca utilizada para a realização das operações específicas com imagens é a ImageJ, construída em Java e de domínio público [93]. A escolha dela é pela fácil integração com o sistema e pela liberdade de uso, possibilitando o desenvolvimento sem necessidade de software propietário.

O mapa de estado inicial é uma imagem que apresenta o espaço a ser começada a simulação (Figura A.3). Ela é basicamente lida sequencialmente e classificada de acordo com as cores dos espaços:

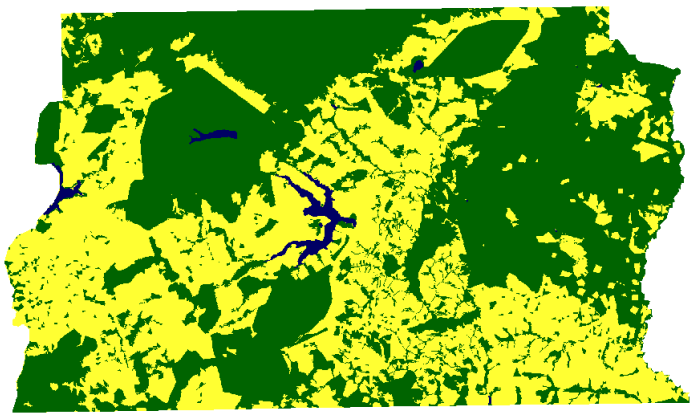


Figura A.3: Espaço inicial de uma simulação.

1. Branco: bordas da figura, representadas pelo código hexadecimal RGB 0xFFFFFF. São marcadas como espaços a serem ignorados pelos agentes transformadores;
2. Preto e azuis: variáveis proximais sem especificação, representadas pelos códigos hexadecimais RGB 0x000000 e 0x0000FF. São também marcadas para serem desconsideradas pelos agentes. No processo de obtenção dessas imagens, é comum essas variáveis aparecerem, uma vez que não foram removidas das camadas georreferenciadas originais.
3. Verdes: mata bruta, codificada pelo hexadecimal 0x006400. São atribuídos a esses espaços potencial de uso da terra de 1500 (Seção 4.1.1).

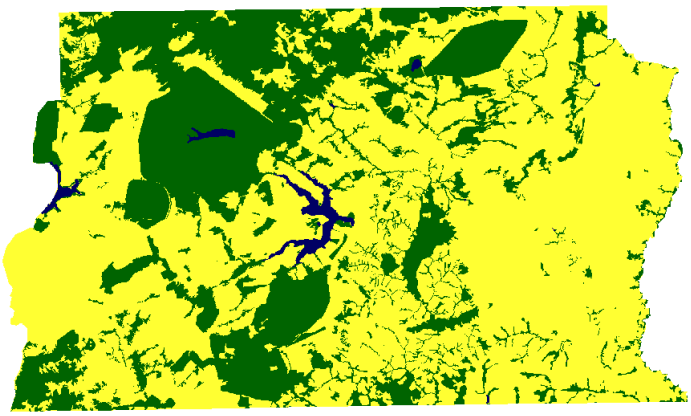


Figura A.4: Espaço final de uma simulação.

4. Amarelas: espaço antropizado, representado pelo código hexadecimal 0xFFFF00. São atribuídos a esses espaços potencial de uso da terra de 500 (Seção 4.1.1).

As variáveis proximais são filtradas de forma a criar a Matriz Proximal. Um exemplo de variável proximal é mostrado na Figura A.5, no caso, Unidades de Conservação no Distrito Federal. Deve-se fazer com que os valores ao redor das variáveis seja alto quão mais próximo for, mas na extensão delas seja nulo. Para isso, utilizou-se um filtro gaussiano para borrar a imagem e assim alterar os valores próximos as bordas da imagem original. Após esse processo, inverteu-se os valores de pixel da imagem e retirou-se os valores originais. Assim, as áreas mais próximas de 0 são pouco atrativas, enquanto as mais esbranquiçadas possuem valor de atração mais alto para o cálculo da Matriz Proximal, mostrada na Figura A.6 (foi aplicado maior contraste a essa Figura intencionalmente para melhor visualização).

Quanto a políticas de uso do solo, a Figura 4.3 mostra um exemplo de polígono de influência para filtragem. É um processamento bastante simples: cada cor recebe um rótulo diferente. A imagem é lida sequencialmente e à cor vermelha é atribuído o rótulo A; à azul, B; e à verde, C. A seguir, essa matriz de rótulos é repassada para pós-processamento do Gerente de GRID, que disponibiliza para o Gerente de Célula aplicar os multiplicadores da Matriz Proximal.

O API é um agente reflexivo, conforme classificação apresentada na Seção 2.1.1. Suas ações são basicamente receber requisições do Gerente de GRID e repassá-las, e são simples condicionais como descreve o Algoritmo 2. Caso a imagem seja um mapa de estado inicial, a ação é a filtragem 1; variável proximal, 2; política de uso do solo, 3.

Figura A.5: Variável Proximal a ser filtrada.

### A.3.2 Tratamento de Resultados nos Cenários 1 e 2

Os experimentos executados e seus resultados, apresentados no Capítulo 3, tiveram os extratos produzidos pelo API. O processamento é feito comparando quanto os agentes exploraram com a somatória dos espaços antropizados na imagem simulada com a imagem original submetida pelo usuário. A Figura 5.8c mostra, por exemplo, a imagem simulada no experimento 3, na simulação com 150 agentes sem variação de comportamento exploratório. Os número de áreas vermelhas (agricultura), azuis (pecuária) e amarelas é contado e armazenado. A seguir, o número de áreas amarelas (espaço final real) é contado e armazenado. Esses dois número são comparados, subtraindo-se um do outro e dividindo pela quantidade de áreas amarelas do espaço final real. Esse é o erro relativo da simulação. O sinal do erro é desconsiderado, e esse valor é então é multiplicado por 100, para obter-se o percentual, e subtraído de 100, para obter-se o percentual de acerto da simulação. Assim é averiguado se os agentes exploraram a uma taxa relativa coerente com a realidade, para validação da ferramenta.

Figura A.6: Variável Proximal filtrada.