Gestão de Elevadores

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Agentes e Inteligência Artificial Distribuída
4º Ano 1º Semestre

Grupo T01_3:
Luís Barbosa - 201405729 - up201405729@fe.up.pt
Paulo Santos - 201403745 - up201403745@fe.up.pt
Sérgio Almeida - 201403074 - up201403074@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

10 de dezembro de 2017

Conteúdo

1 Objetivo

1.1 Descrição do cenário

O cenário do presente trabalho consiste num edifício de vários andares com diversos elevadores. Cada elevador possui uma carga máxima, que corresponde ao número máximo de pessoas que pode transportar.

Os elevadores comunicam entre si informação relevante. Quando há um novo pedido numa determinada zona do edifício, este deve ser alocado a um dos elevadores existentes naquela zona. Os elevadores possuem uma lista de pisos em que têm que parar, podendo a mesma ser alterada dinamicamente para se incluirem novos pedidos.

Periodicamente, os elevadores podem partilhar informação sobre os seus estados, no sentido de poderem atribuir tarefas a outros elevadores.

O programa deve permitir que o utilizador configure o sistema, como o número de pisos do edifício, número de elevadores e a carga máxima de cada elevador.

As necessidades dos utentes, como por exemplo a frequência de chamada de elevador em cada piso do edifício (devendo ser superior para o piso 0) e piso de destino, são geradas aleatoriamente.

Um elevador demora um tempo pré-definido a ir de um piso a outro, sendo que o intervalo de tempo correspondente à paragem do elevador para entrada/saída de utentes também é contabilizado.

1.2 Objetivos do trabalho

O trabalho tem como principal objetivo o desenvolvimento de um sistema multi-agente para a gestão eficiente dos elevadores de um determinado edifício, tendo em conta o cenário descrito acima. Para tal, será necessária a implementação de um algoritmo que permita gerir as chamadas aos elevadores de forma a que o transporte seja feito o mais rápido possível.

2 Especificação

2.1 Identificação e caracterização dos agentes

No nosso projeto, distinguem-se dois tipos de agentes: *Elevator* e *Building*.

2.1.1 *Building*

O *Building* é instanciado apenas uma vez, assim que o programa é iniciado. Regista-se no *DFService*, criando um *DFAgentDescription*, adicionando a este um novo serviço com tipo "Building". Este agente é responsável por gerar elevadores e simular os pedidos existentes num edifício por parte dos utilizadores ao alocar aos elevadores os pedidos. Tem a seguinte **arquitetura**:

- **properties** - informações sobre o número de andares e elevadores, se existe teclado para efetuar pedido e se a negociação entre elevadores é permitida
- **elevatorsProperties** - informações sobre o peso máximo, tempo de movimento e tempo de entrada/saída de pessoas
- **elevators** - lista com o AID de todos os elevadores
- **reqGenInterval** - valor do intervalo de geração de pedidos pelo edifício
- **randNumRequestsPerInterval** - valor aleatório do número de pedidos gerados por intervalo, caso o valor lido do ficheiro seja "rand"
- **numRequestsPerInterval** - valor lido do número de pedidos gerados por intervalo

Todas estas informações são lidas do ficheiro de configuração **default.properties**, de forma a tornar o processo de configuração do cenário mais generalizado.

Este agente tem o seguinte **comportamento**:

- Cria o número de pedidos definidos, definindo inicialmente o andar inicial, sendo que aqui se assumiu que o andar final era o mesmo que o inicial.
- Cada pedido é atribuído a um elevador aleatório, verificando se esse elevador possui teclado. Caso possua, o andar de destino é atualizado uma vez que já era conhecido. Caso contrário, mantém-se a assunção inicial.
- De seguida, cria-se uma *ACLMessage* do tipo *CFP* com o seguinte conteúdo: **Andar Inicial, Andar Destino, Tempo para o Andar Inicial**. Esta é enviada pelo *Building* para o **Elevator** escolhido anteriormente, utilizando o protocolo de interação *FIPA Contract Net*.

2.1.2 *Elevator*

O *Elevator* regista-se no serviço do *Directory Facilitator* com o tipo "Elevator", de forma a que o *Building* (e outros que subscrevam o *Directory Facilitator*) saiba que este existe. Tem a seguinte **arquitetura**:

- **properties** - informações sobre o peso máximo, tempo de movimento, tempo de entrada/saída de pessoas, ter/não ter teclado
- **state** - informações sobre o piso atual, peso atual, estado de movimento ("STOPPED", "GOING_UP"ou "GOING_DOWN") e nº de pessoas

- **internalRequests** - lista interna de pedidos a realizar, que é dinâmica, uma vez que pode ser alterada caso outros elevadores lhe atribuam um novo pedido, ou caso seja feito um novo pedido no *Building*
- **statistics** - estatísticas utilizadas na interface
- **startupTime** - tempo de inicialização
- **numElevators** - o elevador sabe qual elevadores existem no total

Este agente tem o seguinte **comportamento**:

Caso a negociação entre elevadores seja permitida, tenta negociar com outros elevadores os pedidos internos, através do envio de uma *ACLMessage* do tipo *CFP*, utilizando o protocolo de interação *FIPA Contract Net*. Os outros elevadores são adicionados como recetores da mensagem. Para cada pedido que ainda não tenha sido atendido, é procurado o pedido que vai ser mais demorado a executar, sendo esse o *initiator* do *Contract Net Behaviour*.

É feita a verificação da existência de algum pedido para o piso atual do elevador, e se ainda não foi atendido. Caso isto se verifique, o elevador para e é gerado um novo peso que não exceda a capacidade do mesmo. Este peso pretende simular a entrada de uma ou mais pessoas com ou sem pesos. Existem casos em que é gerado o valor 0 de forma a aproximar a uma situação real, pois por vezes não é possível satisfazer o pedido pois o elevador já se encontra lotado. Durante o tempo de entrada de pessoa, é assegurado que não se atendem outros pedidos.

A forma de o elevador decidir para que andar se deve dirigir tem por base o andar mais próximo do atual. É verificado todo o conjunto de pedidos do elevador, isto é, caso já tenha sido um pedido atendido, verifica-se qual é o andar de destino mais próximo. Caso seja um pedido externi, verifica-se qual é o andar de partida mais próximo.

O algoritmo de decisão tenta, em todos os andares, otimizar a distância ao próximo andar. Desta forma, garantimos que não ande um grande número de andares sem parar quando era mais eficiente ter parado a meio para atender mais pedidos.

Verifica, também, para todos os pedidos atendidos do elevador, se estes já se encontram no andar de destino. Quanto isto acontece, o elevador para e é simulada a saída da pessoa que tinha feito o pedido.

2.2 Estratégias e processos de raciocínio

De forma a comparar o desempenho dos elevadores com diferentes heurísticas, foram implementadas quatro estratégias de alocação de pedidos. Estas quatro estratégias suportam um dos seguintes modelos: o primeiro, em que existe apenas um botão de chamada, ou um segundo que assume a existência de um teclado onde é possível indicar o piso de destino.

A **primeira** estratégia assume o sistema tradicional onde cada elevador possui uma estratégia fixa e individual e segue o primeiro modelo: atende o pedido o elevador que se encontra mais próximo do piso onde a chamada foi efetuada. Para isto ser possível, os elevadores comunicam entre si através do protocolo de interação que será descrito mais à frente.

Para tornar o envio de requests para outros elevadores melhor, seria necessário pedir-lhes feedback para todos os requests e isso sobrecarregaria a rede de negociação, só se pedindo ajuda para o pior pedido de cada vez. Se não

podem ajudar para o pior, provavelmente não podem ajudar para outros. E iterativamente, ao alocar os piores pedidos a outros, optimiza-se/reduz-se os tempos de espera.

A **segunda** estratégia segue o segundo modelo, onde o elevador já possui a informação de qual é o piso de destino, conseguindo calcular o tempo de espera exato, permitindo uma alocação mais correta dos pedidos.

As duas estratégias anteriores utilizam o protocolo de interação descrito abaixo para que os elevadores negoceiem entre si. Para termos melhores termos de comparação, achamos correto criar outras duas estratégias em que não existe negociação.

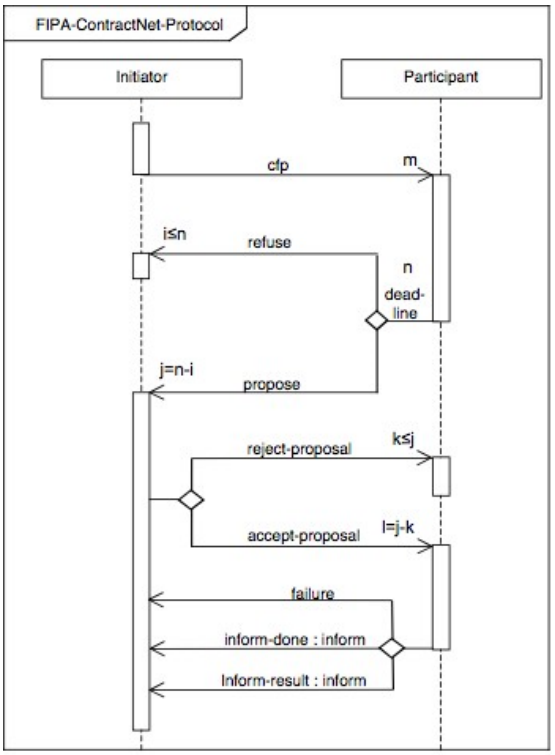
Desta forma, a **terceira** estratégia é idêntica à primeira e a **quarta** idêntica à segunda, exceto na parte em que negoceiam. Assim, ambas estimam o tempo que o elevador demoraria a responder a um pedido caso fosse alocado a ele.

2.3 Protocolos de interação

O protocolo de interação utilizado na decisão para alocação de pedidos aos elevadores é o **FIPA CONTRACT NET**.

A cada período de geração de pedidos do *Building* (reqGenInterval), para cada um destes é gerada uma *ACLMessage* do tipo *CFP*, com o protocolo **FIPA_CONTRACT_NET**, em que os os seus recetores são todos os elevadores do edifício. Em seguida, cada elevador após receber o **CFP**, calcula o tempo que vai demorar até ao andar inicial do pedido. Se este for menor do que o tempo que vem no pedido, o elevador envia uma *ACLMessage* do tipo **PROPOSE**. Caso contrário, o agente envia uma *ACLMessage* do tipo **REFUSE**. Quando o edifício recebe as propostas (*ACLMessage* do tipo **PROPOSE**), analisa a melhor com base nos tempos e envia uma *ACLMessage* do tipo **REJECT_PROPOSAL** para todas, exceto para a melhor proposta, em que envia uma *ACLMessage* do tipo **ACCEPT_PROPOSAL**, ficando assim alocado o pedido a este elevador.

Também um elevador pode negociar o seu pior pedido com os restantes elevadores: assim, cria uma mensagem do tipo **CFP** com a informação do seu pior pedido e envia para os restantes elevadores. A gestão da receção deste pedido é feita de forma igual à da receção de pedidos do *Building*. Caso não consiga atribuir a outro elevador, o pedido mantém-se alocado ao próprio.


FIPA-ContractNet-Protocol
Initiator
Participant
cfp
m
i≤n
refuse
n
dead-
line
j=n-i
propose
reject-proposal
k≤j
accept-proposal
l=j-k
failure
inform-done : inform
Inform-result : inform

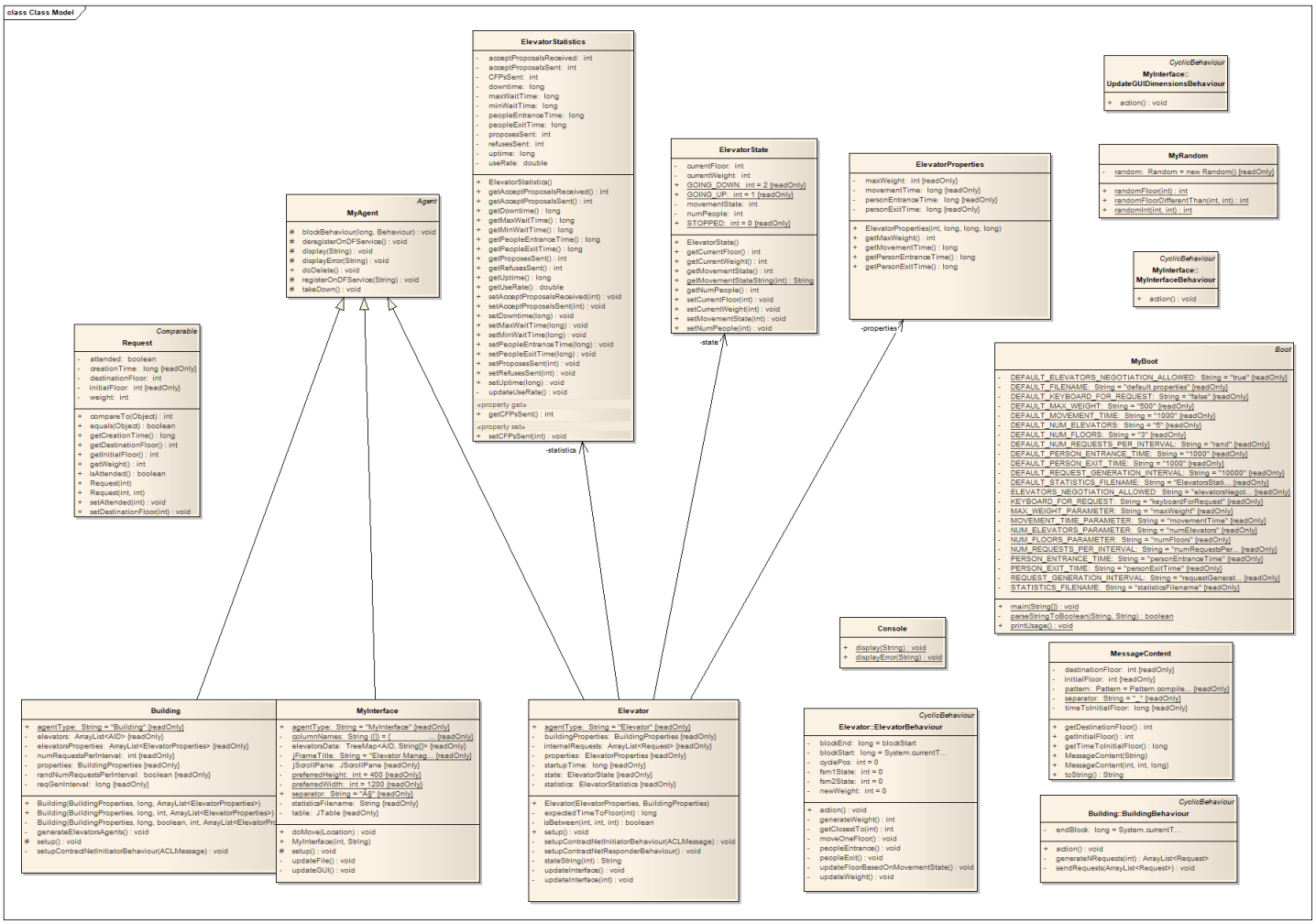
3 Desenvolvimento

3.1 Plataforma/ferramenta utilizada e ambiente de desenvolvimento

A ferramenta escolhida para o desenvolvimento deste trabalho foi JADE. JADE é uma framework que permite o desenvolvimento de sistemas multiagente e é compatível com FIPA, isto é, segue os padrões de software estabelecidos para agentes heterogéneos e interativos e para sistemas baseados em agentes.

O projeto foi desenvolvido utilizando o IntelliJ IDEA Community Edition no sistema operativo Windows 10.

3.2 Estrutura da aplicação



3.3 Detalhes relevantes da implementação

4 Experiências

De seguida, serão detalhadas algumas das experiências efetuadas de forma a ser possível tirar conclusões dos resultados, analisando as estratégias implementadas e a sua adaptação à variação das condições. De notar que para a realização destas experiências, o tempo de movimentação dos elevadores e a capacidade máxima é igual para todos os elevadores. No entanto, o nosso programa permite que os valores sejam diferentes. Pensamos que o mais correto para conseguirmos concluir algo sobre as estratégias era que todos os elevadores tivessem as mesmas condições entre eles, fazendo-as variar (de igual forma entre eles) de experiência para experiência, pois caso contrário iriam existir elevadores a comportarem-se de uma melhor forma entre eles e não implicaria que fosse devido à estratégia.

Todos os resultados foram recolhidos após 2 minutos a correr o programa.

4.1 Experiência 1

4.1.1 Condições iniciais

- **numFloors**=10
- **numElevators**=3
- **requestGenerationInterval**=10000(ms)
- **numRequestsPerInterval**=2
- **maxWeigt Elevator 0,1,2**=[500, 500, 500]
- **movementTime Elevator 0, 1, 2**=[1000, 1000, 1000]

4.1.2 Objetivo

O objetivo desta experiência consiste em comparar os resultados das várias estratégias implementadas com as mesmas condições para todos os elevadores.

4.1.3 Resultados

Utilizando a primeira estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	14	28	18	2	11	7048	53118	61237	46.45
Elevator1	13	30	17	3	15	6017	47065	73290	39.105
Elevator2	8	29	23	4	8	2009	59955	59397	50.23
Average	10.33	28.66	18	2.33	11	5008	49422	64686	47.3

Utilizando a segunda estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	7	30	11	2	10	3005	40101	66248	37.71
Elevator1	9	25	14	2	11	4007	48087	73301	39.61
Elevator2	7	27	14	3	11	3007	27037	65307	29.28
Average	13.3	29.66	22	3	11.33	4493	43584	72963	37.39

Utilizando a terceira estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	0	25	0	0	4	3009	24037	94616	20.26
Elevator1	0	26	0	0	10	6016	42110	64551	39.48
Elevator2	0	25	0	0	10	5001	43072	77594	35.7
Average	0	25.67	0	0	8.33	6390	41522	66216	37.3

Utilizando a quarta estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	0	24	0	0	6	4015	30044	84393	26.25
Elevator1	0	25	0	0	11	17447	80198	38258	67.7
Elevator2	0	25	0	0	7	8019	47104	56341	45.54
Average	0	25.67	0	0	8.67	5005	36718	67679	35.3

4.2 Experiência 2

4.2.1 Condições iniciais

- **numFloors**=10
- **numElevators**=3
- **requestGenerationInterval**=10000(ms)
- **numRequestsPerInterval**=2
- **maxWeigt Elevator 0,1,2**=[500, 500, 500]
- **movementTime Elevator 0, 1, 2**=[300, 300, 300]

4.2.2 Objetivo

O objetivo desta experiência consiste em comparar os resultados das várias estratégias implementadas com as mesmas condições para todos os elevadores, mas com velocidade mais rápida comparada com a experiência anterior.

4.2.3 Resultados

Utilizando a primeira estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	11	27	11	3	12	3172	35584	82124	30.23
Elevator1	3	25	21	0	5	1905	8693	28148	23.60
Elevator2	11	27	11	3	13	3866	19886	86794	18.64
Average	8.33	26.33	14.33	2	10	2981	21388	65689	24.16

Utilizando a segunda estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	3	24	10	1	6	3808	6924	38217	15.34
Elevator1	7	24	6	2	9	1528	9952	93882	9.59
Elevator2	5	25	7	1	11	1576	13751	90953	13.13
Average	5	24.33	7.67	1.33	8.67	2304	10209	74351	12.69

Utilizando a terceira estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	0	24	0	0	11	5933	19591	95421	17.03
Elevator1	0	24	0	0	4	4108	6923	38956	15.09
Elevator2	0	24	0	0	9	2235	13665	70524	16.23
Average	0	24	0	0	8	5005	13393	68300	16.12

Utilizando a quarta estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	0	26	0	0	5	1308	7362	114243	6.05
Elevator1	0	26	0	0	4	1096	2887	49565	5.5
Elevator2	0	26	0	0	17	3125	24272	97331	32.5
Average	0	26	0	0	19.96	5005	11508	87046	10.5

4.3 Experiência 3

4.3.1 Condições iniciais

- **numFloors**=5
- **numElevators**=3
- **requestGenerationInterval**=10000(ms)
- **numRequestsPerInterval**=2
- **maxWeigt Elevator 0,1,2**=[500, 500, 500]
- **movementTime Elevator 0, 1, 2**=[1000, 1000, 1000]

4.3.2 Objetivo

O objetivo desta experiência consiste em comparar os resultados das várias estratégias implementadas com as mesmas condições para todos os elevadores, mas com um número de pisos inferior.

4.3.3 Resultados

Utilizando a primeira estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	10	30	13	3	11	2002	27993	91489	23.43
Elevator1	4	29	20	4	7	1001	16134	105341	13.3
Elevator2	13	30	10	1	16	3003	37062	74525	33.2
Average	9	29.67	14.33	2.67	11.33	2002	27063	90451	23.31

Utilizando a segunda estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	3	26	19	1	6	1001	12013	71402	14.4
Elevator1	12	26	10	2	13	5005	44285	69131	39.05
Elevator2	9	26	13	2	10	2003	51739	62678	45.22
Average	8	26	14	1.67	9.67	2670	36012	67737	32.89

Utilizando a terceira estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	0	22	0	0	7	3003	14062	88279	13.74
Elevator1	0	22	0	0	9	5006	24159	78182	23.61
Elevator2	0	22	0	0	6	2019	12104	36987	24.66
Average	0	22	0	0	7.33	5005	16775	67816	20.67

Utilizando a quarta estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	0	24	0	0	4	8011	13015	81114	13.83
Elevator1	0	24	0	0	14	6640	36735	78388	31.91
Elevator2	0	24	0	0	16	2005	15058	100063	32.5
Average	0	24	0	0	13.1	5552	21603	86522	19.6

4.4 Experiência 4

4.4.1 Condições iniciais

- **numFloors**=20
- **numElevators**=3
- **requestGenerationInterval**=10000(ms)
- **numRequestsPerInterval**=5
- **maxWeigt Elevator 0,1,2**=[500, 500, 500]
- **movementTime Elevator 0, 1, 2**=[1000, 1000, 1000]

4.4.2 Objetivo

O objetivo desta experiência consiste em comparar os resultados das várias estratégias implementadas com as mesmas condições para todos os elevadores, mas com um maior número de pedidos por intervalo e um maior número de pisos.

4.4.3 Resultados

Utilizando a primeira estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	34	72	67	4	24	6039	88099	34081	72.11
Elevator1	34	76	63	8	33	14051	79066	42088	65.3
Elevator2	40	70	63	9	29	8049	86094	34074	71.65
Average	36	72.67	64.33	7	28.67	9380	84420	36478	69.69

Utilizando a segunda estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	33	72	52	7	27	19054	77101	43965	63.69
Elevator1	32	73	52	7	28	14059	90101	31088	74.35
Elevator2	27	73	57	5	29	13083	68083	52065	56.67
Average	30.67	72.67	53.67	6.33	28	15399	78428	42373	64.9

Utilizando a terceira estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	0	65	0	0	25	45094	80108	31014	72.1
Elevator1	0	65	0	0	26	47102	78090	43089	64.4
Elevator2	0	65	0	0	14	31097	94102	26047	78.3
Average	0	65	0	0	21.67	41098	84100	33383	71.6

Utilizando a quarta estratégia,

Agent	CFPs sent	Proposes sent	Refuses sent	Accept proposals sent	Accept proposals received	Max wait time (ms)	Uptime (ms)	Downtime (ms)	Use rate (%)
Elevator0	0	65	0	0	23	54091	79087	41070	65.82
Elevator1	0	65	0	0	26	9092	86101	34088	71.64
Elevator2	0	65	0	0	16	21026	82094	37999	68.36
Average	0	65	0	0	21.67	28070	82427	37719	68.61

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

6 Melhoramentos

6.1 Sugestões para melhoramentos a introduzir no programa

De forma a conseguirmos ter uma melhor comparação era importante adicionar mais estratégias de respostas aos pedidos, conseguindo assim realizar mais experiências e extraindo novas conclusões.

Achamos que a interface podia ser ligeiramente melhorada, de forma a demonstrar melhor o funcionamento dos elevadores, como eles subirem/descerem e abrirem/fechar as portas à entrada/saída dos utentes. No entanto, consideramos que as estatísticas demonstradas conseguem transmitir uma boa ideia daquilo que se está a suceder.

7 Recursos

7.1 Bibliografia

Slides do Moodle sobre JADE
Documentação do JADE
Código fonte do JADE

7.2 Software

JADE (http://jade.tilab.com/)
Eclipse IDE for Java
IntelliJ IDEA Community Edition

7.3 Elementos do grupo

O grupo trabalhou de forma equitativa durante as aulas práticas e houve boa comunicação entre os membros, de forma a que cada elemento soubesse o que estava a ser feito.

8 Apêndice

8.1 Manual do utilizador

Para conseguir correr o projeto, deve:
1. Importar o projeto;
2. Adicionar a *library* do *JADE* ao *build path*;
3. Editar o ficheiro default.properties consoante o que deseja visualizar na interface;
4. Correr, utilizando a classe MyBoot como *main*.