Emergent Firms Model ODD

J M Applegate

06.01.2018

Purpose

The Emergent Firm (EF) model is based on the premise that firms arise out of individuals choosing to work together to advantage themselves of the benefits of returns-to-scale and coordination. The Emergent Firm (EF) model is a new implementation and extension of Rob Axtell's Endogenous Dynamics of Multi-Agent Firms model (Axtell 2015). Like the Axtell model, the EF model describes how economies, composed of firms, form and evolve out of the utility maximizing activity on the part of individual agents. The EF model includes a cash-in-advance constraint on agents changing employment, as well as a universal credit-creating lender to explore how costs and access to capital affect the emergent economy and its macroeconomic characteristics such as firm size distributions, wealth, debt, wages and productivity.

Entities, State Variables and Scale

Agents are individuals with preferences for income, θ, production characteristics, $\mathtt{a}$, $\mathtt{b}$ and β, a savings rate, $\mathtt{s}$, and a position in a social network. During the course of the simulation, agents will make utility calculations and chose to change employment, either by starting a new firm or joining another firm, in order to maximize their utility. The model captures at each time step whether or not an agent recalculated its utility values, whether those calculations resulted in a particular change of employment, if the agent was thwarted in making a change, the agent's current effort, wage, loans and savings, and its firm affiliation.

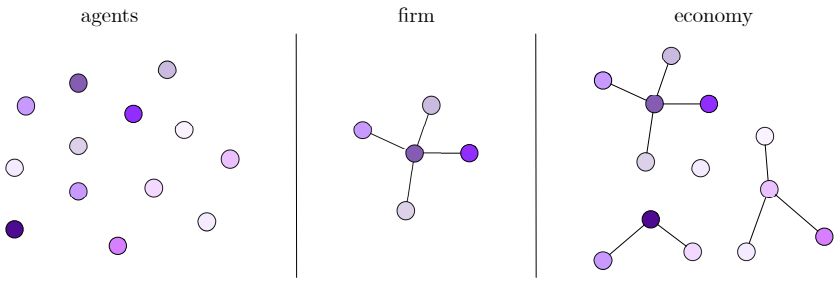

Figure 1: Hierarchical depiction of model entities: individuals compose firms, firms compose the economy.

This agent-level information can be aggregated to describe firm

level information such as size, wage and productivity, and firm level information in turn can be aggregated to describe macroeconomic characteristics of the economy as whole, such as the size distributions of firm populations, mean wage, per capita wealth and debt.

Firms are modeled as star graphs connecting all employees to the firm owner The economy is modeled as a collection of all the star subgraphs and singleton agents, illustrated in Figure 2. Thus each agent has two types of neighbors: those in the social network which define an agent's alternative employment opportunities, and those in the star graph networks which represent an agent's employees or employer.

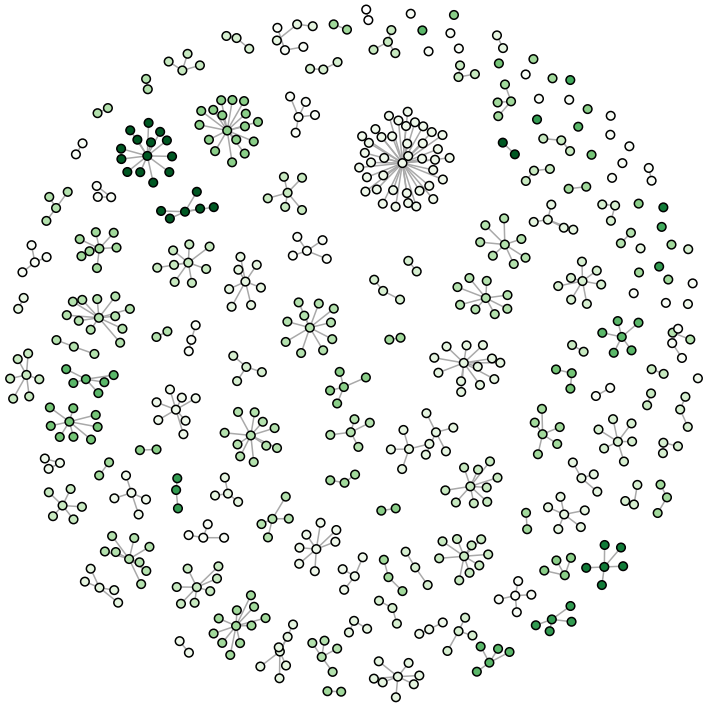

Figure 2: Network graph of an emergent economy consisting of a collection of firms modeled as star sub-graphs connecting individuals. Nodes are colored on employee savings; the more intense the color the higher the savings. Neighbors in this network are either employees or employers.

Process Overview & Scheduling

Each time step agents explore options for utility improvement. Agents decide whether to stay in their current situation, join another firm in their social network, or start their own firms based on which of the options maximizes their utility. The ability to make a desired change requires either sufficient savings or the ability to obtain a loan. After all agents have made changes, firm outputs are calculated and distributed, and loans are repaid. The EF model process is summarized as a flowchart in Figure 3.

To start, the number of agents specified by `N` are instantiated with individual values for parameters θ, `a`, `b`, β and `s`. A random graph is generated to organize the agents as nodes in a social network with a specified range of edges per node. The model then calculates the utility and effort values for each agent as a singleton firm and stores these values, and model setup is complete.

The model activates individual agents with a probability given by `churn`. For each activated agent, the model calculates a Cobb-Douglas utility value for that agent's current position, as well as the expected utility if it were employed by each of the firms employing its neighbors in the social network. These values are compared with the agent's singleton utility calculated during setup, which is the utility for forming a startup. The scenario yielding the highest utility value is the agent's choice: either remain, form a startup or change firms. If the choice requires a change, the model computes the cost of the change, and if the agent has enough savings to cover the costs of the move, or if it can take out a loan to cover any residual costs, the agent's firm affiliation and firm links are changed. An agent cannot take out a loan if he already has an outstanding loan balance. If the agent making a change is the owner of a firm with employees, that firm's ownership is reassigned to another random employee. Wage, effort, savings and loan values are updated accordingly. An agent is considered to be 'thwarted' if it cannot make a desired move.

After all agents have had the opportunity to be activated, the model calculates the output for each firm and distributes a share of that output to all employees and owners, and wage and savings values are updated accordingly. Finally, any agent with a loan will apply their savings toward repaying the loan, loan values are compounded, and savings and loan values are updated.

Initialization & Inputs

The initial values for the model parameters are described in Table 1.

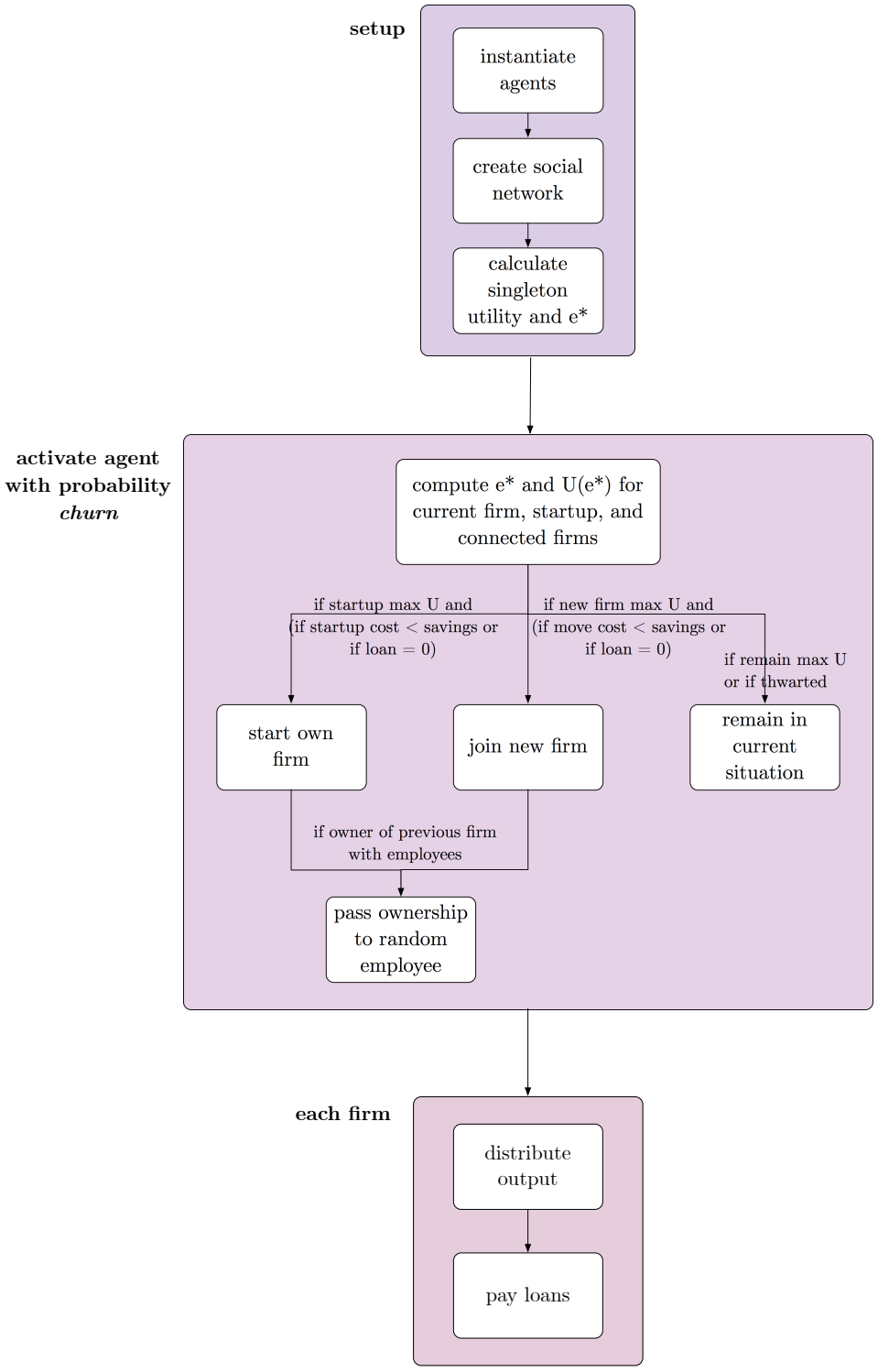

Figure 3: Functional flow for the Emergent Firm model.

Attribute	Description	Value
Agent Variables		
a	effort multiplier in output formula	$\mathcal{U}(0, .5)$
b	exponential effort multiplier	$\mathcal{U}(.75, 1.25)$
β	effort exponent	$\mathcal{U}(1.5, 2)$
θ	preference for income	$\mathcal{U}(0, 1)$
ω	time endowment	1
rate	savings rate, multiplies wage each time step	$\mathcal{N}(.03, .01)$, truncated at 0
Global Variables		
N	number of agents	600
ν	number of social network links	$\mathcal{U}(2, 6)$
churn	agent activation rate	.1
tmax	number of steps	500
move	job change cost, multiplies last wage	1
startup	startup cost, multiplies last wage	2
lendingrate	cost of loan each time step	.03
	compensation rule	equal shares
	initial condition	all singleton firms

Table 1: Agent and global parameters with starting values.

Input & Outputs

The EF model does not require any external input files.

The model produces two files, a `csv` and a network graph, `gml`. Each row in the `csv` file contains an agent's parameters for a given time step. For `tmax`= 500 and $N = 600$ the file will contain 300,000 rows. The agent parameters compose the columns and are described in Table 2. The `gml` graph file contains the network graph describing all the star subgraph firms and singleton firms at the end of the simulation.

Submodels

The EF model is coded in Python 3 with agents implemented as dictionary objects and firms implemented as a network graph. The nine submodels pseudo-coded and described below provide further implementation and functional details.

```
1. instantiate agents

  for i to 1 to N:
       id = i,
       omega = 1.0,
       theta = random value in uniform(0, 1),
       a = random value in uniform(0, .5),
       b = random value in uniform(.75, 1.25),
```

PARAMETER	DESCRIPTION
id	id number
ω	time endowment
θ	preference for income
links	number of neighbors
component	component membership in the social network graph
a	effort multiplier
b	exponential effort multiplier
β	effort exponent
rate	savings rate
U_self	singleton utility
e_self	singleton effort
e_star	current profit maximizing effort
firm	firm affiliation
wage	current wage
savings	current savings
loan	current loan balance
borrow	binary flag indicating agent borrowed current time step
startup	binary flag indicating agent formed startup current time step
move	binary flag indicating agent changed firms current time step
thwart	binary flag indicating agent was thwarted current time step
go	binary flag indicating agent was activated current step

Table 2: Agent parameters captured in the EF model csv output file.

```
      beta = random value in uniform(1.5, 2),
      rate = savingrate if sigma = 0,
              else random value in truncated normal(savingrate, sigma),
      firm = i
  F = network with N unconnected nodes

2. create social network

  degree_list = N random values in uniform(mindegree, maxdegree)
  G = graph with N nodes and degree_list
  find components of G
  for i in agents:
      links = degree for node i
      component = component membership for node i

3. calculate singleton utility and e*

  optimize e for maximum utility for singleton firm
  for i in agents:
      wage = output
      e_star = e_single

4. compute e* and U(e*) for opportunities
```

```
go = 1
```

An agent's utility is calculated via a Cobb-Douglas function with his individual preference set for income and leisure

$$U = \left(\frac{O}{n}\right)^{\theta} (\omega - e^*)^{1-\theta}, \tag{1}$$

where O is total firm output, n the number of persons in the firm, such that $\frac{O}{n}$ is the individual's wage in the current firm configuration. The individual's preference for income is given by θ, therefore preference for leisure is $1 - \theta$. The individual's total time endowment is ω and e^* is the individual's utility-maximizing work effort, thus the individual's leisure is $\omega - e^*$.

Each firm has an owner with unique parameters a, b and β, returns to scale, that characterize the firm's production function

$$O = \frac{aE + bE^{\beta}}{n}, \tag{2}$$

where E is the sum of all the firm members' efforts, e.

To find utility for current position:

```
E = sum off e for all agents in firm
optimize e for maximized utility function
```

To find utility for changing firms:

```
for each neighbor identify firm
find all unique connected firms
for each connected firm:
     E = sum of e for all agents in firm
     optimize e for maximized utility function
choose maximum utility and associated firm
```

Startup utility was already calculated during setup and is stored in the agent dictionary.

5. `start own firm` If starting a new firm provides the maximum utility then the firm affiliation for the agent needs to change its own id, and that node needs to remove its links in the F graph.

```
cost = startup * wage
if savings >= cost:
change firm affiliation and remove firm network links
          savings = savings - cost
          update wage and e* values
```

```
          startup = 1
else if loans < 0:
          loans = costs - savings
          update wage and e* values
          startup, borrow = 1
else thwart = 1
```

6. `join new firm` If joining a new firm provides the maximum utility then the firm affiliation for the agent needs to change, as well as the linkages in the F graph.

```
cost = move * wage
if savings >= cost:
change firm affiliation and firm network links
          savings = savings - cost
          update wage and e* values
          move = 1
else if loans < 0:
          loans = costs - savings
          update wage and e* values
          move, borrow = 1
else thwart = 1
```

   If the agent is a firm owner, then the change ownership routine will run before changing affiliation.

7. `pass ownership`

```
new_owner = random neighbor
firm affiliation for new owner changes to id
firm affiliation for remaining employees changes to new owner
remaining employees link to new owner
```

8. `distribute output`

```
E = sum of the effort for all members of a star subgraph
O_total = (a * E + b * E **beta)
share = O_total / n
for all agents in star subgraph:
     wage = share
     savings = savings + share * rate
```

9. `pay loans`

```
for all agents with loans:
   loan = loan * (1 + lendingrate) - savings
```

```
   if loan < 0:
savings = abs(loan)
loan = 0
   else:
savings = 0
loan = loan
```

References

Axtell, Robert L (2015). "Endogenous Dynamics of Multi-Agent Firms". In: *Available at SSRN*.