Traveling Termites

Philipp K. Janert
(Dated: July/August 2006)

An ant performs random walks on a field strewn with wood chips. If it encounters a wood chip, it picks it up, only to drop it off when it finds the next wood chip. We investigate the structure of the developing aggregates and study their time evolution. We introduce correlation function and correlation length as useful concepts to describe disordered structures.

TERMITES ON A PLANE

Imagine an ant or termite in a forest. It runs about, picking up wood chips and assembling them into piles and hills. There is a very simple model system, capturing some of the ant's behavior.

In our model system, we consider an area with a certain number of wood chips randomly distributed in it. The "ant" performs a random walk in this area (or "field"): at each time step, it moves one step in a random direction: up, down, left, or right. If it encounters a wood chip, it picks it up and starts carrying it. If it bumps into another wood chip *now*, it drops the chip it is carrying and continues to scurry along — until it finds the next chip it can pick up and carry. Repeat!

Those are the rules. They are very simple. What behavior will they bring about?

It is rather straightforward to write a computer program to study this model — the rules are simple enough! Figures 1-5 show some examples from runs of the simulation. Starting out with a uniform distribution of chips, we find that over time they start to accumulate into larger and larger aggregates. At very long times, however, it seems the system reaches a "steady state" — the piles don't seem to get bigger anymore. Or do they?

This is an old problem (cf. the section on Further Reading for references), but it hasn't lost any of its fascination. The accumulation effect is obvious and quite striking, but it is surprisingly hard to make firm, quantitative statements about this model.

Here are some questions one may want to ask: What is the final state of this model? Will all chips end up on a single heap or not? If not, is there a "natural" size for the heaps? What determines this size (or this distribution of sizes)? How does it depend on the number of wood chips? Does it depend on the size of the field(!) as well?

Furthermore: how long will it take to reach the final state? How quickly do the heaps grow? Does speed of their growth increase or decrease over time?

Finally: how does any of this depend on the "density" of wood chips? Is it possible that if there are too many of them, the entire system will "freeze", with the ant being effectively trapped into a small area of the field?

A computer model can show us the behavior of the system. It will be up to us later to *interpret* these results and to find answers to some of the questions above!

DETAILS OF THE MODEL

For simplicity, we assume a square field of N-by-N cells. Each cell may either contain a single wood chip or be empty. Let the number of wood chips be m.

The ant hops from cell to cell in a random fashion. At each time step, one of the four neighbouring cells is chosen at random. What happens then, depends on three things: on the state of the neighbouring cell (full or empty), on the state of the current cell (full or empty), and on the state of the ant (full or empty).

If the neighbouring cell is empty, the ant moves to that cell under all circumstances (if it is carrying a wood chip, it continues to do so). If the neighbouring cell is full and the ant is carrying a chip, the ant drops its chip, so that the current cell now becomes occupied, while the ant itself becomes empty. (The ant does not move in this case.) If the neighbouring cell is occupied, and the ant itself is *not* carrying a chip, *and* the current cell is empty, the ant picks up the chip from the neighbouring cell and starts carrying it. (The ant itself does not move.) If the current cell already contains a chip, the ant does

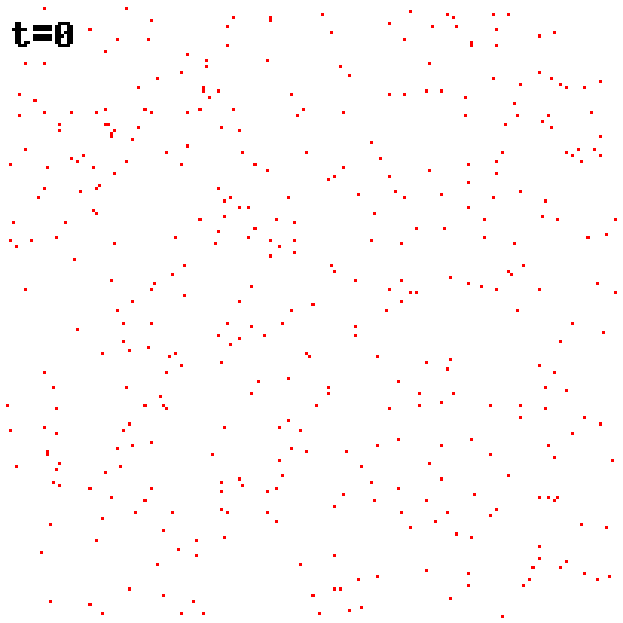

FIG. 1: Initial configuration for a system of 400 chips on a 200-by-200 field.

nothing. (These rules imply that there is never more than a single chip per cell, whether carried by the ant or laying around.)

The table summarizes these rules (a table entry "0" stands for "empty", "1" stands for "full"):

The purpose of these specific rules is to prevent some undesirable behavior. In particular, the requirement that the ant pulls a chip from the neighbouring cell onto its current location (rather than stepping onto the neighbouring cell when picking up a chip) prevents the ant from "burrowing" underneath a pile. I found it important to formulate the rules in such a way that an ant, surrounded by chips on all four sides is, indeed, trapped and cannot move. I wanted to ensure that the ant can only escape such an enclosure by actually moving chips from one cell to another. If we allowed the ant to move onto an occupied cell while picking up a chip, it could escape any enclosure without actually destroying the barrier — not a realistic feature!

Finally, I choose to implement "periodic boundary conditions", where the top of the field is connected to the bottom and the left edge to the right edge. In this way, the ant can walk forever without ever encountering a true boundary. This removes the special case that cells along the boundary have fewer neighbors at the cost of requiring greater care when calculating the positions and distances for cells in the field.

IMPLEMENTATION NOTES

Because the simulation requires many time steps and the structures evolve quite slowly, the core of the program is written in plain C for speed. Also, there is no real-time graphical output - all images are generated after the fact from data produced by the C-program. With these limitations, the simulations run quite fast — on my somewhat dated computer, a simulation on a 200-by-200 grid, with 4000-8000 wood chips, performs 10^8 simulation steps within a few seconds.

The heart of the simulation is a one-dimensional array of type `long` representing the field. The 2-dimensional structure and the periodic boundary conditions are only established when transforming the index `i` of a cell in this array into the corresponding x- and y-coordinates.

For a 2-dimensional field of N-by-N cells, the array contains N^2 elements. The following statements transform between the array index `i` and the (x,y)-coordinates:

```
x = i%n; /* mod-operator */
y = i/n; /* Truncating integer division! */

i = y*n + x;
```

Some macros are provided to help with the bookkeeping required by the periodic boundary condition. They use on the modulus-operator `%`. The C Standard does not fully prescribe its behavior for negative arguments and therefore care has been taken in the macros to ensure that all arguments of `%` are always positive, if necessary by adding `n` to the arguments.

The actual simulation loop is performed within the `main()` routine. Besides this, there is an initialization routine and several output routines. These comprise the core of the simulation. Besides those, the program contains a "monster": the `corr()` and `eqn()` routines. The former calculates the "correlation function", which we will discuss below. Besides calculating the correlation function, the `corr()` routine also fits a curve to the correlation function and determines a parameter known as "correlation length". This is done using the Newton-Raphson root-finding method. The `eqn()` routine is the equation whose root is being sought. Since all of this requires a mere half-dozen lines of code, the entire

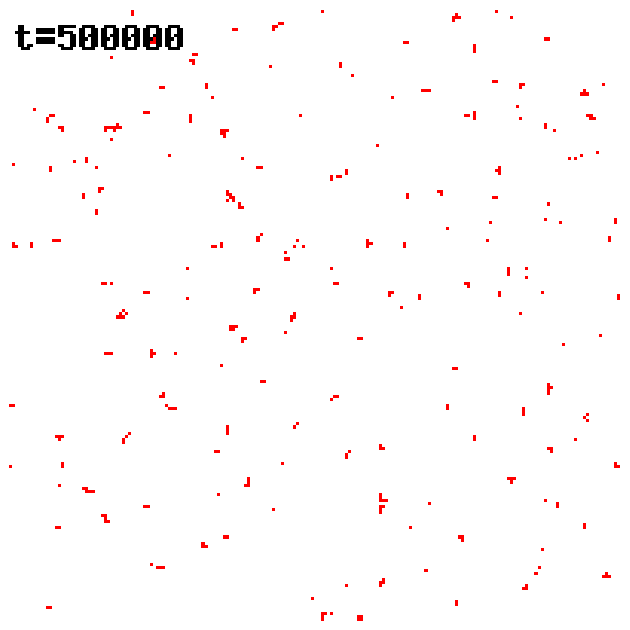

FIG. 2: Same system, after 500,000 time steps.

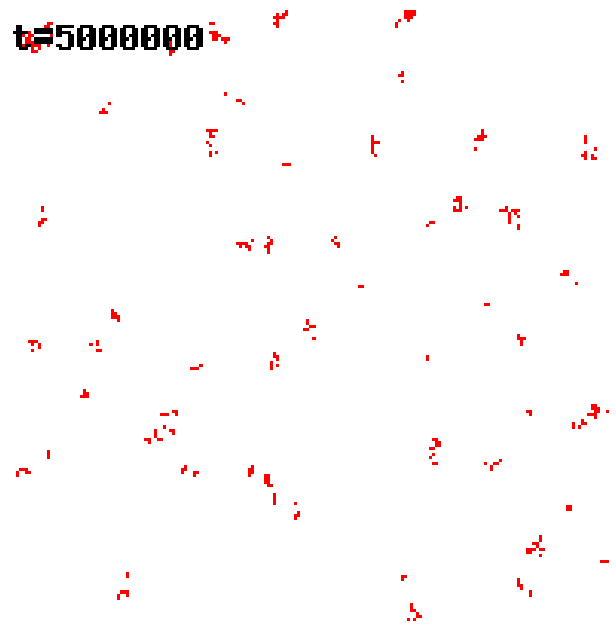

FIG. 3: Typical configuration after 5,000,000 time steps.

TABLE I: Summary of updating rules

Current Time step			Next Time step			
Termite	Current cell	Neighboring cell	Termite	Current cell	Neighboring cell	Move to new cell
0	0	0	0	0	0	yes
1	0	0	1	0	0	yes
0	1	0	0	1	0	yes
1	1	0	n/a (never two chips in one cell)			
0	0	1	1	0	0	no
1	0	1	0	1	1	no
0	1	1	0	1	1	no
1	1	1	n/a (never two chips in one cell)			

Newton-Raphson iteration is simply embedded in `corr()`. *Both `corr()` and `eqn()` are analysis routines — the actual simulation does not depend on them. If they cause confusion, they can be eliminated entirely!*

The main data structure does not just contain information whether a cell is occupied or empty: instead, it stores the time step at which each chip was placed in its current position. This allows to evaluate how often chips move, or whether they mostly stay in place.

The graphs are generated using Perl and GD from data produced by the C program. Since the C program prints results from different time steps to the same file, the Perl graphing script first has to locate the desired time step in the data file (this is done in the first loop), then read all the data for the required time step (this is done in the second loop), and finally create the image.

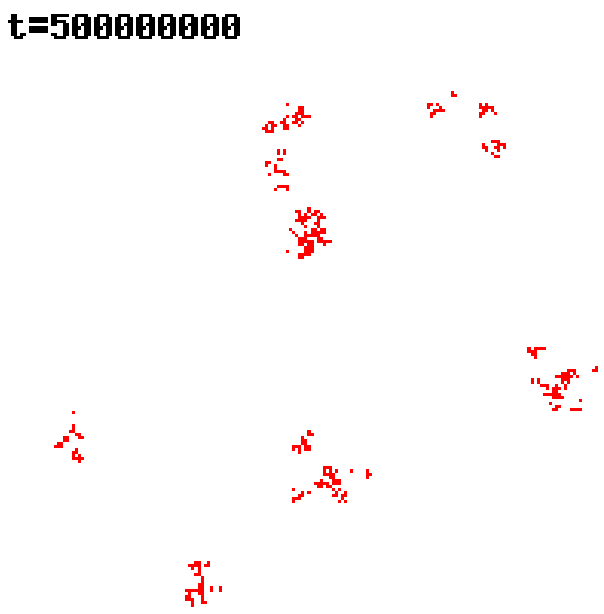

FIG. 4: Long-time behavior after 500,000,000 time steps.

CORRELATIONS, CORRELATION FUNCTION, AND CORRELATION LENGTH

Figures 1-5 show typical results from simulations on a 200-by-200 grid, with 400 wood chips (i.e. a density of $\frac{400}{200^2} = \frac{400}{40000} = 0.01 = 1\%$). Initially, the chips are distributed quite uniformly, but after a few hundred thousand time steps, chips begin to accumulate. The aggregates continue to grow as time passes. The last two figures show late-stage results from a single simulation run. Notice how the size and shape of the heaps does not seem to change, although heaps float slowly across the simulation field. (In fact, every single chip moved several times between Fig. 4 and 5.)

How can we describe the structure of these aggregates in a more quantitative way? One way to do so is to calculate the so-called "correlation function". Given a chip at position (x, y), the correlation function measures the probability to find another chip at a distance r from the first.

FIG. 5: Same system as previous figure, after 900,000,000 time steps.

Given the positions of all chips on the field, the correlation function can be calculated as follows. At the heart of the calculation is a double loop over all *pairs* of chips. For each pair, we calculate the distance between the chips and count how many chips are that far apart. To be precise, we count how many chips fall into the range $0 \le r < 1$, how many fall into the range $1 \le r < 2$, etc. The distance between two points (x_0, y_0) and (x_1, y_1) is given by the well-known Pythagorean formula $r = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$. However, in our case, we have to be aware of the periodic boundary conditions: two particles, one close to the left edge and one close to the right edge are *not* almost N cells apart, but only a few, wrapping around the edge! In fact, a little thought shows that all differences such as $(x_1 - x_0)$ or $(y_1 - y_0)$ must be less than $N/2$!

Finally, we have to normalize the correlation function. First of, we need to divide by the number of "test" particles (i.e. the particles at the "center" from which the distance r is measured) — this is just the overall number of wood chips: m. Then we have to take into account the area of a ring at distance r. It is easy to see, for instance, that a ring of radius $r = 10$ and width 1 has a much greater area (and therefore, we will expect to find many more particles in this ring), than a ring at radius $r = 2$ with the same width. To account for this, we have to divide through by the area. From basic geometry, the area of a circular ring at r with width δr is $\pi(r + \delta r)^2 - \pi r^2 = \pi(2r\delta r + \delta r^2)$. In our case, $\delta r = 1$, so that we must divide by $\pi(2r + 1)$.

Once normalized, the correlation function gives the probability of finding another chip at a distance r. For large r, it approaches the overall density of chips (which is just $\frac{m}{N^2}$) — a useful consistency check!

Figure 6 shows the correlation functions at different times. At zero distance ($r = 0$), the correlation function measures the so-called "self-correlation": the distance of one particle to itself! Initially (for $t = 0$), the correlation function drops immediately to 0.01 (the overall density of chips!), in other words, besides the self-correlation, chips are not correlated with each other at all. However, as t increases, we observe that the correlation function decays ever more slowly, indicating the build-up of correlations over longer and longer length scales. (The acute observer will notice that the correlation function "undershoots" the uniform density value right after the initial decay. This is known as the "correlation hole" and indicates that the space *between* the piles is empty!)

If we could measure how "slowly" the correlation function decays, then we could use this information to describe the amount of structure in the system. Here is a way to accomplish this. (The following section uses some advanced mathematical and numerical concepts. Feel free to skip.)

Let's subtract the uniform density from the correlation function and divide through by its value at $r = 0$. Now it will be equal to 1 at $r = 0$ and decay to zero for large r. Correlation functions in this form (conventionally denoted $g(r)$) are often well approximated by the exponential function: $g(r) \approx e^{-r/\xi}$. This expression defines a quantity ξ, called the "correlation length". It can be interpreted as the length scale, over which particles are correlated (in other words, it is a measure for the size of the piles in our simulation). By fitting our data to this functional form, we can obtain a value for the correlation length. Here, we require ξ to be chosen such that the "mean-squared error" $\chi^2 = \sum_{\text{all points i}} \left[e^{-r_i/\xi} - g(r_i)\right]^2$ is minimized. This is achieved by finding the root of the derivative $\frac{d\chi^2}{d\xi}$ with respect to ξ. The minimization is performed within the code sample using Newton's method, but other numerical root-finding algorithms would work as well.

In Figure 7 we show the correlation lengths as function of time for several different densities. Clearly, the corre-

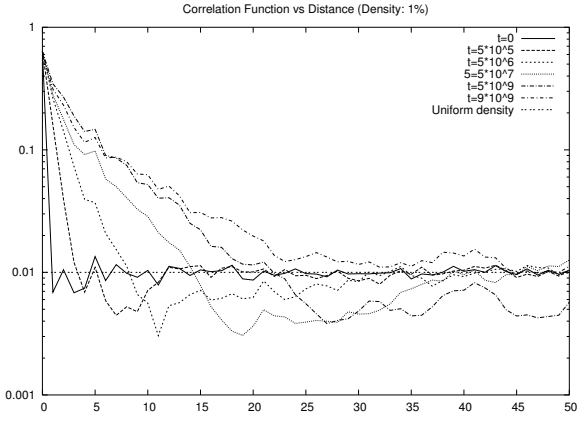

FIG. 6: Correlations function as function of distance for various time steps. Overall density: 1%. Note the logarithmic scale of the y-axis.

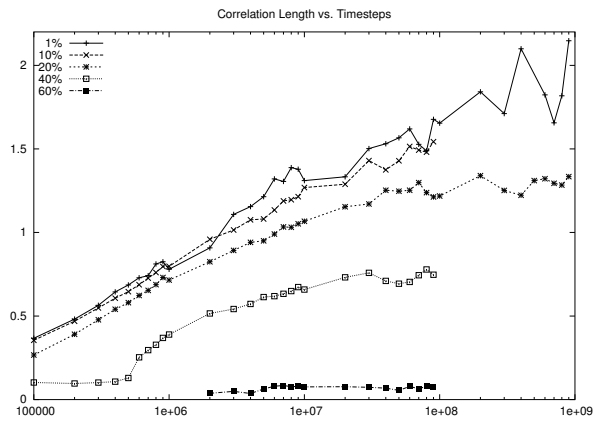

FIG. 7: Correlation length as function of the number of time steps for various densities. Note the logarithmic scale of the x-axis.

lation length (and that means, the size of the piles) grows with time. But they grow very, very slowly! (Notice the logarithmic scale of the x-axis!)

Do they continue growing, or do they eventually level off? From the graph we can't quite tell. For low overall density, the correlation lengths still seem to be growing, but for higher density (20% and 40% specifically), it sure looks as if we have reached a final state, where nothing is happening any more. For even higher density (60%), the system seems to be trapped - no correlations have developed at all: apparently, the ant does not find enough space to move chips around, the system appears grid-locked.

QUESTIONS AND ALTERNATIVES

This is a fascinating system and we have barely started to scratch the surface of its behavior. Here are some further questions for study and contemplation:

- Is there a unique final state, which is reached (with probability 1) in the infinite time limit? If so, what does it look like? How does it depend on N and m? In particular, do things change as we make the size of the field (N) bigger, while keeping the overall density ($\frac{m}{N^2}$) constant?
- What else can we say about the evolving structures? So far, we introduced a single quantity (the correlation length ξ) to describe them. Are there other ways to describe them? What is the shape of the evolving heaps?
- What can we say about the time evolution? Can we find a formula linking ξ to the number of simulation steps passed? How does this behavior change as we increase the overall density? (We already know that things slow down as the system gets more congested. Can we find out more?)
- How does any of this depend on the size of the field? One would expect things to slow down as N gets larger — simply because the ant has to travel farther now.
- For low density, structures form quite "rapidly", but for higher density, Figure 7 shows us, the system is grid-locked. What is the critical density? Does the system grid-lock suddenly, as we sprinkle more and more wood chips on the field, or does it come to a standstill gradually?
- What is the story with the "kink" in Figure 7 for 40% density at 500000 time steps? This kink is not an artifact — it always occurs at this density around that time. Does it occur for other densities as well?
- How far does the ant travel overall? A well-known result from the theory of random walks states that the total distance from the starting point grows like the square root of the number of steps taken. On the other hand, for the grid-locked system at 60% density, I would expect the ant to move hardly at all. Can we use this as criterion to distinguish a grid-locked system?

FURTHER READING

This model was first introduced by Mitchel Resnick: **Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds** (The MIT Press; Reprint edition, 1997), and later discussed in **The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation** by Gary William Flake (The MIT Press; 2000).

Correlation functions are a standard tool in Statistical Mechanics and Statistical Mechanics Simulation. A very good introduction and reference can be found in **Computer Simulation of Liquids** by M. P. Allen, D. J. Tildesley (Oxford University Press, 1989).

The entire area of time-evolution of disordered systems is still relatively new, even to the point that it is not entirely clear what the right questions are to ask. Check the current research literature!