

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

1	Objetivo	3
1.1	Descrição do cenário	3
1.2	Objetivos do trabalho	3
2	Especificação	5
2.1	Identificação e caracterização dos agentes	5
2.1.1	<i>Spotter</i>	5
2.1.2	<i>Producer</i>	6
2.1.3	<i>Transporter</i>	6
2.2	Protocolos de interação	6
2.2.1	Divisão de espaços	7
2.2.2	Afetação de <i>producers</i>	8
2.2.3	Afetação de <i>transporters</i>	8
3	Desenvolvimento	10
3.1	Plataformas utilizadas	10
3.1.1	JADE	10
3.1.2	Repast 3	11
3.1.3	SAJaS	11
3.1.4	Realce das funcionalidades relevantes para o trabalho	11
3.1.5	Ambiente de desenvolvimento	11
3.2	Estrutura da aplicação	12
3.3	Detalhes relevantes da implementação	12
4	Experiências	13
4.1	Objetivos de cada experiência	13
4.1.1	Complexidade da implementação	13
4.1.2	Porcentagem de utilização dos agentes	14
4.1.3	Comparação de diferentes técnicas	14
4.2	Resultados	15
4.2.1	Complexidade da implementação	15
4.2.2	Porcentagem de utilização dos agentes	15

5	Conclusões	16
5.1	Da análise dos resultados das experiências levadas a cabo	16
5.2	Do desenvolvimento do trabalho e aplicabilidade de SMA ao cenário proposto	16
6	Melhoramentos	17
7	Recursos	18
7.1	Bibliografia	18
7.2	Software	18
7.3	Elementos do grupo	18
8	Apêndice	19

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

Existem 3 fases que se destacam no decorrer da simulação: repartição do espaço entre os *spotters* existentes, requerimento de um *producer* no momento de descoberta de um minério e, finalmente, o escalonamento dos *transporters* para a recolha dos fragmentos resultantes de um minério.

incluir diagramas de interação UML, se aplicável

2.2.1 Divisão de espaços

No início da simulação a nave-mãe divide o espaço disponível em linhas de acordo com o número de *spotter*s existentes e entrega cada um dos dados dos sub-espacos a um *spotter* distinto de modo a que este o possa depois confirmar com os restantes *spotter*s que pode realmente ficar responsável pela área. A Figura 2.1 demonstra como a nave-mãe faz esta divisão.

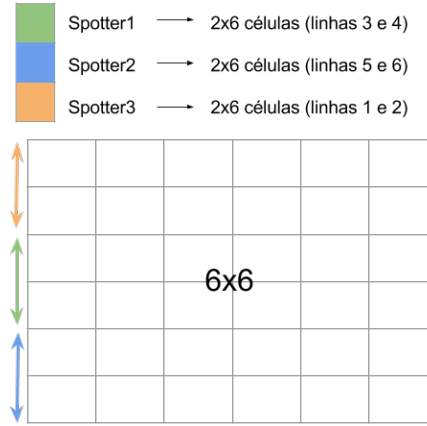


Figura 2.1: Repartição de espaços

Uma vez recebida a sugestão pela nave-mãe o *spotter* inicia então o protocolo de negociação **fipa-propose** (Figura 2.2) com os restantes *spotter*s. Este protocolo define como propôr algo a outros agentes e tratar das suas respostas.

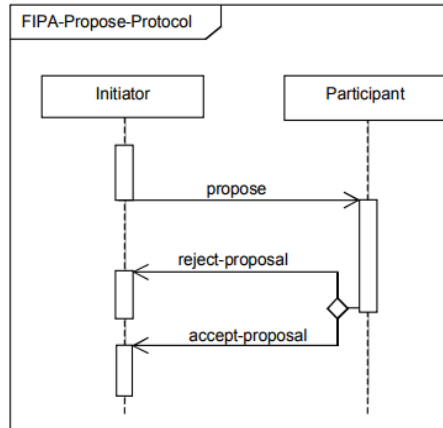


Figura 2.2: Repartição de espaços

Após a receção de todas as respostas é enviada uma última mensagem de volta a informar que a área ficou então afeta ao *spotter* que iniciou a proposta. Esta última fase é complementar ao protocolo. O protocolo **fipa-propose** é suportado pelas interfaces *ProposeInitiator* e *ProposeResponder* que um *spotter* implementa através

de *behaviours* e pelas classes internas *RequestAreaBehaviour* e *AnswerCallBehaviour*, respetivamente. Finalmente, a receção da última mensagem de notificação é feita também através de um *behaviour* na classe interna *AcknowledgeAreaBehaviour*. A mensagem tem a performativa **inform** e no seu conteúdo uma string no formato **offset-height** em que *offset* indica a coordenada no eixo Y em que a área começa e *height* a altura da mesma. Os receptores desta mensagem guardam então internamente num mapa esta afetação para possíveis futuras propostas.

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.

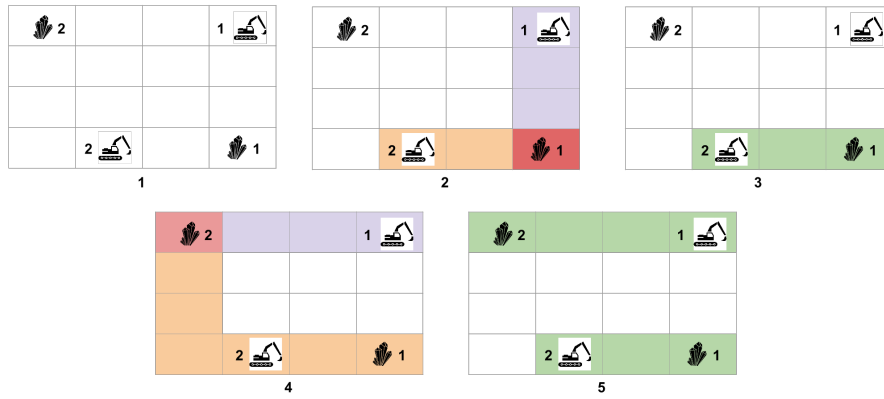


Figura 2.3: Exemplo de afetação de *producers*

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 influ-

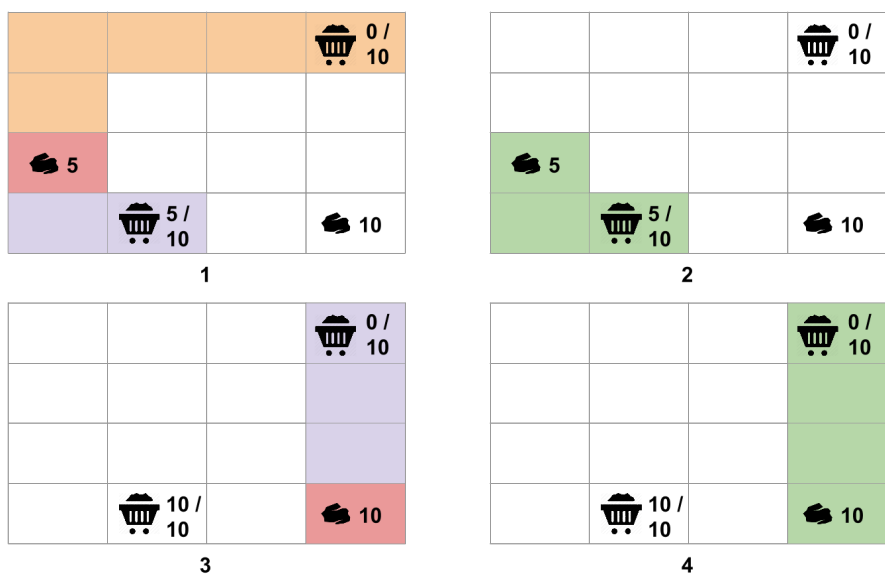


Figura 2.4: Exemplo de afetação de *transporters*

enciar a forma como se negocia a afetação dos *transporters* numa fase mais avançada da implementação.

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
exata-
mente
qual

não sei
exata-
mente
qual

3.2 Estrutura da aplicação

módulos, diagrama de classes...

3.3 Detalhes relevantes da implementação

Capítulo 4

Experiências

Para examinar a qualidade da nossa implementação, desenhamos algumas experiências, que são apresentadas em seguida. Tivemos alguma dificuldade em realizar estas experiências, porque não fomos capazes de as automatizar completamente, devido forma como correm as simulações do **Repast 3**.

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

4.1.1 Complexidade da implementação

Estas primeiras experiências têm o objetivo de determinar a qualidade geral da implementação, nomeadamente investigando a sua complexidade temporal. Uma vez

que há vários parâmetros no nosso modelo, para determinar a sua complexidade, é necessário fixa-los e variar apenas um deles de cada vez. Os parâmetros que consideramos significativos são:

- *size*- tamanho do lado do quadrado que representa a área a explorar;
- *minerals*- número de minérios que serão gerados na área;
- *spotters*- número de *spotters* que serão disponibilizados;
- *producers*- número de *producers* que serão disponibilizados;
- *transporters*- número de *transporters* que serão disponibilizados.

Outros parâmetros, tal como a probabilidade de um dado minério ser extraível, também são relevantes. No entanto, não nos parecem tão importantes como os mencionados acima. Em relação àqueles que destacamos, para que as experiências não se prolongassem demasiado, no que toca a complexidade, estudamos apenas *size* e *minerals*, determinando o número de *ticks* que o modelo demora a correr, conforme varia um, e apenas um, destes parâmetros.

4.1.2 Percentagem de utilização dos agentes

Se, por um lado, parece claro que seja útil ter mais agentes, especialmente quando *size* e *minerals* são elevados. Por outro lado, num contexto de simulação de acordo com a realidade, é óbvio que ter mais agentes é mais dispendioso. Sendo assim, consideramos relevante determinar, para *size* e *minerals* fixos, qual o número ideal de cada tipo de agente. Para isso, fizemos variar *spotters*, *producers* e *transporters* cada um por sua vez, avaliando o número total de passos dados por cada agente, assim como o número de *ticks* em que cada um deles está inativo.

4.1.3 Comparação de diferentes técnicas

Como não chegamos a implementar diferentes técnicas para resolver o problema, não realizamos experiências que permitissem compará-las. No entanto, decidimos ainda assim expor aqui a forma como o faríamos.

A primeira forma de comparação, seria comparar o número de *ticks* que cada técnica usa fixados os vários parâmetros. A experiência mais rápida, seria considerada melhor nestas experiências.

Outra forma de comparação, seria considerar a quantidade total de passos que os vários agentes dão para completar a exploração em cada técnica. Neste caso, seria melhor a técnica que utilizasse menos passos.

4.2 Resultados

4.2.1 Complexidade da implementação

Começamos por variar *size* mantendo fixos os restantes parâmetros, nomeadamente, utilizamos 1 minério e 5 agentes de cada tipo. Com o aumento do tamanho do quadrado a explorar, os *spotters* são os últimos agentes a terminar a sua tarefa. Isto porque têm que percorrer toda a área, enquanto que os restantes agentes apenas têm que se deslocar até ao minério, assim que este for encontrado.

4.2.2 Percentagem de utilização dos agentes

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)