

# Ferramentas de Simulação Multiagente: um estudo comparativo

Ricardo A. Machado, Gúlia B. Silveira, Marla P. Melo, Diana F. Adamatti, Cleo Z. Billa

Programa de Pós-Graduação em Computação (PPGComp)

Universidade Federal do Rio Grande (FURG)

Rio Grande – RS – Brazil

Emails: {ricardoarend, contateagiulia, marlamelo.sinfo}@gmail.com,  
{dianaadamatti, cleobilla}@furg.br

**Resumo**—O uso de ferramentas para auxiliar no processo de modelagem e desenvolvimento de simulações é muito importante, pois agiliza todo processo, bem como tende a minimizar erros. Desta forma, seu estudo e análise são vitais para a escolha da ferramenta que melhor atende aos requisitos desejados para a simulação a ser realizada.

Este artigo apresenta um estudo comparativo entre quatro ferramentas de simulação multiagente, JADE, MESA, CORMAS e NetLogo aplicadas a um mesmo problema. O objetivo é apresentar uma descrição de cada ferramenta, realizar o desenvolvimento de uma mesma aplicação em cada ferramenta e executar simulações. Todo o processo de desenvolvimento até a execução é utilizado para fins comparativos em diferentes parâmetros.

## I. INTRODUÇÃO

A Simulação Multiagente (*Multi-Agent-Based Simulation - MABS*) surge da união das tecnologias de Simulação e Sistemas Multiagente (SMA) [1]. A simulação é uma excelente forma de modelar e entender os processos sociais, pois pode-se analisar aspectos de emergência relacionada a atividades simples. É uma área de pesquisa interdisciplinar, que busca construir dinamicamente a teoria científica, enquanto compreendemos melhor o problema através de simulação multiagente [2].

O uso de ferramentas para auxiliar no processo de modelagem e desenvolvimento das simulações é muito importante, pois agiliza todo processo, bem como tende a minimizar erros. Na área de MABS, diversos grupos de pesquisa desenvolvem e disponibilizam ferramentas. Neste artigo, o objetivo principal é apresentar uma comparação entre quatro ferramentas de simulação, utilizando um mesmo “problema” para testes: “Pegue o Porco”, onde dois agentes fazendeiros tentam capturar um terceiro agente denominado porco. Em Machado et al. [3], os autores apresentam um estudo preliminar sobre ferramentas de simulação multiagente.

As ferramentas comparadas neste artigo são:

- O JADE<sup>1</sup> é uma API (*Application Programming Interface*) da linguagem Java para desenvolvimento de agentes baseada nas especificações da FIPA (*Foundation for Intelligent Physical Agents*);

- O MESA<sup>2</sup> é uma ferramenta de modelagem baseada em agentes na linguagem Python, projetada para realizar simulações customizadas que podem ser visualizadas em navegador web;
- CORMAS<sup>3</sup> é uma plataforma de simulação baseada no ambiente de programação VisualWorks, para linguagem Smalltalk, com foco no desenvolvimento de aplicações para gestão de recursos naturais;
- NetLogo<sup>4</sup> é um ambiente de programação e modelagem de simulação multiagente.

O método proposto para realizar a comparação entre as ferramentas citadas acima, refere-se à especificação de cinco parâmetros para a avaliação das ferramentas e a verificação em termos do desempenho obtido por meio dos resultados da média e desvio padrão da simulação do problema “pegue o porco”, descrito na seção 3.

O artigo está estruturado da seguinte forma: na seção 2 são descritas as ferramentas utilizadas. Na seção 3 é apresentado o problema “pegue o porco” e o processo de desenvolvimento em cada ferramenta. Na seção 4 são apresentadas as vantagens e desvantagens entre as ferramentas, como também, os resultados da simulação e por fim, na seção 5 são apresentadas as conclusões desta análise comparativa entre as ferramentas de simulação baseadas em agentes.

## II. DESCRIÇÃO DAS FERRAMENTAS UTILIZADAS

De forma geral existem dois tipos de ferramentas disponíveis para o desenvolvimento de modelos baseados em agentes: kits de ferramentas e softwares [4].

Os kits de ferramentas, também conhecidos por *frameworks*, fornecem bibliotecas com classes e funções predefinidas, projetadas especificamente para o desenvolvimento de simulações baseadas em agentes. Apesar de apresentarem a desvantagem da alta curva de aprendizagem de uma linguagem, os *frameworks* reduzem esforços com a programação de interfaces gráficas com usuário, importação e exportação de dados e visualização do modelo [4]. Outro aspecto positivo é de que o paradigma orientado à objetos permite estender a capacidades

<sup>2</sup><https://github.com/projectmesa/mesa>

<sup>3</sup><http://cormas.cirad.fr/indexeng.htm>

<sup>4</sup><https://ccl.northwestern.edu/netlogo/>

<sup>1</sup><http://jade.tilab.com/>

dos *frameworks*, integrando novas funcionalidades particulares às bibliotecas.

Em contraste com os kits de ferramentas, os softwares simplificam o processo de implementação, por não exigirem conhecimento em linguagens de programação. Softwares são úteis para o desenvolvimento de modelos básicos ou protótipos em ambientes mais limitados ou restritos à funcionalidade fornecida pelo mesmo [4].

Neste artigo, são comparados os kits de ferramentas MESA, JADE, CORMAS e NetLogo.

#### A. JADE

JADE (*Java Agent Development Framework*) é um *framework* Java completo que trata da comunicação, do ciclo de vida do agente, do monitoramento da execução, entre outras atividades e que tem como objetivo facilitar o desenvolvimento de aplicações multi-agentes em conformidade com as especificações FIPA (*Foundation for Intelligent Physical Agents*) [5].

FIPA é uma associação internacional sem fins lucrativos de empresas e organizações que compartilham o esforço para produzir especificações para tecnologias de agentes genéricos. O FIPA-ACL é uma linguagem que descreve a codificação e semântica de mensagens, mas não exige mecanismos específicos para o transporte de mensagens sendo essa linguagem utilizada por diferentes SMA incluindo o JADE.

O JADE é uma das ferramentas mais utilizadas para o desenvolvimento de sistemas baseados em agentes. De acordo com [6], JADE foi escrito em Java devido a características particulares da linguagem, especificamente pela programação orientada a objetos em ambientes distribuídos heterogêneos. Foram desenvolvidos tanto pacotes Java com funcionalidades prontas pra uso, quanto interfaces abstratas para se adaptar de acordo com a funcionalidade da aplicação de agentes.

Embora pareça uma entidade única para o mundo externo, uma plataforma de agente JADE é, por si só, um sistema distribuído. Um sistema JADE compreende um ou mais contêineres de agentes, cada um numa máquina virtual Java de forma independente [7].

#### B. MESA

O MESA é um *Framework* em Python de código aberto que permite aos usuários criar rapidamente modelos baseados em agentes. A partir dos conceitos do paradigma de programação orientada a objetos, o MESA distribui seus componentes dentro de três módulos que podem facilmente ser combinados e ampliados para construir diferentes tipos de simulações [8]. Os módulos são divididos em três categorias gerais: modelagem, análise e visualização. Em cada módulo, existem conjuntos de classes com atributos e funções predefinidos, que podem ser herdados durante o desenvolvimento de uma simulação.

No módulo de modelagem encontram-se classes destinadas a definição de parâmetros das simulações, são elas; modelo, agentes, agendador e espaço. De forma geral, uma simulação consiste da instância da classe modelo que representará o funcionamento da simulação; uma ou mais instâncias da classe

de agente; uma instância de agendador, que gerencia a ativação de agentes de forma totalmente personalizável; e um espaço que possibilita que os agentes se movimentem e interajam com vizinhos.

A análise de informações após uma simulação ou um lote de simulações é facilitada, através dos componentes fornecidos no módulo de análise. Nesta categoria, encontram-se os coletores de dados, usados para registrar dados de cada execução do modelo ou de um lote de execuções com registros detalhados sobre variáveis do modelo, bem como dos agentes [8].

Um modelo baseado em agente não é particularmente útil se não houver maneira de ver os resultados produzidos pelas simulações. No módulo de visualização, o *framework* oferece a visualização dos resultados em interface gráfica ou através de coleta de dados quantitativos. Para facilitar a última opção, são fornecidas classes genéricas que podem armazenar e exportar dados de variáveis do modelo, variáveis do agente e tabelas que são um resumo da simulação [8].

A interface gráfica pode ser construída livremente utilizando *Javascript*, *Hypertext Markup Language (HTML)* e *Cascading Style Sheets (CSS)* e sua exibição ocorre em um navegador web que recebe dados através de um servidor criado pelo MESA.

O MESA é o primeiro *framework* para MABS na linguagem Python. Por estar sendo construído do zero, tem-se a viabilidade de incorporar novos recursos que ampliem a diversificação de representação das simulações baseadas em agentes.

#### C. CORMAS

O ambiente de simulação multiagente CORMAS (*Common-Pool Resources and Multi-agent Systems*) [9] foi desenvolvido pelo grupo Green-CIRAD, principalmente para simulação de recursos naturais [10], sendo útil para o melhor entendimento das interações complexas entre a dinâmica natural e social, quando se estuda o gerenciamento de recursos renováveis [9].

CORMAS foi desenvolvida a partir do ambiente *Visualworks*, e para o desenvolvimento de modelos, é utilizada a linguagem de programação orientada a objetos *Smalltalk*. Além disso, é possível programar e importar, para a plataforma CORMAS, os códigos na linguagem dinâmica e reflexiva *Pharo*<sup>5</sup>, inspirada na linguagem de programação *Smalltalk*.

A plataforma CORMAS é completa, permite vários agentes, comunicação, movimentação e também possibilita o monitoramento e análise de simulações, bem como a observação de simulações com a noção de ponto de vista (*pov*) que permite ao modelador do sistema multiagente, definir a observação de uma parte do espaço ou de um agente em diferentes pontos de vista [9]. Além disso, possui integração com *R*<sup>6</sup>, permitindo a geração de gráficos e análises estatísticas complexas.

#### D. NetLogo

O NetLogo é uma ferramenta independente, desenvolvida em Java, para ser executado nas principais plataformas. Além

<sup>5</sup><https://pharo.org/>

<sup>6</sup><https://www.r-project.org/>

de possuir licença *freeware*<sup>7</sup>, apresenta uma comunidade de usuários ativa, que oferece uma extensa documentação com tutoriais e coleções de modelos aplicados à diversos contextos que se destacam em relação à outras ferramentas [11].

NetLogo também é uma linguagem de programação<sup>8</sup> incluída em um ambiente de modelagem multiagente para simulação de fenômenos. Historicamente, é caracterizada como a próxima geração de linguagens de modelagem de sistemas de agentes após a StarLogo desenvolvida por [12].

Famosa por sua simplicidade e interface gráfica amigável, a ferramenta foi projetada para pesquisa e é utilizada em uma ampla gama de disciplinas e níveis educacionais, pois não demanda dos pesquisadores conhecimentos avançados em programação.

Através da ferramenta é possível desenvolver instruções e variáveis para milhares de agentes independentes que operam simultaneamente, o que possibilita que sejam exploradas conexões entre comportamentos a micro-nível de indivíduos e padrões de nível macro que emergem das interações entre os agentes.

Os agentes criados na ferramenta são nomeados *turtles* (tartarugas), enquanto cada posição da matriz onde eles existem é nomeada *patch*(terreno). Tanto agentes quanto terrenos possuem atributos básicos que podem ser estendidos pelo pesquisador/modelador [12].

O NetLogo possui uma unidade temporal chamada *tick* (período). Um período compreende a unidade de tempo real necessária para que todos os agentes possam executar o suas instruções. A ferramenta suporta a criação de componentes gráficos que manipulem valores dos parâmetros da simulação [12].

Para usuário avançados, são oferecidos recursos extras na ferramenta tais como *BehaviorSpace*, que executa experimentos automatizados, visualização 3D, extensibilidade do usuário; *System DynamicsModeler*, que permite misturar representações agregadas e baseadas em agentes; e o *NetLogoLab*, que se conecta a dispositivos físicos externos.

Outro aspecto relevante da ferramenta é que os dados gerados a partir das simulações realizadas no NetLogo podem ser manuseados utilizando recursos da linguagem que incluem comandos que permitem ler, escrever, exportar e importar dados em formatos padrões de forma com que o pesquisador consiga explorar seus resultados em diferentes aplicações [13].

### III. DESENVOLVIMENTO DO CENÁRIO PEGUE O PORCO

Para efetuar a comparação das ferramentas, foi desenvolvido o mesmo problema em cada uma das ferramentas apresentadas. Esse problema, denominado “Pegue o Porco”, é composto por um tabuleiro 5x5, conforme Figura 1. Nele, existe um agente denominado porco e dois agentes fazendeiros. O objetivo é fazer com que os agentes fazendeiros colaborem e capturem o porco, da maneira mais eficiente possível.

<sup>7</sup>Licença Freeware: a utilização do software não implica no pagamento de licenças de uso ou royalties.

<sup>8</sup>NetLogo é considerada uma linguagem da família Lisp concebida por John McCarthy em 1958.

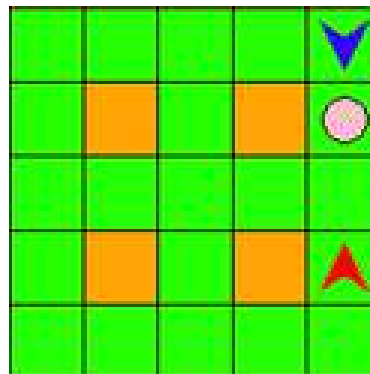


Figura 1. Cenário Proposto

O problema proposto tem como especificações os seguintes requisitos:

- 1) A posição inicial de cada um dos três agentes deve ser aleatória;
- 2) O ambiente é completamente observável, ou seja, todos os agentes sabem a posição de todos;
- 3) O número máximo de movimentos (N) é um parâmetro definido pelo usuário;
- 4) Cada agente se move um quadrado de cada vez e só pode se movimentar para os quadrados adjacentes a sua posição atual. Não é permitido andar na diagonal;
- 5) A movimentação é feita em turnos, uma vez para cada;
- 6) O agente porco é um agente autônomo que deve sempre fugir do fazendeiro mais próximo. Caso o porco não possa se afastar, ele fica imóvel;
- 7) Os agentes fazendeiros também são autônomos e devem colaborar, a fim de cercar o porco para que ele possa ser capturado;
- 8) A simulação acaba quando um dos agentes fazendeiros captura o porco.

A seguir, é detalhado o processo para modelagem e desenvolvimento em cada ferramenta e suas características.

#### A. JADE

O cenário foi desenvolvido a partir de uma classe principal Java chamada Tabuleiro, que gera uma matriz 5x5. A posição inicial dos agentes é gerada de forma randômica. Essa classe também contém diferentes funções que são utilizadas pelos agentes para mapeamento e movimentação dos mesmos. Além disso, uma interface gráfica 2d foi desenvolvida utilizando uma biblioteca Java para visualização do ambiente.

A Figura 2 mostra o ambiente desenvolvido com dois fazendeiros e o porco. Nela, pode-se visualizar o tabuleiro, que contém quatro obstáculos formados pelos quadrados pretos que os agentes não podem entrar. Quanto a estratégia utilizada para capturar o porco: os fazendeiros, ao iniciar seus respectivos turnos calculam a distância em relação ao porco e também verificam qual o melhor caminho a seguir com base na localização do outro fazendeiro no sentido de encurralar o porco. Já o porco escolhe sempre o caminho que contém

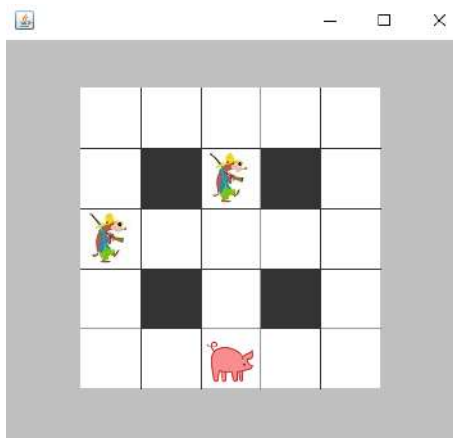


Figura 2. Cenário no JADE

a maior distância em relação aos dois fazendeiros. Nesse exemplo, não foi necessária a utilização da comunicação entre os agentes, porém essa é uma das futuras melhorias que serão realizadas no projeto.

Em JADE, cada tipo diferente de agente é desenvolvido a partir de uma classe própria. Nesse exemplo, foi criada uma classe para cada agente JADE e elas são responsáveis pelo comportamento cíclico do agente e chamamento dos métodos que foram desenvolvidos na classe Tabuleiro. Também foi necessário criar um agente “genérico”, que inicializa um tabuleiro único e pode ser estendido pelos outros agentes evitando assim problemas de redundância dos dados <sup>9</sup>.

### B. MESA

O cenário no MESA pode ser entendido como um elemento espacial, onde os agentes podem ocupar posições e realizar interações com outros agentes e outros objetos [8]. Esses espaços podem ser representações abstratas, como um autômato celular utilizando modelos toroidais, por exemplo; ou escalar, com representações de cidades ou regiões do mundo.

O MESA atualmente implementa duas grandes classes de espaço: o espaço grade e o espaço contínuo. No espaço grade, os agentes e outros objetos só podem estar em uma célula particular (podendo abranger várias células); já no espaço contínuo podem ocupar qualquer posição arbitrária [8].

Existem várias classes de espaços específicas, todas herdadas de uma classe mãe. No núcleo da classe mãe os espaços são representados por uma matriz bidimensional que possui métodos para obtenção de vizinhos, adição e remoção de agentes. Para definir o conteúdo das células da matriz podem ser utilizadas as classes *MultiGrid* ou *SingleGrid* [8]. No *MultiGrid*, vários objetos podem compartilhar uma célula, enquanto no *SingleGrid* no máximo um objeto pode estar contido em uma célula.

Para modelagem do problema “Pegue o Porco”, instanciou-se um cenário de espaço contínuo, de forma que os agentes

<sup>9</sup>Todo o código fonte do projeto aqui descrito está disponibilizado para visualização via github: <https://github.com/ricardoarend/CatchPig>

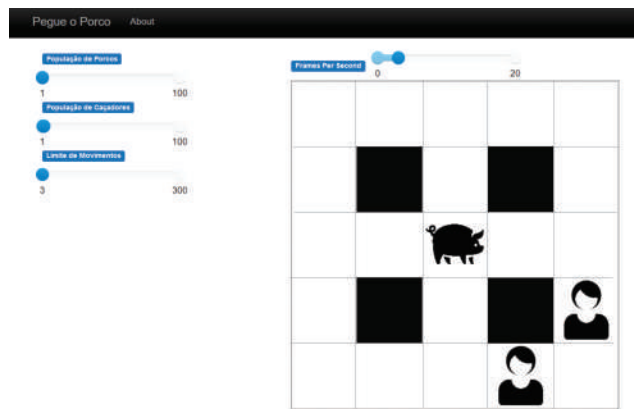


Figura 3. Cenário no MESA

tenham posicionamento inicial randômico e movimentação arbitrária entre as células vizinhas. Utilizou-se a ocupação espacial *MultiGrid* pois, o porco é capturado quando existe uma célula com um agente do tipo fazendeiro e um agente do tipo porco.

A posição dos agentes é armazenada duas vezes, como uma tupla: uma vez é registrada no gráfico da célula e outra nos atributos do agente. Para cada agente definir sua melhor coordenada para movimentação, de acordo com sua estratégia, criou-se um raio de percepção do ambiente.

Por fim, utilizando os recursos do *framework* para visualização da simulação, criou-se uma interface para o problema do porco, com botões para parametrizar a população de agentes, a quantidade de movimentos e a quantidade de *steps* de execução. A interface gráfica da simulação pode ser observada na Figura 3.

### C. CORMAS

A Figura 4 exhibe a implementação do cenário na ferramenta CORMAS.

Na plataforma CORMAS existem basicamente três tipos de entidades genéricas: “agente social”, “entidade espacial” e “entidade passiva” [14].

- Na classe da entidade social são definidos os agentes que se comunicam e interagem com outros agentes, nesse caso, os agentes fazendeiros e o porco;
- A classe da entidade espacial define o ambiente e os elementos que estão localizados nele e dá suporte topológico para simulações, arbitrando na alocação de recursos naturais de acordo com protocolos pré-estabelecidos [15]. Nesse modelo, na entidade *cell*, está definido o *grid* 5x5, onde os obstáculos são definidos e os agentes são situados aleatoriamente;
- E na classe da entidade passiva estão as mensagens de comunicação e objetos. No modelo, os obstáculos situados no ambiente para os agentes.

As células são organizadas hierarquicamente. Desse modo, a entidade *cell*, definida na entidade espacial, os agentes dos tipos fazendeiro e porco da classe social e os objetos obstáculos

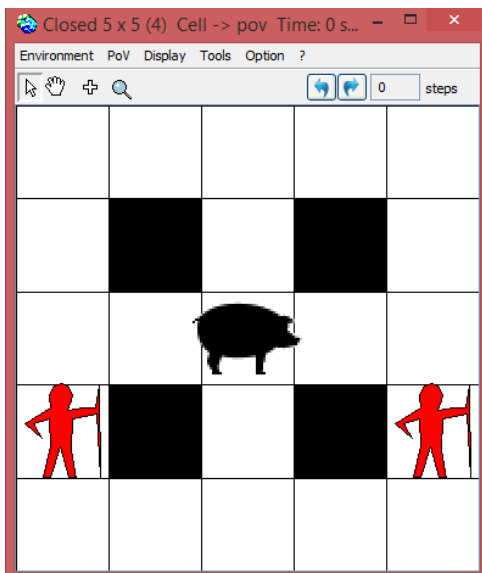


Figura 4. Cenário no CORMAS

definidos na entidade passiva, são subclasses que descrevem o estado atual das células. Estes estados correspondem a implementação dos métodos dos agentes, isto é, as ações que cada agente poderá realizar no ambiente.

Os autores [9] definem um *patch* como uma “classe pré-programada e proposta para a herança”. Esta classe inclui atributos que definem a vizinhança (ordenada ou não, em um raio de percepção variável). Cada *patch* possui um atributo *lesOccupants* que contém automaticamente a lista de agentes localizados nele, classificados por tipo de agente. Assim, para efetuar a captura do porco, o agente fazendeiro escolhe uma célula dentro do raio de percepção que contenha um agente porco.

#### D. NetLogo

Para a construção do cenário no NetLogo foram atribuídas cores para determinados *patches* que representam as bordas e os obstáculos do ambiente. A posição inicial dos agentes é feita randomicamente para *patches* vazios e que tenham cor diferente dos obstáculos. Na Figura 5 é possível ver o cenário pronto para iniciar a simulação, sendo para isso necessário clicar no botão “go” à esquerda. O botão “setup” é responsável por reiniciar o cenário.

A movimentação dos agentes é dada da seguinte maneira: os agentes fazendeiros calculam a distância entre o porco e cada um dos possíveis *patches* disponíveis próximo a ele, e escolhem o *patch* que tem menor distância para se direcionar, sendo que só podem andar uma casa por *tick* (tempo da simulação). O porco, por sua vez, realiza o mesmo cálculo, porém escolhendo o *patch* com maior distância para o fazendeiro mais próximo.

A distância é calculada pela função “distance” disponível na biblioteca do NetLogo, que retorna a distância euclidiana entre dois objetos no mapa, podendo essa distância ser entre

um *patch* e um agente. O porco é capturado toda vez que um fazendeiro fica a um *patch* de distância dele.

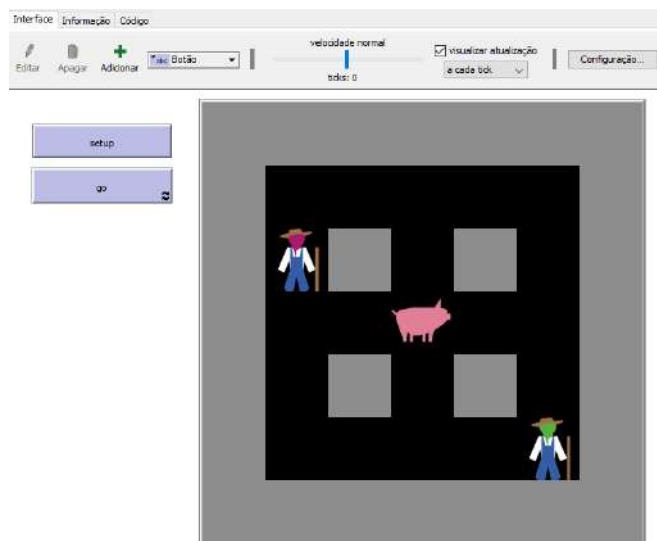


Figura 5. Cenário no NetLogo

#### E. Resultados

Para cada ferramenta foram realizadas 50 simulações, com dois agentes fazendeiros e um agente porco. A Tabela I apresenta a média e o desvio padrão do número de movimentos realizados até a captura do porco em cada ferramenta. O número de movimentos contados é resultado da quantidade de turnos onde cada um dos agentes fez uma movimentação.

Tabela I  
RESULTADOS DAS SIMULAÇÕES

Ferramenta	Média	Desvio Padrão
JADE	5,54	1,98
MESA	5,82	5,05
CORMAS	25,28	24,74
NetLogo	4,28	2,51

Percebe-se que a implementação com maior estabilidade é o JADE, onde a média de movimentos dos agentes foi de aproximadamente 5, com um desvio padrão próximo de 2. O NetLogo teve valores próximos ao JADE, onde obteve-se uma média menor, mas um desvio padrão maior.

No caso do MESA, o número médio de movimentos foi também próximo a 5, como nas simulações realizadas do JADE, porém com um desvio padrão mais alto (próximo de 5), o que indica uma maior variação no número de movimentos necessários para encerrar a simulação.

No caso do CORMAS, houveram simulações muito eficientes e outras muito ruins, já que o desvio padrão ficou próximo a média.

#### IV. VANTAGENS E DESVANTAGENS ENTRE AS FERRAMENTAS ANALISADAS

Com a finalidade de comparar as quatro ferramentas de simulação baseadas em agentes, foram estabelecidos os seguintes parâmetros:

- 1) Material disponível para consulta;
- 2) Complexidade de modelagem da interface;
- 3) Complexidade de programação: métodos pré-prontos/ métodos genéricos;
- 4) Ambiente de modelagem/desenvolvimento;
- 5) Interface gráfica para visualização das simulações.

A Tabela II apresenta esses parâmetros para cada uma das ferramentas. Abaixo são apresentados os motivos que levaram a tais classificações.

Tabela II  
COMPARAÇÃO ENTRE AS FERRAMENTAS

Parâmetro	JADE	MESA	CORMAS	NetLogo
1	✓	✓	✓	✓
2	Difícil	Fácil	Difícil	Fácil
3	Médio	Fácil	Difícil	Médio
4	Diversos	Diversos	Um	Um
5	X	✓	✓	✓

Esses parâmetros foram aplicados com base em um nível acadêmico de conhecimento e não servem como comparação para profissionais da área que tenham maior conhecimento de determinada linguagem.

##### A. JADE

A ferramenta JADE possui um bom material de apoio para iniciantes. Em sua página é possível encontrar todo tipo de tutoriais, inclusive um em português, além de algumas publicações.

No quesito interface, a principal dificuldade foi a ausência de referências para serem utilizadas como base para sua modelagem. Foi preciso utilizar mecanismos para fazer com que a interface do ambiente ficasse sempre atualizada independente de qual agente estivesse se movimentando.

Quanto a complexidade de programação, um conhecimento de java é o suficiente para conseguir desenvolver o sistema. Porém, não existem métodos prontos e tudo tem que ser feito do zero (movimentação, cenário, interação).

O ambiente de desenvolvimento foi o NetBeans, porém também é possível utilizar o Eclipse, ou qualquer outro ambiente para Java.

No JADE não existe uma interface gráfica já pronta. Porém, nada impede que seja criado usando Java, como no caso desse exemplo.

##### B. MESA

Nos últimos anos, o Python tornou-se uma linguagem cada vez mais popular para computação científica [16], apoiado por crescente conjunto de ferramentas para análise e modelagem. Python se destaca em relação ao Smalltalk por ser *mainstream*, o que é uma vantagem no aprendizado e documentação.

A interface gráfica de MESA é totalmente personalizável, podendo-se criar botões para iniciar ou encerrar uma simulação, campos para definir parâmetros de quantidade de agentes predadores instanciados e quantidade de movimentos que os agentes realizarão, por exemplo. Para criar ou realizar modificações na interface são necessários conhecimentos em programação web, contudo a ferramenta oferece interfaces de modelos de simulações que podem demandar pouca ou nenhuma modificação, como no problema tratado neste artigo.

O MESA nativamente oferece classes completas para o desenvolvimento de uma MBA, além de possibilitar a importação de bibliotecas disponíveis, deixando para o desenvolvedor apenas a implementação de funções específicas, como a estratégia de fuga do porco. Além disto, disponibiliza um pacote com diversos exemplos de simulações com abordagens em diferentes cenários.

O ambiente de desenvolvimento utilizado foi PyCharm<sup>10</sup>, mas a escolha para a IDE ideal é realizada pelo o usuário e suas preferências de desenvolvimento.

##### C. CORMAS

No ambiente de modelagem da plataforma CORMAS, em sua interface principal, está disponibilizado um glossário com informações sobre os modelos implementados na plataforma e também um glossário dos métodos de CORMAS. Além disso, não possui flexibilidade de modelagem da interface.

CORMAS possui uma estrutura de alto nível com uma vasta implementação de funções prontas e métodos gerados automaticamente que precisam ser compreendidos para serem aplicados. Pela complexidade da plataforma CORMAS e do não conhecimento da linguagem Smalltalk, o processo de desenvolvimento de modelos pode ser demorado.

Como CORMAS é uma plataforma, ela integra o ambiente de simulação e o desenvolvimento, bem como, disponibiliza uma interface gráfica para visualização das simulações e propicia recursos para análise de simulações.

##### D. NetLogo

NetLogo é uma das ferramentas mais utilizadas para simulação baseada em agentes na atualidade, visto que tem como principal facilidade a fácil implementação de uma interface gráfica (botões, gráficos, slides. etc). Desta forma, possui uma biblioteca de exemplos bastante extensa, auxiliando no desenvolvimento de novas aplicações.

Contudo, quanto a mobilidade do agente, existem poucos exemplos que possuem a restrição dos agentes seguirem por um espaço delimitado de um tabuleiro, sendo a principal dificuldade encontrada na implementação do cenário proposto. Grande parte dos exemplos utilizam o método de andar para frente ou randomicamente, efetuando rotações em ângulos aleatórios.

Quanto a métodos pré-prontos ou genéricos, a complexidade de utilização pode ser considerada média, visto que há uma mudança quanto ao paradigma de programação com a

<sup>10</sup>Disponível em: <https://www.jetbrains.com/pycharm/>

linguagem LOGO, mas a quantidade de métodos existentes é bastante ampla.

#### V. CONCLUSÕES

O artigo apresentou uma breve explanação a respeito de MABS e uma comparação das ferramentas de simulação baseadas em agentes: JADE, MESA, CORMAS e NetLogo. A análise foi feita com base no desenvolvimento da solução para o problema “Pegue o porco”.

Conforme os resultados apresentados na Tabela I, cada ferramenta apresentou um comportamento diferente em relação a média e o desvio padrão do número de movimentos para os agentes fazendeiros capturarem o agente porco. Mostrando assim a diferença entre as características específicas de cada ferramenta na implementação de uma mesma solução para um mesmo problema.

Observa-se que as ferramentas JADE, MESA e NetLogo apresentaram resultados semelhantes. Neste contexto, essas ferramentas mostraram-se eficazes para o problema específico “Pegue o Porco”. No entanto, a plataforma CORMAS teve um destaque negativo, visto que o propósito da plataforma CORMAS é facilitar o desenvolvimento de modelos baseados nas interações entre as dinâmicas naturais e sociais no domínio da gestão de recursos naturais, sem preocupação com desempenho.

Em relação a simulação na Tabela I, pode-se verificar que JADE e NetLogo obtiveram os melhores resultados. Como suas médias e desvios padrão foram bastante próximos, não é possível distinguir o desempenho dos dois.

Já em relação a comparação das ferramentas em si, onde foram apresentados alguns parâmetros, a Tabela II sumarizou o comparativo, demonstrando que cada ferramenta tem vantagens e desvantagens, porém a ferramenta MESA teve uma melhor avaliação ao somar todos os parâmetros analisados.

Como trabalhos futuros, vislumbra-se um cenário mais complexo, onde sejam explorados a comunicação entre os agentes e a adição de outros tipos de agentes. Esses testes visam verificar a robustez das ferramentas analisadas, ou seja, se estas têm suporte para executar essas funcionalidades.

Para finalizar, esse artigo mostrou um comparativo de quatro ferramentas para o desenvolvimento de uma solução para um problema específico, porém seu intuito é de apenas servir como auxílio para a escolha da ferramenta, já que existem muitas variáveis não avaliadas e por isso fica a cargo do desenvolvedor a escolha da ferramenta a ser utilizada.

#### REFERÊNCIAS

- [1] D. F. Adamatti, J. S. Sichman, and H. Coelho, “Utilizaç ao de rpg e mabs no desenvolvimento de sistemas de apoio a decisao em grupos,” *Anais do IV Simpósio Brasileiro de Sistemas Colaborativos SBSC 2007*, p. 15, 2007.
- [2] N. Gilbert and K. G. Troitzsch, *PSimulation for the Social Scientist*. Open University Press Milton Keynes, UK, 2005.
- [3] R. A. Machado, G. B. Silveira, M. P. Melo, D. F. Adamatti, and C. Z. Billa, “Análise comparativa entre ferramentas de simulação multiagente,” in *13th Workshop-School on Agents, Environments, and Applications*. UFSC, 2019.
- [4] C. J. Castle and A. T. Crooks, “Principles and concepts of agent-based modelling for developing geospatial simulations,” 2006.
- [5] F. Bellifemine, A. Poggi, and G. Rimassa, “Developing multi-agent systems with jade,” in *International Workshop on Agent Theories, Architectures, and Languages*. Springer, 2000, pp. 89–103.
- [6] F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa, “Jade programmer’s guide,” *Jade version*, vol. 3, pp. 13–39, 2002.
- [7] F. Bellifemine, A. Poggi, and G. Rimassa, “Jade: a fipa2000 compliant agent development environment,” in *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001, pp. 216–217.
- [8] D. Masad and J. Kazil, “Mesa: an agent-based modeling framework,” in *14th PYTHON in Science Conference*, 2015, pp. 53–60.
- [9] F. Bousquet, I. Bakam, H. Proton, and C. Le Page, “Cormas: common-pool resources and multi-agent systems,” in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 1998, pp. 826–837.
- [10] D. F. Adamatti, “Inserção de jogadores virtuais em jogos de papéis para uso em sistema de apoio a decisão em grupos: um experimento no domínio da gestão dos recursos naturais,” *Universidade de São Paulo, São Paulo*, 2007.
- [11] S. L. Lytinen and S. F. Railsback, “The evolution of agent-based simulation platforms: a review of netlogo 5.0 and relogo,” in *Proceedings of the fourth international symposium on agent-based modeling and simulation*, 2012, p. 19.
- [12] U. Wilensky and M. Resnick, “Thinking in levels: A dynamic systems approach to making sense of the world,” *Journal of Science Education and technology*, vol. 8, no. 1, pp. 3–19, 1999.
- [13] S. Tisue and U. Wilensky, “Netlogo: A simple environment for modeling complexity,” in *International conference on complex systems*, vol. 21. Boston, MA, 2004, pp. 16–21.
- [14] P. Bommela, N. Becuc, C. Le Pageb, F. Bousquetb, and G. Leclercb, “Cormas, una plataforma multiagente para la modelización interactiva,” 2017.
- [15] J. S. Sichman, F. Bousquet, and P. Davidsson, “Multi-agent-based simulation ii: Third international workshop, mabs 2002,” *Lecture Notes in Artificial Intelligence*. Springer, 2003.
- [16] F. Pérez and B. E. Granger, “Ipython: a system for interactive scientific computing,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 21–29, 2007.