Exploração de Marte

Agentes e Inteligência Artificial Distribuída

Universidade do Porto

Faculdade de Engenharia

FEUP

Dezembro 2016

Grupo T05_4
Marina Camilo - up201307722 - up201307722@fe.up.pt
Diogo Ferreira - up201502853 - diogoff@fe.up.pt
Ângela Cardoso - up200204375 - angela.cardoso@fe.up.pt

Conteúdo

Capítulo 1

Objetivo

1.1 Descrição do cenário

No âmbito da unidade curricular de Agentes e Inteligência Artificial Distribuída, o nosso grupo propôs-se a implementar um Sistema Multi-Agente para simulação de um cenário de extração de minérios em Marte. Para tal, é necessário descobrir os minérios, extraí-los e transportá-los para a base. Sendo assim, no sistema pretendido existem três tipos de Agentes:

- *Spotter* – Procura fontes de minérios e inspeciona-as para determinar se podem ser exploradas.
- *Producer* – É chamado a uma fonte de minério por um *spotter* para extrair o máximo de minério possível nessa fonte.
- *Transporter* – É alocado pelo *producer* para carregar o minério obtido para a base.

De forma a facilitar a procura, todos os agentes podem localizar fontes de minérios e enviar a sua localização para os *spotters* que os analisarão. A escolha do *producer* por parte do *spotter* segue um protocolo de negociação. A alocação dos *transporters* a uma determinada fonte segue também um protocolo de negociação, iniciado pelo *producer*. Esta alocação, terá em conta a quantidade de minério a transportar, de modo a determinar mais corretamente o número necessário de *transporters*.

1.2 Objetivos do trabalho

Um dos objetivos deste trabalho é implementar os agentes de forma a que a simulação da exploração seja tão eficiente quanto possível. Para tal devem ser estudadas várias alternativas de implementação, de forma a determinar qual a melhor abordagem.

No caso dos agentes do tipo *spotter*, podem ser utilizados diferentes algoritmos de distribuição do espaço a explorar entre eles, para que cubram toda a região

mais rapidamente. Em relação aos *producers* e *transporters*, podem ser instalados diferentes protocolos de negociação que garantam que o melhor agente é escolhido para a tarefa.

O nosso principal objetivo é utilizar este projeto como forma de melhor interiorizar os conceitos dos Sistemas Multi-Agente, nomeadamente ganhando uma maior familiaridade com as plataformas que permitem implementar e simular estes sistemas.

Capítulo 2

Especificação

2.1 Identificação e caracterização dos agentes

arquitetura, comportamento, estratégias

processos de raciocínio - novo no relatório final

Tal como descrito acima, o nosso sistema tem três tipos de agentes. De seguida descrevem-se mais detalhadamente estes agentes e a forma como decidimos implementá-los.

2.1.1 *Spotter*

A nossa intenção é que cada *spotter* tenha a sua região conexa do espaço para explorar. Uma vez definida essa região, a estratégia do *Spotter* é simples, inspeciona cada célula consecutivamente, determinando se contém minério extraível e guardando registo desta informação. Sempre que encontra minério extraível, entra em contacto com todos os *producers* para negociar e decidir qual deles será chamado para extrair o minério.

Numa segunda fase, depois de introduzirmos as restrições no sistema, os *spotters* receberão notificações dos restantes agentes quando estes últimos detetarem minérios. O *spotter* a quem estiver atribuída essa célula ficará com a responsabilidade de a inspecionar e determinar se o minério pode ser explorado. Em princípio, o mesmo acontecerá na primeira movimentação dos *spotters* para as suas regiões. Isto é, eles detetaram minério e espalharão essa informação, mas apenas inspecionam a sua região.

Mais tarde, pretendemos explorar diferentes formas de dividir o espaço entre os *spotters*, para determinar qual a melhor forma. Entre outras coisas, para que os *producers* iniciam a sua atividade mais cedo, pode ser melhor que na movimentação inicial para as suas regiões os *spotters* inspecionem minério encontrado, mesmo que não esteja nas suas regiões.

2.1.2 *Producer*

Os *producers* entram em ação assim que é encontrado minério extraível por um *spotter*. Nesse momento cada um deles terá que determinar o esforço necessário para se deslocar ao local e extrair o minério. Esse esforço corresponde à distância (em passos) a que o *producer* se encontra do local, mais o tempo (em passos) que ele demorará a terminar a sua tarefa atual, caso esteja a extrair minério, mais o tempo que demorarão todas as outras tarefas com as quais já se tenha comprometido. Para esse efeito, cada *producer* deve guardar uma fila contendo informação sobre as suas extrações, em particular o tempo que cada uma delas demora, incluindo a deslocação da sua posição anterior até ao local da extração.

Numa primeira fase de implementação, assumiremos princípios muito simples relativamente à extração, nomeadamente que requer sempre o mesmo número de passos. Mais tarde, a quantidade e/ou profundidade do minério poderão ser levadas em conta para determinar o tempo de extração.

2.1.3 *Transporter*

Os *transporters* iniciam a sua atividade quando são contactados para transportar minério. À semelhança do que acontece com os *producers*, nesse momento terão de calcular o esforço que necessitam para se deslocarem ao local. Inicialmente, dado que se assume capacidade ilimitada e transporte do minério para a base imediatamente, o esforço corresponde apenas à soma das distâncias que o *transporter* terá que percorrer para se deslocar entre os locais para os quais foi convocado e a base.

Quando for considerada a limitação de carga, o *transporter* poderá ter vários comportamentos diferentes que serão estudados quanto à sua eficiência. Ele poderá regressar à base apenas quando atinge a carga máxima, ou então regressar sempre que não estiver ocupado com transportes. Quando informar os *producers* do esforço que necessita para chegar até um local terá que considerar quais as viagens à base que fará e também qual o espaço que terá disponível quando lá se deslocar. Tudo isso será considerado pelo *producer* ao escolher o seu *transporter*. Se o tempo permitir, consideraremos ainda a possibilidade de vários *transporters* serem alocados à mesma tarefa, caso a quantidade de minério assim o exija ou essa pareça ser a ação mais eficiente.

2.2 Protocolos de interação

incluir diagramas de interação UML, se aplicável

2.2.1 Divisão de espaços

Inicialmente cada *spotter* deve acordar com os restantes a região do espaço que será sua para explorar.

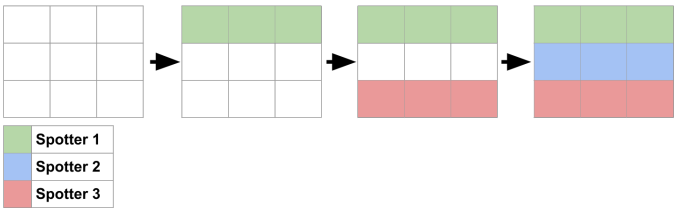

Figura 2.1: Alocação simples por linhas

Sendo o espaço uma matriz retangular, numa primeira fase de implementação, assumiremos que o número de *spotters* é igual ao número de linhas da matriz, de forma a que cada *spotter* fique com uma linha. A Figura 2.1 ilustra esta divisão para três *spotters*.

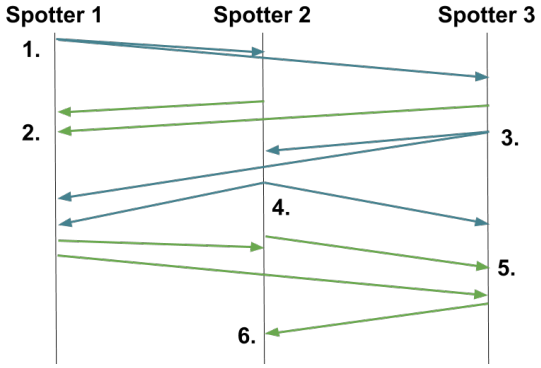

Figura 2.2: Diagrama temporal das comunicações entre *spotters*

Cada *spotter* escolhe uma linha e depois comunica aos restantes a sua escolha, que ficam encarregues de confirmar a afetação. Esta comunicação, correspondente ao exemplo acima, é ilustrada na Figura 2.2:

1. *Spotter* 1 comunica aos restantes o espaço que pretende explorar;
2. *Spotter* 1 recebe confirmação dos restantes e fica afeto ao espaço que pretendia;
3. *Spotter* 3 comunica aos restantes o espaço que pretende explorar;
4. *Spotter* 2 comunica aos restantes o espaço que este pretende explorar;
5. *Spotter* 3 recebe confirmação dos restantes e fica afeto ao espaço que pretendia;
6. *Spotter* 2 recebe confirmação dos restantes e fica afeto ao espaço que pretendia.

Neste exemplo, assumiu-se que não há sobreposição nas escolhas. Quando houver sobreposição, alguns dos *spotters* terão que desistir das suas escolhas e comunicar novas, aguardando confirmação.

2.2.2 Afetação de *producers*

Uma vez encontrado minério é necessário chamar um *producer* para o extrair. O *spotter* envia então a posição do minério a todos os *producers*. Estes respondem com o valor do esforço de que necessitarão para se deslocar até ao local. O *spotter* escolhe o *producer* com o menor esforço e comunica com ele, pedindo para confirmar a afetação. Caso seja recusado, porque o *producer* foi afeto a outro minério entretanto, o *spotter* repete o processo desde o início.

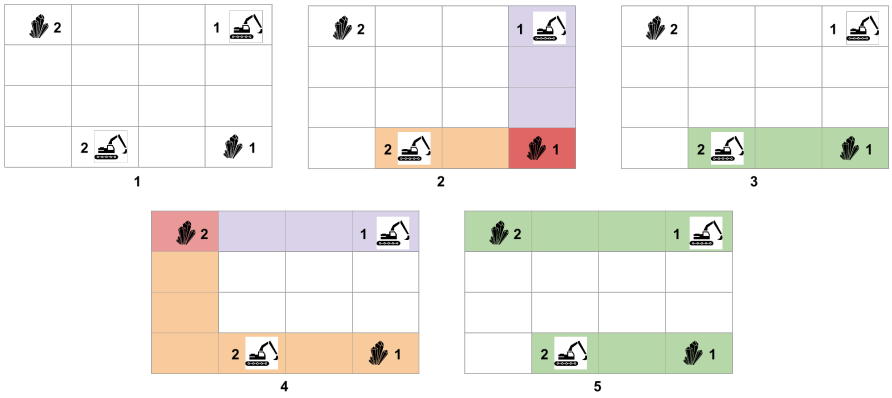

Figura 2.3: Exemplo de afetação de *producers*

A Figura 2.3 exemplifica o processo de escolha de *producers* para os minérios encontrados pelos *spotters* 1 e 2. Neste caso, ficou o *producer* 2 afeto ao minério do *spotter* 1 e o *producer* 1 afeto ao minério do *spotter* 2.

2.2.3 Afetação de *transporters*

Após a extração do minério é necessário transportá-lo para a nave-mãe. O *producer* que acabou de extrair o minério tem que selecionar um *transporter*, do mesmo modo que o *spotter* seleciona um *producer*. Cada *transporter* comunica o valor do esforço e o minério que consegue transportar possibilitando o *producer* de escalonar os diferentes agentes.

A Figura 2.4 exemplifica o processo de escolha de *transporters* para os minérios extraídos por dois *producers*. Consideramos ainda as quantidades de minério extraídas, a capacidade e carga corrente dos *transporters*. Estas restrições irão influenciar a forma como se negoceia a afetação dos *transporters* numa fase mais avançada da implementação.

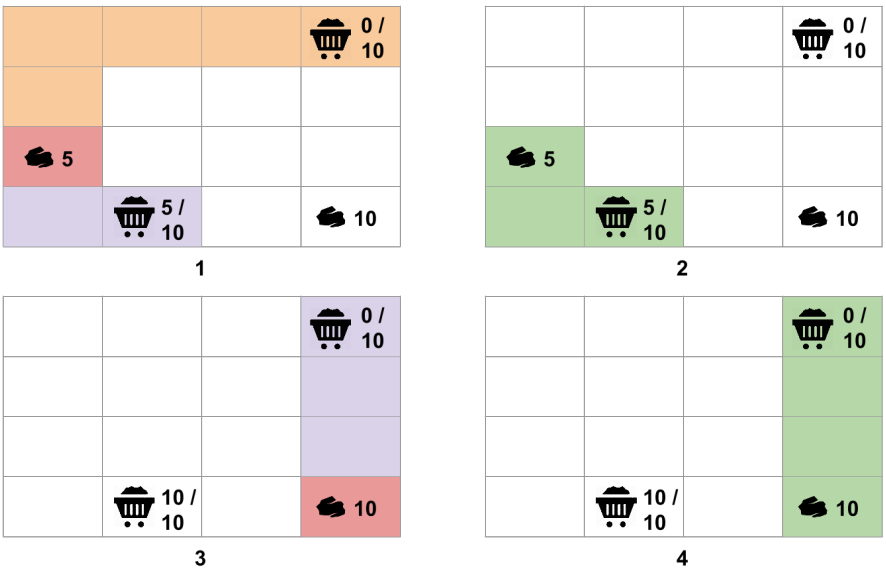

Figura 2.4: Exemplo de afetação de *transporters*

Capítulo 3

Desenvolvimento

capítulo novo mas as secções não são completamente novas

3.1 Plataformas utilizadas

brevíssima descrição o que é - é capaz de ser melhor reduzir isto

3.1.1 JADE

O JADE, *Java Agent DEvelopment Framework*, é um software que permite desenvolver sistemas baseados em agentes. Os agentes são distribuídos por *containers* que podem estar em máquinas diferentes, cada um utiliza uma *thread*.

O JADE é uma ferramenta totalmente escrita em JAVA, que suporta troca de mensagens ACL, seguindo a especificação FIPA. Além disso, possui um sistema de gestão de agentes e um sistema de páginas amarelas. Como os agentes estão distribuídos por contentores que podem estar em máquinas diferentes, permite ter agentes remotos. Os agentes podem migrar entre contentores e ser clonados. Esta plataforma possui ainda uma série de ferramentas que simplificam a administração e o desenvolvimento de aplicações, tais como:

- agente de monitorização remota - interface gráfico para monitorizar a atividade dos agentes;
- agente pateta - permite trocar mensagens com outros agentes;
- agente inspetor - permite inspecionar outros agentes;
- agente introspetivo - permite monitorizar o ciclo de vida de um agente.

A implementação de agentes em JADE é feita recorrendo a comportamentos, que determinam as tarefas a executar pelos agentes consoante o contexto.

3.1.2 Repast 3

O Repast 3, *Recursive Porous Agent Simulation Toolkit*, é uma ferramenta de simulação baseada em agentes, que permite construir simulações locais à máquina com diversos agentes. O processamento de cada agente é distribuído por *threads*.

O Repast 3 suporta simulação de espaços físicos, representação 2D e 3D e análise em tempo real. Possui uma barra de ferramentas para controlar as simulações, um interface gráfico para manipular os parâmetros. Permite recolher dados em vários formatos, incluindo vários tipos de gráficos. A interação entre os agentes pode ser visualizada graficamente. Os espaços físicos simulados podem ser de variados tipos, como por exemplo, grelhas hexagonais ou retangulares, espaços contínuos ou redes. Além de poderem ser corridas com recurso ao interface gráfico, as simulações podem também ser lançadas em *batch*.

O simulador discreto considera unidades de tempo - *ticks* ou passos. Os eventos são planeados para ocorrer em *ticks* específicos e assim é respeitada a ordem dos acontecimentos. Uma simulação considera um conjunto de agentes, cujos comportamentos são controlados recorrendo ao plano.

3.1.3 SAJaS

O SAJaS, *Simple API for JADE-based Simulations*, é uma ferramenta que se propõe servir de ponte entre o desenvolvimento e a simulação de Sistemas Multi-Agente. Desta forma, através do SAJaS, podemos tomar vantagem dos benefícios do JADE e do Repast 3.

O SAJaS permite construir um Sistema Multi-Agente tal como é feito em JADE, existindo até uma ferramenta (MASSim2Dev) que traduz código JADE para SAJaS (e vice-versa), alterando as classes importadas. Ao contrário do JADE, permite ainda, recorrendo para isso ao Repast 3, a simulação deste tipo de sistemas. Além disso, o SAJaS permite obter melhores performances do que numa simulação construída em JADE, que teria que considerar algum tipo de agente representado o 'mundo'.

3.1.4 Realce das funcionalidades relevantes para o trabalho

Com o suporte do JADE são feitos os protocolos de comunicação entre os diferentes agentes, utilizando mensagens ACL. Usando o Repast 3 torna-se fácil simular um espaço físico, popular o espaço com os agentes construídos em JADE, desenhá-los e finalmente vê-los em ação, obtendo gráficos relativos à sua performance. O SAJaS permite-nos recorrer a estas duas ferramentas simultaneamente, construindo e simulando o nosso sistema de forma eficiente.

3.1.5 Ambiente de desenvolvimento

Os elementos do grupo utilizam Sistemas Operativos (SO) e Ambientes de Desenvolvimento Integrado (IDE) diferentes, de acordo com as suas preferências. Para tal, é especialmente útil as plataformas utilizadas serem baseadas em Java. Assim como,

o facto de apenas ser necessário adicionar os respetivos módulos `jar` ao projeto, para que qualquer IDE compile e execute a aplicação devidamente. Desta forma, a combinação SO - IDE utilizada por cada elemento, foi:

- Ângela Cardoso - macOS Sierra - IntelliJ IDEA;
- Diogo Ferreira - Linux- NetBeans;
- Marina Camilo - Windows- Eclipse;

não sei exatamente qual

não sei exatamente qual

3.2 Estrutura da aplicação

módulos, diagrama de classes...

3.3 Detalhes relevantes da implementação

Capítulo 4

Experiências

capítulo novo, mas falamos de experiências no relatório intercalar

esta foi a parte que mais falou de experiências no relatório intercalar

Para podermos ver resultados mais rapidamente, inicialmente serão implementadas apenas as funcionalidades mais básicas de cada tipo de agente. Como tal, a primeira fase de avaliação preocupar-se-á essencialmente em garantir que cada agente faz aquilo que é suposto.

Numa segunda fase, introduziremos as restrições pretendidas para a simulação, nomeadamente a capacidade limitada dos *transporters* e o facto de todos os agentes poderem detetar minério nas suas deambulações. Uma vez realizadas estas alterações aos agentes básicos, a avaliação geral levará em conta o tempo que a simulação demora a correr e o número de passos efetuados por todos os agentes. Usando estas métricas, dados vários tamanhos para o espaço inicial a explorar, iremos determinar a complexidade do nosso sistema. Para podermos melhorar cada componente individual, calcularemos também métricas mais finas, como o tempo de espera médio e máximo, quando é invocado um *producer* ou um *transporter*, ou o tempo de exploração do terreno pelos *spotters* consoante o seu número.

Com as métricas obtidas após a segunda fase de implementação, analisaremos diferentes algoritmos de forma a tornar mais eficiente esta demanda. Nomeadamente, avaliaremos qual a alocação mais eficiente de agentes, se o mapa fica corretamente dividido entre os *spotters* e se o tempo de simulação foi o mínimo para o caso em questão.

4.1 Objetivos de cada experiência

4.2 Resultados

Capítulo 5

Conclusões

5.1 Da análise dos resultados das experiências levadas a cabo

5.2 Do desenvolvimento do trabalho e aplicabilidade de SMA ao cenário proposto

Capítulo 6

Melhoramentos

sugestões para melhoramentos a introduzir no programa

Capítulo 7

Recursos

7.1 Bibliografia

- Enunciado do trabalho
- Slides das aulas teóricas
- Slides sobre `JADE`
- Slides sobre `Repast 3`
- Página do `SAJaS`
- Página da ferramenta `MASSim2Dev`

7.2 Software

- `JADE`
- `Repast 3`
- `SAJaS`

7.3 Elementos do grupo

indicar percentagem aproximada de trabalho efetivo de cada elementos do grupo

- Ângela Cardoso - %
- Diogo Ferreira - %
- Marina Camilo - %

Capítulo 8

Apêndice

manual do utilizador (sucinto)