4. Trading agents competition

Today, we will practice what we learned about Jade, ontologies and communication protocols. Your task is to trade books. Your agent will trade with other agents in the system and compete for limited resources (books).

The task

In the beginning, each agents receives a list of books it should obtain (its goals) and utility these books have for its user (someone who ordered the books). At the same time, it receives a list of books it owns and a given amount of money. The goal of the agent is to maximize its utility. The utility is computed as the sum of the utilities of books it owns and are on the list of books it should obtain, and the money the agent has at the end of the trading. The books the agent owns and but are not on the list have zero utility. It the agent has the same book twice, it is computed only once.

System description

In the system, there are agents of two types. One agent is `Environment` that takes care of assigning the goals to the agents and managing what each agent owns. The agents send transaction requests to this agent. The environment also periodically prints the current utilities of all the agents. At the beginning of the trading, it sends information that the agents can start trading and also provides the information on the goals of the agents and the books it owns.

The other type of agents are the traders themselves. They negotiate the trades directly with other traders using the `Contract-Net` protocol.

The Environment

Agent of type Environment provides the environment service. At the beginning of trading it sends the `StartTrading` REQUEST to all trader. This tells the traders that they can start trading.

The agent additionally provides the following services:

1. `GetMyInfo` – a service, which provides the information on the books owned by the agent, the amount of money it owns, and what are its goals (`REQUEST GetMyInfo`, `REQUEST` protocol, the result is `AgentInfo`).
2. `MakeTransaction` – a service which makes the transaction of books and money between two agents. The agents must specify the sender and the receiver of the books, the lists of books and amount of money the agents sends and also the list of books and money the agents expects to receive. After accepting a request, the Environment waits for a request from the other agent, it checks whether the information on the trade matches and if it does, it performs the transaction. Otherwise, both the traders receive a `FAILURE`. In case of a successful trade they receive an `INFORM` with "done" as content. (`REQUEST MakeTransaction`, `REQUEST` protocol).

Moreover, the agent also prints the current utility of each agent every 15 seconds. Every 5 seconds, it removes all messages – request for book transactions – older than 5 seconds which were not matched with the other transaction request.

Trading agents

The trading agents provide the book-trader service and communicate among themselves using the Contract-Net protocol. Their current implementation can only ask for books, i.e. ask other agents whether they have a book and under which conditions they are willing to sell it. For successful trading the agents need to implement both sides of the Contract-Net protocol.

The trade itself is executed in the following steps:

1. Agent A sends agent B (and potentially others) a `CFP SellMeBooks` with the list of books it wants to buy. The IDs in the list of books have no meaning at this point and should be set to 0.
2. Agent B prepares a list of proposals (`PROPOSE`), he is willing to sell the books to A for, at the same time it adds the IDs of books it would eventually sell. It sends it as a predicate `ChooseFrom`, which contains a list of `Offer`.
3. Agent A selects an `Offer` from the list, and sends back to B a `Chosen` with a single `Offer`. The `Offer` again contains the IDs of books the agent would eventually send (`ACCEPT_PROPOSAL`). If agent A does not choose any offer (or chooses an offer from another agent), it sends `REJECT_PROPOSAL`. For technical reasons, A can accept an offer only from a single agent, as the transactions are identified by the conversation ID of the initial communication.
4. Agent B prepares a `MakeTransaction` message and sends it to the environment. At the same time, it also sends an INFORM to agent A telling the transaction is submitted. The ID of the transaction is set as the ID of the message the agents used to set the deal. The transaction needs to specify the sender and the receiver. At the same time, it must contain the list of books the agent sends and the list of books it expects to receive.
5. Agent A prepares its version of MakeTransaction. It also has to send the sender and the receiver and set the lists of books it sends and expects to receive.
6. Agents A and B receive an `INFORM`, if the transaction is successful, or a `FAILURE`, if an error occurs (i.e. the lists do not match, the other agent sent the message too late, etc.).

The sample trading agent acts according to the description above, it tries not to send books it does not own (but it can do it occasionally if it promises the same book to two agents, the environment makes sure only one of the transactions is successful). The agent does not optimize its utility in any way. Its main goal was to test the environment.

Tips and tricks

1. Try not to trade as fast as you can, the system may not be able to handle that. Put at least a small delay between the individual requests.
2. The trading will take exactly three minutes and the results will be evaluated after that, we will run the trading a few times to limit the effect of the random setting of books and goals in the beginning.
3. If you want to start the system, you need to add all trading agents first and the environment later.
4. Consider the selling price of the books. Especially for books you do not have as a goal, and therefore their value for you is zero. It is always beneficial to sell it for almost anything, however selling it too cheap may be wasteful.
5. Sometimes it is better to sell a book you have in goals, if you receive more money that its utility.
6. Exchanging books among agents can be interesting, e.g. if they both have a book they do not need but the other agents wants it.

7. It may be a good idea to add at least one `Offer` without any books, i.e. only with money.
8. You can try to make money trading books you do not need.

What have we learned today

1. Create a more complicated agent
2. Apply ontologies and communication protocols in a more “practical” example.
3. Create trading strategies and set prices.
4. Survive in a competitive system.