U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

OLX

Second-hand Marketplace Simulation using MAS

Agents and Distributed Artificial Intelligence

1st project - Final Delivery

David Silva - up201705373
Luís Cunha - up201706736
Manuel Coutinho - up201704211

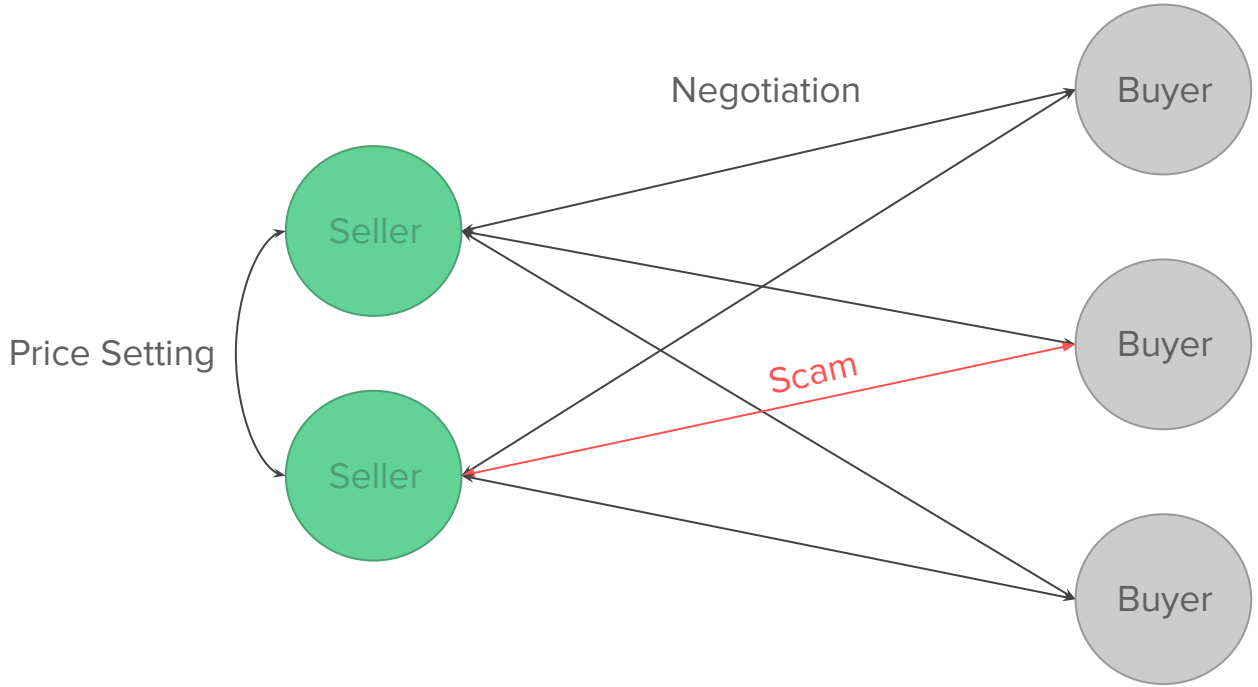
Part I - Presentation

Problem Description

- The modelled multi-agent system represents a **second-hand online marketplace** with two types of agents: **buyers** and **sellers;**
- **Sellers** seek to maximize the selling price of the products while trying to stay competitive with other sellers;
- **Buyers** wish to spend the least cash possible while competing with other buyers and avoiding getting scammed* by sellers;

* When a buyer gets scammed, he loses the money without getting the product and the seller loses credibility (or reputation);

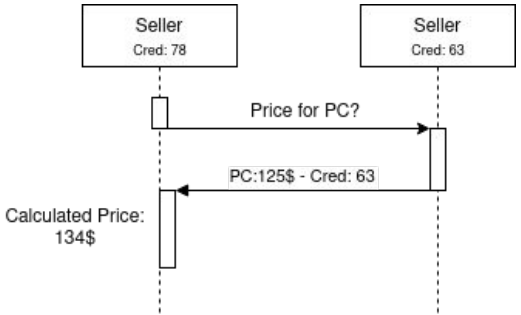
Global Schema



Interactions and Protocols

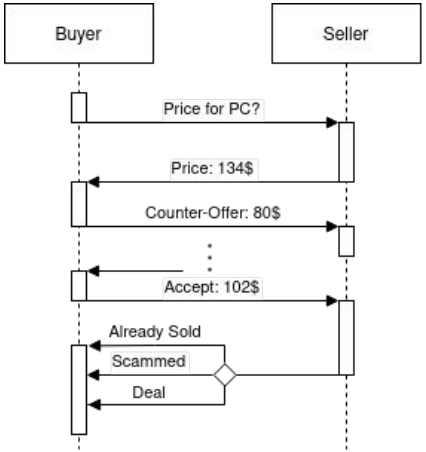
Interaction Seller - Seller: Set Price

- **Request Protocol**
  - Seller asks for other product prices
  - Seller receives prices and credibilities
  - Seller sets price with chosen Price Picking Strategy



Interaction Seller - Buyer: Negotiation

- **Iterated Contract Net Protocol**
  - Buyer asks for the initial prices
  - Buyer and Sellers negotiate a price
  - Seller either sells it, scams the buyer or informs that the product is already sold



Agents Architecture and Strategies

Picking Price Strategies (Seller)

- **No other sellers:**

Naive/Smart: set at a percentage of original product cost (70-90%).

- **Other sellers:**

Naive: set at percentage of minimum offer from other sellers, not taking seller credibility into account.

Smart: set at percentage of average offer from other sellers, weighted by seller credibility.

Counter Offer Strategies (Buyer)

- **Counter-price:**

Smart: Last buyer bid + fraction of difference buyer and seller bids.

Relative and Random Absolute Tit-for-Tat: increase last buyer offer by relative or constant amount. Random absolute TFT changes increment by random ε.

Agents Architecture and Strategies (cont.)

- **Best offer choice:**

Smart: account product price, seller credibility (less credibility, less valued) and time (more rounds, less valued).

Absolute/Relative TFT: naive approach, account only for price.

Offer Strategies (Seller)

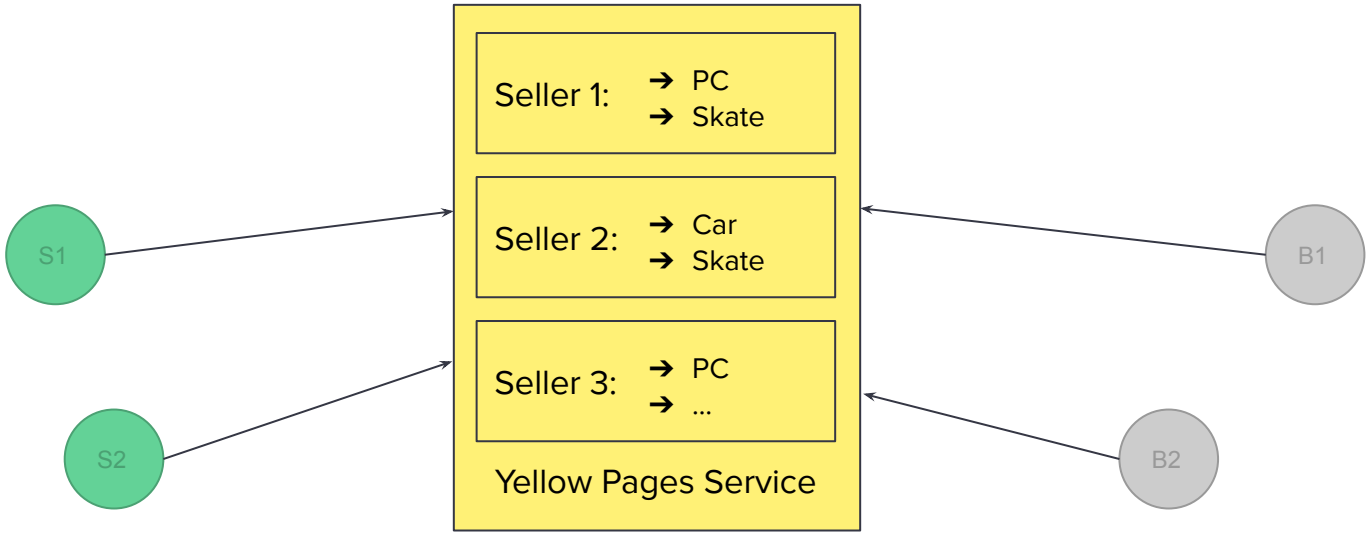
- **Price:**

Smart: Last seller bid - fraction of difference buyer and seller bids.

Relative and Random Absolute Tit-for-Tat: decrease last seller offer by relative or constant amount. Random absolute TFT changes decrement by random ε.

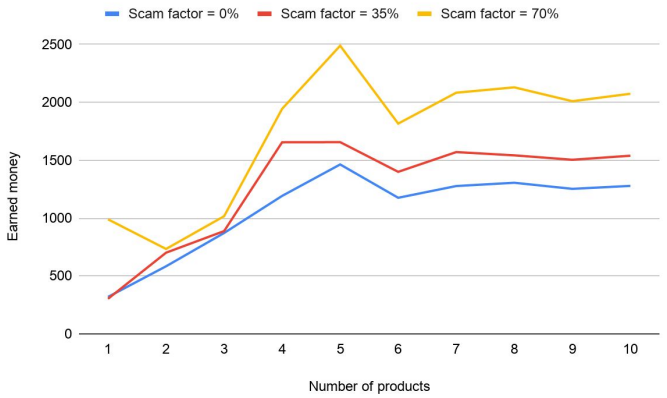
Note: Both offers and counter-offers will stop when the next bid would cross over with the other parties bid or, in the case of the seller, when the lowest allowed price is reached.

Other Mechanisms - Directory Facilitator



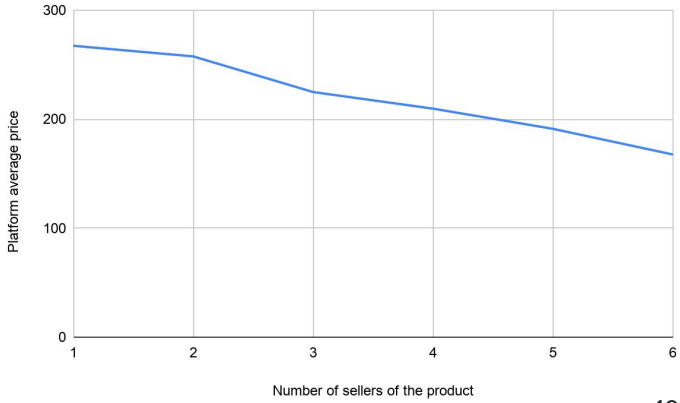
Experiments and Results

- Experiment 1
  - 6 identical buyers, each buying every available product.
  - 3 sellers, each selling every available product with different scam factors.
  - Variable number of products, all with the same price.
  - Goal:
    - Analyse if scammers are more profitable than honest sellers.
  - Results:
    - In contrast to what we expected, the scammer always makes more money.
    - We hypothesise that this is related to having insufficient penalization of scammer's credibility.



Experiments and Results

- Experiment 2
  - 1 type of product.
  - 2 identical buyers trying to acquire the same product.
  - Variable number of sellers (using price picking strategy SMART), trying to sell the same product.
  - Goal:
    - Analyse if the number of sellers of a given product affects the average price at which a product has been sold in the platform.
  - Results:
    - The more sellers trying to sell the same product, the lower the price gets.
    - This is because each seller establishes its price at a little bit lower than the platform average.



Experiments and Results

- Experiment 3
  - 1 type of product.
  - 2 sellers, one with scam factor of 100 (credibility starting at 50) and other with scam factor of 0 (credibility starting at 100).
  - 2 buyers trying to acquire the same product, with differing counter offer strategies: SMART and RELTFT.
  - Goal:
    - Analyse which counter offer strategy is more prone to being scammed.
  - Results:
    - The buyer with RELTFT strategy has a lower average of money saved, since he falls for the scam and only after buys from the honest seller.
    - The SMART strategy, by taking into account the credibility of the seller, avoids falling for scams.

RELTFT	SMART
-69.27	222.67

Conclusions and Future work

- The large amount of controllable variables needed to implement an accurate representation of the real world scenario lead to the creation of a complex system;
- We see the future work on this subject tackling the following issues:
    - Continue and **improve the empirical evaluation** of the results;
    - Use **prediction methods** on the outcome of a negotiation to better estimate its utility;
    - Add the ability of **selling and buying several** of the same product;
    - Better include the concepts of **supply and demand** on the strategies for choosing prices.

Part II - Additional Information

Run Program

- Execute:
  - java -jar olx.jar [-h] [--main] [--kill] --config CONFIG
  - ./gradlew run --args=”[-h] [--main] [--kill] --config CONFIG” in the root directory
- Command line arguments:

-h, --help	shows help message and exit
-m, --main	start agents in new main container
-k, --kill	platform is shutdown after last buyer exits
-c, --config CONFIG	file (YAML or JSON) with buyers and sellers configuration (see Initial Configuration File for further information)

Setting Price Protocol

Seller_0 enters the shop and calculates its initial price using its credibility, elasticity and scam factor (there were no previous sellers selling this product):

```
- START: seller_0{credibility:scamF=43:68, elasticity=14, products={pc:800.0 0.0}}
! seller_0 found no sellers for product pc. Setting price at 368.80
```

Seller_1 asks Seller_0 the price of the PC and calculates its price taking into account the price received:

```
- START: seller_1{credibility:scamF=68:50, elasticity=9, products={pc:800.0=0.0}}
< seller_1 asked the price of pc to the following sellers: [seller_0]
> seller_1 found that product pc has 1 sellers with these prices: [368.80]
! seller_1 set product pc price at 581.00
```

Seller_2 searches for other agents selling the PC and calculates its price taking into account the prices received:

```
- START: seller_2{credibility:scamF=87:25, elasticity=5, products={pc:800.0=0.0}}
< seller_2 asked the price of pc to the following sellers: [seller_0, seller_1]
> seller_2 found that product pc has 2 sellers with these prices: [368.80, 581.00]
! seller_2 set product pc price at 672.15
```

Negotiation Protocol - Buyer POV

1) Buyer asking for PC prices:

```
< buyer_0 ask price of pc to sellers:
- seller_0
- seller_1
- seller_2
```

2) Seller Offer from 3 sellers:

```
> buyer_0 got 3 responses on round 1:
- seller_0 with seller offer pc:800.0 at 648.00$ (with 86 credibility).
- seller_1 with seller offer pc:800.0 at 549.20$ (with 67 credibility).
- seller_2 with seller offer pc:800.0 at 443.30$ (with 52 credibility).
```

3) Buyer Counter-Offers:

```
< buyer_0 sent CFP on round 1:
- seller_0 : pc:800.0 at 324.00$
- seller_1 : pc:800.0 at 234.60$
- seller_2 : pc:800.0 at 241.65$
```

4) Updating buyer's waiting list and evaluating the offers:

```
! buyer_0 reached agreement with seller_2 for pc:
- pc:800.0 at 296.80$ (with 86 credibility) from seller_0 evaluated as 14.277343
- pc:800.0 at 269.80$ (with 67 credibility) from seller_1 evaluated as 7.526041
- pc:800.0 at 242.30$ (with 52 credibility) from seller_2 evaluated as 6.901042
conclusion: best seller is seller_2
! seller_2 is now on wait.
```

5) Seller no longer has product:

```
- seller_1 sent a REFUSE (Seller no
longer selling nintendo).
```

6) Final decision:

```
< buyer_0 sent ACCEPT_PROPOSAL on round 4:
- seller_2 : pc:800.0 at 242.30$
< buyer_0 sent REJECT_PROPOSAL on round 4:
- seller_1
```

7a) Buyer scammed by seller:

```
< buyer_0 received INFORM from seller_2 with pc:800.0 at 242.30$ SCAM!
```

7b) Buyer bought PC from seller:

```
< buyer_0 received INFORM from seller_1 with pc:800.0 at 382.45$
```

Negotiation Protocol - Seller POV

1) Counter-offer from Buyer and Counter-offer from Seller

```
> seller_0 received CFP from agent buyer_1 with raspberry PI:100.0 at 52.83$
< seller_0 sending PROPOSE to agent buyer_1 with raspberry PI:100.0 at 56.50$ (with 90 credibility)
```

2a) Counter-offer from Buyer but seller is no longer selling that product

```
> seller_2 received CFP from agent buyer_3 with Beegee's album (vinil):80.0 at 51,51$
< seller_2 sending REFUSE to agent buyer_3
```

2b) Buyer accepts offer and Seller sells the product (credibility increases)

```
> seller_0 received ACCEPT from agent buyer_1 with offer raspberry PI:100.0 at 56.50$
< seller_0 sent INFORM to agent buyer_1 saying raspberry PI:100.0 at 56.50$, credibility 90 -> 100
```

2c) Buyer accepts offer but Seller sold the product to another buyer

```
> seller_1 received ACCEPT from agent buyer_2 with offer pc:650.0 at 298.97$
< seller_1 sent FAILURE to agent buyer_2 saying Sorry, a better deal came up, already sold it...
```

2d) Buyer accepts offer but Seller scams him, keeps the product and the money

```
> seller_1 received ACCEPT from agent buyer_3 with offer nintendo:240.0 at 164.01$
< seller_1 sent INFORM to agent buyer_3 saying nintendo:240.0 at 164.01$ SCAM!, credibility 100 -> 86
```

Implemented Classes

Main Classes:

- **OLX** - main class responsible for initialization and agents creation
- **Buyer** - agent that represents a buyer
- **Seller** - agent that represents a seller

Models:

- **OfferInfo** - contains the product and offered price
- **Product** - contains the name and its original price
- **Scam** - contains the OfferInfo scammed
- **SellerOfferInfo** - OfferInfo with seller credibility

Utils:

- **Config** - helper class that parses the json/yaml file
- **CoolFormatter** - custom log formatter
- **Stats** - responsible for calculating the statistics of the platform
- **Util** - maths helper class to calculate random numbers, average, ...
- **TerminationAgent** - helper agent used to request platform shutdown

Detailed Agents - Buyer

Attributes:

- **blacklist** - set of Sellers that scammed the buyer (shouldn't be contacted again)
- **counterOfferStrategy** - strategy to decide what to offer for a product (see Agents and Strategies in Part I)
- **patience** - from 0 to 100, lower patience leads to depreciation of value of longer negotiations
- **products** - map of products to their status (either TRYING to buy it, BOUGHT or NO_SELLERS availables)
- **wealth** - amount of money spent

Behaviors:

- **NegotiateBuyer** - one for each product, tries to negotiate with multiple sellers the lowest price possible for a product (see Interaction and Protocols in Part I)

Detailed Agents - Seller

Attributes:

- **credibility** - from 0 to 100, public credibility of a seller influenced by the sells and scams
- **elasticity** - from 0 to 100, representing the % of the advertised price that the seller can negotiate
- **offerStrategy** - strategy to decide what price to ask for
- **pricePickingStrategy** - strategy to decide the initial product price
- **products** - map of products to their decided prices
- **scamFactor** - from 0 to 100, scam probability
- **wealth** - amount of money earned

Behaviors:

- **AskPriceSeller** - asks every other Seller the initial price of their product and calculates its own price before registering a new service in the DF
- **ResponsePrice** - answers to Seller requests for the prices of a product
- **NegotiationDispatcher** - creates the NegotiateSeller behavior
- **NegotiateSeller** - negotiates the prices of different products with various buyers at the same time

Modelling offering and decision-making

- To make the system useful, we needed to **test different approaches** at calculating **offerings** and **choosing** the best sellers/buyers;
- We settled on using an implementation based on the **Strategy design pattern**;
- This pattern allowed us to **decouple the algorithms** for performing these tasks from the rest of the program;
- Create a new strategy by extending the **OfferStrategy**, **CounterOfferStrategy** and/or **PricePickingStrategy** abstract classes;

Initial configuration file - JSON

Products:

```
"products": [
   {
       "name": "pc",
       "price": 800
   },
   {
       "name": "skate",
       "price": 150
   }
 ]
```

Sellers:

```
"sellers": [
   {
       "scamFactor": 68,
       "elasticity": 14,
       "offerStrategy": "ABSTFT",
       "pickingStrategy": "NAIVE",
       "products": ["pc", "skate"]
   },
   …
 ]
```

Buyers:

```
"buyers": [
   {
     "products": ["pc"],
     "counterOfferStrategy": "SMART",
     "patience": 80
   },
   {
     "products": ["skate"],
     "counterOfferStrategy": "test",
     "patience": 63
   }
 ]
```

- Defined offer/counter-offer strategies are ABSTFT (random absolute TFT), RELTFT (relative TFT) and SMART. Picking strategies are SMART and NAIVE.

Initial configuration file - YAML

Products:

```yaml
products:
 - name: pc
   price: 800
 - name: skate
   price: 150
```

Sellers:

```yaml
sellers:
 - scamFactor: 68
   elasticity: 14
   offerStrategy: SMART
   pickingStrategy: NAIVE
   products:
     - pc
     - skate
 - …
```

Buyers:

```yaml
buyers:
 - products:
     - pc
   counterOfferStrategy: ABSTFT
   patience: 80
 - products:
     - skate
   counterOfferStrategy: RELTFT
   patience: 63
```

- Defined offer/counter-offer strategies are ABSTFT (random absolute TFT), RELTFT (relative TFT) and SMART. Picking strategies are SMART and NAIVE.