The Centre for Interdisciplinary Science

Complex Systems-Sugarscape

Contents

1. The Sugarscape
2. The Interface
3. Preset 1 – Movement: Navigating the Sugarscape
4. Preset 2 – Migration and Emergence
5. Preset 3 – Reproduction: Dynamic Populations
6. Preset 4 – Two Resources: Spice comes to the Sugarscape
7. Preset 5 – Trade: The Decentralised Marketplace
8. Preset 6 – Loans: Emergence of Hierarchy
9. Preset 7 – Disease: Immunity and Transmission
10. Preset 8 – Life of the Brink

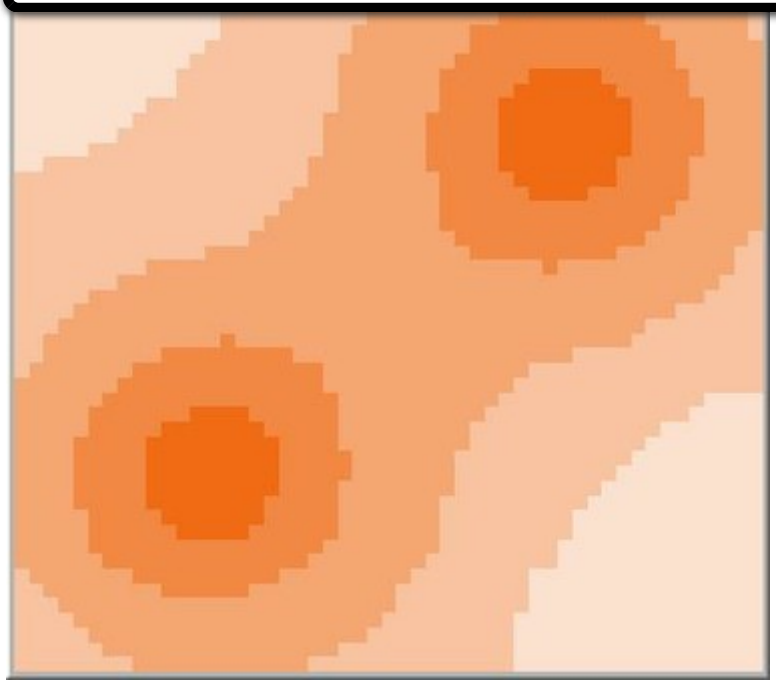
Sugarscape is a model artificially intelligent, agent-based, social simulation following some or all rules presented by Joshua M. Epstein & Robert Axtell in their book *Growing Artificial Societies*.

The simulation was translated into the Netlogo language by Iain Weaver as a summer internship project and is available from the Netlogo library on the Netlogo web site.

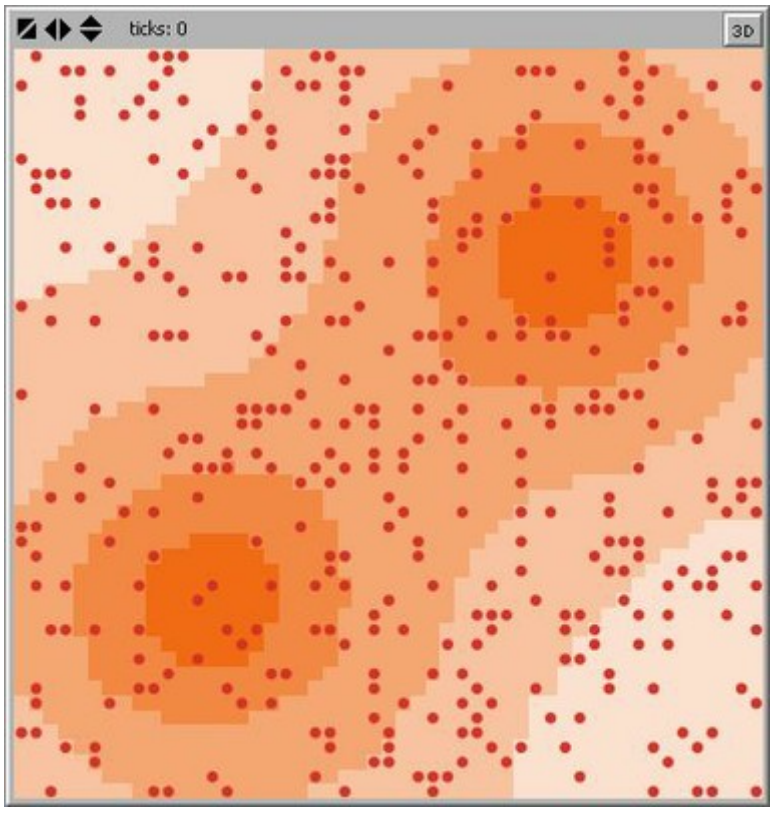
The information below describes some of the available scenarios.

The Sugarscape

The Sugarscape is a 51x51 grid divided into squares called "patches". Each of these patches has 2 variables; amount of sugar and maximum sugar capacity. Initially, the maximum capacity is set from a map, and the amount of sugar is set equal to this (each patch is full). The amount of sugar in each patch is indicated by its colour. Here, the orange shows 4 units of sugar in a patch, while white shows 0 units in the patch.

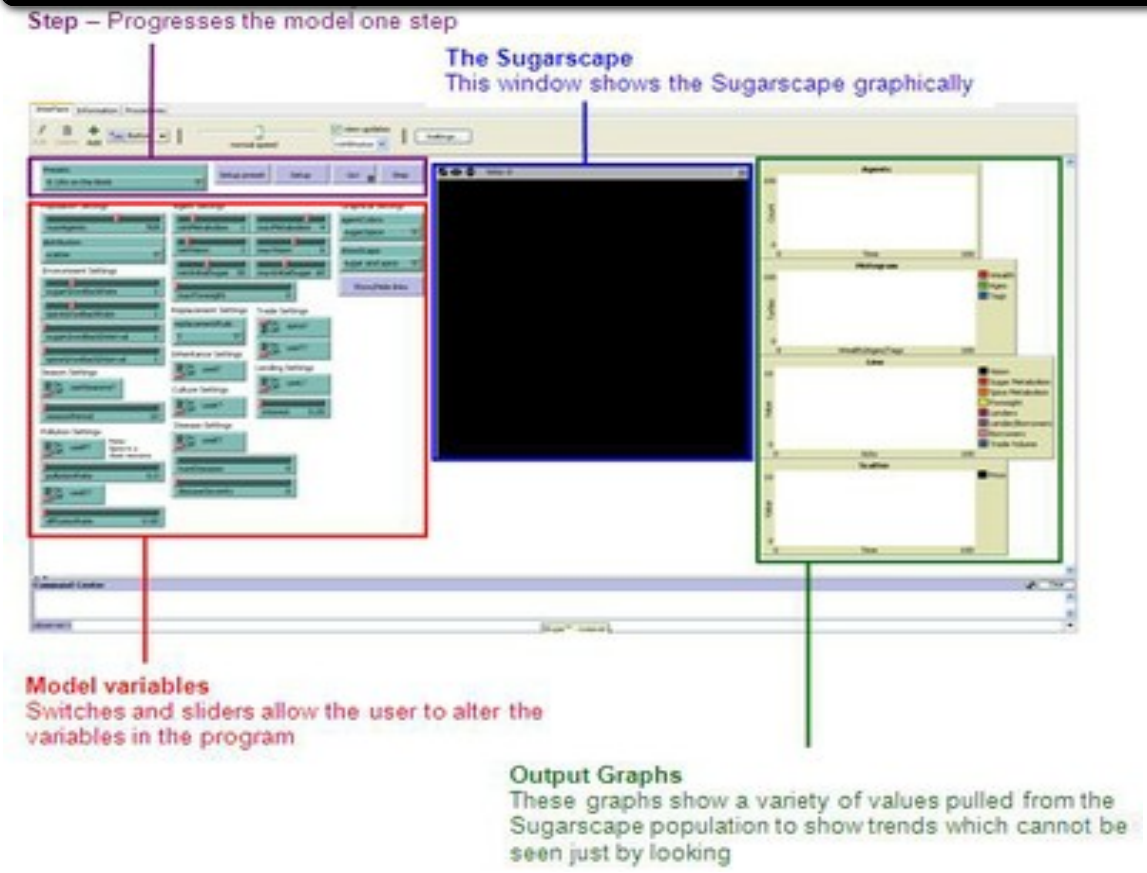


Next, we populate the Sugarscape with automatons called "agents". Each of these agents has 3 variables; *sugar*, *sugarMetabolism* and *vision*. Initially, the population is given values for these variables, randomly selected between given limits. *vision* and *sugarMetabolism* remain fixed for the agents life, we call them the agents' genetic traits. Agents set out into the world with and initial *sugar* endowment of *initSugar*.



Both agents and patches follow simple rules. First each agent is selected in a random order to carry out its rules. When the last agent has acted, the patches carry out their rules, and the turn, or "tick", is complete.

The Interface



(Click on the image for a more detailed view)

Preset 1 – Movement: Navigating the Sugarscape

For the simplest models, agents and patches follow a single rule each:

Patch growback rule G

Grow back *sugarGrowBackRate* units of sugar every *sugarGrowBackInterval* ticks, up to the maximum capacity.

Agent movement rule M

1. Look in the 4 compass directions up to *vision* patches
2. Select vacant patches with the highest sugar value
3. If several patches exist with the same value, choose the closest
4. Move to this patch, take its sugar and add it to the agent's *sugar*
5. Subtract *sugarMetabolism* from *sugar*. If the *sugar* ≤ 0, the agent dies and is removed from the Sugarscape

Both of these rules are fundamental and are included in all presets, though sometimes with slight modifications. The preset gives the agents an initial *sugar* between 5 and 25 units, *vision* and *metabolism* between 1 and 6, and between 1 and 4 respectively. The initial number of agents, *numAgents*, are distributed randomly throughout Sugarscape. The environment parameters *sugarGrowBackRate* and *sugarGrowBackInterval* are set to 1.

Things to try

Select *"1: Movement: Navigating the Sugarscape"* from the dropdown Presets menu and click *"Setup Preset"*. This sets all parameters to preset values to create the Sugarscape and scatter agents across it, then *"Go!"* to see them in action. What do they do?

Increase the *sugarGrowBackRate* to 4. This will mean each site recovers to full capacity in a single tick. Click *"Setup"* to setup with this manually adjusted parameter, and then *"Go!"*. How does this alter the agents' behaviour?

Some agents on the lower tiers will die for this reason since without moving there is no hope of finding richer sugar regions to satisfy their metabolism.

Compared to a rate of 4, the lower *sugarGrowBackRate* forces agents to move around the Sugarscape. The end result is that many more agents find the sugar peaks. This is an example of decentralisation. If you watch a single high-sugar patch at the peaks, you will see that as it approaches capacity, an agent appears to harvest it, and then leaves it to grow. To harvest sugar at a maximum rate, agents at the peaks could be individually programmed to harvest specific patches in a given order. However, simply following **M**, agents produce near-optimal results.

Preset 2 – Migration and Emergence

Preset 1 uses the same rules as above with a few modifications. The maximum limit for agent vision is increased to 10, *sugarGrowBackInterval* is changed to 2 and rather than being scattered on the Sugarscape, agents are set up in a dense cluster in the bottom left corner of the Sugarscape.

Things to try

Select preset 2 and *"Setup Preset"*. Advance the animation one step at a time using the *"Step"* button. What direction does the population travel? Go to "Sandbox" and find how reducing *numAgents* (the initial population size) or increasing *vision* alters this effect.

Things to notice

By progressing the model step by step, waves of agents can clearly be seen propagating diagonally across the Sugarscape, a direction unavailable to individual agents which can be demonstrated by lowering the population to ~100, where the migration no longer takes place. This is an example of emergence since the population as a whole seems to behave very differently to the individual agents which make it up.

Using the slider to increase the mean *vision* of agents means more agents can participate in the waves, and the effect becomes stronger.

Preset 3 – Reproduction: Dynamic Populations

These presets introduce a new rule for sexual reproduction, accompanied by 4 new parameters for individual agents; *gender*, *age*, *maxAge* and *fertileLimits* for the start and end of their fertility. *maxAge* is a value randomly chosen for each agent between 60 and 100 – it gives the number of ticks an agent can occupy the Sugarscape before it dies of old age. A fertile agent must have at least as much sugar as it was initially given, and be within the age boundaries for fertility. Gender is assigned randomly when a new agent is created and *age* incremented each tick.

Agent reproduction rule S

1. Select each neighbour in a random order
2. If the neighbour is fertile, of the opposite sex, and one of the agents is adjacent to an empty patch, both agents deplete their *sugar* by half of their initial endowment and a new agent is created in the empty patch
3. Repeat for each neighbour

New agents created in this way inherit a combination of the genetic characteristics (*vision* and *metabolism*) of their parents, as well as an initial sugar endowment from the combined contributions of the parents.

Things to try

Preset 3 gives agents an initial endowment between 40 and 80 units of sugar. Agents are coloured based on their gender (blue or red) and coloured grey if their *age* is above the maximum limit for fertility. Is a stable population level reached? Try reducing *initSugar* between 10 and 40 units instead. Remember that the initial amount of sugar given to an agent is tied to the amount that agent needs to carry out rule **S**.

Rule ***I*** is an inheritance rule. Upon death, an agent's wealth is simply divided between its children. Load the preset settings, and turn inheritance on by changing the "*useI?*" switch. What effect does this have on the maximum population? Does it alter the rate of natural selection (how quickly the mean metabolism and vision change over time)?

Things to notice

By reducing *unsugar*, the sugar needed to carry out **S** is relatively small compared to the sugar agents can gather, resulting in a much higher population. Due to the density, agents with high metabolism are quickly removed and the limiting factor on the population is the space available. The population varies a great deal in oscillations with a period of ~120 ticks such that no agent will see a full cycle.

Additionally, the histogram displays the distribution of different ages of agents in the population. Every 120 ticks, a sudden "baby-boom" occurs which can be seen to pass through the population as a wave.

Using the inheritance rule, ***I***, has some interesting results. The wealth distribution changes significantly with more rich agents inhabiting the Sugarscape. Natural selection occurs more slowly as agents born with unfortunate genes may be advantaged by a significant inheritance, and their genes will persist.

Preset 4 – Two Resources: Spice comes to the Sugarscape

To create the next scenario, another resource is added to the Sugarscape, spice. Spice is treated identically to sugar. Each patch on the Sugarscape is given a spice capacity and grows according to G – the actual shape of the spice regions here is a mirror image of the one for sugar, so generally sugar hills are in the top right and bottom left while spice hills are in the top left and bottom right. Agent parameters *spice* and *spiceMetabolism* mirror the equivalent parameters for sugar. Life in the Sugarscape is about to get harder though, as agents must keep both sugar and spice > 0 to survive each tick. To help them to this, we must modify **M** so agents can identify the relative importance of sugar and spice.

Another agent parameter, *welfare*, is added. It relates the time until the agent will die from lack of sugar and the time until the agent will die from lack of spice.

Agent movement rule M

1. Look in the 4 compass directions up to *vision* patches
2. Select vacant patches which offer the highest *welfare* once consumed
3. If several patches exist with the same value, choose the closest
4. Move to this patch, and eat all of the sugar and spice
5. Subtract *sugarMetabolism* from *sugar*, and *spiceMetabolism* from *spice*. If the *sugar* or *spice* ≤ 0, the agent dies and is removed from the Sugarscape

Things to try

Life on the Sugarscape is particularly unforgiving under these rules, and the carrying capacity of the environment is much lower than seen in preset 1 – the same conditions with only one resource. In this preset, agents are coloured according to the ratio of the resources they have gathered. Red agents have collected mostly sugar, while blue agents rely mostly on spice. Try watching different agents. What different "tricks" can they use to survive? To watch an agent: pause the model, right click the agent, select "agent" and click "watch".

Things to notice

Agents with a greater need of one resource than the other tend to hive around the peaks, collecting only small amounts of the opposite resource and maintaining their *welfare*.

A more interesting result is that some of the agents with high vision and balanced needs will traverse much more of the Sugarscape. As they accumulate one resource, their welfare will cause them to seek the other to balance this, and they will travel from peak to peak. These agents can often be identified by red-coloured agents (harvested mostly sugar) gathering in the spice peaks, or visa-versa.

Preset 5 – Trade: The Decentralised Marketplace

In preset 4, many agents come into the world with an initial endowment of sugar and spice, and a location which does not meet their requirements. Often, though "fit" in the genetic sense, these individuals die off before they are able to traverse the Sugarscape to a region which meets their requirements. Similarly, agents atop resource hills may accumulate a great wealth of one resource, but run out of the other, and without a high vision these agents will also die. For these reasons, we introduce the concept of trade to the Sugarscape using a new rule, **T**. Along with *welfare*, agents are given another parameter called *MRS* (). This value shows if an agent's need is for spice (*MRS* < 0) or sugar (*MRS* > 0) as well as severity of the need. It is reasonable to expect that an agent with a severe need of spice, or with an excess of sugar is willing to offer its trade partner a better rate of exchange.

Agent trading rule T

1. Compare *MRS* to each neighbour in a random order

ahead

6. Repeat rule until no more trades with any neighbours are valid

Things to try

This preset uses the exact same conditions as the previous one, but with the trade rules turned on. The line graph below shows the total trade volume of each tick, and the scatter graph does the geometric mean p each tick. How does the capacity of the Sugarscape differ from the same conditions without trade? If the mean price is not 1, what does this tell you about the mean sugar and spice metabolisms of the agents? You can manually check the mean metabolism using the "***Command Center***" at the bottom. Pause the animation and type:

show mean [sugarMetabolism] of agents

or

show mean [spiceMetabolism] of agents

Try increasing *growBackInterval* for either sugar or spice, and notice what impact this has on the mean price. Recall that $p > 1$ indicates sugar is valued higher than spice while $p < 1$ indicates spice is more valuable to the agents.

Things to notice

The trade volume is initially very high, and trade prices vary greatly across the Sugarscape. During this stage, agents trade their initial endowments to better suit their metabolic needs. Quickly, the trade volume falls and remains roughly constant. By this stage, the trade prices generally stable and close to the "market-clearing" value of 1. Any deviation from this indicates the mean sugar and spice metabolisms of agents are not equal.

Making either resource scarcer by increasing its *growBackInterval* increases its value. p increases if sugar is reduced, and decreases when spice is reduced. The emergence of steady trade prices is another example of decentralization. Without being told that one resource is worth more than the other, that is to say having access to some global price, the steady price emerges from only local interactions between agents.

In both presets 4 and 5, the population quickly declines to a constant. All settings in these presets are equal except for the presence of **T** in 7. With **T** turned off, the Sugarscape supports an average of roughly 115 agents, while with **T** on, the average increases to 130. This increased survival is explored more fully in preset 8.

Preset 6 – Loans: Emergence of Hierarchy

As the Sugarscape currently exists, there is much waste. Agents too old to reproduce often have the majority of the wealth having accumulated it through their lives and many agents with the ability to reproduce lack the resources. Preset 6 allows hierarchical relationships to emerge, as agents with sufficient wealth to make a loan can lend to others (at an interest rate given by *interest*) for purposes of reproduction. This is done using rule **L**.

Agent lending rule L

1. Identify agent as either being in need of borrowing, or being a potential lender. A borrower is a fertile agent with insufficient sugar to reproduce. A lender is either fertile with an excess of sugar, or infertile. As a borrower, the agent will ask for a maximum of the amount needed to reproduce. As a lender, if fertile, the agent can lend a maximum of the excess above the sugar needed to reproduce. Otherwise the agent can offer up to half of its current wealth
2. As a lender, identify any neighbours who are borrowers, or visa-versa
3. For each valid neighbour a loan is made for the lower of the amounts offered and asked for called *debtAmount*
4. After 10 ticks, the borrower repays *debtAmount * (1 + interest)* to the lender and the loan is concluded
5. If the borrower does not have sufficient funds, half of its current wealth is repaid, and a new loan is issued for the remaining amount multiplied by *(1 + interest)*

Should either the lender or borrower die before the loan is repaid, the loan is cancelled so either the lender takes a loss or the borrower gets lucky.

Things to try

Preset 6 uses the same settings as 3, except with loans turned on. Agents are coloured according to their current credit situation - agents borrowing sugar are coloured pink, agents lending sugar are coloured magenta and agents both lending and borrowing are violet. The

parents were owed (note that they do not inherit loans their parents were yet to pay). As with preset 5, try adding rule **I** to this model. Does the hierarchy alter significantly? **I** can be turned on and off during the run to see any effect more clearly.

Things to notice

The first 3 of these groups form a very clear cut hierarchy with the lenders at the top, and borrowers at the bottom. This is an emergent structure in our artificial society, and the proportions in each category seem to remain static, even with a lower population density (produced by increasing *growBackInterval*).

With rule **I**, the hierarchy remains fundamentally the same, but the tiers are altered slightly. The lenders and the lender/borrowers significantly reduce in population. Rule **I** generally keeps accumulated sugar in the population, and wealthier agents are less likely to require a loan. The population is dominated by black agents which are neither lenders nor borrowers.

Preset 7 – Disease: Immunity and Transmission

Disease is known to have a profound effect on many aspects of a society. Preset 7 explores the effects on the behaviour of this artificial society. Along with the agents in the Sugarscape, a number of different diseases are created, each identified by a random list of up to 10 1's and 0's. To combat these diseases, each agent is also given an immune system in the form of a list of 50 1's and 0's. If a disease is a sub-list of the immune system (the disease's pattern of 1's and 0's occurs in the agent's immune system) then the agent is immune to that disease. If not, the agent is infected and suffers the effects of the disease, in this case a fever, increasing the rate of sugar and spice metabolism by up to *diseaseSeverity*. Infected agents' immune systems are enabled to combat disease through a new rule.

Agent immune system rule E

1. Select a present disease randomly
2. Check the immune system string for the first section matching closest to the disease string
3. If this section is not identical to the disease, flip the first wrong number
4. Check all diseases against the new immune system
5. Any disease which is a sub-list of the immune system list is removed

Additionally, agents should be able to transmit diseases amongst themselves. To achieve this, another new rule is employed.

Agent transmission rule T

1. Select each neighbour in turn
2. Select a random disease from the list of diseases present
3. Pass this disease onto the neighbour

Things to try

The preset infects each agent with 4 diseases initially from a catalogue of 10. Is this population able to rid itself of disease? Try increasing the size of the disease catalogue to 25. How does this affect the health of the agents in the Sugarscape? How does the initial population size affect its ability to combat disease?

Things to notice

With an average disease length of 5.5 and an immune system length 50, it is entirely feasible that an agent may be created with, or generate itself an immune system with a complete defence against all diseases. As this model runs, the population is quickly freed from all 10 diseases.

A disease catalogue of 25 or more presents a much greater challenge. Rarely, individuals will create an immune system to protect from all the diseases, and the population suffers from disease for a much longer period, sometimes never ridding itself entirely of diseases. In an attempt to remove one disease, agents may corrupt the coding for immunity to another and so they find themselves suffering from the same disease several times.

The persistence of disease is extremely sensitive to the initial population density. Anything below an initial population of 500 is very unlikely to have enough encounters for disease to keep hold of the population.

Preset 8 – Life of the Brink

Things to try

Run the preset a few times. Can the population survive under these conditions? Do their chances get any better if you add **T**? How do trade prices fluctuate throughout the model?

Things to notice

Without trade turned on, about 4 out of 5 times the population will crash after one generation, and usually be extinct soon after. The cause of the crash is an artefact of the initial setup, where agents begin at age 0. After 60 ticks, these initial members reach the end of their lives and are completely removed in the next 40 ticks.

What is more interesting is how allowing them to trade to stay alive means that about 4 out of 5 times the population will experience a near-extinction event as before but be able to recover and populate the Sugarscape far above the initial 500. During this crisis, trade prices are very variable. The lower population density means fewer trades take place each tick, so agents are unable to optimise their *MRS* so frequently.