The Bezos Zone

Trabalho realizado por:

- Luís Marques - up201704093
- Nuno Silva - 201404676
- Pedro Azevedo - 201603816

Problema

Os **clientes** têm preferências em comprar alguns **produtos**. Para isso, vão-se informar do **stock** de diferentes **lojas**. Os **clientes** estarão suscetíveis a fazerem mais compras , se preferem comprar **produtos** que estejam em promoção e se as **lojas** onde compram estão na mesma **zona**.

Os **clientes** fazem um pedido a um número pré-definido de **lojas** para elas lhes proporem **produtos** que possam estar interessados em comprar, estes **produtos** são selecionados usando várias **estratégias** (dar os **produtos** mais caros, os **produtos** mais baratos ,, os **produtos** mais comuns e os **produtos** mais raros).

Face a estes **produtos** propostos ao **cliente**, este irá fazer uma **decisão** de compra dos mesmos usando as suas variáveis de **decisão** e faz o pedido à **loja** que por sua vez aumenta o seu **lucro** e manda o pedido ao **Armazém** para remover um **produto** de **stock**.

O objetivo deste trabalho é estudar as **estratégias** das lojas e determinar qual a estratégia de venda que traz mais **lucro** para cada tipo de **cliente**.

Interação e Protocolos

Uma vez que o funcionamento *single threaded* do SAJaS não permite a utilização de *block's*, e de nós fazermos uso delas na entrega anterior, tivemos de alterar de certo modo os Behaviours.

Para a interação entre **agentes** foi utilizada **ACL Messages** sem protocolos de Jade na qual as mensagens são enviadas e recebidas usando **Cyclic Behaviours** ou **Simple Behaviours**, dado que certos **Agentes** necessitam de informações constantemente atualizadas. Para facilitar esta comunicação utilizamos as **Páginas Amarelas** para registar os **Agentes** e tornar a procura deles mais simples.

Loja - Armazém :

Behaviour "C": A **loja** começa por pedir o **stock** total dos **produtos** do **armazém** e espera pela resposta to **armazém**. O **armazém** envia-lhe a lista total de **produtos**, assim como o preço e a quantidade destes. A **loja** recebe, guarda e processa a informação relativa à lista de **produtos**.

Behaviour "B": A **loja**, para confirmar a compra de um **produto**, necessita de remover uma unidade do **produto** ao **stock** do armazém. Para conseguir isto, envia ao armazém o produto e o número de unidades a serem subtraídas ao **stock**. O **armazém** processa o pedido e remove as unidades do stock.

Client - Store:

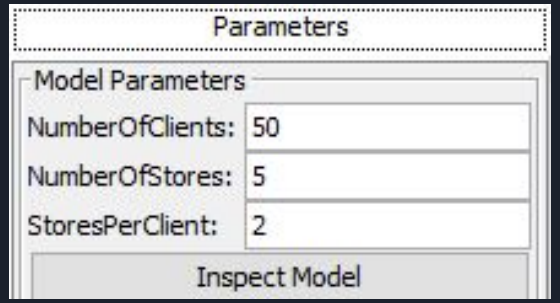
Behaviour "A": O **cliente** inicialmente pede a um número definido de **lojas** que lhe enviem uma lista de **produtos**. A **loja** responde-lhe, enviando então uma lista de **produtos** e os seus preços**.** Após o **cliente** receber a listas de várias lojas, este decide quais os **produtos**, e a que **lojas** os irá comprar.

Behaviour "D": Quando o **cliente** já tem a lista de **produtos** de várias lojas, este envia para as **lojas** a que decidiu comprar **produtos,** uma proposta de compra por cada **produto** que deseja comprar. A **loja** processa o seu pedido, e envia uma confirmação de compra ao cliente, terminando a compra.

Variáveis

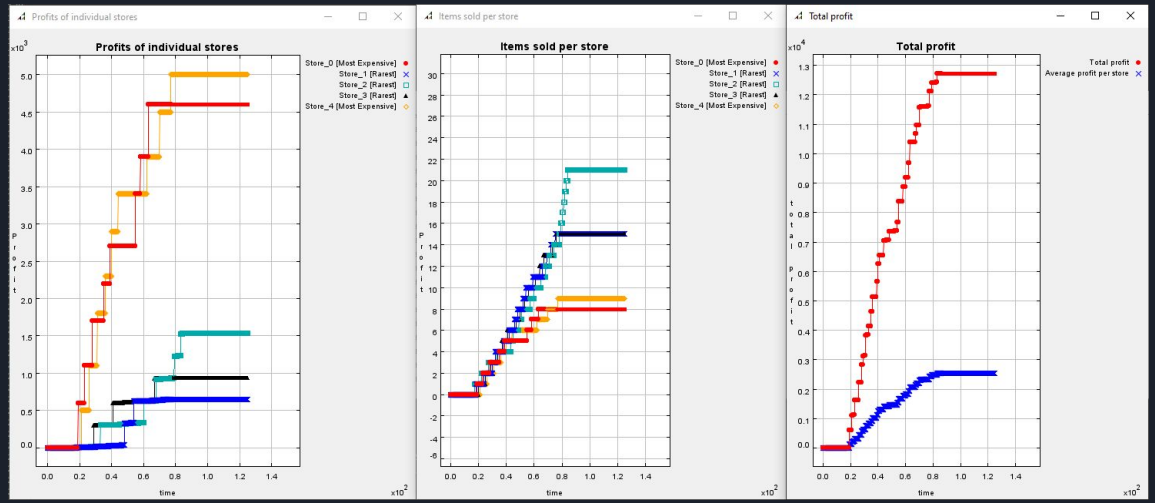
Independentes:

- Número de Clientes
- Número de Lojas
- Número de Lojas visitadas por Cliente



Dependentes:

- Lucro de cada Loja
- Lucro total
- Tempo de execução

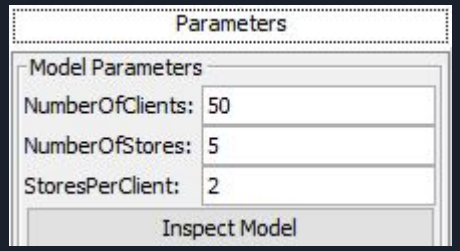


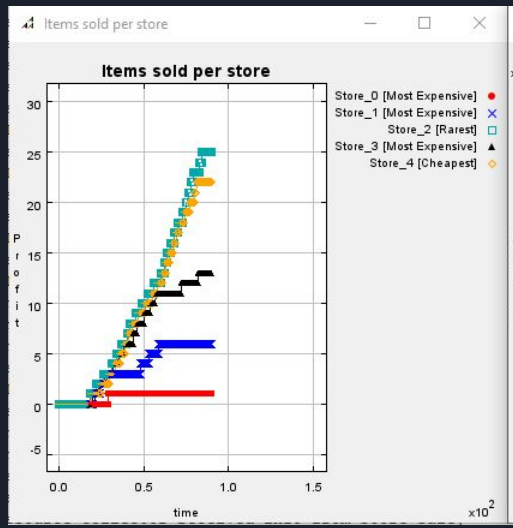
Experiências realizadas e Análise de Resultados - Variação do número de lojas vistas por cliente

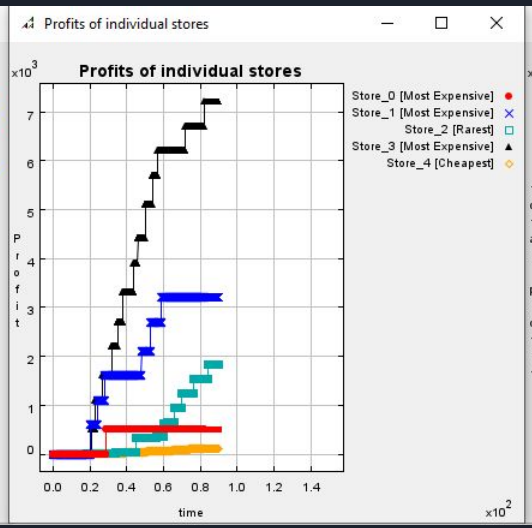
Experiência 1:

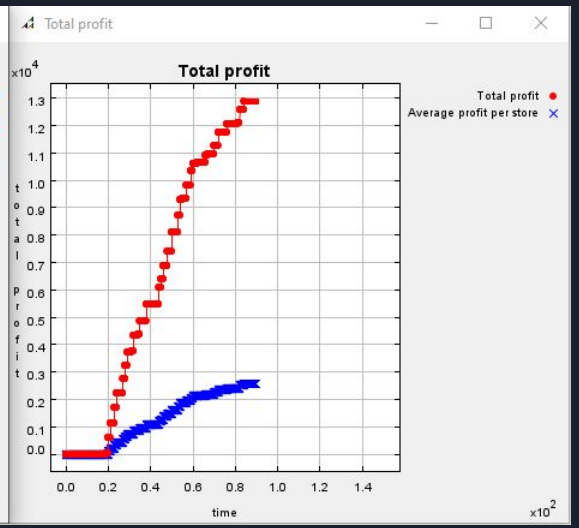
Indicação das variáveis independentes:

Indicação dos lucros e estratégias por loja:



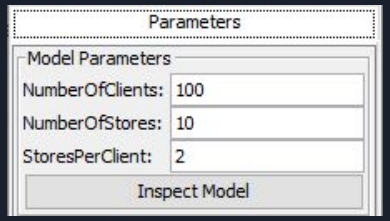






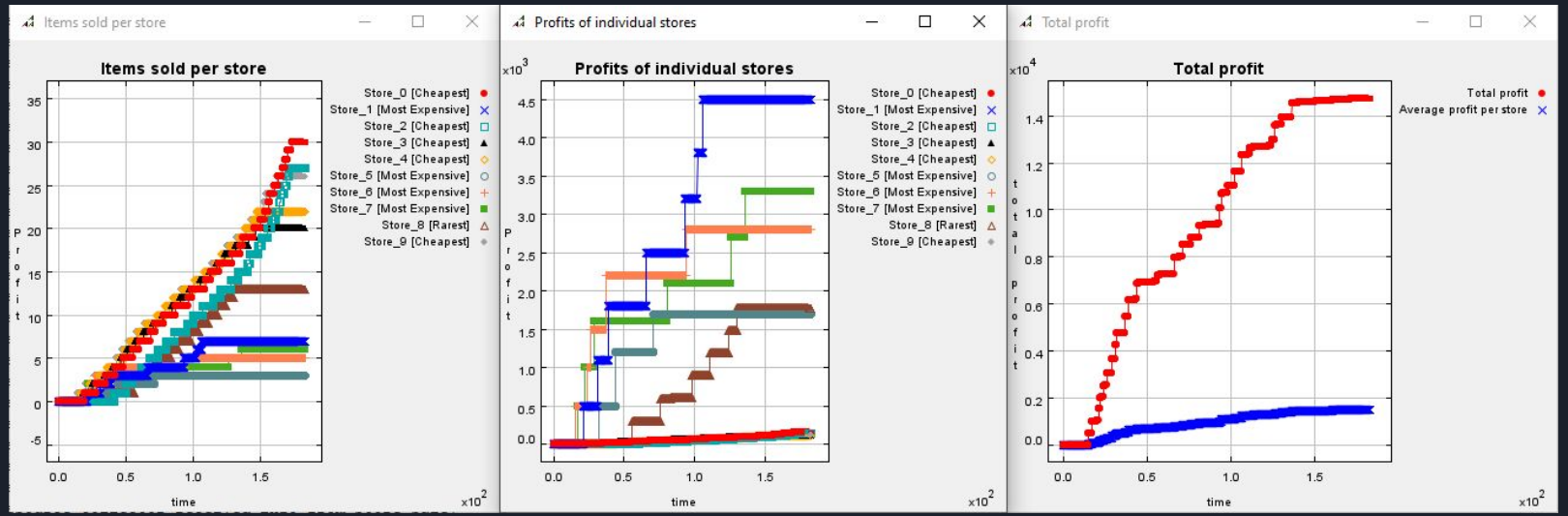
Experiências realizadas e Análise de Resultados - Variação do número de lojas vistas por cliente

Experiência 2:



Indicação das variáveis independentes:

Indicação dos lucros e estratégias por loja:

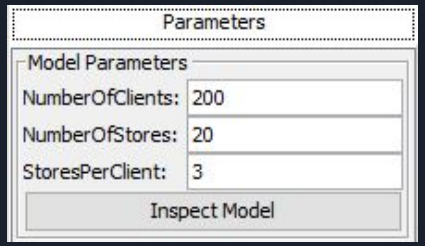


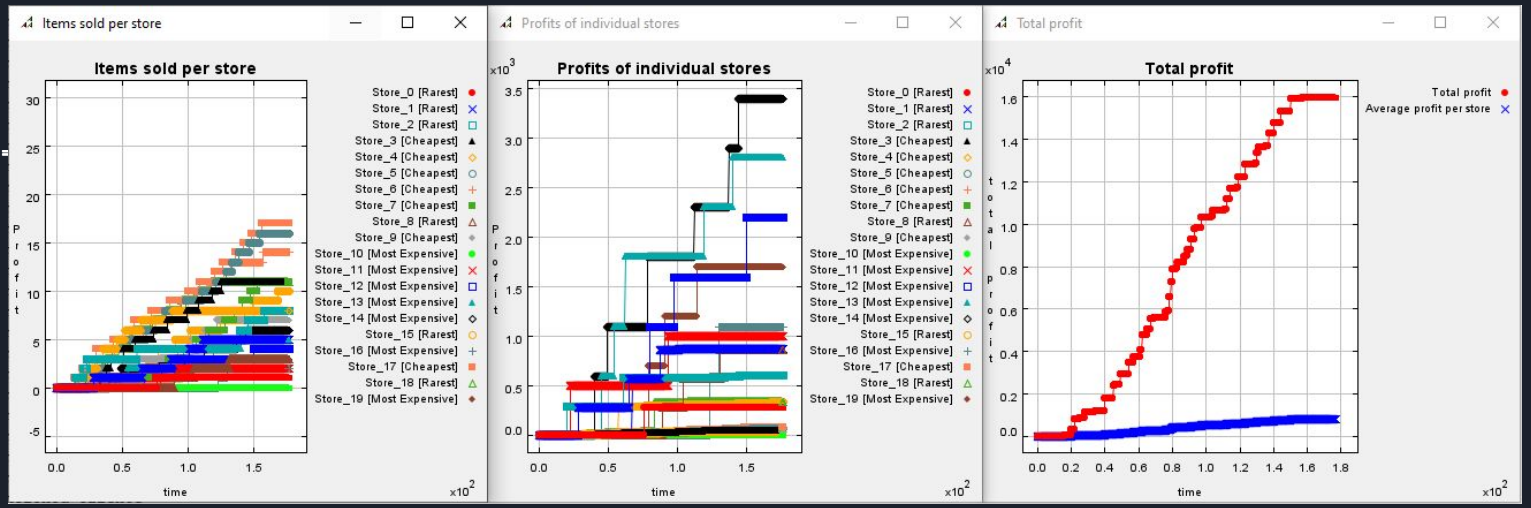
Experiências realizadas e Análise de Resultados - Variação do número de lojas vistas por cliente

Experiência 3:

Indicação das variáveis independentes:

Indicação dos lucros e estratégias por loja:





Análise dos resultados

Após realizar algumas experiências, variando algumas variáveis, foi-nos possível verificar que:

Pudemos realizar várias experiências com um número elevado de clientes e lojas, devido ao SAjaS.

Quanto maior é o número de lojas visitados por cliente, maior é o número de lojas que efetuam vendas e obtém lucro.

As lojas que que obtiveram mais lucro foram as que adotaram estratégias de vender os produtos mais caros, mais baratos, e os mais raros. O facto de haver um lucro alto nas lojas que venderam os produtos caros, foi devido a um produto caro ter o mesmo valor de vários baratos. Logo se um cliente já estivesse interessado em comprar um produto caro, este iria-o fazer. O lucro de uma loja a vender produtos caros pode ser igual ao lucro de uma loja de produtos baratos, mas a quantidade de artigos vendidos ser apenas de um ou dois na de artigos caros, e de muitos nos artigos baratos.

Conclusões

O segundo trabalho da unidade curricular de Agentes e Inteligência Artificial Distribuída teve como foco principal integrar o primeiro trabalho com as funcionalidades de visualização de agentes e as suas interações usando o Repast e o SAJaS.

Numa fase inicial do projeto houve a necessidade de restruturar certas partes do código devido ao facto de o projeto anterior ser multithreaded e haver uma certa incompatibilidade inicial com o SAJaS. No entanto fomos capazes de resolver estes problemas e utilizar uma grande porção das funcionalidades visuais.

É de que acrescentar que aproveitamos para resolver alguns dos problemas que o trabalho anterior tinha e tentamos melhorar o máximo possível a eficiência do programa em si.

Apesar de não termos desenvolvido tudo o que era previsto , principalmente a visualização de interação entre agentes, estamos em geral satisfeitos com o projeto.

The Bezos Zone

Parte 2

Exemplos detalhados de execução

Inicialmente o programa :

- cria o warehouse
- cria as lojas
- pede informação necessária do warehouse para preencher as lojas
- define as estratégias para cada loja
- cria os clientes

```
[MainWarehouse] Created
[Store_0] Created
[Store_1] Created
[Store_2] Created
[Store 2] [MSG SENT; Requested Inventory]
[Warehouse] [MSG RECEIVED; Giving back inventory info now]
[Store 0] [MSG SENT; Requested Inventory]
[Store 1] [MSG SENT; Requested Inventory]
[Warehouse] [MSG RECEIVED; Giving back inventory info now]
[Store 2] [MSG REPLY RECEIVED; GETTING STORE INFO NOW]
[Warehouse] [MSG RECEIVED; Giving back inventory info now]
[Store 0] [MSG REPLY RECEIVED; GETTING STORE INFO NOW]
[Store 1] [MSG REPLY RECEIVED; GETTING STORE INFO NOW]
Most Expensive
In promotion
Most Expensive
[Client_1] Created
[Client_2] Created
[Client_3] Created
[Client_4] Created
[Client_5] Created
[Client_6] Created
[Client_7] Created
[Client_8] Created
[Client_9] Created
[Client_10] Created
```

Exemplos detalhados de execução

```
[Client 2] [MSG SEND; Sending request for products to Store_0]
[Store 0] [Sending Products to Client2]
[Client 2] [MSG RECEIVE; Products received from Store_0]
```

Cada cliente contacta a loja para receber itens da loja e este mesmo recebe os produtos escolhidos.

```
[Client 2] [MSG RECEIVE; Products received from Store_1]
[Client 2] [Deciding which items to buy ]
```

Após os clientes receberem os itens das lojas selecionadas, decidem quais os itens a escolher.

```
[Client 9] [MSG SEND; Want to purchase TV from Store_2]
[Store 2] [Received purchase request from  Client9]
[Store 2] [MSG SENT; Remove item from Warehouse]
[Warehouse] [MSG RECEIVED; Delete Item]
[Warehouse] [The item exists, deleting it now]
[Warehouse] [Sent the confirmation]
[Store 2] [Received Message]
[Store 2] [Confirmed purchase from  Client9]
[Store 2] [Current profit of store is  500.48$]
```

O cliente manda para a loja com o produto desejado para, a loja pede ao warehouse para remover o produto do stock, e manda a confirmação da compra para o cliente aumentando assim os lucros da loja

Classes Implementadas - MainWarehouse

A classe MainWarehouse deriva da classe Agent e representa o armazém partilhado por todas as lojas.

Atributos:

- wares : um objeto da classe Warehouse que guarda uma hashtable contendo os itens e os seus respetivos stocks.

Behaviours:

- C_WAREHOUSE_STORE_RETURN_INVENTORY : behaviour que retorna à loja o inventário atual.

Classes Implementadas - Store

A classe Store deriva da classe Agent e representa a loja que fornece aos clientes os produtos e que realiza a confirmação das compras.

Atributos:

- profit : o lucro atual da loja
- area: zona onde fica a loja

Behaviours:

- C_STORE_WAREHOUSE_REQUEST_INVENTORY: pede ao armazém pelo estado inventário.
- C_STORE_WAREHOUSE_REQUEST_INVENTORY: espera pela resposta do pedido acima.
- A_STORE_CLIENT_PRESENT_PRODUCT_OFFER: apresenta vários itens ao cliente de acordo com a sua estratégia definida, quando lhe é pedido.
- D_STORE_CLIENT_CONFIRM_PURCHASE: a loja recebe a compra do cliente, pergunta ao armazém se existe stock, espera pela sua resposta e confirma ou rejeita a compra.
- E_STORE_RESOURCE_TERMINATION: indica ao resource Collector que a loja está fechada.

Classes Implementadas - Client

A classe Client deriva da classe Agent os cliente que iram fazer compras de acordo com os itens fornecido.

Atributos:

- area : localidade do cliente.
- money_to_spend : dinheiro utilizável.
- spender : se gosta de gastar dinheiro ou não.
- suscetible : se é suscetível a gastar dinheiro em promoções ou não.
- needs : itens que precisa e tem mais probabilidade de comprar.

Behaviours:

- A_CLIENT_STORE_RECEIVE_PRODUCTS_OFFER: Pede á loja para lhe enviar itens que em seguida serão avaliados para serem comprados ou não.
- D_CLIENT_STORE_BUY_ITEM: Indica a loja dos itens que está interessado para realizar a compra..
- D_CLIENT_STORE_RECEIVE_BUY_CONFIRMATION: espera pela confirmação do pedido acima.

Classes Implementadas - ResourceCollector

Este agente comunica com as Stores para receber a informação dos profits de cada loja , que serão utilizados na representação gráfica.

Behaviours:

- E_RESOURCE_LISTEN_RESPONSE: atualiza o profit de uma loja sempre que esta faz uma compra.