Estudo Comparativo entre Mesa e Spade para Modelagem de Problema de Caixeiro Viajante

Timotio Capitão Cubaque, Johann Pinheiro Pires, Eder Mateus Nunes Gonçalves, Diana Francisca Adamatti

Programa de Pós-Graduação em Computação
Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brasil

{edergoncalves, dianaadamatti, piresjohann}@furg.br
{timotiocapitaocubaque}@yahoo.com.br

Abstract. *To assist the modeling and simulation based on agents, the choice of a tool is a necessity. However, these tools MESA and SPADE have different characteristics of methodologies and functionalities, whose careful analysis should be considered, aiming to guide their choice according to the purpose and complexity of the model to be simulated. Based on this need for evaluation, the objective of this paper is to present a brief comparative study between these tools applied to solve the traveling salesman problem.*

Resumo. *Para auxiliar a modelagem e simulação baseada em agentes, a escolha de uma ferramenta é uma necessidade. Entretanto, essas ferramentas MESA e SPADE possuem diferentes características de metodologias e funcionalidades, cuja análise criteriosa deve ser considerada, visando orientar a sua escolha de acordo com a finalidade e complexidade do modelo a ser simulado. Com base nessa necessidade de avaliação, o objetivo do presente artigo é apresentar um breve estudo comparativo entre estas ferramentas aplicadas para resolver o problema do caixeiro viajante.*

1. Introdução

A modelagem e simulação baseada em agente é uma abordagem para o desenvolvimento de sistemas compostos de agentes autônomos que interagem para simular sistemas complexos, nos quais o comportamento emergente é usualmente desconhecido. Com objetivo de apoiar a modelagem e simulação baseada em agentes foram surgindo algumas ferramentas e softwares, entre elas algumas gratuitas e outras pagas, para dar suporte e apoio nas atividades. O presente artigo tem como objetivo, portanto, realizar uma comparação entre MESA e SPADE, ferramentas que auxiliam a simulação baseada em agente, através de critérios que terão como base as suas principais características funcionais. Serão considerados, também, como critérios de avaliação: (1) flexibilidade; (2) comunidade de usuários; (3) disponibilidade de ajuda ou suporte, entre outros. A comparação é feita com base no problema de caixeiro viajante, um problema que tem servido de plataforma de teste para a investigação de diversas ideias, além de ser um problema de larga aplicabilidade no mundo real, é de fácil compreensão e descrição, mas de difícil solução, pois pertence à classe de problemas NP-Completos [Klügl and Bazzan 2012].

2. Ferramentas Escolhidas

Para o desenvolvimento de modelos baseados em agentes, de modo geral existem dois tipos de sistemas disponíveis: toolkits e softwares. As ferramentas MESA e SPADE são apresentadas a seguir como exemplos de toolkits proposto para comparação.

2.1. Mesa

O MESA é uma estrutura modular para construir, analisar e visualizar modelos baseados em agentes ou um framework de modelagem baseada em agente. Ele fornece uma estrutura para o código, bem como ferramentas analíticas e de visualização para os experimentos. Os objetivos de design do Mesa vão além da construção rápida de modelos baseados em agentes em Python, ou exibindo-os no navegador. Semelhante a outros frameworks, o Mesa está focado na funcionalidade principal necessária ao construir modelos baseados em agentes (por exemplo, objetos reutilizáveis, agendamento e interfaces gráficas de usuário), permitindo assim que os modeladores se concentrem no desenvolvimento de modelos em vez de partes do simulação que não são específicos do conteúdo. Existem três módulos principais que compõem o Mesa do ponto de vista do usuário. Estes são a modelagem, a análise e a visualização [Masad and Kazil 2015].

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.

Visualização de interação do modelo, você pode querer observar diretamente seu modelo enquanto ele é executado. A principal ferramenta de visualização do Mesa usa um pequeno servidor web local para renderizar o modelo em um navegador, usando JavaScript [Masad and Kazil 2015].

2.2. Spade

O Spade (Smart Python Agent Development Environment), é uma estrutura de agente escrita em Python para comunicação do agente. De fato, o SPADE usa o framework/protocolo XMPP como camada de comunicação, proporcionando interoperabilidade com outras entidades na Internet. Os servidores suportam um grande número de usuários e mensagens com base em XML (Extensible Markup Language). O XMPP fornece uma arquitetura de servidor federado e aberto (representado na Figura 1), pelo qual qualquer servidor XMPP pode se comunicar com qualquer outro servidor XMPP em execução na Internet (da mesma forma que os servidores de correio SMTP). Ao se conectar a essa federação de servidores, o SPADE permite que qualquer agente se comunique com qualquer outro agente, artefato ou humano no mundo, elevando o conceito de um sistema multiagente aberto a um novo nível. Em particular, o suporte que o SPADE requer de um servidor XMPP pode ser amplamente configurado de três maneiras diferentes, dependendo dos requisitos do aplicativo. Primeiro, um aplicativo de sistema multiagente executado no SPADE pode ser configurado para implantar seu próprio servidor XMPP público (há várias implementações de software de servidor XMPP de código aberto). Uma segunda configuração mais simples pode ser usando qualquer um dos servidores públicos XMPP existentes que estão disponíveis gratuitamente na Internet. Finalmente, uma terceira configuração possível é implantar um servidor XMPP privado, sem conexões de servidor para servidor, junto com o aplicativo SPADE [Gregori et al. 2006, Karagiannis et al. 2015].

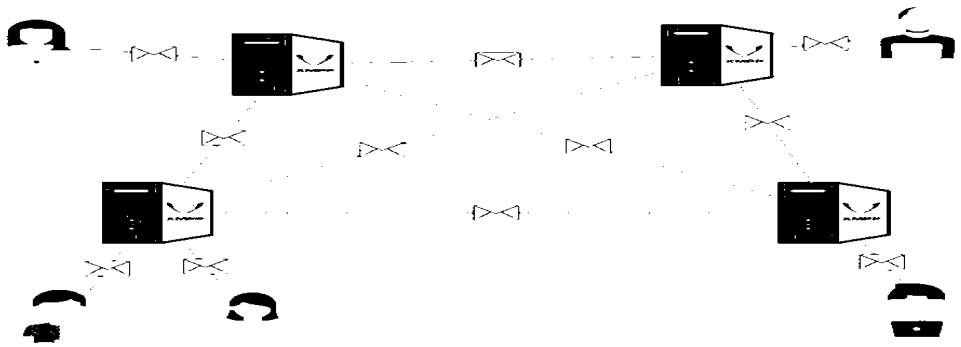

FIGURA 1. Uma representação de como os servidores XMPP federados se interconectam entre si e com os clientes, usando mecanismos de servidor para servidor e de cliente para servidor.

3. Especificação do Problema do Caixeiro Viajante

Suponha que um caixeiro viajante tenha de visitar n cidades diferentes, iniciando e encerrando sua viagem na primeira cidade. Suponha também, que não importa a ordem com que as cidades são visitadas e que de cada uma delas pode-se ir diretamente a qualquer outra. O problema de caixeiro viajante consiste em descobrir a rota que torna mínima a viagem total. No caso estudado, temos uma descrição do processo logístico de entregas utilizados pela empresa Loggi, caracterizando o contexto do problema, o que demanda uma breve explicação do processo. As entregas chegam no Centro de Distribuição, e são agrupadas em pacotes, os quais já tem rotas e entregadores vinculados, e são enviados aos centros intermediários, chamados de Centro de Expedição. Por último, os entregadores responsáveis pegam seus respectivos pacotes e fazem as entregas seguindo a rota pré-estabelecida [Galindo et al. 2020].

Table 1. Variables to be considered on the evaluation of interaction techniques

	Chessboard top view	Chessboard perspective view
Selection with side movements	6.02 ± 5.22	7.01±6.84
Selection with in-depth movements	6.29±4.99	12.22±11.33
Manipulation with side movements	4.66± 4.94	3.47±2.20
Manipulation with in-depth movements	5.71 ±4.55	5.37 ±3.28

4. Implementação

O problema de caixeiro viajante foi implementado utilizando linguagem Python para as duas ferramentas de comparação.

4.1. Spade

Ao inicializar, o sistema troca mensagens com entregadores, confirmando sua existência e conferindo seus dados, que seriam as restrições de cada carro como volume e peso, cria um Agente novo que representará esse entregador, vinculando um número de identificação. Após a definição dos pacotes, o agente notifica o Centro e este vincula os dados aos entregadores, salvando a informação final, conforme Figura 2.

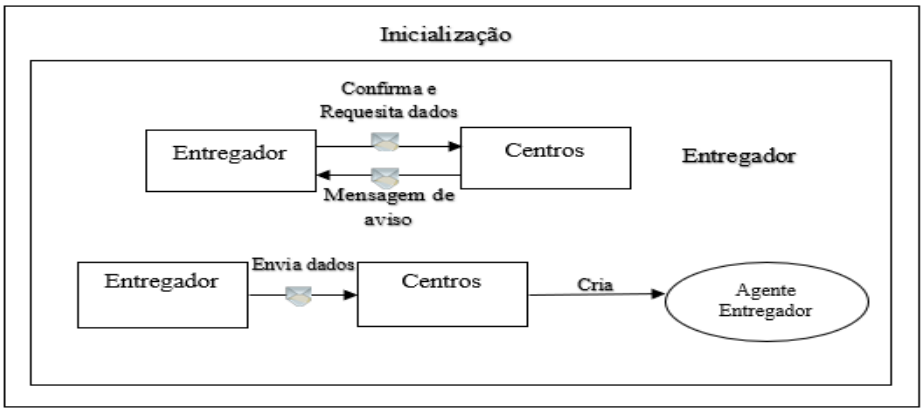

Figura 2. Sistema de troca de mensagem

4.2. Mesa

A simulação começa com duas classes principais: uma para o modelo geral e outra para os agentes. A classe model contém os atributos de nível de modelo. Cada caixeiro viajante terá um identificador exclusivo e serão ativados em ordem da chegada usando o escalonador que é um componente de modelo especial que controla a ordem na qual os agentes são ativados.

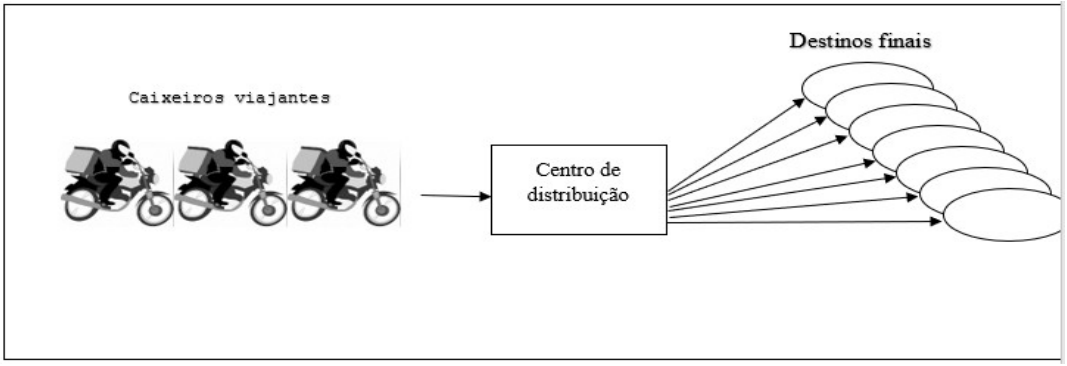

Figura 3: Ordem de ativação de agentes

Um conjunto de caixeiros disponíveis e um conjunto de pedidos que vem chegando em alguma ordem, pega o primeiro que aparecer e joga no primeiro veículo ativado. Coloca todo o pacote que chegar no primeiro caixeiro até que a capacidade dela seja atingida, depois vai pelo segundo caixeiro, assim sucessivamente, conforme Figura 3.

5. Comparação das Ferramentas

Com base nos requisitos das ferramentas e nas suas principais características apresentadas na literatura, foram definidos os critérios de avaliação apresentados na Tabela 1.

Tabela 1: Comparação das ferramentas

Ferramentas	Comunidade de usuário	Disponibilidade de ajuda ou suporte	flexibilidade	distribuição	Interface gráfica para visualização	linguagem
Spade	✓	✓	✓	✓	X	Python 3.9.10
Mesa	✓	✓	✓	✓	✓	Python 3.9.10

A Figura 4 apresenta a comparação de duas implementações diferentes em python de uma mesma entrada para saber qual apresenta melhor performance, com relação ao tempo de execução.

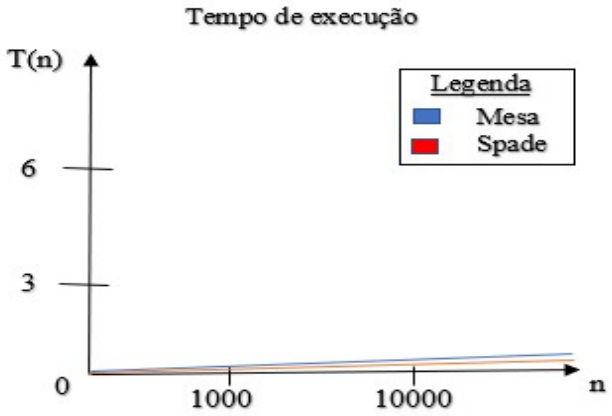

Figura 4: Comparação do tempo de execução

6. Conclusão

Neste artigo foi apresentada uma breve explicação a respeito da modelagem e simulação baseada em agente, além das ferramentas que suportam este tipo de modelagem e alguns critérios que ajudam a escolher a melhor ferramenta. Em seguida foi apresentada a comparação para o problema do caixeiro viajante, tanto nas suas caraterísticas funcionais como o tempo de execução das duas implementações. Desta forma, foram estabelecidos um conjunto de parâmetros comuns que permitisse avaliar as ferramentas de modelagem baseada em agentes e a auxiliar na tarefa de escolher a melhor opção de acordo com as necessidades específicas de cada usuário e situação.

Referências

Galindo, J. C. F., de Castro Surita, G., Neto, J. M., de Castro, C. L., and Lemos, A. P. (2020). A multi-agent system for solving the dynamic capacitated vehicle routing problem with stochastic customers using trajectory data mining. CoRR, abs/2009.12691

Gregori, M. E., Cámara, J. P., and Bada, G. A. (2006). A jabber-based multi-agent system platform. In Nakashima, H., Wellman, M. P., Weiss, G., and Stone, P., editors, 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006, pages 1282–1284. ACM.

Karagiannis, V., Chatzimisios, P., Vázquez-Gallego, F., and Alonso-Zarate, J. (2015). A survey on application layer protocols for the internet of things. Trans. IoT Cloud Comput., 3:11–17.

Klügl, F. and Bazzan, A. L. C. (2012). Agent-based modeling and simulation. AI

Magazine, 33(3):29.

Masad D. and Kazil, J.(2015). Mesa: An Agent-Based Modeling Framework. In Kathryn Huff and James Bergstra, editors, Proceedings of the 14th Python in Science Conference, pages 51 – 58.