University of Turin

School of Management and Economics

Master Degree in Quantitative Finance and Insurance

Simulation of an auction with NetLogo

Stefano Grandi and Sabrina Peretti

Introduction

This model has the purpose to simulate an auction.

In general, an auction is a market where objects (i.e cars, artworks, ecc.) or real estates, securities or rights can be sold, below established rules, which identify:

- the type and the number of goods;
- the action type (private or public; ascending bid auction or descending bid auction; first price sealed bid auction or second price sealed bid auction);
- who is allowed to participate (everyone or only under formal invitation);
- the form of offers (written, oral, online);
- the punishment of not plain dealing.

The auction participants are divided in: the bidders (who make offers), the sellers and the auctioneer. Sometimes the seller and the auctioneer can be the same person.

The bidders' offers are called bids. In general, the bids have a minimum and a maximum value and, depending on the auction type, the winner bid can be the lowest or the highest bid or the second bid lower or higher.

The auctions can be seen like not cooperative plays, because the bidders compete one against the others for the same object and, at the same time, each bidder compete against the seller for maximizing each own profit.

We will use NetLogo 5.2 (https://ccl.northwestern.edu/netlogo/) to create our simulation of auctions.

We decide to simulate a private ascending bid auction (known as English auction), the most popular auction. We illustrate briefly the hypothesis and the fundamental structure of our model.

In an English auction there are sold only objects; therefore the bids are in an oral form and who makes the last maximum bid is the winner of the auction. The last maximum bid is called also the hammer price. We set the minimum raise of the bid in a slider in the interface called 'MinimumBid', in this way the user could run the model with different values of this parameter and compare the results.

In our model the number of bidders is set by the users, using the slider 'Howmanybidders' set in the interface.

Therefore, we set also the maximum duration of an auction by a slider: the user can decide the maximum number of raise, if this number is reach the model decides the winner of the bid, without raise no more. This slider is called 'Time'.

First of all, in our model, we suppose that the seller and the auctioneer are the same person, and that the bidders are at least two.

Furthermore, there is only one object in our model, because our goals are to follow the dynamics of the maximum price in the auction and to know the satisfaction of the auctioneer.

The objects starting prices are defined by the slider 'PriceObject' in the interface.

Each bidder has an amount of money set randomly. The bidders' money are set randomly between a minimum value (50 €) and a maximum value, which could be set by the user using the slider 'HowManyEuro' in the interface.

Each bidder has a reservation price for each object: it is the maximum price that each bidder is willing to spend for that object. It is unknown to the other participants and its initial value is independent from other bidders. We create a variable called 'ReservationPrice' and we set each initial value of reservation price randomly as a function of the object price.

We decide that the variable 'ReservationPrice' is also a feature of the auctioneer, it is equal to the starting price. This occurs because one of our goal is to observe, as we said before, the auctioneer's satisfaction.

It will be displayed in the output window (called 'Satisfaction') and it is the difference between the auctioneer's reservation price and the hammer price.

In this type of auction, the bidders have the advantage of knowing a lot of information. Indeed, each of them can see other competitors, knowing who do the last bid and observing other behaviors. So, they can modify their reservation price according to the behavior of other bidders, during the model's running.

We put also a switch called 'Cooperation' in which the user could activate the cooperation between the bidders. If the switch is open, the cooperation is active. We create a variable G in the interface, setting by the user, which set the number of group of bidders. If the 'Cooperation' works one group does not make offers, except its leader.

In first section we explain the program, focusing our attention on the interface, in the first part, and on the codes in the second part. In section 2 we made some experiments modifying the parameters in the model. We interpret the results for their social - economic aspects. Section 3 concludes.

1. Program

Interface

In Figure 1 we report the interface of our model.

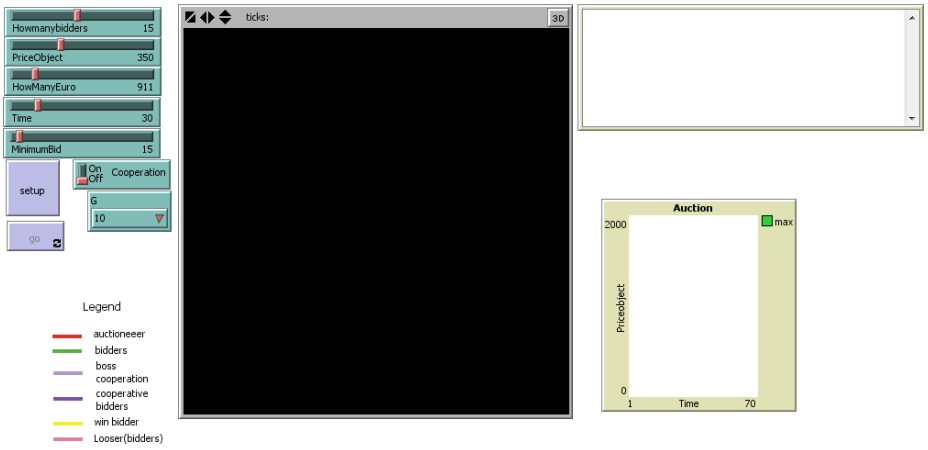

Figure 1: Interface

Now we explain all the part of the interface, to help the user to better interact with the model.

The large black space in the middle of the interface is the world: it is the place where the simulation takes place graphically. We better explain the graphical output of the model in the section 2, with the experiments.

The left part of the interface contain all the input and commands necessary to start the simulation. We have five sliders, one switch, two button and a chooser. We explain these in detail:

- slider 'Howmanybidders': it sets the number of bidders that participate in the auction;
- slider 'PriceObject': it sets the price of the object with a default minimum value of 50 €;
- slider 'HowManyEuro': it sets the maximum amount of € that a bidder could have;
- slider 'Time': it sets the maximum length of the auction;
- slider 'MinimumBid': it sets the minimum raise of each bid;
- switch 'Cooperation': it activates or not the option of cooperation;
- button 'setup': it launches the function setup that we will explain in the next paragraph;
- button 'go': it starts the function go, explained later;

- chooser 'G': it gives the number of group in which the bidders are divided to make cooperation.

Below the inputs there is a legend: it explains all the colors of the agents in the world.

The white space at the right in the Figure 1 is the output space. When the simulation is done, the results are reported in this part. We better understand the numerical output in the section 2, with the experiments.

Below the output space there is a graph which shows the evolution in time of the maximum bid (that is also the object price in each time).

Finally, we want to stress that in our model the flow of the time is monitored by a bidder's offer. The last tick is the period where no offers are made and in which the model chooses the winner bidder.

Functions and codes

In this paragraph we explain all the function and the mechanism of our model, mentioning extracts of codes.

First of all we define all the agents in the model as a whole and as an individual in the following way:

```
breed [auctioneers auctioneer]
breed [bidders bidder]
breed [objects object]
```

We define three agent - sets: the auctioneers, the bidders and the objects. The auctioneers are who sell the object in the auction; the bidders are who buy the object making offers and the third agent - set is simply the object.

In our model auctioneers and bidders have some features. Indeed the auctioneers have the reservation price, that is the price at which they would sell the object, and the satisfaction, that is the difference between the hammer price and the reservation price of an auctioneer, as follows:

$$\text{Satisfaction} = \text{hammer price} - \text{auctioneer's reservation price}$$

Therefore the bidders have the reservation price too, that is the maximum price at which they are willing to buy the object; they also are in a group, using for cooperation, and they have an amount of money in Euro.

Also the objects have their own feature: the price.

As regarding the code, these features are defined as follows:

```
auctioneers-own [ReservationPriceA Satisfaction]
bidders-own [ReservationPriceB  bid euro group]
objects-own [Priceob]
```

In the model we use four variables in more than one functions. They are called globals and they are the object price and other two variables that we explain in the next discussion:

```
globals [n K S Pricebid]
```

The first function in our model is the function *setup*. It creates the agents in the world and it fixes some features of the agents.

The main passages of the function are here reported:

```
set K  false

create-objects 1 [set size 1.5
                  set shape "flag"
                  setxy  16 15
                  set Priceob PriceObject
                  set label Priceob]

create-auctioneers 1 [set size 1.5
                      set shape "person"
                      set color red
                      set size 1.5
                      setxy 16 8
                      set ReservationPriceA PriceObject]

create-bidders Howmanybidders [set shape"person"
                               set color green
                               set size 1.5
     set ReservationPriceB (2 * PriceObject + 80 + random (200))
                               set euro 50 + random(HowManyEuro)
                               set group random(G)]

output-write "Starting Price  "
output-print  PriceObject
output-write "Maximum Time  "
output-print Time
```

The first line initializes the value of the indicator K as false; K is an indicator uses for stopping the model in two particular situation: when there are only one or two bidders that want to make offers for the object and when none bidders makes an offer (all the bidders have a reservation price less than the price of the object).

The remain lines of the function create 1 object, 1 auctioneer and a number of bidders equal to the value insert by the user in the interface. Each object is created with some features set in these lines common to all the agents. Indeed, the agents have the size (how big appears the agent in the interface world), the shape (the form of the object in the world), the color and the position in the interface (except for the bidders, which are disposed by rows in few lines of codes not reported here).

Therefore, each agent has also some typical features: the object has its price, reported graphically in the interface (using the command 'label'); the auctioneer has her reservation price (set equal to the price, as explained before) and finally the bidders have their reservation price (set as a random linear function of the price), their amount of Euro (set as a random integer between 0 and the maximum value fixed in the interface plus 50 €) and their group (set randomly).

Finally the last four rows of the code write in the output the starting price of the object and the value of the slider 'Time'.

The second main function of our model, directly accessible from the interface, is the function *go*. It is a function that refers to other functions, which are executed in the order given by the function *go*. The function is here reported:

```
if (all? bidders [color != yellow]) [tick
                                     cooperative-bidder
                                     make-bid
                                     loss
                                     stop-model
                                     ifelse(ticks >= Time or K = true)
                                      [win stop]
                                     [output-write "Current Price Object  "
                                      output-print Pricebid
                                      output-write "Number of ticks  "
                                      output-print ticks]]
```

First of all we see the loop 'if': this is necessary to improve the function *go* only when the auction is not finished, indeed when the auction finishes the winner is colored in yellow: so, if there is a yellow agent, there is a winner and the auction is concluded. Finally to each interaction between the agents, so to each time that the function *go* is executed, the counter 'ticks' are improved by 1.

There are six action in the function *go*, five of which are other function. The first action is the function *cooperative-bidder* that manages the cooperation, when the corresponding switch is on. the second function is the function *make-bid* that made the offers and the raises. The third function is the *loss*, that divides the bidders still in the game (that have enough money to raise the maximum bid and to buy the object) to the bidders out of the game (that have not enough money to raise the maximum bid). The forty function is *stop-model* that sets the index K true if the conditions of an anticipate end of the auction are respected. The last function *win* happens only when K is true or when the maximum duration of the auction is reached.

There is also a sequence of command that write in the output space in the interface the current object price (the maximum bid) and the number of ticks (interaction) in each step of the auction.

All the functions are seen in detail in the next analysis.

The function that manages the cooperation between the bidders is called *cooperative-bidder*. It is the first function to be executed in the command *go*. The code of this function is here reported and briefly discussed:

```
if (Cooperation) [let gg 0
                  let l count bidders with [group = gg]
                  while [l < 2 and gg < G]
                    [set gg gg + 1
                     set l count bidders with [group = gg]]
                  ifelse (l >= 2)
                    [let cbidder one-of bidders with [group = gg]
                     ask other bidders with [group = gg and
                                   who != [who] of cbidder]
                         [set color violet]
                     ask cbidder [set color violet + 3
                                  if(euro < max[bid] of bidders)
                                  [set ReservationPriceB PriceObject - 1]]
                  set Cooperation false]

                  [ output-print "Reduce G"
                    set Cooperation false]]
```

The entire function is an 'if' loop: it works only when the switch 'Cooperation' is on. The function sets two new variables, the first *gg* that spans all the groups, the variable *l* that checks if there is a group with at least two agents. If this group exists, the model takes randomly one bidders and it selects all the bidders in the same group; all these bidders has color violet and they does not make action in the auction: they are inactive bidders. The selected bidders is the leader of the group and she is the only bidders in that group to make offers; she is colored in light violet, to distinguish her from all the other bidders. Therefore the selected bidder makes an offer, raising the maximum actual offer by the minimum raise. After that the switch 'Cooperation' is set off.

The second function *make bid*. It is the function that makes the initial offer (the first bid) and all the further raises (the bids); finally this function raises the minimum reservation price of the bidders when the bids are raised for almost the minimum raise for three times consecutively. We report here the code of this function:

```
let firstbidder n-of 1 bidders with [ReservationPriceB > PriceObject
                                     and color != violet]

ask firstbidder [if (ticks = 1)
```

```
                [set bid (PriceObject + MinimumBid)
                 set Pricebid max[bid] of bidders
                 if (bid <= euro)
                    [set label bid
                     if (bid > ReservationPriceB)
                        [set bid ReservationPriceB set label bid]]]]

let abidder one-of bidders with [color != pink and
                                 bid != max[bid] of bidders and
                                 color != violet and
                                 ReservationPriceB > PriceObject]
ask abidder [if (ticks > 1)
             [set bid (floor ((0.02) * ReservationPriceB)+
                 (max[bid] of bidders +  MinimumBid))
              if (bid > ReservationPriceB)
                 [set bid 0
                  set label bid
                  set K true]
              set label bid
              set Pricebid max[bid] of bidders]]


let bbidder one-of bidders with [color != pink and color != violet]
ask bbidder [if (ReservationPriceB > PriceObject
                 and ReservationPriceB < euro)
                 [if (ticks mod 3 = 0)
                     [set ReservationPriceB
                       (ReservationPriceB + Pricebid )]]]
```

This code is divided in three parts: the first part mades only the first offer of the auction, the second part mades any other bids, the third part modifies the reservation price of bidders.

The first part takes randomly a bidder with reservation price greater than the object price and that do not take part in the cooperation except its boss. The function sets the bid of this bidder, which is also the first bid of the auction, equal to the price plus the minimum bid. This offer is made only if the bidders has enought money to support the offer and there is the condition of which, if the bidder has a reservation price less than the random offer, her bid is setted exactly equal to the reservation price.

The second part takes a bidder with the following features:

- she has enought money to raise the actual maximum bid, otherwise her color is pink;
- she is not the bidder that has the actual maximum bid (there is not

possible that a bidder raises her own bid);

- she is not in a cooperation situation (not boss), otherwise her color is violet and she does not make an offer;
- she has a reservation price greater than the object price, so she want to buy the object.

This bidder makes an offer raising the previous maximum bid by the minimum bid, set by users in the interface, plus a rate of 2% of the reservation price as increment. If her bid is greater than her reservation price the offer becomes equal to zero and the index k becomes true.

Finally the third part selects an agent with enought money to be in the game and out of any cooperation play. This bidder modifies her reservation price, if it is greater than the price and less than the money, adding to her reservation price the maximum actual offer in the auction.

The third function in *go* is the function *loss*, that checks if the bidders have used up all the money. The code is reported here:

```
ask bidders[if (euro < max[bid] of bidders
                and color != violet
                and color != violet + 3)
                 [set color pink]]
```

If the bidders have consumed all the Euro, their color is set pink. In the function *made-bid* the pink agents are excluded in make offers or raises, so the bidders without enought money do not partecipate any more at the auction.

The following function is the fourty action in the command *go* and it sets the index k equal to true if some conditions are respected. It is called *stop-model* and its code is:

```
set n count bidders with [color != pink
                          and color != violet ]

if (n <= 2  or all? bidders [bid = ReservationPriceB])
                            [ set K true ]
```

The counter n counts the number of bidders with enought money to bid (with color different from pink) and not in a cooperation group (color different from violet), then if these bidders are less than three the index k becomes true. When the k is true the model ends with the function *win*.

The last function in the command *go* is called *win* and defines the action to do when the models ends. The code is here reported:

```
ask bidders[if  (bid = max[bid] of bidders)
                 [set euro (euro - bid)
```

```
                    set color yellow
                    set label bid]]

ask auctioneers [set Satisfaction
                      (max [bid] of bidders - ReservationPriceA)
                   set S Satisfaction
                   output-write "Hammer Price "
                   output-print Pricebid
                   output-write "Satisfaction "
                   output-print Satisfaction
                   output-write "End time "
                   output-print ticks
                   ]
```

The first part of this function selects the bidder with the maximum offer which is colored in yellow, for a graphically result. Therefore the money of this bidder is decreased by the amount spent to buy the object, equal to the offer.

The second part of this function is reserved for the auctioneer. It is computed her satisfaction with the formula previously quote. Therefore her satisfaction and the hammer price are reported in the output window in the interface for each step of the auction.

Notice that if the boss of a cooperation group is the winner of the auction her color is change from light violet to yellow, then as results there is no more agents in light violet in the wordl.

Now we have seen all the code of all the function used in our model. The next section reports some experiment made by changing the parameters with some social - economic analysis.

2. Experiments

First of all we report the final aspect of the interface at the conclusion of one simulation. It is in Figure 2.

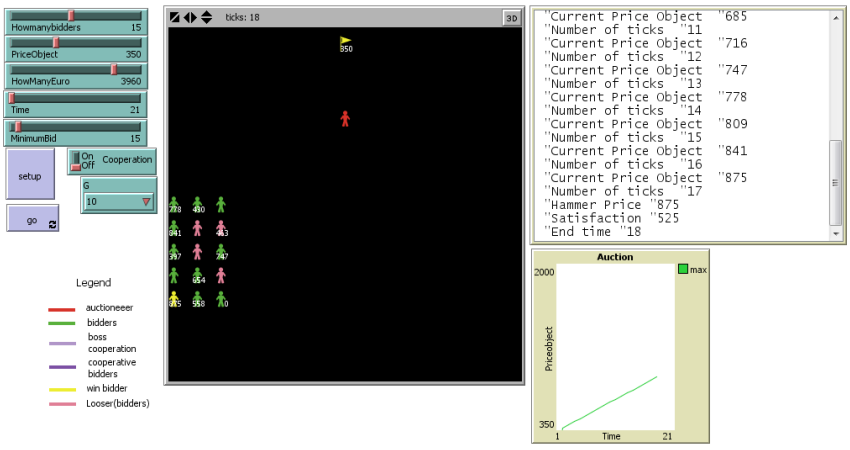

Figure 2: Interface

We make some experiments changing the values of the inputs in our model. In this way it is possible to clarify some interactions among the agents and the final results of the simulation. To do that we use the tools 'Behavior Space'. It is an instrument that combines different values of the indicated variables in many way, making each time the simulation and reporting the final results in an Excel file.

The codes in the 'Behavior Space' is here reported:

```
["Cooperation" false true]
["G" [2 2 10]]
["Time" 30]
["MinimumBid" 15]
["HowManyEuro" [811 100 1000]]
["Howmanybidders" [5 5 30]]
["PriceObject" [50 100 850]]
```

Each line is a variable that we want to modify and in the bracket there is the initial value, the step and the final value. In the following line we set the results that we want to check in the model (the hammer price, the k index and the satisfaction:

```
Pricebid
K
S
```

We report now some results of the 'Behavior Space' experiments:

```
[run number],"Cooperation","G","Time","MinimumBid","HowManyEuro",
"Howmanybidders","PriceObject","[step]","Pricebid","K","S"
```

```
2,"false","2","30","15","811","5","150","15","560","true","410"
3,"false","2","30","15","811","5","250","7","473","true","223"
4,"false","2","30","15","811","5","350","5","529","true","179"
```

As a result we report two relevant experiments: in the first we have the maximum number of bidders (30) with all the other sliders near to their minimum value and without cooperation; the second has the same price as before but an higher values of Euro and minimum bid.

As regarding the first experiment we attach here the final interface, in Figure 3.

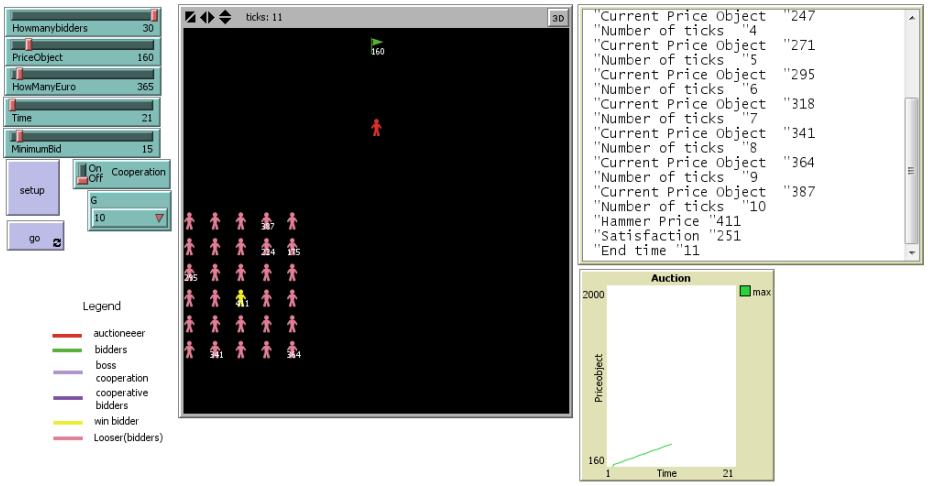

Figure 3: Interface

We notice that all the bidders become pink (not enough money) in few ticks, except one, or in same cases two, agents that have the higher financial means. Among these bidders there is the winner. Therefore, we note that the path of the maximum offer has a low slope but it is increasing anyway.

The second result regards the relation between the number of iteration and the reservation price of the bidders. We report the interface in Figure 4.

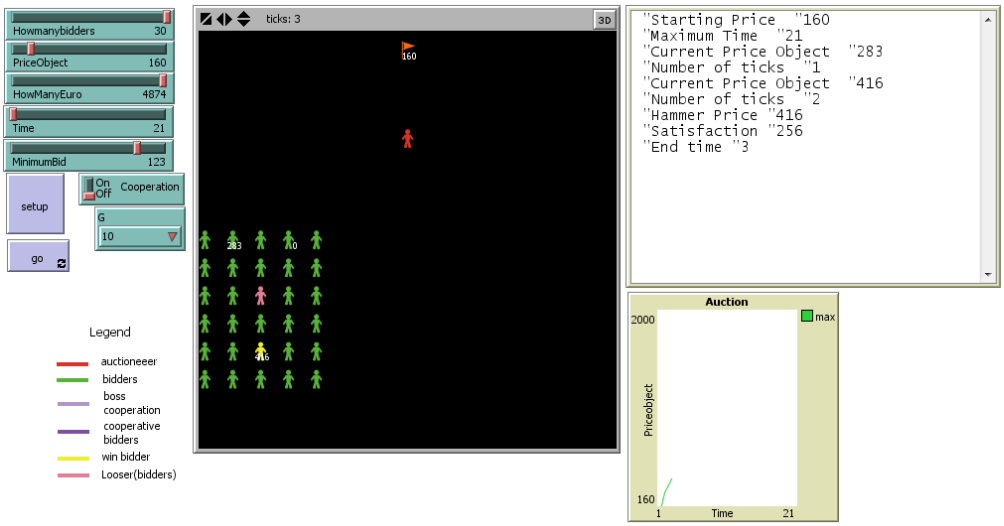

Figure 4: Interface

As you can see the model ends in few iteration despite the high financial means. This happens because the object price is relatively low respect to the money, so the reservation price, linked with the object price, becomes lower and the bidders are less interested into make an offer: smaller competition, smaller iterations.

Therefore, the slope of the hammer price path is higher than before: this happens because there is higher financial means.

Finally we note the following relations:

- reduce the number of bidders has the same results of reducing the money, with all the other variables constant: there is a reduction in the number of iterations;
- if the cooperation is on there is the same results as reducing the number of bidders.

3. Conclusions

We think that this model is a good representation of a real English auction. It has a great variety of inputs, that could be modify by the users; this makes our model very available.

Furthermore, it is possible to analyze the effect of a cooperation among bidders, dishonest behavior usually forbidden and punished.

In the end, the 'Behavior Space' lets users to make an high number of experiments in few minutes.