An Agent-based Approach to ARGESIM Benchmark C16 'Restaurant Business Dynamics' based on NetLogo

Julian Ruths, Günter Schneckenreither*

Dept. Mathematical Modelling and Simualtion, Vienna Univ. of Technology,
Wiedner Haupstraße 8-10, 1040 Vienna, Austria; *guenter.schneckenreither@tuwien.ac.at*

Simulation Notes Europe SNE 24(3-4), 2014, 127 - 130
DOI: 10.11128/sne.24.bn16.10254
Received: January 20, 2014; Revised: October 11, 2014;
Accepted: October 20, 2014;

Abstract. In this comparison the business dynamic of restaurants is modeled. Restaurants open in an area inhabited by people, who visit those restaurants from time to time. Restaurants may close or a new restaurant may open, depending on their business performances. For this comparison, we chose an agent-based approach, using the freely available agent-based programming language and integrated modeling environment NetLogo. We will give a brief overview of the model assumptions, the reasons that led us to the decision of an agent-based approach, as well as a short introduction of NetLogo and its advantages regarding the implementation of the model. We will discuss specific aspects of the implementation itself and the results of several tasks that had to be performed.

Introduction

30 restaurants are initially evenly distributed in a rectangular area. There are 3000 people living in this area, each belonging to one of 5 cities and placed around those cities within a given radius. Every day a changing number of people (depending on their state of 'hunger') visit a restaurant within a given radius and pay the restaurant a defined amount of money for their meals. At the end of each week, restaurants calculate their profit. Depending on the profit the location then may close or a new restaurant might open somewhere else in the area.

1 Model

Although there is no precise definition of what an 'agent' is, general agreements can be found. One of these agreements is stated in [1]:

- Indentifiable: every agent is a discrete individual with behavior rules and attributes that can vary with every agent.
- Situated: agents live in an environment and are able to interact with it as well as other agents.
- Autonomous and self-directed: agents function independently.
- Goal-oriented: agents may have goals they want to achieve.
- Adaptive: agents may be able to learn and adapt their behaviour.

An agent-based model therefore conists of agents living in an environment, interacting with each other or the environment, based on behaviuor rules. Having inspected the model descriptions, an agent-based approach turned out to be applicable to this problem.

1.1 Simulator

We developed the model using NetLogo. NetLogo is a free and open source software and a dialect of the programming language Logo [2].

The NetLogo world is made up of 4 basic elements [3]: *Turtles*, which are agents and inhabit the world; Patches, which make up the environment of the world; *Links*, which can be used to signal relations between turtles; *Observer*, which doesn't have a location and is watching everything.

Turtles are distinguishable by a unique ID and NetLogo allows different types of turtles, which is implemented by defining different turtle-breeds. Each turtle-breed has its own properties and is defined by the user.

NetLogo has implemented a functional-language style, which makes most usages of loops omittable. One can simply address members of a certain breed that fulfil given criteria and go through each one of them without having to know how many there actually are. Since the number of restaurants throughout the course of the simulation can vary extensively, this allows for a simple access without having to first count all the existing restaurants and inserting a loop to go through them all. Very intuitive and readable syntax as well as an easy-to-handle interface made NetLogo a perfect candidate to work with on this comparison.

1.2 Model implementation with NetLogo

Space. The rectangular area is easily defined using the environment settings provided by the NetLogo interface. At startup the cities are placed according to the given coordinates. Every city then creates the number of people belonging to the city. They are placed around the city with the angle being uniformly distributed and the radius being triangularly distributed towards the city center and a maximum of maxR (see cities below). This represents the fact that the people tend to live closer to the citycenters.

Time. Every tick (time-step) represents one day. After seven ticks, a week has passed and restaurants need to calculate their profit.

Persons. Each turtle of this group belongs to the turtle-breed person. Besides their predefined attributes, every person has a *dining intervall* and a list of *destinations.* The dining interval ranges from 0 to 8 and indicates how many days have to go by until the next meal. At the end of every day, this value is reduced by 1. If the value at the start of the day is zero, the person visits a restaurant within dining range and a new dining interval is randomly calculated (one day is added to the new value, since it will be reduced by 1 again at the end of the day). The destination list contains the IDs of all restaurants within dining range of the person. In an earlier version of the model, this attribute was not included as will be explained further down.

Restaurants. Each turtle of this group belongs to the turtle-breed restaurant. Besides their predefined attributes, every restaurant has an *income*, a *profit* and an attribute titled *new*.

Every time a person visits the restaurant, the income is increased by the value of the dining cost. The profit is calculated at the end of every week using a given taxrate and running costs. The value *new* indicates whether the restaurant exists since the beginning of the simulation and was only introduced for the completion of task c.

Cities. Each turtle of this group belongs to the turtle-breed city. Besides their predefined attributes, every city has a population percentage and a radius maxR. At startup every city will be placed at its given location and create a number of persons according to the population percentage. These people will then be distributed around the city area as was described in the Introduction.

Cells. Each turtle of this group belongs to the turtle-breed cell. Besides their predefined attributes, every cell has a population density and a location coefficient. The population density represents the number of people living on that cell and since people don't move, it is calculated once at startup via the following command:

```
ask cells [
set peopledensity count persons with [ abs
   (xcor - cellx) <= 10 and abs (ycor - celly)
    <= 10 ] ]
```

The location coefficient is calculated with the given formula at startup and and again at every time-step where the event of opening a restaurant occurs.

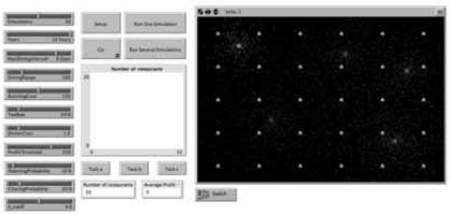

Figure 1: The user-interface of the model allows easy changes to several parameters.

User-Interface. The interface provided by NetLogo, allows easy manipulation of parameters using global variables. In the finished model-interface the user can vary all parameters such as MaxDiningIntervall, runningCost, taxrate etc. A seperate window plots the number of restaurants over time and another one shows the representation of the two-dimensional world with persons, restaurants and cities all visible and accessible via mouse click.

1.3 Specific Implementation

The intuitive syntax of NetLogo allows for quick and easy programming. What follows is an example of a piece of code that we initially used to simulate the process of hungry people choosing a random restaurant within range and the restaurant then increasing its income:

```
ask persons with [diningintervall = 0] [
   ask one-of restaurants in-radius DiningRange
   [ set income income + 1 ] ]
```

Obviously the code doesn't differ much from the in-text formulation, which is one of the advantages of NetLogo. One has to be careful though, as to not overlook more effective ways of programming. The first simulations we ran, used the above stated way and turned out to be much too time consuming. The reason for this is easily found: With around 400 to 500 people being hungry every day, the command 'in-radius', which is a quite complex command in itself, needs to be executed 400 to 500 times as well. It became obvious then, that, since people never change their location after the startup, the restaurants within dining range only change if there is a closing, or an opening nearby. As a solution we introduced the previously mentioned destination list. This simple change resulted in a massive lowering of computation time.

1.4 Flow of Information

A question that we had to ask ourselves in the process of modelling, was 'Who is the acting agent?' At first glance, the answer seems obvious. People look for restaurants nearby; people visit these restaurants randomly and pay money for their consumed meals. Restaurants may close down, but that is only a result of the behaviour of the person.

The conclusion, that the person is the acting agent is therefore quickly drawn. In this particular model however it is more effective to look at the restaurants as the acting agents and reversing the flow of information. Until now, each person was looking for restaurants nearby. Even with the implemented destination list and the update of this list only when a change in restaurants occurs, this results in 3000 executions of the in-radius command. Reversing the information flow, results in restaurants telling people once that they are located within dining range and another time - upon closing - that they are not existing anymore.

```
let nummer who
ask persons in-radius DiningRange [
    set destination remove nummer destination ]
die
```

The simulation of 4 weeks using the first implementation, took about 90 seconds. With all the changes that were proposed in this subsection as well as subsection 1.3, implemented in the final version, allows a cycle of 10 simulations - each set to run for 5 years - to be completed in about 60 seconds.

2 Simulation Tasks

2.1 Task a - Time domain analysis

Because of the high amount of randomness, we had to perform the simulations multiple times and averaged the data to get meaningful results. As demanded, 50 simulations were run and the limit value of the number of restaurants after the 5th year was calculated. This was done by inserting an outer loop and storing the number of restaurants after 5 years in a list. We then averaged the data in this list, using standard methods of NetLogo.

Mean	5.66
Variance	1.98
Deviation	1.41

Table 1. Meanvalue, variance and standard deviation of the number of restaurants after 5 years.

The deviation of the overall number of restaurants is quite high in respect to the mean value. Paying attention to the number of restaurants during a simulation run reveals a great number of openings and closings, due to the fact that restaurants have no memory of previous weeks. This results in abrupt closings of restaurants that just had a bad week.

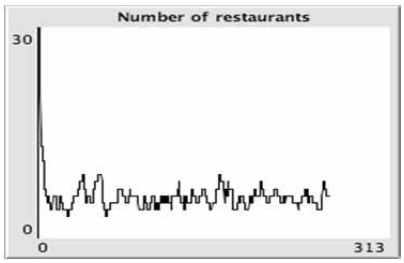

Figure 2: Number of restaurants plotted over the course of 260 weeks.

Due to this fact, the number of restaurants after 5 years can vary a great deal, since a steady state of the system can hardly be achieved.

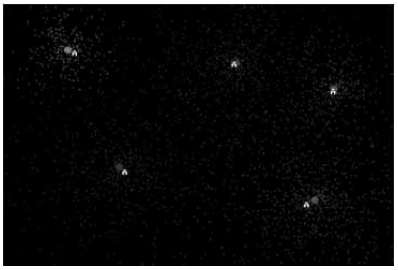

Figure 3: In-simulation view of the deployment of restaurants around the cities.

2.2 Task b – Maximisation of tax income

The relationship between taxrate and taxincome is quite obvious. The lower the taxrate, the more restaurants are able to survive. Therefore a higher number of restaurants pay taxes, but the amount they have to pay is low. A high taxrate results in fewer restaurants paying a very high amount of taxes.

The question of maximising taxincome via variation of the taxrate is strongly dependend on the looked upon time period. Wanting to maximise taxincome over to course of 5 years results in completely different taxrates, than maximising over the course of 6 months, or even 2 weeks. We assumed a longterm wish of steady tax income and ran a cycle consisting of 10 simulations – each lasting 5 years – and averaged the tax income. We then increased the tax rate by 1% starting with 0% and ending with 100%.

The results show a maximum taxincome of 193658,688 currency units at a taxrate of 32% as seen in Figure 4. The high fluctuation in the area of 25% to 50% may be smoothed out when increasing the iterations per taxrate.

2.3 Task c – Maximisation of new restaurants' revenue

The 'best' location for opening a new restaurant is only dependent on the coefficient k. This represents the importance, that the location of other restaurants play in the evaluation of a possible new location. $k = 0$ resulting in restaurants opening where the peopledensity is highest and, with gradually increasing values of k, paying more attention to the restaurants close by. Several variations of the parameter k showed that a maximal revenue of 9151 curreny units, can be found at $k = 1$.

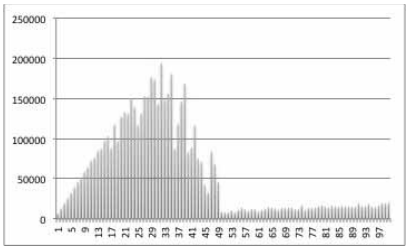

Figure 4: The average tax income after 5 years with varying tax rates.

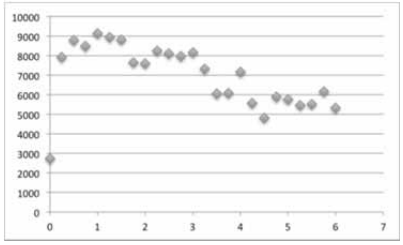

Figure 5: Average revenue of restaurants opened after startup.

3 Conclusion

Agent-based modelling with NetLogo is a very intuitive and straight forward method. But one of the problems with agent-based modelling & simulation is that it often involves a very high number of computations due to the fact that a system is made up of thousands of agents, all acting and interacting with each other. This can result in very time-consuming simulation runs. Therefore considerate thought should go into asking how these interactions should be executed. A simple inversion of the information flow proved to be very effective in this particular case, respective in an efficient simulation.

Model sources

NetLogo is an Opene Source simulator, available from web. For this C16 benchmark approach, Netlogo model files, parametrization files, and a short file description (and a link for NetLogo download) can be downloaded (zip format) by EUROSIM societies' members from SNE website, or are availably from the corresponding author.

References

[1] Macal, CM, North MJ. Tutorial on agent-based modelling and simulation. *Journal of Simulation*. 2010; 4(3), 151-162.

[2] Wilensky, U. *NetLogo*. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling Northwestern University. Evanston, IL. 1999.

[3] Izquierdo, LR. *NetLogo 4.0–Quick Guide*. 2007