Int J Comput Math Learning (2009) 14:195–202
DOI 10.1007/s10758-009-9150-1

Computational Diversions: There Goes the Neighborhood

Michael Eisenberg

Published online: 17 September 2009

A recurring theme of this column is that just a little bit of programming can go a long way—that is, by playing with relatively short, understandable programs, one can explore wondrous, and largely uncharted, intellectual landscapes. A case in point is the famous "self-forming neighborhood" model devised by Thomas Schelling. Schelling (who won the 2005 Nobel Prize in economics) described this model in his remarkable book *Micromotives and Macrobehavior* (Schelling 1978). Numerous researchers have experimented with (and extended) Schelling's model since that time, but it seems to me that the model is so rich in possibilities that there are undoubtedly tons of new experiments just waiting to be tried.

Schelling's model is an attempt to understand and explain the formation of distinct geographic neighborhoods of various types—ethnic, religious, racial, linguistic, among other possibilities. As Schelling writes:

> People get separated along many lines and in many ways. There is segregation by sex, age, income, language, religion, color, personal taste, and the accidents of historical location. Some segregation results from the practices of organizations. Some is deliberately organized. Some results from the interplay of individual choices that discriminate. Some of it results from specialized communication systems, like languages. And some segregation is a corollary of other modes of segregation: residence is correlated with job location and transport. [Schelling 1978, p. 137]

The model described in *Micromotives and Macrobehavior* is especially provocative in this light, since it shows how segregation can emerge from the independent choices of agents, none of whom is all that strongly invested in the homogeneity of their neighborhood. Here's the basic idea: we will create an abstract model of a "town" using a 10-by-10 grid of square cells. (If you like, you can think of each of the cells as a "house" in which a person or family might choose to live.) Our town has two types of inhabitants in it, whom we will call "Type 1" and "Type 2"; at the start of our sample simulation, our little town has 44 cells inhabited by Type 1, 44 cells inhabited by Type 2, and twelve cells empty.

M. Eisenberg (✉)
University of Colorado, Boulder, CO, USA
e-mail: ijcml-diversions@ccl.northwestern.edu; duck@cs.colorado.edu

Figure 1 shows a graphical representation of our town; the cells have been assigned at random to occupants, so that there is no particular prearranged order to the living arrangements of the town. (By the way, a "Type 1" cell is represented in all figures as light gray; "Type 2" as dark gray; and an "empty" cell is white.) One other special feature of our sample grid should be mentioned: we will allow for "wraparound" in this simulation so that each cell has exactly eight surrounding adjacent neighbors. That is, by moving one space to the right of the rightmost column in Fig. 1, for instance, one would find oneself in the leftmost column of the figure. In effect, the grid shown in Fig. 1 can be thought of as sitting on a torus, so I have dubbed our little village "Torustown".

Now, starting from the randomized arrangement in Fig. 1, we let "unhappy" occupants move from their locations. An "unhappy" occupant is one of Type X that has fewer than four neighbors of its own type: that is, a Type 1 occupant needs at least four Type 1 neighbors (among the possible eight) to be content, and a Type 2 occupant needs at least four Type 2 neighbors. To take an example: the Type 1 occupant in the second row from the bottom and second column from the left in Fig. 1 is unhappy, since it has only 1 Type 1 neighbor (along with six Type 2 neighbors and an empty-site neighbor).[1]

Our simulation, then, will proceed as follows: we look to see if there are any unhappy occupants in Torustown. If not, we are done, and the town is in a stable arrangement. On the other hand, if there *are* unhappy occupants, we choose one at random and let that occupant move to a randomly chosen empty site. We continue this process indefinitely, for as many iterations as needed, until the town arrives at a stable arrangement. Note that when an occupant moves, it may make its original, now-abandoned neighbors change from unhappy to contented (or vice versa); or it may make its new neighbors rethink their happiness level. By simulating the town's evolution on the computer screen, one can see the cell contents hopping about the grid; when I ran a simulation starting from Fig. 1, it took 135 iterations before the town settled on the stable configuration shown in Fig. 2.

The rapidity with which independently shifting occupants change the arrangement from Figs. 1 to 2 on the computer screen is rather shocking. In Fig. 2, we have gone from the highly "mixed" arrangement of Figs. 1 to 2 starkly defined monolithic neighborhoods, and we have done it in pretty short order. Note, again, that the rule by which an occupant is "unhappy" does not suggest what we might think of as extreme bigotry: for instance, a Type 1 occupant is content if only four of eight neighbors (but no fewer) are of its own type. Still, from the iterated movements of occupants, none of whom necessarily *wants* to live in a homogeneous neighborhood, we end up with the stable arrangement of Fig. 2.

It's hard not to watch the transition from Figs. 1 to 2 and feel tempted to draw all sorts of troubling, though possibly unwarranted, conclusions about human geography. Speaking from my own political views—and having grown up in New York City and Boston, with their own vivid histories of (not always friendly) neighborhood demarcation—I am disturbed at the ease with which Torustown segregation arises from such straightforward, and apparently innocuous, starting conditions. On the one hand, there are arguably positive aspects to having disparate neighborhoods in a city: they provide places where subcultures can flourish, where particular styles of restaurants or theaters or dance clubs can find a local audience, where newcomers (e.g., immigrants) can find a comforting taste of home. On the other hand, they can also reflect or spawn xenophobia, isolation, and prejudice. One feels there really ought to be some compromise between the extremes: personally, I would

[1] It should be noted here that my simulation differs from Schelling's (1978) in several respects–his criterion for "unhappiness" was a bit more elaborate, and his model (which used dimes and pennies as markers) did not employ wraparound, for instance—but the basic idea is the same.

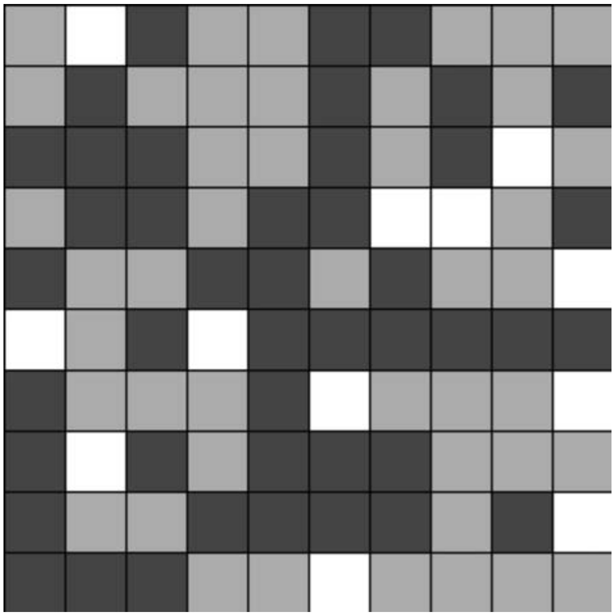

Fig. 1 "Torustown" in its initial state, occupied by 44 (*light gray*) Type 1's and 44 (*dark gray*) Type 2's. The *white* cells are unoccupied. Here, the cells have been randomly assigned to occupants, so there is no particular pattern to the arrangement

not want to live in the (entirely neighborhoodless) Torustown of Fig. 1, or the (highly segregated) Torustown of Fig. 2.

Still, before getting too worried about the meaning of this simulation, it should be pointed out that Schelling's model is abstract in the extreme; it does not reflect all sorts of factors that could either reinforce or mitigate the segregation-forming tendencies suggested by the move from Figs. 1 to 2. With that in mind, we can vary our original simulation rules to see how they affect the (depending on one's point of view) potentially depressing results of our starting example.

Suppose, for instance, that we make slightly more nuanced assumptions about our Type 1 and Type 2 occupants in Torustown. In our original rule, an occupant wanted at least four neighbors just like itself; and even an empty neighbor counted as "unlike" for these purposes. Now, let's say that an occupant does not want more than four, or fewer than one, *occupied* sites unlike itself. We have changed a couple of elements here: first, a Type 1 occupant could be happy with only three Type 1 neighbors, as long as at least one of the remaining five neighboring sites is empty. Second, a Type 1 occupant will not be happy if surrounded entirely by only Type 1 neighbors. (Analogous conditions hold for Type 2's as well.)

Using this rule instead of the somewhat more restrictive rule for the original simulation, and beginning with a random arrangement of 38 Type 1's, 38 Type 2's, and 24 empty sites, I arrived at a stable configuration for Torustown in 344 iterations. The result is shown in Fig. 3.

It seems clear that we've succeeded in "breaking up" the fortress-like neighborhoods of our first simulation—perhaps a bit too well. Note that every single occupied cell in Fig. 3, as per our new rule, has at least one neighbor of the alternate type. Moreover, there are

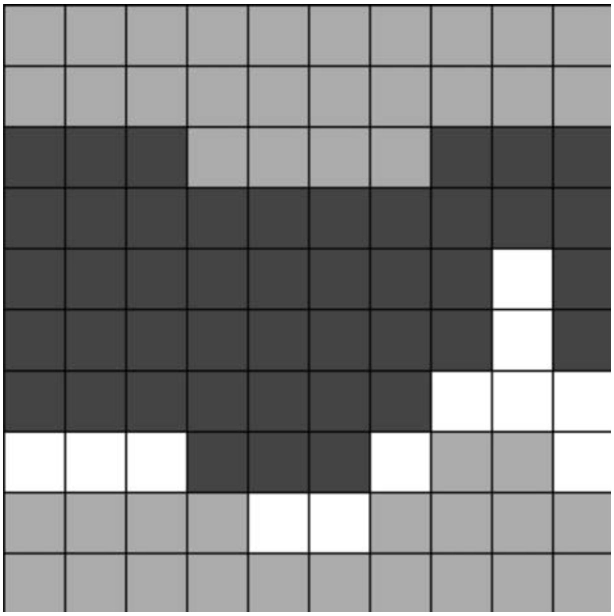

Fig. 2 Torustown, after starting from the arrangement in Fig. 1, and allowing a sequence of moves by "unhappy" occupants. Note that there is now a connected *light gray* Type 1 neighborhood (at the top and bottom of the figure) and a connected *dark gray* Type 2 neighborhood. (Recall that because of the wraparound feature, the *right* and *left columns* are effectively adjacent, and the *top* and *bottom rows* are likewise adjacent.)

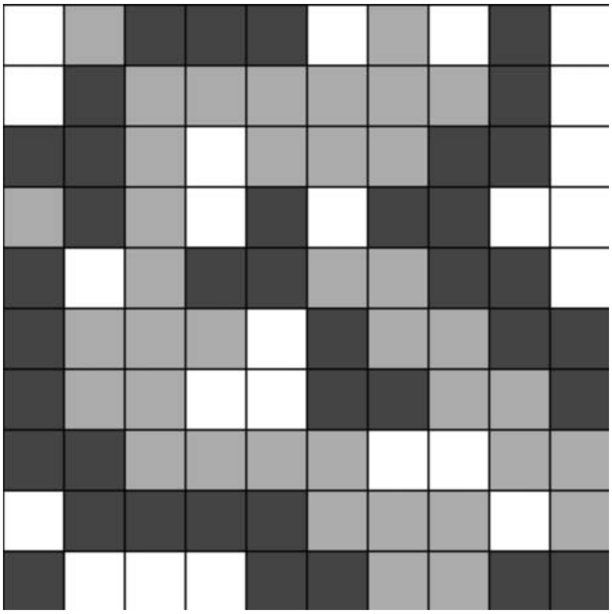

Fig. 3 A stable Torustown configuration with the more "relaxed" neighbor rule described in the text

some cells in the stable configuration of Fig. 3 that clearly would not be content using our original rule. For instance, consider the Type 1 (light gray) cell in the third row counting from the bottom and third column counting from the left. This cell has only three Type 1 neighbors; it also has one empty neighbor, and four Type 2's. Using our new rule, the empty neighbor does not count as an increase in "discomfort" for the cell.

The alert reader may also have noticed that in our second simulation, the number of empty cells has increased from 12 (in Figs. 1, 2) to 24 (in Fig. 3). The reason for this is that, empirically, it is hard to achieve a stable configuration of Torustown cells unless there are a good number of empty sites to distribute as the occupants move about. When I attempted the simulation with only 12 empty cells, the town had still not stabilized even after many hundreds of iterations.[2]

The result of our latest simulation is still, perhaps, not too cheerful: we have destroyed the neighborhoods entirely. Perhaps, then, we could try another approach in which at least one type of occupant is completely comfortable with any arrangement of neighbors. Let's say that the Type 1 occupants are just as they were originally—they want at least four Type 1 neighbors. Type 2 occupants, however, are oblivious to distinction; they are always content, no matter what their neighborhood looks like. With this new condition, we try a third simulation starting with 44 Type 1's and 44 Type 2's, randomly arranged. After 600 iterations, we end up with the still-unstabilized configuration shown in Fig. 4.

Our new rule, in which Type 2's are happy no matter what, has paradoxically prevented Torustown from finding a stable *modus vivendi*. Even after 600 iterations, there are many unhappy Type 1's in Fig. 4—all hoping, eventually, to move elsewhere. Consider the Type 1 occupant in the second column from the right and third row from the bottom: it has five Type 2 neighbors, and none of them will ever move, since they are perfectly content where they are. Thus, that particular site cannot ever be happily occupied: Type 2's will not move into it (they do not need to move at all), and Type 1's will always be discontent with their surroundings. In effect, by making one part of the population more easygoing, we have made the other part unavoidably unhappy: the Type 2's will not move for their own reasons, and thus will never get out of the way of the Type 1's.

It seems, then, that mitigating the tendency of our original model toward segregation is, in fact, not so easy. When we tried a rule in which tenants wanted at least one unlike neighbor, we apparently ended up dissolving our neighborhoods as in Fig. 3; and when we removed any preferences whatever from one part of the population, we ended up with a non-stabilized situation as in Fig. 4.

Let's try one more idea. Suppose that we imagine a third type of tenant, creatively named "Type 3". Our Type 3 tenants represent a small minority, and intuitively we can think of them as having a foot in both the Type 1 and Type 2 communities. (Perhaps, for example, these are bilingual folks, if one thinks of Schelling's model along linguistic lines.) Type 3's do not move; they're happy wherever they are, since they get along just fine with any neighboring occupants. As for Type 1 and Type 2 occupants, they obey the same rule as in the original simulation—they want 4 of "their own" among their neighbors—but now Type 3 occupants can count as "their own" for these purposes. In other words: a Type 1 occupant will be happy with three Type 1's and one Type 3 among its neighbors, and similar conditions hold for Type 2.

[2] Parenthetically, one could artificially impose an arrangement in which all 100 Torustown cells are happily occupied according to our new rule, with 50 each of Type 1 and Type 2. I leave this as an exercise to the reader.

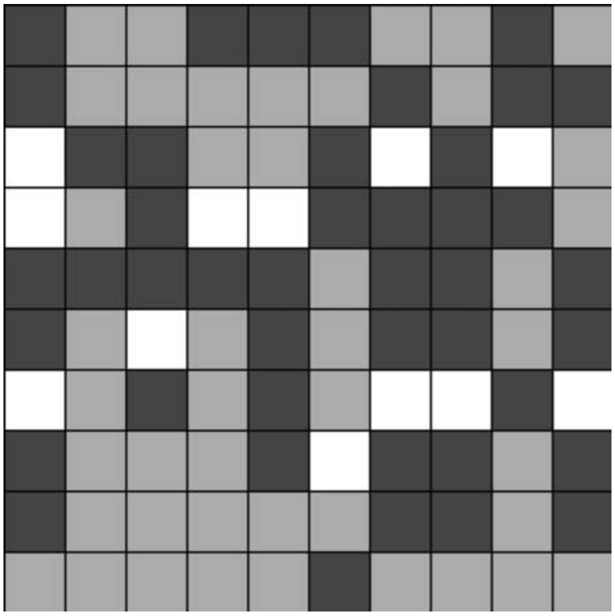

Fig. 4 A (non-stable) Torustown configuration with "restrictive" Type 1's and "oblivious" Type 2's, as described in the text

Figure 5a, b, c shows the result of simulating Torustown with 0, 6, and 10 Type 3 occupants, respectively, until a stable arrangement is found. (In each case, there are 12 empty sites, and the remainder are divided evenly between Type 1 and Type 2.) Figure 5a is just a repeat of our original simulation, and, much as in Fig. 2, two monolithic neighborhoods have formed. In Fig. 5b, we see a little bit of change in the town's style; here, the six Type 3's are shown in an intermediate shade of gray. Consider the Type 3 occupant in the second row from the bottom and third column from the right in Fig. 5b. If that cell were to be either Type 1 or Type 2, it would now have an unsatisfied neighbor to its left or right; but as a Type 3 cell, it allows for some "bridging" between the populations. Figure 5c, with 10 Type 3 cells, allows for even greater mixing at the boundaries between the Type 1 and Type 2 populations, without destroying neighborhoods altogether. I'm not sure if I'd be exactly thrilled to live in the Fig. 5c Torustown, but it looks like it might be a happier place than Fig. 2 (or Fig. 5a), and, with its neighborhoods, a more culturally vibrant place than Fig. 1.

The few simulations described here hardly begin to scratch the surface of investigations that one could try. As mentioned, Schelling's original model has been extended and varied by numerous researchers, and the literature is extensive; I make no claims to originality in the ideas shown here, since I imagine that others have tried these or similar simulations. Still, there are all sorts of additional questions that one might ask, and explore, with the aid of a little programming. Here are just a few:

- One of the less realistic touches in all of our models is the choice of a random empty site into which an occupant will move. In other words, an unhappy Type 1 occupant is liable, in our models, to move to a spot where it would be made even more unhappy. People, realistically, tend to look before they move: so we might alter the simulation so

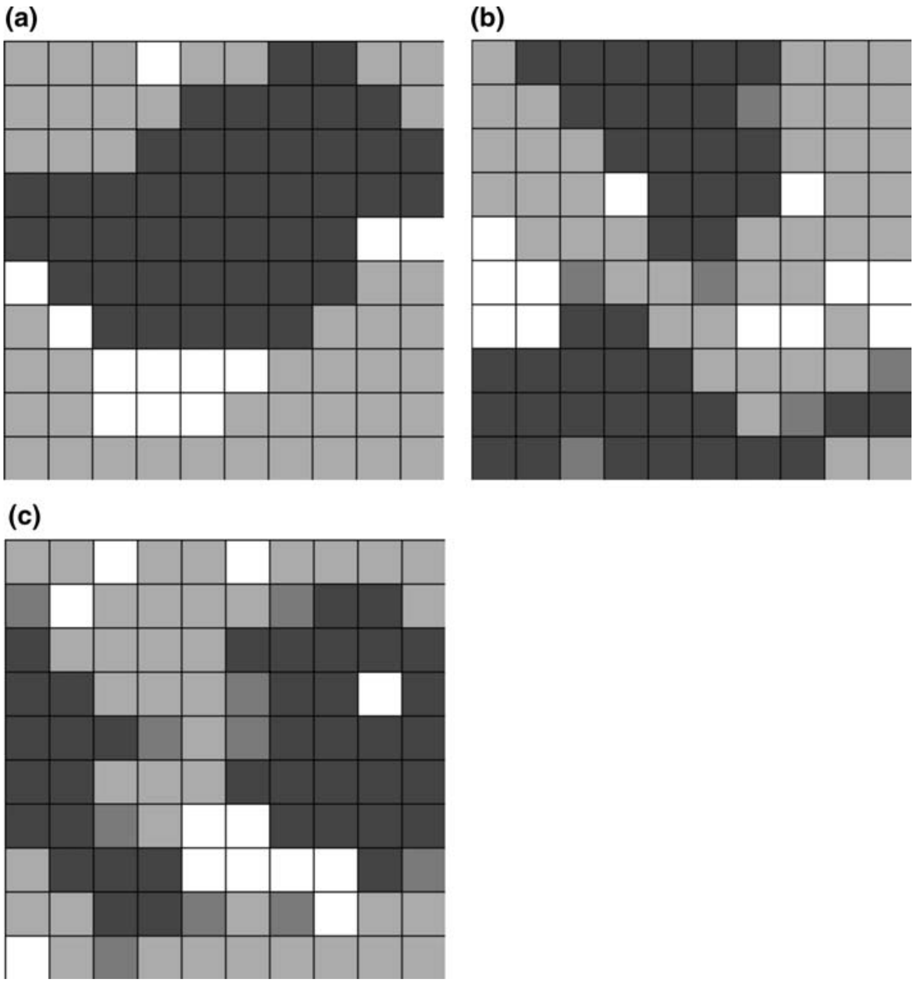

Fig. 5 Stable Torustown arrangements with 0, 6, and 10 Type 3 occupants, shown in medium gray. In **a** (*upper left*), we have our original problem of monolithic neighborhoods. One can see hints of "boundary mixing" in **b**, and much more of this phenomenon in **c**

that a cell will move only to an empty cell in which it be satisfied after the move is complete. How might this alter our simulations?

- Rather than model our populations through uniform traits, we might wish to try some sort of probabilistic decision rules. Just to imagine an example: some Type 1 occupants (maybe a quarter of them) might be content with only three Type 1 neighbors rather than four. How would these probabilistic, or perhaps nondeterministic, rules affect our simulations?
- Suppose rules change over time. For example, a given tenant might not want to move more than three times during any simulation; three moves, and it's ready to put down roots regardless of surroundings. How would dynamically changing rules affect the Torustown world?

I have been sufficiently intrigued by these models, and these untried variations, so that I am sure to return to this topic in a future column. Readers who have their own ideas or experimental results based on the Schelling model are happily urged to send email to: ijcml-diversions@ccl.northwestern.edu.

1 Readers' Diversions

This is also a good time to thank several readers who have sent email in response to earlier columns. Replying to the "Rounded Fractals" column (13:1), Forrest Stonedahl of Northwestern University sent in a correct solution to the "circularized pattern generation" problem of Fig. 5b. Michelle Wilkerson, also of Northwestern, incorporated "circularization" into a lovely IFS-generation applet that she has written: readers are encouraged to play with the system at: http://ccl.northwestern.edu/~michelle/ifs/ChaosFractalToolkitCirc.html.

There were several responses to the column on recursive humor (13:2). Forrest Stonedahl (gently, but correctly) noted that I played a bit fast-and-loose with the term "tail recursion" in the column, using it as more or less synonymous with "infinite looping". Strictly speaking, a "tail recursive" call is one in which there is no additional computational work to do once the call has been evaluated (see the textbook by Abelson and Sussman (Abelson et al. 1996) for a good discussion). As such, tail recursion is a handy way to write programs with infinite loops in them, but it need not be used for that purpose; there are all sorts of reasons to write tail recursive procedures, many of which have nothing to do with infinite loops.

Dor Abrahamson of Berkeley wrote a wonderful anecdote in response to the column in which he recounted a recursive conversation with his 4-year-old daughter that left her helpless with laughter. (The conversation began with "What did you do today?" and the answer to that question ended up including the answering of the question itself, which included answering the question itself…) My colleague Clayton Lewis at CU Boulder also pointed out that fans of recreational self-reference and recursion could do no better than explore the wonderful puzzle books of the logician Raymond Smullyan.

References

Abelson, H., Sussman, G. J., & Sussman, J. (1996). *Structure and interpretation of computer programs* (2nd ed.). Cambridge, MA: MIT Press.

Schelling, T. (1978). *Micromotives and macrobehavior*. New York: W.W. Norton.