

---

# Self-organizing Computation and Information Systems: Ant Systems and Algorithms

---

**Daniel R. Kunkle**

Computer Science Dept.  
College of Computing and Information Sciences  
Rochester Institute of Technology  
Rochester, NY, 14623  
drk4633@rit.edu  
<http://www.rit.edu/~drk4633/>

For: Introduction to Artificial Intelligence  
November, 2001

## Abstract

In general, this paper is about self-organizing systems (which are responsible for most, if not all of the complex order we see in our universe) and how the science of self-organization can be applied to computation and information systems. In specific, this paper will examine ant systems and algorithms as an example of a self-organizing computation system. Further, the ant algorithm will be applied to a classic optimization algorithm benchmark, the Traveling Salesman Problem (TSP).

computation that could self-organize based on its environment (our computational needs) would be a powerful tool.

To reach such a goal, we must take many steps, a very few of which will be examined here. First, the science of self-organization is examined in its own right. Next, information systems, both as a natural phenomenon and an artificial one, are examined in light of self-organization theory. Then, as a concrete example, the social organization of ants as a problem-solving construct is studied. Specifically, an implementation of an ant algorithm to solve the Traveling Salesman Problem will put forth.

## 1 INTRODUCTION

Nature, it seems, has an innate ability for computation. Physical systems find complex states, as is portrayed by intricate crystal formation. Seemingly despite the second law of thermodynamics, life arises, systems act on their own behalf, apparently going beyond the simple laws of physics. Living systems work to survive in the world, adapting to complex, varying situations. Life evolves into intelligent life, not only adapting to situations, but intentionally molding their environment as well.

All of this computational power comes not from explicit manipulation of a programmer, but from the very nature of the universe. Simple, unordered systems have the ability to self-organize into complex structures, able to maintain themselves in the face of ever-changing environments. Some recent scientists, such as Stuart Kauffman (1993), believe that systems not only have the ability to self-organize from disordered environments, but that it is inevitable. A complex, ordered universe is not surprising, but expected by new theories of science.

Now that a science of self-organization has been well established in regards to the physical universe (Heylighen, 1996) it can be applied to other disciplines as well, including computer science. A fundamental universe of

## 2 SELF-ORGANIZATION

Self-organization has definitions of varying specificity. Three simple interpretations come from the FAQ for the newsgroup comp.theory.self-org-sys (Lucas, 2001).

1. The evolution of a system into an organized form in the absence of external constraints.
2. A move from a large region of state space to a persistent smaller one, under the control of the system itself. This smaller region of state space is called an attractor.
3. The introduction of correlations (pattern) over time or space for previously independent variables operating under local rules.

These definitions are equivalent, the first being the most intuitive. Simply, a system self-organizes if it requires no external force to organize it.

### 2.1 Conditions for Self-organization

Obviously, not all systems will self-organize. A system must meet a number of conditions, or constraints, to be able to move from a disordered state to an ordered one.

These constraints can be any number of things, and depend upon the system being studied.

It could be the number of elements in the system. For example, convection cycles of heat transfer, such as Bénard rolls (Heylighen, 1996), need to have some minimum number of molecules (near  $10^{32}$ ) to achieve spontaneous organization.

The convection cells also need a sufficient heat source. In fact, all systems that move to an organized state from a disorganized state need to make use of some free energy, as is specified by the laws of thermodynamics.

As more complex systems are studied the constraints become more numerous. For example, most biological life on earth requires the right levels of certain gases, such as oxygen, and the right temperatures.

The creation of problem solving and computation systems can be seen as the act of constructing a system that has no inherent or explicit organization, but one that will self-organize when given the right conditions. This may be more difficult than constructing the organized system from the start, but self-organizing systems have characteristics that are beneficial to many computational environments, as will be explored in the next section.

## 2.2 Characteristics of Self-organizing Systems

The characteristics of self-organizing systems have much to do with modern computational systems. Heylighen (1996) lists seven characteristics, the first few being fundamental to all self-organizing systems, and the later few applying mainly to more complex systems.

### 2.2.1 *Global order from local interactions*

Organized systems are those that have global order, or correlation between the elements of the system. By definition, a self-organized system is not controlled by some external force. So, the interactions of the elements on a local level are those responsible for the organization.

### 2.2.2 *Distributed Control*

This is a corollary to the first characteristic. Because the local interactions are responsible for global order, the control of the organized system is also distributed among the elements.

### 2.2.3 *Robustness and Resilience*

The distributed control of the system contributes to its robustness and resilience. That is, the system is resistant to perturbations and has a strong capability to restore itself after damage.

### 2.2.4 *Non-linearity and Feedback*

Many complex self-organizing systems are complex because of their non-linear nature. That is, even if the elements themselves are very simple their interaction and relationships can become complex. Small changes can have large effects (through positive feedback), and conversely, large changes can have relatively small affect (through negative feedback).

### 2.2.5 *Organizational Closure, Hierarchy and Emergence*

Like many artificially constructed systems, self-organizing systems form a hierarchy of levels. Each of these levels can be seen as having a boundary, an inside and outside, and is therefore organizationally closed. Very often, the higher levels can not be described or understood by examining the properties of the lower levels and are said to be emergent.

### 2.2.6 *Bifurcations and Symmetry Breaking*

Very often a self-organizing system will have more than one stable organized state, or attractor. While moving from the initial, disorganized state, the system must make some "decision" as to which organized state it will seek (these decisions typically being made by random differences or perturbations in the initial state). These decision points are called "bifurcations" and are a form of symmetry breaking.

### 2.2.7 *Far-from-equilibrium Dynamics*

For a sufficiently complex system to maintain its organization it must be constantly presented with a source of free energy and be able to make use of that energy. Consequently, the system can not be at equilibrium with its environment. It has a constant flow of resources moving through it to rebuild the organization it has constructed and to build new organization in an effort to adapt to new environmental conditions.

## 3 EXAMPLES OF SELF-ORGANIZING COMPUTER SYSTEMS

### 3.1 Quantum Life Forms

Imagine a large torus, initially covered at random by thousands of small red dots, so that little of the surface can be seen. These dots are simple creatures that follow only one rule. Each of them chooses at random one other dot, and then moves in the opposite direction of that one dot. They move in time steps; the distance they move each step is inversely proportional to their current distance from the one they are focused on eluding.

Each of these thousands of dots moves in parallel. Each only knows of one other dot. There is no overseeing coordinator of the movement of the population of dots.

From the description of the simple rules, one might predict that the collective movement would be simple and chaotic. One dot avoiding another, which in turn moves because of the location of another, and so on. Each dot could run around the torus it sits on forever trying to reach the location farthest from the one other dot they know about.

But, as unlikely as it may seem, complex order does arise from the simple, decentralized actions of each dot. They aggregate into a small number of "Quantum Life Forms" (QLFs) (see figure 1 below).

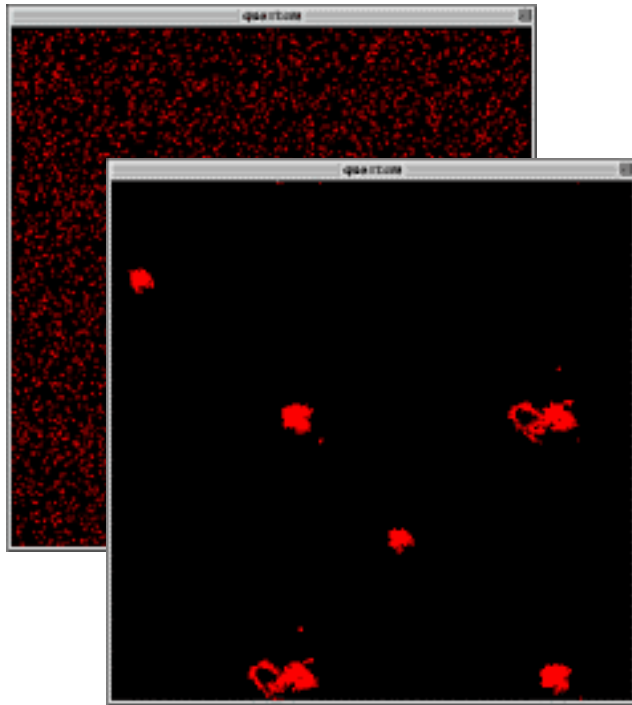


Figure 1: QLF Simulation at time zero with a random distribution of dots (top, back). QLF Simulation after 2000 time steps with a small number of large collectives (bottom, front).

This simulation was programmed using StarLogo, which can be downloaded free from the MIT Media Laboratory StarLogo site (2001). In the words of StarLogo creators, "StarLogo is a programmable modeling environment for exploring the workings of decentralized systems -- systems that are organized without an organizer, coordinated without a coordinator."

### 3.2 Force Directed Graph Layout

This computer system is similar to the Quantum Life Forms in that it has many of the character of self-organizing systems; global order from local interactions, distributed control, non-linearity, emergence, etc. It differs, however, in that it solves a problem, or completes a task.

The task in this case is laying out the nodes of a graph in some logical manner, so that the reader of the graph can understand its structure.

One possible method may be to have an algorithm that examines the global structure of the graph and attempts to minimize edge crossings while keeping highly connected nodes close to each other and separating non-related nodes.

This may need to be a relatively complex algorithm. If the algorithm simply looked at each node one at a time and placed it in the optimal position it would be ruined when the nodes that are related to it are moved later. There

needs to be a solution where all of the nodes positions are optimized in parallel.

The force directed layout method treats the nodes as physical objects and the edges of the graph as springs connecting the nodes. Physical forces are simulated in time steps, which organize the graph as the springs attempt to find their resting length. In a short period of time the graph will find the lowest energy state, which is one in which the springs are closest to their resting length. Related nodes will remain close to each other while non-related nodes will be free to move apart.

This method works in both 2 and 3 dimensional space. A 3D version of this technique was programmed (see figure 2). The graph layout was tested using known graph shapes, including all of the platonic solids. Each of these graphs found their low energy state easily over many test runs. Random graphs were also arranged easily into low energy configurations.

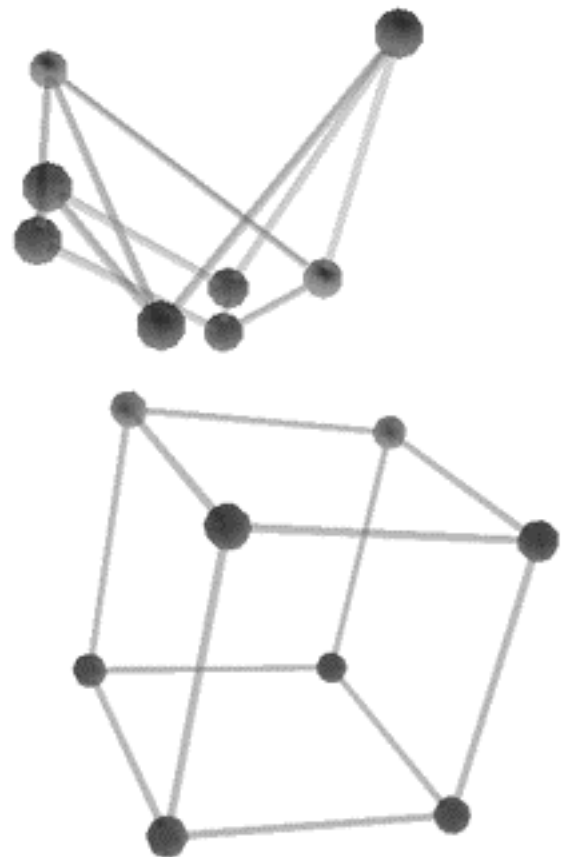


Figure 2: Force directed graph layout at time zero; nodes in random 3D locations (top). Force directed graph layout after simulated physics (bottom).

## 4 ANTS AS A SELF-ORGANIZING INFORMATION SYSTEM

Ants are a biological example of a self-organizing system. They go another step beyond the previous two examples in that they use their environment to their own benefit. By using chemical pheromones, they store information in the world. These chemicals are markers that signal paths between ant nests and food sources. The ants organize a pattern of chemicals in order to organize themselves.

In mathematical terms, the networks constructed by the ants form a minimum spanning tree (Goss et al., 1989). The minimum spanning tree allows the ants to retrieve food with a minimum amount of energy expenditure. Minimum spanning tree can be found through many algorithms, defined by graph theory, but ants don't use these conventional methods. Ants form their globally optimal networks through local interactions with each other and their environment (Parunak, 1997). The ant networks emerge.

## 5 ANTS AS AN OPTIMIZATION ALGORITHM

Ant algorithms are based on the behavior of natural ant systems, but will often have enhancements to allow them to better solve problems.

Ant algorithms can be put in the class of general optimization tools along with genetic algorithms, simulated annealing and tabu search (Dorigo, 1996). The algorithm described by Dorigo et al., "Ant Cycle", was found to be competitive with these generic optimization methods when applied to the traveling salesman problem and is examined in more detail as applied to other problems in Dorigo et al. (1999).

When considered as a method of reinforcement learning, the Ant Cycle algorithm is equivalent to a Monte Carlo method (Nowé, 1999).

### 5.1 The Traveling Salesman Problem

Because ants find minimum spanning tree of graphs, it is easy to see how ant algorithms can be applied to solving problems that can be represented as a graph, where the nodes represent states and the edges represent the cost of moving from one state to another.

One such problem, which is often used for benchmarking optimization methods, is the traveling salesman problem (TSP). Each node in the graph represents a town, the edges are the distances between each town. A path must be found that visits each city once. The goal is to find the shortest such path.

In Section 6 an implementation of the Ant Cycle algorithm to solve the TSP will be outlined.

## 5.2 Network Routing

As has been explored, ant systems are a self-organizing system, a method for optimization and a reinforcement learning method. It has many of the characteristics of self-organizing systems as described earlier. Most importantly, they have distributed control and are robust and resilient.

These features make ants a good choice for dynamic environments where short paths need to be found, such as a communications network.

New systems based on swarm intelligence, including ants, are being developed to provide higher performance and greater reliability on communications networks that are always changing and growing and are increasingly diverse and heterogeneous (Kassabalidis, 2001).

Kassabalidis et al. found seven areas in which swarm intelligence techniques improved over traditional network routing techniques: scalability, fault tolerance, adaptation, speed, modularity, autonomy and parallelism.

## 6 ANTS IMPLEMENTED

The Ant Cycle algorithm described by Dorigo et al. (1996) has been implemented as an interactive application for solving the TSP and is available at <http://www.rit.edu/~drk4633/ants.html>

This application will allow the user to enter a number of towns (up to 40) and execute the Ant Cycle algorithm to search for the shortest tour of all towns.

### 6.1 Using the System

When first started the map (white area) is cleared, with no towns present. Four statistics are presented at the top of the display. They are, the number of cities, the shortest tour found so far, the number of time steps and the number of cycles. These will be updated as the algorithm runs.

To place a town, click in the map area. You can repeat the placing of towns up to 40 times. A town can be removed by clicking on it again. All towns can be cleared with the "Clear All" button in the bottom left.

Once a set of towns has been constructed the algorithm can be run by clicking "Run Ants". If there are 16 or fewer towns every edge will be drawn (one between each pair of towns). The algorithm will be run for 10 cycles for each time the "Run Ants" button is clicked. At the end of each cycle the best tour so far will be updated and the edges of the map will be redrawn to correspond to the best tour found. The "Run Ants" button can be clicked as many times as desired. After a few runs the best tour will change more rarely, as the best tour so far approaches or equals the best possible tour.

Once the map is edited the algorithm will be reset to work on the new set of towns.

## 6.2 System Structure

The system is implemented using Macromedia Director 8. The programming language for this environment is Lingo.

The algorithm is the same as that described by Dorigo et al. (1996).

When the number of ants used is about equal to the number of towns (as was found to be optimal) the complexity of this algorithm is  $O(NC \cdot n^3)$ .

The basic method of the ant algorithm is as follows:

- 1 Each town starts with one ant on it. All edges are initialized to have a trail intensity of 1. Trail intensity is a measure of how "fit" the edge has been found to be by the ants that have used that edge. Higher trail intensities correspond to better edges.
- 2 Each ant makes a complete tour in parallel with each other. Each ant will choose the next edge to use (and therefore the next town to visit) based on the length of the edge (the cost of using that edge) and the trail intensity present on that edge. The ant will choose the next edge probabilistically, and can therefore choose longer edges over shorter ones sometimes, but will usually choose the better edge. This probabilistic choosing allows some ants to choose a longer edge earlier in the tour so that they might be able to find a number of shorter edges later on to more than compensate for that choice. Each town an ant visits is placed in its "tabu list", disallowing the ant the chance to revisit that town on the same tour.
- 3 Once the tour is completed, each ant will add some amount of trail intensity to each edge they used during their last tour. The amount of trail increase is inversely proportional to the total length of their tour. Hence, ants finding shorter paths will lay more trail (because they found a better set of edges). If any of the ants have found the shortest path so far, that path is saved as the current solution.
- 4 This completes a cycle. If the number of cycles completed so far is not equal to the maximum number of cycles desired, the process continues. The trail on the edges has some rate of evaporation, so that initially strong tours do not dominate later, potentially better tours. The ants' tabu lists are emptied and the process is repeated, starting with step 2.

## 6.3 Results

The Any Cycle algorithm implemented here was found to have results very similar to those found by Dorigo et al. (1996) in their implementation. The method was tested on a number of sets of towns in which the optimal solution is known. Each of these is a grid of evenly spaced towns of

varying degree. Results for town grids ranging from 4x4 to 8x8 from Dorigo et al. (1996) are as follows:

Problem	n (towns)	Best tour	Avg. # cycles to find best tour
4 x 4	16	160	5.6
5 x 5	25	254.1	13.6
6 x 6	36	360	60.0
7 x 7	49	494.1	320.0
8 x 8	64	640	970.0

## 7 REFERENCES

- 1 Dorigo, M., Maniezzo, V., and Colorni, A. (1996): "The ant system: optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics--Part B, vol. 26, No. 2, pp. 29-41.  
<http://citeseer.nj.nec.com/dorigo96ant.html>
- 2 Dorigo M., Di Caro, G. and Gambardella, L.M. (1998): "Ant Algorithms for Discrete Optimization", Technical Report IRIDIA/98-10, Universite Libre de Bruxelles, Belgium. To appear in Artificial Life.  
<http://citeseer.nj.nec.com/article/dorigo99ant.html>
- 3 Heylighen, F. (1996): "The Science of Self-organization and Adaptivity". to be published in the Encyclopedia of Life Support Systems.  
<http://pespmc1.vub.ac.be/Papers/EOLSS-Self-Organiz.pdf>
- 4 Kassabalidis, M.A. El-Sharkawi, R.J. Marks II, P. Arabshahi, and A.A. Gray. (2001): "Swarm intelligence for routing in communication networks," IEEE Globecom 2001, Nov. 25-29, 2001, San Antonio, Texas, (submitted April 2001, in review).
- 5 Kauffman S. A. (1993). The Origins of Order: Self-Organization and Selection in Evolution, Oxford University Press, New York.
- 6 Kugler P.N., Turvey M.T (1989): "Self-Organization, Flow Fields, and Information", in Kugler P.N., Turvey M.T. (eds.), Selforganization in biological work spaces, North-Holland, Amsterdam, pp.1-33.
- 7 Lucas, C., (2001): "Self-Organizing Systems (SOS) F A Q ". Available at <http://www.calresco.org/sos/sosfaq.htm>
- 8 Media Laboratory (2001): "StarLogo on the Web". Available at <http://www.media.mit.edu/starlogo/>
- 9 Parunak, H.V.D. (1997): "Go to the Ant: Engineering Principles from Natural Multi-Agent Systems." Forthcoming in Annals of Operations Research. Available at <http://www.iti.org/~van/gotoant.ps>
- 10 Parunak, H. V. D. and Brueckner, S. (2001): "Entropy and SelfOrganization in Multi-Agent

Systems". In Proceedings of International Conference on Autonomous Agents, pages (forthcoming). <http://citeseer.nj.nec.com/parunak01entropy.html>

- 11 Nowé A., Verbeeck K. (1999): "Formalizing the Ant Algorithms in terms of Reinforcement Learning",

Proceedings of the European Conference on Artificial Life, Lausanne, Switzerland, pages: 616-620. <http://prog2.vub.ac.be/~kaverbee/s091.ps>