Universidade do Minho

Agentes Inteligentes

Trabalho Prático
Agentes e Sistemas Multiagente

Mestrado de Engenharia Informática

Grupo 1

Ângela Barros - PG38407

Rui Fontes - PG39296

Braga, Portugal

27 de Outubro de 2019

Conteúdo

Lista de Figuras

1 Contextualização

O cenário escolhido para este projeto é um simulador de combate a incêndios florestais com o uso de veículo autónomos e inteligentes (voadores e terrestres), através do uso de aeronaves, drones e camiões de bombeiros (agentes participativos). Estes agentes serão monitorizados por um quartel de bombeiros (agente central), responsável pela coordenação e distribuição de tarefas entre os agentes participativos.

A tomada de decisão por parte do quartel de bombeiros será influenciada por um conjunto de fatores existentes no meio, tais como: localizações de locais de abastecimento disponíveis de água (rios, mares, lagoas entre outros) e combustível. Em contrapartida, como forma de gerar os incêndios em momentos/posições aleatórias, deverá ser criado um agente secundário (ateador de fogos) responsável por esta tarefa.

1.1 Estruturação do Documento

Relativamente à estrutura deste documento, no capítulo 1, é realizada uma pequena introdução do projeto. Por último, é abordada a estrutura do mesmo documento.

No capítulo 2 será apresentado o estudo do Estado da Arte. No capítulo 3 serão apresentados os diagramas UML desenvolvidos na fase de conceptualização do problema. A arquitetura encontra-se no capítulo 3.1 juntamente com o *Workflow* do projeto. No capítulo 4 é feita uma conclusão do trabalho desenvolvido até à data, e é ainda fornecida informação sobre trabalho futuro no capítulo 4.2.

2 Estado da Arte

Um Sistema Multiagente (SMA) é um sistema computacional em que dois ou mais agentes interagem ou trabalham em conjunto de forma a desempenhar determinadas tarefas ou satisfazer um conjunto de objectivos.

Um dos pontos essenciais para permitir a construção de sociedades de agentes, consiste em conseguir gerir as interacções e as dependências das actividades dos diferentes agentes no contexto do Sistema Multiagente, ou seja, coordenar esses agentes. Desta forma, a coordenação desempenha um papel essencial nos SMA porque estes sistemas são inerentemente distribuídos.

Diversas metodologias de coordenação foram propostas por diferentes autores, dividindo-se em dois grupos principais:

- metodologias aplicáveis em domínios contendo **agentes competitivos** (*"self-interested"*), ou seja, agentes preocupados com o seu bem próprio;
- metodologias aplicáveis a domínios contendo **agentes cooperativos**, ou seja, agentes que incluem uma noção de preocupação pelo bem do conjunto.

Os Sistemas Multiagente incluem diversos agentes que interagem ou trabalham em conjunto, podendo estes serem homogéneos ou heterogéneos. Cada agente é basicamente um elemento capaz de resolução autónoma de problemas e opera assincronamente, com respeito aos outros agentes.

2.1 Distribuição de Tarefas

A alocação de tarefas pode ser realizada de duas formas:

1. **Alocação Centralizada de Tarefas** - um planeador central é utilizado para alocar as tarefas para os restantes agentes.
2. **Alocação Distribuida de Tarefas** - A tarefa pode ser recebida por todos os agentes e estes comunicam entre si para completar a tarefa.

2.2 Ferramentas de desenvolvimento

- **Aglet** - Sistema para desenvolvimento de agentes móveis desenvolvido pela IBM. Um *aglet* é um agente Java que pode mover-se autonomamente e espontaneamente de um host para outro, carregando consigo um pedaço de código.
- **JADE** - O Java Agent Development Framework, mais conhecido pelo acrónimo JADE, é um framework de software para o desenvolvimento de agentes inteligentes, implementado em Java.
- **Jack** - É uma framework em JAVA para o desenvolvimento de sistemas multiagente. JACK foi desenvolvido por *Agent Oriented Software Pty.* (AOS)

2.3 Exemplos de Sistemas Multiagentes

Existem diversos exemplos reais de Sistemas Multiagentes cujo foco é a coordenação de distribuição de tarefas. A título de exemplo, apresenta-se, de seguida, dois exemplos:

- **Sistema multiagente para coordenação de ambulâncias para serviços médicos de emergência** - Para coordenar ambulâncias em serviços de urgência, um sistema multiagente usa um mecanismo de leilão baseado em confiança. Resultados de testes utilizando dados reais mostram que este sistema pode eficazmente atribuir ambulâncias a pacientes e, consequentemente, reduzir o tempo de transporte. [1]

- ***RoboCup Rescue League* - Competição de simulação de agentes** - A competição de simulação de agentes envolva maioritariamente a avaliação da performance de equipas de agentes em diferentes mapas na plataforma do RoboCup Rescue Agent Simulation (RCRS), um ambiente de simulação para pesquisar/analizar a gestão de respostas em casos de catástrofe. [2]

2.4 Protocolos de Comunicação

2.4.1 Arquiteturas do Sub-Sistema de Comunicação

- **Comunicação Directa** - Os agentes tratam da sua própria comunicação sem intervenção de qualquer outro agente. Para tal, partilham especificações, enviando aos outros agentes as suas capacidades e/ou necessidades de forma a cada agente poder tomar, individualmente, as suas decisões relativas à comunicação.

  Neste tipo de arquitectura cada agente comunica directamente com qualquer outro agente, sem qualquer intermediário. Um dos principais problemas que se coloca nesta arquitectura está relacionado com a inexistência de um elemento coordenador da comunicação, o que pode originar o bloqueio do sistema se, por exemplo, todos os agentes decidirem enviar mensagens ao mesmo tempo.

- **Comunicação Assistida** - Os agentes apoiam-se em agentes especiais designados "agentes facilitadores", de forma a efectuarem a comunicação com os outros agentes. Nesta arquitectura, a organização de agentes é do tipo sistema federado. Nestes casos, se um dado agente i desejar enviar uma mensagem a um outro agente j, terá primeiro de a enviar para o "agente facilitador", que se encarregará de a reencaminhar ao seu destinatário.

  Esta arquitectura resolve parcialmente o problema da coordenação da comunicação e diminui consideravelmente a complexidade necessária aos agentes individuais na realização de comunicação. Os agentes não necessitam de armazenar informações detalhadas sobre todos os outros agentes e nem sequer necessitam de saber o seu endereço de forma a comunicarem com eles. Basta comunicar com o "agente facilitador" . No entanto, a existência do " agente facilitador" pode introduzir uma certa centralização no sistema e um estrangulamento (*bottleneck*) no sistema de comunicações. Se este agente deixar de funcionar, o sistema de comunicações deixa também de funcionar.

A comunicação tem dois fins principais: partilha do conhecimento/informação com outros agentes, e coordenação de actividades entre agentes. No entanto a realização de comunicação que permita atingir estas duas metas, requer a definição de uma linguagem comum, caracterizada por:

- **Sintaxe** - A parte da estrutura gramatical da linguagem que contém as regras relativas à combinação das palavras;

- **Semântica** - Significado dos símbolos e das suas combinações;

- **Vocabulário** - Conjunto de símbolos usados;
- **Pragmática** - Conjunto de regras de ação e fórmulas de interpretação dos símbolos utilizados na comunicação;
- **Modelo de domínio do discurso** - Significado que um conjunto de símbolos assume quando interpretado num determinado contexto de conversação.

3 Diagramas UML

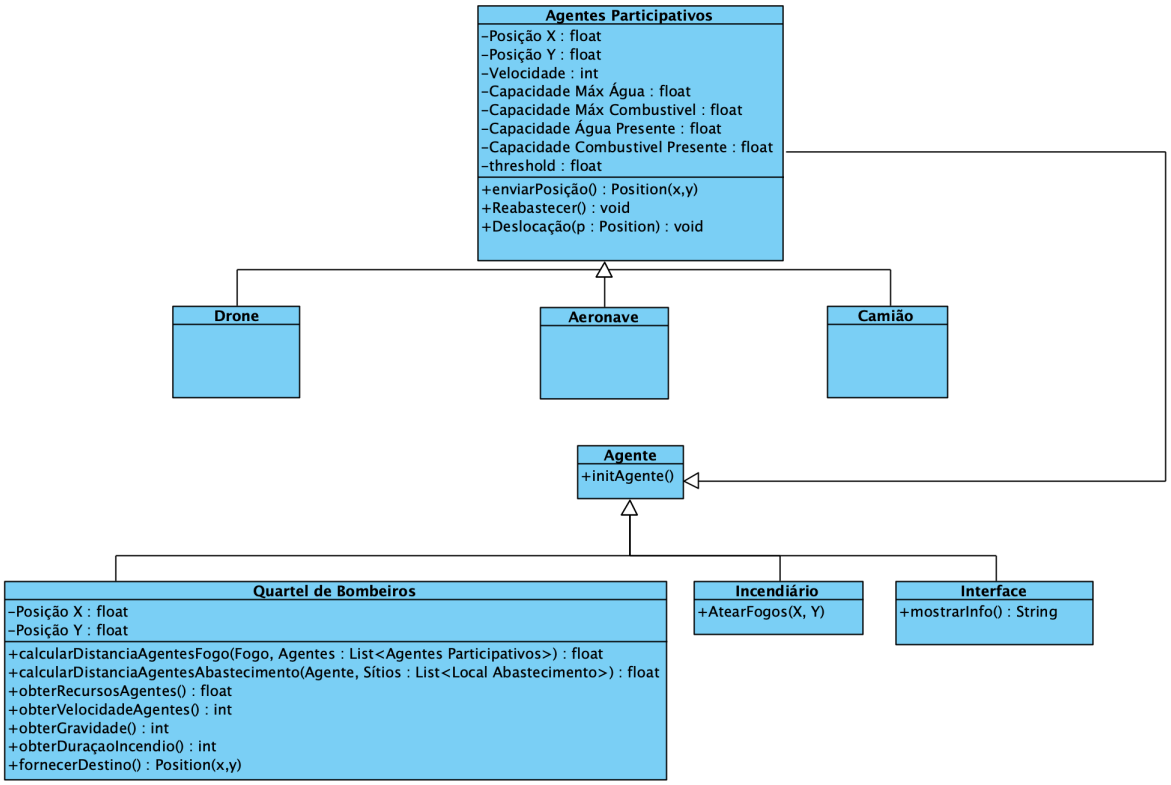

Figura 1: *Diagrama de Classes*

3.1 Classes do Sistema Multiagente

Para o desenvolvimento do presente trabalho foi necessário criar várias classes de forma a dar resposta ao problema em questão. Como tal, a arquitectura do projeto tem como base:

- **Agentes Participativos** - serão os agentes que irão apagar os fogos, ou seja, os Drones, Aeronaves e Camiões. Todos eles terão associados a si uma posição X e Y no mapa. Também terão associados a si uma capacidade máxima de combustível/água tal como a presente capacidade de combustível/água, isto permitirá ao agente *Quartel de Bombeiros* fazer uma gestão dos recursos de cada agente no terreno.

- **Quartel de Bombeiros** - Este é o agente central que irá controlar todos os agentes participativos e verificar a existência de fogos ativos no mapa. Este será responsável por avaliar a distância de todos os agentes participativos até ao incêndio ativo. Após ter todas as distâncias, o Quartel dos Bombeiros deverá escolher o agente participativo que se encontre mais perto do local do incêndio e ordenar que este apague o incêndio, fornecendo-lhe, para tal, as coordenadas do incêndio. Para tal, o agente central irá verificar se cada agente participativo possui recursos acima ou abaixo do *threshold* definido, caso possuam poucos recursos, os agentes participativos em causa serão ordenados a irem abastecer.

Toda a gestão relacionada com o incêndio é uma responsabilidade do agente Central, tal como calcular a gravidade do incêndio ou obter a duração do incêndio.

- **Incendiário** - Agente responsável por iniciar os fogos no mapa. Estes serão gerados em coordenadas aleatórias.
- **Interface** - Agente responsável por demonstrar a informação relativa às posições dos agentes participativos e dos fogos ativos no mapa. Na interface também deverá ser possível ver a deslocação dos agentes ao longo do mapa.

3.2 *Workflow*

O sistema irá ser modelado em função do agente incendiário. Todas as ações dos outros agentes são feitas de acordo com este primeiro agente.

Mal os sensores detetem um incêndio, informam o quartel da sua posição e gravidade. Com esta informação o quartel vai verificar quais são os agentes mais próximos do incêndio e a quantidade de combustível e água que tem disponível.

- Caso o agente tenha recursos suficientes para fazer a viagem, apagar o incêndio e ainda ficar acima do *threshold*, irá ser escolhido para se deslocar e apagar o incêndio (o quartel atualiza no mapa a posição do agente). Quando o incêndio é apagado, o quartel é informado e atualiza o mapa.
- Caso os recursos do agente sejam inferiores ao *threshold*, ele irá deslocar-se à estação de abastecimento mais próxima para reabastecer. Mal chegue ao seu destino, o quartel irá atualizar o mapa.

O *threshold* é uma medida de segurança criada pelos programadores, de modo a que o agente possa ter combustível suficiente para se deslocar para uma estação de abastecimento e poder reabastecer.

Com esta sequência, mesmo que nenhum agente tenha recursos suficientes para se deslocar para o incêncido e o apagar, uma vez reabastecidos, irão ser novamente considerados no cálculo de distância ao incêndio. Deste modo, impede-se que um agente se mova e depois não possa ir reabastecer.

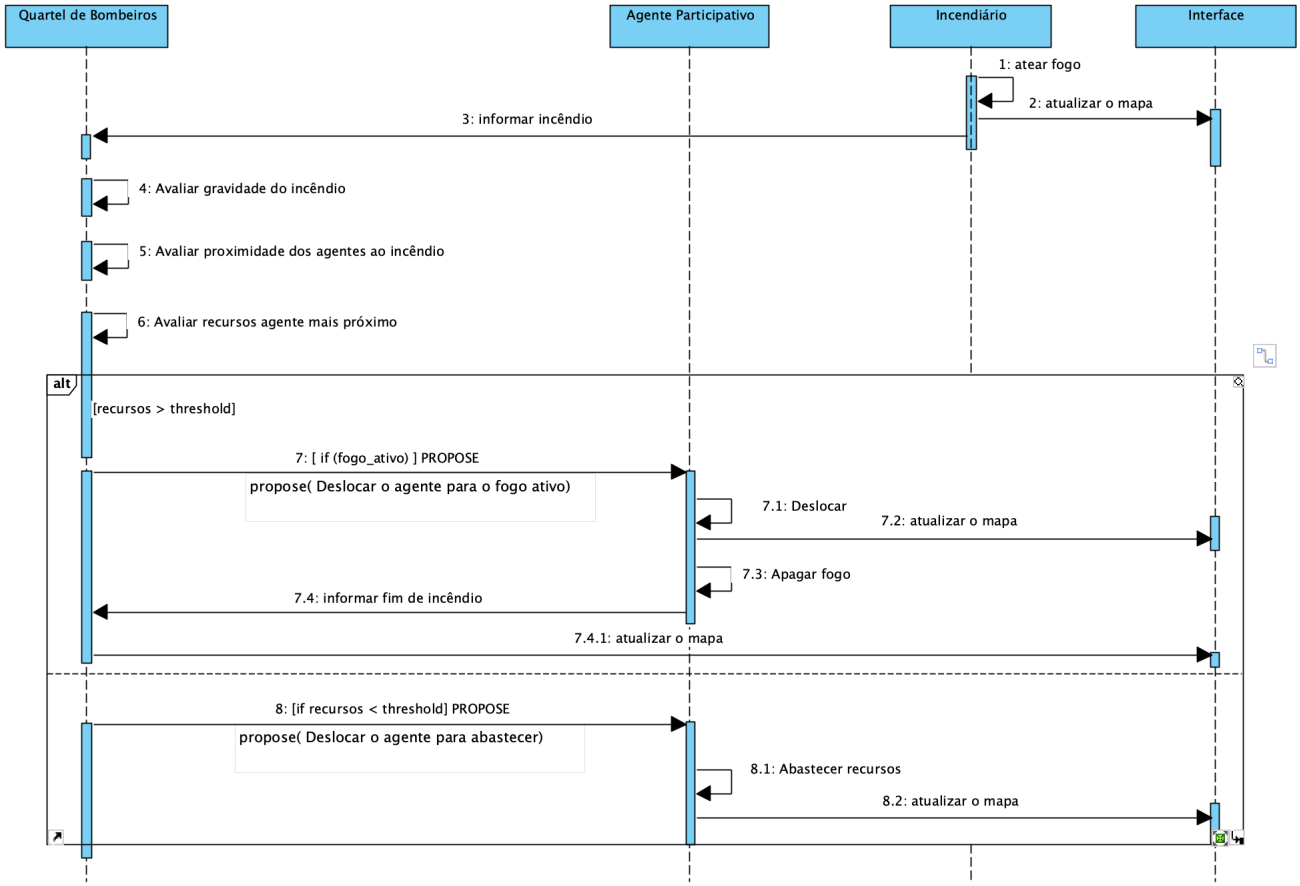

Figura 2: *Diagrama de Sequência*

3.3 O Raciocínio dos Agentes Participativos

No diagrama de estados da figura 3, encontra-se o raciocínio dos agentes participativos.

Enquanto que não tem nenhuma ação para executar, o agente está em estado *idle*, ou seja, em estado inativo. Mal o quartel receba informação de que há necessidade de movimento, este faz os cálculos de distâncias no contexto do problema e, mal acabe os cálculos, dá ordem de movimento ao agente participativo. Para a ação de movimento, há duas possíveis execuções. Ou um agente se move para apagar um incêndio, ou se move para reabastecer.

Quando o agente chega ao seu destino, seja ele qual for, irá executar a ação adequada (apagar o incêndio ou reabastecer). Uma vez terminada, o agente informa o Quartel do seu sucesso e, mais uma vez, o agente fica inativo, aguardando novas intruções.

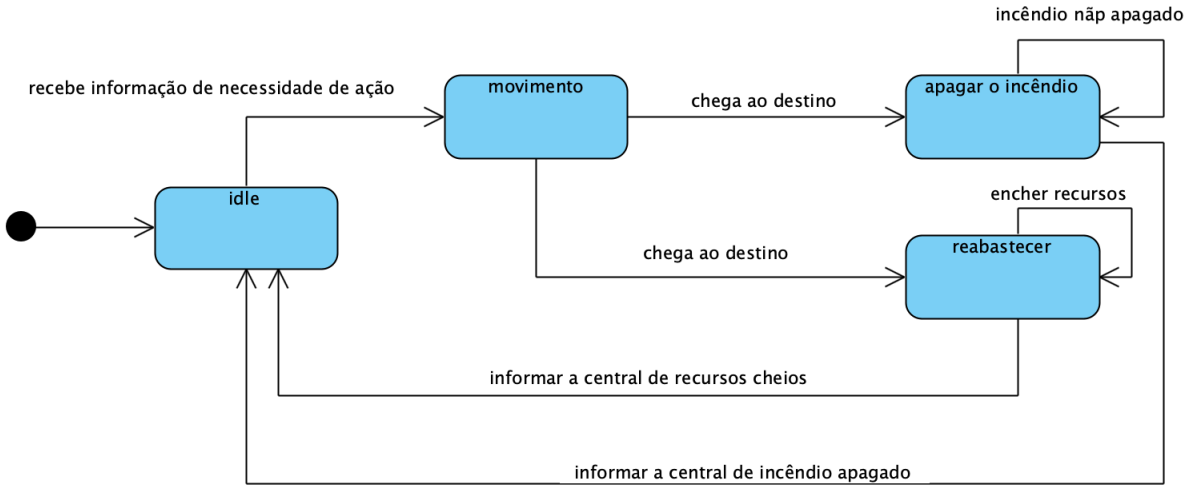

Figura 3: *Diagrama de Estados*

4 Conclusão

Para esta primeira fase do desenvolvimento do projeto foram encontrados alguns desafios, contudo, alguns pontos já foram identificados que poderão tornar-se em problemas no futuro.

4.1 Problemas identificados

No estado atual do desenvolvimento do projeto, o agente central pode vir a tornar-se um *bottleneck* e fazer com que a performance do sistema sofra como consequência. Algo que poderá ser considerado será a passagem de um maior número de tomadas de decisão por parte dos agentes participativos, aliviando assim o *bottleneck* que possa criar.

Também nesta fase do projeto, poderá haver um possível problema no que diz respeito ao abastecimento dos agentes participativos pois quem toma essa decisão é o agente central *após* ter sido comunicado a ocorrência de um incêndio. Ou seja, no pior caso poderá acontecer que todos os recursos estejam com os recursos abaixo do *threshold* fazendo com que *nenhum* agente participativo consiga ir apagar o incêndio. Contudo, tendo em conta que há 17 agentes participativos, e assumindo que a probabilidade de cada um deles ter os recursos abaixo do *threshold* ser 1/2 e considerando que são eventos independentes, dá uma probabilidade do cenário de pior caso de 0,0007%, ou seja, há uma probabilidade de 0,0007% de que todos os agentes participativos tenham poucos recursos e que todos precisem de ir abastecer atrasando assim o processo de apagar o incêndio. Para já, optou-se por esta opção de desenvolvimento para se evitar o uso desnecessário de recursos, contudo, tendo em conta que a eficiência a apagar incêndios também é fulcral, aquando do desenvolvimento será feita uma análise para avaliar o *trade-off* entre utilização de recursos e a eficácia ao apagar os incêndios. Portanto, a opção de avaliação dos recursos estar encarregue ao agente central ou ao próprio agente participativo é algo que, para já, fica em aberto.

4.2 Trabalho futuro

Como trabalho futuro, será necessário desenvolver todo o código necessário para simular o ambiente do problema descrito. Isso será feito num modo incremental para garantir que todo o código que seja feito consegue manter uma boa performance e os agentes participativos consigam eficazmente apagar os incêndios. Caso a performance da solução que foi projetada não seja boa o suficiente, modificações serão feitas de forma a se poder garantir que a performance seja razoável o suficiente para o problema.

Tendo em conta o tempo restante para o desenvolvimento do projeto, será possível garantir uma versão estável do ambiente de simulação em que os agentes consigam eficazmente apagar os fogos e que as mensagem sejam mostradas através de uma consola simples. A utilização de uma interface gráfica ficará para uma iteração posterior que será atendida caso a equipa de desenvolvimento ainda tiver tempo suficiente de desenvolvimento. O objetivo principal nesta fase é garantir uma versão estável de todo o sistema, mesmo que seja numa interface não tão *user-friendly*.

Referências

[1] Beatriz López, Bianca Innocenti, and Dídac Busquets. A multiagent system for coordinating ambulances for emergency medical services. *Intelligent Systems, IEEE*, pages 50 – 57, 10 2008.

[2] RoboCup. Agent simulation,http://rescuesim.robocup.org/competitions/agent-simulation-competition/, 2019.