Algoritmos de coordenação para veículos autônomos em cidades inteligentes

Vilson de Deus Corrêa Júnior[1], **Tiago Luiz Schmitz**[1]

[1]Departamento de Engenharia de Software – Universidade do Estado de Santa Catarina (UDESC)
89140-000 – Ibirama – SC – Brazil

vilson.junior@edu.udesc.br, tiago.schmitz@udesc.br

Abstract. *The current paper presents multi-agents systems coordination algorithms. The introduced solution is a variation of the system developed and used by Akuanduba-UDESC, on the annual Multi-Agent Programming Contest - 2018. Throughout the paper it is going to be shown the utilized strategies, features and the preliminary results.*

Resumo. *O presente trabalho apresenta algoritmos para coordenação de sistemas multiagentes. A solução aqui apresentada é uma variação da solução desenvolvida e utilizada pela equipe Akuaduba-UDESC, na competição anual Multi-agent Programming Contest - 2018. Com isso, será apresentado ao longo do trabalho as estratégias adotadas, ferramentas e resultados preliminares.*

1. Introdução

Com a globalização o conceito de cidades inteligentes é utilizado em diversos países. Esse conceito se apresenta de modo disruptivo, com objetivo de solucionar problemas do dia-dia com o uso de sistemas computacionais. Tais sistemas podem analisar o ambiente por meio de sensores e com isso realizar deliberações sob o mesmo, com intuito de melhorar ou solucionar problemas cotidianos [FAPESP 2018].

Além disso, veículos autônomos podem ser empregados nesses ambientes, como carros, caminhões, drones e motos. Ao adotar tal prática, de modo parcial ou total pode-se retirar a responsabilidade humana da atividade, com isso reduzir o número de acidentes de trânsito [de Sousa Pissardini et al. 2013]. Dessa forma, ao utilizar veículos autônomos, como agentes atuadores em uma cidade inteligente seria um exemplo de sistema multiagentes (SMA).

Para programar esse tipo de sistema em que o objetivo é atuar em um ambiente colaborativo ou competitivo é necessário desenvolver agentes com noções mentais de crenças, desejos, intenções e objetivos [Wooldridge 2001]. Entretanto a coordenação de um SMA tende a ser uma tarefa complexa, para isso é necessário desenvolver algoritmos de coordenação. Ao desenvolver tal sistema é possível reduzir problemas de mobilidade urbana, sustentabilidade e habitabilidade [Taniguchi 2012].

No presente trabalho, os agentes terão responsabilidade de atuar em uma cadeia logística, realizando tarefas com uma abordagem distribuída na busca, montagem, entrega de itens, sem desprezar o gerenciamento da bateria. Para isso foi utilizado o cenário de 2018 da *Multi-Agent Programming Contest* [1] (seção 2). O objetivo principal

[1]*Multi-Agent Programming Contest - https://multiagentcontest.org/*

da competição é que equipes desenvolvam algoritmos de coordenação para SMA, com o intuito atuar em uma cidade inteligente.

A solução apresentada nesse artigo é uma variação da solução apresentada pela equipe Akuanduba-UDESC na competição, ao qual a equipe adotou uma estratégia centralizada no gerenciamento do estoque. Suspeita-se que essa abordagem é ineficiente, tendo em vista que os itens ficarão em somente um estoque, reduzindo assim a eficiência no atendimento de demandas de trabalho. Esse artigo propõe uma abordagem distribuída dos estoques para melhorar o acesso aos itens e consequentemente melhorar a eficiência no atendimento dos trabalhos. As seções 3 e 4 apresentam a estrutura geral e as estratégias adotadas.

2. Cenário da *Multi-Agent Programming Contest*

O ambiente utilizado para simular cidades inteligentes será o da competição anual *Multi-Agent Programming Contest* (MAPC) do ano de 2018. O cenário se passa no ano de 2045 dC após a colonização de Marte. O mapa é retirado do *OpenStreetMap* contém instalações como: estações de recarga, lixões, lojas, depósitos, *workshops*, nós de recurso e poços de água. Todas as instalações, detalhes, rotas e tempo são abstraídos e fornecidos pelo servidor da competição [Ahlbrecht 2018]. A unidade de medida de tempo do ambiente é medida em *steps*.

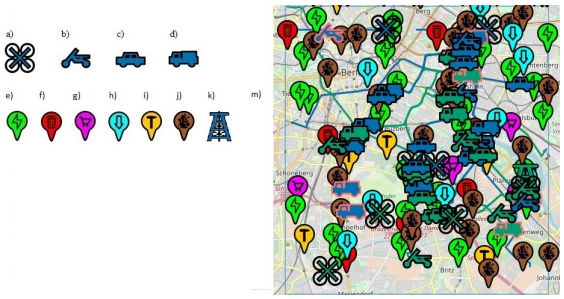

Figura 1. Cenário da competição

Nas estações de recarga é possível o agente recarregar sua bateria. Nos lixões, o agente poderá descartar itens que não seja mais do seu interesse. Nas lojas é possível o agente realizar melhorias em suas características e/ou adquirir itens. Nos depósitos, os agentes poderão guardar os itens obtidos nas lojas e/ou nós de recursos. Enquanto no *workshop* é possível confeccionar itens secundários compostos de itens primários. Em nós de recursos é possível obter itens do tipo primário. Cada equipe pode construir poços de água com objetivo de ganhar pontos na partida.

Os agentes que atuam sob o ambiente necessitam ser coordenados com objetivo de cumprir tarefas como a construção de poços, recarga, confecção de itens e a realização de demandas de trabalho. Eles podem ser do tipo carro, caminhão, drone e moto. Cada um deles contém características particulares como bateria, capacidade de carga, visão, velocidade e habilidade. Além disso, a partida conta com recursos randômicos no tipo de itens, trabalhos, volumes e recompensas [Ahlbrecht 2018].

3. Arquitetura Geral

Para programação do SMA foi utilizado o *framework* JaCaMo, sendo ele composto pelas ferramentas *Jason, CArtAgO* e *Moise* [Boissier et al. 2013]. Dos quais foram utilizado apenas o *Jason* e o *CArtAgO*. O *Jason* implementa o modelo BDI - *Belief, Desires and Intention*, dessa forma é possível desenvolver os agentes com crenças, desejos e intenções [Rao and Georgeff 1991]. Para gerenciar as diferentes tarefas dos agentes foi desenvolvido uma fila de prioridades por meio de crenças, desejos e planos. O desejo *addtask* responsável por adicionar uma nova tarefa ao agente. Os parâmetros do addtask são: o rótulo da tarefa a ser cumprida, a prioridade, a lista de passos para atingir o objetivo e uma lista com o histórico dos passos executados.

O rótulo das tarefas que podem ser desempenhadas pelos agentes ao longo da partida, bem como a prioridade são descritos na tabela 1. Sendo a tarefa de recarga a mais prioritária de todas, com objetivo de garantir a autonomia do agentes em toda partida. Quando existir uma tarefa mais prioritária que a tarefa em execução, haverá uma preempção de tarefas, caso isso ocorra o agente, a tarefa será retomada que não houverem tarefas de maior prioridade.

Tabela 1. Tarefas e Prioridades

Tarefa	Prioridade	Tarefa	Prioridade
Recarga	10	Ajudar outro agente	8,2
Missões	9	Montagem de item	8
Exploração	9	Obter item	8
Devolver itens	8,9	Realizar Trabalho	5
Atualizar Capacidade	8,5	Obter itens primários	4

O CArtAgO foi utilizado para implementar o ambiente, encapsular serviços, funcionalidades e prever situações em tempo de execução [Boissier et al. 2013]. Para estabelecer e manter a conexão ao servidor da competição *MASSim*, foi utilizado o *EISMASSim* do padrão *Environment Interface Standard - EIS*. Por meio dessa solução é possível mapear as chamadas de métodos Java e assim realizar troca de mensagens em XML [Ahlbrecht 2018].

A estrutura geral é representada pela figura 2. O artefato *EISAccess*, filtra todas as propriedades e percepções recebidas do servidor da competição (*EISMASSim*). A coordenação é feita pelo artefato *CoordinationArtifact*, ele delega responsabilidades aos agentes ao longo da partida, como gerenciamento de estoques e assegura que determinada tarefa não tenha mais de um agente comprometido em sua realização. Além disso, os agentes poderão se comunicar diretamente com outros agentes, por meio do sistema de mensagens do *Jason*. Sendo assim, exercer a comunicação e o gerenciamento é fundamental na descentralização de tarefas [Christie et al. 2003].

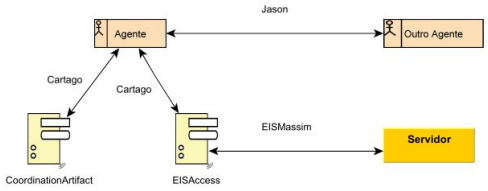

Figura 2. Estrutura Geral

4. Estratégias Adotadas

As estratégias apresentadas nesse artigo são uma variação da solução desenvolvida pela equipe Akuanduba-UDESC, participante da competição MAPC de 2018. No inicio da simulação os agentes exploram o ambiente com o objetivo de descobrir instalações ao longo do mapa. A exploração é realizada com uso de *drones* e o mapa é dividido em quatro partes, para isso, eles informam sua latitude e longitude para o artefato de coordenação.

Que por sua vez, soluciona um problema de programação linear binária, com objetivo de minimizar a distância a ser percorrida pelos agentes [Longaray and Beuren 2001]. Ao longo da exploração, conforme os agentes descobrem as instalações, os agentes do tipo carro, moto e caminhão são alocados por meio de troca de mensagens para buscar o máximo de itens nos nós de recursos e leva-los até os estoques. Na solução apresentada pela equipe Akuanduba-UDESC, a estratégia era centralizar todos os itens em um estoque central, nesse artigo propomos a distribuição deles.

A distribuição dos estoques é feita com base em um fecho convexo cujos vértices são as coordenadas dos estoques, que interligados formam uma estrutura que incorpora todas as outras localizações. Para que não haja recálculo, somente um agente informa ao artefato as posições dos estoques e por sua vez ele retorna a melhor distribuição deles. Com isso o agente informa aos outros agentes os estoques selecionados (figura 3).

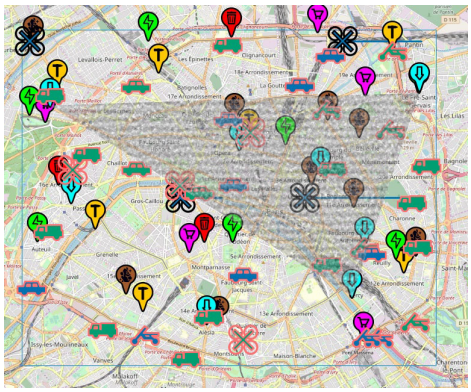

Figura 3. Fecho convexo de estoques

O algoritmo para seleção de estoques (algoritmo 1), pode ser divido em três partes, o cálculo do fecho convexo, cálculo do ponto médio e seleção dos estoques por abrangência.

Algoritmo 1: Seleção de estoques

Entrada: Estoques do Mapa
Saída: Estoques Selecionados
início
    PONTOS = Cálculo do Fecho Convexo(estoques);
    M = Cálculo do Ponto Médio(pontos);
    ES = Seleção de Estoques Por Abrangência(pontos, m);
fim

4.1. Cálculo do Fecho Convexo

Com objetivo de calcular o fecho convexo, uma solução inicial é selecionar o estoques mais ao norte, sul, leste e oeste. Sendo assim, o ponto mais ao norte, é aquele

estoque ao qual possui a maior latitude. A mesma seleção ocorre para os pontos mais ao leste, mais ao sul e mais ao oeste respectivamente. Após obter aos quatro pontos mais extremos, eles são interligados por segmentos de retas de modo a obter o fecho.

O cálculo do ponto médio são considerados os pontos norte, sul, leste e oeste selecionados no passo anterior. Desses pontos é calculado a média das latitudes e longitudes para obter um ponto central do fecho.

Para cada estoque que não faz parte do fecho deverá ser feito uma comparação com cada uma das retas do fecho atual, para verificar se o estoque está no lado interno ou externo do fecho. Para isso são calculadas duas determinantes: *d1* - posição do estoque em análise em relação aresta e *d2* - ponto médio em relação a aresta.

Caso para uma mesma aresta do fecho, os determinantes *d1* e *d2* tenham sinais opostos, significa que o estoque analisado está fora do fecho convexo. Com isso, ocorre um relaxamento da solução e o ponto passa a fazer parte do fecho. Após analisar todos os pontos e não houverem mais possibilidades de relaxamento o menor fecho convexo que abrange todos os estoque é encontrado.

4.2. Seleção de Estoques Por Abrangência

Ao obter a lista de estoques selecionados, podem haver estoques próximos um ao outro. Com objetivo de reduzir a sobreposição é realizado uma rotina que busca eliminar pontos redundantes do fecho. Dessa forma é calculado o ponto médio m e, caso um estoque p possua um raio de abrangência r em relação a m que não alcance algumas das extremidades do mapa, p é desconsiderado e dado como um estoque pouco eficiente.

4.3. Comportamento do Agente

Ao surgir uma demanda de trabalho o agente (independente do tipo) irá analisar qual a sua latitude e longitude atual e consultar a crença do fecho calculado. Ao obter o ponto do fecho, mais próximo dado a sua localização atual ele irá se locomover até o estoque para buscar ou entregar itens, conforme apresenta o algoritmo 2.

Algoritmo 2: Qual estoque utilizar

```
Entrada: Lat, Lon
Saída: Estoque Selecionado
início
    Pontos = Obter pontos do fecho;
    Estoque Selecionado = Obter ponto mais próximo(Lat,Lon,Pontos);
fim
```

4.4. Testes Preliminares

Com o objetivo de validar as estratégias aqui apresentadas foi proposto um cenário de teste contra a equipe Akuanduba-UDESC. Os cenários propostos foram as cidades de Berlim, Copenhague e Paris. Os testes preliminares foram executadas em sementes aleatórias, em partidas com mil *steps* e com trinta e quatro agentes em cada equipe, entre eles quatro drones, dez carros, doze caminhões e oito motos. Os critérios para analise de desempenho foram quem arrecadou mais *massium* (moeda da partida) e realizou mais entregas de trabalhos.

O *massium*, é obtido como recompensa de trabalhos entregues pelas equipes. A probabilidade de surgir uma demanda de trabalho em dado *step* da partida é de 20%, desse modo, em média uma partida de mil *steps* surgem duzentas demandas de trabalho. Porém dado a complexidade dos mesmos, a proporção dos que surgem e dos que são atendidos, ficam em torno de 4%. Sendo assim, ao executar a solução proposta nesse artigo contra a solução da equipe Akuanduba-UDESC, foi observado que a solução distribuída tem desempenho melhor que a solução centralizada, conforme a tabela 2. Sendo a equipe *A* o desempenho da solução apresentada nesse artigo e a equipe *B* a solução da equipe Akuanduba-UDESC.

Tabela 2. Resultados preliminares

Cenário	Dinheiro		Quantidade Trabalhos da Partida	Trabalhos Atendido (%)	
	A	B		A	B
Paris	458	257	195	1,03	0
Copenhague	1068	860	189	2,11	1,59
Berlim	2036	2757	203	4,43	4,43

5. Considerações Finais

Nesse artigo pode-se apresentar como os agentes podem ser coordenados com abordagem distribuída dentro de uma cidade inteligente, com uma arquitetura modular e com o uso de uma fila de prioridades. A arquitetura pode ser adaptada e utilizada para problemas recorrentes do mundo real, pois, ela proporciona criar novas tarefas e novas funcionalidades sem muitas linhas de código. Ao propor uma abordagem distribuída no gerenciamento de estoque, para melhorar a eficiência no atendimento de demandas de trabalho, obteve resultados preliminares satisfatórios. De modo que a solução atual apresentou resultados melhores, que os obtidos anteriormente pela equipe Akuanduba-UDESC na competição MAPC - 2018.

Referências

Ahlbrecht, T. (2018). Simulation platform for the multi-agent programming contest.

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

Christie, A. A., Joye, M. P., and Watts, R. L. (2003). Decentralization of the firm: theory and evidence. *Journal of Corporate Finance*, 9(1):3–13.

de Sousa Pissardini, R., Wei, D. C. M., and da Fonseca Júnior, E. S. (2013). Veículos autônomos: conceitos, histórico e estado-da-arte. In *Anais do XXVII Congresso de Pesquisa e Ensino em Transportes–ANPET*, page 2.

FAPESP (2018). Chamada em cidades inteligentes tem resultado de etapa de enquadramento.

Longaray, A. A. and Beuren, I. M. (2001). Cálculo de minimização dos custos de produção por meio da programação linear.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a bdi-architecture. *KR*, 91:473–484.

Taniguchi, E. (2012). The future of city logistics.

Wooldridge, M. (2001). Intelligent agents: The key concepts. In *ECCAI Advanced Course on Artificial Intelligence*, pages 3–43. Springer.