Gestão de Elevadores

Relatório Intercalar

Mestrado Integrado em
Engenharia Informática e Computação

Agentes e Inteligência Artificial Distribuída

Grupo T01_4:
António Ramadas - up201303568@fe.up.pt
Duarte Pinto - up201304777@fe.up.pt
Gustavo Silva - up201304143@fe.up.pt

Faculdade de Engenharia da Universidade do Porto

19 de Novembro de 2016

Conteúdo

1 Enunciado

1.1 Descrição do cenário

Pretende-se implementar um sistema para a gestão eficiente de elevadores num edifício, em que cada um é representado por um agente num sistema multi-agente.

Os agentes (elevadores) comunicam entre si informação relevante. Quando é necessário tomar uma decisão, o elevador pode atender ou não uma chamada dependendo do seu estado (andar onde se encontra, direção que está a seguir e peso transportado) e de informação sobre o estado de outros elevadores. Desta forma, os agentes podem combinar ações conjuntas para satisfazer mais rapidamente os pedidos dos utentes.

O programa deve permitir a configuração dos seguintes parâmetros por parte do utilizador:

- número de pisos do edifício;
- número de elevadores;
- carga máxima de cada elevador.

Cada elevador demora um certo tempo a ir de um piso a outro, assim como em paragens para entrada e/ou saída de utentes.

Devem ser considerados métodos diferentes de chamada do elevador:

- dois botões de chamada (subir/descer);
- teclado para indicação do piso destino.

1.2 Objetivos do trabalho

O objetivo deste trabalho consiste em comparar o desempenho deste sistema multi-agente usando diferentes estratégias de cooperação com um sistema tradicional onde cada elevador possui uma estratégia fixa e individual: atende o pedido o elevador que se encontra mais próximo do piso onde a chamada foi efetuada.

1.3 Resultados esperados e forma de avaliação

Considerar-se-á como índice de desempenho o tempo médio que os utilizadores do sistema perdem desde que carregam no botão para chamar um elevador até que este chega ao piso de destino. Desta forma, o objetivo na implementação dos diferentes comportamentos e estratégias será a minimização deste índice de desempenho.

Espera-se que o uso de um sistema de chamada constituído por um teclado para indicação do piso destino seja mais eficiente do que a utilização de apenas dois botões de chamada (subir/descer).

Para se efetuar a comparação entre as diferentes estratégias, ter-se-á estatísticas relativamente ao tempo de espera máximo e mínimo, ao desvião padrão entre tempos de espera, à taxa de ocupação dos elevadores, entre outros.

2 Plataforma/Ferramenta

Para este trabalho é sugerido o uso de uma das seguintes hipóteses: JADE, Respast+SAJaS ou Jadex. Após alguma pesquisa sobre as diversas ferramentas que permitissem simular o melhor possível um sistema multi-agente, a nossa escolha recaiu sobre a conjugação entre Repast e SAJaS.

Na verdade, o Repast permite a modulação baseada em agentes, isto é, visa criar, analisar e experimentar mundos artificiais populados por agentes. Logo, os elevadores serão agentes e o mundo um edifício. Adicionalmente, um importante fator para o uso desta ferramenta é a existência de uma unidade de tempo (*tick*) que permite atribuir eventos a determinados períodos. Assim, esta medida temporal permite medir o desempenho ao longo do tempo como o tempo de espera desde a chamada de um elevador até à chegada no respetivo tempo. Relativamente ao SAJaS, esta ferramenta é uma API para simulações baseadas em Jade que permite ocultar a inexistência de uma ponte entre simulação (Repast) e desenvolvimento sistemas multi-agente (Jade).

Apesar do Jade ser uma ferramenta útil para desenvolvimento de sistemas multi-agente através de uma plataforma que permite a gestão de agentes e o uso de contentores, nós optamos por utilizar

o SAJaS como API para desenvolvimento de agentes que depois são utilizados em simulações. Já o Jadex foi descartado por estar mais direcionado a agentes BDI (*Beliefs, Goals and Intentions*) que não se adequam ao propósito do tema.

3 Especificação

3.1 Identificação e caracterização dos agentes

3.1.1 Arquitetura

Serão considerados dois tipos de agentes: elevador e edifício. Existe apenas um agente do tipo edifício, que é o responsável por simular as chamadas dos utilizadores do sistema e alocar esses pedidos a elevadores.

Cada elevador tem conhecimento de:

- Lista de paragens que vai fazer;
- Posição;
- Estado (subir, descer, entrada/saída de pessoas ou em espera);
- Capacidade atual.

3.1.2 Comportamento e estratégias

Uma das estratégias comportamentais que pretendemos implementar é um algoritmo tradicional e bastante básico segundo o qual um pedido é satisfeito pelo elevador que se encontra mais próximo do sítio onde foi chamado. Nesta situação, a chamada é feita carregando num dos botões de um dispositivo com um botão de subida e um de descida. De resto, o algoritmo é semelhante ao utilizado para determinar o movimento da cabeça de um disco rígido ao servir pedidos de leitura e escrita no mesmo (*LOOK disk scheduling algorithm*) [1]. A ideia é que o elevador continue a deslocar-se no mesmo sentido até que esteja vazio, parando apenas para a entrada e saída de utilizadores que pretendam seguir na mesma direção.

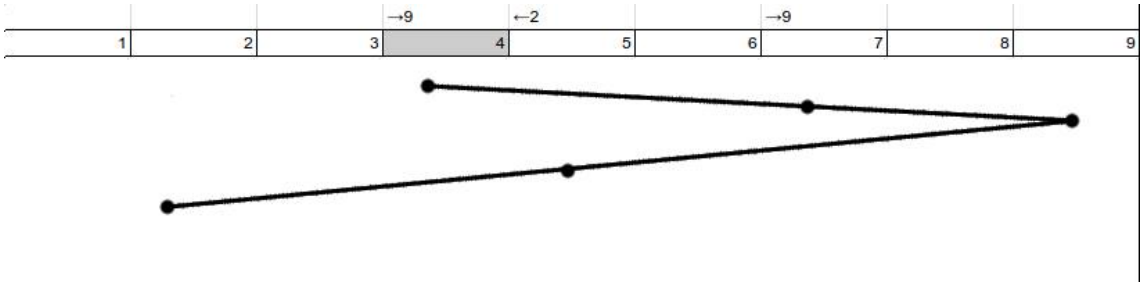

Figura 1: Algoritmo baseado no *LOOK disk scheduling algorithm* [1] - O elevador começa no piso 4 e vai para o 9. Assim que sai, o elevador é chamado no piso 5 e 9. Como o cliente no piso 5 pretende ir no sentido contrário aquele que o elevador está a fazer no momento, o elevador não pára no piso 5. Como o cliente no piso 7 pretende ir no mesmo sentido que o elevador e até para o mesmo destino, o elevador pára no piso 7. Quando chega ao piso 9 e deixa os dois clientes, o elevador desce para apanhar o cliente no piso 5 uma vez que o seu pedido ainda não foi atendido, e leva o cliente do piso 5 ao seu destino.

Ainda outra estratégia comportamental que pretendemos implementar é a de um algortimo também bastante simples que consiste em atribuir o pedido ao elevador que se encontra mais perto independentemente do seu sentido atual (subir ou descer). Como podemos facilmente concluir um pedido pode demorar bastante tempo a ser atendido, pois num grande edifício um pedido do piso mais baixo até ao topo pode demorar bastante tempo a ser concluído caso hajam vários pedidos no meio. Contudo, em certos casos esta solução pode revelar-se mais económica e rápida que o algoritmo acima. Veja-se o exemplo: um elevador está no piso 1 e está a seguir até ao topo (3º andar), mas surge um pedido no piso 0 para subir. No algoritmo acima, o elevador percorreria os

seguintes pisos: 1->2->3->2->1->0->1->2 (total de 8). Neste algoritmo, o elevador percorreria: 1->0->1->2->3 (total de 5). Assim, neste caso haveria uma redução da travessia em 37.5%.

A última estratégia comportamental a que nos propomos é a mais complexa seria a que faria uso da indicação do piso aquando da chamada do elevador. Este algoritmo visa agrupar no mesmo elevador os pedidos cujo destino sejam próximos uns dos outros. Em casos específicos este algoritmo revela-se bastante expedito e com um desempenho superior em qualquer um dos outros edifícios. De facto, esta estratégia já se encontra comprovada e a funcionar no Hotel New York Marriott Marquis [4] e um simples exemplo pode ser consultado no folheto promocional [3, p.3].

Por fim, apesar da diferença dos algoritmos, quando um elevador não pode atender um pedido que lhe foi atribuído e teve de passar pelo respetivo piso sem parar, a ação desencadeada será a de o elevador encaminhar o pedido para o *building* e depois este inicia um protocolo para decidir qual o novo elevador ao qual esse pedido irá ser alocado.

3.2 Protocolos de interação

O protocolo de cooperação rede contratual (Fig. 2) será utilizado sempre que houver um novo pedido. O objetivo da utilização deste protocolo é decidir qual o elevador que vai ficar responsável por atender esse mesmo pedido.

Assim que um pedido é recebido, é enviado um *call for proposals* do *building* para todos os elevadores. De seguida, os elevadores respondem com um *propose* no qual incluem o tempo que levarão até à satisfação do pedido, caso este lhes seja alocado. Este tempo tem em conta a lista de tarefas que cada elevador tem na fila de execução em cada momento. Assim que o *building* (*initiator*) receber as respostas de todos os elevadores, escolhe aquele que for capaz de satisfazer o pedido mais rapidamente, enviando-lhe uma mensagem *accept-proposal* e enviando a mensagem *reject-proposal* a todos os outros elevadores. O elevador que ficou responsável por executar a tarefa envia uma mensagem *inform-result* quando a terminar.

Em certas situações, quando um elevador fica cheio e incapaz de satisfazer novos pedidos até que as pessoas saiam do mesmo, é necessário voltar a correr o protocolo de cooperação rede contratual para determinar novamente qual o elevador que se encontra em melhores condições para satisfazer os pedidos que foram ignorados por ter sido atingida a capacidade máxima. Nesse sentido, o elevador que atingiu a capacidade máxima inicia um protocolo de pedido de interação (Fig. 3) com o *building*, enviando-lhe uma mensagem do tipo *request* pedindo-lhe para voltar a alocar o pedido que foi ignorado a um novo elevador. Note-se que o elevador agora alocado pode ser o mesmo, embora este vá demorar mais tempo a desempenhar a tarefa, pois terá que deixar sair pessoas antes de satisfazer o pedido.

3.3 Faseamento do projeto

No sentido de desenvolver o projeto de um modo incremental, pretende-se dividir o trabalho a desenvolver nas seguintes fases:

1. Criação do mundo e geração automática de pedidos;
2. Implementação do algoritmo tradicional com um só agente;
3. Expansão do algoritmo tradicional para múltiplos agentes, com comunicação entre si;
4. Implementação do algoritmo com indicação prévia do destino;
5. Finalização da GUI, permitindo configurar o número de pisos do edifício, o número de elevadores e a carga máxima de cada elevador;
6. Geração de estatísticas;
7. Experiências e análise de resultados.

4 Recursos

4.1 Referências

[1] http://courses.teresco.org/cs432_f02/lectures/17-files/17-files.html - "Lecture 17 - Disk Scheduling"

[2] Y. Gu, "Multi-objetive Optimization of Multi-Agent Elevador Group Control System Based on Real-time Particle Swarm Optimization Algorithm,"*Engineering*, Vol. 4 No. 7, 2012, pp. 368-378. doi: 10.4236/eng.201247048

[3] https://goo.gl/CVlODq - Shindler Miconic 10 brochure

[4] https://youtu.be/T6gzm_ifzg8 - "The Smartest Elevators with destination dispatch"

4.2 Software

Repast + SAJaS

Figura 2: SC00029 - FIPA Contract Net Interaction Protocol

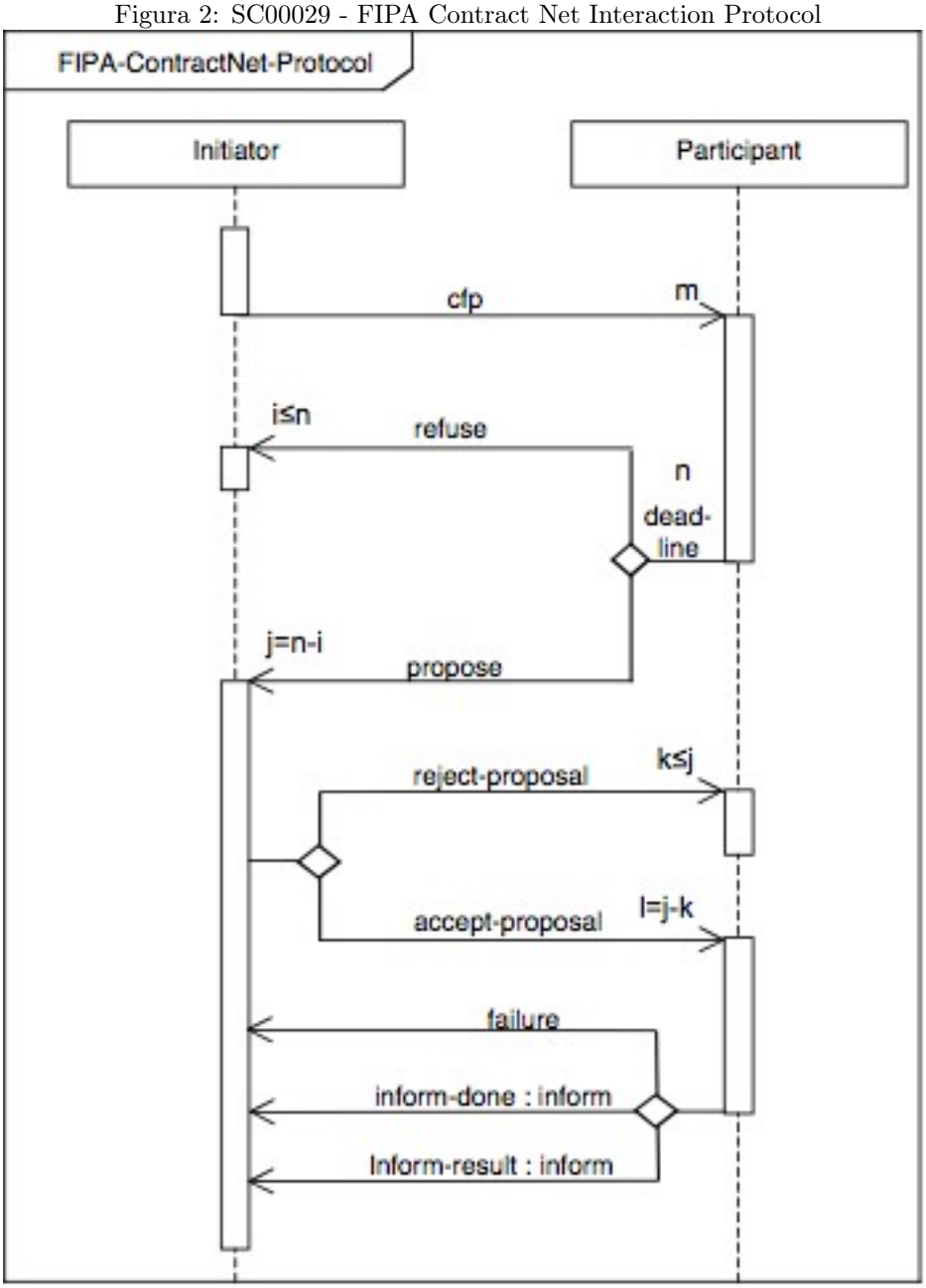

Figura 3: SC00026 - FIPA Request Interaction Protocol