



TAC Techniques

Programming Trading Agents

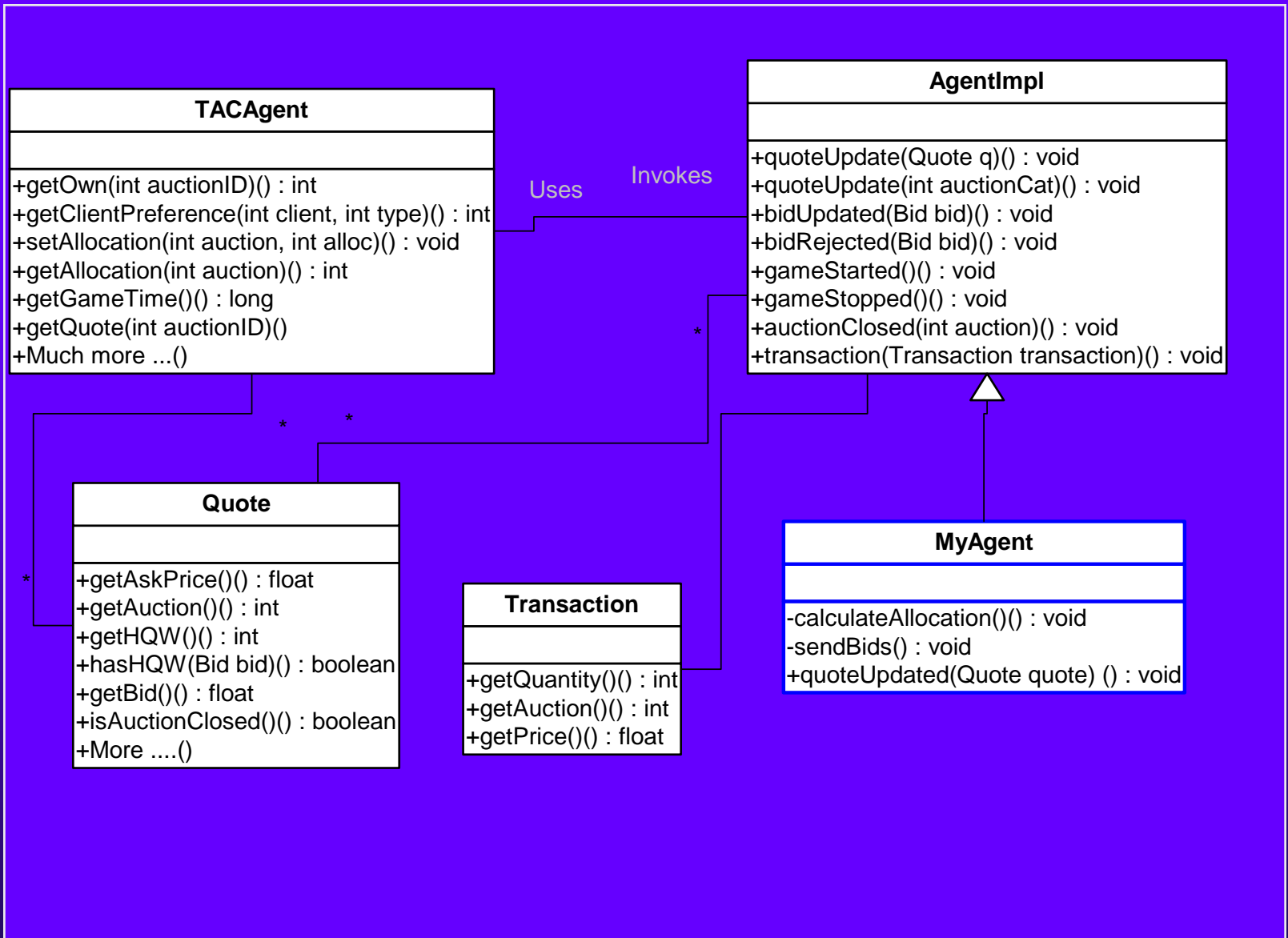


Subjects

- ◆ Design overview
- ◆ Planning your bids
- ◆ Buying flights
- ◆ Buying hotels
- ◆ Buying entertainment



Design overview





| TACAgent |
|---|
| +getOwn(int auctionID()) : int +getClientPreference(int client, int type)() : int +setAllocation(int auction, int alloc)() : void +getAllocation(int auction)() : int +getGameTime()() : long +getQuote(int auctionID)() +Much more ...() |

Provides game information



| Quote |
|---|
| |
| +getAskPrice()() : float +getAuction()() : int +getHqw()() : int +hasHqw(Bid bid)() : boolean +getBid()() : float +isAuctionClosed()() : boolean +More() |

Provides price information ...



| Transaction |
|--|
| |
| +getQuantity() : int +getAuction() : int +getPrice() : float |

Information about received goods



| AgentImpl |
|--|
| <pre>+quoteUpdate(Quote q)() : void +quoteUpdate(int auctionCat)() : void +bidUpdated(Bid bid)() : void +bidRejected(Bid bid)() : void +gameStarted()() : void +gameStopped()() : void +auctionClosed(int auction)() : void +transaction(Transaction transaction)() : void</pre> |

All methods an agent should have...



MyAgent

```
-calculateAllocation()() : void  
-sendBids() : void  
+quoteUpdated(Quote quote) () : void
```

Your agent...



Planning your bids

- ◆ Allocation: the total quantity of an item required
- ◆ The planner should calculate the allocation
- ◆ $\text{Quantity To Buy} = \text{Allocation} - \text{Owned}$



Naïve planning

1. Retrieve client preferences
2. Decide on the hotel type
3. Allocate the rooms for each auction
4. Allocate in-flight and out-flight on each end
5. Allocate the best entertainment on the best day



Naïve planning

Retrieving client preferences:

```
for (int i = 0; i < 8; i++) {  
    int inFlight =  
        agent.getClientPreference(i, TACAgent.ARRIVAL);  
    int outFlight =  
        agent.getClientPreference(i, TACAgent.DEPARTURE);  
    int hotel =  
        agent.getClientPreference(i, TACAgent.HOTEL_VALUE);  
}
```



Naïve planning

Decide on the hotel type:

```
if (hotel > 70) {  
    type = TACAgent.TYPE_GOOD_HOTEL;  
} else {  
    type = TACAgent.TYPE_CHEAP_HOTEL;  
}
```



Naïve planning

Allocating the rooms:

```
for (int d = inFlight; d < outFlight; d++) {  
  
    auction = agent.getAuctionFor(TACAgent.CAT_HOTEL, type, d);  
    agent.setAllocation(auction, agent.getAllocation(auction) + 1);  
  
}
```



Naïve planning

Allocating flights:

```
auction = agent.getAuctionFor(TACAgent.CAT_FLIGHT,  
TACAgent.TYPE_INFLIGHT, inFlight);
```

```
agent.setAllocation(auction, agent.getAllocation(auction) + 1);
```

```
auction = agent.getAuctionFor(TACAgent.CAT_FLIGHT,  
TACAgent.TYPE_OUTFLIGHT, outFlight);
```

```
agent.setAllocation(auction, agent.getAllocation(auction) + 1);
```



Naïve planning

Allocate the best entertainment on the best day:

```
int eType = -1;
while((eType = nextEntType(i, eType)) > 0) {

    auction = bestEntDay(inFlight, outFlight, eType);
    agent.setAllocation(auction, agent.getAllocation(auction) + 1);

}
```



Why is it naïve?

As time progresses hotel auctions close

Planning is done according to preferences

Planning becomes infeasible

Useless goods are bought

The planning should be adapted



Why do we use it?

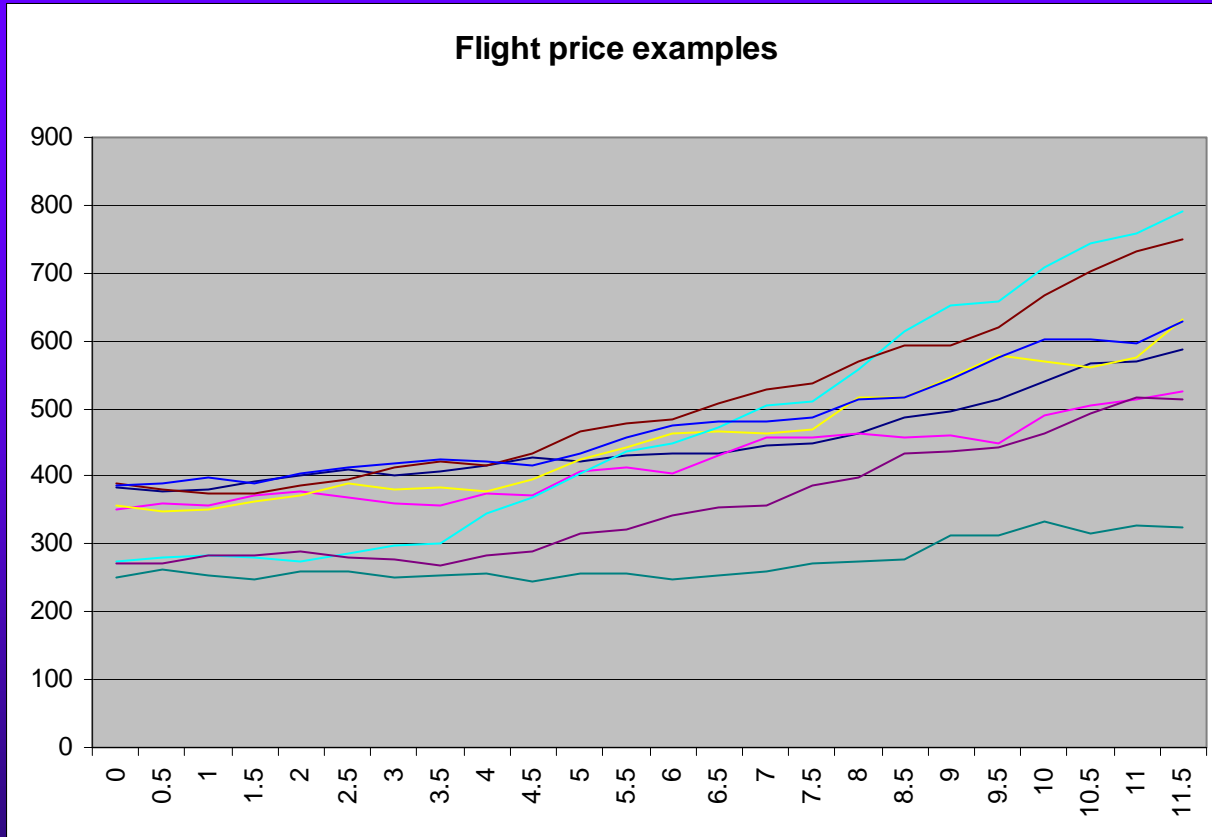
Difficulty of the decision



Buying flights

- ◆ Flight prices are decided by the server
- ◆ One can accept this price or not
- ◆ Usually prices will rise
- ◆ There are always enough flights
- ◆ The basic bid strategy is just to bid 1000

Flight prices





Flight prices

- ◆ Start price randomly chosen between 250 and 400
- ◆ Every 24 ... 32 seconds changes are made
- ◆ The change is chosen between -10 and $x(t)$
- ◆ $x(t) = 10 + (\text{time}/12:00) \cdot (x - 10)$
- ◆ x is randomly chosen 10 ... 90 for each flight auction



Buying hotels

- ◆ The first hotel auction closes after 4 minutes
- ◆ Then one hotel auction closes every 1 minute
- ◆ This problem is an allocation problem
- ◆ This is also a bidding problem
- ◆ The value of possession depends on it



Naïve hotel bidding

1. Retrieve the quantity to buy
2. Is the Hypothetical Quantity Won adequate?
3. If not, then increase the ask price with 50



Naïve hotel bidding

Retrieving quantity & comparing with HQW:

```
if (auctionCategory == TACAgent.CAT_HOTEL) {  
    int alloc = agent.getAllocation(auction);  
    if (alloc > 0 && quote.hasHQW(agent.getBid(auction)) &&  
        quote.getHQW() < alloc) {  
        .....  
    }  
}
```



Naïve hotel bidding

Increase the ask price and submit the bid

```
Bid bid = new Bid(auction);  
prices[auction] = quote.getAskPrice() + 50;  
bid.addBidPoint(alloc, prices[auction]);  
agent.submitBid(bid);
```



Why is it naïve?

- ◆ Hotel bidding becomes more competitive
- ◆ It is possible to construct a bid when your competitor is not
- ◆ Your bid price can become more than its value



Planning
should be
changed when your



Buying entertainment

- ◆ Continuous auction: at any moment in time any agent can place a sell or buy bid
- ◆ Ask price = minimum selling price
- ◆ Bid price = maximal buying price
- ◆ If ask price $<$ bid price then the transaction is done



Naïve entertainment bidding

1. Calculate the quantity: allocation – owned goods
2. Either sell the redundant entertainment at a certain rate
3. Or buy the shortage with a price that rises linear over time



Naïve entertainment bidding

Calculate the quantity:

```
int alloc = agent.getAllocation(auction) - agent.getOwn(auction);  
if (alloc != 0) {  
    Bid bid = new Bid(auction);  
    .....  
}
```



Naïve entertainment bidding

Selling redundant entertainment:

```
if (alloc < 0)
prices[auction] = 200f - (agent.getGameTime() * 120f) / 720000;
...
...
```



Naïve entertainment bidding

Buying entertainment tickets:

```
Else
```

```
prices[auction] = 50f + (agent.getGameTime() * 100f) / 720000;
```

```
bid.addBidPoint(alloc, prices[auction]);
```

```
agent.submitBid(bid);
```

```
}
```



Why is it naïve?

- ◆ Your selling price can be higher than the rest (constant)
 - Your agent will have more redundant entertainment tickets with penalty of 200
- ◆ A bid price is not decided for every client
 - It is possible you bid more than the bonus you will receive



Why do we use it?

- ◆ One should first link every bid to a client
 - Only then would we know if a bid is higher than its utility
- ◆ This is difficult: owned information is inadequate



The End

- ◆ The Exercises give an introduction in programming TAC Agents.