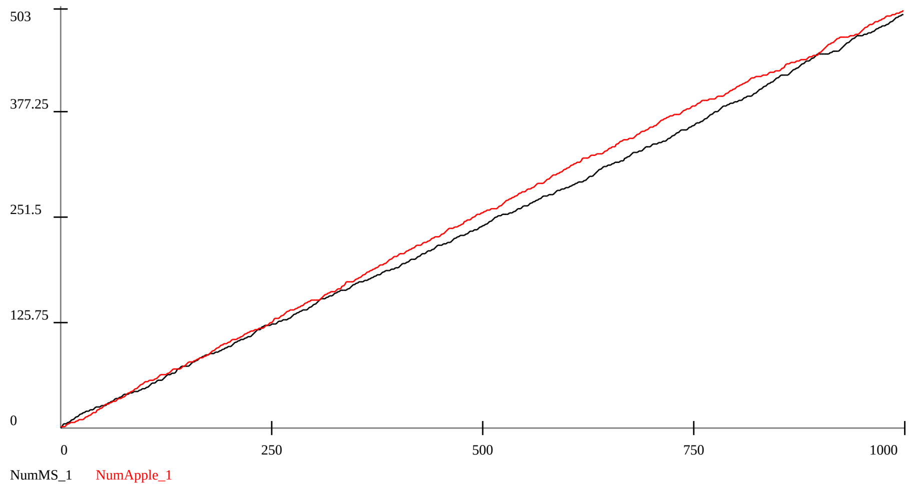
Studying B.Arthur's model

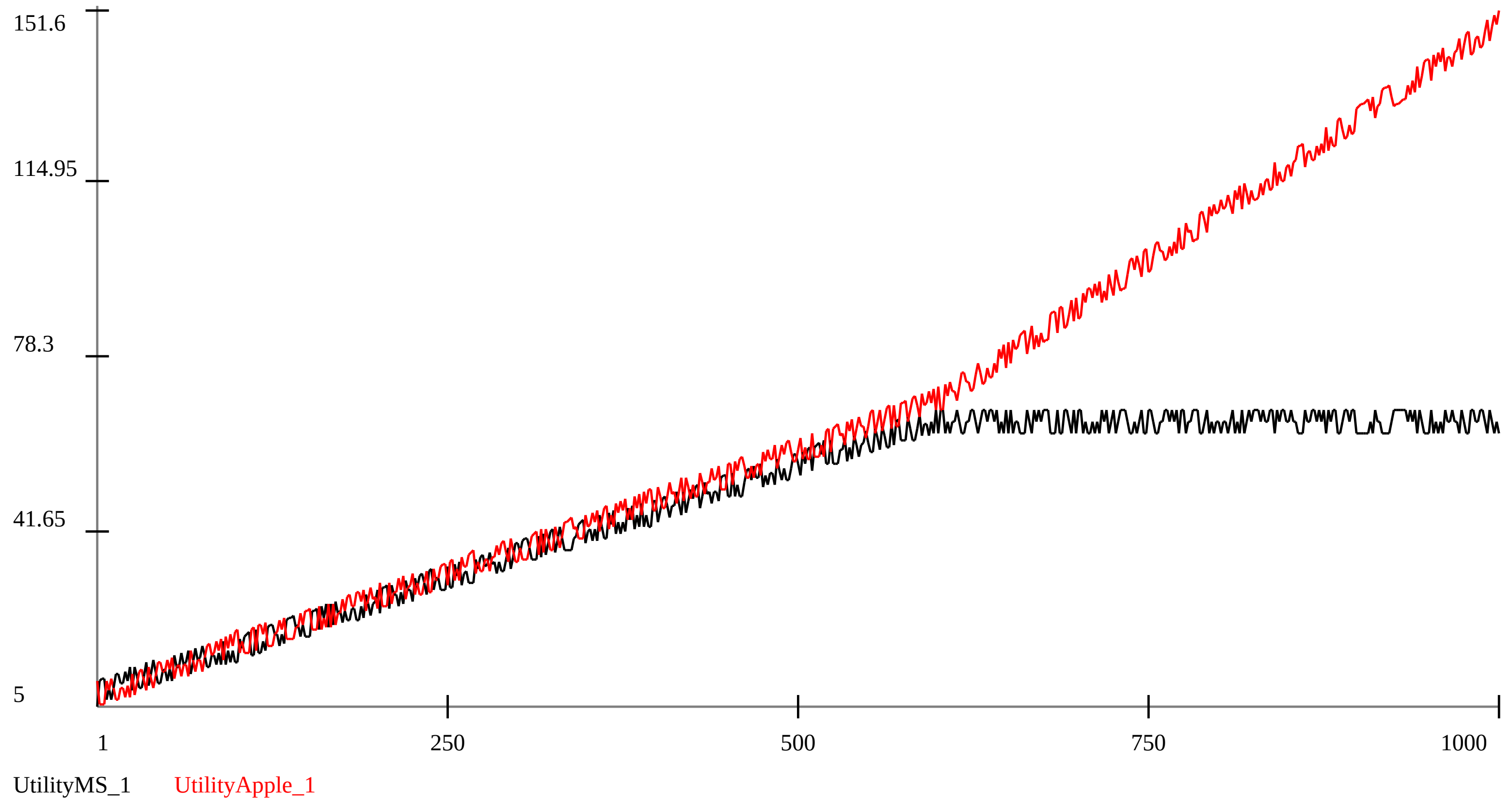
We have two types of agents, each with a *a priori* preference for a product. Therefore, without newtwork externalities, we would have that each the `NumMS` and `NumApple` would be identical to the number of users of the two types. The graph below is an example with the coefficients `User0Net`=0 and `User1Net`=0.



Studying B.Arthur's model

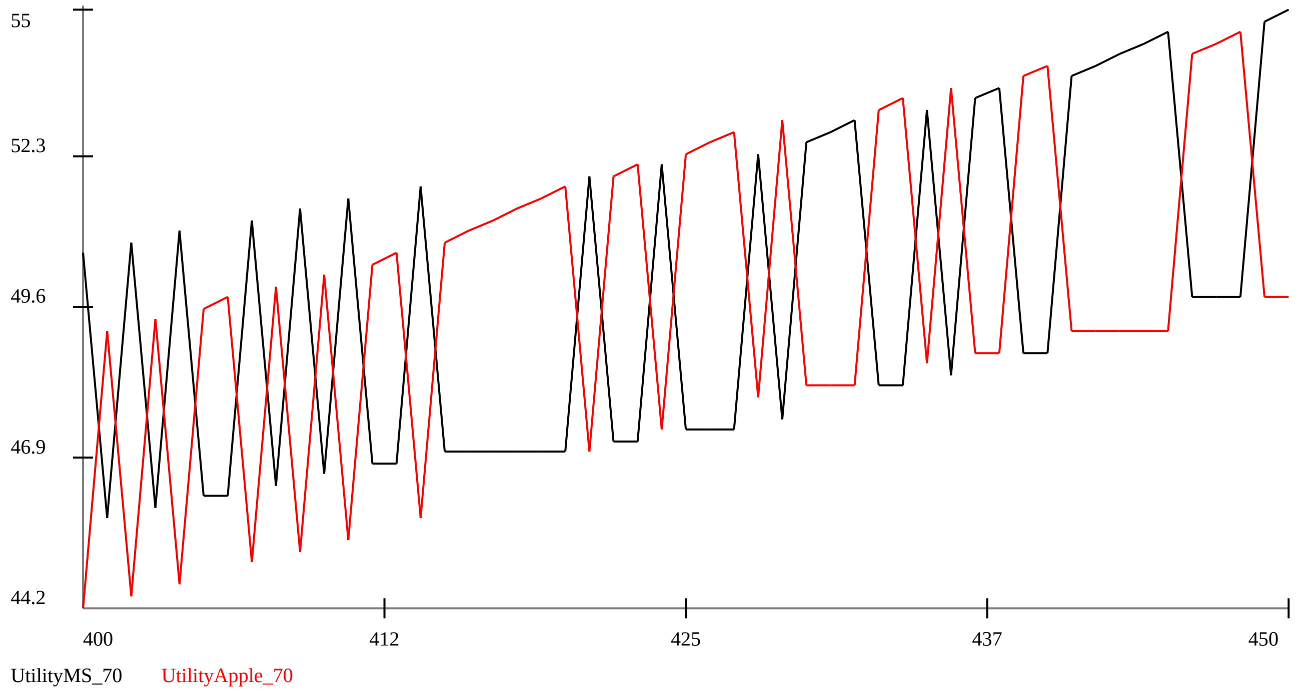
With the existence of network externalities, the utilities change as a function of the number of users for the two products. They are oscillating because different types of agents (0 or 1) prefer one product or the other, MS or Apple. But it is possible that, for random reasons, there is a series of agents of one type in sequence. This "temporary" majority increases the number of users for one product. At a certain moment, even the other users have the utility with the "wrong" product higher than that with the right product. At this point, the winning product cannot but increase its appeal with both types of consumers.

Studying B.Arthur's model



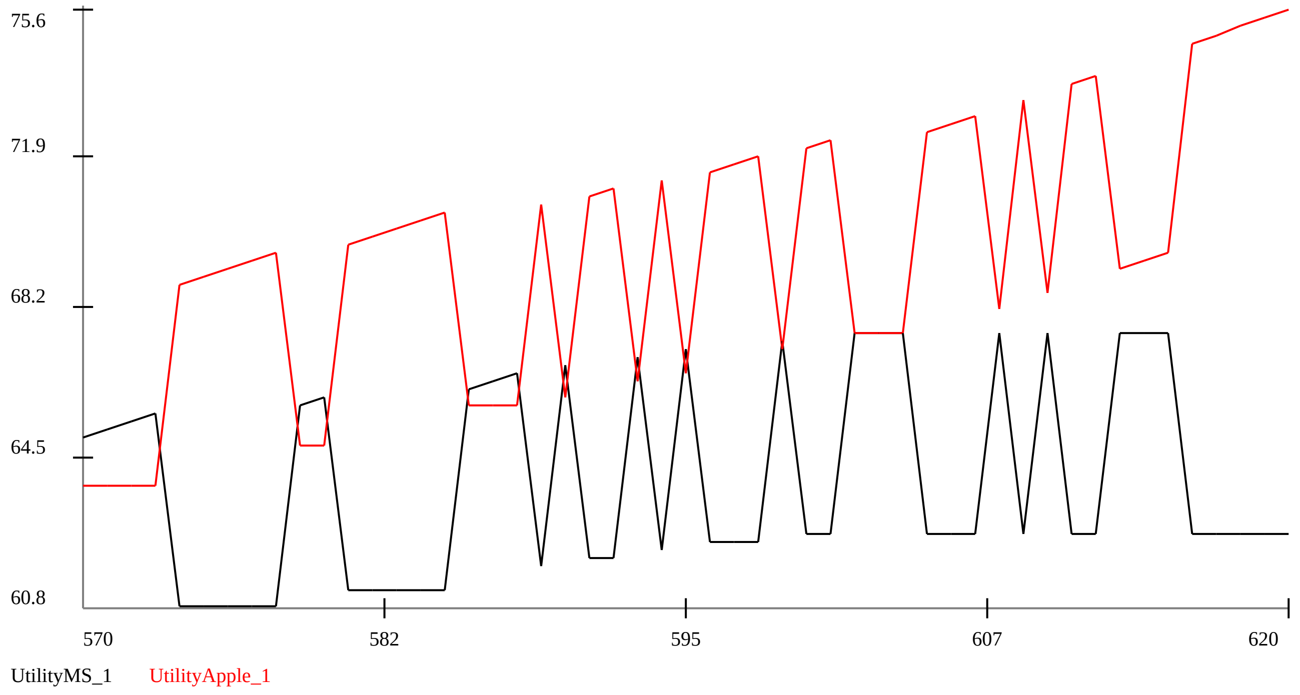
Studying B.Arthur's model

In general, utilities oscillate showing a higher value for one type of agent and a lower value for the other type.



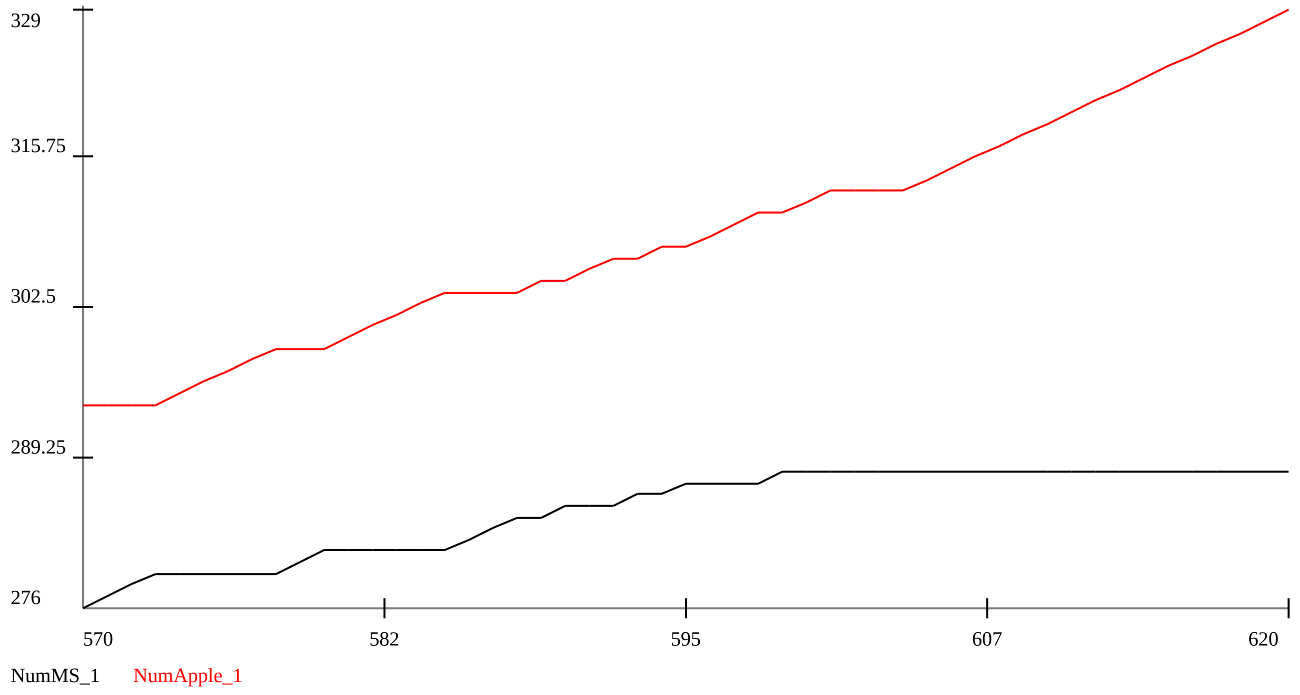
Studying B.Arthur's model

But just at the moment when too many consumers are placed in one product, than the two series "split".



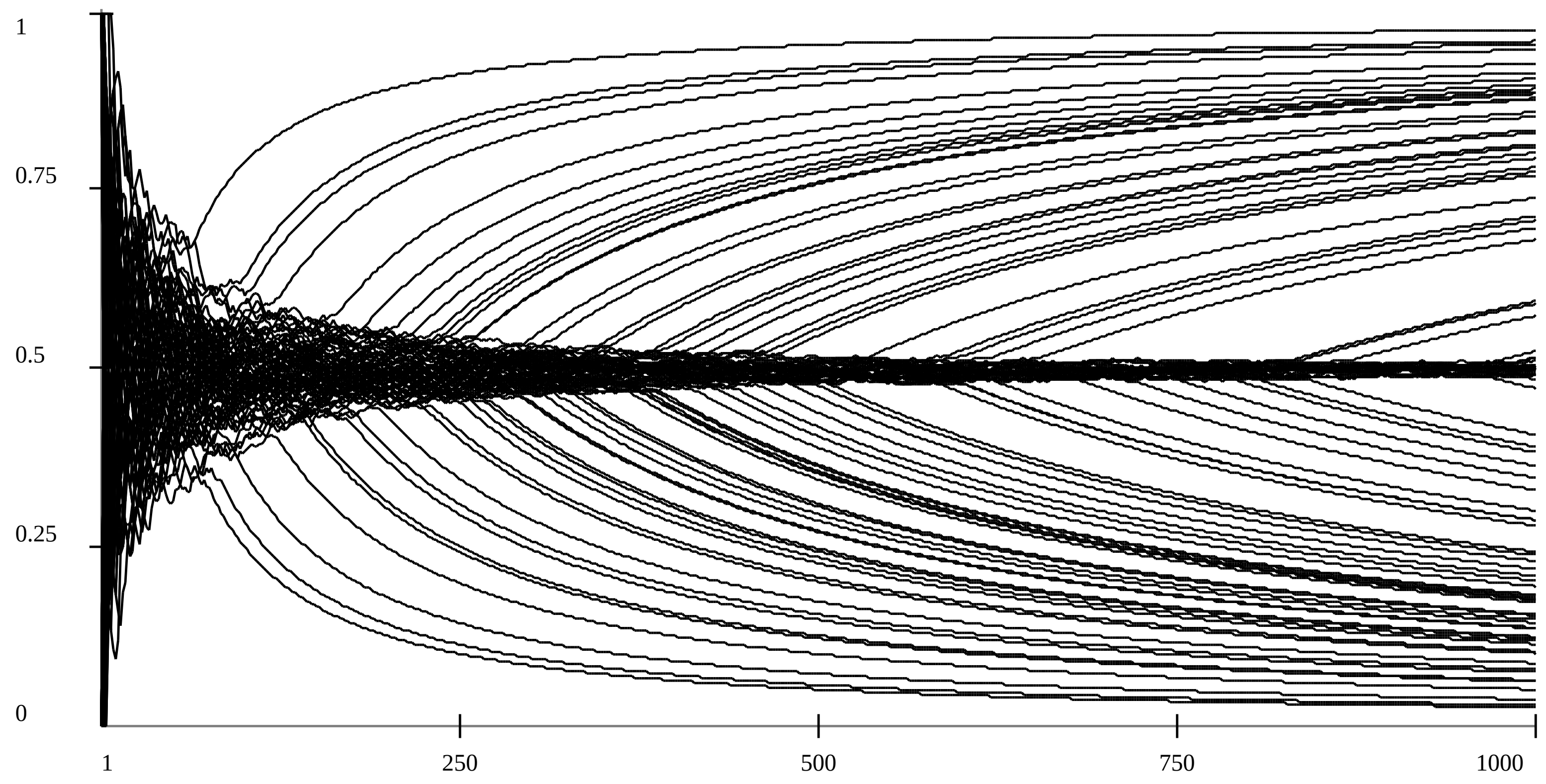
Studying B.Arthur's model

Consequently, the number of users of the "loosing" product stops, while the other increases at a double speed.



Studying B.Arthur's model

Another aspect is why some markets concentrate faster than others. That is, a product becomes dominant when, for random reasons, there are more agents of one type in respect of the other type. But why markets concentrating earlier are also faster, while market concentrating later are slower?


1
0.75
0.5
0.25
0
1
250
500
750
1000

Studying B.Arthur's model

The reason is purely algebraic. At time step t there are exactly t consumers in total. Therefore, adding one consumer modifies the share of consumers of $\frac{N_i}{t}$ of the value $\frac{1}{t+1}$. The larger is t the smaller is the increment, and therefore the slower is the increase of the share.

Studying B.Arthur's model

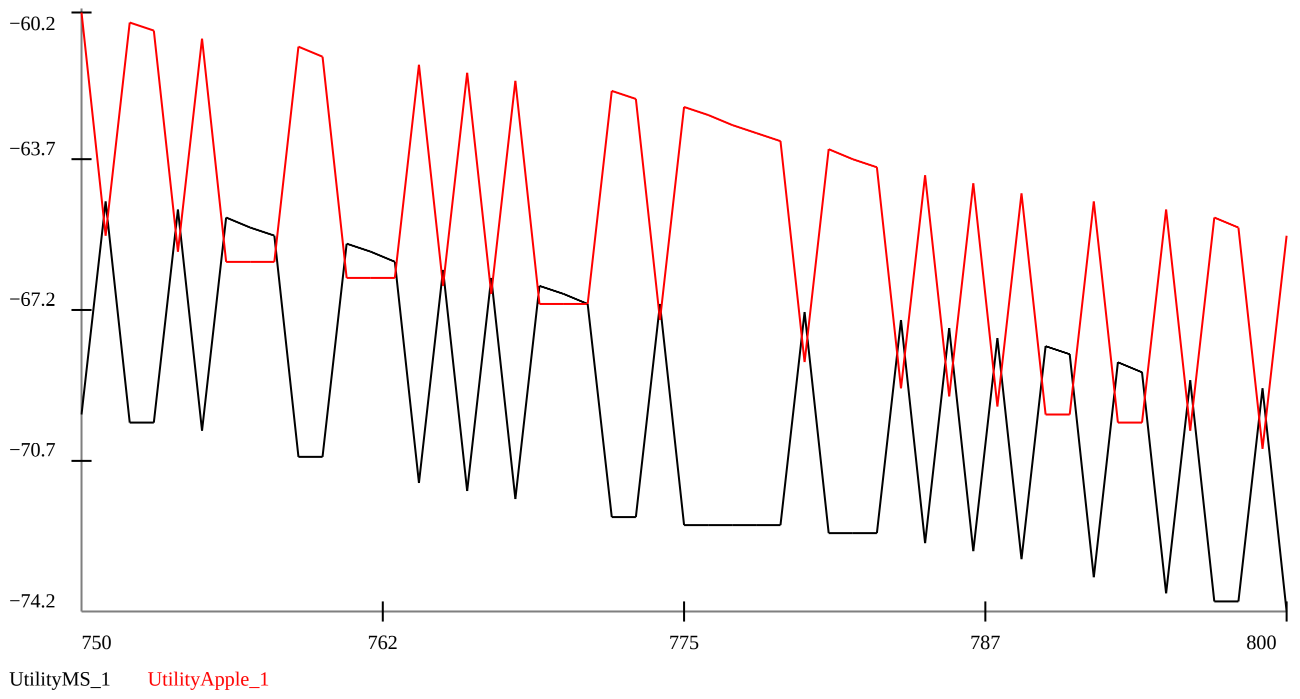
When there are not network externalities the number of products sold depends on the type of consumers, which is random. What about when there are *negative* externalities?

In this case, the *more* consumer choose a given product, the *lower* is the utility. Therefore, as soon as one type of consumer is larger than the other, for random reasons, the less attractive becomes its reference product.

Utilities decrease constantly, and decrease more the one for the product with a larger number of consumers.

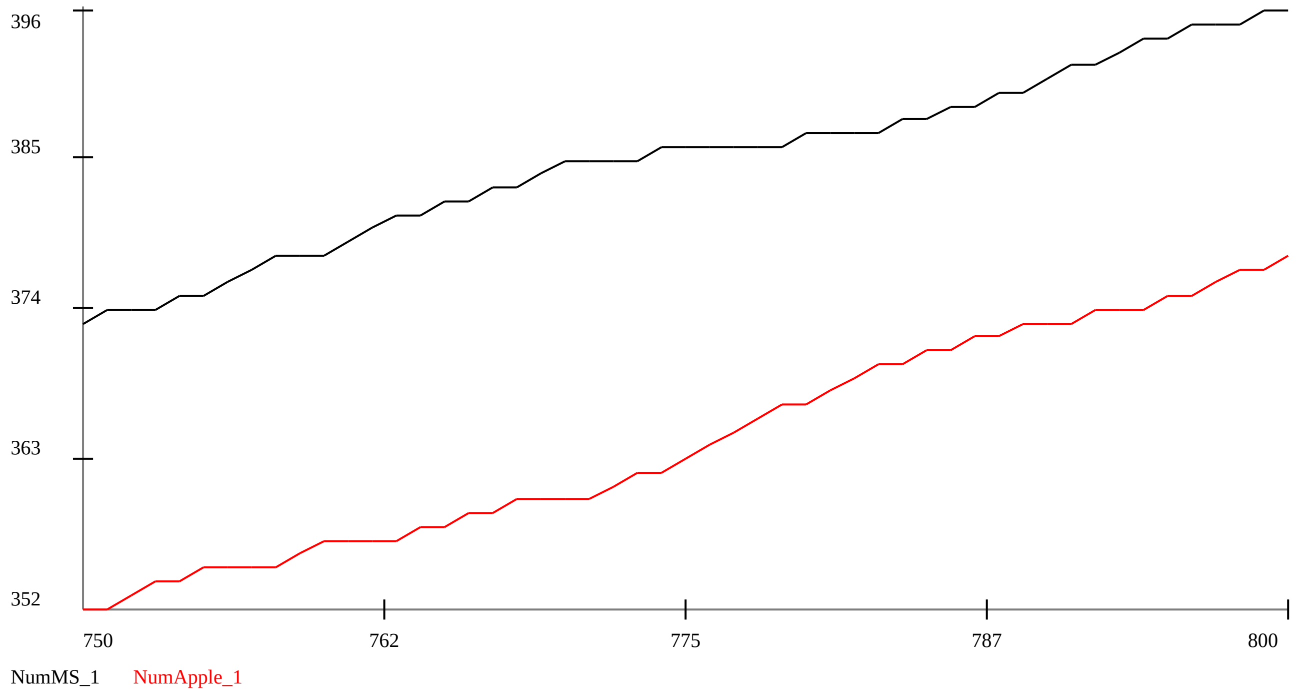
Studying B.Arthur's model

Negative externalities: utilities.



Studying B.Arthur's model

Negative externalities: number of consumers.



Modelling fashion

Many, if not most, of the decisions of consumers in industrialized countries are motivated by fashion. A very important aspect of many consumption purchases are determined by the fashion, that is, how people "interpret" a person by means of their clothes, car, tie, mobile phone, etc.

A group of economists worked on the analysis of this behaviour. We are going to build a model considering two types of consumers:

- **Type 0**: the **rich** consumers. They aim at getting as similar as possible to the fellow type 0 consumers and as different as possible to the type 1 consumers.
- **Type 1**: the **poor** consumers. They simply want to get as similar as possible to the majority of type 0 consumers.

Modelling fashion

Let's see how to model these behaviours. We assume that there are two types of product, as usual, 0 and 1. Let's consider that in a population we have:

- N_0^R: the number of *rich* consumers currently owning product 0;
- N_1^R: the number of *rich* consumers currently owning product 1;
- N_0^P: the number of *poor* consumers currently owning product 0;
- N_1^P: the number of *poor* consumers currently owning product 1;
- N^R: the number of *rich* consumers ($N^R = N_0^R + N_1^R$);
- N^P: the number of *poor* consumers ($N^P = N_0^P + N_1^P$);

Modelling rich consumers

We assume that from time to time a consumer needs to buy a new product, and must therefore decide whether to buy product 0 or product 1. Rich consumer want the product that is most popular among rich consumers, and least popular among poor consumers. Therefore, rich consumer choose product 0 if:

$$\frac{N_0^R}{N^R} > \frac{N_0^P}{N^P}$$

and choose product 1 if:

$$\frac{N_0^R}{N^R} < \frac{N_0^P}{N^P}$$

Modelling poor consumers

Poor consumers look only at rich's behaviour. They buy product 0 if:

$$N_0^R > N_1^R$$

and choose product 1 if:

$$N_0^R < N_1^R$$

Model configuration

Let's see how to implement the model. We consider an object **Population** containing the summary values:

`NRich`, `NRich0`, `NRich1`, `NPoor`, `NPoor0`, `NPoor1`.

All these are variables with 1 lag. Moreover, the **Population** contains also the probability of making a purchase, a parameter `ProbAction`.

Then, we have an object **Agent** contained in **Population**. This object contains: `Type(P)` and `Status(1)`. The `Type` is a parameter fixed in the beginning, being 0 for Rich and 1 for Poor agents. `Status` is a variable, with a lag, assuming value 0 for product 0 and value 1 for product 1.

Model configuration

The central equation is the one for variable **Status**. Let's see the code. Firstly, the equation returns the previous **Status** with a given probability, otherwise make the computation.

```
EQUATION("Status")
/* Determine the status of this agent */
v[0]=VS(p->up, "ProbAction");
if(RND>v[0])
 v[1]=VL("Status",1); //don't change
else
 {//do change, if you want
```

(continue ...)

Then, collects the values relevant for the decision, and reads the type of the agent.

```
v[2]=VL("NRich",1);
v[3]=VL("NRich0",1);
v[4]=VL("NRich1",1);
v[5]=VL("NPoor",1);
v[6]=VL("NPoor0",1);
v[7]=VL("NPoor1",1); //collect data
v[10]=V("Type"); //type of the agent
```

(continue ...)

Makes the comparison between two values, different depending on the type of agent.

```
if(v[10]==0)
 {//type RICH
 v[8]=(v[3]-v[4])/v[2]; //share of rich having 0
 v[9]=(v[6]-v[7])/v[5]; //share of poor having 0
 }
else
 {//type POOR
 v[8]=v[3]; //number of rich having 0
 v[9]=v[4]; //number of rich having 1
 }
if(v[8]>v[9])
 v[1]=0;
else
 v[1]=1;
```

(continue ...)

Finally, resolve possible ties, choosing randomly, and return the result.

```
  if(v[8]==v[9]) //in a tie, choose randomly
   {
   if(RND<0.5)
    v[1]=0;
   else
    v[1]=1;
   }

 }
RESULT(v[1] )
```

Equations for Fashion

The other equations are pretty obvious. For example, the equation for `NRich0` is:

```
EQUATION("NRich0")
/*
Number of rich people with status 0
*/
v[0]=0;
CYCLE(cur, "Agent")
 {
  v[2]=VS(cur,"Type");
  v[3]=VS(cur,"Status");
  if(v[2]==0 && v[3]==0) //if it is a rich agent and the status is 0
    v[0]++;
 }
RESULT(v[0] )
```

Equations for Fashion

Write the equivalent equations for `NRich`, `NRich1`, `NPoor`, `NPoor0`, `NPoor1`.

Moreover, write the equations for `Share0` and `Share1` expressing the share of users of product 0 and 1, disregarding the type of consumer:

```
EQUATION("Share0")
/*
Number of agents with status 0
*/
v[0]=V("NRich0");
v[1]=V("NRich");
v[3]=V("NPoor0");
v[4]=V("NPoor");
RESULT((v[0]+v[3])/(v[1]+v[4]) )
```

Initial values for Fashion

Create 200 agents. Initialize the `Status(1)` with 0 or 1 randomly (in Set All use **Random Integer**).

Initialize the `Type` to 0 for the first 100 agent and to 1 for the remaining 100 (in Set All use **Selected Cases**).

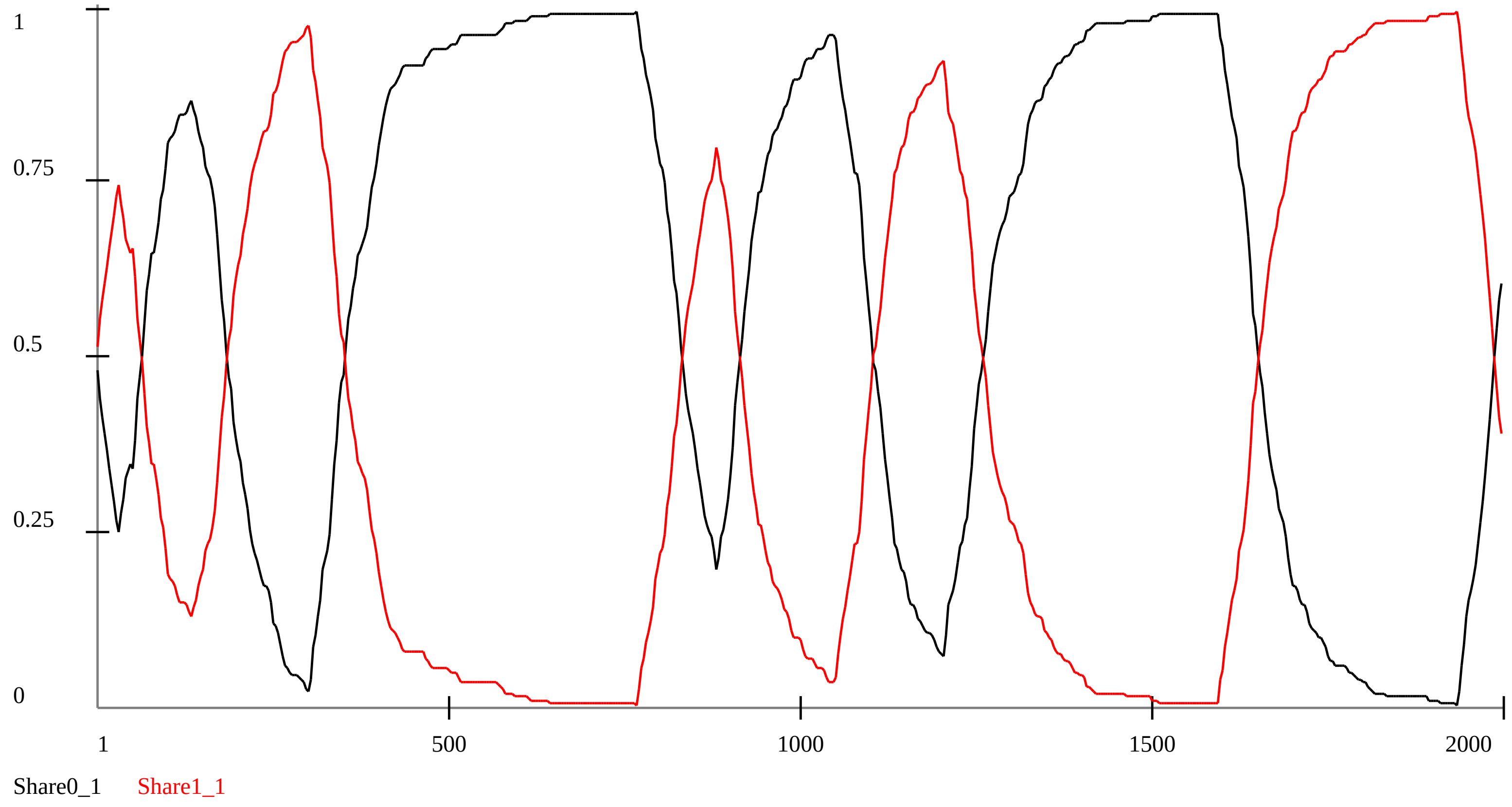
Set `NRich`=100, `NRich0`=50, `NRich1`=50, `NPoor`=100, `NPoor0`=50, `NPoor1`=50.

Set `ProbAction`=0.02.

Set to save all variables in **Population**. Set for **Run Time Plot** only the **Share0** and **Share1**.

Run 2000 time steps.

Fashion waves


Share0_1
Share1_1