TAC Game Report

Tristan Claverie
950418P612
trcl16@student.bth.se

Damien Duvacher
XXXXXXXXXX
dadu16@student.bth.se

Introduction

In computer science, an agent is a software who works autonomously. It acts like an automaton according to how it was developped.
It's a kind of artificial intelligence.
An agent can be :

- Reactive: It can act regarding to his environment.
- Proactive: It can take initiative to reach its goal.
- "Sociable": It can communicate with other agents.

Agents can be use in many fields, for example there is an agent who organize import and export of containers in a container terminal at Göteborg, this agent permits to fill, empty and stack containers from ships to the docks via machines. Machine moving are managed by agents and are optimized to be the quickest as possible.
Another example very simple is just a reactive agent which can send automatic responses when one orders a command or something else.
So, agent are useful to do some tasks and can make the life easier.
The TAC agent competition is an international forum designed to promote and encourage high quality research into the trading agent problem.
In the TAC classic, we are an agent reponsible of 8 clients who want to go in holidays. We have to create travel packages to answer to their preferences (Flight date, Hotel preferences, Entertainements asked).
We have to participate to different auctions to make the packages. There are 3 different auctions:

- Flight auctions
- Hotel auctions
- Entertainement auctions

We will describe the behaviour of each auction in the rest of this report.
We will participate to a competition between each agent of the different group of student.
At the end of the competion we will have a score regarding our holdings and the higher score will win.
So we have to design an agent to win this competition, this report will describe how we've developped our agent.

I. Intuition

In this part we'll talk about what we've done before starting the development of our agent and what was our first thinking about the project.
Our first step was to analyse the dummy agent code to see how it works and how we can adapt the code for our agent.
First we wil describe the different auctions:

- Flight auction: for this auction, a price is set at the begining of the game between 250 and 400 and then this price changes at regular intervals during the game (increase or decrease). There are no limited places for one flight so the only thing we have to do is do decide when we buy our flight.
- Hotel auction: this auction is an english auction. We put a bid with a price and when the auction is closed, the ask price become the 16th higher price and then all the bids with a price over the ask price (according to the stock: there are 16 places available for each hotel and each night) are validated and players get the rooms. So to be sure to get an hotel we have to put an high enough price.
- Entertainement: players can send bid to sell or buy entertainements. If some bids match together, the transaction is done.

For the flight auctions, the dummy agent buys everything at the beginning of the game, he buys all the ones he needs. That can be the best solution if by luck all the auctions will grow but it's not always the case so we have to think about another strategy.
For the hotel auctions, the dummy agent updates his bid each time the quote is updated. He puts a new bid, the price of this bid is equal to the ask price + 50. It's not a really good strategy because 50 is a fix variable and variation of the ask price can be higher or lower in different games. Furthermore maybe this price is not enough high to get the hotel needed.
And for the entertainement, the dummy agent tries to sell entertainements that it doesn't need at the higest price and it tries to buy what each client's most wanted entertainement at the lowest price. When the quote is updated, he increases or decreases the price accordingly.
For our agent, we decided to adopt a global strategy wich is to get the best packages regardless of the price. With this strategy we will have a good score in satisfaction and the score about the price depends on the game and on the other players. Now let's present our strategy in details.

II. Agent Pascal

A. Flight strategy

For the hotel auction strategy, we've decided to put 2 limits:

- A low limit: if the price of the hotel that we want goes under this limit, we buy it.
- A high limit: if the price become too high and goes over this limit we buy it too, to prevent a too big spending.

At the begining we just put a low limit, but after many tests, we saw that the price usually become really big after some times then, we decided to put a high limit. In a first step we've put some fixed limits (200 for the low limit and 400 for the high limit) but after thought, we realized that was not really relevant because the prices begin between 250 and 400 so there can be a gap between some auction.
So we decided to adapt the limits for each auction. We've put different limits for the each auctions, regarding the starting price of the auction:

- Low limit = Starting price - 10% of the starting price
- High limit = Starting price + 10% of the starting price

This is a better strategy than before because we adapt our price according to each auction and not only a global strat for all the auctions.

B. Hotel strategy

For the hotel strategy, it's a bit more complicated than the flight one.
As these auctions are english auction, we thought that we have to put a enough big price to get our hotels but not to big to not make the price grow too much.
So this strategy is based on a variable "delta". This delta is the average ask price changes, each time the auction change, we update the δ by this function: $\delta = (\delta + newAskPrice - oldAskPrice)/2$. With the function we can have an overview of what will happen in the future rounds.
Each time the quote for the auction is updated, we compare the ask price with our bid and then there is 2 possibility:

- The ask price is higher than our bid, in that case we change our bid price for the ask price+δ.
- The ask price is lower than our bid in that case we change nothing

This strategy permits to have a bigger price than the asked price because each time the asked price go over our bid we update it. And it permits to not have a too big price because if our price is higher than the asked price we don't update our bid.
Updating with the delta allows to predict the future growth of the price and permits us to not update our price each time there is a new ask price but just in case we will not have the hotel.

C. Entertainement strategy

Concerning the entertainements, the major improvement was to buy a complete set of entertainements, and not just the best as does the Dummy agent. For this purpose, we had to rank the preferences of each client, from his most wanted entertainement to his less wanted.
Our purpose being to provide complete packages at all cost, we decided to buy the entertainements right from the start, at the highest price possible (200) in order to be sure to complete our packages.
Furthermore, in a competition, it is sometimes important to focus on how to annoy other competitors, that's why we decided to keep all of our original entertainments, expecting a starvation for the others.
This strategy allows us to content the clients, while annoying the other agents because we have more entertainements than needed.

III. Results

The first result that jumped to our eyes was the score : we ended up in all the game with a negative score, revealing unsatisfaction of our clients.

A. Flight

This part of the game went smoothly : our strategy allowed us to limit the price at which we buy the flights, and it resulted in a moderate price at the end.
Furthermore, we have not even once disappointed our clients : we have had at all time all the fights wanted.
Thus, we believe our strategy to be suited for the TAC game.

B. Hotel

This part is more complicated. In two games out of three, we didn't get one hotel night for several clients.
The results were numerous infeasible packages (4 or 5 out of 8) and a huge drop in client's satisfaction.
This problem can come from a problem with our implementation or with the communication to the server, however we've not been able to pinpoint it. The second problem was the price at which we get our hotels.
During the game, prices increased almost endlessly, and we've had our hotel nights for a very big price usually (sometimes more than 500/night).
However, we did not have a swap strategy, and usually we bought the good hotel for a huge price whereas the cheap one was available. However, this strategy is far more difficult to implement because a client can't change hotels during his stay.

C. Entertainement

This part of the auction went quite well.
As expected, we got for each game all of the entertainements we asked for, which results in a high satisfaction.
However, it may have append that we bought an entertainement too high for a client, and got a negative score for it. Our strategy was good but could still be improved.
As for the other side of our strategy, we've seen very little

effects : the other agents did not seem to suffer from the starvation we tried to induce.

D. General

As we've seen, the main problem with our strategies was the price. We had one of the highest satisfaction score in each game, but ended up with a negative result because we bought hotels and entertainements too expensive.
However, this behaviour was a risk we were willing to take, so the real problem according to us is the hotel nights we couldn't get, which was a real hard hit.

Concusion

In the end, we believe our ideas for the TAC game were decent and worked well, except for the hotel auction.
We chose to optimize client satisfaction, however another choice could have been to maximize the final score : difference between client satisfaction and insatisfaction.
Furthermore, we provide three other ways to improve our agent, but they are difficult to implement and we prefered to focus on having a less performant, working agent rather than a more performant, buggy agent.

- Using a genetic algorithm, we could have had a dynamic optimal state of the game, refreshing at each update in the game (auction closed, bid accepted, auction updated, . . .). This way, we could ensure an optimal or quasi-optimal set of packages for a reasonable price.
- With machine learning over the auctions of the hotels, we might have been able to infer which set of nights did the other agent wanted and go for the other hotel if the competition was too high.
- Using Markov chains, it would have been possible to predict the next state of the game according to the current one and react accordingly. However, this is hardest strategy to implement out of the three.

Finally, we were pleased to take part in this competition, even though our agent was lacking in some ways.