SYSC557 Artificial Life
Final Project Report
Guy Cutting

Introduction and Background

The idea of self-organized criticality (SOC) makes numerous appearances in the Artificial Life literature, both in attempts to address the question of the origins of life, and more generally as a conceptual framework in which to understand life, life-like processes, and complex adaptive systems (CAS). SOC, originally introduced by Bak, Tang, and Weisenfeld, is a holistic property that governs dynamical systems of all types. The concept originated in statistical physics, but finds wide application across many diverse fields. SOC has put been forward as a candidate for explaining earthquakes, forest fires, neuronal cascades, and many other natural phenomena. These phenomena all exhibit certain common features, such as scale invariance (power-law distribution as in the Gutenberg-Richter law describing the statistical distribution of earthquake sizes), fractal geometry, and 1/f noise (the most common type of signal in biological systems. The original SOC model is a sandpile to which grains of sand are added slowly, one at a time. Periodically the addition of a single grain destabilizes the pile and causes a cascade or avalanche involving a very large number of grains. As the sandpile model illustrates, SOC is typically seen in slowly-driven nonlinear systems with many degrees of freedom [1].

SOC is conceived as an abstract set of ideas that can allow systems in many different disciplines to be analyzed in a common framework. It has proven powerful in understanding the behavior of “composite” systems, that is those containing a very large number of elements that interact over a short range. An obvious non-physical application of SOC is to economics. An economy can be understood as system which is self-organized by its participants to the region of criticality and which exhibits, at least on occasion, nonlinear behavior. Though a minor topic in the SOC literature, the connection between economics and criticality is made explicitly by some key authors in the field. Bak himself highlighted this application of SOC in early papers with economist Michael Woodford [2]. Stuart Kauffman also discusses the applicability of criticality mechanisms to understanding economic dynamics [4].

Though complexity science has made inroads among economists, the complex systems approach is still decidedly unconventional in the field. The neoclassical paradigm is built around the view that market mechanisms give an efficient allocation of resources, establishing equilibrium between supply and demand. Thus typical models assume an equilibrium (of prices, income, etc.) which might be temporarily disturbed by an exogenous shock - that is, one arising from outside the model - but which is eventually restored by the price mechanism or some other process. This view can perhaps be understood as a relic of early quantitative economic theory, in which such equilibrium was a simplifying assumption necessary to produce mathematically tractable models. Regardless of its historical origins, this fixation on equilibrium models is still a common feature of economics research, though increasingly scholars in economics and other disciplines have turned to ideas from the complex systems literature to find models for highly dynamic economic processes, and recognizing that the concept of equilibrium is of limited usefulness in modelling a complex adaptive system.

There are a number of possible ways to apply SOC to modelling economic systems. Economies are an excellent example of composite systems, consisting as they do of many agents who interact locally and nonlocally in extremely complex patterns, and this means that there are many interacting mechanisms that contribute to criticality. The original idea of this project was to explore applying Stuart Kauffman's work on autocatalytic sets to understanding criticality in economics. Kauffman himself has made several appeals for further exploration of this question, and in particular he has highlighted the link between criticality and crisis behavior, and made appeals for further exploration of the analogy between autocatalytic behavior and economic activity. Certain aspects of self-organized criticality have received sustained attention in the areas of economics and finance. In particular, Mandelbrot's work, which showed that asset prices demonstrate a power-law distribution and fractal properties, is widely regarded as formative in the area of quantitative finance [5]. But this is not the only possible application of SOC and related ideas. Of particular interest is using SOC to gain insight into crisis or crash behavior. Though economic crashes have been widely studied, the most recent financial crisis of 2008-2009 and the subsequent recession revealed that the economics discipline still has only a limited understanding of these types of events, the conditions that create them, and how to prevent them (or indeed if they can be prevented). Some of the most interesting and important insights into crisis behavior were put forward by scholars outside the field of economics. This behavior was widely discussed in the network science literature, for example, with a number of interesting results being produced. Network dynamics offer an explanation for the failure of the financial system to properly manage risk: the same highly connected structure which, under most conditions, makes the financial system more resilient, can, under certain conditions, make financial contagion spread more rapidly and widely, and thus worsen a crisis.

While the topic of self-organized criticality has gained some attention in the literature, with a few authors seeing it as possibly a major step forward, this research is still of quite limited influence. SOC is promising as a way forward in addressing questions about dynamics in micro- and macro-economic behavior, and crashes in particular, because it seems to offer an explanation (or set of them) for how a system which is organized with stability as a primary goal can still be subject to unexpected, costly, and difficult-to-predict events: if some complex ongoing process is required for the system to achieve optimality, then the possibility of uncontrollable events will always exist since this dependence makes the system sensitive to small changes.

To sum up the motivation behind this project, It has long been of interest to me to explore ABM and other non-standard economic models which can give some insight into the highly dynamic economic behavior that we witness in the real world. Self-organized criticality seemed to be a particularly powerful framework in which to analyze economic behavior, and the SOC literature contains reference to criticality mechanisms that could provide the basis for a rich model of a dynamic economy that capture a number of significant features: variability in asset prices, aggregate income, and other key measures, crash-type behavior involving large changes in such measures across the whole economy, optimization under various regimes of agent learning,

The Basic Model

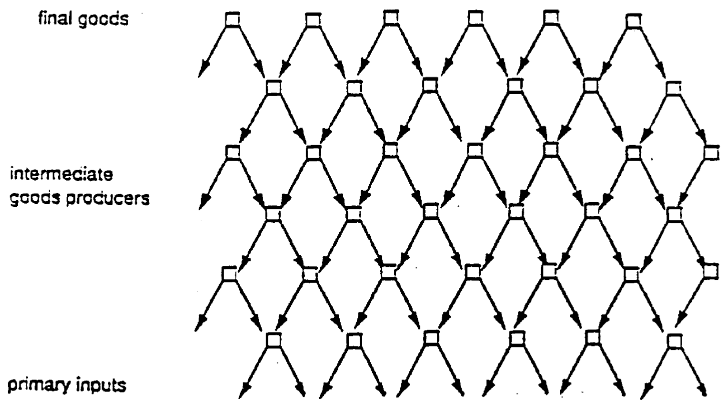
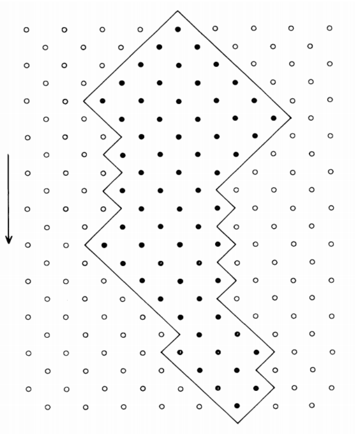


Figure 1 - Basic PDC Model network structure (taken from [2])

Bak and Woodford propose a criticality mechanism which uses an existing model in the economics literature. They adopt a model taken from work by Dhar and Ramaswamy - a variant of a PDC (Producer Distributor Consumer) model [3]. I chose this model as a starting point because it is relatively simple (an exact solution exists for the distribution of cascade sizes) but suggests many improvements, and even in its most basic version still yields somewhat interesting behavior. In this model, Each agent (excluding raw materials producers at the bottom of the ordering network) have two downward connections to suppliers or producers. Consumers are connected to first level producers, who are connected to second level producers, and so on down to raw materials producers at the bottom. In each period, some consumers attempt to make purchases from final goods producers. If the producer has enough inventory to fill the order, they do so, and the product is consumed - the producer's inventory decreases by one unit, and the income in this period increases by one unit. When a producer's inventory goes to zero, they attempt to purchase input goods from their suppliers in order to produce more goods for sale to their buyers. Once again, if the supplier has inventory, they fill the order, and if not, they place an order with producers at the next level down for input goods. If a producer is at the bottom (raw goods producer), they do not purchase input goods - rather, they simply increase their inventory if necessary to fill orders (in other words, we assume that raw materials are always readily available).

This networked agent structure, though fairly simple, can still yield interesting, and critical, behavior. The criticality mechanism depends on inventory availability and local information. Producers hold inventory to sell to other higher-level producers or to consumers. In order to manufacture more manufactured goods inventory, producers must have inventory of both manufacturing inputs (each agent has two downward connections, and we assume for producers that each connection represents one input good that is necessary for production). In this model, producers order more inputs when their inventory is exhausted, which is obviously not fully realistic, since firms will not in general wait until they have zero inventory to obtain more inputs for further production. This is one of the many simplified features present in this initial version of the model that needs to be improved in future research.

Usually final goods producers will have available inventory to meet consumer demand in the same period. Occasionally, though, unavailability of inventory at the top of the network (final goods producer) will result in a cascade of orders down the network if producers at more than one level have to initiate orders for more input inventory in order to meet demand. In the extreme case, a consumer demand of one unit can result in a cascade of orders that go all the way to the bottom of the producer network (raw goods producers). The maximum cascade size is given by $\sum_{i=1}^{L} \quad 2^{L} = 2^{L+1} - 2$. If the network depth (number of production levels) is 4, for a example, the maximum cascade size is 30. Maximum cascade size increases exponentially with network depth, so even with a relatively small network (say L = 10), we expect to occasionally witness cascades of hundreds of orders. This is similar to the sandpile SOC model, in which the addition of a single grain of sand can destabilize the pile, causing an avalanche involving a large number of grains. This model is a variant of the Beer



Cluster of sites in a typical avalanche formed under the evolution dynamics of model *A* in two dimensions. Solid circles denote sites that have toppled. This cluster has duration 22 and mass 64, and the random walks that make the cluster boundary are shown.

Figure 2 - Example cascade shape (taken from [3]

Game in the System Dynamics literature. The complexity of the behavior is a result of information locality. Agents only have information from their immediate neighbors, and cannot base their ordering decisions on explicit knowledge decisions that have already been made further up the chain of producers. Therefore small changes in demand (or, as I will describe further in the discussion of experiments, even no changes in demand) can result in large period-to-period jumps in income (order delivery).

Extending The Model

I made a number of changes and extensions to the model. The first is to ignore spatiality. Dhar and Ramaswamy [3], and Bak, Chen, and Woodford [2], discuss a purely analytical (non-simulated) use of this model. Implementing the model in simulation means that we do not have to keep the model as simplified and restrictive, because here we do not require a model with a purely analytical solution. In the original model, an agent in one layer is connected only to its left and right neighbors in the next layer. In my model model, I preserve the degree of connectedness (each agent is connected to two other agents), but do not require that agents connect only to their spatial neighbors. This results in a more realistic model, because in the real economy agents do not interact only with spatial neighbors; modern economies are, in fact, highly non-localized. An important consequence of non-spatiality is that it changes the network configuration from run to run. In the original Bak-Chen model, a choice of grid size would yield a network with a fixed configuration. In our model, the network topology is initialized randomly every time the model is instantiated. Each agent randomly chooses two agents in the next level as suppliers. In the original model every intermediate agent had two outgoing, and two incoming, connections. In our model, every agent has two outgoing connections, but the number of incoming connections will vary. So if the user chooses N, L, and $P(C)$, the number of levels and agents, and the distribution of agents among the levels, will remain the same with each run; but the wiring of the network (which agents are connected to other agents) will change.

Second, we allow more variation in the distribution of agent types. In the original model, the number of consumers is precisely determined by the grid size. Consumers are one layer in the model, so with an $Mx\,N$ (width x height) grid of agents, the number of consumers will be exactly M. This means that the deeper the agent network, the smaller the ratio of consumers to producers. This is also not very realistic. In reality, consumers usually far exceed the number of producers. As we will see further in experiments, this tends to produce a model in which demand coverage is high, even if the distributor/producer network is inefficient. In a model with a large number of producers relative to consumers, the network will usually be able to meet demand even when the network is very deep, which means that delays are longer and the maximum cascade size is much larger. In my model, I allow the user to specify a total number of agents (N), a number of production levels (L, the depth of the agent network in addition to the consumer layer) and a proportion of consumers ($P(C)$). The remaining portion of the network is distributed evenly among the other production levels. So if$N = 1000$,$L = 5$, and $P(C) = .5$, 500 agents will be consumers, and there will be 100 producer-distributors at each of the five production levels.

Third, we allow demand to be set through an order probability parameter, $P(O)$, which represents the probability that each agent will place an order in each period. If $N = 100$, one would use a $P(O)$ of .1 to get a period demand of 10 units. In the original model, the probability of an order in each period is fixed.

These previous extensions are all variations of the same underlying model. I consider the possibility, though, that a single criticality mechanism is not sufficient to achieve crash behavior. The PDC model generates behavior in which income is variable and oscillates around some mean, but this pattern of behavior is not enough to describe a crisis-type situation. In the real economy, we typically observe income (GDP) growing gradually. Averaging reduces noise in the signal, and GDP is calculated quarterly or yearly, so typical reported GDP numbers display a small amount of noise. Fig. N shows GDP data from 1974 - 2014, and we can see that there is some variability to GDP on a short (monthly) timescale, with a steady upward trend for both nominal GDP and real (adjusted for inflation) GDP. Both GDP numbers take a noticeable dip in 2008, reflecting the significant reduction in economic activity caused by the 2008-2009 financial crisis. It seems reasonable that the criticality mechanism in our basic model would be sufficient to produce the kind of noisy

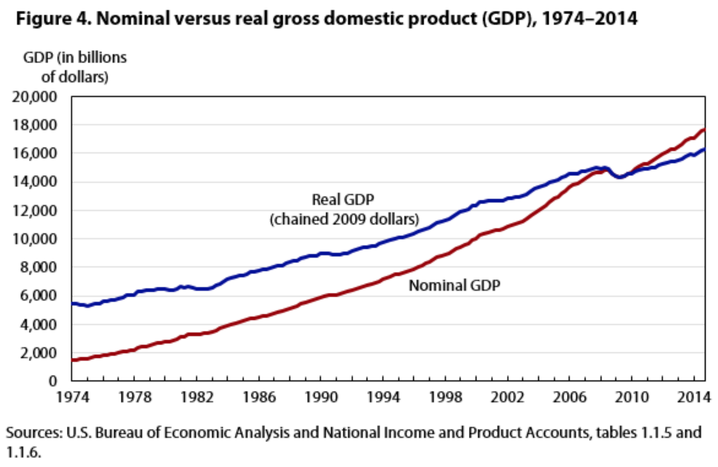


Figure 3 - Plot of GDP from 1974 - 2014 from Bureau of Economic Analysis

short-period behavior we see in real GDP data, but it was not self-evident that this mechanism, by itself, would also produce periodic crashes in GDP - that is, a short-term decrease in GDP that is much larger than the average noise.

A brief discussion of the exogeneity problem is warranted here. As discussed, many economic models assume an equilibrium and achieve dynamic behavior by introducing an exogenous shock (one from outside the model) to demand or some other factor. The first problem is that this is not realistic. Macroeconomic changes almost always arise from within the system, that is they are endogenous and not exogenous. Exogeneity is not completely unrealistic - the oil price shocks of the 1970's are perhaps a good example of an event that can be thought of as exogenous. These rapid changes in price had their source in change in oil output that had a strong geopolitical component, and this factor is difficult to incorporate into an economic model. But such events are the exception rather than the rule - I believe that one sign of a good model is it should be able to produce the macroeconomic events that interest us by mechanisms contained in the model, and not be forcibly introduced from outside.

A key purpose of this project is to begin exploring the possible mechanisms for generating macroeconomic crash behavior endogenously. Sirghi and Dumitrescu [6] develop a similar model to the one I develop here with the same question of economic crisis behavior in mind. They adopt a PDC-type model, but also introduce dynamic rewiring of the agent network based on the profitability of connections in the agent network. I adopted this scheme but slightly modify the fitness function and rewiring conditions, as explained in more detail later in this paper, in the context of experiments. Each agent keeps an order history with each of its suppliers, and if the fulfillment rate of any suppliers (number of filled orders out of the last 5) is less than 60%, the consumer agent abandons the supplier connection and connects to a new supplier. Their results were intriguing but very sketchy, so one goal of this project was to further explore the details of this type of model configuration.

Python ABM Framework

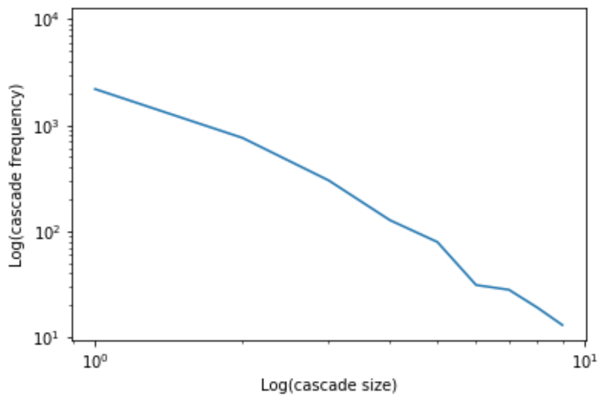
Though Bak and Woodford do not discuss a fully agent-based specification for the underlying criticality model, it (like most economics models) is obviously well-represented in an ABM framework. The basic model could be implemented outside of an ABM context, but the desired model extensions, and particularly the addition of an agent learning component, strongly indicate the use of an ABM. Eventual development work, such as introducing a labor market (producers purchase labor from other agents) or the price mechanism, would be virtually impossible without using an ABM.

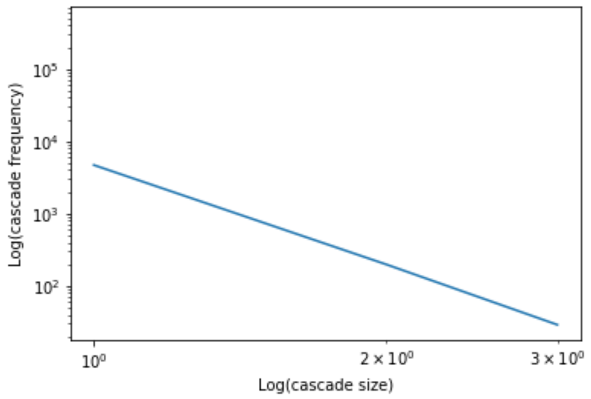
For building the ABM we chose to use the mesa package for Python. Mesa is intended as an alternative to MASON (Java) and Repast (C++). The learning curve is less steep, and the initial model setup much easier, than for these alternative ABM packages. In Python, the modeller also has easy access to a huge ecosystem of existing packages and functionality. This is one of the drawbacks of Netlogo - though the model library is vast, NetLogo is a stand-alone tool, and the body of existing code is small. A motivating factor in initially seeking a tool other than NetLogo for ABM work is the possible need for running models in a HPC (High Performance Computing) environment. ABMs can quickly because extremely computationally complex, and it is easy to imagine a complex ABM model with a large number of agents requiring this level of resources. Dr. Wakefield indicated in past discussions that he had attempted to bridge this gap with NetLogo by having William Garrett, who manages the Portland State HPC resources, explore the possibility of recoding/recompiling NetLogo (or parts of it) to make it take advantage of parallelization, but this proved infeasible. Using mesa should address this problem: there are a number of parallelization libraries available in Python, it is well-supported in nearly every HPC environment. Because running the model can be decoupled from the front-end graphical visualization, any mesa model can be run without an Xserver or other graphical environment being present or accessible. Additionally, the implementation of mesa visualization tools (in a web server), in principle allows the visualization components to be viewed on a user's local machine while the model runs on a cluster or other HPC.

Verification

Implementing the model presented challenges along the way - though there were a few relevant references in the literature that were helpful, there was no clear template to work from in implementing the model in a python-based ABM, and my extensions to the model were, as far as I could tell, original and thus not discussed in detail anywhere else. The first key verification issue was confirming the network structure. In early implementations there were small errors in creating the network. To correct these I implemented a verbose mode in which agents printed their suppliers, inventory status, and other key variables, instantiated the model, and checked this output in detail to confirm that the network structure that was being produced was, in fact, the one I thought I was producing. Another key issue was getting the ordering mechanism, and cascade size tracking, working correctly. Producer and distributor agents keep a list of output orders that need to be filled. When they fill these orders, they transfer product to an agent at the next level up in the network. But when orders are filled, the receiving agent must know which of their inputs the incoming product represents (we require that the producing agent have inventory of both inputs in order to manufacture output). This seemed straightforward but the implementation proved tricky. More development time was required, and because the implementation was non-trivial, more time was also spent in verifying that the ordering and order filling mechanism (the heart of this component of the model) was working as expected.

A related issue was implementing the tracking of cascade sizes, and making sure that they were being tracked correctly. Cascades are tracked by attaching an cascade ID to a series of orders - if an agent attempts to buy product that is not available, initiating an order (and possibly a cascade of orders), a new cascade ID is generated and attached to this initial order and any subsequent orders that it generates in turn. This required me to experiment with different data structures to identify the right mechanisms. One prominent feature of self-organized critical systems is the cascade events exhibit a power law distribution, characterized by a linear relationship between the logarithm of event size and the logarithm of event frequency. Figures N and N show the results of experiments which helped to verify that cascade sizes were being tracked properly, and more generally that the criticality mechanism was working as expected. The longer the simulation, and thus the more cascade events in our sample, the more closely the distribution approaches an ideal power-law distribution.





Figures 4 and 5 – Loglog plots of cascade frequency vs. cascade size showing power-law distribution

A final key verification issue (or set of them) was presented by the dynamic network rewiring. Agents track an order history with each supplier, from which they calculate a simple fitness function (proportion of the last five orders filled on demand). At first I was at times allowing agents to rewire to suppliers in a non-adjacent level. While in future work this might be an interesting feature, for now I wanted to preserve the level structure of the Bak and Woodford model. This and other elements of rewiring presented some challenges in implementation, but by the end of the term I was confident that the rewiring mechanism was working as I expected, and that when the simple criteria for rewiring is replaced with a more complex decision process using a neural network or some other similar model that can be trained with agent and model data, the rewiring process will proceed correctly once the decision is made to find a new supplier connection.

Experiments

Exploring the Parameter Space

Once a working version of a basic model was developed, an obvious next step was to get a general understanding of the model behavior across the parameter space. As with most ABMs, the size of this parameter space is large, roughly10^7. At an intermediate stage of model development, even before the model was finalized, it was important to identify the key regimes of behavior, both in order to identify more useful experiments, and also to aid in further model development. This is, in part, an illustration of the fact that in practice model development is almost never a process that proceeds in orderly and well-defined steps from a comprehensive plan of development. While it is undoubtedly best practice to identify key research goals before beginning model development, and to thoroughly plan the process, the development process almost always involves back and forth between design goals and insight gained from actually working with the model once implementation has begun.

The first behavioral testing was simply to run the model with a small number of agents, $N <= 100$ to make sure I could do multiple runs very quickly while the model was still being verified. Figure 6 gives an example of a model run with a small number of agents. We can see that in this scenario the income is extremely variable from period to period. This is not very

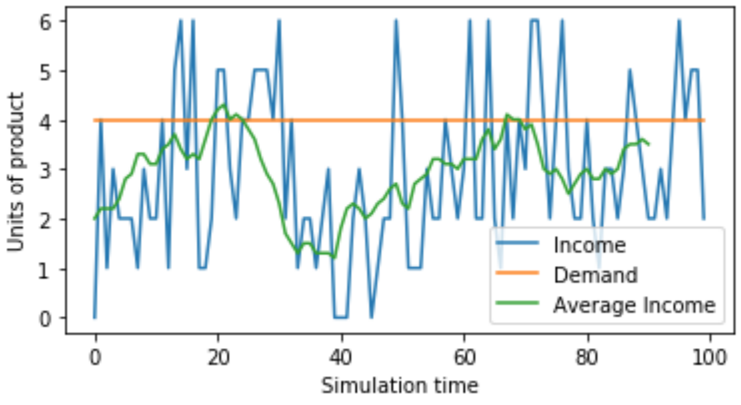
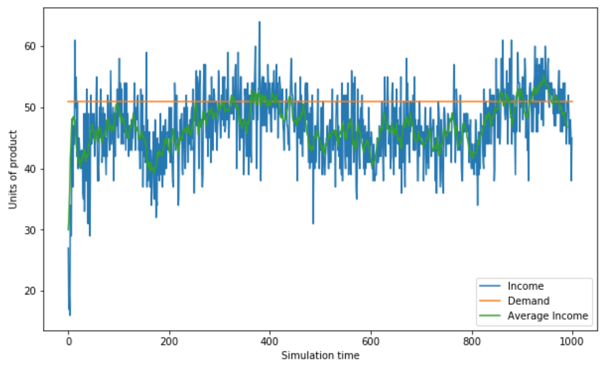
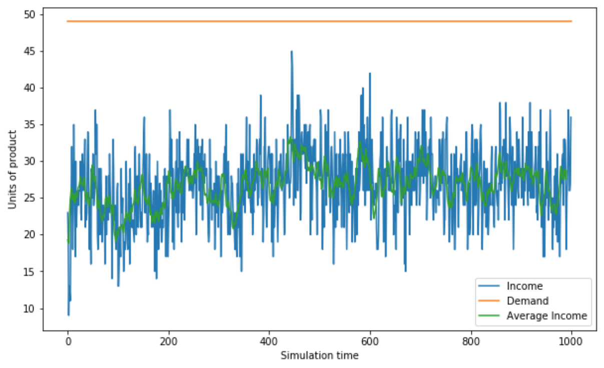


Figure 6 – Model behavior with small number of agents

realistic behavior, but it shows that period income depends heavily on the underlying dynamic of orders across the network, which are difficult to predict.

Next I increased the number of agents to 1000 and tried $P(C)$ values in the middle of the range (~.5) and low values of $P(O)$, reasoning that this would give a manageable ratio of consumers to producers and a demand level that the network could cover. Under this range of parameters, the model still ran quickly and was starting to show interesting behavior. An obvious question was how sensitive the model is to small changes in the network depth. As previously discussed, the maximum cascade size varies exponentially with network depth (L). Therefore as we increase the number of production levels, we expect the network to quickly become much less efficient at meeting some given level of consumer demand. The following figures show this strong dependence of consumer demand coverage on number of production levels.





Figures 7 and 8 - Two simulations runs with N = 1000, P(consumer) = .5, and P(order) = .1. For the first run, L = 3, and for the second, L = 4.

All parameters are the same between these two simulation runs, except that for the second run we increase L (number of production levels) from three to four. This change results in a significant reduction in consumer demand coverage, falling from 91.9% with three production levels to 54.5% with four levels. Going from three to four production levels more than doubles the maximum cascade size from 14 to 30, so though not all consumer demand will requiring ordering, there is the possibility that some orders will take much longer to fill. We see this reflected in the large decrease in demand coverage going from three to four production levels. I performed similar experiments with different values of P(C) - higher proportion of consumers will give higher demand with fewer suppliers to cover it. I am not including output for these experiments since they are similar to figures N and N for comparing different network depths.

I also performed simulations with larger numbers of agents (10000 and higher). Fig N shows model output with a large number of agents$N = 10000, L = 3, P(C) = .7, P(O) = 1$, with rewiring). We can see that with a larger number of agents, demand is higher, and income is less noisy. With N=10,000, typical run time for a 1000-step simulation was ~2 minutes on a several-year-old MacBook pro with 2.7ghz core i5 CPU and 8GB of memory. So, in this simple configuration, we can easily explore the behavior of the model with fairly large N and number of time steps. Future model development, and particularly the introduction of more sophisticated learning mechanisms, is of course expected to dramatically increase computational complexity, very likely requiring a much greater level of computing resources to run longer simulations with large agent numbers. This is why the ability to run the model in a HPC environment is of particular interest.

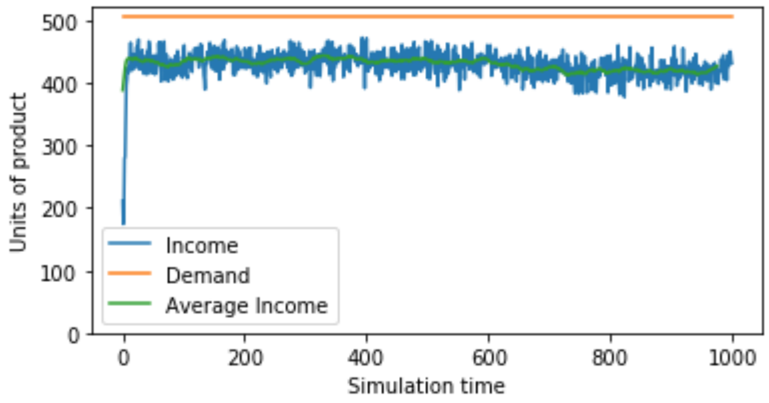


Figure 9 – Model behavior with large number of agents (10,000)

Visualization

Mesa includes a well-developed visualization component which allows real-time, interactive visualization. The package authors attempted to replicate the functionality of NetLogo, in part to widen the appeal of the package.

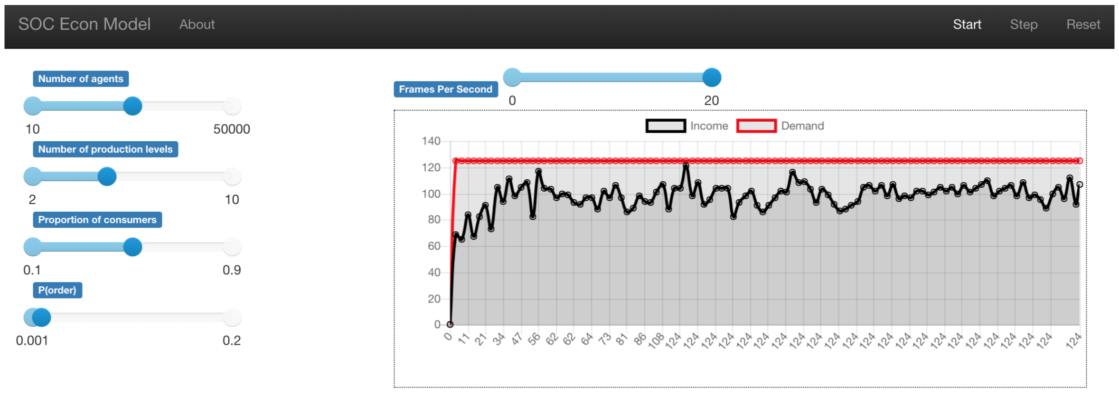


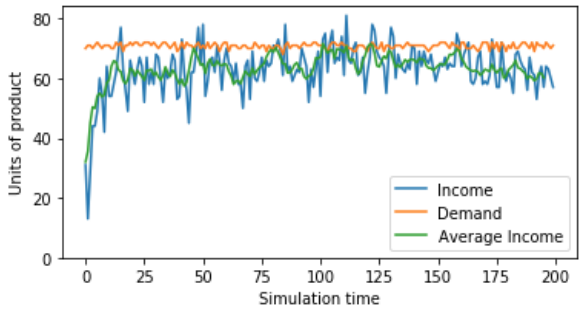
Figure 10 - View of browser-based model visualization in mesa

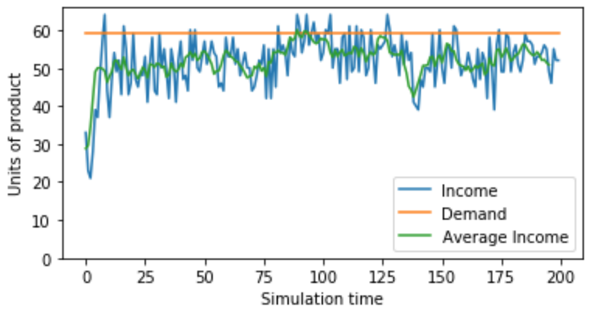
Though developing the visualization component of the model was time-consuming, in part because the mesa documentation is minimal, it seemed worth the time get a working visualization component in order to make it easier to explore the model behavior, identify parameter combinations of interest, and perform experiments. It was evident from the early stages of the project that there would be many fruitful potential developments of the model that would be outside the scope of a single term project, but I felt it was important to get a working visualization component in order to get a better understanding of how the model was performing. The interactive model allows the following choices of parameters:$10 <= N <= 50{,}000; 2 <= L <= 10; .1 <= P(C) <= .9; .001 <= P(O) <= .2$. As in NetLogo, the user can adjust the simulation time to speed up or slow down the simulation, and can also advance the model in single steps to more closely examine the behavior over simulation time.

Testing the Income-Demand Relationship

I also wished to explore the variability in income, and to what degree it depends on changes in demand. This is an important question because many models which exhibit this type of behavior still require some small changes in demand in order to see significant income variability. In the real world, demand is never exactly constant, but this model allows us to test whether exactly constant demand will still yield interesting income behavior. I wished to test whether the variability in income depended on the presence of small changes in demand (i.e., that order fulfillment across the network was amplifying variations in demand). A short discussion of the way in which demand is determined in the model is relevant here. In the early development stages, $P(O)$was used in the following way: at each step, each consumer would have a $P(O)$probability of initiating a purchase (order). More specifically, if a randomly chosen number on (0,1) is less than $P(O)$, that consumer attempts to purchase a unit of product in that step. Under this procedure, the mean of consumer demand over many model iterations will (in the limit) be $p(order) * N$. Demand will follow a binomial distribution, $B(N, P(O))$. Under this model configuration, income was variable and this variability could be quite high under certain sets of parameters, but demand was also quite variable. Therefore, it was possible that the changes in income from period to period were simply due to similar changes in demand. A key behavior of interest is changes in income which are unexpectedly large given small demand changes. So it was necessary to further explore the relationship between changes in demand

and changes in income from period to period. This would act as an additional verification step, but also as an experiment in its own right.





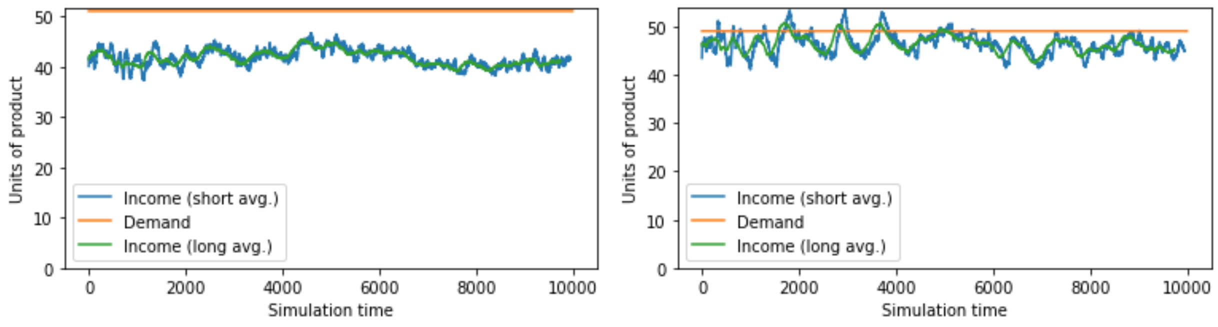
*Figures 11 and 12 - Results of experiments comparing income behavior under fixed and variable demand condition*s

Figures N and N show the sample output from such an experiment. In one case we allow small changes in demand, and in the other demand is fixed. The income behavior does not appear to differ significantly between these two demand regimes. Full testing of this comparison will statistical comparison of the results of multiple runs under each configuration, but these early results strongly suggest that even small changes in demand are not required to maintain highly variable income from period to period, and this finding is potentially of significant research interest.

Network Rewiring

Following the example of Sirghi and Dumitrescu, I wished to develop the model to include more than a single behavior pattern of interest. A high-level goal of this project is, eventually, to find a model configuration that includes the presence of periodic large crash events. The above authors claim to witness this type of behavior in a model very similar to ours, shown in figure 15. There are a couple of small differences that should be pointed out. First, Sirghi and Dumitrescu use a rewiring method which uses a slightly different fitness function than I adopted. Both methods, though, make a rewiring decision based on a proportion of recent orders that were filled on demand. In my model, each agent keeps a history of order fulfillment with each of its suppliers. If the supplier has not been able to fill at least a certain proportion (60%) of the last five orders, the agent removes this connection and connects to a random new supplier in the next level down.

.Though full exploration of the effectiveness of the networking rewiring is reserved for future work, I performed some initial experiments to see if they suggested that the rewiring was improving demand coverage. The (eventual) goal of the model is not necessarily to solve the problem of optimal distribution network structure - this topic has been addressed in detail elsewhere. I wish to identify a reasonably effective network optimization method not to answer practical questions of ideal supply chain structure but to incorporate a second mechanism of criticality (or at least add another dimension of behavioral complexity).



Figures 13 and 14 – Demand coverage without and with network rewiring

Figures 13 and 14 show the results of two simulation runs, one with network rewiring and one without. On the one hand, it seems that the version of the model with dynamic rewiring does perform better - average demand coverage with rewiring is 94.6% compared to 81.8% without rewiring. We do not see the behavior we would expect from optimization taking place, though, which is that the demand coverage should gradually improve over the run. In the simulation with rewiring, the demand coverage starts higher and stays higher. It occurred to me that this could be due to randomness in the network initialization - even if the parameters do not change, the network configuration will be slightly different with every run of the model, so perhaps these results just reflect this variation in network structure. I tried to control for this effect by using a random seed to fix the outcome of any random functions from one run to the next, but am not entirely confident that the network structure is exactly the same from run to run. Thorough testing of the effectiveness of the networking rewiring would involve performing large number of simulation runs with and without rewiring and comparing the results statistically. I decided not to spend too much time on this testing at this phase of the project, because this simple learning/rewiring mechanism is intended as a placeholder for a better mechanism to be developed later. These initial findings call into question the results discussed in [6]. Those authors argue that, after a few dozen model steps, their network is achieving an optimized meta state in which demand coverage is very high. They do not discuss their model or results in enough detail, though, to confirm that this is the case. Their results are also consistent with a situation in which the model exhibits some warm-up time and eventually goes to consistent high demand coverage under certain parameter choices. More work is needed to understand and test these results. Closer examination of their results (fig. N) reveals that they are still introducing an exogenous demand shock, so their apparently crash behavior can be explained as simply an artefact of artificially increasing demand in one period - if demand goes up 5% from one period to the next, and nothing about the order fulfillment network and inventory levels is different, then of course we would expect a temporary 5% reduction in demand coverage until the orders triggered by this increased demand can be filled by the network.

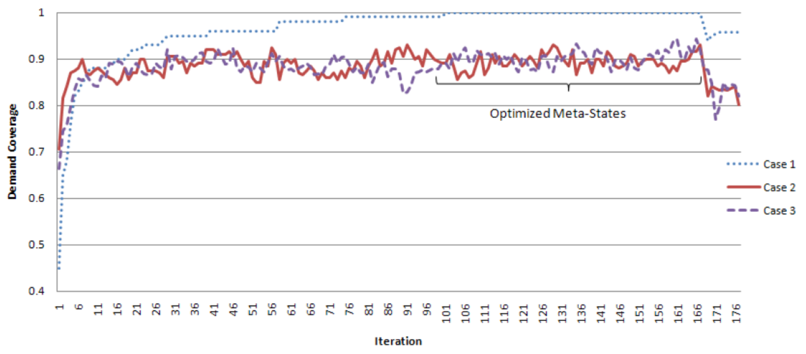


Figure 15 – Claimed optimization behavior showing reference behavior pattern (taken from [6]

A final note on model testing. A fully developed simulation study would include a validation step. I reserve this step for future work. In order to properly validate the model, it would first need to be calibrated so that the results had some meaningful relationship to real quantities. This will certainly be possible, and is a key topic of future research, but was beyond the scope of this initial project. There are a number of other model developments which I would like to put in place to better capture a variety of key patterns in highly dynamic economic behavior, and before the model is further developed in this way it would not be meaningful to perform this calibration and validation.

Further Research

The initial results presented here were promising, and they suggested numerous topics for future research. The first area of development is to continue to extend the PDC component of the model. There are a number of possible extensions, all of which would make the model more realistic. Some of the more obvious are:

- Allow variable ordering by producers, based on recent demand history, instead of producers ordering a fixed number of units
- Add population growth - this would also increase realism, and could introduce new dynamics as the network size grows and undergoes reconfiguration
- Add variation in producer size (manufacturing output) and number of connections to other agents (perhaps according to a preferential attachment scheme)
- Profitability. In this initial version, I do not include production costs explicitly, but I would like to introduce some variation in cost structure, and track profitability. This would have many advantages for model behavior, not least that firm failure would make the network structure more dynamic and possibly allow endogenous cascading failure events
- Make rewiring probability and order history length user parameters

In addition to these extensions to the basic PDC portion of the model, we can identify another category of development, adding other important (possibly critical) mechanisms which are key features of real economies, and which would couple with the producer-distributor structure to introduce new behavioral dynamics:

- Confidence. We know that confidence is a key factor in economic behavior. Changes in confidence can significantly contribute to major macroeconomic events (such as bank runs, or in a modern version such as the events of 2008-2009, a credit crash that significantly reduces overall economic activity
- Price mechanism. This is another fundamental feature of economic activity. Though I reject equilibrium as a governing principle in economics and economic models, it is undoubtedly true that market mechanisms do work to equalize supply and demand, and this this is in part how prices are determined. Of particular interest is information locality - not enforcing global price equilibrium but allowing prices to vary based on local information and conditions.
- Labor Market. In the basic model, I ignore the sources of income. We assume consumers can purchase goods when they choose to, and this is an obvious oversimplification. Introducing the labor market would capture another dynamic element, one which might also contribute to non-equilibrium behavior and large macroeconomic events.

These are just some of the economic mechanisms that a more robust model could contain. We are always simplifying when we build a model, and it is important not to gratuitously introduce complexity without justification - simple models can still give good results, and overly complex models can seem to perform well under certain conditions, but fail to generalize well, and to give misleading or useless results. But economic behavior in the real world is so complex that it seems reasonable that a model which will capture key features of this behavior will need to contain multiple mechanisms of the type discussed above. Another criticality mechanism I wish to touch on is autocatalysis. Though I took a different direction with the underlying criticality mechanism for our model for this project, there is still reason to believe that autocatalytic sets and their behavior have a strong analogy in economics, and that modeling this mechanism could very well provide new key insight into economic behavior and the view of economies as complex adaptive systems.

Finally, it is of great interest to develop the machine learning portion of the model. The learning component contained in the current version is, admittedly, a sort of placeholder for something better to be developed later. If nothing more than as a proof of concept, I wanted to get a very basic learning mechanism in place in an early version of the model. Though in this version of our model we do not see strong evidence that the learning component is significantly optimizing agent behavior, now that the basic structure of dynamically rewiring the network based on some decision criteria has been put in place, I can develop this learning component in a number of ways. Order history or profitability could be used to a train a neural network, whose output represents a decision of whether to rewire or not. There are many ways to introduce more sophisticated learning mechanisms, both to give richer behavior, and to test hypotheses related to economic competitiveness, information availability, etc. Coupling ML with an agent-based model is an exciting new area of research, and this project was valuable, in part, for enabling an exploration of the basic structure of such a model and what a more fully formed version might look like.

Summary

An underexplored application area of self-organized criticality is in economics, and in analyzing economic behavior as a complex adaptive system. I identified and implemented a useful candidate mechanism from the literature, and introduced a basic learning component with the hope of reproducing crash-type behavior without exogenous shocks. While the version of the model developed here is of limited usefulness without further development, calibration, and validation, this project was extremely worthwhile. It served as proof of concept that a simple critical mechanism can still yield interesting behavior. Our findings suggest that even small changes in demand are not necessary to cause significant variability in aggregate income from period to period, and this finding is promising because it further reinforces the belief that it is possible to develop economic models in which large macroeconomic events can be generated without introducing exogenous shocks (those from outside the model). Particularly when confidence or some other mechanism is combined with the criticality mechanism in this model, I believe that it is possible to produce an agent-based model of economic activity which reproduces, at least, the key features of this aggregate income variability and, in addition, occasional large crash events in which income falls rapidly across the whole economy. Though I was not able to fully reproduce this behavior in this early version of the model, I am confident that that overall structure that has been employed here holds great promise, and that with more work could provide some results of real significance in understanding previously unexplained (or only partially explained) features of real-world economic system.

References

[1] Bak, P., and Chen, K. “Self-Organized Criticality.” *Scientific American*, Vol. 264, No. 1 (January 1991), pp. 46-53.

[2] Bak, P., Chen, K., Scheinkman, J., and Woodford, M. First M. “Aggregate Fluctuations From Independent Sectoral Shocks: Self-Organized Criticality In A Model Of Production And Inventory Dynamics.” National Bureau of Economic Research Working Paper No. 4241 (December 1992).

[3] Dhar, D., and Ramaswamy, D. “Exactly Solved Model of Self-Organized Critical Phenomena.” *Studia Univ. Babes-Bolyai, Informatica,* Vol. LVI, No. 2 (2011), pp. 107-112.

[4] Kauffman, Stuart. "Whispers From Carnot: The Origins of Order and Principles of Adaptations in Complex Nonequilibrium Systems." *Complexity: Metaphors, Models, and Reality*, edited by Cowan, G., Pines, D., and Meltzer, D. Sciences of Complexity, Proceedings, Vol XIX, Addison-Wesley, 1994, pp. 83-160.

[5] Mandelbrot, M. “The Variation of Certain Speculative Prices.” *The Journal Of Business*, Vol. 36, No. 4 (October 1963), pp. 394-419.

[6] Sirghi, A. and Dumistrescu, D. “Self-Organized Criticality And Economic Crises.” *Studia Univ. Babes-Bolyia, Informatica*, Vol. LVI, No. 2 (2011), pp. 107-112.