Hotelling model of spatial competition: a NetLogo agent-based simulation

Lorenzo Gambino

Simulation models for economics

a.y. 2015-2016

Introduction

The aim of the work is to simulate, using the software NetLogo, the interaction among buyers and sellers in a single good oligopolistic market. This is the typical theoretical framework of the Bertrand competition model, that is price competition among a small number of firms in an otherwise competitive market. In details, the purpose is to assess the predictions of the first theory devised for the description of positional interaction between producers, the Hotelling model[1], formulated by Harold Hotelling, an American statistician, in 1929.

The main intuition underlining Hotelling work is the evidence that competition in the real world take place in a geographical space (that can take different dimensions); this is likely to introduce additional (with respect to the price of the good itself) costs for the consumers in the market. These transport costs can be easily thought of as an element of (horizontal) differentiation among the goods sold; for every consumer in the market, buying from a vendor or from another may imply a different overall expenditure. Therefore the good homogeneity assumption of the neo-classical competition analysis breaks down. What this imply for firms' behaviour was not clear until Hotelling step in.

In the original model formulation, a large number of consumers is spread along a unit long segment; the two sellers are at the extreme of the line. For every buyer the overall cost is given by the sum of the price set simultaneously by the producers plus the transport costs, a function of the consumer placement. Assuming that every consumer want to buy at least one unit of good, the only relevant choice is from which firms: overall cost minimization lead to the optimal outcome. It is then easy to show that, even if a seller set a higher price with respect to the other she will not lose all her customers (the Bertrand model result): for a given number of consumers it is still optimal to buy the good from the costlier producers, because reaching the other seller will imply very high transport costs. There is, therefore, a degree of market power and a positive mark-up in this framework, both functions of the firms' differentiation decisions.

Using very simple game theory instruments Hotelling showed that two are the drivers of firm placement and price decisions: direct effect and strategic effect. The first imply that, all things equal, the nearest is one firm to the other the greater will be its demand (and profit): for a large share of consumers, it will be optimal to buy from the firm. The second effect becomes relevant when we allow the price to change and to be simultaneously determined by sellers: the price chosen by a firm is a function of the competitor position (like in every strategic interaction model, other agents behaviour influence decision-making process). The shorter the distance between firms the stronger the incentive to steal market share by reducing price, leading to a fiercer price competition. This twofold overall effect will determine the optimal firms' placement. By looking at market characteristics, it is therefore possible to predict the likelier outcome in term of price and placement, determining the dominant effect between direct and strategic drivers and therefore the optimal position firms will choose in order to maximize their profit. This is the most important contribution of Hotelling work to the field of industrial organization.

[1] For a complete description of Hotelling's work see L.Cabral, Economia Industriale, Carocci Editore or J.Tirole, The Theory of Industrial Organization, MIT Press.

Will this theoretical framework be consistent with an agent-based simulation? Will the interaction of artificial buyers and sellers match the Hotelling predictions? This work will try to answer to this kind of questions and to assess the real usefulness of agent-based simulation in modelling economic behaviour.

NetLogo will allow us to study this problem from a graphical point of view, comparing the outcomes of different versions of the simulation, corresponding to different hypothesis on agents' behaviour. For the sake of the experiment, we will use a silly but meaningful example: competition among drinks sellers in a city park. Given the fact that every seller offers the same kind of good the competition is twofold, in terms of placement and selling price, precisely as in the Hotelling framework. The city park users are the potential customers of the simulation: they will walk around searching for a fresh drink to be bought from the cheapest seller. Moving around is not free and its cost must be included in the buying process: although every vendor sells the same good, this is not homogeneous, transport costs must be taken into account. It is worth to spend a couple of words clarifying the assumptions that underpin city park users' behaviour: there are no budget constraints, willingness to buy is taken for granted, and information is incomplete in terms of both market prices and sellers' placement. Even if not all these were included in Hotelling model original formulation, their inclusion make the simulation much more interesting and allow focusing on agents' interactive learning and information collection (this is one of the most relevant advantage of agent-based simulation). The same line of reasoning holds true for drinks sellers with some relevant differences: there are no capacities constraint (they will limit the competition), marginal cost is very small (actually equal to zero) and constant, information is limited both with respect to competitors' prices/placement and to customers' position. The sellers' objective function is the maximization of market share.

In order to clearly understand every variables effect, we will start from a simplified version of the model gradually increasing the degree of complexity by including new elements in the simulation. In the first version of the model, we can assume the sellers' behaviour as fixed focusing our attention only on consumers' movements and buying decisions, in order to study the effect of (random) initial placement on firms' market share and profits. This is a useful step but clearly not sufficient for our work purpose; ruling out sellers' reactions to market conditions does not allow to check the theoretical insights of the Hotelling model. In order to do so we will introduce sellers' pricing and movement decisions, obtaining a more complex and meaningful model.

In the following section we will describe the simulation in detail, explaining both the code and the line of reasoning behind them; we will also comment and compare the result of the simulation, with the original Hotelling model as a milestone for comparison. The conclusion deal with the very problem at the core of our analysis: we will try to response to the questions previously formulated about the consistency of Hotelling model theoretical prediction and the role agent-based simulation play in explaining economic behaviour.

The code and its explanation[2]

One of most useful feature of the NetLogo programming language is that it allow to simulate complex interactions among different sets of agents in a graphical intuitive way; agent's behaviour can be modelled using the program's user-friendly language and simulation's outcomes can be studied using 2D/3D representation and 2D graphs. It is therefore the right tool for building a simple Hotelling's model simulation.
The very first step in creating a new NetLogo model[3] is to generate of the classes of agents needed for our purpose; this is exactly the rationale of the "breed" command.

```
breed [rational_buyers rational_buyer]
breed [lazy_buyers lazy_buyer]
breed [sellers seller]
breed [trees tree]
```

Since NetLogo agents are turtles, all the program's syntax is written in "biological" terms; to generate new type of "turtles" we must "breed" them. In our case the sets of agents required for the simulation are three (or four): two different kind of buyers (the difference between the two will be clear later on), sellers and an optional environment-improving one (the city park trees of our silly example). Each set of agents has its own general name and an identification for single class member: this make possible assigning different portion of code to different type of agent, or even to a specific agent.
The second step is the assignment of the simulation's variable to the agents; it is possible to assign different list of attributes to the different classes of agents using the "-own" command.

```
rational_buyers-own [lazy initial_xposition initial_yposition my_seller purchases pantry
last_seller last_seller_xposition last_seller_yposition movement_cost overall_price
last_price last_movement_cost]
lazy_buyers-own [lazy initial_xposition initial_yposition my_seller purchases pantry
last_seller last_seller_xposition last_seller_yposition movement_cost overall_price
last_price last_movement_cost]
sellers-own [price number_of_customers last_sales sales overall_sales missed_sales
profits]
globals [buyers]
```

Our two different type of consumers share the same list of variable: "lazy" is a true-false attribute that model that willingness to search for better price when a vendor is first encountered, the buyer starting position is stored in the "initial_xposition"/"initial_yposition", "my_seller" check if a

[2] The NetLogo version used is the 5.2.1 release; older version may present a slightly different syntax.
[3] For a full explanation of the NetLogo programming language and its syntax and command, check the official website at http://www.ccl.northwestern.edu/netlogo/; a complete documentation along with tutorials give a thorough overview on the program functioning.

purchase has been made in the current cycle, “movement_cost” tracks the cost of buyer movement in the simulation area and “overall-price” the actual price of a purchase. The other attributes are related to the buying process and allow to store in memory the number of goods bought and kept in the buyer’s pantry, the existence, price and position of the last seller with which a transaction has taken place. The sellers set the “price”, track the “number_of_customers”, count “sales” in the current cycle and from the starting date (“overall_sales”), keep memory of the last cycle final sales, keep records of “missed_sales”, that is the sales lost due to too high price from a buyer point of view. The definition of a global variable “buyers” will be explained later on.
After the definition of the different classes of agents, the actual creation take place; this is usually done using a portion of code identified with the “setup” name and run with the proper button in the graphical interface.

```
to setup
 clear-all
 setup-patches
 setup-sellers
 setup-trees
 create-and-merge-buyers
 setup-buyers
 reset-ticks
end
```

A “clear-all” command remove all the previous simulation outcomes, whereas the “reset-ticks” reset the time counter. A list of actions to be executed before the actual start is included; these are the lines of code that “physically” create our agents. For a section of the program to be run on call, it must be generated using a “to” introductory command and a final “end” instruction; the code in between collect all the tasks the simulation will execute when run.

```
to setup-patches
 ask patches [ set pcolor green ]
end


to setup-trees
 create-trees random 10 [setxy random-xcor random-ycor
                set shape "tree"
                set size 3]
end
```

“Setup-patches” modify the existing patches of the simulation area assigning them the green colour. The optional “setup-trees” command create a random number of non-interacting agents,

setting their position randomly, and shape and size properly. It is worth to outline that the “ask” code, that assign a list of instructions to an agent, can be used only with already existing agents; this is not the case of our city park trees, whereas it is for simulation area’s patches. Before “ask” something to someone we must first “create” her.
The following command allow us to create and differentiate the two class of buyers.

```
to create-and-merge-buyers
 create-rational_buyers number_of_rational_buyers
 [ set lazy false
  set color yellow ]
 create-lazy_buyers number_of_lazy_buyers
 [ set lazy true
  set color pink ]
 set buyers (turtle-set rational_buyers lazy_buyers)
end
```

The number of buyers to be generated is chosen via the two sliders in the interface section: it is possible to change freely the composition of our market demand to assess the effects on the simulation outcomes. The “lazy” attribute is set false for the rational class of consumers and viceversa; also the colour of the two types of agents is different, it is therefore easier to spot them graphically. Since we need differentiation only in the creation process, and, starting from this point onwards, buyers will behave in almost the same way, we will assign them the same list of commands; in order to reduce the amount of code we merge the two sets together in the global variable “buyers” via “turtle-set”.
Now we assign the starting value to consumers’ variables.

```
to setup-buyers
 ask buyers
 [set shape "person"
  set initial_xposition pxcor
  set initial_yposition pycor
  set my_seller false
  set last_seller false
  set movement_cost 0
  set overall_price 0
  set last_price 1000
  set purchases 0
  set pantry 0
  if random_buyers_position = true[
```

```
    setxy random-xcor random-ycor]
  ]
end
```

For most variables the starting value is obvious (no movements, no transactions have taken place) but it is worth to spend a couple of words on “last price”: we set an arbitrary high price in order to avoid initial cycle paradox. Also the initial position worth a discussion: the “random_buyers_position” slider allow to choose whether buyers should start the simulation in the origin (the lower left corner) or in a random position. As we will see later on this choice shapes dramatically the simulation’s outcomes.
Now we need to create and characterize our sellers.

```
to setup-sellers
  create-sellers 3
  [set shape "truck"
   set size 2
   set price 1 + random 9
   set number_of_customers 0
set last_sales 0
   set sales 0
   set overall_sales 0
   set missed_sales 0
   set profits 0
  ]
  ask sellers [ setxy random-xcor random-ycor
              set heading towardsxy 0 0 ]
  ask seller 0 [set color red
    set label "I'm the crazy one"]
  ask seller 1 [set color black]
  ask seller 2 [set color gray]
end
```

The number of sellers is fixed for operational simplicity, this is likely not to affect very much the results (see the following section for further discussion). Most of the variables take an obvious starting value (including the shape!), except for price and position. The price is randomly determined in a reasonable range: since no sellers know what her competitor are doing, she set the starting price with a reasonable guess. Also the positioning is a random decision, with the main difference of a degree of knowledge regarding the buyers starting point: when no random

consumers positioning is implemented the sellers know from where the buyers come from (they set their heading towards the origin). For the sake of differentiation different colours are assigned to the sellers and we introduce also a non-rational producer ("seller 0"), which will behave randomly through all the simulation.
After the end of the creation process, it is time to explain the core of the program, where interaction is designed and implemented and agents behave accordingly through time. All this instruction all collected in the "go" part of the code.

```
to go
 if ticks >= how_long_the_simulation [ stop ]
 restart
 price_adjustment
 buy
 move_buyers
 tick
end
```

After a simple condition on the end of the simulation (after a chosen number of iteration), the four model's building blocks are introduced. The first is the "restart" part, the one that reset parts of the simulation after a given number of ticks has been reached.

```
to restart
   if ticks > 0 and remainder ticks restarting_value = 0
            [ask buyers
              [setxy initial_xposition initial_yposition
               set my_seller false
               set purchases 0
               set pantry (pantry - (1 + random 1))
               set movement_cost 0
               set overall_price 0
              ]
If sellers_movement = true[
              ask seller 0 [
               set label "I'm the crazy one"
               right random 360
               forward random 10
              ]
```

```
            ask seller 1 [ if number_of_customers < count buyers and distance patch 0 0
>= 5 [
              if random_sellers_movement = true [
                right random 360]
              forward repositioning_speed ]
            ]
            ask seller 2 [ if number_of_customers < count buyers and distance patch 0 0
>= 5 [
              if random_sellers_movement = true [
                right random 360]
              forward repositioning_speed ]
            ]
]
            ask sellers
               [set last_sales sales
                set sales 0
                set number_of_customers 0
                set missed_sales 0]
          ]
end
```

Starting from the observation that the consumers' buying process does not take place continuously over time, but only once in a given time interval, and the same line of reasoning applies to firm placement decisions, we introduce a set of actions to be executed only every chosen number of ticks. This time interval can be thought of as a working day, week or even longer time period depending on the market to be simulated. When the logical time condition is verified the buyers return to their starting position (that was registered at the beginning), reset movement cost and price, deplete their inventory (that is consumption take place) and a new transaction is therefore needed. This is the right moment for the sellers to change their position (we must allow them to do so by setting the "seller_movement" variable to true), if the previous one was not satisfactory (from a market share or profits point of view). The movement can be random or rational according to the degree of information we are willing to assign to the producers (the "random_sellers_movement" switch serves our purpose). The speed of the adjustment can be chosen from the usual slider. All this does not apply to the irrational seller whose decision-making process is completely random. After that, all temporary variables indexing sales and customers are set equal to zero.

The next step is the price revision decision: If this section of the program is enabled ("price_adjustment" switch), every specified number of ticks sellers change the price of the good sold. The process does not take place in the "restart" section since price can be changed more

often and more easily with respect to market position; it seems therefore more realistic to separate the two decisions.

```
to price_adjustment
 if price_change = true and ticks > 0 and remainder ticks price_adjustment_speed = 0[
 ask seller 0[
  set price 1 + random 9]
 ask seller 1[
  ifelse (missed_sales >= (sales / 10) or missed_sales >= (last_sales / 20)) and price > 1[
   set price (price  - 1)]
  [set price (price + 1)]
  ]
 ask seller 2[
  ifelse (missed_sales >= (sales / 10) or missed_sales >= (last_sales / 20)) and price > 1[
   set price (price - 1)]
  [set price (price + 1)]
  ]
 ]
end
```

As before seller 0 behave randomly; the other two sellers instead, modify the price looking at the amount of potential sales lost due to excessive price tag. When this quantity exceed a specified proportion of the current or last cycle sales, price is revised downwards (if it is not already near zero); the opposite happens when the proportion is relatively low.
The next step is the modelling of consumers buying decisions.

```
to buy
 ask buyers [if my_seller = false
  [ let nearest_seller one-of sellers in-radius 2
   if nearest_seller != nobody[
    set overall_price (overall_price + [price] of nearest_seller)
    ifelse lazy = true or (lazy = not true and (overall_price <= (last_price *
price_elasticity) or pantry <= 0))[
      set my_seller true
      set last_seller true
      set purchases 2
      set pantry (pantry + purchases)
```

```
      set last_price overall_price
      set last_movement_cost movement_cost
      set last_seller_xposition [xcor] of nearest_seller
      set last_seller_yposition [ycor] of nearest_seller
      ask nearest_seller[
        set number_of_customers (number_of_customers + 1)
        set sales (sales + 2)
        set overall_sales (overall_sales + 2)
        set profits (profits + 2 * price)
        set label sales]
    ]
    [ask nearest_seller[
       set missed_sales (missed_sales + 2)]
    ]
    ]
  ]
 ]
end
```

Each buyer check whether a transaction as already taken place in the cycle; if it is not the case she looks around searching for a potential seller: this is defined with the "let" command by the temporary (it exists only in this portion of code) variable "nearest_seller". If a producer (or even more than one) is found, the consumer checks the seller's price, taking into account also the previous transportation costs. Now the buyers' differentiation kicks in: "lazy" buyer will buy whatever the "overall_price", they do not want to search further for better conditions (remember that no income-constraint are imposed). Rational buyers will compare the price with the last price at which they have bought, and only if price has not increased too much they will complete the transaction (the "price_elasticity" variable capture this price increases tolerance). Transaction can even take place at un-economic conditions if the "pantry" is empty: consumption must take place every cycle and all agents want to consume. This decision process is reproduced using a series of (nested) logical conditions ("and"/"or"operators); when one (or more) of this condition is true the next section of code is executed. If the purchase take place and both the buyer and seller update information regarding the last seller. Since we have used an "ifelse" structure, when no purchase take place the number of seller's missed sales increases.
The last set of instructions is related to buyers' movement.

```
to move_buyers
 ask buyers [ if my_seller = false
```

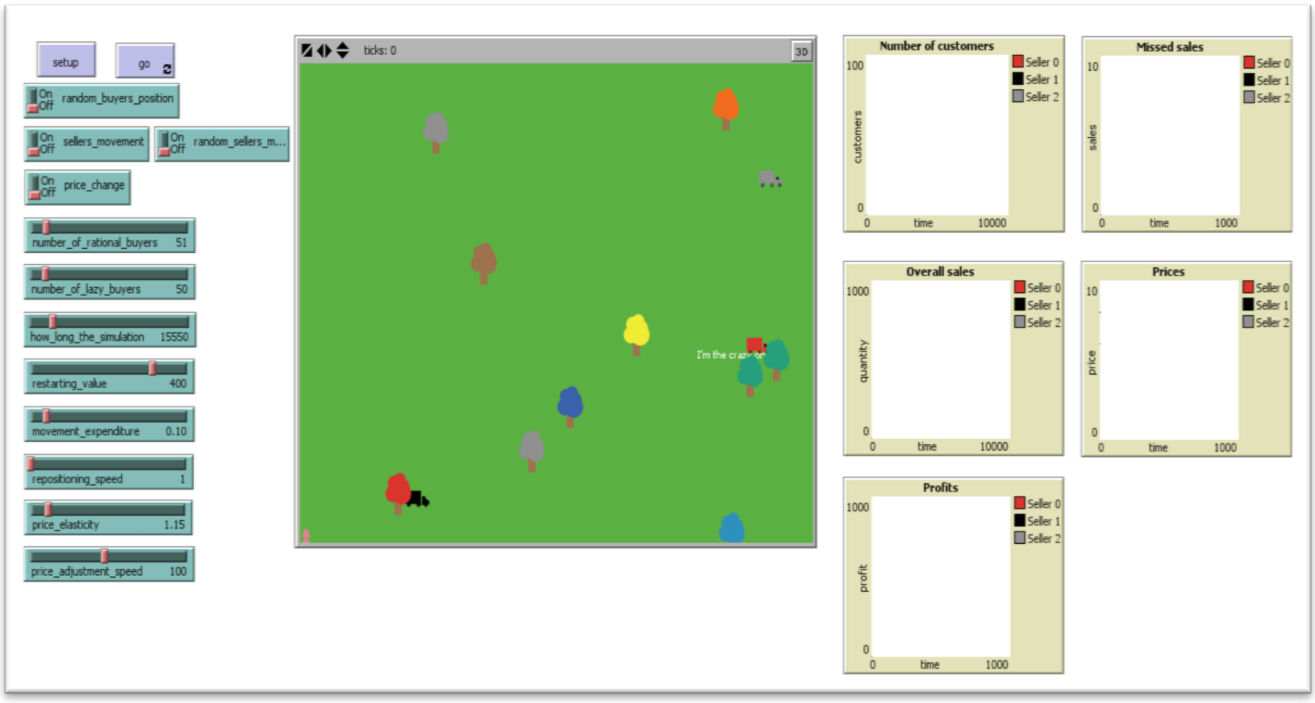
```
        [ ifelse last_seller = true and xcor < last_seller_xposition and ycor <
last_seller_yposition [
            set heading towardsxy last_seller_xposition last_seller_yposition
            forward 1
              set movement_cost (movement_cost + movement_expenditure)
              set overall_price movement_cost]
           [right random 360
            forward 1
            set movement_cost (movement_cost + movement_expenditure)
            set overall_price movement_cost]
        ]
 ]
end
```

Consumers move in search of a potential seller only when no transaction has taken place in the cycle. The type of movement depends on the existence and position of the last seller: while is still convenient ("movement_cost" lower than "last_movement_cost") to move in the direction of the last seller, buyers will try to reach her. When the stored information becomes irrelevant (or no previous transaction has taken place), movement follow a random path simulating the consumers' search process.

The combination of all this portion of code allow to simulate, under different hypothesis, the demand-supply interaction in a Hotelling-like market. In the following section we assess the simulation outcomes, comparing them to the Hotelling theory predictions, in order to find an answers to the very first questions we asked in the introduction.

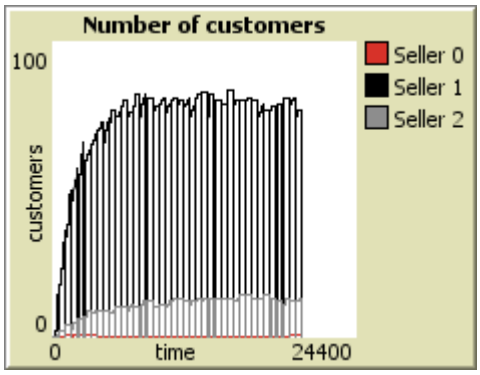
The simulation

Before introducing the simulation results let's take a look at the interface section of the program, displaying the starting settings.



On the left we can find the interactive tools: the "setup" and "go" buttons, which run the code, and the switchers and sliders that allow the user to easily modify the program; we can choose which section of the program we want to run and also the value of the main inputs. The centre is entirely occupied by the graphical display, in which agents movements and interactions are reproduced. On the right we find the graphs, showing the dynamics of the model's main variables.

The first simulation[4] we run is the most simple possible: no sellers' actions will take place. Price and placement decisions are therefore excluded; the starting value of sellers' variables is kept constant until the simulation's end. We will focus on consumers' behaviour, movement and buying decisions, and how these are related to the (random) starting position of our sellers. Since, before any interactions take place, market size and composition are unknown, the producers will set their price and position as a reasonable guess (this is likely to happen when a firm first enter in the market and does not have sufficient information on the market demand and characteristics).

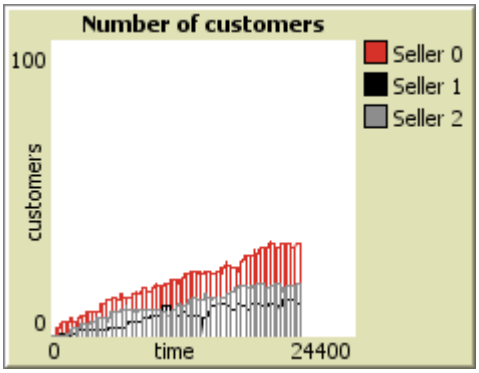


1.Market share of a firm (black) located near the origin, with distant competitor

The (not so surprising) outcome is that the firm located near the origin grab the lion' share of the market, selling to all the "lazy" buyers in every cycle and collecting also a relevant proportion of the "rational" ones.

[4] All the different simulation are run for 20000 ticks, with 100 buyers evenly divided among "lazy" and "rational", and the restarting value is fixed at 200 ticks. When prices are allowed to be mo ticks and price elasticity is 1.15.

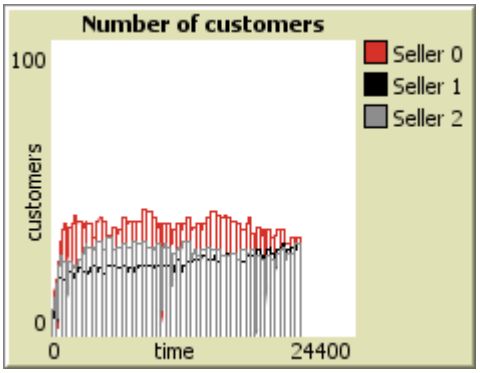
This results is stronger the greater the competitors' distance; even the "rational" consumers will end up buying form the nearest seller, whatever her price, given the enormous transportation cost needed in order to reach a different firm and the need to replace depleting stocks of good.



2.Market shares when all firms are distant from the origin

Results are not so clear when no producers is located near the buyers' starting point. As before the more efficiently located, that is the one near the origin will acquire the higher number of customers; however it will be only a small fraction of the overall market, even if the cycle duration is increased (here things improve a bit but again only a small fraction of buyers reach a potential seller: this is due to very high movement costs needed in order to reach a seller position and to a lack time to be devoted to searching). What is really interesting to point out is the importance, from a producer point of view, of selling to "lazy" customers: since they will buy always from the first firm they will encounter, they are high fidelity buyers and therefore represent the bulk of a firm's customers base. Capturing them grants a consistent amount of profits in time, even when a competitor is more efficiently located.

Outcomes are less straightforward when we allow the buyers to be randomly distributed in the simulation area; here there is no "optimal" starting place for our sellers, even firms at great distance from the origin can attract a relevant share of customers. Interestingly, what we observe is a more or less even division of the market: each seller consistently grab a portion of the overall sales, whatever her starting price. Market shares are even more equally split when the distance between producers is relatively low and producers are located near the simulation space centre. It is interesting to point out that market shares seem to be not so strongly related to prices: in contrast to what one may expect lower starting price does not necessarily lead to a bigger customers pool. It is the position (or differentiation) that drives market outcomes. If a seller is able to differentiate herself from the other, without moving too far from the city park centre, she will be able to conquer a relevant share of the overall market.

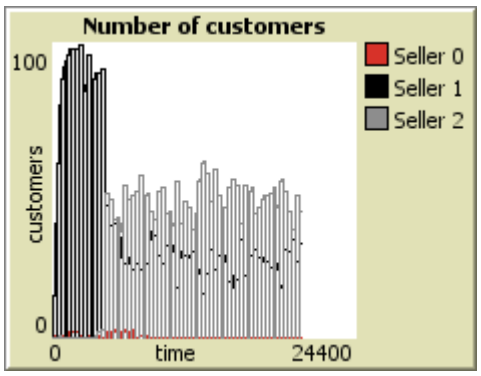


3.An example of perfect market division

So far our analysis seems to be quite consistent with Hotelling model's predictions: in a market with a small number of firms, with homogeneous good and positive transport cost, the more a seller is located near a high concentration of buyers the better will be her performance. That is, a successful differentiation decision implies locating near the greater number of customers, while in the same time, maximizing the distance from the other competitors; in this case the firm enjoys a significant degree of market power. But since all producers share the same objective, information and degree of rationality they will end up locating closer one to each other, trying to capture the greatest possible market share (in the absence of price competition). This is at least a reasonable predictions based on the Hotelling theory insights. We can try to test it empirically by allow our sellers agents to freely move in the simulation area.

In the first iterations buyers will start again from the origin, the upper left corner of the simulation area. More importantly we assign a relevant degree of knowledge to our two rational sellers: they are fully informed on the potential customers' starting position and on the existence of transportation costs. Their optimal strategy will be to reduce as much as possible the distance separating them from the origin, reducing buyers movement cost (this is relevant only for "rational" buyers), and reaching first the "lazy" ones. As seen in the code explanation, the third firm will behave randomly, acting a control agent for the rational sellers.

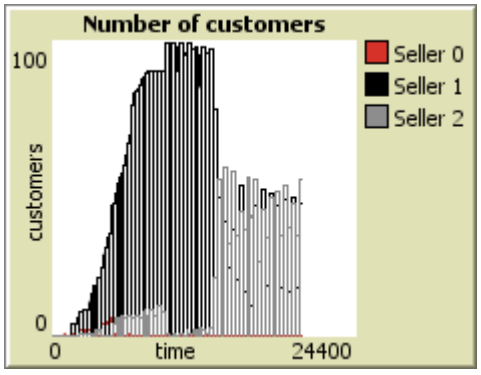
What we observe consistently through all the repetitions of the program is the division of the market between the two rational sellers: since from their starting position they are unable to serve all the market's customers, they will try to reach as soon as possible the buyers' starting location. There is a clear advantage for the firms located near the origin; she will reach more rapidly the optimal degree of differentiation and therefore capture a significant market share. Prices are not so important as long as firms are moving; they become relevant when both rational sellers are located near the origin. The one who originally set a lower price will attract a higher number of customers: "rational" buyers will choose the cheapest firm. The crazy seller will soon be kicked out of market, so we can argue that, in this particular setting, rationality matters.



4.The market share of seller 1 start to decline as soon as her competitor (grey) reach the optimal position: a lower price helps in gaining market shares

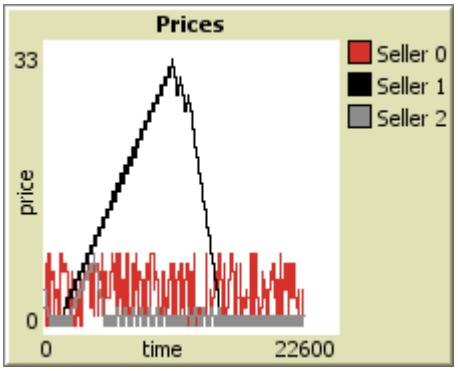
When we introduce random buyer's position (and therefore a lower amount of information for our sellers) outcomes change a lot: the sellers will move around trying serve the greater amount possible of customers but no clear winner can be found. In a way this setting's results resemble the ones from the no sellers movement situation; the market shares are more or less evenly split among the producers, prices seem to be not so relevant in determining market outcomes, and no equilibrium can be found. Our irrational seller 0 displays a highier degree of volatily in sales as a result of less predictable movement: she covers a greater distance with respect to her competitor in each repositioning step, and ends up losing (or gaining) a relevant number of customers, mainly "lazy" ones. Great repositioning seems to be a risky strategy: it can lead to very high, but also low sales and therefore profits.

In final version of the model we finally introduce price revisions, trying to replicate all market mechanism; as before we will first locate all the consumers in an initial point and then randomly distribute them through the simulation area. In the first case firms will end up positioning in the optimal place that as mentioned before: as close as possible to the lower left corner. The market is split between the efficient producers, and the irrational one is soon excluded. The outcome is therefore quite similar with respect to the previous setting; what is completely different is the prices dynamic. The firm located near the origin has the clear advantage of reaching more easily the bulk of demand, and can therefore charge a high price for her goods, without losing too many customers. In the first cycles she



5.The incumbent enjoys an almost unchallenged market power, that ends only when the new entrant catch up charging a lower price

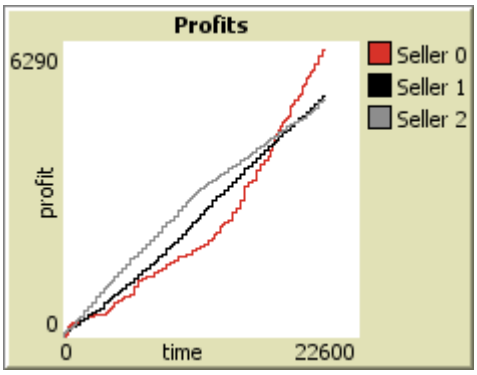
can therefore make very huge profits, with a monopoly like power. Things change dramatically when the newcomer catch up: since she was in desperate need of customers, she set the price at the minimum possible level, just above zero price level. When the entrant finally reach the optimal position she charge a consistently lower price with respect to the incumbent, and she is able to lure away not only the "rational buyers" (due to the lower price) but also a fraction of the "lazy" ones. The only possible answers from the other firm point of view is to reduce price accordingly, trying to keep customers in front of aggressive price competition. So prices will stay in the range lower bound, quite close to the firms' marginal cost (here assumed to be zero). This results seems quite realistic: the only viable strategy for the laggard is to reduce the price in order to gain a customers base. Differentiation is not possible since there is only one optimal position that minimize overall movement costs. In the short run there is a clear advantage for the incumbent; very high price will lead to huge profits.



6.The incumbent is able to charge very high price but is exposed to competitor under cutting price. In the long run prices will be equal to marginal costs

This advantage however, is highly dependent on the proportion and movements of "rational" buyers; even if a seller is quite unchallenged by competitors, she can be constrained in increasing price by a relevant amount of missed sales. The consumers' willingness to accept price increases (a concept quite similar to demand elasticity) will shape firm behaviour and reduce (or increase) the degree of market power enjoyed by the incumbent (and by all firms in general). "Rational buyers" discipline the firms' behaviour, constraining price increases and promoting a fierce price competition in the long run, increasing consumers' overall welfare.

Things are surprisingly similar when random positioning is introduced. Since no optimal position exists in this setting, sellers are quite differentiated and will cover different sections of the simulation area. They will end up with quite similar market share and sales. One can expect that this different positions will imply prices significantly different from their lower level (this is the actual prediction of Hotelling model) but this is not the case: firms will again set low prices, trying to lure customers and increase market share, even if they are not informed on competitors' behaviour. This price strategy allows the "crazy" producer to gain significant profits: she is not constrained by the market share maximization, and can therefore charge randomly high prices, boosting considerably revenues and profits. This result is likely the result of widespread movement across the simulation area, which allow the firm to find in every cycle potential buyers, even if the price is relatively high, thanks to the high degree of differentiation among the different firms.



7. The "crazy" seller consistently outperform competitors in a random movement setting

Hotelling vs ABS

Given the previous section results what can we say about Hotelling model and agent based simulation? The first clear result is that, in the presence of buyers movement cost, horizontal differentiation play a relevant role in shaping market structure and outcomes. Even if in our experiments consumers were not uniformly distributed along a straight line (as in the Hotelling model), firms' placement decision play a relevant role. They have a clear to incentive position themselves as close as possible to a high concentration of buyers, in order to obtain a significant market share. This seems to be consistent with the theory first insight that predicts the even division of market shares among firms, as a results of producers' placement in the line's centre. The observation that transport costs play an important role in the buying process is enhanced by the simulation's results. Distance are important and they must be properly taken into account. Is the Hotelling theory the right model for this purpose? Let's look at another example.

When a random distribution is imposed the results differs considerably: firms perform better if they are quite differentiated with respect to the others. Since is physically impossible to cover all the market, specialization pays of and firms ends up presiding a specific market sector. Even if movement is allowed things do not change much. The high degree of differentiation allow the firms to charge prices well above their minimum level; this was one of the most important predictions of the Hotelling work, one of the major contribution in overcoming the traditional Bertrand paradox of price competition in an oligopoly market.

Quite surprisingly, the introduction of a price revision mechanism does not seem to alter too much the simulation outcomes, in terms of market shares and sales. What is really interesting is the price dynamic per se: whatever the demand distribution, rational sellers will try attract customers by reducing prices towards their lower bound (the marginal costs, here imposed equal to zero). This strategy is quite effective, and consistent with the theory predictions (at least in the low differentiation scenario); when an optimal position exists and sellers are endowed with a certain degree of information about the market characteristics competition will push price down. Since there is no differentiation the only way to gain additional customers is by reducing price. The direct effect outweigh the strategic one and the results are similar to the Bertrand paradox and oligopoly price competition outcomes.

However, this strategy is much less effective when random buyers' distribution is imposed; prices should be higher than marginal costs, because of the high degree of horizontal differentiation and the relevant market power this situation grants, but this is not the case. We even observe the "crazy" seller consistently achieving greater profits with respect to her competitors; this is quite inconsistent with theory predictions and general economic reasoning.

The results are therefore mixed and it is worth searching for the possible sources of inconsistency in our program specification. The main issue is related to the very notion of computer agent based simulation: since building a computer simulation implies write down a specific code, it gives the writer a great degree of arbitrary decision-making. The agents' behaviour is the results of specific hypothesis formulated when the program was designed: different starting assumptions may lead to different outcomes, even greatly diverse ones. Results must therefore be assessed in the light of the designer choices.

In this specific situation our hypothesis on the agents' movement, buying and price previsioning decision were reasonable but not the less arbitrary; they work well (i.e. realistic) in certain settings

and worse in others, affecting experiment's outcomes. Different hypothesis may have led to completely different results. What about introducing a different objective function, profit maximization or buyers' turnover minimization? Also price revision mechanism, placement decision and buying processes could have been designed in a different way. We also have not modelled a realistic firms' cost structure: marginal cost was negligible, no capacity constraints were introduced and repositioning was free. This is hardly the case in real markets.
It is also worth to outline the fact that, despite having simulate buyers-sellers interaction in an oligopoly market, we have not introduced strategic interaction (at least direct): this is an always used behavioural hypothesis in industrial organization studies, like the spatial competition ones. The vast majority of the recent study on this topic use game theory tools and modelling to reproduced firms' behaviour in non-competitive markets; all this kind of literature is based on the assumption that producers in a market sector have some variable amount of information on competitors' strategies. This hypothesis can be quite reasonable in certain market and much less viable in other ones; we choose the latter scenario but simulate strategic interaction surely worth a try. This way of modelling agents' decision process is likely to affect profoundly every simulation results and perhaps will be included in future Hotelling based simulation studies.

Conclusion

The Agent Based Simulation approach is a very powerful tool both for natural sciences and social ones. In economics studies in particular, it allows the users to simulate and replicate market agents' behaviours and interactions, testing model predictions and hypothesis in a controlled and scientific way; this is even more useful in given the high cost and obstacles in performing laboratory test in economics. NetLogo is a great programming language to be used for the purpose, since it allows to build easily very complex models and structure. As we have seen no straightforward general conclusions can be drawn in our case, but it was not our purpose, since we were mainly interested in recreating a Hotelling-like situation and its dynamic in time. A thorough assessment can perhaps be the subject of further studies.