Utilização de grau de confiança entre agentes para alocação de vagas em um Smart Parking

Lucas Fernando Souza de Castro, Gleifer Vaz Alves, André Pinz Borges

[1]Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR)
CEP 84016 – 210 – Ponta Grossa – PR – Brasil

l.castropg@gmail.com, {gleifer,apborges}@utfpr.edu.br

Abstract. *Looking for a parking spot in a big city could be a problem. There are computing solutions that are being developed to optimize this problem. One of these solutions is using multiagent systems (MAS). In this paper a MAS is developed in order to allocate spots in a smart parking using the framework JaCaMo. This paper comprises of two types of agents: manager and drivers. The manager is responsible to manage the parking spots which will be assigned for drivers according to a corresponding degree of trust. In order to verify the effectiveness of the MAS, several simulations were conducted in empirical scenarios. Experiments shows that the degree of trust impacts in the parking spot allocation process.*

Resumo. *Procurar por uma vaga de estacionamento em uma grande cidade pode ser um problema. Soluções computacionais estão sendo desenvolvidas para otimizar este problema. Uma dessas soluções é utilizando sistemas multi-agentes (SMA). Neste artigo um SMA é desenvolvido com o objetivo de alocar vagas em um estacionamento inteligente usando o framework JaCaMo. Este trabalho compreende dois tipos de agentes: gerente e motoristas. O gerente é responsável por gerenciar as vagas, as quais são atribuídas aos motoristas de acordo com um grau correspondente de confiança. Para verificar a eficácia do SMA proposto, foram conduzidas simulações em cenários empíricos que mostram que o grau de confiança impacta o processo de alocação de vagas.*

1. Introdução

Tecnologias a fim de facilitar a vida do homem moderno possuem uma grande demanda nos dias atuais, em função da necessidade do homem em realizar tarefas com cada vez menos recursos e tempo. Contudo, tais tecnologias muitas vezes não são capazes de atender tais demandas, sendo necessário otimizá-las, uma vez que estão inseridas em um mundo finito de recursos. Locais como cidades, pontos comerciais e lares deixaram de ser apenas um lugar de convívio de pessoas, mas sim um sistema social entre pessoas e objetos tecnológicos [Caragliu et al. 2011].

Cidades inteligentes têm como objetivo geral proporcionar elementos que facilitem a vida da população por meio do uso de tecnologia da informação para a minimização de custos e uso de recursos. Nestas cidades ainda há sub-elementos que podem ser otimizados, tais como: *economy*, *mobility*, *enviroment*, *people*, *living* e *governance*

[Caragliu et al. 2011]. Inseridos nas cidades inteligentes, os estacionamentos inteligentes ou *smart parking* são definidos como locais que utilizam tecnologias para automatizar e aprimorar as tarefas diárias de um estacionamento, como por exemplo, a alocação de vagas [Revathi and Dhulipala 2012]. Cidades como San Francisco nos Estados Unidos [SFPark 2015] e outras espalhadas pela América do Norte [Parkingedge 2013] desenvolveram sistemas computacionais para a automatização do processo de alocação e precificação das vagas de acordo com a sua utilização. Logo, é possível que motoristas verifiquem a disponibilidade e valores de vagas antes mesmo de saírem de suas casas.

No desenvolvimento de soluções computacionais para os *smart parkings* destacam-se o uso de sistemas multiagentes (SMA), devido a natureza do cenário dos estacionamentos. Nestes locais há um grande número de variáveis que devem ser controladas simultaneamente, no caso dos estacionamentos, os motoristas, vagas, pagamentos e outras variáveis. Além do processo de alocação de vagas, destaca-se o processo de negociação de vagas devido ao número limitado de vagas. Além de cidades americanas, em Napoles na Itália foi desenvolvido uma solução para *smart parking* que compreende o processo de negociação e precificação de vagas utilizando SMA [Di Napoli et al. 2014]. Destaca-se também que soluções envolvendo SMA possuem um fator social empregado, onde os agentes podem cooperar ou disputar um recurso, sendo no caso dos estacionamentos uma vaga [Di Nocera et al. 2014].

Sistemas multiagentes são sistemas compostos por agentes autônomos com um nível social. Os agentes possuem objetivos a serem atingidos e estão inseridos em um ambiente dinâmico [Wooldridge 2009]. A fim de tornar o desenvolvimento do SMA capaz de proporcionar características de um raciocínio humano, são utilizado modelos que abstraem tal raciocínio, como o modelo BDI (*Belief, Desire, Intention*). Além do modelo, é necessário utilizar ferramentas para desenvolver o SMA com estas características. Um exemplo de tal ferramenta é o *framework* JaCaMo .

O JaCaMo é um *framework* baseado no modelo BDI sendo composto por três módulos para o desenvolvimento de SMAs: uma linguagem de desenvolvimento para os agentes, um *framework* para o desenvolvimento dos artefatos presentes no ambiente que os agentes estão inseridos e uma ferramenta responsável pela normalização social do sistema [Boissier et al. 2013]. Para a implementação dos agentes, é utilizada a linguagem Jason, a qual foi desenvolvida baseada na linguagem AgentSpeak(L) para a programação de agentes BDI. O *framework* Cartago atua nos artefatos do ambiente que os agentes estão inseridos. Os artefatos presentes no ambiente de um modo geral fornecem funcionalidades aos agentes. E por fim, a ferramenta Moise atua na organização social dos agentes, delimitando o que os agentes podem e devem fazer e como realizam suas tarefas.

O presente trabalho destina-se a modelagem e desenvolvimento de um SMA capaz de alocar vagas em estacionamentos baseado na utilização de um agente centralizador (*manager*) para o gerenciamento destas vagas. A utilização das vagas é dada pelos agentes *drivers*, sendo que eles possuem um grau de confiança (*trust*) variável entre $(0-999)$. A confiança é proporcional a preferência de cada agente em possuir tal vaga devido ao número limitado de vagas no estacionamento. O atual trabalho está inserido em projeto do grupo de pesquisa GPAS (Grupo de Pesquisa em Agentes e Sotfware) denominado `MAPS` (***Multi**A**gent** P**arking** S**ystem***). O objetivo do `MAPS` é estender o trabalho desenvolvido em pesquisas anteriores [Gonçalves and Alves 2015] e também desta pesquisa.

Há soluções computacionais para a problemática dos *Smart parkings* [Zhao et al. 2014], inclusive com a utilização de SMA em diferentes plataformas, como por exemplo a linguagem Jade [Di Napoli et al. 2014]. O principal objetivo deste trabalho consiste na análise do grau de confiança entre os agentes presentes no estacionamento e como este grau impacta no processo de alocação de vagas do estacionamento, visto que há agentes *drivers* com graus completamente distintos de confiança, mas que compartilham o mesmo objetivo, i.e., uma vaga no *Smart Parking*.

O restante do artigo está organizado como segue: Na seção 2 é apresentado a modelagem e implementação do sistema multiagente para o `MAPS`. Na seção 3 são apresentados os resultados que o SMA apresentou em diferentes de cenários de utilização, e por fim, na seção 4 a conclusão.

2. Modelo do SMA para o projeto MAPS

O objetivo do SMA desenvolvido é alocar vagas para agentes *drivers*. O agente responsável pelo processo de alocação de vagas é o agente *manager*. Inserido no processo de alocação de vagas, há sub-processos que ocorrem para que a vaga seja devidamente alocada. A figura 1 ilustra uma diagrama de casos de uso para demonstrar as ações que os agentes podem realizar.

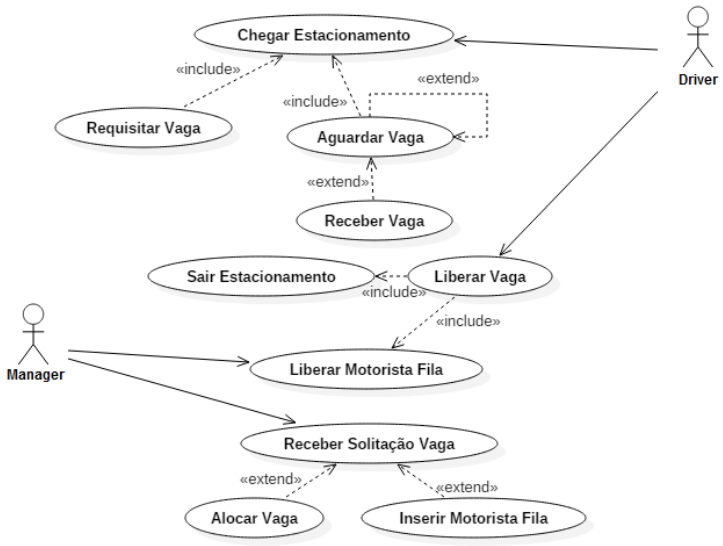

Figura 1. Digrama de casos de uso - SMA

2.1. Principais funcionalidades

O primeiro passo do desenvolvimento do SMA foi elencar os requisitos do sistema providos aos agentes para o tornar o sistema robusto e flexível a futuras extensões do projeto `MAPS`, dentre eles:

Ator: Agente *Manager*

1. **Receber solicitação de requisição de vaga:** O *manager* pode a qualquer momento receber requisições providas dos agentes *driver* para vagas no estacionamento. Nem sempre uma requisição de vaga é atendida, pois o estacionamento pode estar lotado;

2. **Decidir para qual agente alocar uma vaga:** Caso o estacionamento esteja lotado, ou próximo da sua lotação máxima, o *manager* se baseia no grau de confiança (*degree of trust*) dos agentes *drivers* e no tempo que ele está aguardando a vaga, caso for acima de 60 segundos. O tempo de 60 segundos foi estipulado devido a aplicabilidade do SMA desenvolvido a estacionamentos de médio porte, podendo este valor ser ajustável de acordo com a utilização do estacionamento. Para o cálculo do grau de confiança dos agentes, a seguinte fórmuma é utilizada:

$$degreeTrust(X) = degreeTrust(X) + \alpha$$

$\alpha = 1$ caso o *driver* utilizou a vaga selecionada pelo *manager*

$\alpha = 0$ caso contrário

3. **Possuir conhecimento e controle sobre o estacionamento:** O agente *manager* possui todo controle do estacionamento, pois é por meio dele que um agente requisita, ganha, utiliza e libera uma vaga. Além disso, o *manager* tem conhecimento de todas as vagas, o seu estado (livre, ocupada ou reservada), sua localização e quem está ocupando-a;
4. **Receber o aviso de um motorista que está saindo do estacionamento:** Ao sair do estacionamento, o *driver* deve avisar ao *manager* que está deixando a vaga livre. Após isso ocorrer, o *manager* decide qual agente *driver* irá receber a vaga, caso exista uma fila de espera.
5. **Controlar a fila do estacionamento:** Caso o estacionamento esteja cheio, os novos *drivers* que requisitarem uma vaga serão inseridos em uma fila. A fila é organizada de acordo com o grau de confiança que os agentes *drivers* possuem. Porém, pode haver um *driver* esperando a mais de 60 segundos nessa fila, sendo assim será dado prioridade a este agente.

Ator: Agente *Driver*

1. **Requisitar uma vaga ao agente *manager*:** O *driver* ao chegar no estacionamento requisita uma vaga ao agente *manager*, podendo ser respondido com uma vaga ou tendo que aguardar por uma;
2. **Receber uma vaga e estacionar:** Ao receber uma vaga, o agente deverá dirigir-se a esta vaga e estacionar o seu veículo;
3. **Aguardar uma vaga:** Caso o estacionamento esteja lotado ao requisitar uma vaga, o agente *driver* deverá aguardar até que o agente *manager* o notifique com uma liberação de vaga;
4. **Possuir características e crenças de um usuário de estacionamento:** A fim de tornar a abstração do agente *driver* próxima a um motorista real, os agentes *driver* presentes no sistema possuem as seguintes crenças: tempo para chegada no estacionamento, tempo aproximado de estadia no estacionamento, vaga que está estacionado e grau de confiança.
5. **Possuir um grau de confiança perante o agente *manager*:** Assim como descrito no item anterior, um agente *driver* é capaz de possuir um grau de confiança perante o *manager*. Este grau compreende as atitudes desse motorista no estacionamento, como por exemplo: não infringir as regras e utilizar o estacionamento de forma regular. O valor poderá diminuir caso o motorista não cumpra as regras estabelecidas.

Na figura 2 é ilustrado de uma maneira geral o funcionamento do *Smart parking* e como o grau de confiança impacta no processo de alocação de vagas.

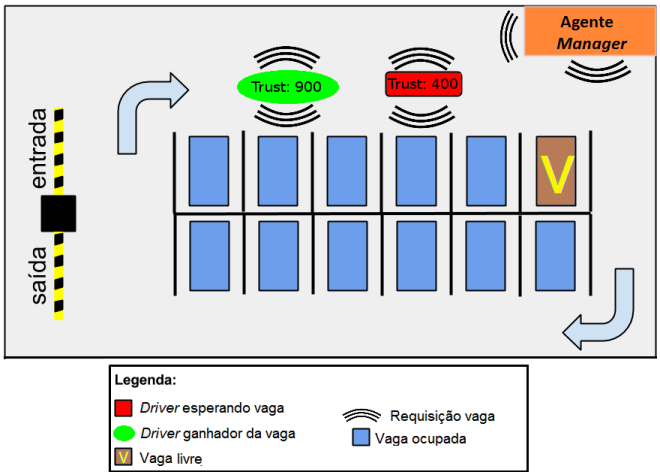

Figura 2. Estrutura básica do estacionamento - Adaptado de [Gonçalves and Alves 2015]

2.2. Desenvolvimento do SMA utilizando JaCaMo

A etapa de implementação do SMA é subdividida em três etapas. A etapa inicial da implementação é o desenvolvimento dos agentes e suas interações. A segunda etapa é a implementação dos artefatos do ambiente e suas interações com os agentes e por fim a etapa de otimização da utilização desses artefatos. Destaca-se que essa implementação do `MAPS` apresenta apenas o uso do Jason e Cartago. A programação normativa do sistema através do Moise será desenvolvida em uma etapa subsequente do projeto. A seguir é apresentado na figura 3 um diagrama geral elaborado por meio da metodologia Prometheus da arquitetura do SMA.

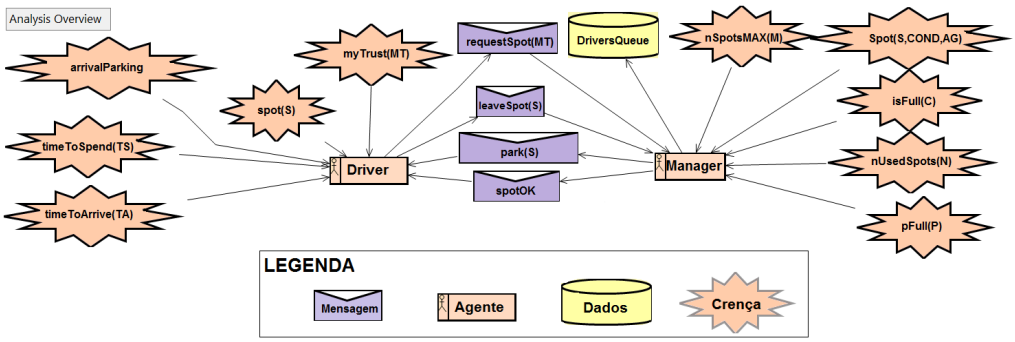

Figura 3. Visão geral do SMA

Etapa 1: Implementação dos agentes

A implementação dos agentes utilizando o *framework* JaCaMo é feita na linguagem Jason. O modelo BDI denota que as crenças são as informações a respeito do agente e do ambiente em que ele está inserido. Já os desejos são denotados como os objetivos que

o agente possui, ou o que deseja alcançar, e por fim, as intenções são como o agente irá atingir esses objetivos, o que no Jason são denotados como planos.

O JaCaMo possui uma linguagem chamada JCM a qual é possível determinar as crenças iniciais do agente, seus objetivos, artefatos do ambiente e outros parâmetros relacionados ao SMA. No código 1 é apresentado o arquivo JCM do sistema multiagente, exibindo dois agentes *drivers* (`m1` e `m2`) com suas crenças iniciais e a declaração do agente *manager*. Além das crenças iniciais, há crenças adquiridas ao decorrer da execução. Um exemplo de uma crença adquirida durante a execução é a vaga que um *driver* recebe.

```
1  mas mAPS{
2  agent manager
3          focus: maS3.Control
4          focus: maS3.Gate
5  agent m1: driver.asl {
6          beliefs: myTrust(100)
7                   timeToSpend(10000)
8                   timeToArrive(17514)}
9
10 agent m2: driver.asl {
11         beliefs: myTrust(450)
12                  timeToSpend(3000)
13                  timeToArrive(25307)}
```

Código 1. Arquivo JCM

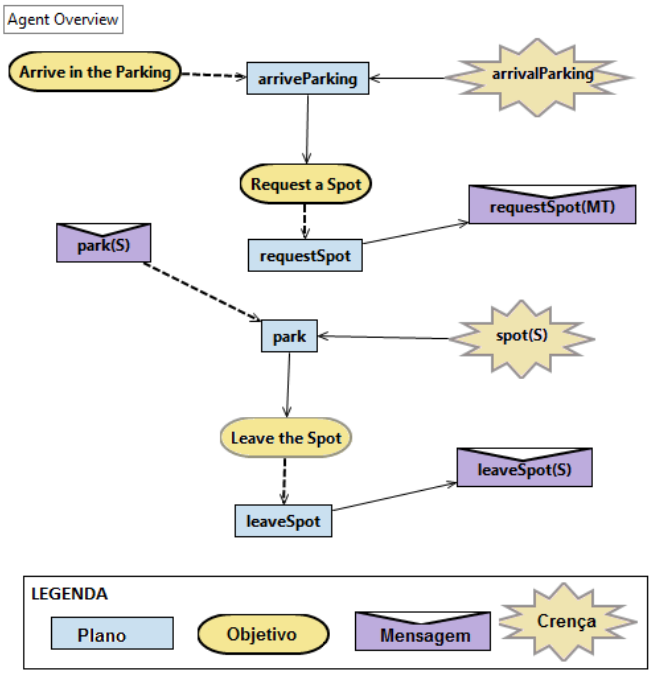

Figura 4. Perspectiva geral do agente *Driver*

As crenças iniciais do agente *driver* exibidas no código 1 e as crenças adquiridas também ilustradas na figura 4 são descritas nos itens abaixo.

- **myTrust(MT)**: Crença inicial do grau de confiança que o agente possui, sendo `MT` o valor correspondente da crença a respeito do grau de confiança;

- **timeToSpend(TS)**: Crença inicial do tempo que o agente permanecerá dentro do estacionamento, onde `TS` é o valor correspondente desse tempo em segundos;
- **timeToArrive(TA)**: Crença inicial a respeito do tempo que o agente levará para chegar ao estacionamento e requisitar uma vaga, sendo `TA` o valor representativo da crença em segundos;
- **spot(S)**: Crença adquirida durante a execução do SMA após a requisição aceita da vaga. O agente *manager* envia ao agente *driver* o identificador da vaga para que o agente utilize a vaga, sendo `S` o identificador dessa vaga;
- **arrivalParking**: Crença adquirida após a chegada do agente *driver* no estacionamento.

Além das crenças, os agentes implementados possuem objetivos e planos. Na figura 4 é apresentado um diagrama de perspectiva geral do agente *driver* desenvolvido por meio da metodologia Prometheus. Esse diagrama ilustra as crenças adquiridas, objetivos, planos e as mensagens que o agente troca durante a execução do SMA.

As crenças do agente *manager* não encontram-se no arquivo JCM, elas foram inseridas diretamente no arquivo MANAGER.ASL (vide código 2). A figura 5 apresenta o diagrama perspectiva geral do agente *manager*, ilustrando as suas crenças, objetivos, planos e trocas de mensagens com os agentes *drivers*.

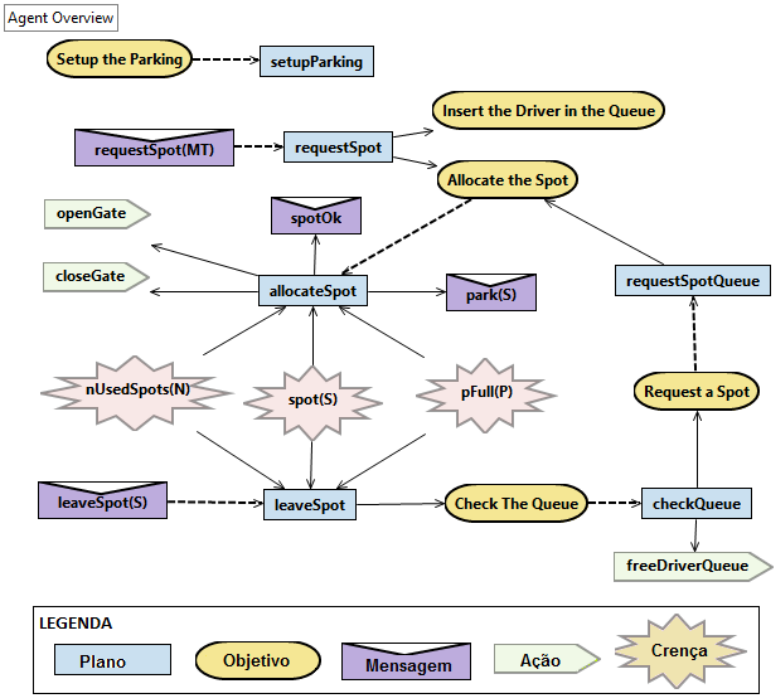

Figura 5. Perspectiva geral do agente *Manager*

No código 2 é apresentado uma parte do código MANAGER.ASL correspondente as crenças iniciais do agente *manager*. As crenças iniciais do agente *manager* exibidas na figura 5 e no código 2 são descritas a seguir.

```
1 nSpotsMAX(4).
2 nUsedSpots(0).
3 isFull(false).
4 pFull(0).
5 spot(0,0, "EMPTY").
6 spot(1,0, "EMPTY").
7 spot(2,0, "EMPTY").
8 spot(3,0, "EMPTY").
```

Código 2. Crenças e objetivos iniciais - Agente *Manager*

- **nSpotsUsed(N)**: Número de vagas utilizadas no momento, sendo `N` o valor correspondente inteiro correspondente a essas vagas;
- **nSpotsMAX(M)**: Número máximo de vagas que o estacionamento comporta, onde `M` é o valor inteiro máximo destas vagas;
- **isFull(C)**: Condição booleana para verificar se o estacionamento está cheio. Caso sim, os *drivers* que requisitarem uma vaga serão destinados para a fila, onde `C` expressa tal condição;
- **pUsage(P)**: Percentual de uso do estacionamento, sendo `P` um valor inteiro que representa essa porcentagem;
- **spot(S,COND,AG)**: Diferente do spot(S) do agente *driver* , essa crença determina as características da vaga em uma tupla de valores, onde o S é responsável pelo identificador da vaga; COND para verificar se há um agente ocupando a vaga e AG para identificar qual *driver* está ocupando a vaga;

Etapa 2: Implementação dos artefatos

Os artefatos em Cartago são os responsáveis por prover ações e funcionalidades para os agentes inseridos no ambiente. No contexto desse trabalho, o ambiente em si é o estacionamento. Para a versão do sistema do atual trabalho foram implementados dois artefatos: *Control* e *Gate*.

- **Artefato *Gate*:** Situando na entrada do estacionamento, responsável pela abertura e fechamento da cancela;
- **Artefato *Control*:** Responsável pelo gerenciamento dos *drivers* na fila, sendo inserindo-os ou selecionando o *driver* com as melhores condições de receber uma vaga.

A fila que o artefato *Control* gerencia é ordenada de acordo com o grau de confiança que cada *driver* possui, sendo assim, quanto maior o grau de confiança desse agente, mais rápido será a sua alocação para a vaga. Contudo, caso exista algum *driver* com um grau de confiança baixo, mas que já esteja na fila de espera por um tempo considerável. Esse *driver* não ficará aguardando na fila por mais tempo que um certo tempo limite. Na atual versão do SMA o limite é igual a 60 segundos, visto que o SMA aqui modelado é de um estacionamento privado e de médio porte. Portanto, caso existam motoristas na fila aguardando por uma vaga e uma vaga é liberada, o agente *manager* irá primeiro verificar se há algum *driver* com um tempo de espera na fila igual ou superior a 60 segundos. Se não houver, é analisado o grau de confiança e o agente com o maior valor recebe a vaga.

3. Resultados

Com o objetivo de demonstrar o impacto que o grau de confiança gera no processo de alocação de vagas, a seguir são descritos três cenários em que o sistema foi submetido com diferentes perfis de agentes *drivers*. Para cada perfil foi estipulado valores aleatórios para as crenças de cada agente. Na tabela 1 são apresentado os cenários C1, C2 e C3, bem como a quantidade de *drivers* presentes e a quantidade de vagas para cada cenário. A tabela 2 apresenta a configuração dos *drivers* com os seus respectivos valores para as crenças `timeToArrive`, `timeToSpend` e `myTrust`. Inicialmente o `MAPS` foi submetido a dez cenários para teste, porém neste trabalho serão descritos apenas os cenários mais críticos, os quais tem uma grande quantidade agentes *drivers* disputando por poucas vagas.

Tabela 1. Configuração dos cenários

Cenário	Número de *Drivers*	Número de vagas	Razão *Driver*/Vaga
1	50	1	50
2	50	2	25
3	10	1	10

Tabela 2. Configuração dos *drivers*

Agente	*timeToArrive(TA)*	*timeToSpend(TS)*	*myTrust(MT)*
m1	4	10	100
m2	2	3	450
m3	7	5	999
m4	3,5	7,5	4
m5	4,5	10	120
m6	6,7	9	770
m7	9	12	180
m8	10,29	8,5	10
m9	9,9	6,6	803
m10	4,6	6,6	5

Obs: Valores das crenças *timeToSpend* e *timeToArrive* são denotadas em segundos.

Os cenários propostos na tabela 1 apresentam de 10 a 50 agentes. Não foram utilizados mais agentes devido aos estacionamentos compreendidos pelo atual SMA serem de médio porte. A tabela 2 apresenta 10 perfis de agentes *drivers* . Para os cenários com 50 agentes foram criadas 5 instâncias de cada perfil para totalizar os 50 agentes propostos nos cenários C1 e C2.

Cenário 1

Neste cenário, tem-se 50 *drivers* disputando uma única vaga. A seguir na figura 6 é apresentado um gráfico do grau de confiança *versus* o tempo de alocação de vaga em segundos.

De acordo com o gráfico da figura 6, o grau de confiança impactou positivamente no tempo de alocação da vaga. Durante a execução do Cenário 1, existem vários *drivers* na fila de espera. Contudo, devido ao limite de tempo máximo de espera na fila, em certo

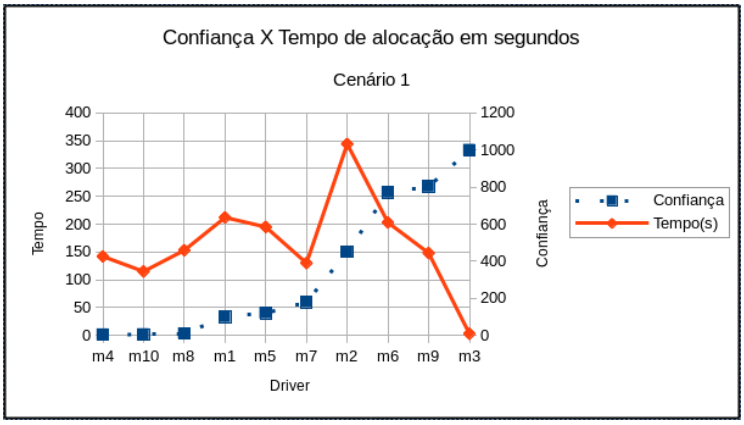

Figura 6. Cenário 1 - Relação grau de confiança *versus* tempo de alocação

momento será dada prioridade aqueles agentes que estão na fila por mais de 60 segundos. Ainda destaca-se que o *driver* `m3` (que possui um grau máximo de confiança - 999), tem o menor tempo de espera na fila de motoristas.

Cenário 2

Nesse cenário, há 50 agentes *drivers* para duas vagas. O gráfico da figura 7 apresenta a relação grau de confiança *versus* o tempo de alocação da vaga em segundos.

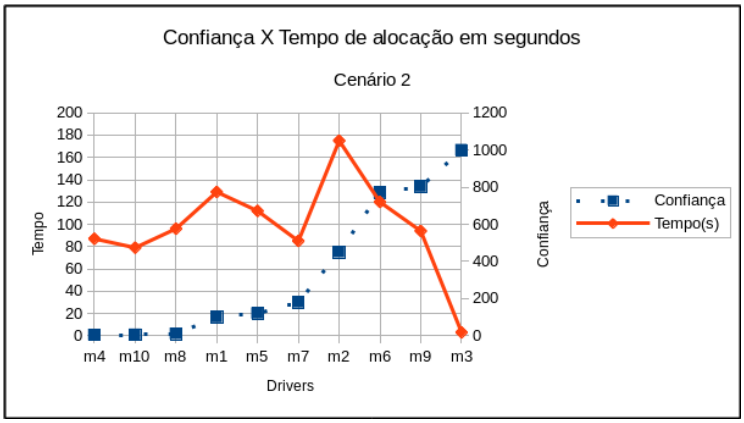

Figura 7. Cenário 2 - Relação grau confiança *versus* tempo de alocação

Neste cenário em contraste com cenário 1 possui uma vaga de estacionamento a mais disponível, diminuindo assim a média geral de tempo de alocação em segundos, sendo a média de tempo de alocação geral do Cenário 1 de 153 segundos e no Cenário 2 de 88 segundos. De modo simular ao cenário anterior, o grau de confiança impactou positivamente no tempo de alocação das vagas.

Cenário 3

No cenário 3 tem-se 10 agentes disputando uma única vaga. O gráfico apresentado na figura 8 demonstra o grau de confiança e seu impacto no processo de alocação de vagas.

Mesmo com um número reduzido de agentes utilizando o estacionamento, o grau de confiança ainda impactou no processo de alocação das vagas, mas um impacto menor

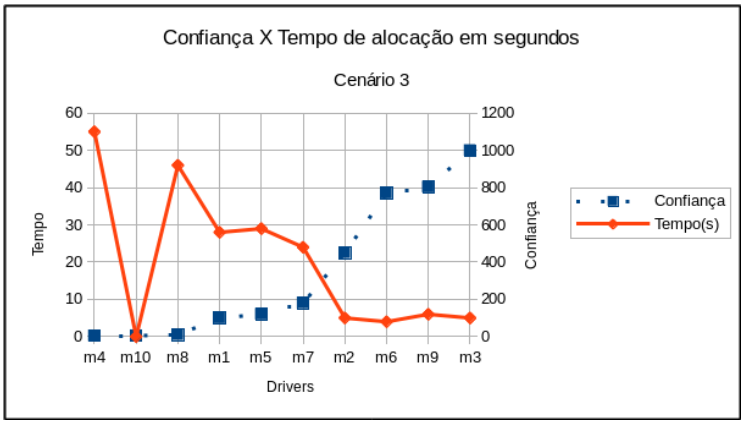

Figura 8. Cenário 3 - Relação grau de confiança *versus* tempo de alocação

em relação aos cenários anteriores. Porém, devido ao número reduzido de *drivers*, os tempos de espera não foram elevados ($>$ 60 segundos).

O principal objetivo da descrição dos três cenários é a análise do impacto que o grau de confiança gera na alocação de vagas dos agentes *Drivers*. Vale notar que durante a alocação de vagas também é analisado o tempo de espera do *Driver*. A fim de sumarizar os cenários apresentados, com base na tabela 2 foram selecionados agentes para cada nível de valor de *trust*: dois agentes (`m9` - 803) e (`m3` - 999) para o valor alto (700 $\rightarrow$ 999), um agente (`m2` - 450) para o valor médio (350 $\rightarrow$ 699) e dois agentes (`m4` - 4) e (`m7` - 180) para o valor baixo (0 $\rightarrow$ 349). A tabela 3 ilustra os agentes selecionados e seu tempo de alocação para receber uma vaga nos cenários C1,C2 e C3. Além disso é apresentado a média aritmética de alocação do agente nos cenários baseada no tempo de alocação para os cenários.

Tabela 3. Impacto do grau de confiança no processo de alocação de vagas

Agente	*myTrust (MT)*	*timeToArrive (TA)*	*timeToSpend (TS)*	C1	C2	C3	Média
m4	4	3,5	7,5	142	87	55	94.7
m7	180	9	12	130	85	24	79.7
m2	450	2	3s	344	175	5	175
m9	803	9,9	6,6	148	94	6	82
m3	999	7	5	3	3	5	3.7

Obs: Valores para as crenças: *timeToSpend*, *timeToArrive*, tempos de alocação de vaga para os cenários C1,C2,C3 e para a média são denotados em segundos.

De acordo com a tabela 3 é possível notar o impacto que o grau de confiança causa no processo de alocação de vagas. Entretanto há casos que esse valor entra em segundo plano devido ao tempo máximo de espera na fila de 60 segundos. O *driver* `m2` possui um grau médio de confiança e ainda assim possui a maior média, sendo assim nem sempre um *driver* com um valor médio terá um tempo de alocação médio. Por outro lado, na maioria dos casos dos *drivers* com um grau alto de confiança (*Driver* `m3` - *Trust*: 999) possuem em média um valor baixo para o tempo de alocação de uma vaga, em contrapartida os *drivers* com um grau baixo de confiança (*Driver* `m4` - *Trust*: 4) possuem em média um valor alto de tempo para alocação de uma vaga.

4. Conclusão

O principal objetivo do trabalho aqui apresentado é a modelagem e desenvolvimento de um SMA para alocação de vagas com base em graus de confiança. O desenvolvimento do SMA no *framework* JaCaMo proporcionou uma implementação robusta e flexível para futuras expansões do projeto. Além dos graus de confiança, há o fator do tempo limite de espera que influencia o processo de alocação de vagas, tornando-se prioridade para alguns *drivers* em relação à confiança em alguns cenários. O principal objetivo de inserir o tempo limite para alocação é priorizar também os *drivers* que estão a um longo tempo aguardando por uma vaga, todavia sem tirar o mérito dos agentes que possuem um grau de confiança alto. De fato, a inserção do tempo limite também busca a adequação do SMA a uma simulação de um caso mais próximo da realidade, visto que dificilmente um motorista iria esperar muito tempo na fila de um estacionamento privado.

Ainda é possível destacar possíveis extensões do projeto `MAPS`, como: (i) um artefato de persistência dos graus de confiança, de maneira que o grau de confiança possa ser incrementado ou decrementado conforme as utilizações dos *drivers*; (ii) artefato com uma interface gráfica do estado atual do sistema multiagente; (iii) organização social do SMA via Moise; (iv) futuras integrações com as plataformas Android e Arduino a fim de permitir que o `MAPS` seja aplicável em cenários reais.

Referências

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., and Santi, A. (2013). Multi-agent oriented programming with jacamo. *Sci. Comput. Program.*, 78(6):747–761.

Caragliu, A., Bo, C. D., and Nijkamp, P. (2011). Smart cities in europe. *Journal of Urban Technology*, 18(2):65–82.

Di Napoli, C., Di Nocera, D., and Rossi, S. (2014). Negotiating parking spaces in smart cities. In *Proceeding of the 8th International Workshop on Agents in Traffic and Transportation, in conjunction with AAMAS*.

Di Nocera, D., Di Napoli, C., and Rossi, S. (2014). A Social-Aware Smart Parking Application. In *Proceedings of the 15th Workshop "From Objects to Agents* , volume 1269.

Gonçalves, W. R. C. and Alves, G. V. (2015). Smart parking: mecanismo de leilão de vagas de estacionamento usando reputação entre agentes. In *Anais do IX Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações – IX WESAAC*.

Parkingedge (2013). Best parking. Disponível em <http://www.bestparking.com>.

Revathi, G. and Dhulipala, V. (2012). Smart parking systems and sensors: A survey. In *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, pages 1–5.

SFPark (2015). Sfpark. Disponível em <http://www.sfpark.org>.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition.

Zhao, X., Zhao, K., and Hai, F. (2014). An algorithm of parking planning for smart parking system. pages 4965–4969. IEEE.