Exercício Prático 2: MAPC Agents on Mars

Alan Rafael Fachini[1], Lucas Nascimento[1], and Márcio F. Stabile Jr.[2]

[1] PCS-EPUSP
[2] IME-USP

Resumo Este relatório descreve a implementação de um time usando a abordagem de sistema multi-agente para a competição no senário "Agents on Mars" desenvolvida pelo "Multi-Agent Programming Contest" (MAPC) [3]. Neste senários os agentes devem encontrar as melhores zonas com os maiores pesos nos nós do grafo. A implementação apresentada usou como base o sistema "LTI-USP Team" desenvolvida por Franco e Sichman [2] para a competição de 2013, e testes com estratégias diferentes para a organização dos times foram realizados a fim de tentar identificar estratégias mais eficientes.

1 Introdução

O "Multi-Agent Programming Contest"(MAPC) é uma competição realizada todos os anos, onde são propostos problemas de domínios variados para a resolução utilizando sistemas Multi-agentes. Seu principal objetivo é o fomento a pesquisa nessa área de estudo. No MAPC, dois times de agentes competem diretamente no mesmo ambiente, oferecendo assim uma oportunidade para a comparação de estratégias e arquiteturas. Desde 2011 o cenário proposto, Agents on Mars, incentiva soluções baseadas em cooperação e coordenação. O objetivo geral do cenário é o controle de zonas em um mapa, representado por um grafo, por se posicionar os agentes em posições, vértices apropriados. A história de fundo do cenário gira em torno da exploração de marte por áreas com concentração de água.

O presente artigo apresenta a estratégia para o time ALM, desenvolvido com base no modelo do time participante da competição de 2013 do Laboratório de Técnicas Inteligentes (LTI) da USP. O time ALM foi desenvolvido como parte da avaliação da disciplina de Sistemas Multi-Agentes pelos alunos Alan Fachini, Lucas Nascimento dos Santos da Silva e Márcio Fernando Stabile Júnior.

2 System Analysis and Design

O time ALM foi desenvolvido com base no time LTI-USP participante das competições de 2012 e 2013. O time ALM, assim como o time em que é baseado, utiliza a framework de desenvolvimento para sistemas multi-agentes JaCaMo [1].

[3] `http://multiagentcontest.org/2013`

JaCaMo é uma plataforma para a programação de sistemas multi-agentes que suporta vários níveis de abstração (agentes, ambiente, organização). JaCaMo combina três tecnologias para isso: Jason , para programar agentes autônomos; CArtAgO (Common ARTifact infrastructure for AGents Open environments), para programar artefatos do ambiente e Moise, para programar organizações de multi-agentes.

Nenhuma metodologia especifica para o desenvolvimento de sistemas multi-agentes foi utilizada, visto nenhum dos membros possuírem experiência para a mesma e dado o tempo necessário para a conclusão do trabalho.

Baseado no time LTI-USP da versão da competição de 2013, nosso time apresenta uma proposta não centralizada. Apesar de apresentar um agente coordenador, cada agente possui autonomia para decidir quais vértices irá ocupar para criar ou expandir uma zona. Cada agente possui seu modelo interno do ambiente e se comunica com outros agentes para informar a estrutura do mapa, posições dos oponentes ou solicitar reparos. De acordo com as regras da competição de 2013 o time é composto de 28 agentes.

A arquitetura dos agentes é baseada no modelo BDI, com cada agente possuindo sua base de crenças, desejos e intenções. Cada agente tem autonomia para decidir sua próxima ação em cooperação com os demais.

Os agentes se comunicam em forma de broadcast enviando, uns para os outros, mensagens somente em relação às novas percepções recebidas, assim os agentes só enviam mensagens nos casos em que a percepção altera o modelo interno do ambiente do agente. A sobrecarga causada por esse tipo de comunicação no começo da simulação, quando as percepções recebidas pelos agentes são novas, diminui com o progresso da simulação conforme os agentes completam um modelo do ambiente.

O ALM é um sistema multi-agente real no sentido que seus agentes são autônomos, proativos e reativos. São autônomos no sentido de decidir por si próprios qual a próxima ação a ser realizada, ainda que em coordenação com outros agentes e com o agente coordenador. Eles apresentam comportamento proativo quando selecionam o melhor vértice no mapa para se movimentarem. E apresentam comportamento reativo quando, por exemplo, o agente realiza uma recarga ao ficar com baixa energia.

3 Software Architecture

A arquitetura do time ALM é a mesma utilizada pelo time LTI-USP na competição de 2013, exibida na Figura 1. Nessa arquitetura foi utilizada a linguagem de programação Jason que provê conceitos de BDI, disponibilizando conceitos abstratos tais como planos, crenças e objetivos. Jason é baseado na plataforma Java e implementa uma extensão da linguagem de programação para agentes AgentSpeak. Jason provê speach acts que são utilizado no trabalho para a comunicação dos agentes. Uma vantagem da utilização da linguagem Jason é a possibilidade de invocar métodos escritos em Java o que permite maior versatilidade na implementação dos agentes e suas ações. Por exemplo, no presente

trabalho, conforme descrito anteriormente, a representação interna dos agentes do mundo, ou ambiente, foi implementada em Java na forma de um grafo. Os planos que compõem os agentes são definidos em AgentSpeak, e o agente decide qual plano utilizar de acordo com suas crenças e representação interna do mundo.

O modelo interno do mundo utilizado pelos agentes foi criado em Java usando estruturas de dados e classes, que representam todos os aspectos recebidos do simulador. A cada interação da simulação os agentes atualizam seu modelo interno do mundo de acordo com as percepções recebidas do servidor.

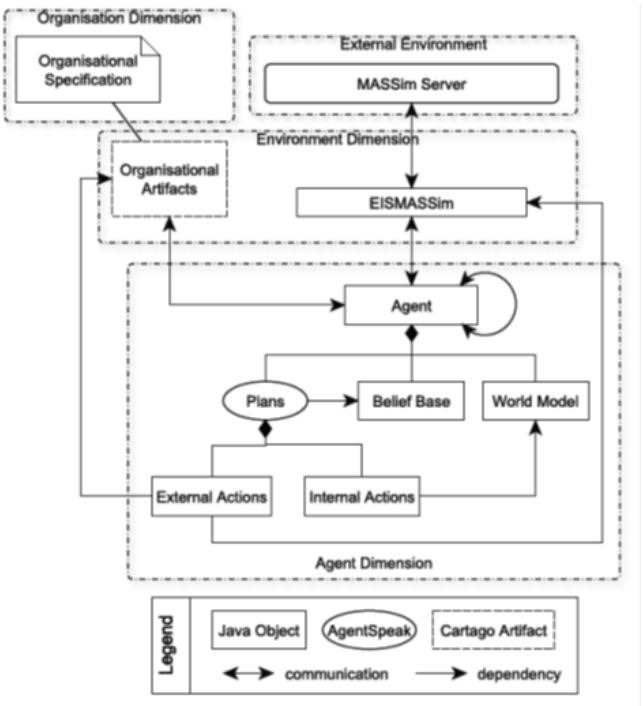


Figura 1: Arquitetura dos times LTI-USP e ALM.

A framework CArtAgO foi utilizada para usar os artefatos organizacionais disponíveis em Moise. CArtAgO é uma framework para programação de ambientes baseada em Agentes & Artifacts (A&A) para desenhar e modelar sistemas multi-agentes. A framework inclui várias metáforas baseadas em ambientes cooperativos de sociedades humanas, tais como: agentes como entidades computacionais que executam uma tarefa (análogo a trabalhadores humanos); e artefatos, como recursos e ferramentas dinamicamente construídos para o uso e manipulação dos agentes na execução de suas tarefas. Cada artefato pode ser usado e manipulado pelos agentes em tempo de execução. Nesse trabalho não foram

desenhados novos artefatos para o uso dos agentes, tal qual o trabalho no qual o presente projeto é baseado os únicos artefatos existentes são os relacionados ao modelo organizacional feito em Moise.

Moise é um modelo organizacional para sistemas multi-agentes baseado em três dimensões: estrutural, funcional e normativa. A dimensão estrutural é construída sob três níveis. O nível individual responde pelo comportamento que um agente adota ao assumir determinado papel. O nível social diz respeito a relação entre papeis e comunicação. E o nível coletivo responde pela agregação de papéis em grupos. A dimensão funcional é formada por planos, metas e missões que representam a forma pela qual o sistema multi-agentes alcança o seu objetivo global. O objetivo global é decomposto em planos, que são distribuídos aos agentes na forma de missões, quando um agente recebe uma missão ele se responsabiliza por todas as metas que compõem a missão. E a dimensão normativa, ou deôntica, abrange a autonomia dos agentes, especificando o que é ou não permitido dentro da organização. A especificação dessa dimensão diz respeito as permissões e obrigações relacionadas a um papel. O modelo em Moise permite ao desenvolvedor delimitar as restrições da sociedade de agentes, e também pode ser usada para que os agentes raciocinem em relação a sua organização. Os agentes se comunicam com o servidor MASSim por meio da interface EISMASSim, baseada em EIS, e distribuída para a competição. Para realizar a comunicação com a interface do servidor para da competição a arquitetura padrão foi alterada para atuar não somente em artefatos do CArtAgO, como também no ambiente EIS.

4 Strategies, Details and Statistics

De acordo com a definição do cenário criada para o MAPC, são definidos cinco tipos diferentes de agentes, cada um com sua especialidade, são: explorador, reparador, sabotador, sentinela e inspetor. Cada classe de agente tem assim um conjunto de ações que lhe são permitidas realizarem, a tabela 1 exibe qual o conjunto de ações para cada tipo de agente.

	explorador	reparador	sabotador	sentinela	inspetor
recharge	X	X	X	X	X
attack			X		
parry		X	X	X	
goto	X	X	X	X	X
probe	X				
survey	X	X	X	X	X
inspect					X
buy	X	X	X	X	X
repair		X			
skip	X	X	X	X	X

Tabela 1: Conjunto de ações por agente.

Usando como base o time LTI-USP, os papeis assumidos por nossos agentes não são diretamente mapeáveis para os times definidos para a competição. Assim tipos adicionais de agentes foram definidos de acordo com a estratégia usada. Cada papel tem uma missão associada que pode ser assumida por um ou mais tipos de agente. Segue a descrição dos tipos originalmente definidos para o time LTI-USP:

- map_explorer(explorador): Explora todo o mapa realizando a ação probe em todos os vértices e realizando a identificação de todas as arestas em seu caminho;
- map_explorer_helper(explorador): Ajuda os agentes do tipo map_explorer em sua tarefa durante os primeiros 250 passos, posteriormente assume o papel de soldier;
- soldier(todos os tipos): Tenta ocupar a melhor área (ou segunda melhor) no ambiente indicada pelo agente coordenador;
- guardian(sabotador): Defende a melhor (ou segunda melhor) área de invasores por ataca-los caso se aproximes da área do time ou tentem invadi-la;
- medic (reparador): Ocupa o centro da área ocupada e é responsável por reparar os agentes no grupo. Os agentes que precisam de reparos se movem em direção aos agentes medic para serem reparados;
- zone_explorer(explorador): Explora a área ocupada pelo time identificando os vértices cujo valor seja desconhecido. Quando toda área é identifica o zone_explorer auxilia os agentes soldier a expandir a área ocupada;
- saboteur(sabotador): Ataca qualquer agente oponente próximo, ou que esteja ocupando um bom vértice;
- sentinel(sentinela): Tenta sabotar o oponente se movendo para dentro da zona inimiga;
- repairer(reparador): Segue os agentes do tipo saboteur para realizar reparos, sempre seguindo a duas células de distância para evirar perigo;
- coordinator(nenhum): agente interno do sistema que possui um modelo interno do ambiente construído com base nas informações passadas pelos outros agentes. Esse agente determina quais as melhores áreas de ocupação e informa os agentes de sua posição.

Em relação ao modelo de agentes utilizado pelo time LTI-USP, o time desenvolvido nesse trabalho muda um pouco a estrutura e os tipos utilizados. Diferente do time LTI-USP nossa abordagem não utiliza os agentes do tipo guardian, optando por uma estratégia mais agressiva. Passamos assim os agentes que estavam sendo usados como guardian para o tipo saboteur com a missão exclusiva de eliminar oponentes. Seguindo essa linha, devido ao maior número de agentes do tipo saboteur atacando preferimos transferir o agente do tipo medic responsável pelo reparo nas zonas ocupadas, para o tipo repairer no grupo de ataque, para prestar assistência aos nossos agentes saboteur. A Figura 2 mostra o modelo Moise após essas alterações.

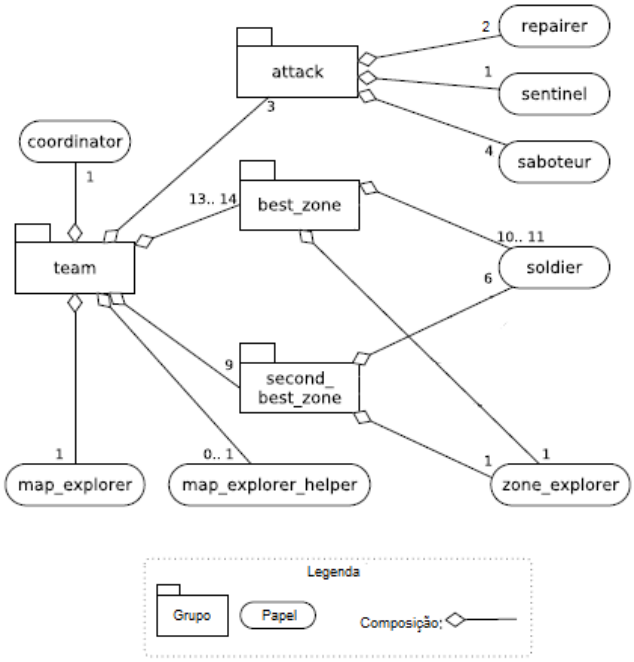


Figura 2: Modelo Moise dos tipos de agente utilizados pelo time ALM

5 Experimentos

As partidas no ambiente MAPC acontecem em mapas gerados aleatoriamente dados parâmetros que definem o número de arestas e vértices. Cada time joga três vezes um contra o outro em partidas que duram 750 passos. Para os experimentos realizados neste trabalho escolheu-se reduzir o número de passos para 200, visto que nos experimentos realizados um dos times já havia vencido até esse passo.

Os experimentos realizados colocaram o time ALM para competir contra o time LTI-USP que ficou em terceiro lugar na competição MAPC de 2013. Os times competiram em quatro ambientes diferentes definidos pelas propriedades apresentadas na Tabela 2. Procurou-se definir ambientes similares aos testados pela equipe que desenvolveu o time LTI-USP. Testou-se ambientes com diferentes quantidades de vértices e aretas. A coluna semente define a semente utilizada para se gerar os mapas. Também é possível encontrar os arquivos de configuração utilizados no repositório de código deste trabalho [4]. A Figura 3 apresenta gráfica dos ambientes utilizados nos testes.

[4] Repositório de Código: https://github.com/mfstabile/PCS5703

	Vértices	Arestas	N. Zonas	Semente
Ambiente 1	500	40%		1402107246426
Ambiente 2	500	40%		1402129496795
Ambiente 3	600	10%		1402141987750
Ambiente 4	400	20%		1402235674563

Tabela 2: Conjunto de ações por agente.

6 Análise dos Resultados

7 Conclusion

1. What have you learned from the participation in the contest?
2. Which are the strong and weak points of the team?
3. How suitable was the chosen programming language, methodology, tools, and algorithms?
4. What can be improved in the context for next year?
5. Why did your team perform as it did? Why did the other teams perform better/worse than you did.
6. Which other research fields might be interested in the Multi-Agent Programming Contest?
7. How can the current scenario be optimized? How would those optimization pay off?

Short Answers

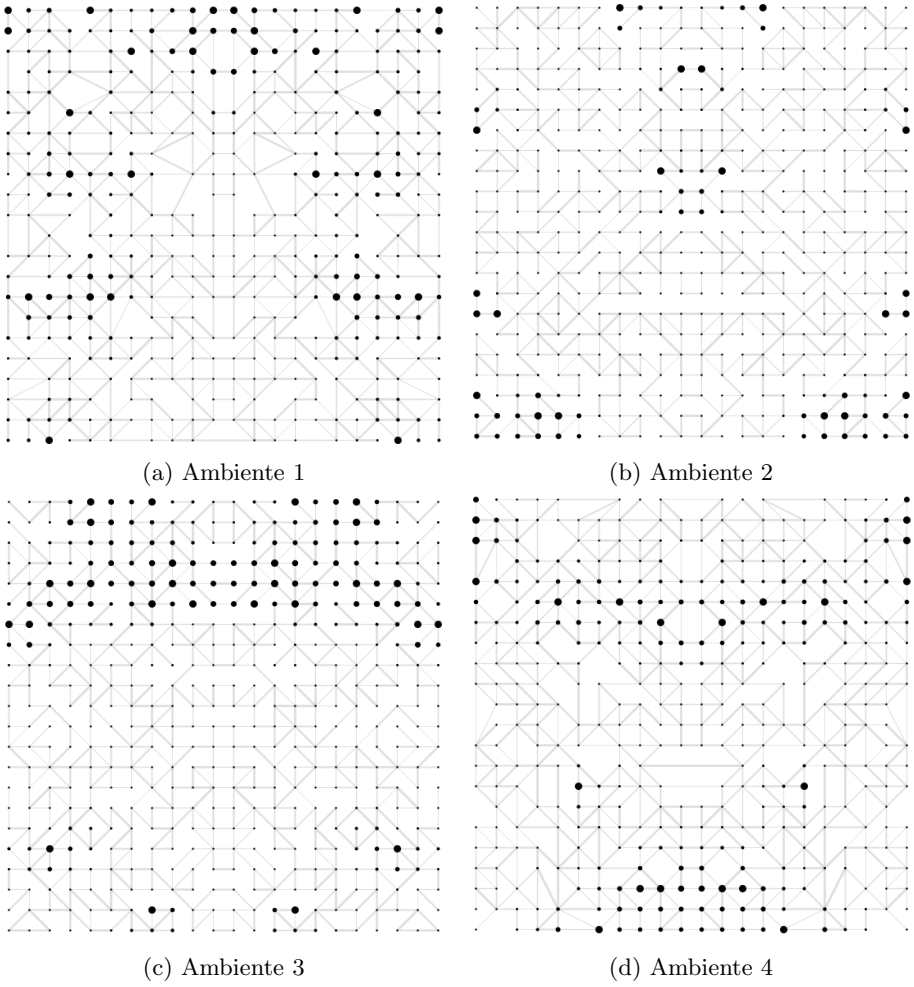
Please provide short answers to all the questions in a separate section. This does not count for the 10 pages limit. Please use the following style for this section:

```
\newpage
\section*{Short Answers}
\appendix
\section{Introduction}
\begin{enumerate}
\item What was the motivation to participate in the contest?
\item[A:] Our motiviation was ...
\item What is the (brief) history of the team?
(MAS course project, thesis evaluation, $\ldots$)
\item[A:] In 2006...
\end{enumerate}
```

Please note: The `A:` stands for "'Answer"'.

Referências

1. Olivier Boissier, Rafael H. Bordini, Jomi Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, 78(6):747–761, June 2013. Special section on Agent-oriented Design Methods and Programming Techniques for Distributed Computing in Dynamic and Complex Environments.
2. Mariana Ramos Franco and Jaime Simão Sichman. Improving the lti-usp team: A new jacamo based mas for the mapc 2013. In *Engineering Multi-Agent Systems*, pages 339–348. Springer, 2013.



(a) Ambiente 1

(b) Ambiente 2

(c) Ambiente 3

(d) Ambiente 4

Figura 3: Ambientes testados nos experimentos