Implementation of an Extensible, Agent-Based Simulation Program for Barter Economics

Pelle Evensen & Mait Märdin

December 18, 2008

Agent-Based Computational Economics (ACE)

Definition

The computational study of economic processes modelled as dynamic systems of interacting agents.

Agent-Based Computational Economics (ACE)

Definition

The computational study of economic processes modelled as dynamic systems of interacting agents.

What it typically looks like...

- Define privately motivated agents with learning capabilities.
- Construct an agent-based world that defines the interactions between these agents.
- Start the model and let the world develop over time.
- If we are lucky, surprising things will happen!

Agent-Based Computational Economics (ACE)

The objectives of ACE

- To understand why particular global regularities have evolved and persisted in real-world economies. The aim is to create an agent-based model that would generate some particular regularity.
- To discover good economic designs. Agent-based models are used as laboratories to evaluate the efficiency of various economic designs.
- To advance the methods and tools used to create the agent-based models.

Agent-Based Computational Economics (ACE)

The advantages over mathematical modelling

- Agents do not have to be rational.
  Limit the rationality of agents and see what happens.
- It is easy to make simulations with heterogenous agent behaviours.
- Easy to model physical space or social networks between the agents.
- An agent-based model is “solved” merely by executing it.
  We can sit back and see what happens.

Agent-Based Computational Economics (ACE)

The advantages over mathematical modelling

- Agents do not have to be rational.
  Limit the rationality of agents and see what happens.
- It is easy to make simulations with heterogenous agent behaviours.
- Easy to model physical space or social networks between the agents.
- An agent-based model is "solved" merely by executing it.
  We can sit back and see what happens.

The disadvantage

A single run of a simulation does not say anything about the robustness of the results.

Agent-Based Computational Economics (ACE)

The advantages over mathematical modelling

- Agents do not have to be rational.
  Limit the rationality of agents and see what happens.
- It is easy to make simulations with heterogenous agent behaviours.
- Easy to model physical space or social networks between the agents.
- An agent-based model is “solved” merely by executing it.
  We can sit back and see what happens.

The disadvantage

A single run of a simulation does not say anything about the robustness of the results.

Overall

ACE is a young field without well established development standards.

The BARTER Economy

The origin

- Devised by Herbert Gintis and published in 2006; *"The Emergence of a Price System from Decentralized Bilateral Exchange"*.
- A proof of concept implementation in Delphi.
- Aims to find a price adjustment mechanism that would lead to *equilibrium prices* (the prices for which demand equals supply for all goods).

The Barter Economy

The origin

- Devised by Herbert Gintis and published in 2006; *"The Emergence of a Price System from Decentralized Bilateral Exchange"*.
- A proof of concept implementation in Delphi.
- Aims to find a price adjustment mechanism that would lead to *equilibrium prices* (the prices for which demand equals supply for all goods).

The defining properties of the model

- Simple bartering of goods; no money or firms.
- Private prices for all agents.
- Less successful agents are regularly replaced by more successful ones.
- The emergence of equilibrium prices in the long run.

The BARTER Economy

The main parameters of the model

- g number of goods
- n number of agents per good
- p number of bartering periods
- m maximum number of trade attempts in one period

The BARTER Economy

The main parameters of the model

- g number of goods
- n number of agents per good
- p number of bartering periods
- m maximum number of trade attempts in one period

The agent in the BARTER economy

- Produces a single good.
- Has its own price idea for every good.
- Barters the produced good for other desirable goods.
- Only agrees to barter if received value $\geq$ given value.
- Eats the obtained goods.
- The more it eats, the higher the score.

Our goals

Our goals

- Replicate Gintis' BARTER economy model.

Our goals

- Replicate Gintis' BARTER economy model.
- At the same time, re-engineer the proof of concept implementation to an extensible simulation tool for barter economics.

Replication of agent-based models

What do we mean by replication?

The implementation of a conceptual model described and already implemented (the original model) at a previous time.

Replication of agent-based models

What do we mean by replication?

The implementation of a conceptual model described and already implemented (the original model) at a previous time.

The problem

More and more agent-based models are created in natural and social sciences. Most of these models have never been replicated by anyone but the original developer.

Replication of agent-based models

What do we mean by replication?

The implementation of a conceptual model described and already implemented (the original model) at a previous time.

The problem

More and more agent-based models are created in natural and social sciences. Most of these models have never been replicated by anyone but the original developer.

The importance of replication

- Verification—to determine whether the implemented model corresponds to the target conceptual model.
- Validation—to determine whether the implemented model corresponds to and explains some phenomenon in the real world.

Three categories of replication standards

Numerical identity

Showing that the original and replicated model produce the exact same numerical results.

Three categories of replication standards

Numerical identity

Showing that the original and replicated model produce the exact same numerical results.

Distributional equivalence

Showing that the two implemented models are statistically similar to each other.

Three categories of replication standards

Numerical identity

Showing that the original and replicated model produce the exact same numerical results.

Distributional equivalence

Showing that the two implemented models are statistically similar to each other.

Relational alignment

Showing that the results of the two implemented models have qualitatively similar relationships between input and output variables.

Replicating the Barter economy

Verification rather than validation

We can verify whether the original implementation is correct with regards to its description. Validation is up to economists.

Replicating the BARTER economy

Verification rather than validation

We can verify whether the original implementation is correct with regards to its description. Validation is up to economists.

Our method

- Start from the source code.
- Port from Delphi to Java.
- Use the multi-agent simulation toolkit MASON.
- Compare the results.

Replicating the Barter economy

Verification rather than validation

We can verify whether the original implementation is correct with regards to its description. Validation is up to economists.

Our method

- Start from the source code.
- Port from Delphi to Java.
- Use the multi-agent simulation toolkit MASON.
- Compare the results.

Distributional equivalence rather than numerical identity

Our choices ruled out the numerical identity: different implementation language and different random number generator (enforced by the framework).

Replicating the BARTER economy

Verification rather than validation

We can verify whether the original implementation is correct with regards to its description. Validation is up to economists.

Our method

- Start from the source code.
- Port from Delphi to Java.
- Use the multi-agent simulation toolkit MASON.
- Compare the results.

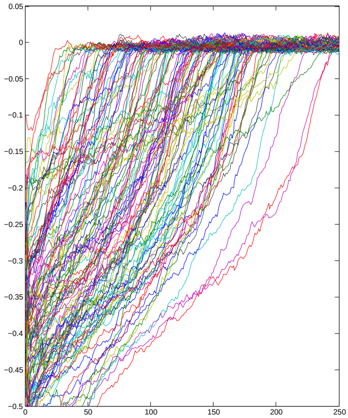
Distributional equivalence rather than numerical identity

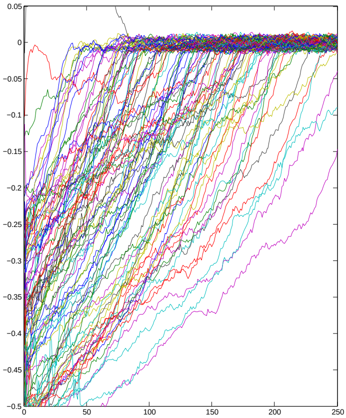
Our choices ruled out the numerical identity: different implementation language and different random number generator (enforced by the framework).

How did it go?

How similar are our programs?

Similar enough?





Testing distribution properties

We want to see if we have distributional equivalence, i.e. if we can distinguish the two programs with regards to the distribution of their outputs.

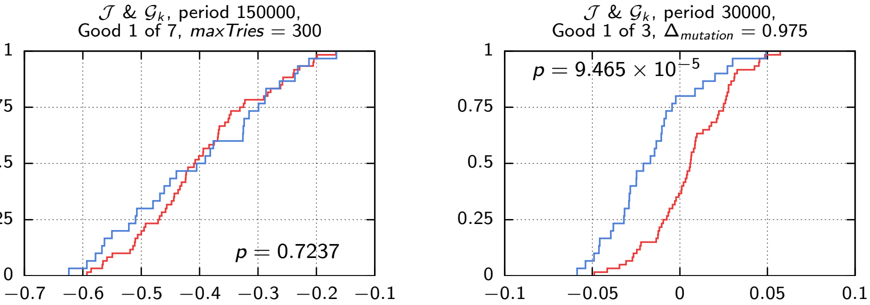
Testing distribution properties

We want to see if we have distributional equivalence, i.e. if we can distinguish the two programs with regards to the distribution of their outputs.

The two-sample Kolmogorov-Smirnov test

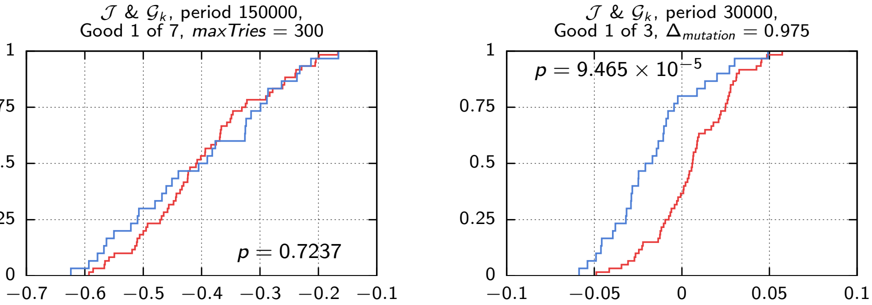
- Gives the probability that two samples are drawn from the same distribution.
- Only assumption is that the distribution is continuous.

Empirical distribution functions



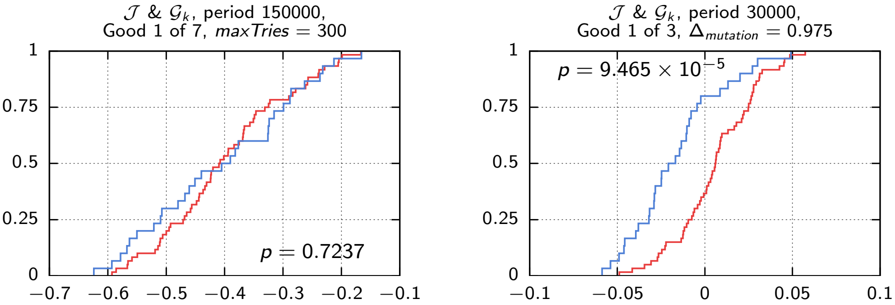
- Curves show the proportion of samples (y-axis) that are less than the x-axis value. E.g. about half the values of the "red" program in the left chart are less than -0.4 & 75% of the values of the "red" program to the right are less than 0.025.
- If we sample from the same distribution, we should expect the difference between the blue & red lines to approach zero as the sample size increases.
- The KS-test gives us the probability that the maximum measured difference is due to chance.

Empirical distribution functions



- Curves show the proportion of samples (y-axis) that are less than the x-axis value. E.g. about half the values of the "red" program in the left chart are less than -0.4 & 75% of the values of the "red" program to the right are less than 0.025.
- If we sample from the same distribution, we should expect the difference between the blue & red lines to approach zero as the sample size increases.
- The KS-test gives us the probability that the maximum measured difference is due to chance.

Empirical distribution functions



- Curves show the proportion of samples (y-axis) that are less than the x-axis value. E.g. about half the values of the "red" program in the left chart are less than -0.4 & 75% of the values of the "red" program to the right are less than 0.025.
- If we sample from the same distribution, we should expect the difference between the blue & red lines to approach zero as the sample size increases.
- The KS-test gives us the probability that the maximum measured difference is due to chance.

Testing procedure

1. Choose a set of parameters.

Testing procedure

1. Choose a set of parameters.
2. Run both programs multiple times with different seeds but same parameters and record the average prices for each good (vs. some reference good) at regular intervals for each run.

Testing procedure

1. Choose a set of parameters.
2. Run both programs multiple times with different seeds but same parameters and record the average prices for each good (vs. some reference good) at regular intervals for each run.
3. Compare the distributions for the two program at some time periods.

Testing procedure

1. Choose a set of parameters.
2. Run both programs multiple times with different seeds but same parameters and record the average prices for each good (vs. some reference good) at regular intervals for each run.
3. Compare the distributions for the two program at some time periods.
4. Consider the programs to not match for the parameter set under test if the KS-test fails spectacularly ($p < 10^{-6}$ or so) for any good.

What's the verdict?

Unfortunately, for 4 parameter sets out of the 20 we have tested, the KS-test gives a very low probability for at least one good and for at least some part of the time series.

What's the verdict?

Unfortunately, for 4 parameter sets out of the 20 we have tested, the KS-test gives a very low probability for at least one good and for at least some part of the time series.

We conclude that our program does not (for all parameter sets) have the same distribution of prices over time as Gintis' program has.

All is lost?

All is lost?

Fortunately, no!

From a user's perspective, we are interested in seeing that we get a similar convergence behaviour on average.

If the distributions don't match, we can do a confidence interval on the difference of means for the two programs.

When we check what the worst $\mu_1 - \mu_2$ is for the 5 periods closest to the stopping point we find that it is on the interval $\{-0.028, -0.021\}$ with 99.9% probability.

In other words; the worst difference we have found between our programs for the parameter sets we have been using is that our program is at most 3% low (-0.028) compared to Gintis' program.

How come?

In the interest of time, we have continued our thesis work after having spent around one working week looking for what may be a bug. Whether it is a bug in our program or in Gintis' program is hard to tell; almost all methods in both programs rely heavily on (pseudo-)random numbers.

How come?

In the interest of time, we have continued our thesis work after having spent around one working week looking for what may be a bug. Whether it is a bug in our program or in Gintis' program is hard to tell; almost all methods in both programs rely heavily on (pseudo-)random numbers.

Even if they did not, we have not had the time necessary to invest in checking that Java and Delphi are using the same floating point arithmetic standard for 64-bit floats. Unless we know that they are behaving identically, testing for numerical identity is pointless.

Generalisations

Generalising Barter

Gintis has published at least one more complex model called GenEqui. GenEqui includes concepts such as money, workers and factories. Should we have implemented GenEqui instead and provided an API to implement Barter?

Generalisations

Generalising BARTER

Gintis has published at least one more complex model called GENEQUI. GENEQUI includes concepts such as money, workers and factories. Should we have implemented GENEQUI instead and provided an API to implement BARTER?

BARTER $\not\subseteq$ GENEQUI

Making an implementation of BARTER that encompasses GENEQUI would force overly complex interfaces onto BARTER.

Extension types

Two types

- Extensions to individual agent behaviours
- Extensions to the the market (the rules that *all* agents have to honor).

We treat them separately in favour of making studies of heterogeneous agent behaviours easy to carry out.

Orthogonality

"Orthogonality guarantees that modifying the technical effect produced by a component of a system neither creates nor propagates side effects to other components of the system."
—Wikipedia

Orthogonality

"Orthogonality guarantees that modifying the technical effect produced by a component of a system neither creates nor propagates side effects to other components of the system."
—Wikipedia

Example

Generalise to include a monetary good so that agents can trade either the good they produce or the money good.

Orthogonality

"Orthogonality guarantees that modifying the technical effect produced by a component of a system neither creates nor propagates side effects to other components of the system."
—Wikipedia

Example

Generalise to include a monetary good so that agents can trade either the good they produce or the money good.

Consequence

One could not implement handling of the money good as a trade agent behaviour without also making sure that the market (and *all* trade agents) also are aware of the new, special purpose, good $\Longrightarrow$ non-orthogonality.

Orthogonality

"Orthogonality guarantees that modifying the technical effect produced by a component of a system neither creates nor propagates side effects to other components of the system."
—Wikipedia

Example

Generalise to include a monetary good so that agents can trade either the good they produce or the money good.

Consequence

One could not implement handling of the money good as a trade agent behaviour without also making sure that the market (and *all* trade agents) also are aware of the new, special purpose, good $\Longrightarrow$ non-orthogonality.

- Making the model more general is difficult if we want to keep the extensions orthogonal to each other.

Orthogonality

"Orthogonality guarantees that modifying the technical effect produced by a component of a system neither creates nor propagates side effects to other components of the system."
—Wikipedia

Example

Generalise to include a monetary good so that agents can trade either the good they produce or the money good.

Consequence

One could not implement handling of the money good as a trade agent behaviour without also making sure that the market (and *all* trade agents) also are aware of the new, special purpose, good $\Longrightarrow$ non-orthogonality.

- Making the model more general is difficult if we want to keep the extensions orthogonal to each other.
- Each extension would either have to handle a very general interface and/or depend on other extensions.

Extension Points

An *extension point* describes a point in a use case where an extending use case may provide additional behaviour.

- Figuring out what should be useful extension points is made difficult by neither of us having a background in ACE.
- We may have provided points that were easy to handle rather than points that are “useful”.

Extension Points, cont.

We explicitly prevent extension by implementation inheritance through making the two main classes, TradeAgent and BarterEconomy, final. All extensions have to be done by providing the agent or market with *strategies*.

Extension Points, cont.

We explicitly prevent extension by implementation inheritance through making the two main classes, TradeAgent and BarterEconomy, final. All extensions have to be done by providing the agent or market with *strategies*.

Advantages

- Assuming the strategies are not too powerful, we can always guarantee that the class invariants of the extended classes hold to the same extent they do in the non-extended case.
- A programmer can not, deliberately or not, violate any assumptions on scope we had in mind. cf. *protection proxy, facet*

Extension Points, cont.

We explicitly prevent extension by implementation inheritance through making the two main classes, $\mathrm{TradeAgent}$ and $\mathrm{BarterEconomy}$, final. All extensions have to be done by providing the agent or market with *strategies*.

Advantages

- Assuming the strategies are not too powerful, we can always guarantee that the class invariants of the extended classes hold to the same extent they do in the non-extended case.
- A programmer can not, deliberately or not, violate any assumptions on scope we had in mind. cf. *protection proxy, facet*

Disadvantage

- The classes can only be extended in the ways we have chosen to be useful.

Concrete extension points

Agent Barter Strategy

Concrete extension points

Agent Barter Strategy

Agent Improvement Strategies

Concrete extension points

Agent Barter Strategy

Agent Improvement Strategies

Market Replacement/Mutation Strategy

Algorithm complexity for the model

```
function MODEL(p, n, g, m)
 for A = 1 to p
  for B = 1 to g
   for C = 1 to n
    for D = 1 to g
     for E = 1 to m + g
      Traded or tried m times
     end for
    end for
   end for
  end for
 end for
end function
```

Parameters

p : periods, n : agents per good, g : number of goods, m : maximum number of barter attempts, c : number of CPUs

Algorithm complexity for the model

```
function MODEL(p, n, g, m)
 for A = 1 to p
  for B = 1 to g
   for C = 1 to n
    for D = 1 to g
     for E = 1 to m + g
      Traded or tried m times
     end for
    end for
   end for
  end for
 end for
end function
```

Parameters

p : periods, n : agents per good, g : number of goods, m : maximum number of barter attempts, c : number of CPUs

Asymptotic worst case complexity

$$O(png^2(m+g))$$

Linear in number of agents and periods but note the g^3.

Algorithm complexity for the model

```
function MODEL(p, n, g, m)
 for A = 1 to p
  for B = 1 to g
   for C = 1 to n
    for D = 1 to g
     for E = 1 to m + g
      Traded or tried m times
     end for
    end for
   end for
  end for
 end for
end function
```

Parameters

p : periods, n : agents per good, g : number of goods, m : maximum number of barter attempts, c : number of CPUs

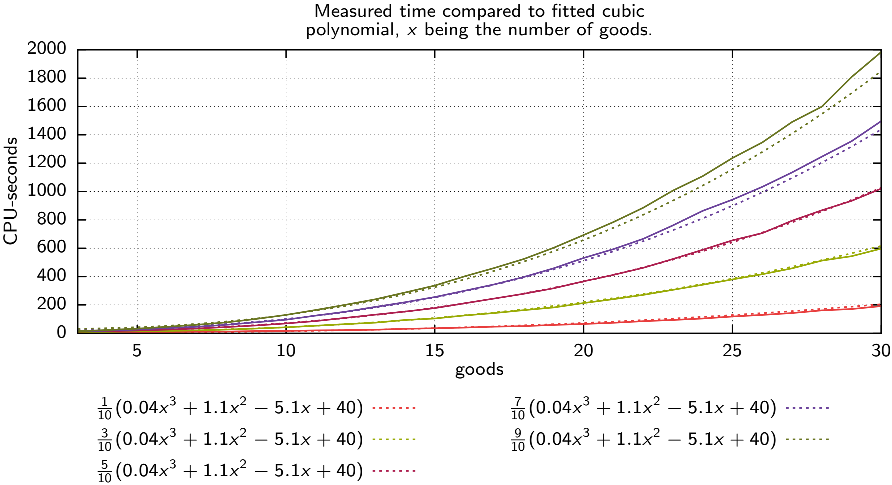
Asymptotic worst case complexity

$$O(png^2(m+g))$$

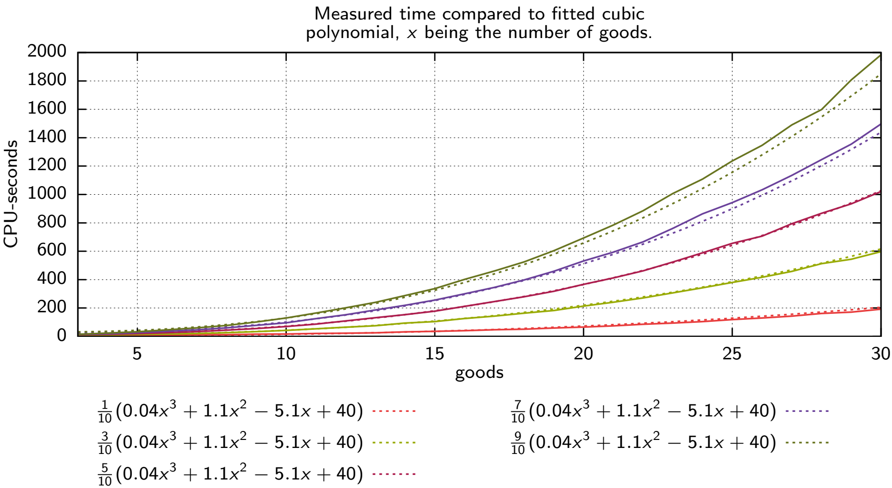
Linear in number of agents and periods but note the g^3.

How far is this from the average time complexity?

Timing the program

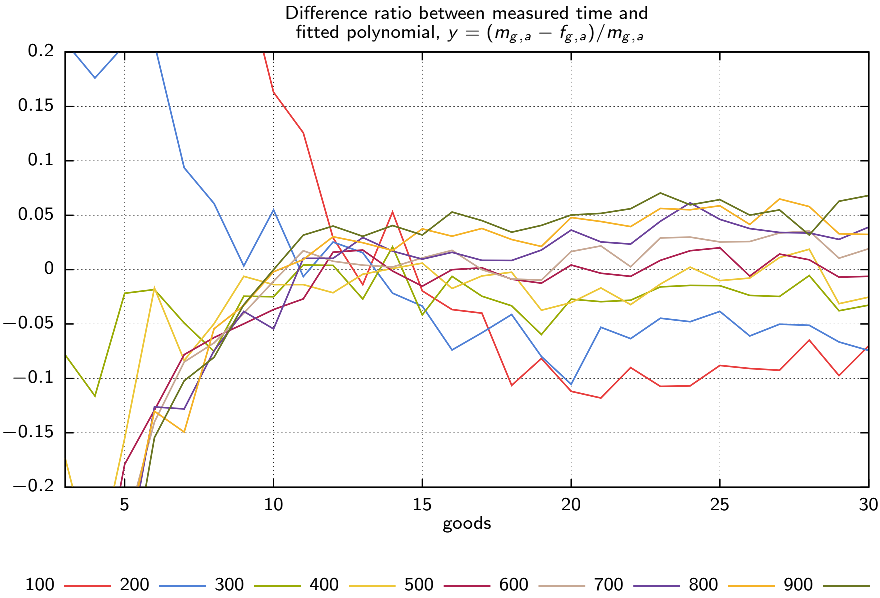


Timing the program



The cubic coefficient is more than one order of magnitude smaller than the quadratic coefficient. Perhaps not *that* bad after all.

Fitting errors



Possible gains from concurrency

Our program is *not* concurrent save for the Swing-thread.

Possible gains from concurrency

Our program is *not* concurrent save for the Swing-thread.

```
function MODEL(p, n, g, m)
 for A = 1 to p
  for B = 1 to g
   for C = 1 to n
    for D = 1 to g
     for E = 1 to m + g
      Traded or tried m times
     end for
    end for
   end for
  end for
 end for
end function
```

Parameters

p : periods, n : agents per good, g : number of goods, m : maximum number of barter attempts, c : number of CPUs

- As seen from the time measurements, the *average* complexity for the D-loop is closer to $O(g)$ (or possibly $O(gm)$).
- The C loop, having complexity $O(ng(m+g))$, is possible to parallelise over n for average complexity $O(c^{-1}g)$.
- Possible average running time of order $O(c^{-1}ng(m+g))$ for the complete program.

Possible gains from concurrency

Our program is *not* concurrent save for the Swing-thread.

function MODEL(p, n, g, m)
  for $A = 1$ **to** p
    for $B = 1$ **to** g
      for $C = 1$ **to** n
        for $D = 1$ **to** g
          for $E = 1$ **to** $m + g$
            Traded or tried m times
          end for
        end for
      end for
    end for
  end for
end function

Parameters

p : periods, n : agents per good, g : number of goods, m : maximum number of barter attempts, c : number of CPUs

- As seen from the time measurements, the *average* complexity for the D-loop is closer to $O(g)$ (or possibly $O(gm)$).
- The C loop, having complexity $O(ng(m+g))$, is possible to parallelise over n for average complexity $O(c^{-1}g)$.
- Possible average running time of order $O(c^{-1}ng(m+g))$ for the complete program.

Possible gains from concurrency

Our program is *not* concurrent save for the Swing-thread.

function MODEL(p, n, g, m)
  for $A = 1$ **to** p
    for $B = 1$ **to** g
      for $C = 1$ **to** n
        for $D = 1$ **to** g
          for $E = 1$ **to** $m + g$
            Traded or tried m times
          end for
        end for
      end for
    end for
  end for
end function

Parameters

p : periods, n : agents per good, g : number of goods, m : maximum number of barter attempts, c : number of CPUs

- As seen from the time measurements, the *average* complexity for the D-loop is closer to $O(g)$ (or possibly $O(gm)$).
- The C loop, having complexity $O(ng(m+g))$, is possible to parallelise over n for average complexity $O(c^{-1}g)$.
- Possible average running time of order $O(c^{-1}ng(m+g))$ for the complete program.

And now for something completely different!

- What does our program look like?

Contributions

What have we done?

- Replicated the original BARTER economy model.
- Provided a new, portable and extensible simulation tool for barter economies.
- Provided means for improved intuitive understanding of the model by adding individual agent visualisation.

Conclusions

Our conclusions

- Replication is important. We found discrepancies between the description of the model and the original implementation.

Conclusions

Our conclusions

- Replication is important. We found discrepancies between the description of the model and the original implementation.
- Replicating an agent-based model is hard. The global model behaviour is sensitive to seemingly small details such as the random number generation, rounding of the floating-point values, or the order of simulated events.

Conclusions

Our conclusions

- Replication is important. We found discrepancies between the description of the model and the original implementation.
- Replicating an agent-based model is hard. The global model behaviour is sensitive to seemingly small details such as the random number generation, rounding of the floating-point values, or the order of simulated events.
- Defining what convergence means is *hard*. If the numerical identity property is not feasible, we need a relevant statistic to say that the original model and the reimplementation behave the same way.

Conclusions

Our conclusions

- Replication is important. We found discrepancies between the description of the model and the original implementation.
- Replicating an agent-based model is hard. The global model behaviour is sensitive to seemingly small details such as the random number generation, rounding of the floating-point values, or the order of simulated events.
- Defining what convergence means is *hard*. If the numerical identity property is not feasible, we need a relevant statistic to say that the original model and the reimplementation behave the same way.
- If we were to do it over, we would first try to achieve the numerical identity at the function level. Then change the design step by step while observing that the numerical identity is preserved.

Thanks for listening

Questions?