Report on Netlogo project:

Happiness from Consumerism or Altruism

MENARDI LUCA

1 - Introduction to model

I start the essay on my Netlogo project with a very general introduction about the basic framework of the model, delaying the deeper comment on the specific structure at successive stages.

The simulation aims to analyze the relationships among agents, specifically I want examine the links among level of happiness of each agents with level of his own income and that of the others; the mechanism of comparison is always present over the program, but there are differences upon the comparative attitude of agents, in fact, the user can set the rules followed by agents in order to see if arises different situations; more specifically agents can be provided with a consumerist attitude or with an altruistic one. Moreover, other variations are available both on the particular reference group with which agents make comparison and both on the working strategy adopted.

Let's start to define more specifically the world in which agents live; first of all the world is characterized by the presence of *only one good* (there is an initial endowment of the good in the world and after it is produced by the agents in exchange for payment). All agents desire the only good because it makes them happy, the agents use all their income in order to buy the maximum available amount of good and later they compare their own received amount with the amounts received by a particular group of other agents, after that they set their happiness status following the rule of behavior defined by the user: if the rule is "consumerist" then each agent is happy if he received an amount of good equal or greater than the mean of amount received by others agents belonging to its own reference group (roughly speaking the group is composed by neighbors close to agent within a specified distance), otherwise the agent is unhappy.
If, instead the rule is "altruistic", then each agent is happy if the variance of the amounts of good received within the specific reference group is lower than a satisfaction index fixed by user; however, it should be stressed that the altruistic behavior differentiates not only in the reference for the comparison mechanism, but it is characterized by the action of donating to others: in fact, agents who received an amount of good greater than the mean of good received within the reference group donate the surplus part. So, while the ultimate goal in the consumer context is to have more than others, in the altruistic framework the target is to reduce the gap between different amounts of good received, so that there are less differences among agents.

After the process of comparison and assessment of happiness condition each agent works, first of all to earn an income useful for future purchase of the good and secondly to participate in the production of the good that will be requested in the future; the user can choose between two different working strategies (one completely random and the other addressed to reach an uniformation of behavior

among agents).

It is now necessary to highlight explicity the stronger assumptions upon which the model is based. Among the others, the definition of the working behaviors is highly discretionary and if you want questionable: for agents the choice of how much work is eventually disconnected from reasoning about their previous situation, there are no possibility to implement adaptive strategies evaluating the results of previous choices and furthermore is recurrent in the model the assumption that agents happy are less involved in work because they fell themselves satisfied whereas agents unhappy have stronger incentives toward work in order to have in future the opportunity to be happier than today. Secondly, each agent has the same attitude toward greed and by no way it is possible to contrast different attitudes in research of optimum goal for himself: so, the conducts (perfectly observable in the real world) like greater stimuli toward the goal increasing with the level of income are completely rules out as a reduction of target after have felt a continuous state of inconvenience. The agents in the model do not show greater efforts toward a always close target, firstly because they are not able to evaluate the closeness of the goal.

Another topic I think is important to deal with is that of quantification of the level of happiness; initially I thought to determine the happiness looking both at the own absolute level of income and levels of others both at the amount of good received but this approach very soon raise walls in the definition of the happiness both within the same agent in different moment of time both among different agents; each agents in fact could have a different perception of his threshold that trigger the status of happiness and may exists a different scale of satisfaction connected both to general environmental variables both to subjective considerations but these issues lead to important theoretical question about quantification of qualitative measure of well-being like, what are the levels of good received that trigger the modification of status? How define an arbitrary scale of happiness able to shape itself to changes in structure of the world and in absolute value of variables? Differently from the works from which I was inspired (Easterlin 1994 and Clark – Oswald 1994), I decided to get rid of the above issues assuming that agents can feel only two different way of happiness, happy or not happy without intermediate step. In addition, looking at the results I aggregate respectively the total number of happy agents and that of unhappy ones taking for granted that independently from peculiarities of individual's condition, one happy agent is happy in the same way as another different happy agent; so, a question like - take two happy agents who is the happier? - is meaningless in the model. The above choice is motivated by the fact that in the studies cited the assessment of satisfaction level is let free to qualitative and subjective judgment of each singular person while in the simulation the rules of the game are obviously different.

2 - Implementation of model

2.1 – description of program

After the above general description of the overall model I will dig deeper firstly in the detailed implementation of the world in Netlogo and then I will describe the more interesting part of the code.
Differently from the first version of the program, in this last release the agents in the world are represented by patches, each agent/patch is equipped with an initial own_income and a color that define his status about happiness (green in different shades means happy while red in different shades means unhappy); the price of the only one good existing is fixed and the initial amount available of good is defined by user. The user has also the opportunity to identify the more wealthy agents within the whole population in order to understand clearly the dynamics that could emerge during the execution the simulation; the differentiation is driven through a color specification and a variable assignment.
The program follow a linear structure that define the actions that agents must do, the structure is divided in three parts:

I. actions concerning the general structure

   A) *evaluate rank*: in relation to the choose done by the user about the differentiation of wealthier agents from others, each agent/patch is asked to evaluate if it belongs to the upper group or to the lower one

   B) *compute desired*: each agent/patch compute the potential amount of the desired good that he is able to tackle with his own level of income and put the result in a variable he owns

   C) *receive good*: since the global amount of available good can be lower than the total sum of good desired by patches all together, it can show up a problem of parallelization that it is solved giving to each agent an amount proportional to that desired

   D) user define the rules related to management of the eventual stock of good that arise when total desired amount of good is lower than the available one

   E) *compute reference*: each agent/patch, following the rules defined by user about radius of looking and target of comparison (show attention to everybody or show attention only to similar), becomes aware of its own reference group of patches. After that, the agents perform a basic analysis of the structure of the reference group computing some indicators within

the same group: mean of the amounts of good received and the variance of these received amounts

II. actions that follow different patters: consumerist or altruistic approach

- in the consumerist behavior context defined in the model the agents are happy if they received an amount of good equal or greater to the mean received in their own reference group, that leads the agents to perform a check of these values and then set the status of happiness coherent

or

- in the altruistic behavior contest defined in the model the agents are happy if the variance of the received amounts of good (after the donation step) within the reference group is equal or lower than a user specified index of altruistic satisfaction. Under this assumption the actions taken by agents are extended with respect to consumerist context:

  A) *donate*: each agent/patch firstly compare its own received amount of good to the mean of the amounts received within the reference group and if there is a surplus he donates this to global population

  B) *accept donation*: after having sum up all the donated amounts the issue is to redistribute this quantity to agents characterized by null donation in the step above; it can show up a problem of parallelization that is solved giving a proportional amount to all agents which quantity of received good was lower than mean in its own reference group. It is valuable to note that the assumptions taken here are strong, in fact the dynamics of donation behavior could be more complex and geared toward help more who is poorer on global scale and maybe it could also possible to think about inefficiencies in the mechanism; however these considerations though are not been taken in practice

  C) *compare to others*: each agent/patch recompute the indicators of its own reference group and then make a comparison of the variance with the index of altruistic satisfaction set by user; after that each agent modify the status of happiness according to result

III. actions related to work

- Independently from the pattern followed in the previous step, each agent/patch is asked to work following the working strategy defined by the user. Moreover in this part of the program there are instructions on how

manage the amount of good produced by agents and there are also indications on what variables plot in the graphs

2.2 - code

Now, I will expose in more detail the most interested part of code actually implied for the realization of the model described above.
Firstly I defined the globals (fig 2.2.1) that allow to track the changes in the global amount of good and those which are useful to solve the problems of parallelization that can arise in the model, principally when dealing with the reception of the desired amount of good and in relation with the action of receiving the assigned quota of donation. Among the others, the variable named ordered_list_income assumed the specific form of a list and store in itself the different levels of income among patches in order to identify if a patch belongs to upper or lower group. Instead, the variable named num_top store the approximate number of patches that must be highlighted as belonging to upper (top) class as suggested by user through the slider %_top. Secondly I defined the variables (fig 2.2.2) that are owned by each patch: I deserve attention to the variable named comparison_group that assumed the form of an agentset composed by all patches that satisfy the parameters set by user about both closeness (via radius_of_looking) both target (via only_look_to_similar). Another variable interesting is that named last_wage that in the execution of the simulation is useful for the purpose of maintain an history of the last own_income since the specific value of the latter is subject to change in the simulation; the remaining variables have pretty self-explanatory names.

```
globals [sum_desired
  available_good
  sum_donated
  sum_inequality
  num_top
  ordered_list_income
  price_good]
```

fig 2.2.1 globals assignment

```
patches-own [own_income
  last_wage
  comparison_group
  others_received_mean
  others_received_var
  others_received_stdev
  received
  desired
  donated
  inequality
  category
  production
  happiness]
```

fig2.2.2 patches-own assignment

Then I populate the world with the setup procedure (fig 2.2.3) ; it is assigned to each patch an initial random amount of income (with a level of reference defined by the user through the slider initial_income_reference) and after that the initial status of satisfaction of each patch is given in a random way, completely

uncorrelated with the level of income.

```
to setup
  clear-all
  set price_good 10
  ask patches [
    set own_income (random-float initial_income_reference)
    set happiness one-of ["yes" "no"]]
  evaluate_rank
  assign_color
end
```

fig 2.2.3 setup procedure

Then within setup are called two procedure that will be used also in the go procedure: evaluate_rank and assign_color. The procedure evaluate_rank (fig 2.2.4) represent a tricky way to solve the problem related to distribution of wealthier agents in the world: in fact, given the characteristic of immobility of patches (with respect to turtles) it is impossible to assume, in a dynamic context as is a simulation, that a particular area of the world groups the agents belonging to upper class. The interactions that arise among agents can change the personal condition of each agent and maybe an agent belonging initially to upper class will fall in lower class or vice versa, for that reason it is not feasible to assign a label to agents forever. The procedure evaluate_rank, as structured, first of all look at %_top indicated by user and compute the number of agents among all existing that have to be classified as top (belonging to upper class), then it creates a list of incomes of all patches in reverse order so that the greater value is listed as first. Then the value of income in position defined by num_top variable is taken as threshold above which each patch classify itself as belonging to upper class an set the category variable to top while if own_income of the patch is lower it sets its category variable to down. From a technical point of view the presence of -1 derives from the behavior of Netlogo that considers the list starting with the index 0. In case user does not want to characterize the agents then all patches are equipped with the category down.

```
to evaluate_rank
  set num_top (round(count patches * %_top))
  set ordered_list_income reverse(sort([own_income] of patches))
  ask patches [ifelse %_top != 0 [
    ifelse (own_income >= item (num_top - 1) ordered_list_income) [set category "top"]
    [set category "down"]]
  [set category "down"]]
end
```

fig 2.2.4 evaluate_rank procedure

The procedure assign_color (fig 2.2.5) represents a simple assignment code, each patch depending on the happiness status and reference category set its own color. The trick implemented here to distinguish visually the upper class from the

lower one is that of use of different gradations of the colors red and green.

```
to assign_color
  ask patches [
    if (happiness = "yes") and (category = "top") [set pcolor 57]
    if (happiness = "no") and (category = "top") [set pcolor 17]
    if (happiness = "yes") and (category = "down") [set pcolor 55]
    if (happiness = "no") and (category = "down") [set pcolor 15]]
end
```

fig 2.2.5 assign_color procedure

Ending with the setup of the world, the other core procedure of the model is the go procedure (fig 2.2.6) flagged with the take forever option. Initially the patches became aware of their rank in the same manner as in setup procedure; then they compute the potential amount of good they can tackle and after that the distribution of the available amount takes place (solving also the eventually problem of parallelization). Then, a procedure named evaluate_stock is executed: here it is took in account the indication given by user on the management of the stock.

```
to go
  set available_good global_good_quantity
  evaluate_rank
  do_plots_income
  ask patches [set last_wage own_income]
  ask patches [compute_desired]
  set sum_desired (sum [desired] of patches)
  ask patches [receive]
  evaluate_stock
  ask patches [compute_reference]
  if behaviour = "altruistic"[ask patches [donate]
    set sum_donated (sum [donated] of patches)
    set sum_inequality (sum [inequality] of patches)
    ask patches with [donated = 0] [accept_donated]
    ask patches [compare_altruistic]
    ask patches [work]]
  if behaviour = "consumerist"[ask patches [compare_consumerist]
    ask patches [work]]
  set global_good_quantity ((sum [production] of patches) + available_good)
  assign_color
  do_plots_happiness
  ask patches [set others_received_var 0
    set others_received_stdev 0]
  tick
end
```

fig 2.2.6 go procedure

The next important procedure within go is that named compute_reference (fig 2.2.7), firstly each agent/patch following the rules defined by user through the slider radius_of_looking and the chooser only_look_to_similar becomes aware of its own reference group of patches. After that, the agents perform a basic analysis of the structure of the reference group computing the mean of the amounts of good

received within the group and the variance of these received amounts. The computation of the variance and standard deviation is limited only in cases characterized by an agentset that compose the comparison group different from value one.

```
to compute_reference
  if only_look_to_similar = "no" [set comparison_group patches in-radius radius_of_looking]
  if only_look_to_similar = "yes" [set comparison_group patches in-radius radius_of_looking with [category = [category] of myself]]
  set others_received_mean (mean [received] of comparison_group)
  if count comparison_group != 1 [set others_received_var (variance [received] of comparison_group)
    set others_received_stdev (standard-deviation [received] of comparison_group)]
end
```

fig 2.2.7 compute_reference procedure

Therefore the go procedure forks in accordance to what specified by user regarding the behaviour that agents must follow: if the chooser behaviour is on position consumerist than each patch follow the procedure named compare consumerist through which it reaches as final result a status of happiness (yes or no); if instead the chooser behaviour is on position altruists then patches start the donate and receive donation processes and finally following the procedure named compare altruistic they set their status of happiness. Hence the program back from fork and each agent is required to work. The procedure work (fig 2.2.8) allows two different strategies depending on the chooser work-behaviour. As stated previously (in the introduction) the work action is object of very strong assumptions: if the choice is copying_other_people then agents who received more with respect to others became lazy because they are satisfied and they are happy until they receive a greater than mean of good, so they ended working less than the previous tick; while agents who received less tah others make more effort in order to gain more and so have more opportunities to be happy in the next tick (so they level their work to gain an amount necessary to buy the amount of good corresponding to the mean in reference group). Obviously these are completely arbitrary simplifications to maintain the model simple. The other possible behaviour for what concern work is that of a random behaviour of agents centred on the level of the last wage. Moreover to generate some noise I contemplate a random result from the work, the level of good produced is not predetermined in value but fall in a range specified by a random normal distribution whit mean 1 and variance 0.05 (for now fix) without differences related to income absolute value.

```
to work
  if work_behaviour = "random" [set own_income (last_wage * random-normal 1 0.15)
    set production ((own_income / price_good) * random-normal 1 0.05)]
  if work_behaviour = "copying_other_people" [set own_income ((others_received_mean * price_good) * random-normal 1 0.0)
    set production ((own_income / price_good) * random-normal 1 0.05)]
end
```

fig 2.2.8 work procedure

Then the program deals with the management of the amounts of good produced by all agents plus the possible stock upgrading the variable named global_good_quantity. The last thing to notice about the go code is the position of the the procedures used to plot the statistics about the simulation; at the beginning of go there is do_plots_income while at the end there is do_plots_happiness, the reason why the former and the latter are separated both in code both in time of execution is due to the need of greater clarification between a starting point (represented by income) and an end point reach after the execution of the program (represented by the status of happiness).

3 - Experiments

This section is deserved to analyze some of the various result that the program generate when it is executed, I decide to maintain a stable choice for the indicators that determine the initial welfare of agents and the values that allow each patch to define how to shape its own comparison_group. Otherwise I will focus the attention on the parameters that determine the behavior of each patch, trying to understand the mechanisms that lead to results, being conscious of the strong assumptions taken in the model.

3.1 – consumerist situation

In this first trial I will interpret the results controlling for the parameter *only_look_to_similar* taking the others parameters as display in fig 3.1.1. The underlying behavior of all agents is *consumerist*.

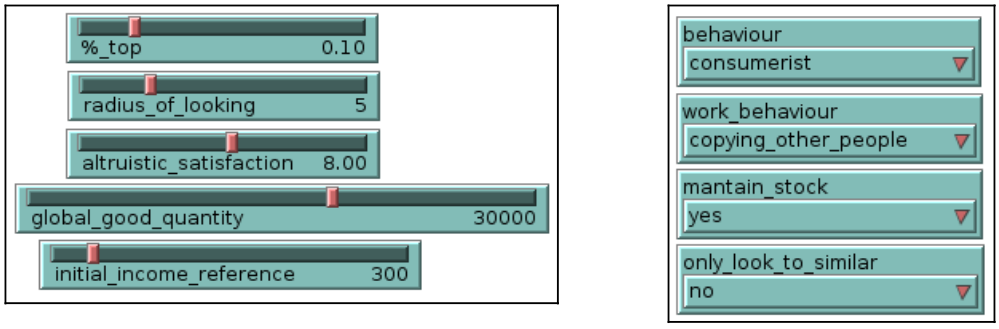
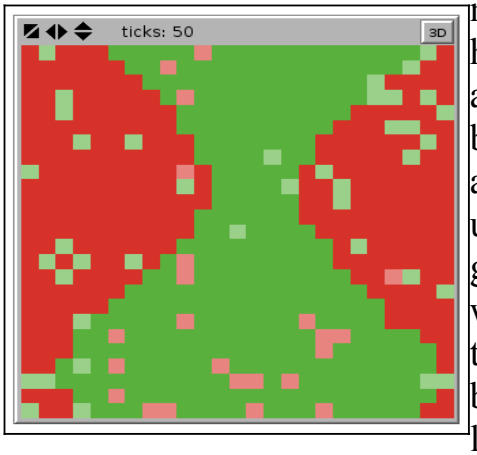


fig 3.1.1

During the execution of the simulation arise not only a different final situation but also an intermediate one when agents of one category are allowed or not to look at agents of the other category in order to compute their own comparison_group.

- *only_look_to_similar* → *yes*

When agents are asked to look only to similar then then the agents belonging to category larger (“down” in this case) initially aggregate more in the world maintaining a substantially equal distribution of happy and unhappy;



after this initial step the mechanism of work behavior (addressed to copy the behavior of others agents in the same comparison group) reach the ultimate result of equalize the received amount of good with the mean of amounts of received good within the reference group and so patches classify themselves as happy. Otherwise the fact that agents belonging to lower populated category have a longer period of time with no aggregation in the world is probably due to the lower number of agents belonging to each comparison group; restarting the simulation with a high radius_of_looking like 12 reduces the attending time to reach a more stable situation.

Looking at the final result (fig 3.1.2) it is possible to note that agents belonging to greater group (her those which category is “down”) are quite all happy while agents belonging to smaller group converge to happy status but through a more difficult pattern. The reason may be find both in copying_other_people strategy of work both in the different granularity of category in the world; in fact the mechanism of work here leads the lower_people to uniform almost immediately their income and so their received amount of good, in that way they fell happy. The uniformation process for the upper_people instead is slower.

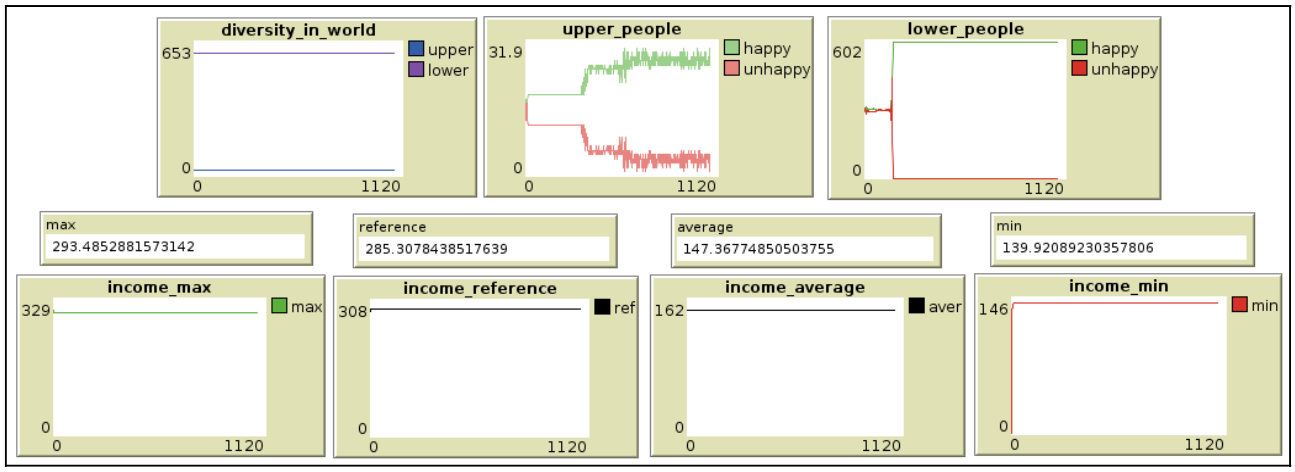
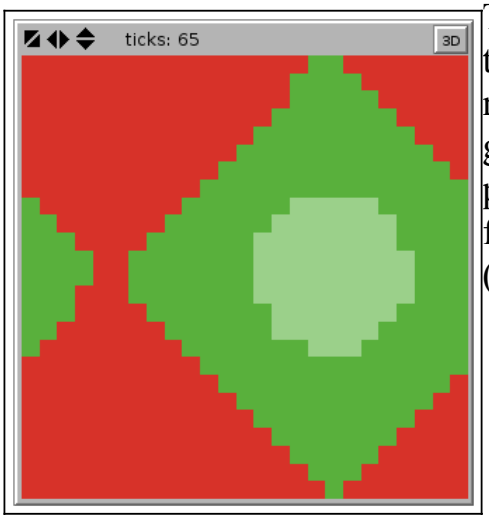


fig 3.1.2

- *only_look_to_similar → no*

When instead agents are asked to look to everybody without distinctions of

category then the structure that at intermediate level arise display an ordered aggregation of happy-agents-belonging to smaller group, then happy-agents_belonging to greater group and finally unhappy-agents-belonging to smaller group; moreover there are not unhappy-agents-belonging to greater group.



The structure has a shape similar to two donuts and the agent with the highest received amount of received good is situated in the center of the light green circle; the form is due to the choice of patches as agents, in fact when patches look around for example to radius 1 they meet 5 patches (including himself) and so on.

Looking at the final result (fig 3.1.3) it is possible to note that all agents end with an happy status, moreover the plot named diversity_in_world suggest that in final situation there is only one category. The reason for this jump come from the equalization of income (and so indirectly of received good) among all patches (the graphs show clearly this phenomena); when it is time for patches to became aware of their category they face a reference level that is equal to their equalized level of own_income and so following the rule indicated in "evaluate_rank" procedure they set their category to top. If you modify the evaluate_rank procedure removing the equal sign than all patches categorize themselves as belonging to lower category.

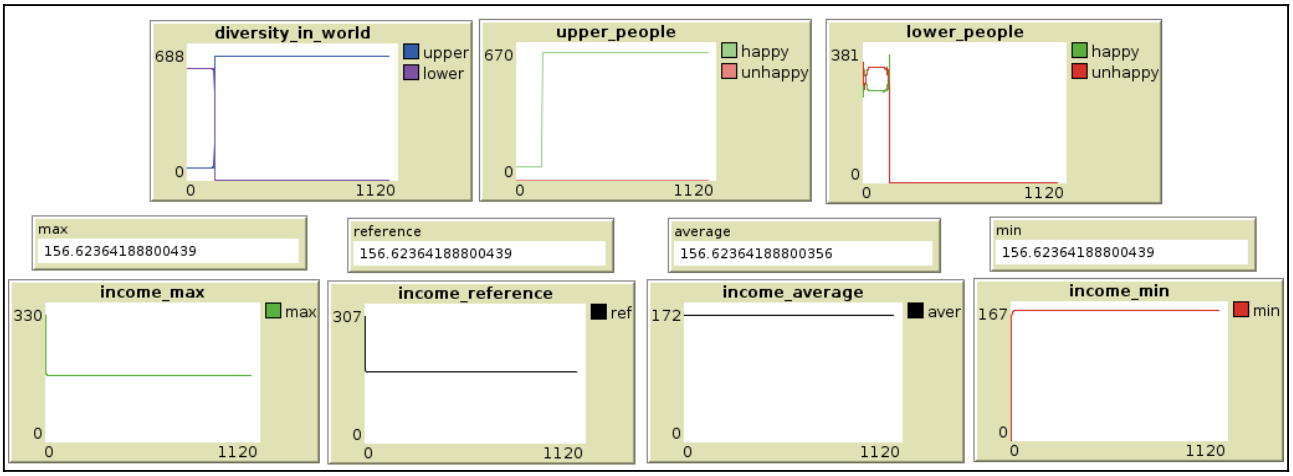


fig 3.1.3

3.2 – altruistic situation

In this second trial I will interpret the results controlling for the parameter *only_look_to_similar* taking the others parameters as display in fig 3.2.1. The underlying behavior of all agents is *altruistic*.

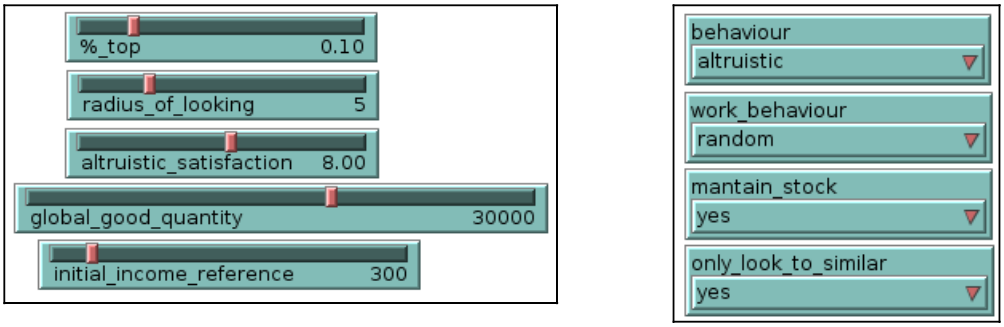
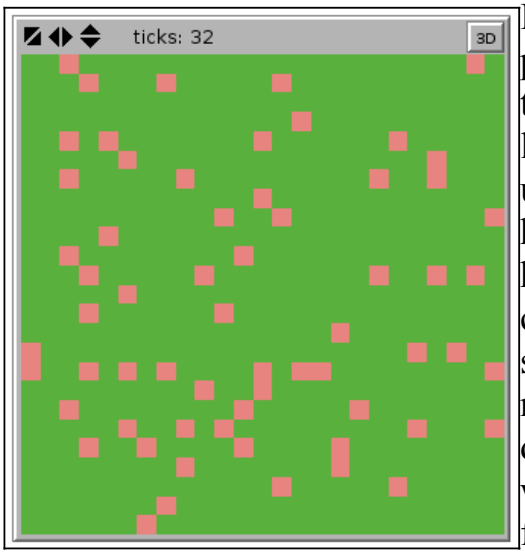


fig 3.2.1

- only_look_to_similar → yes

When agents are asked to look only to similar then then the agents belonging to the category relatively poorer ("down" in this case) initially aggregate more in the world and almost everybody became soon happy while the agents belonging to upper class are all unhappy.



If we repeat the experiment changing the % of patches that must be characterized as top fixing the slider to 0.50, we can see that at intermediate level again the agents classified as top are unhappy while in the lower group they are all happy, this attempt can show that this structure has nothing to do with the granularity of categories in the world. Another test is that of shocking the world repeatedly with the button named increase_own_income of 100%, after doing so we can see an immediate reaction in the world, in fact now all patches independently from the typology are unhappy; this can allow us to argue that an increase in the absolute value of own_income of all patches disturbs the mechanism inside the compare_altruistic procedure creating a rise in the others_received_variance.

Looking at the final result (fig 3.2.2) it is possible to note that all agents end with an happy status, however the way to reach the status was different for the two category. While the agents belonging to group poorer became almost immediately happy, those richer fell a quite long period of unhappiness until the highest values

of income fall down; after that they start converging toward happiness. The hypothesized explanation (of absolute level of income that determine the different pattern of happiness) can be supported not only by the shock hitting the world (as seen just above) but also by a fast change in the procedure evaluate_rank: if we order the income in ordered_list_income from low to higher and fixing a % of poor people very small we see that those became happy soon while richer became happy slower.

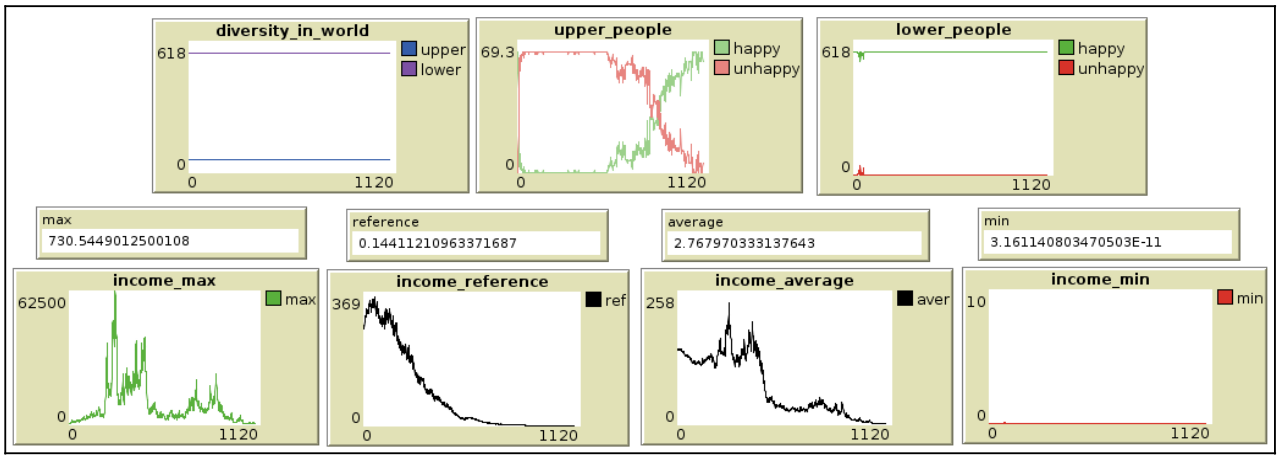
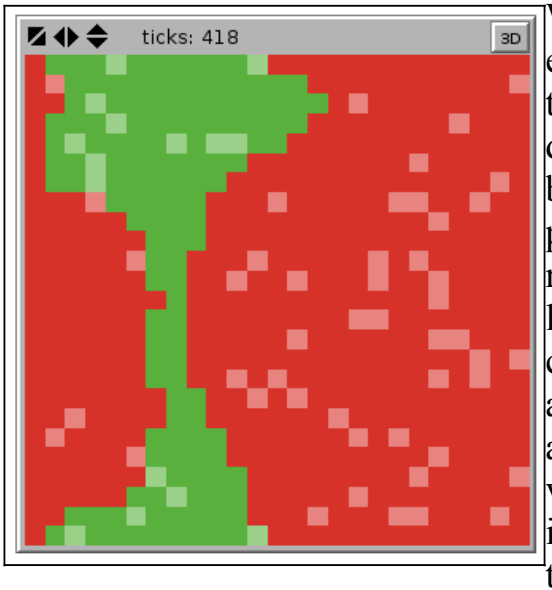


Fig 3.2.2

- only_look_to_similar → no



When instead agents are asked to look to everybody without distinctions of category then the structure that at intermediate level arise display a sort of "wave" composed by agents belonging to greater group that became happy, probably because of favorable conditions from random work behavior in connection with a lower absolute value of income. The peculiar characteristic of this wave is that it moves in aggregate way, a motive for this shape is ascribable to the fact that agents with the variable others_received_variance lower than index of altruistic_satisfaction are most likely to be close regardless the category of membership since the context here analyzed force agents to look to everybody.

Looking at the final result (fig 3.2.3) it is possible to note that all agents end with an happy status and both the upper group of agents both the lower one follow the same pattern to reach this final situation. It is important to note that the situation is quite

stable until the income_max (and so consequently the other income levels) remains under a specific level; if you try to shock consecutively the world increasing all income then the equilibrium breaks. Moreover, differently from the previous simulation, in this case the shock strikes the world regardless of category (without specific effect on only one group of people); the effect can be seen as arising of red circles in the world.

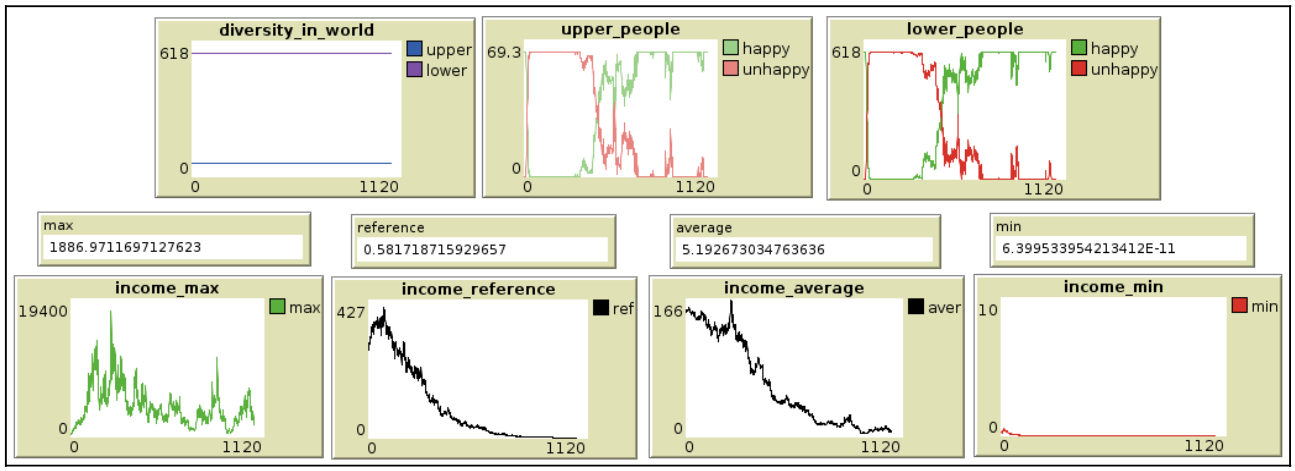


fig 3.2.3

3.3 – final considerations

The experiments above are been chosen because I think they are flexible enough to show the different situations that can arise from this simple model, the selection of different value for parameters and slider leads to variations on the theme: usually a greater or smaller radius_of_looking imply a different speed in the process of adjustment to quasi-stable status of happiness (especially when the work strategy is set on copying_other_people), the same can be said about the altruistic_satisfaction_index. A last remark on the effect of a shock represented by a huge increase of the level of income of all agents in a consumerist context; during the various simulation a linear shock of this type has never influenced the level of happiness, maybe for a future work can be interesting to experiment with different type of shock.