Comparison of three implementations of Schelling's spatial segregation model

Éric Daudé, Patrice Langlois

Chapter 13

Comparison of three implementations of Schelling's spatial segregation model

Éric Daudé and Patrice Langlois

13.1. Emergence of a spatial structure: Schelling's spatial segregation model

The price of economy "of the bank of Sweden to the memory of Alfred Nobel" 2005 awarded to Thomas Schelling is, with the Nobel Prize of chemistry awarded in 1977 to Ilya Prigogine, for *his* work on the dissipative structures, one of the unusual prices which challenge directly the community of the geographic modelers. Because it "contributed to a significant degree to reduce the gap between the social economy and other sciences and of the behaviour", Thomas Schelling is indeed largely quoted in geography, mainly by the searchers involved in the geographic complex systems. If he owes mainly this price to his work in game theory, it is his contribution to a fundamental question - How individual behaviours may produce phenomena which pure theory cannot anticipate? – which is related to our purpose.

At the beginning of the seventies, Schelling [SCH 71], [SCH 69] attempted to show how cities can be structured in Community blocks, where white and black face themselves without never mixing. The investigations which he carried out show however that people do not want to be a majority in a district, they are thus not segregationnists, but the mere will not to constitute a too large minority produces a segregation. In other words, there can be segregation without individual will of segregation. Schelling is thus able to highlight simple and at the same time

---

Chapter written by Eric DAUDE and Patrice LANGLOIS

comprehensible and very profound ideas, he demonstrates it in "*Micromotives and macrobehaviour*" [SCH 78].

In this book published in 1978, Schelling proposes a model to explore the paradox of segregation. He places for that on a chess-board with N squares N_1 grey pawns, N_2 black pawns and N_3 empty pawns, where: $N_1 = N_2$ and $N_1 + N_2 + N_3 = N$.

The strategy used to explore the dynamics of the paradox mentioned above is relatively simple. It is based on the degree of tolerance of an individual to the individuals different from him, present in his vicinity. Each pawn is concerned with its immediate vicinity, defined by the number of occupants of the 8 contiguous squares. A pawn will move from its square only if the number of "foreign" pawns in its vicinity exceeds the fixed tolerance level, identical for all. In this case, it will take a new position randomly on a vacant square. For example if an individual accepts for neighbours up to two thirds of different neighbours (diff) and thus one third of identical neighbours (similar), it will stay in its place if its vicinity contains at least 1 neighbour identical to it if it has only 1 or 2 neighbours; at least 2 similar neighbours if there is between 3 and 5 neighbours and a minimum of 3 identical neighbours when the individual is surrounded by 6 to 8 neighbours. Let us notice that the occupancy is calculated here compared to the number of individuals present in the vicinity, and not compared to the number of cells of the vicinity, which is always 8 in our case.

neighbours	diff 0	diff 1	diff 2	diff 3	diff 4	diff 5	diff 6	diff 7	diff 8
similar 0	1	1	1	1	1	1	**0**	**0**	**0**
similar 1	1	1	1	1	1	1	**0**	**0**	
similar 2	1	1	1	1	1	1	**0**		
similar 3	1	1	1	1	1	1			
similar 4	1	1	1	1	1				
similar 5	1	1	1	1					
similar 6	1	1	1						
similar 7	1	1							
Similar 8	1								

Number of neighbors

(similar + diff) = number of neighbors.
Leaves if equal to 0, stay if equal to 1.

Table 13.1. *Configurations of the vicinity of a centre square and action according to the tolerance level (fixed here at 66%) with a rule taking into account the density of population.*

Various rules are tested by Thomas Schelling, which are alternatives of the previous model: The groups may have different tolerance levels; equality or not of the number of individuals per group etc. One of the alternatives of the model which we will explore here takes into account a dimension left aside by Schelling, the global density of the population. One can indeed consider that the individual decisions are at the same time imposed by local requirements, namely the social

structure of the vicinity, but depend also partly on more global constraints like the density of population. One thus widens the concept of social environment to the concept of spatial environment: material space, the free cells which surround me, plays a considerable role in my appreciation of tolerance. An individual thus defines his requirements not only by the social composition of his environment but also in reference to the density in this vicinity: the tolerance level is thus defined as the maximum proportion of foreigners whom I accept in my vicinity made up of eight houses (Table 13.1). Once the used strategy is set down, it remains to implement and explore the total behaviour of the model according to the various concerned parameters.

13.2. Translation of the assumptions in a CA and MAS context

We are first setting up a metaphoric representation of reality by means of a Cellular Automata (CA) and a Multi-Agent System (MAS). The square domain of the automat represents a city (the chess-board for Schelling) where each cell represents a dwelling. An inhabitant is represented by an agent (a pawn for Schelling). A cell accommodates at the most one agent (at the most one pawn per box of the chess-board). The data-processing implementation of this model needs to clarify assumptions and parameters which are often ignored in the presentations:

- the size of the domain plays an important role in the combinative of the possible reorganizations of the individuals. One will discuss its effects in the following section;

- Schelling uses a chess-board which is a finished and limited domain. To avoid important edge effects, a total topological structure of toroidal type will be used, which gives a finished but unbounded domain (N cells). The structure of the vicinity is then homogeneous for all cells, there is no edge effect anymore;

- taking into account a parameter of total density of population in the domain, which plays an important role in the dynamics of the model. In this case it is important to take into account the number of blank cells in the evaluation of the proportion of foreigners in the vicinity. In other words, one will calculate the densities of neighbouring population compared to the number of houses to the vicinity, and not to the number of inhabitants of this vicinity;

- One will be able also to vary the number of social groups which will reveal a more complex dynamics of regrouping;

- Within the framework of a MAS, one distinguishes the cells (dwellings) and the agents (inhabitants). The cells may be inhabited or not by an agent, and only one. The population of agents does not vary, any agent dissatisfied of its environment moves towards an unoccupied cell. One will thus be brought to manage a list of free dwellings, a kind of newspaper of "property announcements" consultable by the agents before any displacement. This list will be updated after each move;

- Contrary to MAS, the mechanism of displacement does not exist in a strict cellular automaton. So a stratagem will be used to replace it. This stratagem will lead us to modify the model: the moves in and out of the dwellings are then independent and occur respectively towards and from outside. Thus, in this case, the population of each group is not fixed anymore. It cannot exceed the fixed threshold, but can fluctuate under this threshold.

13.3. Software design of the model

It is a requisite to delimit the scope of a model before launching out into its phase of data-processing implementation: a data-processing program starts always by a preparatory through, which can be based on a methodology of assistance to the design, like UML[1]. It is then a question of describing the various handled objects, the state variables and the action during a step of time. We present in the following sections the activity chart and the incremental diagram of this model.

13.3.1. Object-oriented UML processing through activity diagrams

The activity diagram during a time step is shown by figure 13.1. This process will be repeated until a checking condition to produce the complete simulation. It would also be necessary to define the activity during the initialisation process which includes the creation of the initial pattern: randomised location of N_1 inhabitants of type A, of N_2 inhabitants of type B, initialisation of the list of N_3 empty places.

In this scheme, two types of objects are defined: houses and inhabitants. Two houses (*i* and *j*) are represented and one inhabitant. The houses are represented by objects of "cell" type of the cellular automaton and the inhabitants by objects of "agent" type of the multi-agents system. A cell possesses an internal state variable, *dwelling*, which takes the values of "occupied" or "unoccupied". An agent possesses two state variables: its *type* (A or B) which indicates the social group of the inhabitant and its *satisfaction* which takes the values of "satisfied" or "not-satisfied" due to the type of population in its vicinity.

The following diagram thus represents the activity of the objects during a time step (axis going down).

One starts by evaluating for each inhabited cell the level of satisfaction of its inhabitant. An inhabitant is satisfied if the number of neighbours of the type different from him (which one calls thereafter a number of foreigners) is lower or equal to its threshold *S* of tolerance. A dissatisfied inhabitant then tries to find a free

[1] Unified Modeling Language, see Chapter 13.

house. If he finds one, he moves from his house *j* towards a new unoccupied house *i*. To accelerate the search for houses, the system memorizes a list of the unoccupied houses which is updated after each move.

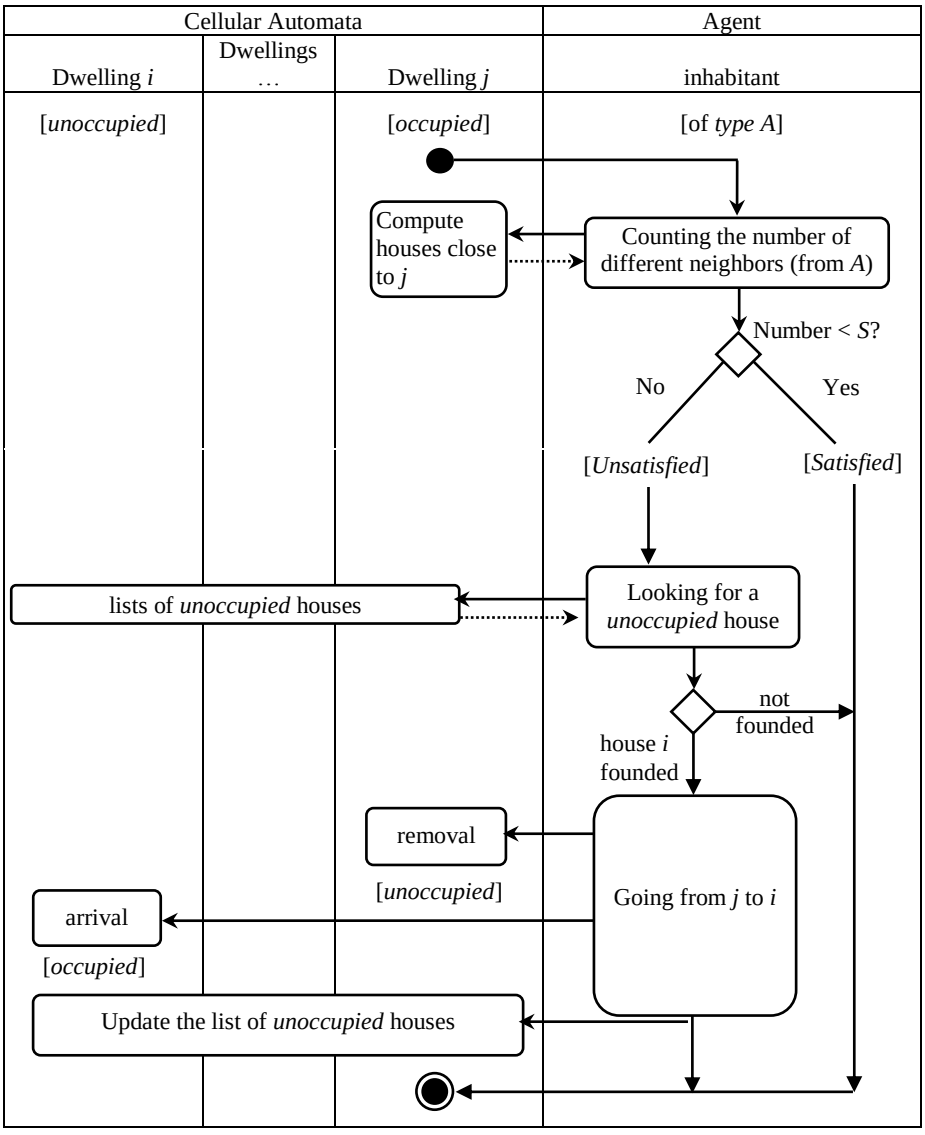


Figure 13.1. *Activity UML Diagrams corresponding to a time step in simulation*.

13.3.1.1. Algorithmic processing

The UML diagram is translated rather simply into an algorithm which resumes the two phases of evaluation and action. We programmed this algorithm under Excel in VBA language.

The houses are arranged on a cellular grid where each cell is a square (and also a cell in the Excel language). The index of house *j* of the UML diagram is then replaced by a double index (*i, j*) which indicates the numbers of line and column of the square cell. The number of line *i* varies from 1 to N_L and the number of column *j* varies from 1 to N_C, with $N = N_C * N_L$.

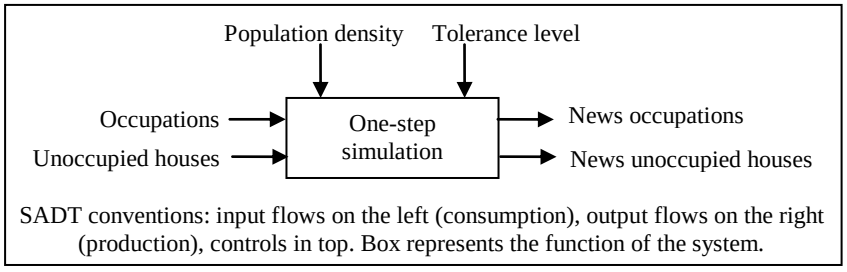


Figure 13.2. *Systemic diagram of the model*

The algorithm takes two tables in entry: *Occupation(i,j)* and *ListOfUnoccupiedHouses(k)* which respectively indicate the type of occupation of the cell (*i,j*) and the position of the k^{th} free house. These tables are filled during the initialisation phase. The table *ListOfUnoccupiedHouses* is not essential, it only allows a reduction of the processing time when the number of cells is important compared to the number of free boxes, by avoiding a long research for each iteration. The algorithm calculates the table *Satisfaction(i, j)* of each house if it is occupied, then proceeds to the moves of dissatisfied individuals by updating the list of the empty houses. It provides at the end a new pattern of inhabitants in the houses. After all two control parameters are important in the global behaviour of the model: the density of population and the level of tolerance, they remain fixed during a simulation. One can simplify these specifications by a box (Figure 13.2) and describe as follow the algorithm which will be resumed for the implementation under Excel:

The evaluation uses the function *NbrDifferent(i, j)* which calculates the number of individuals of the vicinity of cell *j* from which the social group is different (*i, j*). The subroutine “action” uses the *ListOfUnoccupiedHouses* list which contains the positions of the free cells. This list requires two procedures: *ChooseHouse* to choose a free house in the list and to update the list after a move. We will not detail here

these subroutines referred in the principal algorithm, because they are detailed in the implementation under Excel and StarLogo.

```
Algorithm OneStep
Begin
    for all cell i, j do
        if Occupation(i, j) is Occupied then
            if NbrDifferent(i, j) > S then              // the dweller is unsatisfied
                if ListOfUnoccupiedHouses is not empty then   //action
                    Choose a house m in ListOfUnoccupiedHouses
                    Compute position (ii, jj) of m
                    Go from (i, j) to (ii, jj)
                    Update ListOfUnoccupiedHouses
                end If
            end If
        end If
    end for
End Algorithm
```

In the case of a square grid with a vicinity of Moore (8 neighbours for a radius of 1), the operator of vicinity contains a table V having $nV = 8$ cells, each cell containing a vector of shift of the form $(\Delta x, \Delta y)$. the table V contains the following data:

$V = ((0, -1), (1, 0), (0, 1), (-1, 0), (-1, -1), (1, -1), (1, 1), (-1, 1))$

Let us notice that in a square grid of N_L lines and N_C columns, the cells can be numbered of 0 in n-1, the cell of number m has as co-ordinates j (number of column), and i (number of line), which are calculated in the following way:

$$j = (m \bmod N_C) + 1 \qquad [13.1]$$

$$i = (m \operatorname{div} N_C) + 1 \qquad [13.2]$$

where "mod" and "div" represent the remainder and the whole quotient of the division of the integer m by the integer N_C. Reciprocally, if one knows i and j, the number m of the corresponding cell is:

$$m = (i - 1).N_C + j - 1 \qquad [13.3]$$

This method is used for example to memorize the numbers of free cells in *ListOfUnoccupiedHouses*, then to return easily to the co-ordinates i and j associated with these numbers, for example to calculate the location of the number m cell.

13.4. Computer implementations

To allow a comparison between various programming methods, we have implemented this model into three very different environments: StarLogo, Excel and SpaCelle. We discuss the differences in terms of objects implied by these three environments.

13.4.1. Implementation into StarLogo

StarLogo[2] is developed by the Media Laboratory (Cambridge), the MIT (Massachusetts) with the support of the National Science Foundation and of the LEGO Company. The user language of StarLogo is called Logo. It is translated into Java code before being carried out. This platform is adapted to experiment theories concerning the emergent phenomena with a focus on the interaction between a great numbers of autonomous agents, with simple behaviours (reactive agents).

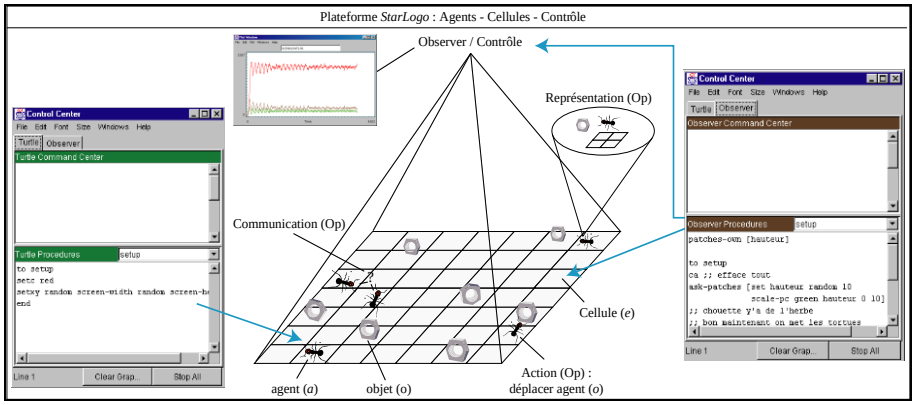


Figure 13.3. *General outline of the StarLogo platform.*

The platform is divided into two units, the **Control Centre** which is centred on programming and the **StarLogo unit** which is the window of simulation and visualization. The procedures which are written in the Control Centre are thus simulated in the StarLogo window. The names of the fundamentals are common for both units: the cells called **Patches** and the agents called **Turtles**. Finally the **Observer** realizes, in the ideal, what does not concern the method of agent such as the synchronisation procedure, the operations of aggregation, the realization and visualization of graphs etc. (in the ideal because there may exist a small confusion between Observer and Patches, those being managed in the same tab, whereas the methods specific to the agents are written in the Turtle tab). In short all what

[2] http://education.mit.edu/starlogo/

concerns the environment of simulation (graphs, counting, sequence of methods) and of the cellular automaton will be thus implemented in the Observer tab of the Control Centre, all what concerns the behaviour of the agents will be implemented in the Turtle tab of the Control Centre (Figure 13.3).

A relatively significant number of primitives (words of the language) are associated with *Turtles*, *Patches* and *Observer* and allow writing procedures which, by combination, define the universe of the phenomenon to be simulated. We are presenting here the fundamental procedures to operate the Schelling's model, the initialization [setup] procedure which defines the groups [breeds], their number [NbGroup], the calculation of the density [Density(%)], the tolerance level [ToleranceMax] as well as the initial location of the agents can be consulted by downloading the application[3].

An essential question which must be asked relates to the choice between a synchronous or asynchronous mode for the course of the program. In the synchronous mode, the agents dissatisfied by their vicinity move at the same time, one thus obtains a wave of migration during one iteration. In fact the sequential nature of the personal computers compels to simulate this mode with the use of temporary variables memorizing the situation at the beginning of the iteration. The agents move one after one but according to their satisfaction calculated from this common "image": they do not take into account the possible effects of the moves of their predecessors. On the contrary, in asynchronous mode, the agents formally move one after one and are able to modify local or global variables, the non use of temporary variables makes it thus possible to take into account the effects of these microchanges. In short, in synchronous mode on an iteration, a departure or an arrival in the vicinity of an inhabitant does not affect his choices, whereas in asynchronous mode, this change is taken into account by the inhabitant and can modify his satisfaction. From a technical point of view, the course of an iteration cannot be carried out completely in synchronous mode. The evaluation of the situation can be done only in this mode:

- Creation of the list of the vacant, not inhabited dwellings [*CreateListOfUnoccupiedHouses* procedure, *ListOfUnoccupiedHouses* list];
- Each agent observes its vicinity, the operation consists in counting the number of foreigners in a vicinity of order 1 [*ObserveNeighbours* procedure];
- Each agent compares the tolerance level to his personal observations in order to decide if yes or no he must move [*ToLeave?* procedure]. If the answer is yes, his name is added to the list of the agents who must leave [*ListOfUnsatisfied*];

[3] http://www.univ-rouen.fr/MTG/EricDaude.htm

The moves cannot be carried out in synchronous mode, because several dissatisfied inhabitants could then move to the same free cell. Any agent in the *ListOfUnsatisfied* list randomly selects a free dwelling in the *ListOfUnoccupiedHouses* list and goes there. He leaves the dissatisfied list, removes his new address of the list of the free dwellings and adds in this list his old address, which is now vacant. When the *ListOfUnsatisfied* list is empty, the iteration is finished, the procedure is repeated.

In asynchronous mode, the procedure is slightly different:

- creation of the list of the vacant, not inhabited dwellings [*CreateListOfUnoccupiedHouses* procedure, *ListOfUnoccupiedHouses* list];
- creation of the list of all the agents present in the universe [*ListAgents* list];
- one repeats the following procedures until *ListeAgents* is empty: random selection of an agent in the list, observation of its vicinity [*ObserveNeighbours* procedure], and according to his satisfaction he remains or leaves [*ToLeave?* procedure].

The difference between these two alternatives is thus that in the first one, during an iteration, only the unsatisfied individuals are moving, a criterion defined by an "image of reality" at a given moment, the same for all. In the second one, in asynchronous mode, one treats all the individuals one after the other, at distinct moments, and the evaluation of the environment is done on an "image of reality" at this moment, valid for the selected individual. It is the latter mode which was selected, the various procedures are detailed in the following paragraphs[4].

The *ListOfUnoccupiedHouses* procedure allows the creation of the complete list of the cells which are not inhabited:

```
to CreateListOfUnoccupiedHouses
 set ListOfUnoccupiedHouses [ ] ; list is free at departure
 ask-patches [
  if count-turtles-here = 0 [      ; if no inhabitant
   let [:k (((xcor + screen-half-height) * screen-width ) +
    (ycor + screen-half-width) )] ;compute the number of the
    cell
   set ListOfUnoccupiedHouses lput :k ListOfUnoccupiedHouses]]
 ]                                  ; and one insert in the list
end
```

The *ObserveNeighbours* procedure allows each agent to enter the number of foreigners in the vicinity, one uses here the operator of vicinity previously presented.

[4] The terms in **fat** represent the StarLogo primitives, in "normal" the author language's.

```
To ObserveNeighbours
 set diff 0
 let [:a breed-at 0 1]     ; one start to observe in north (N)
  if (:a != breed) and     ; if inhabitant is different from me
     (:a != 0)
       [set diff diff + 1]] ; then increment the state variable
 let [:a breed-at 1 1]; same operation in north-east (N-E)
  if (:a != breed) and (:a != 0) [set diff diff + 1] ]
 let [:a breed-at 1 0] ; same operation in east (E)
  if (:a != breed) and (:a != 0) [set diff diff + 1] ]
 let [:a breed-at 1 -1] ; same operation in S-E
  if (:a != breed) and (:a != 0) [set diff diff + 1] ]
 let [:a breed-at 0 -1] ; same operation in S
  if (:a != breed) and (:a != 0) [set diff diff + 1] ]
 let [:a breed-at -1 -1]; same operation in S-O
  if (:a != breed) and (:a != 0) [set diff diff + 1] ]
 let [:a breed-at -1 0] ; same operation in O
  if (:a != breed) and (:a != 0) [set diff diff + 1] ]
 let [:a breed-at -1 1] ; same operation in N-O
  if (:a != breed) and (:a != 0) [set diff diff + 1] ]
end
```

The *ToLeave?* procedure determines if the agent must leave or not the cell according to the fixed tolerance level.

```
to ToLeave?
  if diff > ((Tolerance / 100) * 8) [ChooseNewHouse]
end
```

The *ChooseNewHouse* procedure is called by any individual answering "true" to the previous test (*ToLeave?* procedure). The agent selects a random unoccupied cell in the list of the free dwellings, removes it from the list and adds to it the number of the cell which it leaves vacant. It takes position then on the cell previously selected.

```
to ChooseNewHouse
  let [:a pick ListOfUnoccupiedHouses
       :b ( ((xcor + screen-half-height) * screen-width ) +
            (ycor + screen-half-width) )]
  set ListOfUnoccupiedHouses remove-element :a
      ListOfUnoccupiedHouses
  set ListOfUnoccupiedHouses insert (1+(random length
      ListOfUnoccupiedHouses)) ListOfUnoccupiedHouses :b
  setx  ((:a div screen-width) - screen-half-width)
  sety  ((:a mod screen-width) - screen-half-height)
end
```

Finally *Go* procedure represents the sequence of all the procedures in a loop, here asynchronous.

```
to Go
 set iteration iteration + 1    ; global variable
 CreateListOfUnoccupiedHouses
 set ListAgents List-of-turtles     ; list of all agents
 Loop [
   ifelse empty? ListAgents
     [stop]
     [let [:you pick ListAgents]    ; local variable
     set ListAgents remove-element :you ListAgents
     ask-turtle :you [     ; selectioned agent
        ObserveNeighbours]
        ToLeave?]
     ]
  ]
end
```

The link between the algorithmic part of the model, the *Control Center*, and the window part of simulation, the *Starlogo unit*, is done via buttons on the graphic interface (Figure 13.4). The **Setup** button allows initialising the space of simulation, to create the agents according to the desired density and to choose the number of groups of agents. The buttons **asynchronous** and **synchronous** make an addressing loop for the iteration procedure corresponding to the chosen mode, in fact to the previous iterate procedure for the asynchronous button. It is finally possible to vary the tolerance level during the simulation. The example of Figure 13.5 illustrates the dynamics of the system composed of three groups.

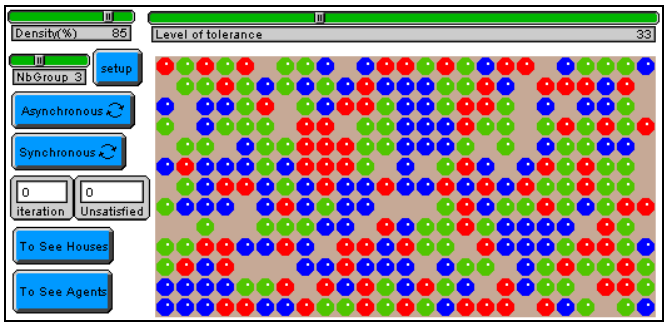


Figure 13.4. *Window for StarLogo simulation of the Schelling's model*

If StarLogo is relatively well adapted for the modelling of this type of model based on agents, it is also possible to implement the Schelling's model in the shape of a cellular automaton, as it will be proposed thereafter. Two limits must be mentioned in connection with this platform.

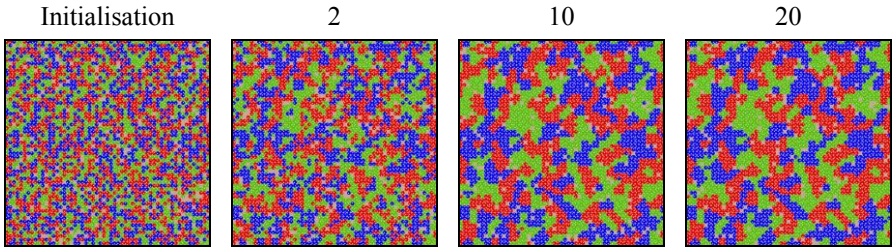


Figure 13.5. *Simulation with a density of 95%, tolerance level at 66 and 3 groups*

A first limit results from the fields of application which are limited by the available stock of primitives: some actions or behaviours have no proper primitives and oblige the model builder to compose with the existing primitives. For example, there is curiously no primitive allowing an agent to observe a status variable in its vicinity[5], which is on the other hand the case for the cellular automaton (primitives **nsum** and **nsum4**). The fact of adding the management of two lists (*UnoccupiedHouses* and *Agents*) increases considerably the computing times, as it will be stressed with the comparative assessment of the three platforms.

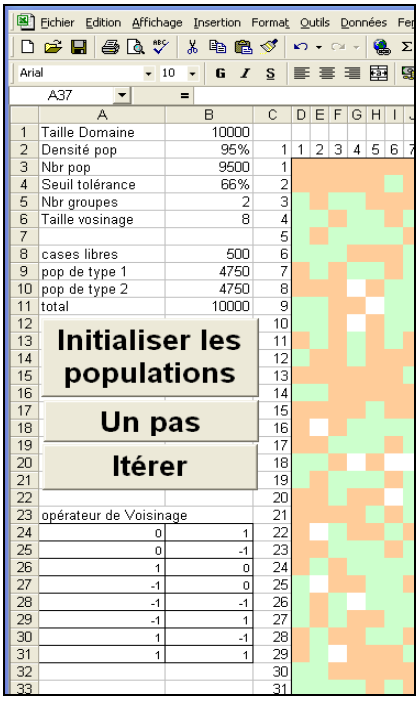
The second limit of StarLogo relates to the transfer of data from and towards the outside. It is indeed not possible to import data resulting from spreadsheets for example and the importation of images containing a too important panel of colours produces hardly satisfactory results. In spite of these remarks, this platform remains faithful to its initial teaching objectives, combining simplicity and exploration of emergent phenomena. The model builder anxious to go beyond these constraints will then have the possibility to turn either towards more generic platforms, like RePast, Swarm (programming Objective C) or AgentSheet (Java programming), or towards platforms adapted to specific objectives, but more flexible as regards programming, like Cormas (programming in Small Talk), or finally towards the direct creation of his own program in one of the numerous generalist programming languages (Pascal Objet, C^{++}).

13.4.2. Implementation under Excel (VBA language)

The advantage to present an implementation under **Excel** is due to the fact that this tool is known by a large majority of the students and searchers in social sciences, since the majority of them do not suspect that one can do an effective simulation into Excel. Indeed, if this software is especially used to organize data and calculations by formulas, it has also a true programming language, VBA (Visual

[5] At least till the 2.1 version of StarLogo.

BASIC for Application) which allows the programming of models of considerable size and complexity (here we use a model of 10000 cells).



	values	Variables
Domain Size	10000	SizeD
Density of population	95%	Density
Number of cells	9500	*K*
Tolerance threshol	66%	S
Number of groups	2	*G*
Vicinity size	8	*NV*
Free cells	500	*Nb Free*

Figure 13.6. *Overlook of the implementation on the model under Excel*

The realization of a simulation model into Excel may include some parts without programming, like the pictorial display of the cell values in a conditional shape, the seizure, the importation or random generation of the initial configuration, the realization of curves or of various table for the observation of the model, etc. Before programming the model, one can define names for the various groups of cells as Domain which represents the cells of the cellular automaton, and other names for the variables defined in the following table, which will facilitate the initialization of the internal variables of the program.

Simulation can be controlled from some buttons which one lays out on the sheet containing the cells (by using the bar of tools “toolbox Control”). By a double-click on the button, one then associates to it a subroutine, written in the worksheet containing the cells, which contains a call to the *OneStep* procedure. The managers of the event “click-mouse” associated with these buttons may be written in the following way:

```
' button management « initialization » :
Private Sub ButtonInit_Click()
  Call InitDomain      ' This subroutine is not described here
End Sub

' button management « One Step » :
Private Sub ButtonOneStep_Click()
  Dim nUnsatisfied As Integer
  Dim nIter As Integer
  nIter = Range("nIter")'one copy cell "nIter" in nIter
  Call OneStep(nUnsatisfied, nIter) ' call subroutine "OneStep"
End Sub

' button management « Iterate » :
Private Sub ButtonIterate_Click()
  Dim nUnsatisfied As Integer
  Dim nIter As Integer
  Dim MaxIter As Integer
  MaxIter = Range("NbMaxIter")
  nIter = Range("nIter")
  Do
   Call OneStep(nUnsatisfied, nIter) ' call subroutine "OneStep"
  Loop Until (nUnsatisfied = 0) Or (nIter >= MaxIter)
End Sub
```

To write the *OneStep* procedure, it is initially necessary to pass into the editor Visual BASIC of Excel (Tools > Macro > Visual BASIC Editor) then to insert a new module of code (Insert > Modul). One can then write directly the following Visual BASIC code.

```
'declaration of the global variables :
Dim T() As Integer        ' cellular domain
Dim V() As Integer        ' neighbor operator
Dim nV As Integer         ' number of neighbors
Dim nc As Integer         ' number of lines of the domain
Dim nl As Integer         ' number of columns of the domain
Dim nT As Integer         ' total number of cells
Dim nIter As Integer      ' number of iteration
Dim PopTot As Integer     ' total inhabitants in the domain
Dim NbGroups As Integer ' Number of social groups
Dim NbFree As Integer  ' Number of free cells = Nt-PopTot
Dim Threshold As Double       ' Tolerance level
Dim EstInitialize As Boolean  ' =TRUE if initialised
Dim CellsFree() As Integer  'contain the number of free cells
Dim Rg() As Integer           'define a random order
```

```
Public Sub ASTEP(nUnsatisfied As Integer,_
                 nIteration As Integer)
```

```
  Dim i As Integer ' n° of line
  Dim j As Integer ' n° de column
  Dim p As Integer ' random cell number
  Dim n As Integer ' number of the cell
  Dim SV As Double ' threshold for the neighbors

  If Not IsInitialize Then Call InitDomain
  Application.ScreenUpdating = False
  SV = Threshold * nV
  nUnsatisfied = 0
  'one calculate a random order to read the cells
  Call RandomRanks(nT,Rg)
  For n = 0 To nT - 1
    p = Rg(n)           ' number of random rank
    i = (p \ nc) + 1    ' number of the line associated to p
    j = (p Mod nc) + 1 ' number of the column associated to p
    If (T(i,j) > 0) And (NbDifferent(i, j, T(i, j)) > SV) Then
      nUnsatisfied = nUnsatisfied + 1
      Call ModeIndividual(i, j, T(i, j))
    End If
  Next n
  nIteration = nIteration + 1
  ' ouputs :
  Range("nUnsatisfied") = nUnsatisfied
  Range("nIter") = nIteration
  Range("Domain") = T
  Application.ScreenUpdating = True
End Sub 'AStep

' function to compute the number of foreigners in the vicinity :
Public Function NbDifferent(i As Integer, j As Integer,_
                            nG As Integer) As Integer
  Dim k As Integer
  Dim ii As Integer
  Dim jj As Integer
  Dim n As Integer
  Dim m As Integer

  n = 0
  For k = 1 To nV
    'compute the position of the neighbor n° k :
    ii = i + V(k, 1)
    jj = j + V(k, 2)
    'to take into account a toroïdal domain :
    If ii < 1 Then
      ii = ii + nl
    ElseIf ii > nl Then
      ii = ii - nl
    End If
    If jj < 1 Then
      jj = jj + nc
```

```
    ElseIf jj > nc Then
      jj = jj - nc
    End If
    'to compute the number of neighbors different from me (from n° of
nG group)
    m = T(ii, jj)
    If (m > 0) And (m <> nG) Then
      n = n + 1
    End If
  Next k
  NbDifferent = n
End Function

' subroutine of displacement of an individual of the type « numGr »
from the cell (i,j) to a randomly selected free cell :
Public Sub MooveIndividual(ii As Integer, jj As Integer, numGr
As Integer)
  Dim i As Integer
  Dim j As Integer
  Dim p As Integer
  Dim m As Integer
  Dim n As Integer

  ' random selection of a free cell
  m = Int(Rnd * NbLibres)
  ' to compure the coordonate of the cell
  n = CellsFree(m)
  i = n \ nc + 1
  j = n Mod nc + 1
  T(i, j) = numGr
  T(ii, jj) = 0
  FreeCells(m) = (ii - 1) * nc + jj - 1
End Sub

' subroutine to compute a random order to read the cells
' stocked in the data Rg() :
Public Sub RandomRank(Nb As Integer, Rg() As Integer)
  Dim i As Integer, j As Integer, tampon As Integer

  'one initialize with similar ranks
  ReDim Rg(Nb) As Integer
  For i = 0 To Nb - 1
   Rg(i) = i
  Next i
  Randomize
  For i = 0 To Nb - 1
    j = Int(Nb * Rnd) 'one choose randomly j in [0..Nb[
    'one permute i with j :
    tampon = Rg(i)
    Rg(i) = Rg(j)
    Rg(j) = tampon
```

```
  Next i
End Sub
```

13.4.3. Implementation into SpaCelle

Position of the problem

With **SpaCelle**[6], the modelling process is radically different for several reasons. The first one is due to the fact that SpaCelle is a genuine cellular automaton, but only a cellular automaton. So the only operation which a cell "can" do is to calculate its new state according to its own current state and to the state of the neighbouring cells. The transition function carries this treatment, which is the same for all the cells. The automaton is thus unable, for example, to move an "inhabitant" from a cell to another, for the good reason that there cannot exist an "inhabitant" in a cell and that the function of transition cannot treat a move, but only a change of cellular state. The general model of cellular automaton is thus very simple. The class of the models which one can simulate is in consequence limited. It is thus necessary to conceive the model in this specific context.

The second fundamental difference is in the description of the transition function. This one is not written in an algorithmic language like Logo or VBA. It is based on rules where order does not have any importance. These rules can be written initially in natural language, then, starting from a single diagram, one codes these rules in the Spacelle language. The syntactic form of a rule is written:

action = evaluation

where the part "action" is of the form: X > Y

and represents the transition to be carried out, i.e. the passage of state X to state Y. The sign " = " is the separator between the two parts of the rule and does not have other significance. The part "evaluation" is composed of an expression which combines various functions which evaluate the contents of the vicinity and of which the result represents the relevance of the transition to be carried out for a given cell.

Formulation of the Schelling's model in the cellular automata paradigm

The state of each cell can take three possible values (for two social groups):

L: means that the cell is not inhabited, it is free;

A: means that the cell is inhabited by an element of social group A;

B: means that the cell is inhabited by an element of social group B.

Two types of rules are defined:

[6] http://www.univ-rouen.fr/MTG/PatriceLanglois2.htm

1) Rule of removal: When an inhabitant (of type A or B) is dissatisfied, instead of moving inside the field, he leaves the field and disappears. The state of the cell then undergoes a transition of the type A > L or B > L. This departure produces a fall of the density of population. This rule requires an evaluation of the satisfaction of the inhabitant. That depends on the number of foreigners around him. If this number exceeds the tolerance level, there is dissatisfaction, the evaluation is 0, if not the inhabitant is satisfied, the evaluation must be 1.

2) Rule of moving in: When a cell is free (state L) and if the density of its population allows it (it should not exceed a certain threshold, for example 47.5% for each population, which leaves 2% of free boxes), it can receive a new inhabitant who comes then from outside. The state of the cell undergoes a transition L > A or L > B and the density of population increases a bit. The application of this rule depends only on the density of each population, if it is lower than the acceptable threshold, the evaluation must be 1, if not it will be given 0.

Formulation of the rules: as there are two categories of population, A and B, there are two rules for moving out and two rules for moving in. The rule basis is thus made of the four following rules:

```
A > L = PV(1 ; B ; 0.66 ; 1)
B > L = PV(1 ; A ; 0.66 ; 1)
L > A = DE(A ; 0 ; 0.475)
L > B = DE(B ; 0 ; 0.475)
```

The Schelling's model programmed with *SpaCelle* is thus reduced to these four lines! The first rule is read: "A becomes L when the proportion of B in a vicinity of radius 1 is higher or equal to 0.66". The 4th rule is read: "L becomes B when the density of B is lower or equal to 47.5%".

One can schematize these four rules by the following transition graph:

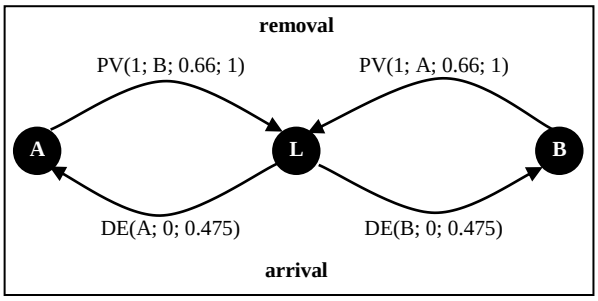


Figure 13.7. *Transition Graph.*

Other information, proper to the model, is defined in the window of parameter setting: for example, one can choose square or hexagonal cells, and one must choose the vicinity type: here the vicinity of Moore is selected (eight neighbours for square cells). Finally an asynchronous or random procedure is selected, allowing the actions (moving in and removals) to be done successively. In asynchronous mode, all the cells are treated only once in a different random order in each iteration. In random mode, at each iteration, one proceeds to N random drawing of cells. In this mode, some cells can thus be treated several times and other ones no time.

Operation: When a simulation is launched, for each cell, if it is in the status L, the two rules of moving in will arise, but only one, with the better relevance, will be carried out. In our case, they may give the same result: 1 if the density is lower than 47.5%. In this case of equality, a drawing of lots is carried out to choose one of the two rules. For a cell in the status A, it is the first rule (removal) which arises, it will be carried out if the number of B exceeds the tolerance level. It is the same if the cell is in the status B, the rule number 2 is carried out if the number of A in the vicinity exceeds the tolerance level (here 66%). It appears that a move has been almost simulated, since the departure of an individual A may reduce the density of A below the threshold, which allows the rule moving in of a new A to start next turn in a free cell, restoring the maximum density.

Parameter setting and alternatives of the model: The two fundamental parameters of the model which are the tolerance level (here 66%) and the density of population of each social group (here 47.5%) may be modified easily. One can also slightly modify the model by fixing the total density of inhabitants (for example 95%), without any imposed density for each social group. That gives an additional degree of freedom of which the analysis is of interest. That would thus result in the following rules:

```
A > L = PV(1 ; B ; 0.66 ; 1)
B > L = PV(1 ; A ; 0.66 ; 1)
L > A = DE(A+B ; 0 ; 0.95)
L > B = DE(A+B ; 0 ; 0.95)
```

A second alternative could be to bring some risk by adding a limited life span to the inhabitants (for example 1000 units of time in average with a standard deviation of 100). This allows releasing some cells from time to time, which causes a renewal of the locations, and improves little by little the regrouping of the individuals of the same social group.

```
A > L = DA(1000 ; 100)
B > L = DA(1000 ; 100)
```

A third alternative consists in changing the ray of the vicinity, instead of 1 one can take for example 3, 4 or 5 etc. A greater difficulty of regrouping is then noted.

Finally one can easily increase the number of social groups by duplicating the rules. For 3 groups A, B and C, a threshold of 66% and one total density of 96% one would thus have the following rules:

```
A > L = PV(1 ; B+C ; 0.66 ; 1)
B > L = PV(1 ; A+C ; 0.66 ; 1)
C > L = PV(1 ; A+B ; 0.66 ; 1)
L > A = DE(A ; 0 ; 0.32)
L > B = DE(B ; 0 ; 0.32)
L > C = DE(C ; 0 ; 0.32)
```

13.5. Comparison of the three implementations

The three implementations which we just presented enable us to understand easily the difference between cellular automata type (SpaCelle), multi-agents type (StarLogo) and general practitioner (Excel) platforms. In StarLogo, there are indeed two types of distinct entities, cells and agents, which is not the case with SpaCelle where a coding of the state of the cells is necessary to represent individuals (state L: unoccupied cell, state A or B, occupied cell). In StarLogo, the management of the moves of the individuals is done naturally through the mediation of the agents, which is impossible in a cellular automaton. For this reason the model under SpaCelle dissociates each move into two independent mechanisms, the departure and the arrival, whereas there is only one mechanism of moves with agent-based model in StarLogo. This uniqueness ensures the invariance of the total population. The simulation behaviour may thus lead to differences of results between SpaCelle and StarLogo. It is the case when the density of population is equal to 98% and the tolerance level to 20%. A strong aggregation appears quickly in SpaCelle. This is explained by the fact that the number of dissatisfied cells is very large compared to the number of free cells. Thus, at the first iteration, there are much more departures than arrivals and there is a temporary decrease of the population which gives to the individuals more freedom to aggregate. This phenomenon does not occur with StarLogo because the density of free cells (houses) remains by construction equal to 2%.

Another great difference between these implementations refers to the language of description of the model. In StarLogo as in Excel, it is an algorithmic language (Logo and Visual BASIC) whereas SpaCelle uses a very simple language in which the writing order of the rules has no importance. This difference is fundamental on several points:

- an algorithmic language requires the acquisition of a programming competence long to acquire, contrary to the language of SpaCelle;

- on the other hand the class of the models likely to be implemented with a programming language is larger than with this language of rules;

- it has been seen that four lines were enough to describe the model of Schelling in SpaCelle, whereas forty lines are necessary in StarLogo or Excel.

Last important difference: for a simulation of 10000 cells, the number of iterations treated in 5 minutes of execution varies considerably according to the implementation: StarLogo treats 6 iterations, Excel 1730 and SpaCelle 16000.

13.6. Analysis of the model

The study of a model, or more exactly a family of models, is based on the study of the properties of the simulations results of certain models of this family, according to the values given to the parameters. It seems interesting to study here two properties resulting from simulations of Schelling's model, which are materialized by what is called observation or output variables of the model. We will study here the ***convergence*** and the dynamics of ***aggregation*** through two observation variables, "number of dissatisfied" and "average aggregate size" which we will specify. We will see that the model may produce a strong aggregation of the population without converging, and that it may also converge without producing aggregation. We indeed note, that for certain values of the parameters, the behaviour of the outputs is stable, i.e. it gives reproducible series on several simulations with a weak fluctuation. On the other hand, for other values of the parameters, the behaviour becomes chaotic, the time of convergence is unforeseeable, and then the average behaviour does not have significance anymore. In particular, we cannot explain why, in the zone of instability, which corresponds to a very high density of population (around 98%), one goes suddenly from a total absence of aggregate for a tolerance of 2 foreigners, to a maximum aggregation for 3 foreigners.

13..6.1. Family of models and elementary model

We saw that Schelling's model could be materialized in different ways, according to the values given to some parameters or to the selected transition mechanism. What is called "Schelling's model" is in fact a family of models. An element of this family corresponds to a concrete model, workable, obtained after having fixed all the parameters. We are using here the notation $M(N, d, n, S)$ to indicate one of Schelling's models, where the brackets contain the parameters. If the parameters are fixed at concrete values, we obtain a **concrete model**, if the parameters are regarded as variables, the notation designates the **family of models** (or general model). N indicates the total number of cells (generally arranged within a

square field), *d* represents the total density of population, at a rate of 1 individual per cell at most, *n* indicates the number of cells of any vicinity (*n* = 8 in general), *S* is the tolerance level, that is to say the maximum proportion of foreigners (i.e. individuals belonging to social groups different from that of the central cell) with whom any individual is able to put up, in order to be in a "satisfied" state, on the contrary, he is in a "dissatisfied" state and will have to move towards a free cell. The value of *S* gives thus the maximum number *k* of foreigners in the vicinity of a cell:

K	S values
0	0 <= S < 0.125
1	0.125 <= S < 0.25
2	0.25 <= S < 0.375
3	0.375 <= S < 0.5
4	0.5 <= S < 0.625
5	0.625 <= S < 0.75
6	0.75 <= S < 0.875
7	0.875 <= S < 1
8	S = 1

Figure 13.8. *Number of tolerated foreigners according to the threshold's values S.*

13.6.2. Measure of convergence

One notes C_t a ***configuration*** at the moment *t*, (series of states of all the cells at the moment *t*) and T the ***global transition mechanism*** which, to any C_t configuration, associates a C_{t+1} configuration at the following moment. It will be said that the model converges if there is a value of time beyond which all the configurations are equal. A simulation is a finite or infinite set of successive configurations (C_0, C_1... C_i...), built from an initial configuration C_0, by successive applications of the transition mechanism: $C_{i+1} = T(C_i)$.

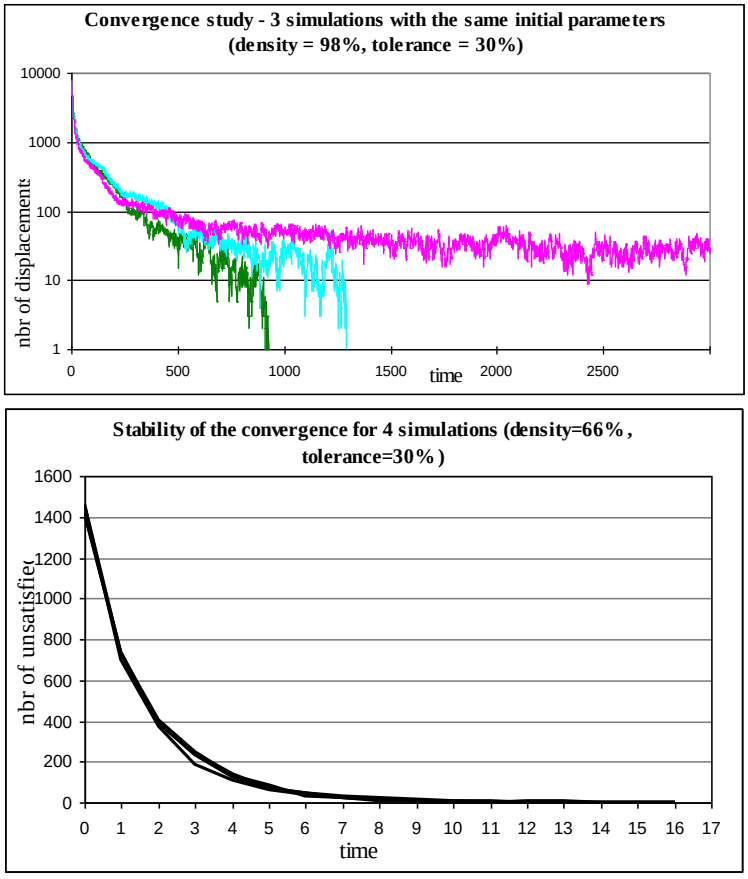


Figure 13.9. The number of dissatisfied evolves in a regular or chaotic way.

In the case of Schelling's model, the convergence is studied through the output variable x_i, which counts the number of dissatisfied present in the configuration *Ci*. One says that the simulation is converging at time T when the series (x_0, x_1...,x_i...) is null from the value *t* of the index onwards. From this moment, all the individuals are satisfied, therefore all the following configurations are equal and the simulation can stop. Figure 13.9. shows the regular or chaotic character of the output variable x_i = "number of dissatisfied in the configuration C_i" according to the parameters of the model. Initially for three simulations carried out with d = 98% and s = 30%, only two simulations are converging before 3000 iterations. Convergence is possible, but the moment of convergence is unforeseeable for this model. In the second figure, for d = 66% and s = 30%, convergence is regular and fast, it is stable from one simulation to another.

13.6.3. Measure of aggregation

The objective of Schelling's model is to show that a spatial gathering of individuals (a socio-spatial segregation) is produced, even when their tolerance level is rather high. But for an accurate analysis of this characteristic, one cannot be satisfied by a simple visual appreciation of the aggregation. It is necessary to measure it. We have chosen to measure for each configuration of a simulation, ***the average size of the horizontal and vertical homogeneous transects***, which we will more simply express by "***average size of aggregate***". This observation variable is calculated in the following way: for each line and each column of the configuration, one calculates the average number of contiguous cells of the same population A or B. In the example of a chessboard with the alternation of a square of population A and of population B, one finds exactly 1: there is no aggregation. Conversely, if the individuals of the type A are grouped in only one related and compact package, the B remaining around (also forming a related package), one can reach an average size of aggregate higher than 50 for a field of a side of 100 cells.

13.6.4. Choice of the transition mechanism

The system's dynamics are dependent upon the various programming strategies of the global transition mechanism *T*. When a family leaves its dwelling place, it does not know in advance if it will be satisfied or not by its new residence. It is why an individual is transferred toward a free cell, without considering if this new location is satisfactory for him and even less if it will remain so for a long time. These reasons lead to define a "mechanism" of probabilistic transition. But there are several ways for it and these ways are not equivalent. The mechanism used for the test is completely asynchronous, (it is the model developed with Excel). It consists in sweeping all the cells or all the inhabitants in a random order and moving immediately every dissatisfied inhabitant towards a randomly selected free cell.

One should not believe besides, that the model would be more powerful by choosing as a destination, a free cell immediately satisfactory for the new comer. A maximum limitation of the useless displacements prevents from the random production of some agitation. That allows a progressive auto-organization of the inhabitants, who are randomly falling on small islands of stability, which will increase and solidify by themselves in the course of time.

13.6.5. Necessary condition for convergence

Let $M(N, d, n, S)$ be one of Schelling's models, taking into account its parameters N, d, n, S. It may converge if a configuration exists, where all the

individuals are satisfied. Such a configuration will be considered satisfactory. A satisfactory configuration for all tolerance levels is known as completely satisfactory. For example, a configuration in which the number of free cells is sufficient to allow a complete partition of the two populations by interposing free cells is completely satisfactory. Conversely the impossibility of building a satisfactory configuration for given parameters (*N*, *d*, *n*, *S*) prohibits any convergence. But the existence of a satisfactory configuration *C* is not sufficient for the convergence. In the case of some initial configurations, the dynamics might prevent the model from converging towards *C* or give a negligible probability to succeed.

Let us take a square field of 100 by 100 cells. Let us take N = 10 000, and a density of population d = 98%, which gives N_L = 200 free cells. With this size, a configuration containing two parallel lines of free cells may be built, dividing the space into two homogeneous strips, with the same number of cells, in which N_1 = 4900 cells of group A and N_2 = 4900 cells of group B are laid. This strip configuration is completely satisfactory for it is satisfactory whatever the tolerance level. On the other hand its probability of production by the model is very weak. One could think that the circle, which has the most "concentrated" shape in an Euclidean metric space, would give a better result. In fact, the round shape is not optimal, for its perimeter measures at least 246 cells, against 200 cells for the strip. This comes from the Moore topology and of the toric closing of space.

The density of 98% is thus an upper limit for a complete partition of the two populations. It is so the maximum density for the existence of a completely satisfactory configuration of 100 by 100 cells. For a field of 10 by 10 cells, this percentage falls to 80%.

13.6.6. Study of the convergence of the model with d=98% and S=66%

We have chosen these parameters (with always = 10 000, N_1 = N2 = 4900, N_L = 200) because the density of 98% matches the limit of existence of a completely satisfactory configuration and the threshold of 66% (2/3) is "socially" interesting since it suits a rather large tolerance (in any case, higher than the proportion of foreigners in the area, which is 49%). This choice of parameters converges rather quickly (on average 15 iterations with a standard deviation of 2,4) towards a rather well aggregated configuration (average size of the aggregate of the order of 4). Moreover the convergence is regular (few variations from a simulation to another). We will try to understand through this observation (but not to explain it here in a mathematical way) why the system converges and produces an interesting level of aggregation.

The patterns of vicinity including 6, 7 or 8 foreigners exceed the tolerance level of the central individual. It will be dissatisfied and have then to move (Table 13.1). So the observed probabilities concerning these values will fall to zero, to the benefit of the values from 0 to 5, the probability for 0 being the highest since it corresponds to cells without contact with foreigners, they are completely satisfactory vicinity patterns. The reason comes from the fact that these patterns, when they are in contact with each other, become more and more stable as their size is growing. The disturbance can then only occur on the edges of this homogeneous, aggregated form.

Figure 13.10 makes clear this mechanism, it represents the evolution along the time of the probability $P(X = n)$ for a cell to have n foreign neighbours. At the stage of initialisation, (back part of the graph) the cells have observed probabilities in conformity with the theory (hypergeometric law) because of the randomness of the configuration. Few cells have completely identical or completely different neighbours, a majority have between 3 and 5 different neighbours. The application of the transition mechanism with a threshold fixed at 2/3 generates a chain of reconfigurations, the individuals satisfied with their vicinity are surrounded gradually by individuals of the same group, favouring thus the construction of "blocks" of identical individuals, which increases the number of cells having no foreign neighbour, to the detriment of the vicinities of 4 foreigners or more. This leads, after about twenty iterations, to a deep modification of the vicinity patterns, and to the emergence of aggregates (front part of the graph).

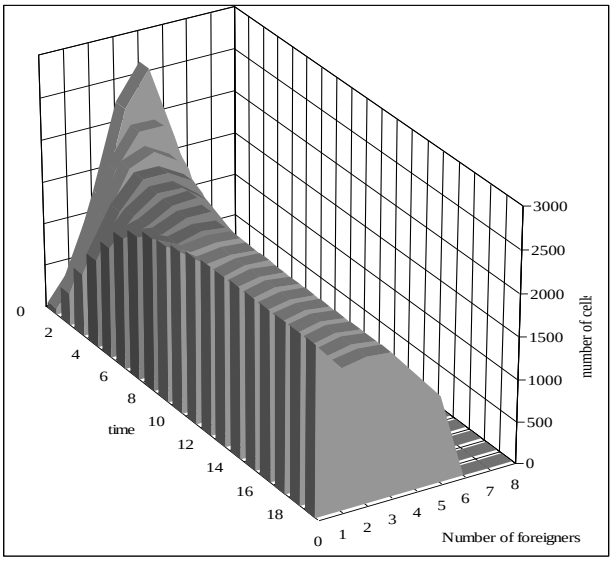


Figure 13.10. *Evolution of the number of cells having N foreign neighbours (N = 10000 cells, d = 95% and S=66%).*

13.6.7. Behaviour of the model in the space of parameters

The study of the behaviour of the output variables in the space of the parameters, given in figure 13.11 for the size of the aggregate, highlights four different zones of the behaviour. A great zone (1) (grey on the graph) where the model is stable, converges quickly (in less than 15 iterations) and produces small aggregates (size lower than 5). An hatched zone (2) where the tolerance is 0 or 1 foreigner and the density between 86% and 98%. In this zone, convergence is difficult and unforeseeable and there is no aggregate. A third small zone (the peak in black and white) produces like the previous one a chaotic convergence, but is the place of a very strong aggregation, of which the maximum is reached for 2 foreigners (the average size of the aggregate is then 58). It appears that this peak of aggregation is close to an abyss, the hatched zone, where curiously, there is no aggregation. Finally a fourth zone, not represented on the graph, relates to the band located beyond the density of 98% where there cannot be convergence anymore. These various zones may be observed on figure 13.12, which shows a mosaic of final configurations, obtained by the variation of the two parameters of density (*d*) and tolerance (*t*).

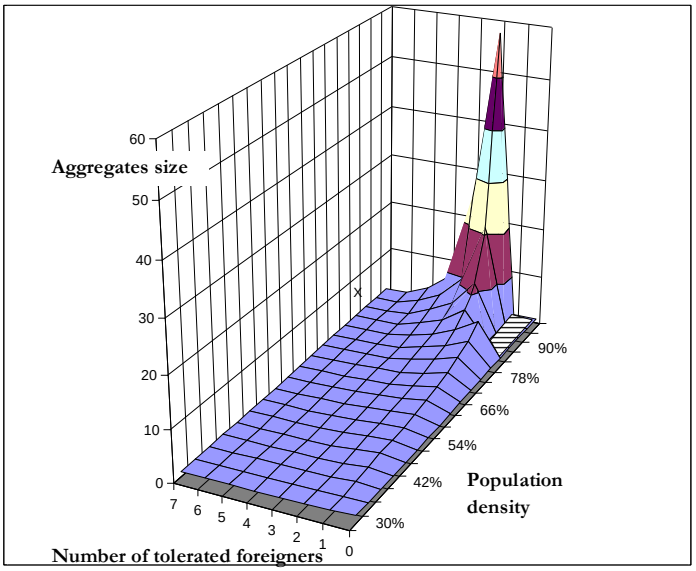


Figure 13.11. *Sizes of aggregates according to the number of foreigners and the density of population.*

The first two columns represent a sample of zone (1), the convergence is fast and the aggregation weak. For the first column, there are sufficient free places for all the individuals, a great number find themselves with only one or two neighbours, even

completely isolated in their environment. For the second column (d = 66%), the model converges also rather quickly and produces more important aggregates, even with relatively weak tolerance levels. Thus, in zone 1, the convergence is explained by the ability of the model "to use" the free places for the division of the groups. As the density increases, the division of the groups by free places implies that the number of groups decreases and then the aggregation increases.

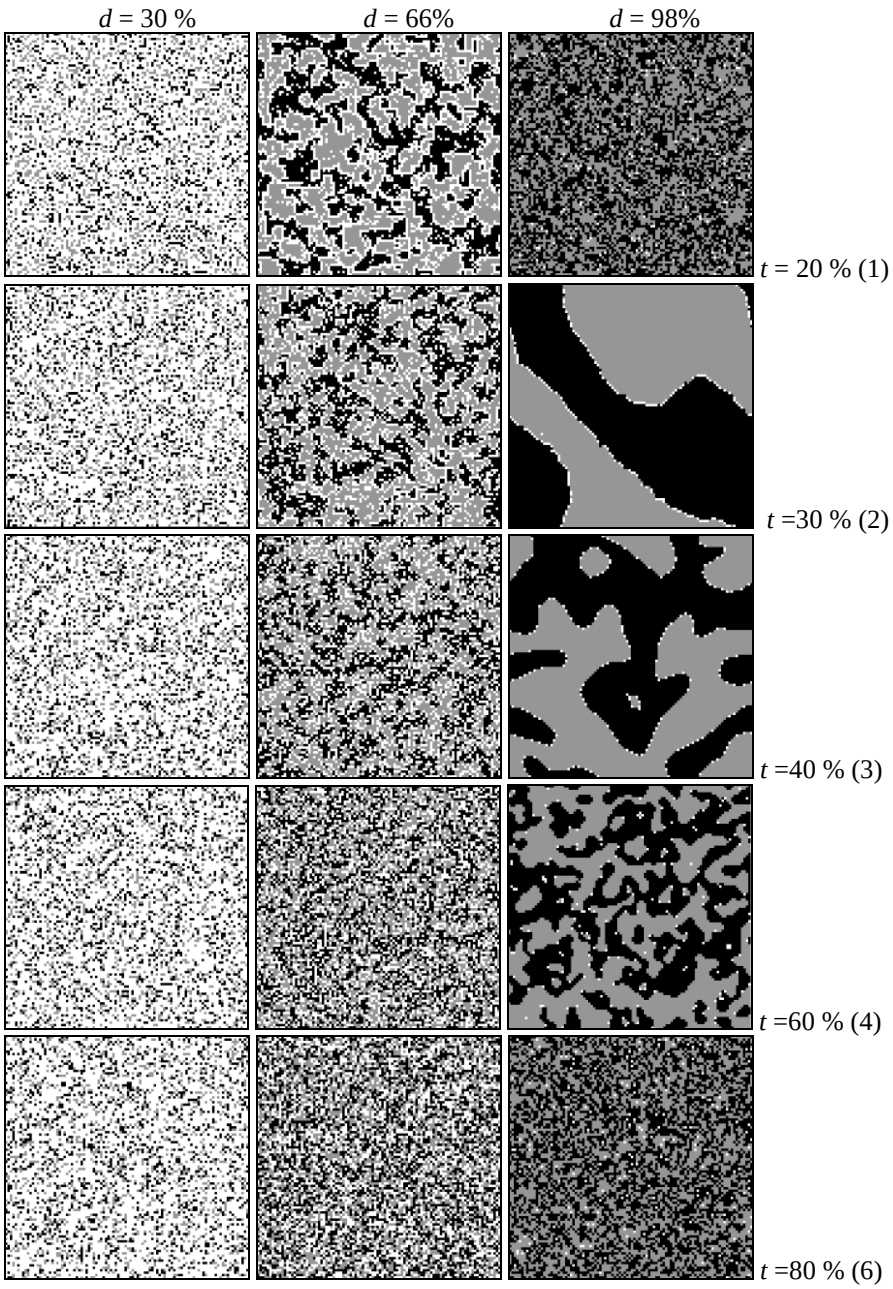


Figure 13.12. *Fifteen simulations on an area of 10 000 cells, two populations of the same size, according to the density d of population and to the tolerance threshold.*

The last column incorporates zones 2 and 3. In zone 3, the model converges by producing all the more important aggregates that the tolerance is weak. Under the threshold of 2 foreigners, one is situated in zone 2, the probability of convergence of the model is extremely weak, the number of free places is too limited to allow the formation of small pockets of stability which could develop. There is here a bifurcation point, with a first type of behaviour for tolerance levels equal to or higher than 25%, for which the model produces increasingly important aggregates, and a second type for lower thresholds where the model does not produce aggregates anymore.

13.7. Conclusion

The aim of this chapter was to show the different stages of the building of a simulation model, through the example of Schelling's model. The theoretical development of the model, its diagrammatic construction according to the UML standard, its implementation on various computing platforms, the simulation and exploration of its behaviour in the space of the parameters were thus presented.

This exercise has enabled the study of an alternative to Schelling's model based on the influence of the density of population on the construction of segregated areas. This work made us discover a very variable behaviour of the model according to the selected zone in the space of the parameters. It has showed that the domain in which it has been used to the benefit of a pedagogic purpose - the social segregation does not proceed inevitably from a lack of tolerance of the individuals - is very narrow and hides in fact great variety of other rather astonishing behaviours, for a system which is simple by its rules, but of which combining complexity is at the same time considerable. The mathematical explanation of this behavioural diversity for very close values of the parameters remains to be established.

13.8. References

[PAN 03] PANCS R., VRIEND N. J. "Schelling's Spatial Proximity Model of Segregation Revisited", Working Paper n° 487, Dept. of Economics Queen Mary, University of London, 2003.

[SCH 69] SCHELLING T.S. "Models of Segregation", *American Economic Review*, 59-2, p.488-493, 1969.

[SCH 71] SCHELLING T.S. "Dynamic Models of Segregation", *Journal of Mathematical Sociology*, 1, p.143-186, 1971.

[SCH 78] SCHELLING T.S. *Micromotives and Macrobehaviour* N.Y Norton and Co, 1978, traduction française: *La tyrannie des petites décisions*; Paris, Presses Universitaires de France,1980.