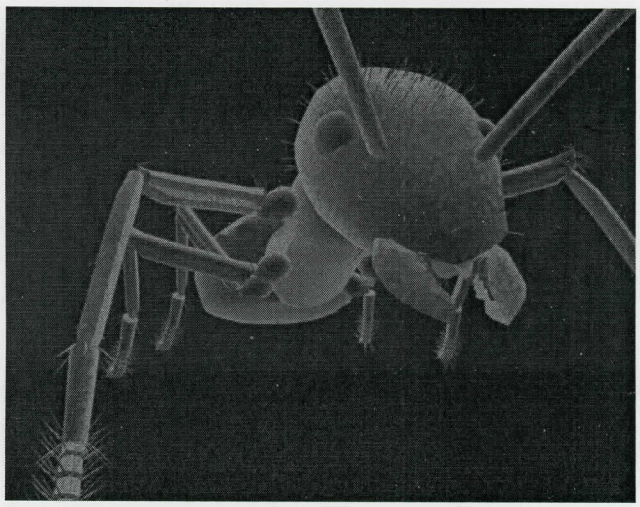
A

nice!

6.836 Embodied Intelligence
Research Assignment 4:
Ant Farm

Gleb Chuvpilo (chuvpilo@mit.edu)
April 12, 2002



Rod was excited about your excitement over this RA ..

Problem 1: Genetic Encoding and Hand-made Ants

The sensory-motor coordination for the ant can be implemented using a finite state automaton (FSA). Design a "genetic encoding" representation, specified by a fixed-length bit string, that encodes ant sensory-motor FSA transition tables. Then, by hand, try to design the best possible ant for the John Muir trail. Show the encoding and the corresponding FSA state-transition table and diagram. What fitness does your ant controller score? Limit yourself to, at most, a 16-state FSA.

The "genetic encoding" representation needs to satify two goals. First, it has to be **scalable**, or able to support ant definitions with various numbers of states in their FSAs. Second, it needs to be **fast** for use in the simulation program, or, in other words, it should be easy to extract the necessary information from the encoding. For the aforementioned reasons we decided to use the genetic encoding shown in Figure 1, representing a general 16-state FSA.

As you can see, the encoding is an array of 17 elements. The first 16 of them represent the binary-coded State Transition Table for the 16 states of the FSA by encoding the state to which the ant's control should go if there is food (`state_one`)/there is no food (`state_zero`) in front of the creature, as well as the action taken upon this transition for both of these cases (`act_one`, `act_zero`): do nothing, turn right, turn right, and move forward.

Each of the elements is a `UInt16` (`unsigned short int` in the C language) shown in Figure 2. The lower-order two bits [0..1] are used to hold the action taken if there is food in front; the bits [2..5] are used for the transition if the ant can see the food; the bits [6..7] and [8..11] represent the same fields, but in case there is no food in front. The higher-order four bits [12..15] are unused for the case of a 16-state FSA.

It is straightforward to see why this encoding satisfies the stated goals. It is scalable up to 64-state FSAs, since the four unused bits can be split into two and two, each concatenated with the `state_zero` and `state_one` fields (which gives $16 \times 4 = 64$ states). It is also a fast encoding, since bit extraction from each of these fields takes at most two assembly operations (shift right and mask using AND).

Now we will address the second half of the question – designing a the best possible ant for the John Muir trail. Before doing that, we will take a brief look at the anatomy of an ant. The first thing you will notice in the diagram in Figure 3 is the three main divisions: the **head**, the **trunk**, and the **metasoma**. Ant bodies, like other insects have an **exoskeleton**. Their skeleton is on the outside - not covered by skin, muscles, and tissue like humans.

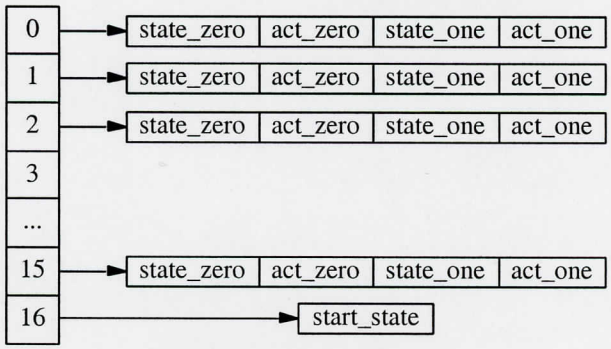


Figure 1. Genetic encoding.

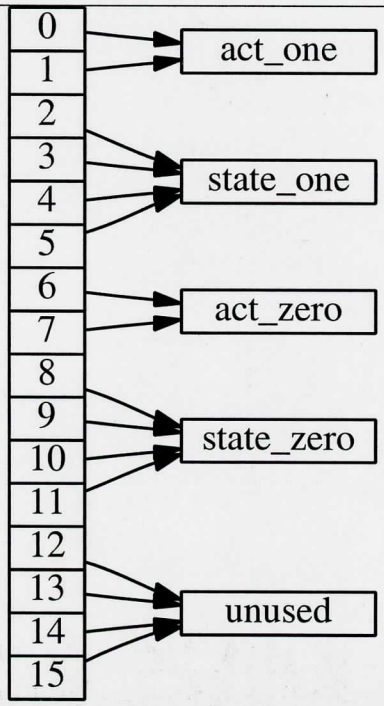


Figure 2. Packing fields into UInt16.

The **head** of an ant has several imporatant parts. First the eyes, which are a lot like the facet eyes of a fly. This structure enables ants to see movement very well. It is important to notice that ants can't hear. Also attached to the head of the ant are two **feelers**. The feelers are special organs of smell, as well as tactile organs. Ants release pheromones to communicate with each other, and the feelers receive these smells as three-dimensional signals (ants can smell the **3D direction vector** to an object!). The head also has two strong pinchers which are used to carry food, to dig, and to defend. Just inside the mouth is a small pocket where ants can store food. They can share this food with other ants in need, which biologists call *tropholaxis*. The only other case of animals exhibiting tropholaxis is observed in wolves (*Canis lupus*).

The **trunk** of the ant is where all six legs are attached. At the end of each leg there is a sharp claw that helps ants climb and hang onto things. The **metasoma** of the ant is a poison sack. Ants are closely related to wasps and other stinging insects. Many types of ants have stingers and can inflict a very painful sting. This is a useful way to defend against the many predators ants have.

Inside ants do not have lungs. Oxygen enters through tiny holes all over the body, and Carbon Dioxide leaves through the same holes. There are no blood vessels. The **heart** is a long tube that pumps colorless blood from the head back to the rear and then back up to the head again. The **nervous system** of ants consists of a long nerve cord that also runs from head to rear with branches leading to the parts of the body, like a human spinal cord.

Now that we know that ants can smell in 3D, it becomes clear why they can easily find the direction to the source of smell (trail or food, for example). When designing our hand-made ant, we would like to **imitate** the nature by somehow allowing our ant smell in 3D. By doing so we are guaranteed to obtain behavior as robust as the one of real ants. Unfortunately, we are given an ant which can only smell the food in front of him. How can we imitate the nature given just that? The answer is simple and logical: we need to "program" our ant to turn around full 360 degrees in four

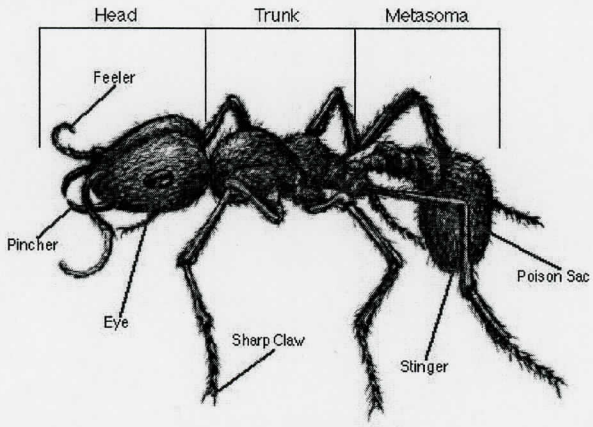


Figure 3. Ant anatomy.

Old State	Input	New State	Action
00	0	01	1
00	1	00	3
01	0	02	1
01	1	00	3
02	0	03	1
02	1	00	3
03	0	04	1
03	1	00	3
04	0	00	3
04	1	00	3

Table 1. Hand-designed FSA state-transition table. Start state: [00]. Actions: "0"=no-op, "1"=right, "2"=left, "3"=forward.

steps if it sees no food in front of him, and keep going in case no food (trail) was found around the cell it was standing on. Figure 4 shows an FSA which provides exactly this kind of behavior. The start state is state 0, and the other four states signify the four 90-degree turns of the ant and a step forward from state 4, if no food was found. Figure 5 and Table 1 show the encoding of this ant and the FSA state-transition table respectively (notice that this ant only uses five states out of possible sixteen).

This approach has both advantages and disadvantages. On the one hand, the most serious disadvantage is the inefficiency of this "algorithm" for a particular trail. On the other hand a "specialized" ant can do better on specific trail, while as it will most likely fail miserably on an unknown trail. Our approach is thus **generality**, which leads to good results on any types of trails. To see that, let's take a look at how our ant traverses the John Muir trail.

Figures 6, 7, and 8 represent the John Muir Trail and two walks of the hand-made ant, the first one given enough time to finish (full score of 89 points), and the second one limited to 200 time

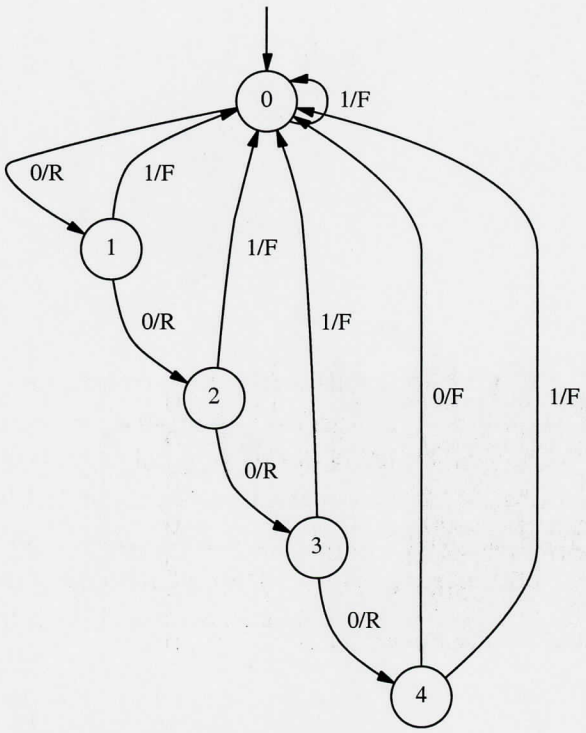


Figure 4. Hand-designed FSA.

143	243	343	443	0c3	000	000	000	000	000	000	000	000	000	000	000	0

Figure 5. Hand-designed genome (hexadecimal representation).

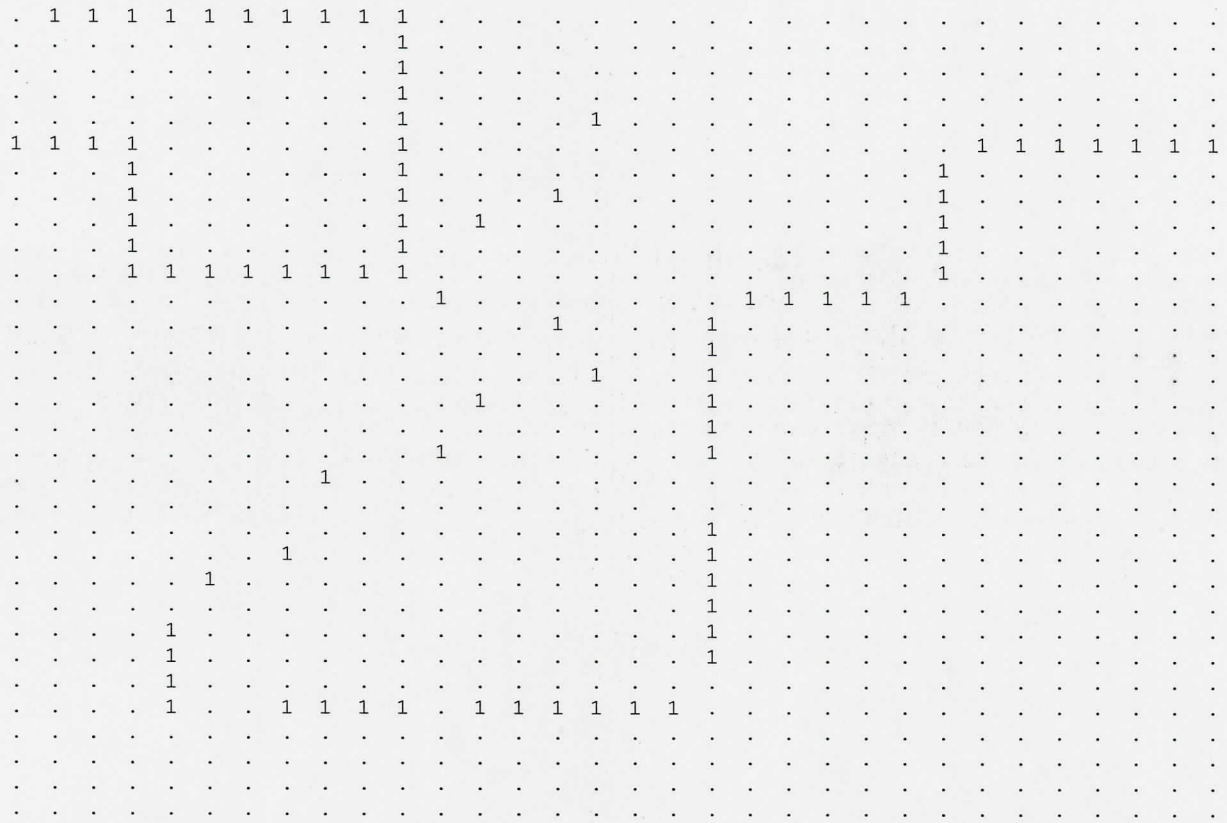


Figure 6. The John Muir Trail ("1"=food, "."=no food). The ant starts out at the topmost left cell facing East.

```
00 00 00 00 00 00 00 00 00 00 01 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  00 .  .  .  01 00 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
00 00 00 03 .  .  .  .  .  .  00 .  .  .  04 .  .  .  .  .  .  .  .  .  03 04 00 00 00 00 00 00
.  .  .  00 .  .  .  .  .  .  00 .  .  .  04 .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .
.  .  .  00 .  .  .  .  .  .  00 .  .  .  04 .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .
.  .  .  00 .  .  .  .  .  .  00 01 04 04 03 .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .
.  .  .  00 .  .  .  .  .  .  00 04 .  .  .  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .
.  .  .  01 00 00 00 00 00 00 01 04 .  .  .  .  .  .  .  .  .  .  .  .  04 .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  04 .  .  .  .  .  .  03 04 00 00 00 00 01 .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  01 04 04 04 03 .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  04 .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  04 .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  01 04 04 04 03 .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  04 .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  04 .  .  .  .  .  .  04 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  01 04 04 04 03 .  .  .  .  .  .  04 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  04 .  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  04 .  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  04 .  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  01 04 04 03 .  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  04 .  .  .  .  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  04 .  .  .  .  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .  04 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  00 .  .  .  .  .  .  .  .  .  .  .  .  .  04 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  01 00 04 04 00 00 00 00 04 00 00 00 00 00 01 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Food gathered: 89
```

Figure 7. Hand-made ant walks the John Muir Trail (values represent the states of its FSA) and gets all 89 food points in 314 steps.

```
00 00 00 00 00 00 00 00 00 00 01 . . . . . .
. . . . . . . . . . . 00 . . . . . .
. . . . . . . . . . . 00 . . . . . .
. . . . . . . . . . . 00 . . . . . .
. . . . . . . . . . . 00 . . . . . .
00 00 00 03 . . . . . 00 . . . . . 03 04 00 00 00 00 00 00
. . . 00 . . . . . 00 . . . . . 00 . . . . . .
. . . 00 . . . . . 00 . . . . . 00 . . . . . .
. . . 00 . . . . . 00 . . . . . 00 . . . . . .
. . . 00 . . . . . 00 . . . . . 00 . . . . . .
. . . 01 00 00 00 00 00 00 01 . . . . . 04 . . . . . .
. . . . . 03 04 00 00 00 00 01 . . . . . .
. . . . . 00 . . . . . .
. . . . . 00 . . . . . .
. . . . . 00 . . . . . .
. . . . . 00 . . . . . .
. . . . . 04 . . . . . .
. . . . . 01 04 04 03 . . . . . 04 . . . . . .
. . . . . 04 . . . . . 00 . . . . . .
. . . . . 04 . . . . . 00 . . . . . .
. . . . . 04 . . . . . 00 . . . . . .
. . . . 01 04 04 03 . . . . . 00 . . . . . .
. . . . 04 . . . . . 00 . . . . . .
. . . . 04 . . . . . 00 . . . . . .
. . . . 00 . . . . . 04 . . . . . .
. . . . 00 . . . . . 04 . . . . . .
. . . . 01 00 04 04 00 00 00 00 04 00 00 00 00 00 01 . . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
Food gathered: 81
```

Figure 8. When given only 200 time steps, the ant scores 81 out of 89.

steps[1]. Thus, we see that the fitness of our hand-made ant is 81 out of 89. This fitness is lower than that of a specialized ant for this particular trail, but in our simulation we wanted to create nature-like behavior (see the argument above), and our goal, again, is a **universal** ant, which would be able to act reasonably well under any conditions, such as other trails (see following sections).

10/10

[1]Having finished the simulation, we read the Jefferson paper and noticed that we came up with the same solution of the hand-made ant. This is logical, since this is the solution that the nature gave millions of years ago. Also, it is interesting to notice that changing "turn right" behavior to "turn left" behavior results in longer time (322 steps) to get all the food.

Problem 2: Using "no-ops"

Does your ant use the `N` *operation? Under what circumstances might the* `N` *operation be useful?*

The hand-made ant given in Problem 1 does not use no-ops. Generally, this action could be useful in order to let an ant change its internal state without changing its orientation in the environment or moving away from its current position. However, it is unlikely that the evolution would be using this action, because taking one step out of possible 200 for "changing mind" is too expensive (0.5% of the total time), and thus the animals able to do well without using no-ops are more likely to procreate.

? useful in perhaps a dynamically changing environ...

3/3

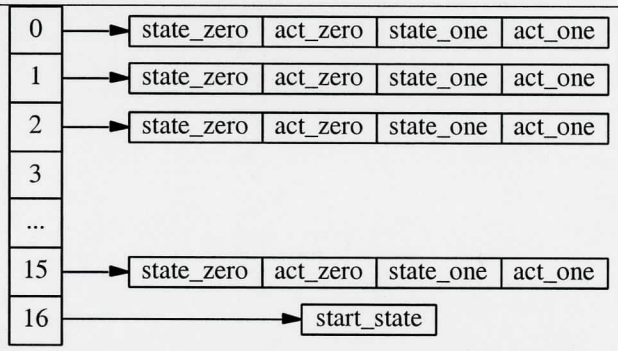


Figure 9. Genetic encoding (repeated for convenience).

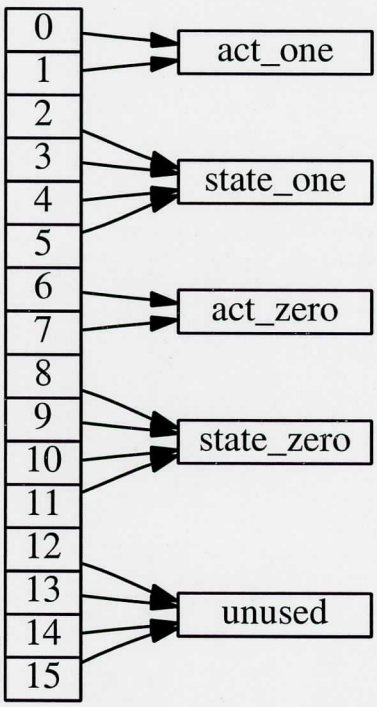


Figure 10. Packing fields into UInt16 (repeated for convenience).

Problem 3: Counting Individuals

How many different individuals are possible in your representation?

As you can see in Figures 9 and 10, which are exactly the same as Figures 1 and 2, the length of the ant's genome is $16 \times 12 + 4 = 196$ bits (each state transition encoding takes up 4 bits, and each action encoding takes up 2 bits). Thus, since each bit can take two values (0 or 1), there are 2^{196} possible different individuals in this representation, which is the number on the order of 10^{59} (this number is roughly equal to the time it would take a black hole a few times the mass of our sun to evaporate; another way to think about this number is that it is a billion times greater than the mass of the Universe).

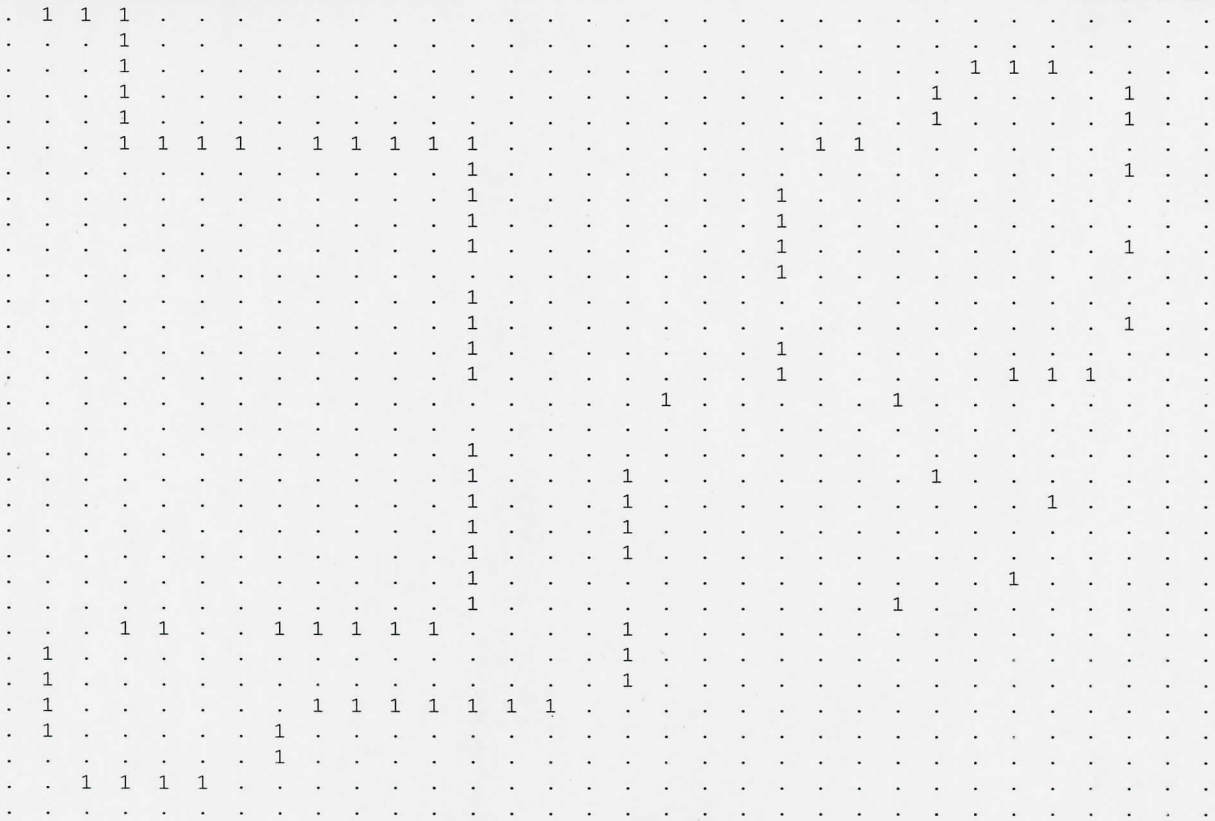


Figure 11. The Santa Fe trail ("1"=food, "."=no food). The ant starts out at the topmost left cell facing East.

Problem 4: Santa Fe Performance

How does your ant fare on the "Santa Fe" trail?

Figure 11 shows the Santa Fe trail, and Figure 12 shows the walk of our ant on that trail. As you can see, the argument presented in Problem 1 claiming that a universal ant is better that the one specialized for a particular trail does work! Indeed, the same ant scores 63 points in 200 time steps, which is a very good result proving the robustness of our ant.

2/3 where did it go wrong?

```
00 00 00 01 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 03 00 00 04 00 00 00 00 00 01 . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 04 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 04 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 04 . . . 01 04 04 01 . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . 04 . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . 04 . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . 04 . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 00 . . . 00 . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 04 . . . 04 . . . . . . . . . . . . . . . . . . . . . . . . . . .
. 03 04 04 00 00 04 04 00 00 00 00 01 . . . 04 . . . . . . . . . . . . . .
. 00 . . . . . 00 . . . . . . . . . . . . . .
. 00 . . . . . 00 . . . . . . . . . . . . . .
. 00 . . . . . 01 00 00 00 00 00 00 04 04 03 . . . . . . . . . . . . . . .
. 04 . . . . . 04 . . . . . . . . . . . . . .
. 04 . . . . . 00 . . . . . . . . . . . . . .
. 03 00 00 00 04 04 03 . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Food gathered: 63
```

Figure 12. Hand-made ant walks the Santa Fe trail (the score is 63 out of 89).

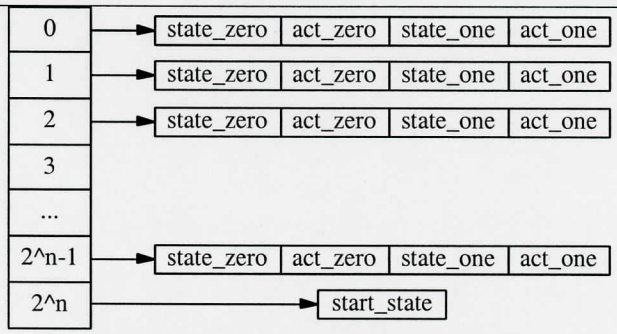


Figure 13. Genetic encoding for n states.

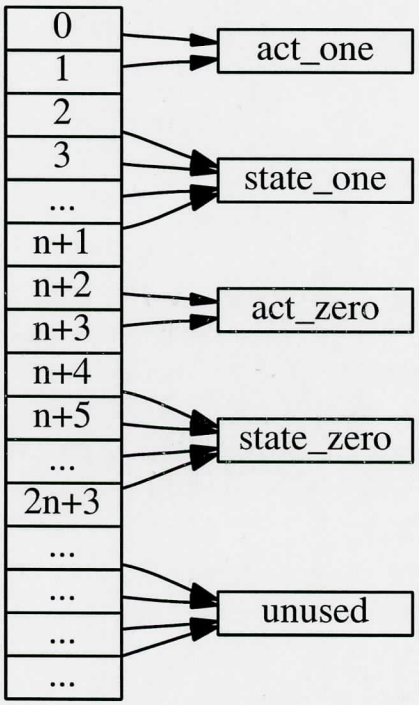


Figure 14. Packing fields for n states.

Problem 5: Some Generalizations on Representation

Generalize your representation to allow 2^n states. Express the number of bits in your representation, and the number of possible individuals, as functions of n.

Figures 13 and 14 show the genetic encoding for the general case of n states. These figures suggest that the number of bits in this representation is $2^n \times (2n+4) + n = 2^{n+1} \times (n+2) + n$. In order to make sure this is correct, let's try $n = 4$ from Problem 3: $2^{n+1} \times (n+2) + n = 2^5 \times 6 + 4 = 196$, which is the same result.

As far as the number of possible individuals is concerned, we can compute it by assuming two distinct values of 0 and 1 for each bit of the genetic encoding. Thus, we get $2^{2^{n+1} \times (n+2) + n}$, which grows faster than the exponent.

Problem 6a: Designing Evolution

Write an outline in English of how your code works and what representations you use. Tell us about the algorithm for selection of individuals for the next generation. Run your system and plot how fitness increases by generation.

As we know, Darwin's theory of evolution includes two sets of balanced biological phenomena. One set provides relative stability in plant and animal species from generation to generation. Another set contributes some source of variation to plant and animal species. The first are called **mechanisms of continuity**, and the latter **mechanisms of variation**. While both sets are necessary for evolution, it is one of the paradoxes of biology that one, the mechanisms of continuity, reflects the perfection of biological systems and that the other, the mechanisms of variation, is nothing more than mistakes or errors in a process of replication. Without these mistakes there could be no evolution, and evolution, therefore, is in a very real sense an accidental process. What this means is that change does not occur in response to need. Nature does not provide species with the inherent ability to adapt to environmental variation. An evolutionary change can occur only if some of the variation already present within the population has a certain value as far as adaptation to new conditions is concerned. In our system we want to imitate the process of **natural selection** under the influence of both of these mechanisms.

The evolution is started with a set of 1,000 randomly generated genotypes, each of which is a bit string of length 196 described in Problem 1. During a generation, each genotype is translated into its respective phenotype, which in our case is an ant's FSA control. Then each ant gets a chance to walk the trail, and its score is recorded. In order to form the next generation, we draw a pair of parent ants randomly from the pool of the best-scoring 1% genomes, and we cross the genotypes of parents to form a child's genotype, who is placed into the new generation. The evolution is not elitist, which means that the best individual is not retained in the gene pool from one generation to the other.

The crossover is probabilistic: the probability of changing from one parent to the other while copying each bit of the genome is set to 10%. Therefore, the expected switch from one parent to the other happens every ten bits of the genome, and since the total length of the genome is 196 bits, the expected value of the number of crossovers to happen in each genome is about 20. The mutation is also probabilistic, and it is set to 1% for each bit to be copied. The mutation itself is a simple bit flip. We can apply the previous argument to see that the expected mutation will happen once every 100 bits, or twice per genome.

The evolution is not elitist, i.e. the best individual of the population is not carried over to the next population explicitly. Nevertheless, as you can see from the explanation of crossover and mutation rules, the probability that one of the high-scoring ants will make it to the next generation without too many damaging mutations is high. In fact, this was our reasoning behind the choice of evolution parameters; otherwise, if we chose to go for higher constants of crossover and mutation probability, we would get an unstable population (see Problem 6c), because high degree of mutation in genomes does not leave much chance to the high-scoring ants to procreate the copies of themselves (although higher mutation rates lead to better adaptation of a species to an environment).

The evolution program is written in ANSI C, and Figures 15 through 19 show the snippets of the representation that we are using. As you can see in Figure 16, each genome is comprised of

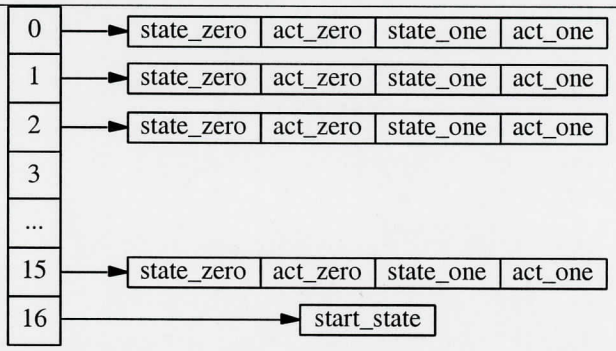


Figure 15. Genetic encoding (repeated for convenience).

```
/* ant's genotype */
typedef struct {
  /* each of the first 16 codons is:

  |<----- 4 ------>|<---- 2 ---->|<----- 4 ------>|<---- 2 ---->|
  | new_state_on_0 | action_on_0 | new_state_on_1 | action_on_1 |

  aligned to the right boundary of UInt16;
  the last codon (codon[FSA_STATES]) is the start state (same right alignment)*/
  UInt16 codon[FSA_STATES+1];
} Tgenotype;
```

Figure 16. Genotype.

seventeen codons (we are using 16-state FSAs), the first sixteen of which encode state transitions and actions taken upon these transitions, and the last one encodes the start state of the FSA (see Figure 15). We decided to expose the start state to the evolution in order to give it more freedom. The phenotype representation (Figure 17) is a set of arrays encoding transitions and actions, as well as the start state. The ant (Figure 18) is simply a type containing both the genotype and the phenotype. However, the population is comprised of genomes, but not ants, in order to lessen memory requirements of the program and allow up to 65,536 individuals in the population (otherwise, the program could segfault). The representation of the trail is given in Figure 19. Due to our representation, the code is fast (evolving 1,000 ants for 2,000 generations takes about half-an-hour on a 1-GHz Pentium-III Linux machine). To see that, look at Figure 20: most of the code is simple shifting and masking.

Now, having described the representation, let's see what actually is happening and how evolution works. Figure 21 shows the champion of all randomly-generated ants of generation 0 and the history of its walk on the John Muir trail. As you can see, this random ant does a pretty good job and scores 42 points, but gets lost at the first missing cell. However, after 2,000 generations, the

```
/* ant's phenotype (FSA control) */
typedef struct {
  UInt8 start;                  /* start state */
  UInt8 state_zero[FSA_STATES]; /* new state on 0 */
  UInt8 state_one[FSA_STATES];  /* new state on 1 */
  UInt8 act_zero[FSA_STATES];   /* action on 0 */
  UInt8 act_one[FSA_STATES];    /* action on 1 */
} Tphenotype;
```

Figure 17. Phenotype.

```
/* ant */
typedef struct {
  Tgenotype gene;  /* genotype */
  Tphenotype phen; /* phenotype */
} Tant;
```

Figure 18. Ant.

```
/* trail */
typedef struct {
  UInt32 cell[TRAIL_SIZE][TRAIL_SIZE]; /* array of cells */
  UInt32 maxfood;                      /* amount of food in the trail */
} Ttrail;
```

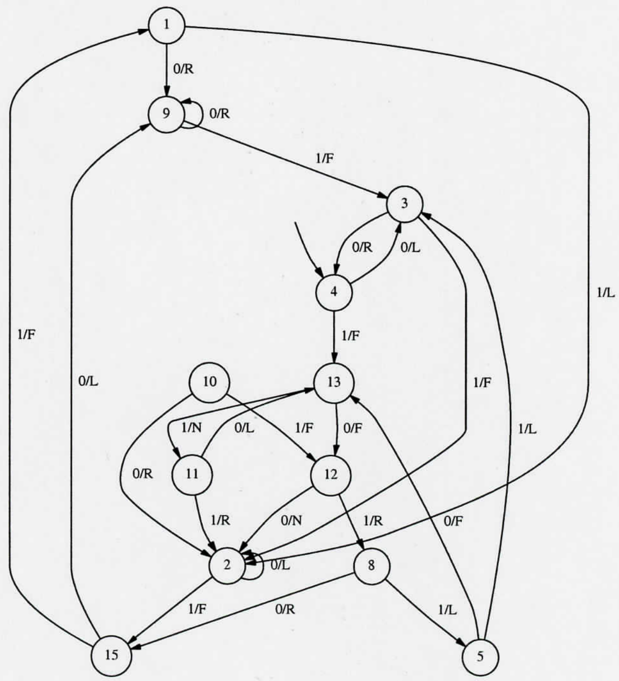
Figure 19. Trail.

```
/* mutate each bit of the gene with a given probability */
void mutate_genome (Tgenotype *gene, float prob) {
  UInt16 i;      /* counter */
  UInt32 thresh; /* mutation threshold */
  UInt16 pos;    /* potential flip position */
  UInt16 mask;   /* mask in the form of 0x00...010...0, with "1" at pos=i */
  UInt16 temp;   /* temp variable */

  thresh = (unsigned int) ((1+(unsigned int)RAND_MAX)*prob);

  /* repeat for all codons */
  for (i=0; i<=FSA_STATES; i++) {
    mask = 0x1; /* reset mask */
    for (pos=0; pos<12; pos++) {
      mask = 0x1 << pos; /* update mask */
      /* if flip */
      if (rand()<thresh) {
        temp = gene->codon[i];
        temp = (temp & (~mask)) | ((~temp)& mask);
        gene->codon[i] = temp;
      }
    }
  }

  /* reset the unimportant part to zero*/
  gene->codon[FSA_STATES] = gene->codon[FSA_STATES] & 0xF;
  return;
}
```

Figure 20. Mutation code.

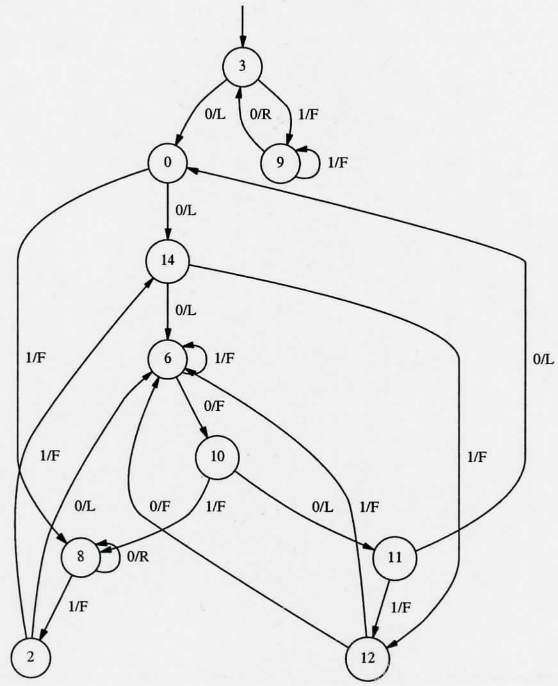
best ant can get the full score!! It's FSA, as well as the history of its walk on the trail, are given in Figure 22. As you can see, the evolution has done a very good job in specializing ants to the John Muir trail! It is important to notice that all FSAs have been automatically minimized to remove unreachable states.

Now let's take a look at how fitness increases by generation. Figure 23 shows fitness as a function of generation. The mean score is shown by solid line, the standard deviation score by medium dashed line, and the maximum score by short dashed line. As you can see, it only takes the evolution thirty generations or so to produce an ant scoring 80 points out of 89. Another observation is that the score is not non-decreasing – generations around 1,200 are a good example. After 2,000 generations, the final statistics is the following: $max = 89$, $mean = 58.6$, $stdev = 23.7$. Mode 1 is not shown on the graph, because it adds more noise to the graph than opportunity of useful analysis.



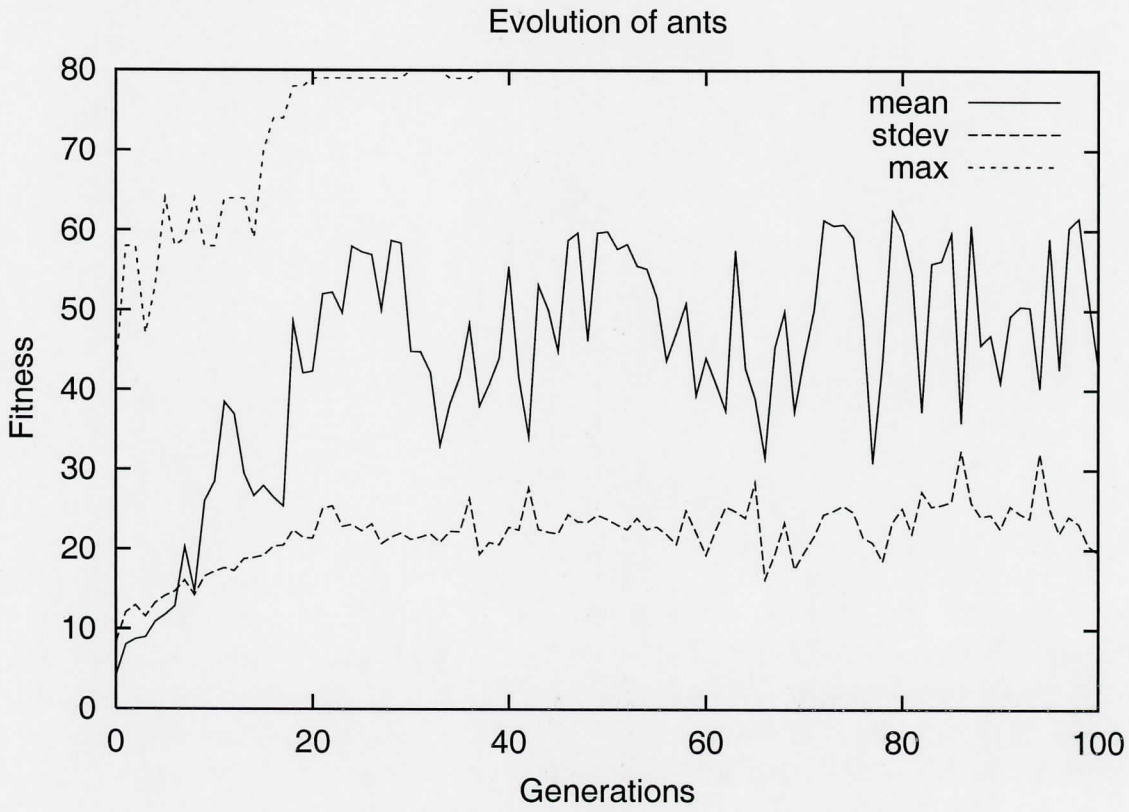

```
04 02 15 02 15 02 15 02 15 02 09 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  03 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  15 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
15 02 03 09 .  .  .  .  .  .  15 .  .  .  .  .  .  .  .  .  .  .  .  .  .  09 15 02 15 02 15 02
.  .  .  02 .  .  .  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  15 .  .  .  .  .  .  15 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  02 .  .  .  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  03 .  .  .  .  .  .  15 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  09 02 15 02 15 02 03 09 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Food gathered: 42
```

Figure 21. Champion of generation 0.




```
03 09 09 09 09 09 09 09 09 09 03 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  09 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  09 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  09 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  09 .  .  .  14 12 06 10 .  .  .  .  .  .  .  .  .  .  .  .  .  .
06 06 12 14 .  .  .  .  .  .  09 .  .  .  06 .  .  .  .  .  .  .  .  .  11 06 06 06 06 06 06 06
.  .  .  09 .  .  .  .  .  .  09 .  .  .  06 .  .  .  .  .  .  .  .  .  12 .  .  .  .  .  .  .
.  .  .  09 .  .  .  .  .  .  09 .  .  .  12 .  .  .  .  .  .  .  .  .  06 .  .  .  .  .  .  .
.  .  .  09 .  .  .  .  .  .  09 14 12 06 11 .  .  .  .  .  .  .  .  .  06 .  .  .  .  .  .  .
.  .  .  09 .  .  .  .  .  .  09 06 .  .  .  .  .  .  .  .  .  .  .  .  06 .  .  .  .  .  .  .
.  .  .  03 09 09 09 09 09 09 03 06 .  .  .  .  .  .  .  .  .  .  .  .  06 .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  12 .  .  .  .  .  .  11 06 06 06 06 12 14 .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  14 06 06 12 11 .  .  12 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  06 .  .  06 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  12 .  .  06 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  14 12 06 06 11 .  .  06 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  06 .  .  .  .  .  .  06 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  12 .  .  .  .  .  .  06 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  14 12 06 06 11 .  .  .  .  .  .  06 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  06 .  .  .  .  .  .  .  .  .  .  10 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  06 .  .  .  .  .  .  .  .  .  .  08 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  12 .  .  .  .  .  .  .  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  14 12 06 11 .  .  .  .  .  .  .  .  .  .  14 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  06 .  .  .  .  .  .  .  .  .  .  .  .  .  12 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  12 .  .  .  .  .  .  .  .  .  .  .  .  .  06 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  14 .  .  .  .  .  .  .  .  .  .  .  .  .  06 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  06 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  08 10 06 12 14 02 08 10 06 06 06 06 06 12 14 .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Food gathered: 89
```

Figure 22. Champion of generation 2,000.



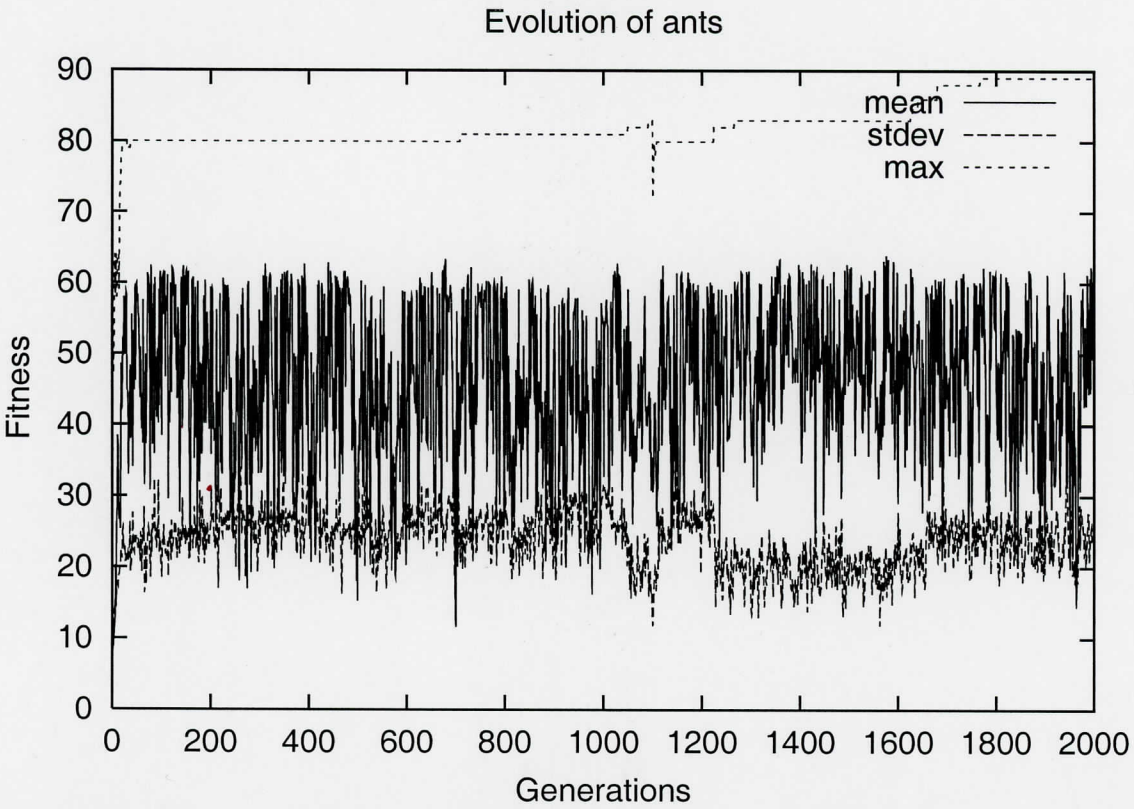


Figure 23. Fitness as a function of generation (score=89; population=1,000; mutation=1%; crossover=10%; breed=1%).

10/10 great

Problem 6b: Cross-Trail Performance

Make your system work on some number of the `aannn.txt` *files to evolve your ant. Take one of the highest scoring ants evolved and try it on the* `bbnnn.txt` *files. Report the results of this to us. If there is a significant difference what could be the explanation?*

It is known that species which have recently invaded a particular geographic zone and which therefore have not adapted to it are said to be **generalized** for that zone. As the process of adaptation unfolds, the resulting species become more and more **specialized**. That is, their adaptation becomes more and more efficient for a specific environment. Specialization represents an increase in population efficiency since it reflects successful adaptation. Specialization, however, also leads to restricted variation, since any significant change from a specialized form is highly likely to be less viable that the established type. Thus old, fully established species, well adapted to a specific and rather invariant environment, are likely to change little or not at all. They have "traded" variability for continuity geared to long-standing conditions.

To see the phenomenon of specialization in our evolution system, we evolved a population of 1,000 ants on three trails from the "a" suite for 2,000 generations and simultaneously measured their performance on the three other trails of the "b" suite. The results are shown in Figure 24. As you can see, there is a huge fitness gap in these two measurements, which illustrates the aforementioned principle of specialization. The results are quite remarkable: the champion of generation 2,000 scores 248 out of 267 points (93%) on the "a" suite, while as it scores 126 out of 267 (47%) on the "b" suite. We will analyze the specificity of this ant's behavior in problem 7.

should really have done more trails — some did all 572... but I give full credit b/c of depth of answer

10/10

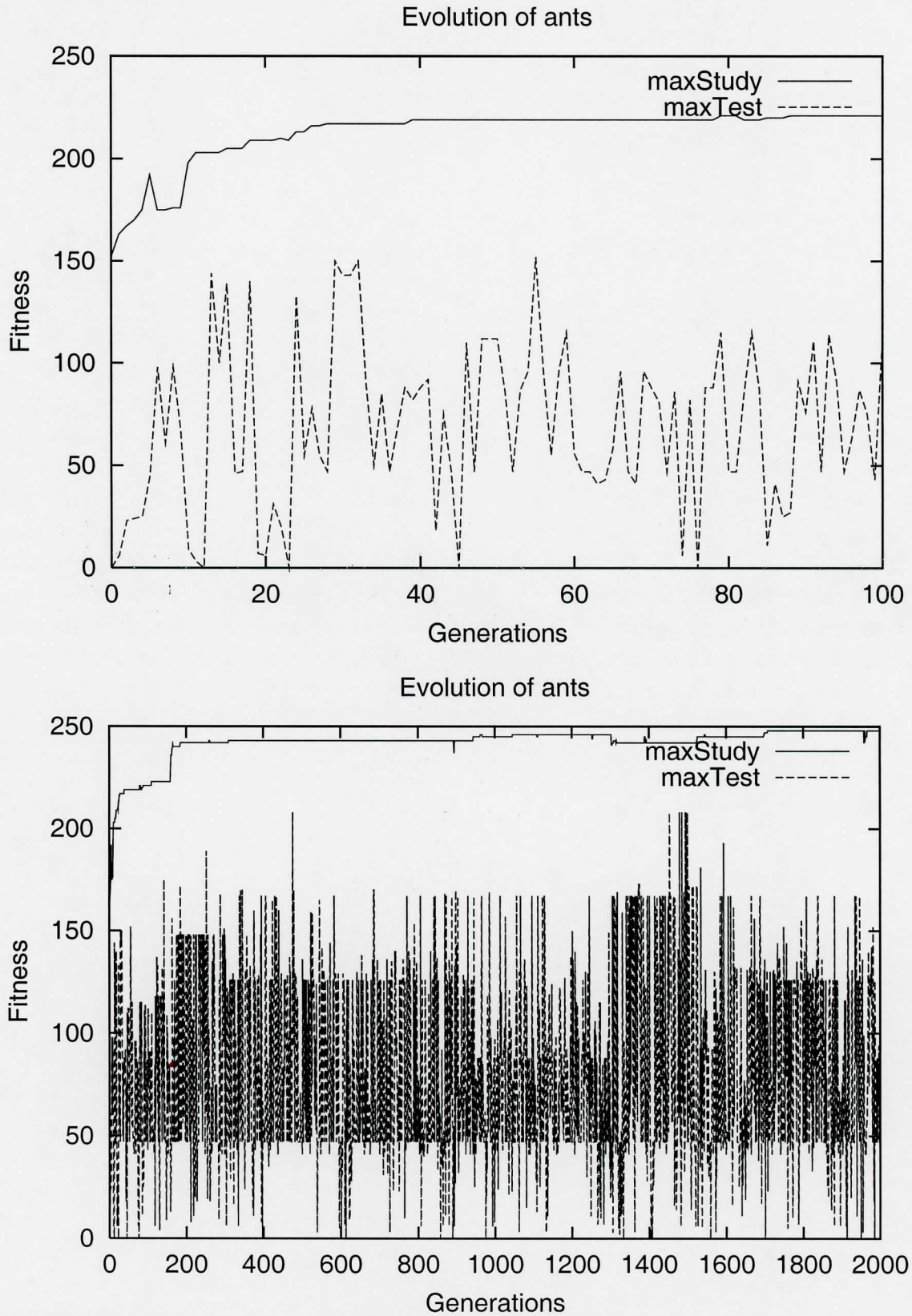


Figure 24. Relative performance of specialized evolved ants in "friendly" (maxStudy) and "hostile" (maxTest) environments.

Problem 6c: Fitness Analysis

Generate some tables on how the overall fitness varies on each set of files for different population sizes, different parameters for what proportion of the fittest individuals are retained, what proportion are used for reproduction, and what levels of mutation are used.

This part of the assignment presents a sensitivity study of evolution behavior under changes of evolution parameters, namely crossover, mutation, and breeding threshold rates, as well as population size. Figure 25 showing the normal evolutionary process is repeated here for convenience of comparison.

Crossover. Figure 26 shows evolution under extremely low crossover rate. As you can see, the general picture is not different from that of the normal evolution (Figure 25), but the maximum fitness line has less oscillations and is more smooth. The maximum fitness achieved is in high 80s. However, if the crossover rate is set to an extremely high value (Figure 27), the general picture is the same, but the evolution only makes it to the score a little over 80. This probably happens because high level of crossover introduces too much randomness, and the best ants are not retained. Thus, we can conclude that evolution is robust under any reasonable changes of the crossover parameter.

Mutation. The consequences of an extremely low mutation rate are depicted in Figure 28. As you can witness, the mean fitness and the maximum fitness lines collide a little above 45 points, and the standard deviation line representing the variation of the population is located at the very bottom of the graph, at the level of around 3. This certainly means that the population is very uniform, and no evolution change is happening – the mechanism of variation is turned off (see the first paragraph of Problem 6a for further explanation). However, if we turn mutation to a very high level (see Figure 29), we will not let successful ants procreate properly, and the evolutionary change will be destroyed by this randomness (we can see that the mean value oscillates at the level of 5 units, and the best ants are very unstable in successive generations). The conclusion is that evolution is sensitive to changes of mutation.

Breeding threshold. Figure 30 represents the process of evolution under an extremely low breeding threshold rate. As you see, the graph is practically identical to Figure 25 of the normal evolution, except for there are no oscillations of the maximum fitness. The reason is obvious – since less individuals are used for breeding, the children generation is more uniform. However, if we tune the breeding threshold high up (Figure 31), the evolution is dead. Again, the reason for that is simple – there is no natural selection in the population, so the high-scoring ants mate with poor-scoring ants, which results in poor-scoring children (the process of spoiling of the genome). Thus, the conclusion is that evolution is very robust to decreasing the breeding threshold, but it is destroyed by increasing breeding threshold. As a side note to prove the conclusion let's recall that 6% of the sea lions inseminate 99% of females – a phenomenon frequently observed within various animal species.

Population size. Here we see an interesting phenomenon: if you look at Figures 25, 32, and 33, you will notice that the number of generations that it takes to evolve a reasonably well-performing ant (scoring 80 points) is inversely proportional to the size of the population, and the product $|population| \times |generations| = 100 \times 600 = 1,000 \times 60 = 65,536 \times 1$ remains constant for all population sizes. Therefore, the evolution process is robust with respect to the population size, and

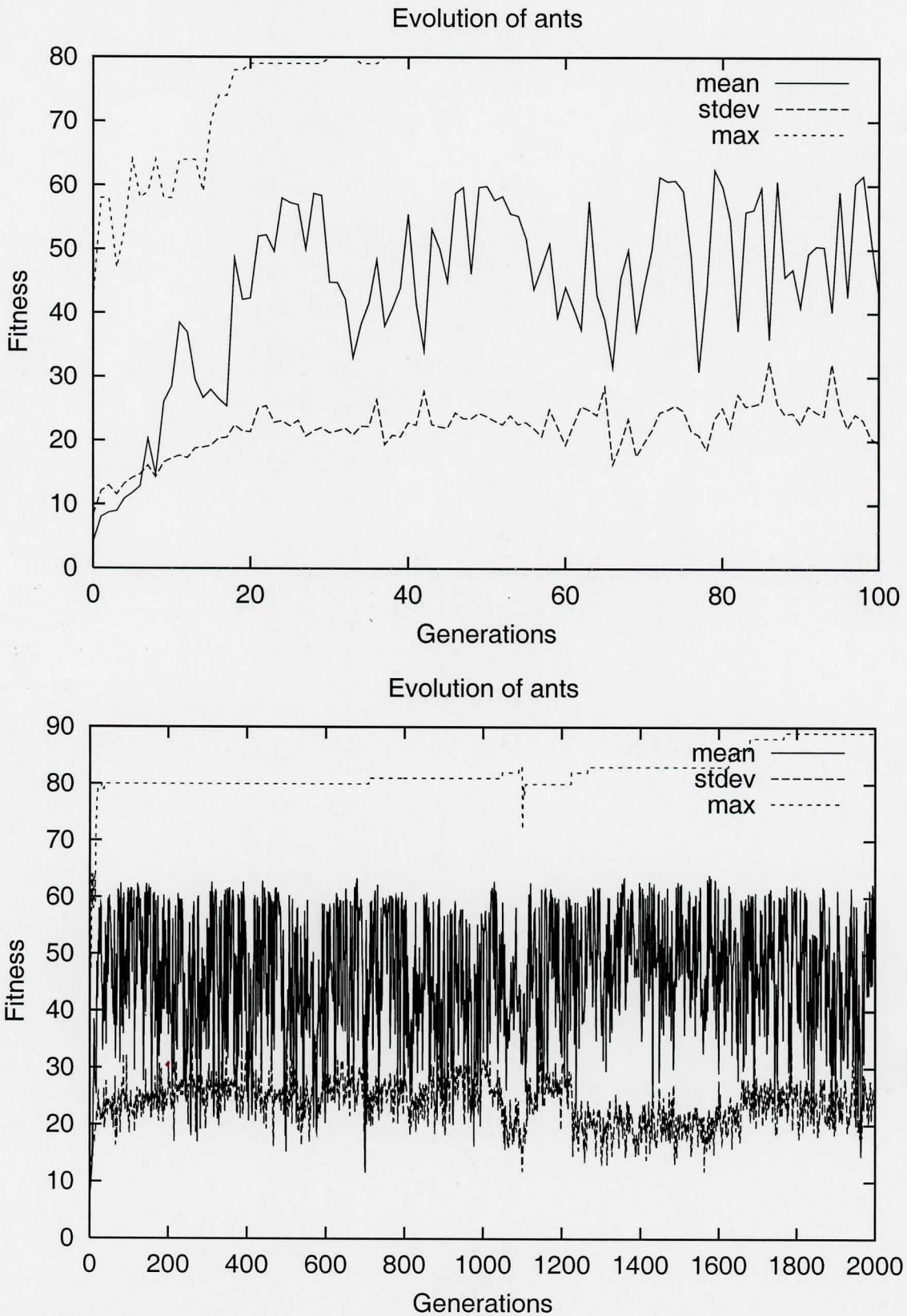
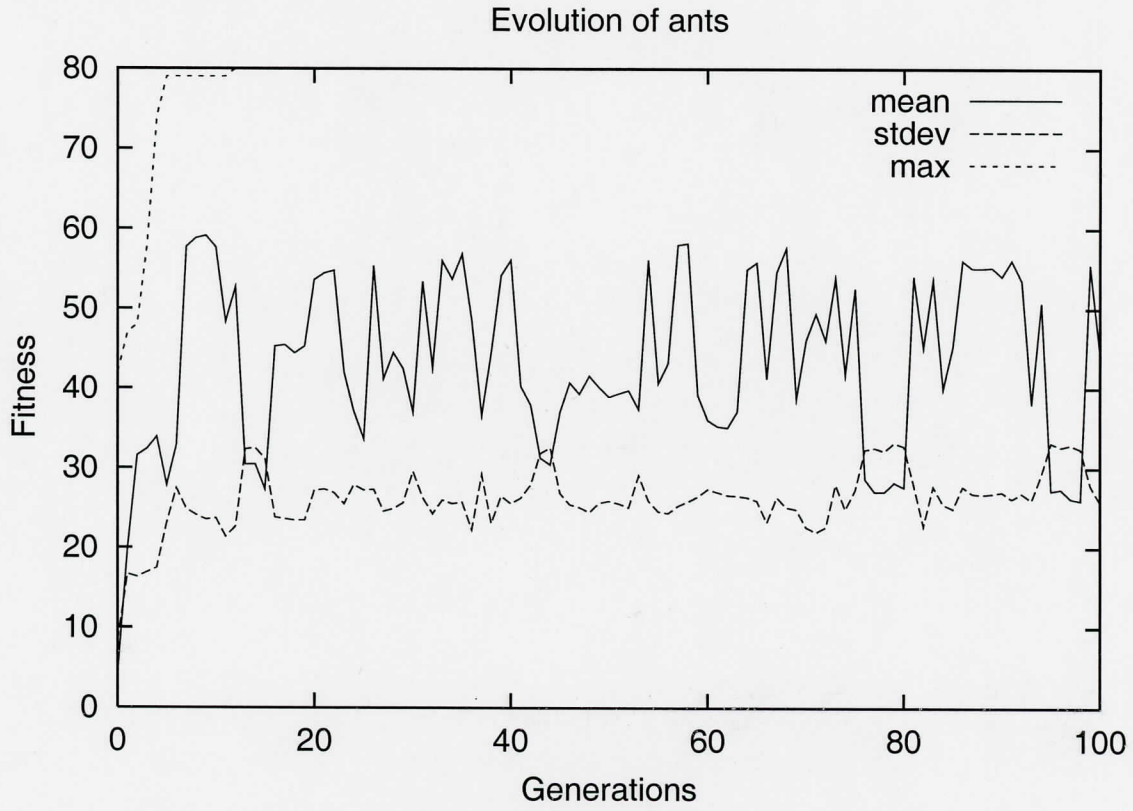


Figure 25. Normal evolutionary process (score=89; population=1,000; mutation=1%; crossover=10%; breed=1%).



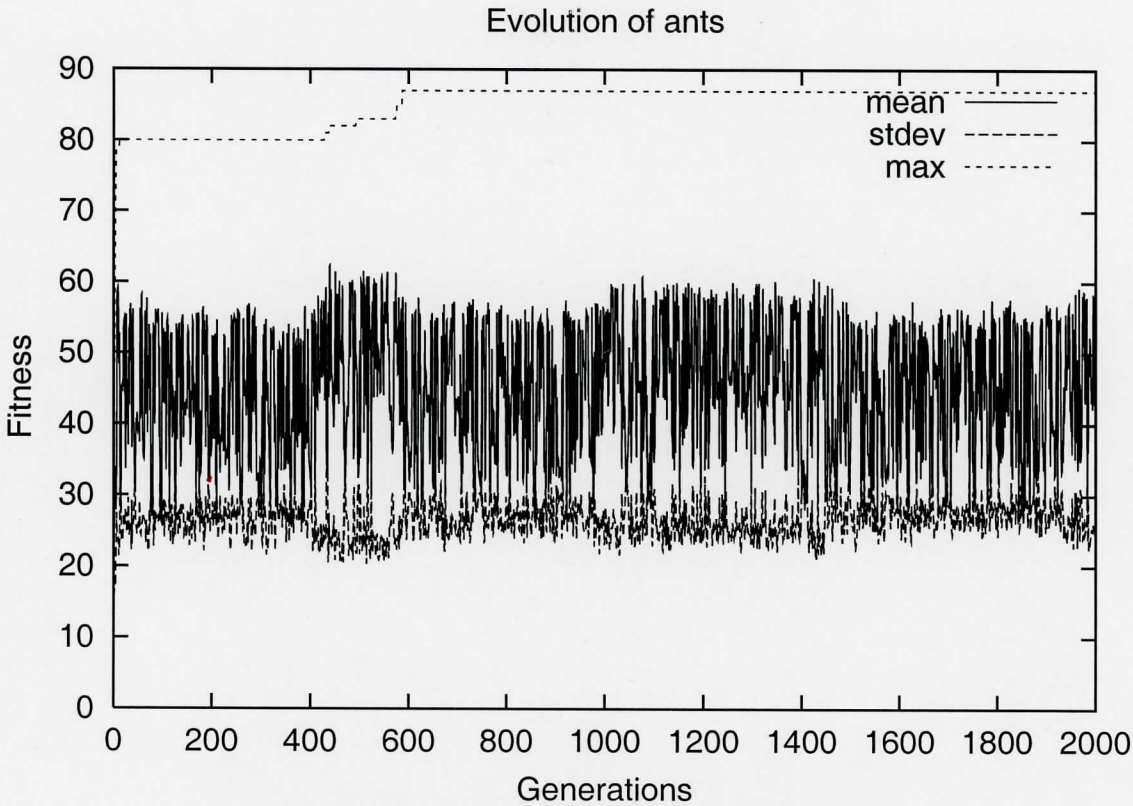
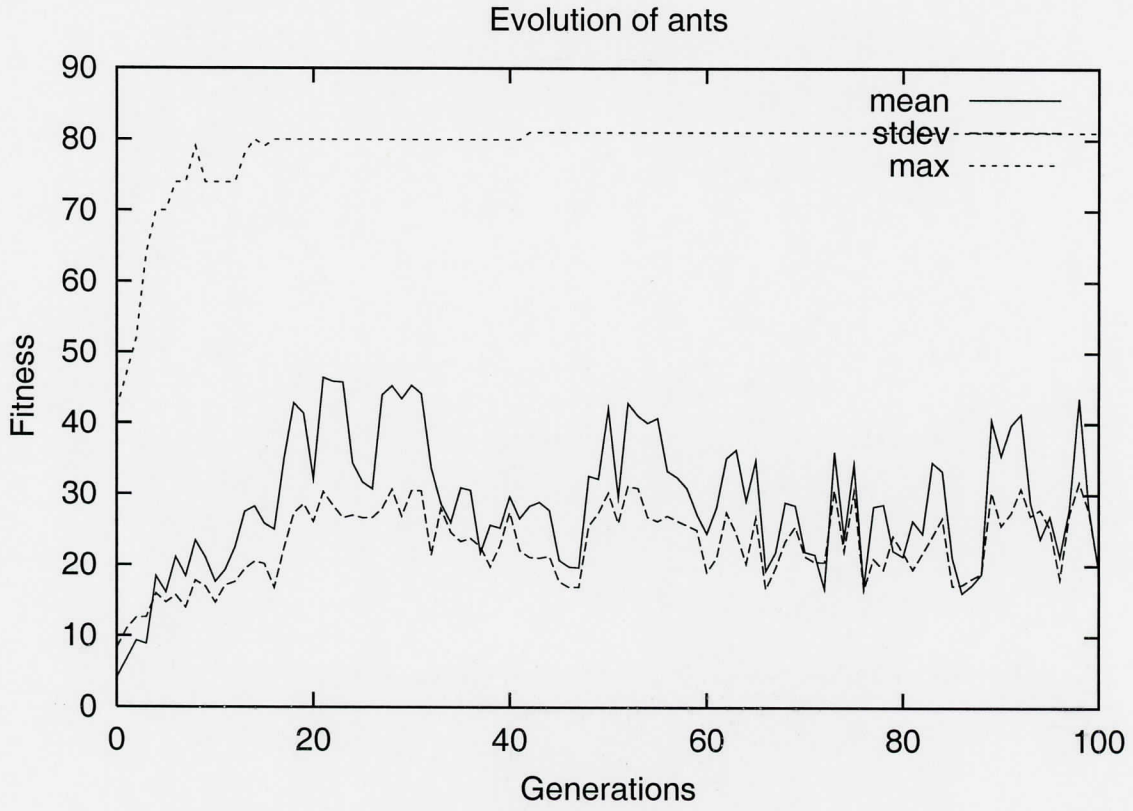


Figure 26. Extremely low crossover rate (0.01%).



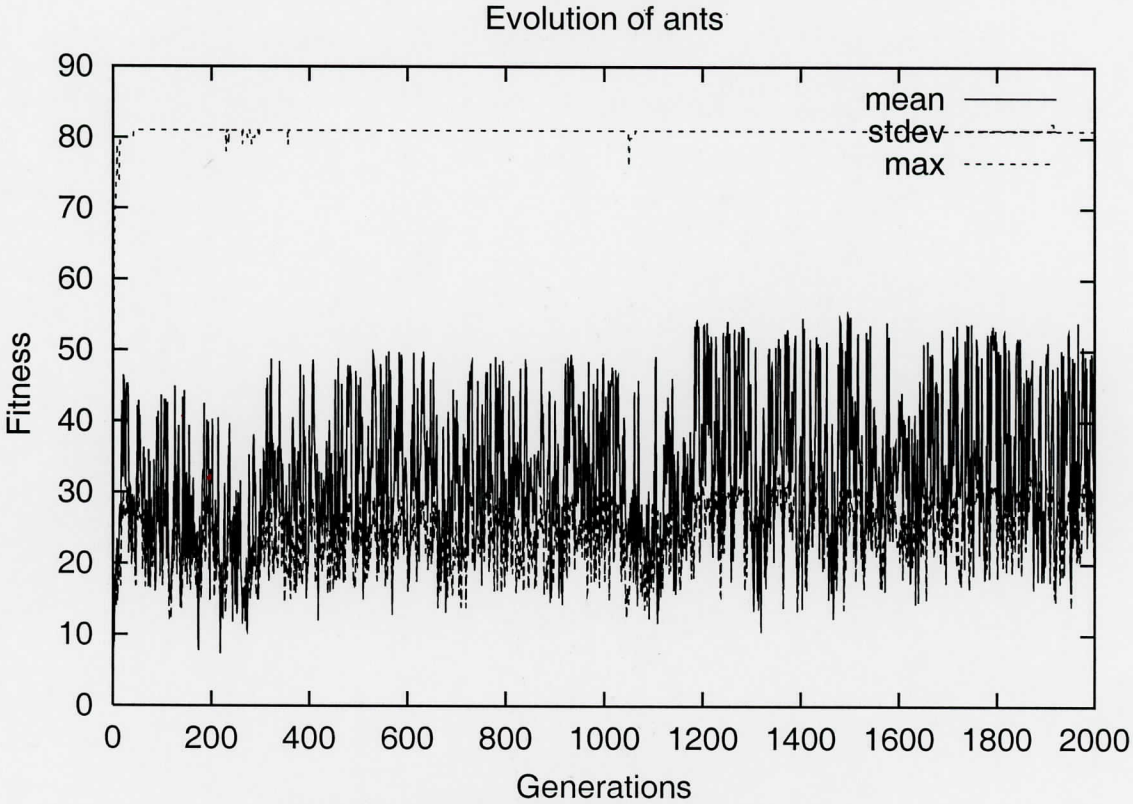
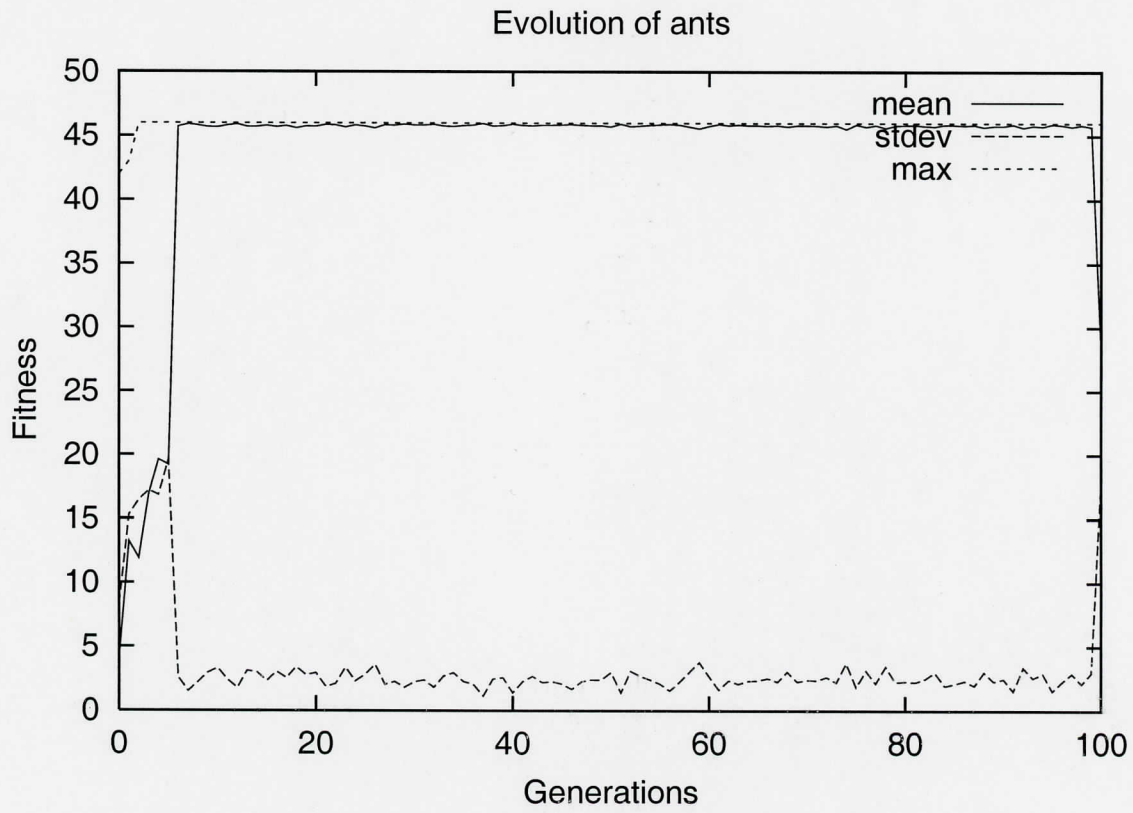


Figure 27. Extremely high crossover rate (50%).



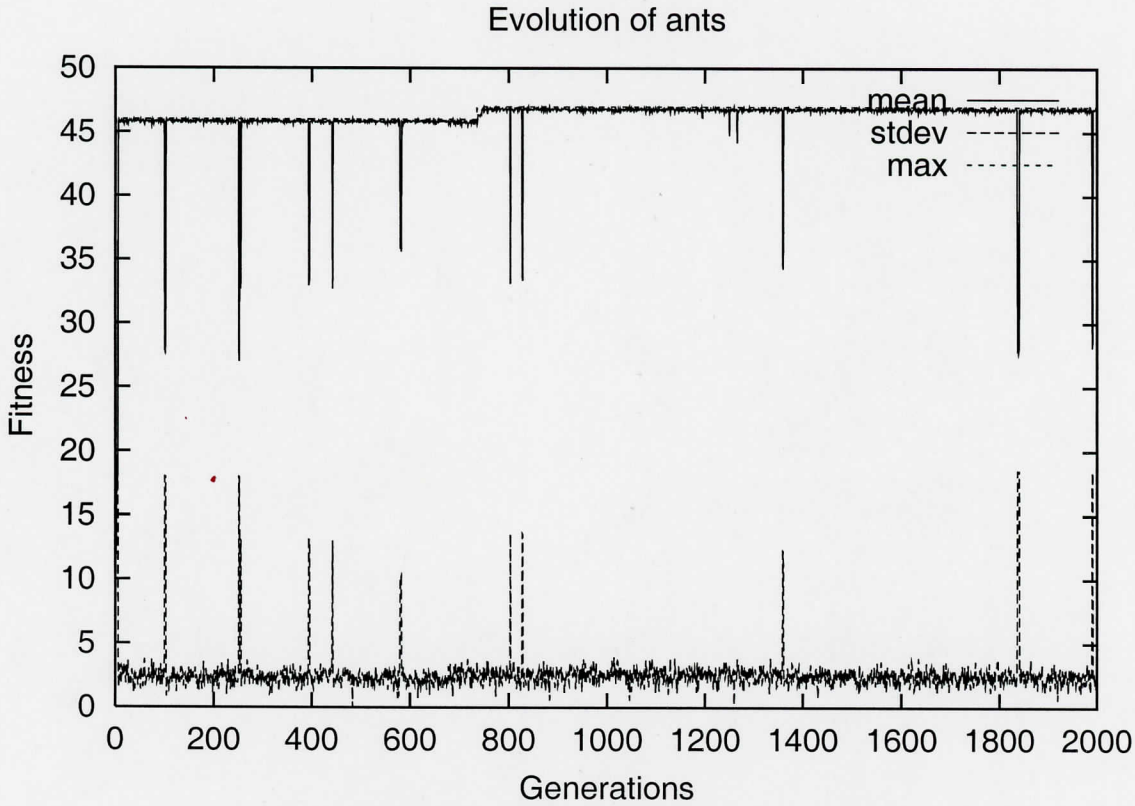


Figure 28. Extremely low mutation rate (0.01%).

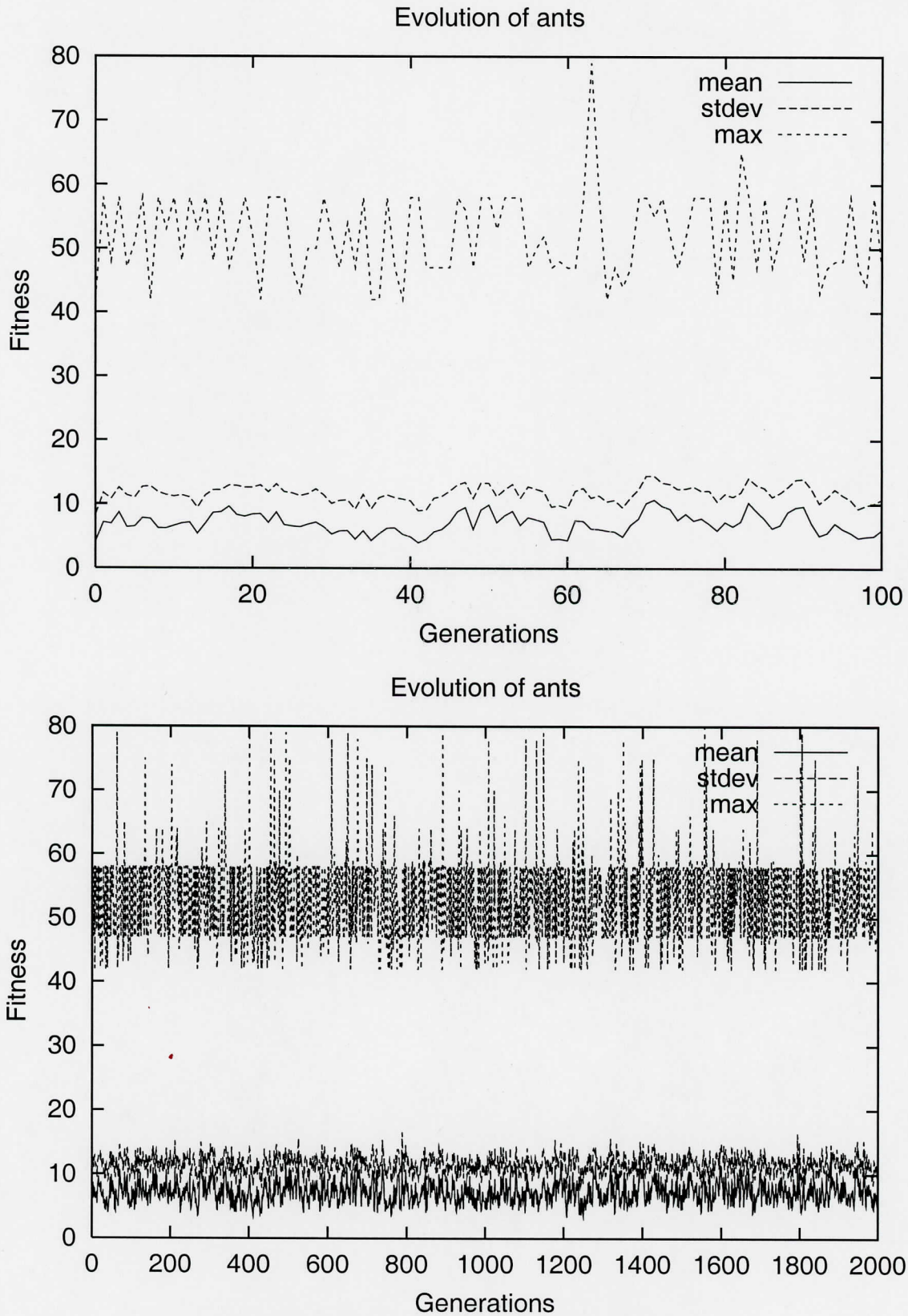
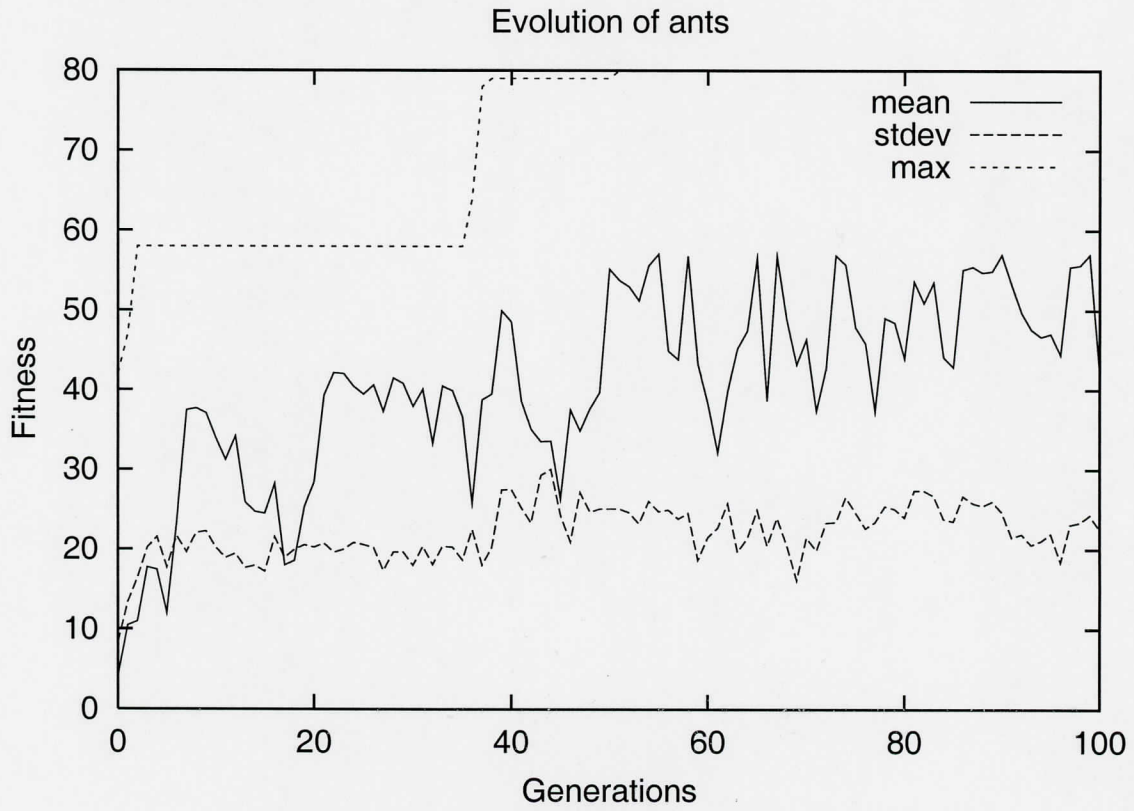


Figure 29. Extremely high mutation rate (50%).



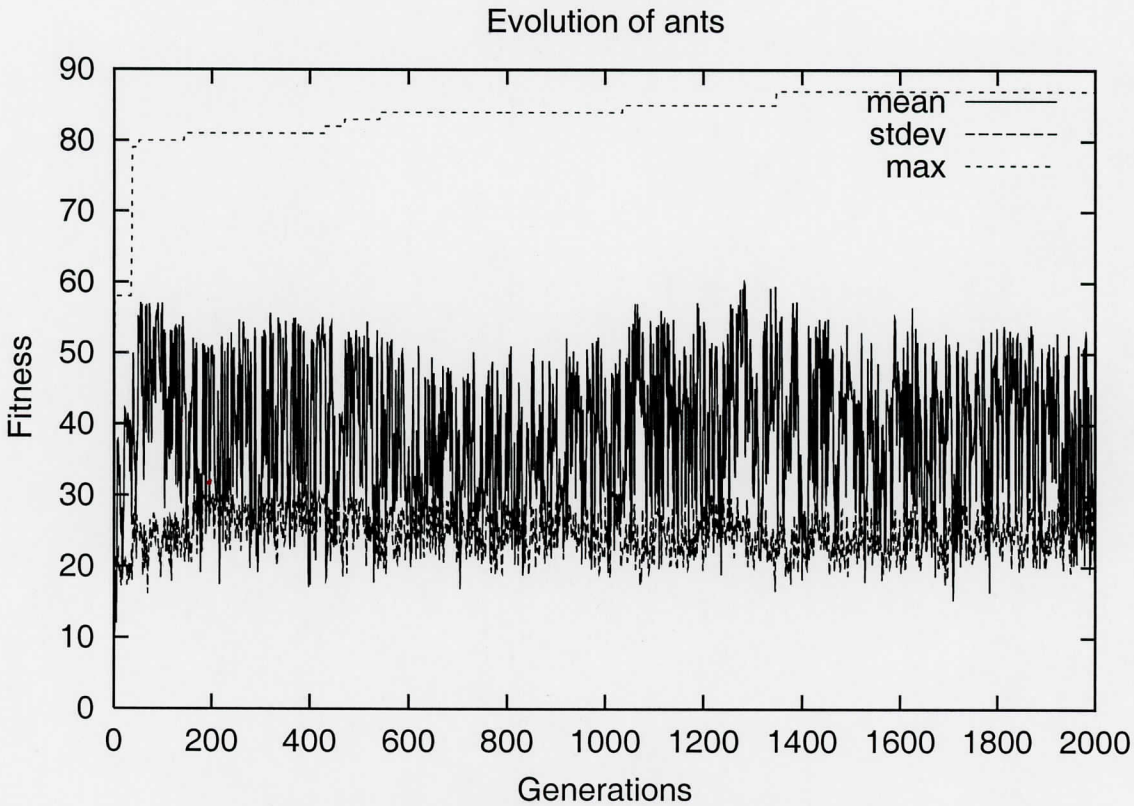
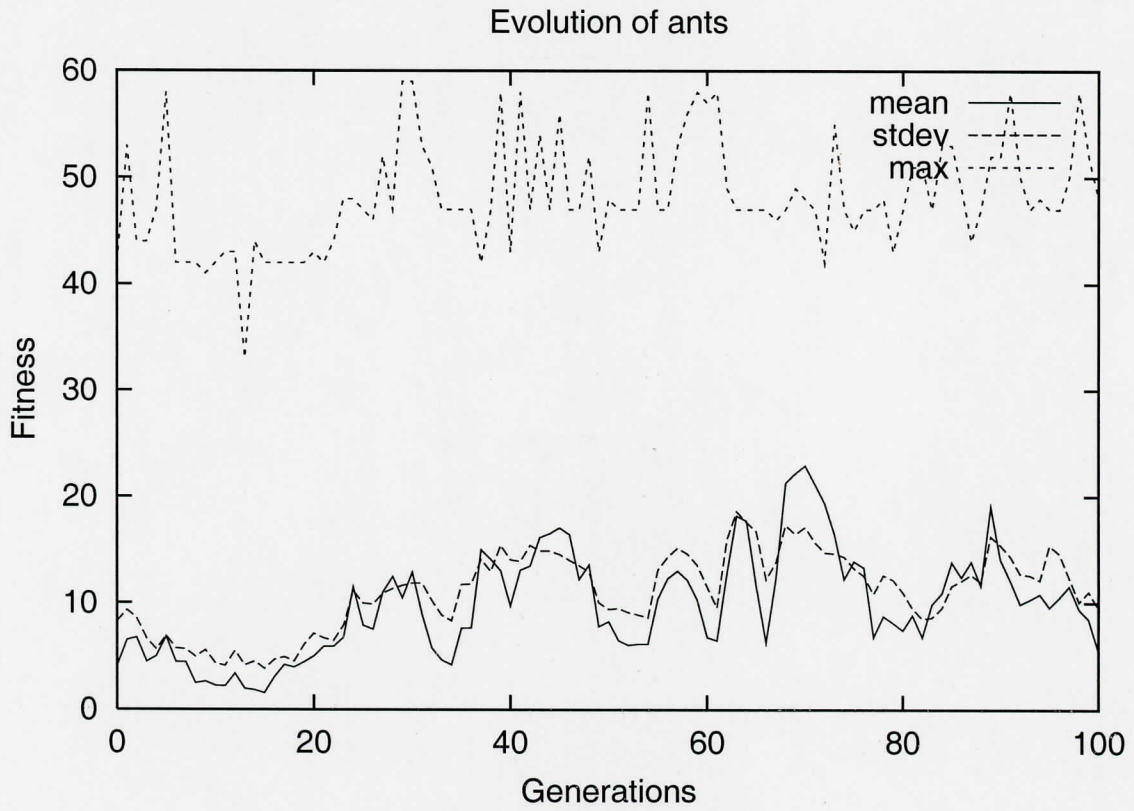


Figure 30. Extremely low breeding threshold rate (0.01%).



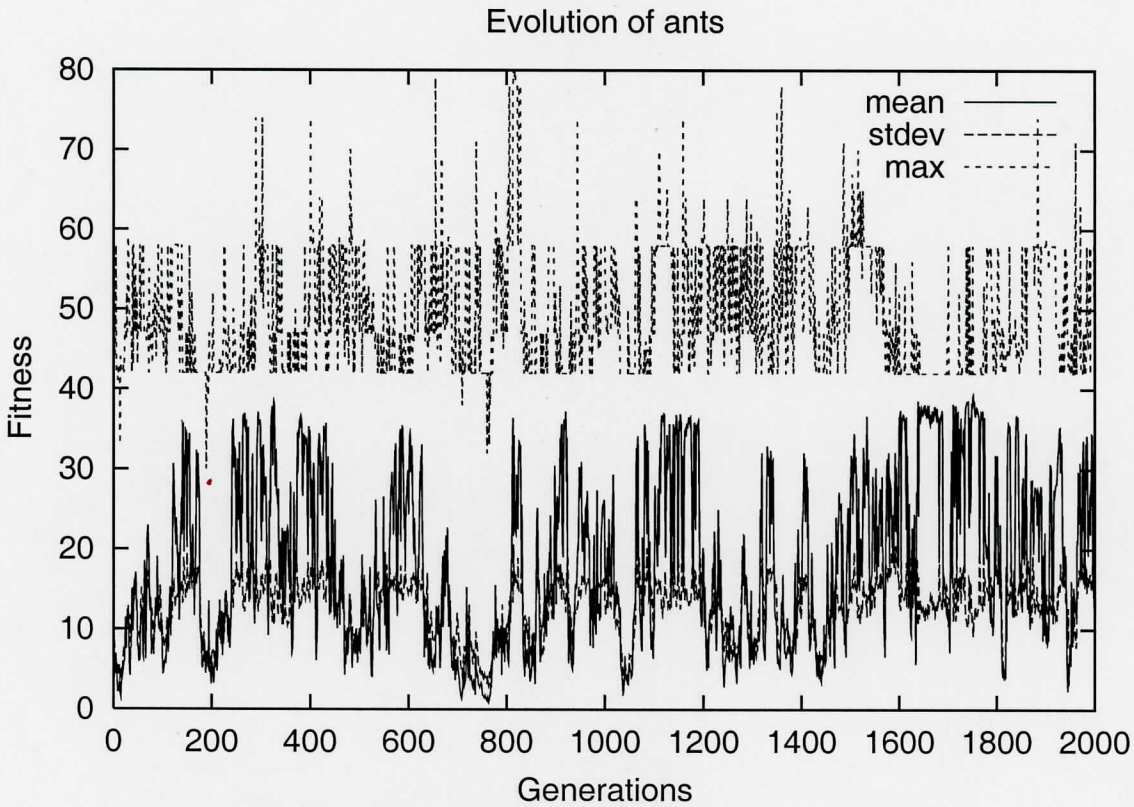


Figure 31. Extremely high breeding threshold rate (50%).

the emergence of high-scoring ants is only the matter of time, shorter for bigger populations and longer for smaller ones.

Thus, the general conclusion is that the qualitative nature of the evolutionary process is extremely robust over wide variations of the evolutionary parameters.

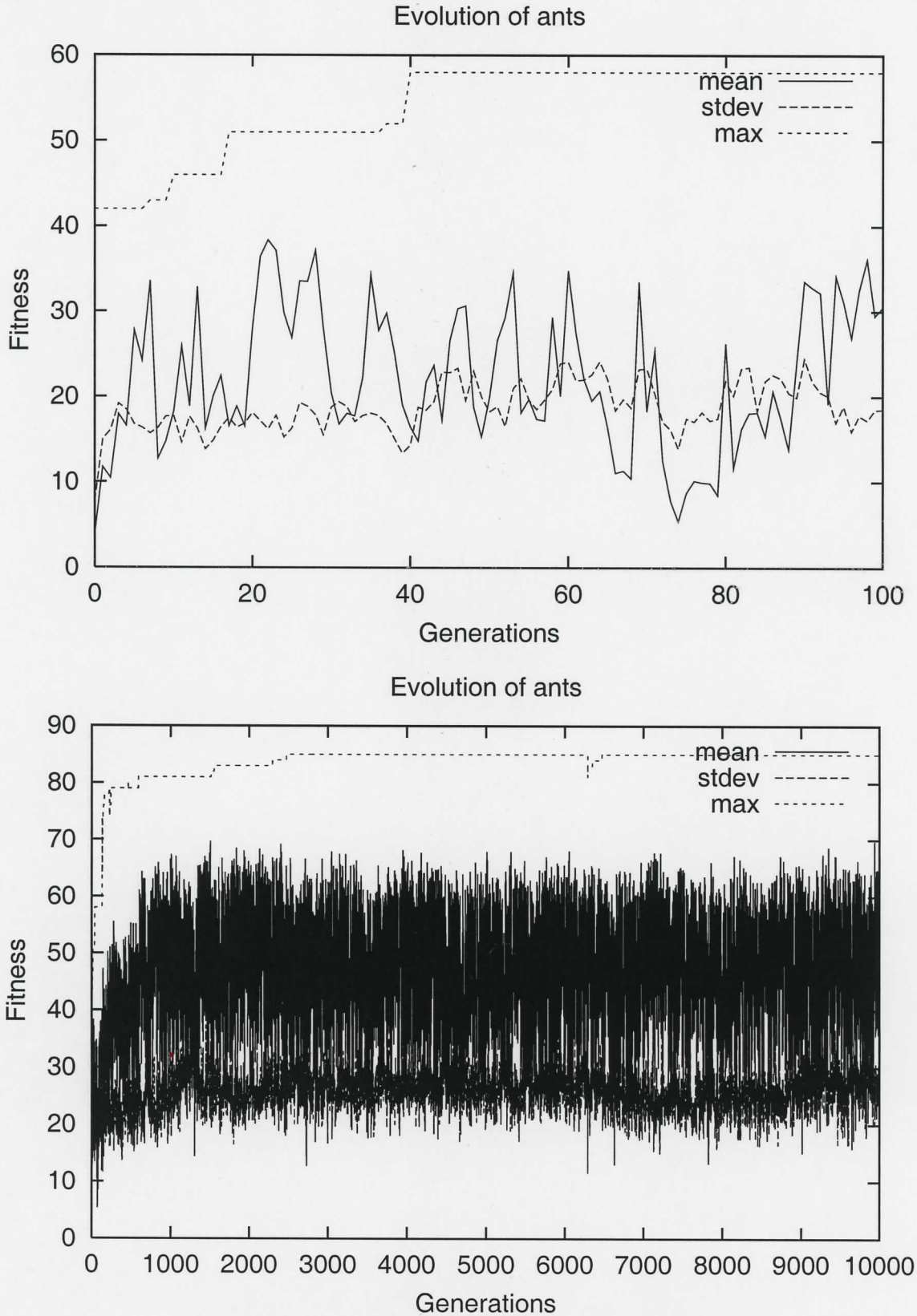
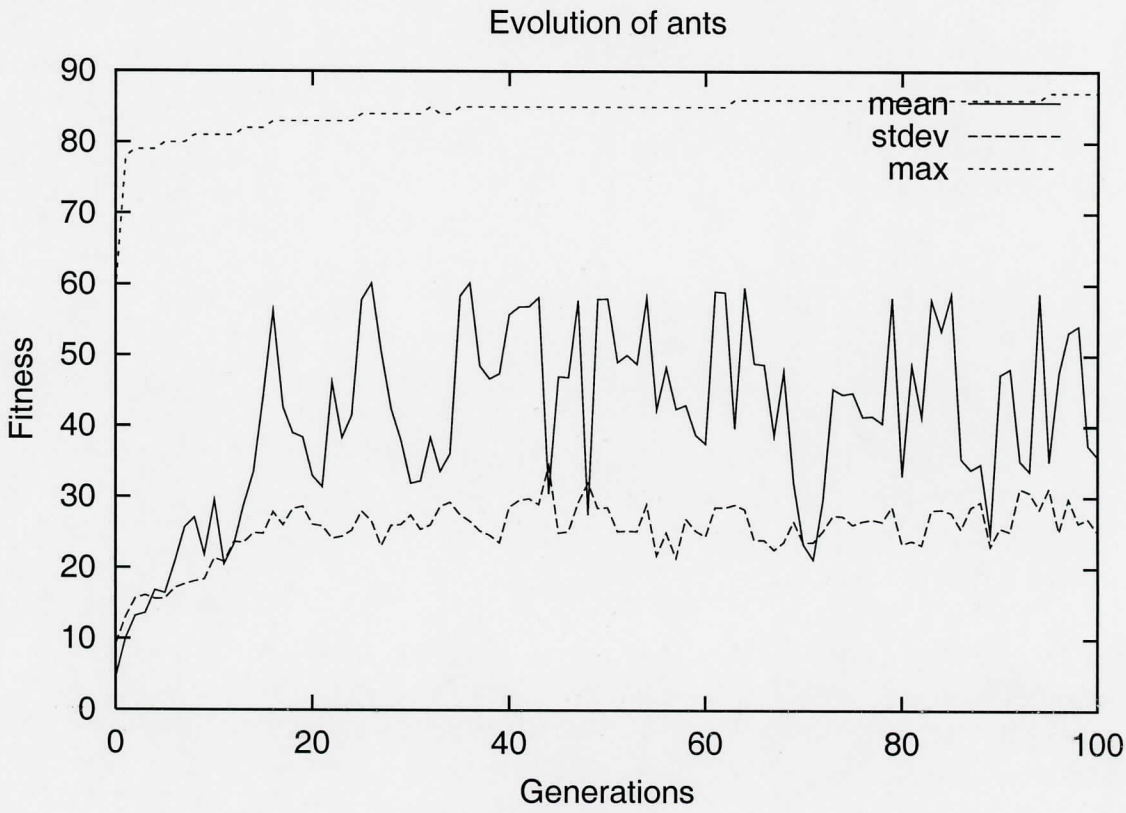


Figure 32. Extremely small population size (100 ants).



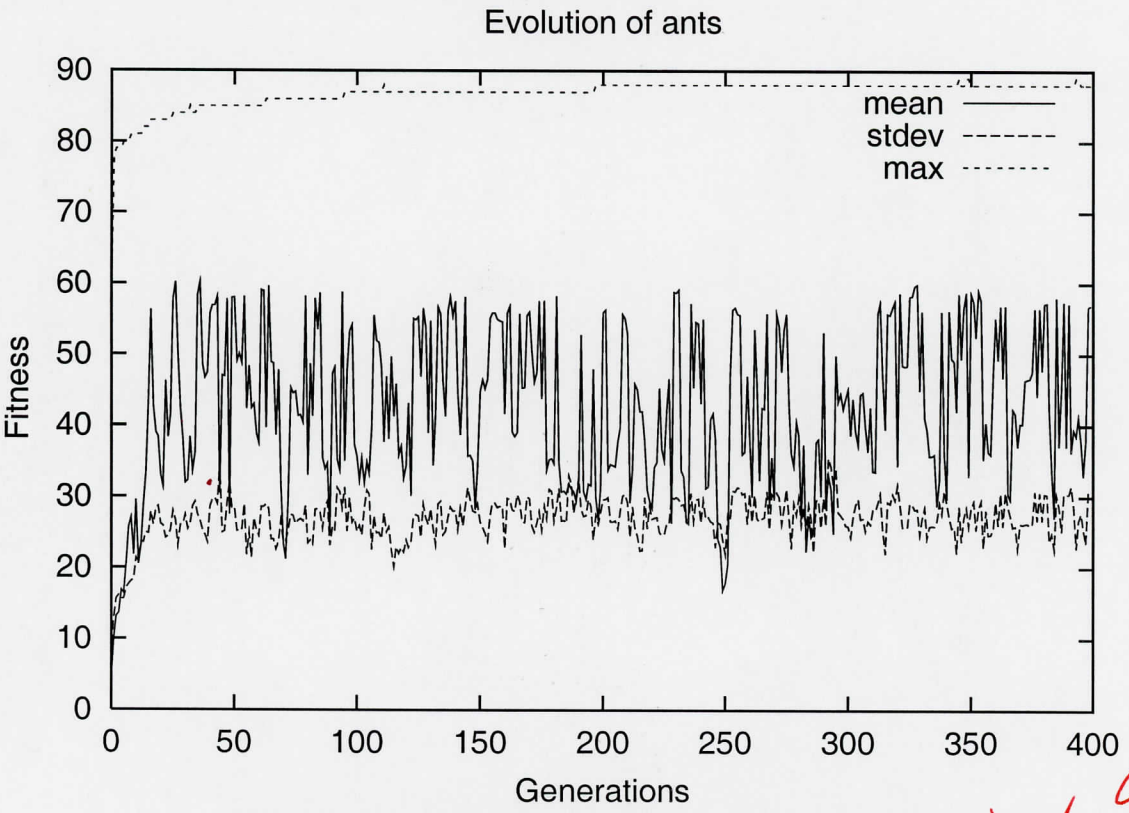


Figure 33. Extremely big population size (65,536 ants).

8/8
very thorough

Problem 7: Best Behavior Analysis

Take one of the best individuals your system produced for each of problem 6a and 6b (i.e., their FSA state-transition tables and diagrams), and analyze its behavior. Does the problem 6a ant demonstrate any particular specializations? What about the 6b ant? And if it does better on the "a" files that the "b" files, what might be going on? How do both compare to your hand-coded solution for Problem 1?

We will now attempt to analyze the behavior of evolved ants for Problems 6a and 6b. Let's first start with the John Muir trail walker (see Figures 34 and 35). This ant seems to have absorbed the knowledge of this particular trail into its FSA. Indeed, since there is a bias towards right turns in the beginning of the trail, it performs right turns in a single step, and otherwise efficiently walks the trail while staying in one of the states 09 or 06. However, the left turn is done with extra two steps to turn right and then back left. As you can see, this ant efficiently reuses the sequence of states (06-11-12) seven times in order to do a left turn in the absense of food. The respective right turn in the absense of food is done through the sequence (06-14-12), which is repeated eight times. Also, the ant actually manages to do one extra step after it has eaten all of the food!

As for Problem 6b ant, things aren't that simple here – the FSA is more complicated to suite the needs of all three trails on which the ants have been evolved (see Figures 36 through 42). But again, even with this complexity, there is some nice reuse of state sequences, such as that of (02-11-10-12-09-07-13-02), which looks sort of like "dancing", and (02-13-02), which is simply a turn right along the trail. However, now the ant is so specialized that it scores 86, 82, and 80 points respectively in all three trails on which it is "trained"! Alas, this over-specialization fails to work in other environments of the "b" test trails, where the ant can only score 31, 23, and 71 points. As you see, the performance is 248 out of 267 points (93%) on the "a" suite versus 126 out of 267 (47%) on the "b" suite. We send you back to Problem 6b for our argument which explains why there is this 2x gap (the argument of specialization versus generalization as evolutionary forces).

As far as the comparison to the hand-coded solution is concerned, there is an obvious difference: our hand-coded solution is very general (see the argument in Problem 1), and thus it is guaranteed to work in any reasonable environments, while as the evolved ant is highly specialized to either walk the John Muir trail (as in Problem 6a) or a set of study trails (Problem 6b) perfectly or almost perfectly. Both approaches have their niches and applications, but let's not forget that evolution tries to "optimize" solutions for a particular environment.

14
14

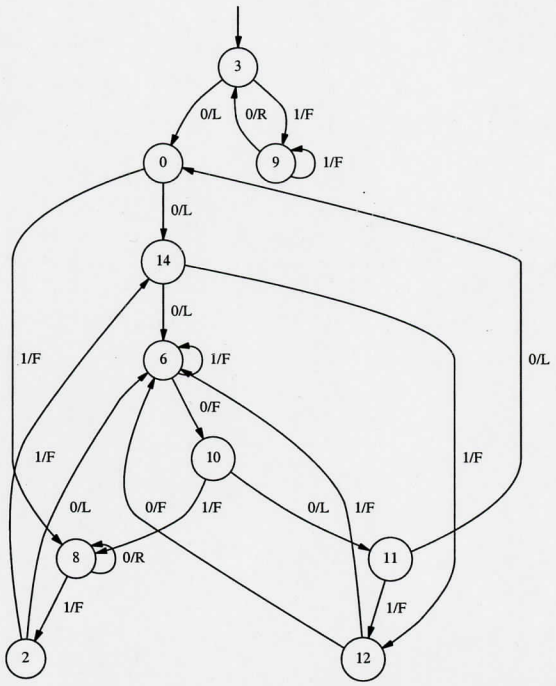


Figure 34. Champion of generation 2,000 from Problem 6a (John Muir trail).

```
. 1 1 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . 1 . . . . 1 . . . . . . . . . . . . . . . . . . . .
1 1 1 1 . . . . . . 1 . . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1
. . . 1 . . . . . . 1 . . . . . . . . . . . . . . . . . 1 . . . . . . .
. . . 1 . . . . . . 1 . . . 1 . . . . . . . . . . . . . 1 . . . . . . .
. . . 1 . . . . . . 1 . 1 . . . . . . . . . . . . . . . 1 . . . . . . .
. . . 1 . . . . . . 1 . . . . . . . . . . . . . . . . . 1 . . . . . . .
. . . 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . 1 . . . . . . .
. . . . . . . . . . . 1 . . . . . . . . . 1 1 1 1 1 . . . . . . . . . .
. . . . . . . . . . . . . . . 1 . . . . 1 . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . 1 . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . 1 . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . 1 . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . 1 . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . 1 . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . 1 . . 1 1 1 1 . 1 1 1 1 1 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Amount of food in trail: 89

03 09 09 09 09 09 09 09 09 09 03 . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . 09 . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . 09 . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . 09 . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . 09 . . . 14 12 06 10 . . . . . . . . . . . . . . . .
06 06 12 14 . . . . . . . . 09 . . . 06 . . . . . . . . . . 11 06 06 06 06 06 06 06
. . . 09 . . . . . . . . 09 . . . 06 . . . . . . . . . . 12 . . . . . . .
. . . 09 . . . . . . . . 09 . . . 12 . . . . . . . . . . 06 . . . . . . .
. . . 09 . . . . . . . . 09 14 12 06 11 . . . . . . . . . . 06 . . . . . . .
. . . 09 . . . . . . . . 09 06 . . . . . . . . . . . . . . 06 . . . . . . .
. . . 03 09 09 09 09 09 09 03 06 . . . . . . . . . . . . . 06 . . . . . . .
. . . . . . . . . . . . 12 . . . . . . . 11 06 06 06 06 12 14 . . . . . . .
. . . . . . . . . . . . 14 06 06 12 11 . . 12 . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . 06 . . 06 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . 12 . . 06 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . 14 12 06 06 11 . . 06 . . . . . . . . . . . . . . .
. . . . . . . . . . . . 06 . . . . . . 06 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . 12 . . . . . . 06 . . . . . . . . . . . . . . . .
. . . . . . . . 14 12 06 06 11 . . . . . 06 . . . . . . . . . . . . . . .
. . . . . . . . 06 . . . . . . . . . . 10 . . . . . . . . . . . . . . . .
. . . . . . . . 06 . . . . . . . . . . 08 . . . . . . . . . . . . . . . .
. . . . . . . . 12 . . . . . . . . . . 02 . . . . . . . . . . . . . . . .
. . . . . 14 12 06 11 . . . . . . . . 14 . . . . . . . . . . . . . . . .
. . . . . 06 . . . . . . . . . . . . 12 . . . . . . . . . . . . . . . .
. . . . . 12 . . . . . . . . . . . . 06 . . . . . . . . . . . . . . . .
. . . . . 14 . . . . . . . . . . . . 06 . . . . . . . . . . . . . . . .
. . . . . 02 . . . . . . . . . . . . 06 . . . . . . . . . . . . . . . .
. . . . . 08 10 06 12 14 02 08 10 06 06 06 06 06 12 14 . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Food gathered: 89
```

Figure 35. Champion of generation 2,000 from Problem 6a walking the John Muir trail.

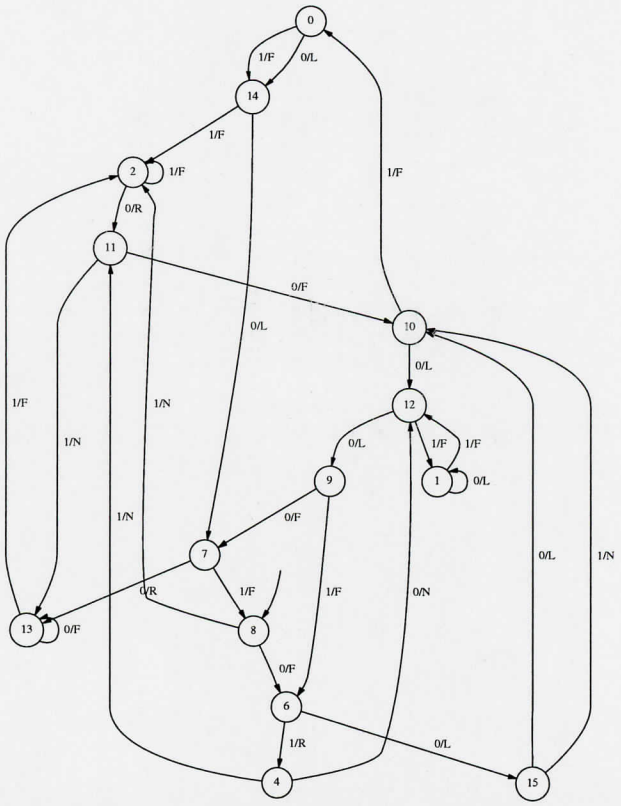


Figure 36. Champion of generation 2,000 for Problem 6b (multitrail).

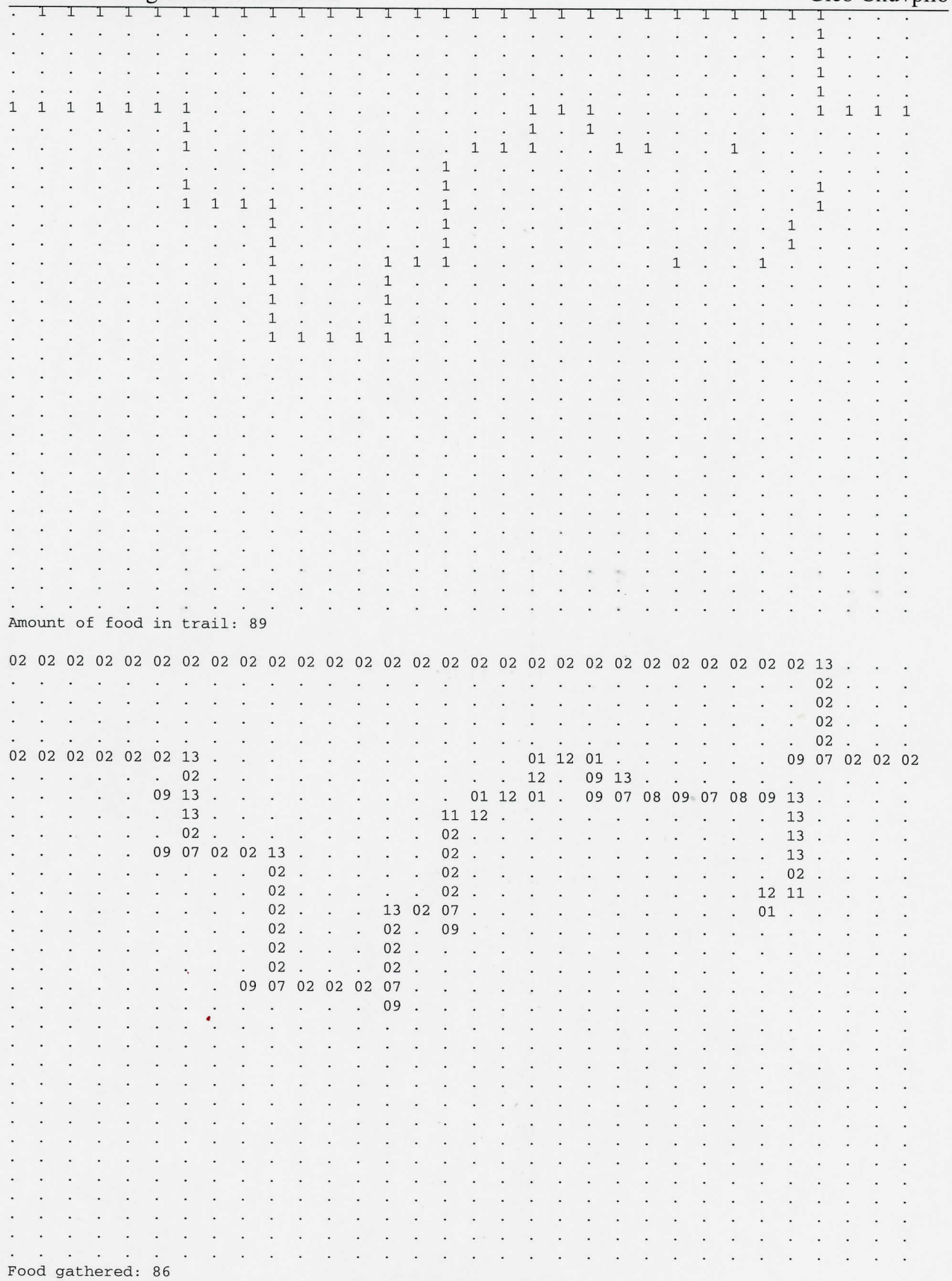


Figure 37. Trail aa000.

```
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
1 1 1 . . . . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 1 1
. . 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . .
1 . . . . . . . 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . .
1 . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . 1 .
. . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . 1 .
. . . . . . . . . . . . . 1 1 1 . . . . . . . . . . . . . . 1 .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 1 .
. . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . 1 . . . . . . . . . . . . . . 1 . . .
1 . 1 1 . . 1 . . 1 1 1 . 1 . . . . . . . . . . . . . . 1 1 1 1
. . . . . . . . . . . 1 1 . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Amount of food in trail: 89
```

```
13 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 13 . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . 02 . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . 02 . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . 09 13 . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . 13 . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . 13 . . . . . . . .
13 02 13 . . . . . . . . . . . . . . . . . . . 09 07 02 02 02 02 02 02 02 02
13 09 07 02 02 02 02 02 13 . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . 02 . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . 02 . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . 02 . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . 02 . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . 02 . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . 02 . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . 02 . . . . . . . . . . . . . . . . . . . . . . .
13 09 . . . . . 09 07 02 02 02 02 02 02 13 . . . . . . . . . . . . . . . .
14 . . . . . . . . . . . . . . 02 . . . . . . . . . . . . . . 11 10
. . . . . . . . . . . . . 09 . 02 . . . . . . . . . . . . . . 02 .
13 13 13 13 13 13 13 13 13 13 13 13 13 13 02 13 . . . . . . . . . . . . . . 13 13
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . 11 . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . 14 . . . . . . . . . . . . . . . . . .
11 10 00 13 13 13 13 13 13 02 02 13 . 00 . . . . . . . . . . . . . . . . . .
13 . . . . . 09 . . . 09 07 08 10 . . . . . . . . . . . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
13 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Food gathered: 82
```

Figure 38. Trail aa001.

```
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . 1 .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 .
1 1 1 1 . 1 1 1 1 1 1 1 1 . 1 1 1 . . . . . . . . . . . . . 1 1
. . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . 1 1 1 . 1 . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . 1 1 1 . 1 1 1 . 1 1 1 1
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . 1 1 . 1 1 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . 1 1 . . 1 . . . . . . 1 1
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 .
. . . . . . . . . . . . . . . . . . . . . . . . 1 1 . . . . 1 .
. . . . . . . . . . . . . . . . . . . . . . . . . 1 . . 1 . 1 .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 1 1 .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Amount of food in trail: 89

Printing history...
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 13 13 13 .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 09  . 02  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 02  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 02  .
02 02 02 13 13 02 02 02 02 02 02 02 13 13 02 02 13  .  .  .  .  .  .  .  .  .  .  .  . 09 07 02
 .  .  . 09  .  .  .  .  .  .  .  . 09  .  .  . 02  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 09 07 02 13 13 13  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 09 09 07 02 13 13 02 02 13 13 02 02 02 13
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 09  .  .  . 09  .  .  . 09 13
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 13
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 02
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 09 13
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 13
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 13
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 13
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 09 13
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 13  .  .  .  .  .  .  . 07 13
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 13  .  .  .  .  .  .  . 02  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 13 09 08 07 09  .  .  . 02  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 13 13 13 13  . 02  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 10  . 02  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 11 02 13  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Food gathered: 80
```

Figure 39. Trail aa002.

```
. 1 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 . . . . 1 . 1 . . . . . . .
. . . . . . . . . . . . . . . . . 1 . . . . 1 . 1 . . . . . . .
. . . . . . . . . . . . . . . . . 1 1 1 1 1 . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . 1 . . 1 1 1 1 . . . .
. . . . . . . . . . . . . . . . . . . . . 1 1 1 1 . . 1 . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . 1 1 1 . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . .
. . . . . . . . . . . . . . . . . . . . . . 1 1 1 . . 1 1 1 . .
. . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . 1
. . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . 1
. . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . 1
. . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . 1
1 . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . 1
1 . . . . . . . . . . . . . . . . . . 1 1 1 1 . . . . . . . . .
1 . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . 1
. . . . . . . . . . . . . . . . . . . 1 1 1 1 . 1 . 1 1 . 1 1 1
Amount of food in trail: 89

Printing history...
02 02 02 02 02 02 02 02 02 02 02 02 02 13 13 02 02 13 .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  09 .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  09 07 02 02 02 11 .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  10 .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  00 .  .  .  .  .  .  .  .  .  .
13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  .  09 .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  .  08 .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  09 07 02 02 07 .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  09 .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Food gathered: 31
```

Figure 40. Trail bb000.

```
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 . . . . .
1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . 1 1 1 1 1 1
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . 1 1 1 . 1 1 1 1 . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . 1 . . . 1 1 1 1 . . . . . . . . . . . . .
. . . . . . . . . . . . . . 1 . . . 1 . . . . . . . . . . . . .
. . . . . . . . . . . 1 1 1 1 . . . 1 . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . 1 1 . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . 1 1 1 1 1 1 . . . . . . . . .
. 1 . . . . . . . . . . . . . . . . . . . . 1 . . . . 1 1 1 . .
1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 1 1
Amount of food in trail: 89

Printing history...
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . 09 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . 12 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . 01 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . 09 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 13 . . . . . . . . . . . . . . . .
Food gathered: 24
```

Figure 41. Trail bb001.

```
. 1 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 . . . . . 1 . . . . .
. . . . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . 1 1 1 1 . 1 .
. . . 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . . . 1 .
. . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . 1 . . . . . . . . . . . . . . . . . 1 . . . . . .
. . . . . . . 1 . . . . . . . . . . . . . . . . 1 1 . . . . . .
. . . . . . . 1 . . . . . . . . . . . . . . . . 1 . . . . . . .
. . . . . . . 1 . . . . . . . . . . . . . . . . 1 . . . . . . .
1 1 . 1 1 1 1 1 . . . . . . . . . . . . . . . . . . . . 1 1 . 1
. . . . . . . . . . . . . . . . . . . . . . . . 1 . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . 1 . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . 1 . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . 1 . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . 1 . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . 1 . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . 1 . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . 1 . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . 1 . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . 1 1 1 . . .
Amount of food in trail: 89

Printing history...
02 02 02 02 02 02 02 02 02 02 02 02 02 13 13 02 02 02 02 02 13 .  .  .  .  .  02 .  09 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  09 .  .  .  .  .  09 07 02 02 02 02 02 07 .  .  .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  09 .  .  .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  13 .
.  .  .  09 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  13 09
13 13 13 13 02 02 02 07 09 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  13 13
.  .  .  .  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  13 .
.  .  .  .  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  13 .
.  .  .  .  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  13 .
.  .  .  .  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  13 .
.  .  .  .  .  .  .  02 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  13 .
02 13 13 02 02 02 02 07 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  13 13 13 02
.  09 .  .  .  .  .  09 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 09 13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  02 .  13 .
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  13 02 07 .  13 .
Food gathered: 71
```

Figure 42. Trail bb002.

```
/*****************************************************************************
 * Ant Farm By Gleb Chuvpilo (chuvpilo@mit.Edu)
 * 11-Apr-2002
 *
 * (Based On Genesys/Tracker System as described in the Jefferson paper)
 *****************************************************************************/

/* we need some external files */
#include <stdio.h>
#include <stdlib.h>
#include "math.h"

#define DEBUG 0

/* ---------------------------------------------------------------------------*/
/* set up some useful stuff */
#define FALSE           (0)
#define TRUE            (1)
#define max(A,B)        ((A) > (B) ? (A):(B))
#define min(A,B)        ((A) > (B) ? (B):(A))
#define ZERO            0

#ifndef RAND_MAX
#define RAND_MAX 2^^15
#endif

#define LEFT_SHIFT_2    <<2
#define LEFT_SHIFT_6    <<6
#define RIGHT_SHIFT_2   >>2
#define RIGHT_SHIFT_8   >>8
#define RIGHT_SHIFT_6   >>6



/* ant's control and trail */
#define FSA_STATES      16                  /* number of states in the FSA */
#define GENE_LENGTH     (4+12*(FSA_STATES)) /* genotype length = 4+(2*LOG_2 (FSA_STATES+4))*FSA_ST
ATES */
#define TRAIL_SIZE      32                  /* trail square side length, in cells */
#define TIME_LIMIT      200                 /* how many steps to simulate */
#define CELL_UNUSED     (FSA_STATES)        /* default value for ant's history of states on trail
*/

/* default evolution parameters */
/* default evolution parameters */
#define POPULATION      1000                /* population size */
#define DEFAULT_SEED    787                 /* init random number generator */
#define GENERATIONS     2000                /* number of generations to simulate */
#define MUTATION_RATE   0.01                /* mutation rate */
#define CROSSOVER_RATE  0.1                /* crossover rate */
#define BREED_THRESH    0.01                /* breeding threshold */

static int crossover_threshold = (int) ((1+(unsigned int)RAND_MAX)*CROSSOVER_RATE);
static int mutation_threshold =  (int) ((1+(unsigned int)RAND_MAX)*MUTATION_RATE);

/* ---------------------------------------------------------------------------*/
/* useful control constants */
/* states */
#define STATE_0  0x0
#define STATE_1  0x1
#define STATE_2  0x2
#define STATE_3  0x3
#define STATE_4  0x4
#define STATE_5  0x5
#define STATE_6  0x6
#define STATE_7  0x7
#define STATE_8  0x8
#define STATE_9  0x9
#define STATE_10 0xA
#define STATE_11 0xB
#define STATE_12 0xC
#define STATE_13 0xD
```

```
#define STATE_14 0xE
#define STATE_15 0xF

/* actions */
#define ACT_N 0x0 /* 00 = no-op */
#define ACT_R 0x1 /* 01 = right */
#define ACT_L 0x2 /* 10 = left */
#define ACT_F 0x3 /* 11 = forward */

/* directions the ant can be looking at */
#define NORTH 0x0
#define EAST  0x1
#define SOUTH 0x2
#define WEST  0x3

/* food */
#define FOOD 1

/* masks for phenotype */
#define M_PHEN_STATE_ZERO 0xF00
#define M_PHEN_ACT_ZERO   0xC0
#define M_PHEN_STATE_ONE  0x3C
#define M_PHEN_ACT_ONE    0x3



/* ------------------------------------------------------------------------*/
/* define new convenient types */
typedef char               SInt8;
typedef unsigned char      UInt8;
typedef short int          SInt16;
typedef unsigned short int UInt16;
typedef int                SInt32;
typedef unsigned int       UInt32;
typedef UInt8              boolean;


/* ------------------------------------------------------------------------*/
/* trail */
typedef struct {
  UInt32 cell[TRAIL_SIZE][TRAIL_SIZE]; /* array of cells */
  UInt32 maxfood;                      /* amount of food in the trail */
} Ttrail;


/* ------------------------------------------------------------------------*/
/* best genotypes */
typedef struct {
  SInt32 geneID; /* gene, -1 = no record */
  UInt32 score;  /* the score of that gene */
} Tbest;


/* ------------------------------------------------------------------------*/
/* history of ant's walk along the trail */
/* should be pre-initialized to CELL_UNUSED */
typedef struct {
  UInt32 cell[TRAIL_SIZE][TRAIL_SIZE]; /* array of states in cells */
  UInt32 food;                         /* amount of food the ant has gathered */
} Thistory;


/* ------------------------------------------------------------------------*/
/* ant's phenotype (FSA control) */
typedef struct {
  UInt8 start;                  /* start state */
  UInt8 state_zero[FSA_STATES]; /* new state on 0 */
  UInt8 state_one[FSA_STATES];  /* new state on 1 */
  UInt8 act_zero[FSA_STATES];   /* action on 0 */
  UInt8 act_one[FSA_STATES];    /* action on 1 */
```

```
} Tphenotype;


/* ------------------------------------------------------------------------*/
/* ant's genotype */
typedef struct {
  /* each of the first 16 codons is:

   |<----- 4 ------>|<---- 2 ---->|<----- 4 ------>|<---- 2 ---->|
   | new_state_on_0 | action_on_0 | new_state_on_1 | action_on_1 |

  aligned to the right boundary of UInt16;
  the last codon (codon[FSA_STATES]) is the start state (same right alignment)*/
  UInt16 codon[FSA_STATES+1];
} Tgenotype;


/* ------------------------------------------------------------------------*/
/* ant */
typedef struct {
  Tgenotype gene;  /* genotype */
  Tphenotype phen; /* phenotype */
} Tant;


/* ------------------------------------------------------------------------*/
/* set seed */
void set_seed (int seed) {
  srand(seed);
  return;
}


/* ------------------------------------------------------------------------*/
/* read file into trail */
void get_trail ( Ttrail *trail, char **trailfile) {
  UInt32 i, j, num, val;            /* counters, temp variables */
  FILE *fp;                         /* pointer to file */

#if DEBUG==1
  printf ("Reading trail from file '%s'...\n", *trailfile);
#endif

  fp = fopen (*trailfile, "r");

  /* exit if no file */
  if (fp==NULL) {
    printf ("File not found: %s. Exiting...\n", *trailfile);
    exit(1);
  }

  trail->maxfood = 0;

  /* read and ignore the first two numbers (dimensions) */
  num = fscanf (fp, "%d ", &val);
  num = fscanf (fp, "%d ", &val);

  /* read trail from file */
  for (i=0; i<TRAIL_SIZE; i++)
    for (j=0; j<TRAIL_SIZE; j++) {
      num = fscanf (fp, "%d ", &val);
      /* exit if value in cell is illegal */
      if (!((val==0)||(val==1))) {
        printf ("Trail entry is not 0 or 1: %d. Exiting...\n", val);
        exit(1);
      }
      trail->cell[i][j] = val;
      trail->maxfood += val;
```

```
    }

  fclose (fp);
#if DEBUG==1
  printf ("Reading trail... Done.\n");
#endif
  return;
}



/* ------------------------------------------------------------------------*/
/* print trail */
void print_trail (Ttrail *trail) {
  UInt32 i, j;          /* counters */
  UInt32 maxfood = 0;   /* amount of food in the trail */

  printf ("Printing trail...\n");

  /* print trail to screen */
  for (i=0; i<TRAIL_SIZE; i++) {
    for (j=0; j<TRAIL_SIZE; j++) {
      if (trail->cell[i][j] == FOOD)
        printf("%d  ", FOOD);
      else
        printf (".  ");
    }
    printf ("\n");
  }

  printf ("Amount of food in trail: %d\n", trail->maxfood);
  printf ("Printing trail... Done.\n");
  return;
}



/* ------------------------------------------------------------------------*/
/* reset an ant */
void reset_ant (Tant *ant) {
  UInt8 i; /* counter */

  /* reset genotype - 17 elements */
  for (i=0; i<=FSA_STATES; i++)
    ant->gene.codon[i] = 0;

  /* reset phenotype */
  ant->phen.start = 0;
  for (i=0; i<FSA_STATES; i++) {
    ant->phen.state_zero[i] = 0;
    ant->phen.state_one[i] = 0;
    ant->phen.act_zero[i] = 0;
    ant->phen.act_one[i] = 0;
  }
  return;
}



/* ------------------------------------------------------------------------*/
/* set a hand-made genotype */
void set_handmade (Tant *ant) {
#if DEBUG==1
  printf ("Transcribing hand-made phenotype...\n");
#endif

  /* reset ant */
  reset_ant (ant);

  /* set FSA states (packed new state and action for "0" and "1" transitions);*/
  /* description: go forward if see food, look around and go forward if no food; */
```

```
  /* state 0 */
  ant->gene.codon[0] =
    (((STATE_1 LEFT_SHIFT_2) + ACT_R) LEFT_SHIFT_6) + /* "0" transition */
    ((STATE_0 LEFT_SHIFT_2) + ACT_F);                 /* "1" transition */

  /* state 1 */
  ant->gene.codon[1] =
    (((STATE_2 LEFT_SHIFT_2) + ACT_R) LEFT_SHIFT_6) + /* "0" transition */
    ((STATE_0 LEFT_SHIFT_2) + ACT_F);                 /* "1" transition */

  /* state 2 */
  ant->gene.codon[2] =
    (((STATE_3 LEFT_SHIFT_2) + ACT_R) LEFT_SHIFT_6) + /* "0" transition */
    ((STATE_0 LEFT_SHIFT_2) + ACT_F);                 /* "1" transition */

  /* state 3 */
  ant->gene.codon[3] =
    (((STATE_4 LEFT_SHIFT_2) + ACT_R) LEFT_SHIFT_6) + /* "0" transition */
    ((STATE_0 LEFT_SHIFT_2) + ACT_F);                 /* "1" transition */

  /* state 4 */
  ant->gene.codon[4] =
    (((STATE_0 LEFT_SHIFT_2) + ACT_F) LEFT_SHIFT_6) + /* "0" transition */
    ((STATE_0 LEFT_SHIFT_2) + ACT_F);                 /* "1" transition */

  /* set start state - #17 */
  ant->gene.codon[FSA_STATES] = STATE_0;

#if DEBUG==1
  printf ("Transcribing hand-made phenotype... Done.\n");
#endif

  return;
}



/* ------------------------------------------------------------------------*/
/* interpret genotype to phenotype */
void interpret (Tant *ant) {
  UInt8 i; /* counter */
#if DEBUG==1
  printf ("Translating genotype to phenotype...\n");
#endif

  /* set transitions */
  for (i=0; i<FSA_STATES; i++) {
    ant->phen.state_zero[i] = (ant->gene.codon[i] & M_PHEN_STATE_ZERO) RIGHT_SHIFT_8;
    ant->phen.act_zero[i]   = (ant->gene.codon[i] & M_PHEN_ACT_ZERO) RIGHT_SHIFT_6;
    ant->phen.state_one[i]  = (ant->gene.codon[i] & M_PHEN_STATE_ONE) RIGHT_SHIFT_2;
    ant->phen.act_one[i]    = (ant->gene.codon[i] & M_PHEN_ACT_ONE);
  }

  /* set start state */
  ant->phen.start = ant->gene.codon[FSA_STATES];

#if DEBUG==1
  printf ("Translating genotype to phenotype... Done.\n");
#endif

  return;
}



/* ------------------------------------------------------------------------*/
/* print an ant's description */
void print_ant (Tant *ant) {
  UInt8 i;                  /* counter */
  char s[80];               /* for printing bit strings */
  boolean used[FSA_STATES]; /* array of "used" FSA states */
```

```
  printf ("Printing ant...\n");

  /* create array of "used" nodes of the FSA */
  /* reset all */
  for (i=0; i<FSA_STATES; i++) {
    used[i] = FALSE;
  }
  /* update for all states, except for start */
  for (i=0; i<FSA_STATES; i++) {
    used[ant->phen.state_zero[i]] = 1;
    used[ant->phen.state_one[i]] = 1;
  }
  /* update for start state */
  used[ant->phen.start] = 1;


  /* print genotype: */
  /* transitions */
  printf ("Genome (hex): ");
  for (i=0; i<FSA_STATES; i++){
    printf ("%03x ", ant->gene.codon[i]);
  }
  /* start state */
  printf ("%01x\n", ant->gene.codon[i]);

  /* print phenotype */
  printf ("Phenotype (action: 0=no-op, 1=right, 2=left, 3=forward):\n");
  printf ("Start state: [%02d]\n", ant->phen.start);
  printf ("[old]\t[input]\t[new]\t[act]\n");

  /* print only used states */
  for (i=0; i<FSA_STATES; i++) {
    if (used[i]) {
      printf (" %02d\t0\t%02d\t%d\n", i, ant->phen.state_zero[i], ant->phen.act_zero[i]);
      printf (" %02d\t1\t%02d\t%d\n", i, ant->phen.state_one[i], ant->phen.act_one[i]);
    }
  }
  printf ("Printing ant... Done.\n");
  return;
}



/* ------------------------------------------------------------------------*/
/* modulo operations (modulo = val % base) */
UInt16 modulo(SInt16 val, UInt16 base) {
  if (val < 0)
    val += base;
  val = val % base;
  return val;
}



/* ------------------------------------------------------------------------*/
/* reset history */
void reset_history (Thistory *history) {
  UInt16 i, j;
#if DEBUG==1
  printf ("Resetting history...\n");
#endif

  /* reset map */
  for (i=0; i<TRAIL_SIZE; i++) {
    for (j=0; j<TRAIL_SIZE; j++) {
      history->cell[i][j] = CELL_UNUSED;
    }
  }

  /* reset food */
  history->food = 0;
#if DEBUG==1
```

```
  printf ("Resetting history... Done.\n");
#endif
  return;
}



/* ------------------------------------------------------------------------*/
/* print ant's history to screen */
void print_history (Thistory *history) {
  UInt16 i, j;

  printf ("Printing history...\n");
  /* print map */
  for (i=0; i<TRAIL_SIZE; i++) {
    for (j=0; j<TRAIL_SIZE; j++) {
      if (history->cell[i][j] == CELL_UNUSED)
        printf (".  ");
      else
        printf ("%02d ", history->cell[i][j]);
    }
    printf ("\n");
  }

  /* print food gathered*/
  printf ("Food gathered: %d\n", history->food);
  printf ("Printing history... Done.\n");
  return;
}



/* ------------------------------------------------------------------------*/
/* run ant on the trail (simulate FSA), set ant's score */
void run_ant (Ttrail *trail, Tant *ant, Thistory *history) {
  UInt16 i;               /* counter */
  UInt16 state;           /* ant's state */
  SInt16 x, y, dir;       /* ant's state, coordinates, direction (SIGNED INT!! -- for modulo)*/
  UInt16 food;            /* food so far */
  UInt16 peek_x, peek_y;  /* coordinates of cell into which the ant is peeking */

#if DEBUG==1
  printf ("Running ant...\n");
#endif

  reset_history (history ); /* reset history first */

  /* set initial parameters, (0,0) is top left */
  state = ant->phen.start;
  dir = EAST;
  x = 0;
  y = 0;
  food = 0;

  /* simulate ant for TIME_LIMIT steps, sense-and-act loop, wrap-around the trail */
  for (i=0; i<TIME_LIMIT; i++) {
    /* record history */
    history->cell[y][x] = state;

    /* if the ant is on cell with food, consume it and update map */
    if (trail->cell[y][x] == FOOD) {
      food++;
      trail->cell[y][x] = ZERO;

      /* TEMP!!! */
      /*       if (food == 89) */
      /*        printf("STEPS FOR FULL FOOD: %d\n", i); */
    }

    /* compute the coordinates of the cell into which the ant is peeking */
    switch (dir) {
```

```
  case NORTH:
    peek_x = x;
    peek_y = modulo (y-1, TRAIL_SIZE);
    break;

  case EAST:
    peek_x = modulo (x+1, TRAIL_SIZE);
    peek_y = y;
    break;

  case SOUTH:
    peek_x = x;
    peek_y = modulo (y+1, TRAIL_SIZE);
    break;

  case WEST:
    peek_x = modulo (x-1, TRAIL_SIZE);
    peek_y = y;
    break;
  }


  /* if see food; note indexing: trail[y][x] */
  if (trail->cell[peek_y][peek_x] == FOOD) {
    /* first, need to take an action */
    switch (ant->phen.act_one[state]) {
    case ACT_N:
      /* do nothing */
      break;

    case ACT_R:
      /* turn right */
      dir = modulo (dir + 1, 4);
      break;

    case ACT_L:
      /* turn left */
      dir = modulo (dir - 1, 4);
      break;

    case ACT_F:
      /* go straight (use precomputed "peek" values) */
      x = peek_x;
      y = peek_y;
      break;
    }

    /* second, update the state of the FSA */
    state = ant->phen.state_one[state];
  }

  /* else, if no food */
  else {
    /* first, need to take an action */
    switch (ant->phen.act_zero[state]) {
    case ACT_N:
      /* do nothing */
      break;

    case ACT_R:
      /* turn right */
      dir = modulo (dir + 1, 4);
      break;

    case ACT_L:
      /* turn left */
      dir = modulo (dir - 1, 4);
      break;

    case ACT_F:
      /* go straight (use precomputed "peek" values) */
      x = peek_x;
```

```
        y = peek_y;
        break;
      }

      /* second, update the state of the FSA */
      state = ant->phen.state_zero[state];
    }


  } /* end of if-else */

  /* save value of food gathered in history */
  history->food = food;
#if DEBUG==1
  printf ("Running ant... Done.\n");
#endif
  return;
}



/* ------------------------------------------------------------------------*/
/* print the FSA of the ant to the "dot" file */
void phen2dot (Tant *ant, char **dotfile) {
  UInt32 i;                 /* counter */
  FILE *fp;                 /* pointer to file */
  char *act_str;            /* temp char to hold actions */
  boolean used[FSA_STATES]; /* array of "used" FSA states */

  printf ("Printing fsa to file '%s'...\n", *dotfile);

  /* create array of "used" nodes of the FSA */
  /* reset all */
  for (i=0; i<FSA_STATES; i++) {
    used[i] = FALSE;
  }
  /* update for all states, except for start */
  for (i=0; i<FSA_STATES; i++) {
    used[ant->phen.state_zero[i]] = 1;
    used[ant->phen.state_one[i]] = 1;
  }
  /* update for start state */
  used[ant->phen.start] = 1;


  fp = fopen (*dotfile, "w");

  /* form a dot description */
  fprintf (fp, "digraph G {\n");
  fprintf (fp, "  size = \"7.5,7.5\";\n");
  fprintf (fp, "  node [shape = circle];\n", ant->phen.start);
  for (i=0; i<FSA_STATES; i++) {

    /* zero transitions */
    switch (ant->phen.act_zero[i]) {
    case ACT_N:
      act_str = "N";
      break;

    case ACT_L:
      act_str = "L";
      break;

    case ACT_R:
      act_str = "R";
      break;

    case ACT_F:
      act_str = "F";
      break;

    }
```

```
    /* print only if used */
    if (used[i])
      fprintf (fp, "  %d -> %d [label=\"0/%s\"];\n", i, ant->phen.state_zero[i], act_str);

    /* one transitions */
    switch (ant->phen.act_one[i]) {
    case ACT_N:
      act_str = "N";
      break;

    case ACT_L:
      act_str = "L";
      break;

    case ACT_R:
      act_str = "R";
      break;

    case ACT_F:
      act_str = "F";
      break;

    }

    /* print only if used */
    if (used[i])
      fprintf (fp, "  %d -> %d [label=\"1/%s\"];\n", i, ant->phen.state_one[i], act_str);
  }

  /* draw arrow to start state */
  fprintf (fp, "  node [shape = plaintext];\n", ant->phen.start);
  fprintf (fp, "  \" \" -> %d;\n", ant->phen.start);
  fprintf (fp, "}\n");

  fclose (fp);
  printf ("Printing fsa to file '%s'... Done.\n", *dotfile);
  return;

}



/* ------------------------------------------------------------------------*/
/* do experiment with a hand-made ant */
void main_handmade () {
  /* init */
  Ttrail trail;                         /* trail */
  UInt16 generation = 0;                /* current generation */
  Tant antA;                            /* ant */
  char *trailfile = "trails/muir.txt";  /* file to open */
  char *dotfile = "dot/hand.dot";       /* file to print dot graphs to */
  Thistory history;                     /* history of states for an ant on the trail */

  /* work */
  printf ("Starting the 'handmade' experiment...\n");
  get_trail (&trail, &trailfile);       /* load trail */
  set_handmade (&antA);           /* create a hand-made ant */
  interpret (&antA);                    /* get an ant's fsa from its genome */
  print_ant (&antA);                    /* print ant to screen */
  phen2dot (&antA, &dotfile);           /* print the FSA to a dot file */
  print_trail (&trail);                 /* print trail to screen */
  run_ant (&trail, &antA, &history);    /* run ant on trail */
  print_history (&history);             /* print ant's history on that trail */
  printf ("Starting the 'handmade' experiment... Done.\n");
  return;
}



/* ------------------------------------------------------------------------*/
/* mutate each bit of the gene with a given probability */
void mutate_genome (Tgenotype *gene, float prob) {
```

```
  UInt16 i;      /* counter */
  UInt32 thresh; /* mutation threshold */
  UInt16 pos;    /* potential flip position */
  UInt16 mask;   /* mask in the form of 0x00...010...0, with "1" at pos=i */
  UInt16 temp;   /* temp variable */

  thresh = (unsigned int) ((1+(unsigned int)RAND_MAX)*prob);

  /* repeat for all codons */
  for (i=0; i<=FSA_STATES; i++) {
    mask = 0x1; /* reset mask */
    for (pos=0; pos<12; pos++) {
      mask = 0x1 << pos; /* update mask */
      /* if flip */
      if (rand()<thresh) {
        temp = gene->codon[i];
        temp = (temp & (~mask)) | ((~temp)& mask);
        gene->codon[i] = temp;
      }
    }
  }

  /* reset the unimportant part to zero*/
  gene->codon[FSA_STATES] = gene->codon[FSA_STATES] & 0xF;
  return;
}



/* ------------------------------------------------------------------------*/
/* generate random ant */
void set_random (Tgenotype *gene) {
  UInt16 i;    /* counter */

  /* reset genome - 17 elements */
  for (i=0; i<=FSA_STATES; i++) {
    gene->codon[i] = 0;
  }

  /* now, flip each bit with probability 0.5 */
  mutate_genome (gene, 0.5);
  return;
}



/* ------------------------------------------------------------------------*/
/* inject genome to ant */
void inject_genome (Tgenotype *gene, Tant *ant) {
  UInt16 i;    /* counter */

  /* copy genome to ant */
  for (i=0; i<=FSA_STATES; i++) {
    ant->gene.codon[i] = gene->codon[i];
  }
  return;
}



/* ------------------------------------------------------------------------*/
/* copy genome gene1 to gene2  */
void copy_genome (Tgenotype *gene1, Tgenotype *gene2 ) {
  UInt16 i;    /* counter */

  /* copy genome */
  for (i=0; i<=FSA_STATES; i++) {
    gene2->codon[i] = gene1->codon[i];
  }
  return;
}
```

```
/* ------------------------------------------------------------------------*/
/* conceive genome with a given crossover probability */
void conceive_genome (Tgenotype *parent1, Tgenotype *parent2, Tgenotype *child, float prob) {
  UInt16 i;       /* counter */
  UInt32 thresh; /* crossover threshold */
  UInt16 pos;     /* potential flip position */
  UInt16 mask;    /* mask in the form of 0x00...010...0, with "1" at pos=i */
  UInt8 parent;  /* who is the current parent (source of bits) */

  thresh = (unsigned int) ((1+(unsigned int)RAND_MAX)*prob);

  if ((rand() % 2) == 0)
    parent = 1;
  else
    parent = 2;

  /* repeat for all codons */
  for (i=0; i<=FSA_STATES; i++) {
    child->codon[i] = 0; /* reset child codon */
    mask = 0x1; /* reset mask */
    for (pos=0; pos<12; pos++) {
      mask = 0x1 << pos; /* update mask */
      if (parent == 1)
        child->codon[i] += (parent1->codon[i]) & mask;
      else
        child->codon[i] += (parent2->codon[i]) & mask;

      /* if change the parent */
      if (rand()<thresh) {
        if (parent == 1)
          parent = 2;
        else
          parent = 1;
      }
    }
  }
  return;
}



/* ------------------------------------------------------------------------*/
/* do evolution on the Muir trail */
void main_g0_muir () {
  /* init */
  Ttrail trail;                                    /* trail */
  Tgenotype pop_base[POPULATION];                  /* current and next population */
  Tant ant;                                        /* ant */
  UInt16 generation = 0;                           /* current generation */
  UInt32 score[POPULATION];                        /* array of scores */
  UInt32 best_score;                               /* best ant */
  SInt32 best_num;
  Thistory history;                                /* history of states for an ant on the trai
l */

  char *trailfile = "trails/muir.txt";             /* file to open */
  char *dotfile_best_g0 = "dot/muir_best_g0.dot";   /* file to print dot graphs to */
  Thistory history_best_g0;                        /* history of states for an ant on the trai
l */
  Thistory history_best_g200;                      /* history of states for an ant on the trai
l */
  UInt32 i;                                        /* counter */

  /* work */
  printf ("Starting the 'best g0' experiment...\n");
  set_seed (DEFAULT_SEED);

  printf ("Generating initial population g0... ");
  /* generate initial population */
  for (i=0; i<POPULATION; i++) {
```

```
    set_random (&pop_base[i]);                 /* create a random ant */
  }
  printf ("Done.\n");

  printf ("Evolving... ");
  /* find best ant in g0 */
  best_num = -1;
  best_score = 0;
  for (i=0; i<POPULATION; i++) {
    if ((i % 100) == 0)
      printf ("Starting i=%d...\n", i);
    get_trail (&trail, &trailfile);            /* load trail */
    inject_genome (&pop_base[i], &ant);        /* inject genome to ant */
    interpret (&ant);                          /* get an ant's fsa from its genome */
    run_ant (&trail, &ant, &history);          /* run ant on trail */
    if (history.food > best_score) {
      best_score = history.food;
      best_num =i;
      printf ("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!HERE, i= %d, score
= %d\n", i, best_score);
    }
  }

  printf (" Done.\n");

  if (best_num != -1) {
    printf ("Printing best ant: number %d, score %d...\n", best_num, best_score);
    get_trail (&trail, &trailfile);                    /* load trail */
    print_trail (&trail);                              /* print trail to screen */
    inject_genome (&pop_base[best_num], &ant);         /* inject genome to ant */
    interpret (&ant);                                  /* get an ant's fsa from its genome */
    print_ant (&ant);                                  /* print ant to screen */
    phen2dot (&ant, &dotfile_best_g0);                 /* print the FSA to a dot file */
    run_ant (&trail, &ant, &history);                  /* run ant on trail */
    print_history (&history);                          /* print ant's history on that trail */
  }

  printf ("Printing best ant... Done.\n");
  printf ("Starting the 'best g0' experiment... Done.\n");
  return;
}


/* ----------------------------------------------------------------------------*/
/* do evolution on the Muir trail */
void main_evolve_muir () {
  /* main stuff */
  Ttrail trail;                                                /* trail */
  Tgenotype pop_base[POPULATION], pop_child[POPULATION]; /* base and children populations */
  Tant ant;                                                    /* ant */
  UInt16 generation = 0;                                       /* current generation */
  UInt32 score[POPULATION];                                    /* array of scores */
  Thistory history;                                            /* history of states for an ant on the t
rail */
  UInt32 parent1, parent2;                                     /* parents */

  char *trailfile = "trails/muir.txt";                         /* file to open */
  char *dotfile_best_g0 = "dot/muir_best_g0.dot";              /* file to print dot graphs to */
  char *dotfile_best_g_last = "dot/muir_best_g_last.dot";/* file to print dot graphs to */

  UInt32 i, j, k;                                              /* counters */

  /* procreation */
  Tbest best[POPULATION];                                      /* best genotypes */
  UInt32 bestMax;                                              /* pointer to the last element of the ar
ray */

  /* statistics */
  char *statfile = "plot/stat.dat";                           /* file to print statistics to */
  FILE *fp;                                                    /* pointer to file */
  float mean, stdev;                                           /* stats */
```

```
  UInt32 modes[1000];                                   /* modes */
  UInt32 maxMode;                                       /* max mode used */
  UInt32 mostMode;                                      /* most frequently used mode */
  UInt32 allScores[POPULATION];                         /* array to compute st. dev. */
  UInt32 max;                                           /* food gathered by the best-scoring ant
 */

  /* work */
  printf ("Starting the 'evolve muir' experiment...\n");
  set_seed (DEFAULT_SEED);
  bestMax = (unsigned int) (BREED_THRESH * POPULATION)+1;

  printf ("Generating initial population g0... ");
  /* generate initial population */
  for (i=0; i<POPULATION; i++) {
    set_random (&pop_base[i]);          /* create a random ant */
  }
  printf ("Done.\n");

  printf ("Evolving...\n");

  fp = fopen (statfile, "w");
  fprintf (fp, "# Evolution statistics\n");
  printf ("# Evolution statistics\n");
  fprintf (fp, "# gen \t\tmean \t\t\tstdev \t\t\tmax \t\tmode1\n");
  printf ("# gen \t\tmean \t\t\tstdev \t\t\tmax \t\tmode1\n");

  get_trail (&trail, &trailfile);                   /* load trail */

  /* TODO: NEEDS TO BE CHANGED IF EVOLVE ON MULTIPLE FILES !!! */
  maxMode = trail.maxfood;

  /* evolve population and find the best ant in the last generation */
  for (generation=0; generation<GENERATIONS; generation++) {
/*     printf ("Generation: %d\n", generation); */

    /* reset stats */
    mean = 0;
    stdev = 0;
    max = 0;
    mostMode = 0;

    /* reset modes */
    for (i=0; i<=maxMode; i++)
      modes[i] = 0;

    /* reset "best genomes" set */
    for (i=0; i<bestMax; i++) {
      best[i].geneID = -1;
      best[i].score = 0;
    }

    for (i=0; i<POPULATION; i++) {
      get_trail (&trail, &trailfile);       /* load trail */
      inject_genome (&pop_base[i], &ant);   /* inject genome to ant */
      interpret (&ant);                     /* get an ant's fsa from its genome */
      run_ant (&trail, &ant, &history);     /* run ant on trail */

      /* update stats */
      mean += history.food;
      allScores[i] = history.food;
      modes[history.food]++;

      /* update the "best genomes" list */
      for (j=0; j<bestMax; j++) {
        /* if found better, shift old right and exit the "while" loop*/
        if (history.food > best[j].score) {
          /* shift right */
          for (k=bestMax; k>j; k--) {
            best[k].geneID = best[k-1].geneID;
          }
          /* insert */
```

```
          best[j].geneID = i;
          best[j].score = history.food;
          break;
        }
      }
    }

    /* compute stats */
    max = best[0].score;
    mean = mean/POPULATION;

    for (j=0; j<POPULATION; j++) {
      stdev += pow((allScores[j]-mean),2);
    }
    stdev = sqrt(stdev/POPULATION);

    mostMode = 0;

    /* find most frequently used mode */
    for (j=0; j<=maxMode; j++) {
      if (modes[j]>modes[mostMode]) {
        mostMode = j;
      }
    }
/*     printf ("mostMode=%u, repeated %u times\n", mostMode, modes[mostMode]); */


    fprintf (fp, "  %u \t\t%f \t\t%f \t\t%u \t\t%u\n", generation, mean, stdev, max, mostMode);
    printf ("  %u \t\t%f \t\t%f \t\t%u \t\t%u\n", generation, mean, stdev, max, mostMode);

/*     printf ("\tFitness: %d\n", best[0].score); */

    /* select random parents and let them procreate */
    for (j=0; j<POPULATION; j++) {
      parent1 = best[rand() % bestMax].geneID;
      parent1 = best[rand() % bestMax].geneID;
      conceive_genome (&pop_base[parent1], &pop_base[parent2], &pop_child[j], CROSSOVER_RATE);

      /* now, flip each bit with a given probability */
      mutate_genome (&pop_child[j], MUTATION_RATE);
    }

    /* copy children back into the base population */
    if (generation < (GENERATIONS-1)) {
      for (j=0; j<POPULATION; j++) {
        copy_genome (&pop_child[j], &pop_base[j]);
      }
    }
  }

  /* close stats file */
  fclose (fp);

  printf ("Evolving... Done\n");

  /* print best ant */
  printf ("Printing best ant in generation %d: geneID=%d, score=%d\n", (generation-1), pop_base[b
est[0].geneID], pop_base[best[0].score] );
  get_trail (&trail, &trailfile);                    /* load trail */
  print_trail (&trail);                              /* print trail to screen */
  inject_genome (&pop_base[best[0].geneID], &ant); /* inject genome to ant */
  interpret (&ant);                                  /* get an ant's fsa from its genome */
  print_ant (&ant);                                  /* print ant to screen */
  phen2dot (&ant, &dotfile_best_g_last);             /* print the FSA to a dot file */
  run_ant (&trail, &ant, &history);                  /* run ant on trail */
  print_history (&history);                          /* print ant's history on that trail */

  printf ("Printing best ant... Done.\n");
  printf ("Starting the 'evolve muir' experiment... Done.\n");
  return;
}
```

```
/* ------------------------------------------------------------------------*/
/* do evolution on multiple trails */
void main_evolve_multitrail () {
  /* main stuff */
  Ttrail trail;                                          /* trail */
  Tgenotype pop_base[POPULATION], pop_child[POPULATION]; /* base and children populations */
  Tant ant;                                              /* ant */
  UInt16 generation = 0;                                 /* current generation */
  UInt32 score[POPULATION];                              /* array of scores */
  Thistory history;                                      /* history of states for an ant on the t
rail */
  UInt32 parent1, parent2;                               /* parents */

  char *trailfile = "trails/muir.txt";                   /* file to open */

  char *dotfile_best_g0 = "dot/muir_best_g0.dot";        /* file to print dot graphs to */
  char *dotfile_best_g_last = "dot/muir_best_g_last.dot";/* file to print dot graphs to */

  UInt32 i, j, k;                                        /* counters */

  /* procreation */
  Tbest best[POPULATION];                                /* best genotypes */
  UInt32 bestMax;                                        /* pointer to the last element of the ar
ray */

  /* statistics */
  char *statfile = "plot/stat.dat";                      /* file to print statistics to */
  FILE *fp;                                              /* pointer to file */
  float mean, stdev;                                     /* stats */
  UInt32 modes[1000];                                    /* modes */
  UInt32 maxMode;                                        /* max mode used */
  UInt32 mostMode;                                       /* most frequently used mode */
  UInt32 allScores[POPULATION];                          /* array to compute st. dev. */
  UInt32 max;                                            /* food gathered by the best-scoring ant
 */

  /* multitrail stuff */
  UInt32 nTrails=3;                                      /* the number of trails */
  char *studyTrails[3] = {"trails/aa000.txt",            /* study files to open */
                          "trails/aa001.txt",
                          "trails/aa002.txt"};

  char *testTrails[3] = {"trails/bb000.txt",             /* test files to open */
                         "trails/bb001.txt",
                         "trails/bb002.txt"};
  UInt32 maxFitness;                                     /* max fitness = sum of all trails */
  UInt32 curTrail;                                       /* counter into current trail */
  UInt32 globalFitness;                                  /* ant's global fitness on all trails */
  float ratio;                                           /* ratio of fitnesses */


  /* work */
  printf ("Starting the 'evolve multitrail' experiment...\n");
  set_seed (DEFAULT_SEED);
  bestMax = (unsigned int) (BREED_THRESH * POPULATION)+1;

  printf ("Generating initial population g0... ");
  /* generate initial population */
  for (i=0; i<POPULATION; i++) {
    set_random (&pop_base[i]);          /* create a random ant */
  }
  printf ("Done.\n");

  printf ("Evolving...\n");

  fp = fopen (statfile, "w");
  fprintf (fp, "# Evolution statistics\n");
  printf ("# Evolution statistics\n");


  maxFitness = 0;
```

```
/* compute max fitness */
for (curTrail=0; curTrail<nTrails; curTrail++) {
  trailfile = studyTrails[curTrail];
  get_trail (&trail, &trailfile);                    /* load trail */
  maxFitness += trail.maxfood;
}

printf ("maxFitness=%d\n", maxFitness);
fprintf (fp, "# gen \t\tmaxStudy \tmaxTest\n");
printf      ("# gen \t\tmaxStudy \tmaxTest\n");


maxMode = maxFitness; /* mode 1 */

/* evolve population and find the best ant in the last generation */
for (generation=0; generation<GENERATIONS; generation++) {

  /* reset stats */
  mean = 0;
  stdev = 0;
  max = 0;
  mostMode = 0;

  /* reset modes */
  for (i=0; i<=maxMode; i++)
    modes[i] = 0;

  /* reset "best genomes" set */
  for (i=0; i<bestMax; i++) {
    best[i].geneID = -1;
    best[i].score = 0;
  }


  for (i=0; i<POPULATION; i++) {
    globalFitness = 0;
    for (curTrail=0; curTrail<nTrails; curTrail++) {
      trailfile = studyTrails[curTrail];     /* pick current trail */
      get_trail (&trail, &trailfile);        /* load trail */
      inject_genome (&pop_base[i], &ant);    /* inject genome to ant */
      interpret (&ant);                      /* get an ant's fsa from its genome */
      run_ant (&trail, &ant, &history);      /* run ant on trail */
      globalFitness += history.food;         /* update ant's global fitness */
    }

    /* update stats */
    mean += globalFitness;
    allScores[i] = globalFitness;
    modes[globalFitness]++;

    /* update the "best genomes" list */
    for (j=0; j<bestMax; j++) {
      /* if found better, shift old right and exit the "while" loop*/
      if (globalFitness > best[j].score) {
        /* shift right */
        for (k=bestMax; k>j; k--) {
          best[k].geneID = best[k-1].geneID;
        }
        /* insert */
        best[j].geneID = i;
        best[j].score = globalFitness;
        break;
      }
    }
  }

  /* compute stats */
  max = best[0].score;
  mean = mean/POPULATION;

  for (j=0; j<POPULATION; j++) {
    stdev += pow((allScores[j]-mean),2);
```

```
    }
    stdev = sqrt(stdev/POPULATION);

    mostMode = 0;

    /* find most frequently used mode */
    for (j=0; j<=maxMode; j++) {
      if (modes[j]>modes[mostMode]) {
        mostMode = j;
      }
    }


    /* select random parents and let them procreate */
    for (j=0; j<POPULATION; j++) {
      parent1 = best[rand() % bestMax].geneID;
      parent1 = best[rand() % bestMax].geneID;
      conceive_genome (&pop_base[parent1], &pop_base[parent2], &pop_child[j], CROSSOVER_RATE);

      /* now, flip each bit with a given probability */
      mutate_genome (&pop_child[j], MUTATION_RATE);
    }

    /* copy children back into the base population */
    if (generation < (GENERATIONS-1)) {
      for (j=0; j<POPULATION; j++) {
        copy_genome (&pop_child[j], &pop_base[j]);
      }
    }

    /* run best ant on test trails and print statistics */
    globalFitness = 0;
    for (curTrail=0; curTrail<nTrails; curTrail++) {
      trailfile = testTrails[curTrail];                /* pick current trail */
      get_trail (&trail, &trailfile);                  /* load trail */
      inject_genome (&pop_base[best[0].geneID], &ant); /* inject genome to ant */
      interpret (&ant);                                /* get an ant's fsa from its genome */
      run_ant (&trail, &ant, &history);                /* run ant on trail */
      globalFitness += history.food;                   /* update ant's global fitness */
    }

    fprintf (fp, "  %u \t\t%u \t\t%u\n", generation, max, globalFitness);
    printf      ("  %u \t\t%u \t\t%u\n", generation, max, globalFitness);

  }

  /* close stats file */
  fclose (fp);

  printf ("Evolving... Done\n");


  /* EVALUATE ANT */
  printf ("Printing best ant in generation %d: geneID=%d, score=%d\n", (generation-1), pop_base[b
est[0].geneID], pop_base[best[0].score] );
  inject_genome (&pop_base[best[0].geneID], &ant); /* inject genome to ant */
  interpret (&ant);                                /* get an ant's fsa from its genome */
  print_ant (&ant);                                /* print ant to screen */
  phen2dot (&ant, &dotfile_best_g_last);           /* print the FSA to a dot file */


  /* study trails */
  printf ("-------STUDY TRAILS-------\n");
  maxFitness = 0;
  /* compute max fitness on study trails */
  for (curTrail=0; curTrail<nTrails; curTrail++) {
    trailfile = studyTrails[curTrail];
    get_trail (&trail, &trailfile);                /* load trail */
    maxFitness += trail.maxfood;
  }

  globalFitness = 0;
```

```
  for (curTrail=0; curTrail<nTrails; curTrail++) {
    trailfile = studyTrails[curTrail];                /* pick current trail */
    get_trail (&trail, &trailfile);                   /* load trail */
    print_trail (&trail);                             /* print trail to screen */
    inject_genome (&pop_base[best[0].geneID], &ant); /* inject genome to ant */
    interpret (&ant);                                 /* get an ant's fsa from its genome */
    run_ant (&trail, &ant, &history);                 /* run ant on trail */
    print_history (&history);                         /* print ant's history on that trail */
    globalFitness += history.food;                    /* update ant's global fitness */
  }
  ratio = (float) globalFitness/maxFitness;
  printf ("Global fitness on STUDY trails is %d out of %d (%f\%)\n", globalFitness, maxFitness, r
atio*100);


  /* test trails */
  printf ("-------TEST TRAILS-------\n");
  maxFitness = 0;
  /* compute max fitness on test trails */
  for (curTrail=0; curTrail<nTrails; curTrail++) {
    trailfile = testTrails[curTrail];
    get_trail (&trail, &trailfile);                   /* load trail */
    maxFitness += trail.maxfood;
  }

  globalFitness = 0;
  for (curTrail=0; curTrail<nTrails; curTrail++) {
    trailfile = testTrails[curTrail];                 /* pick current trail */
    get_trail (&trail, &trailfile);                   /* load trail */
    print_trail (&trail);                             /* print trail to screen */
    inject_genome (&pop_base[best[0].geneID], &ant); /* inject genome to ant */
    interpret (&ant);                                 /* get an ant's fsa from its genome */
    run_ant (&trail, &ant, &history);                 /* run ant on trail */
    print_history (&history);                         /* print ant's history on that trail */
    globalFitness += history.food;                    /* update ant's global fitness */
  }
  ratio = (float) globalFitness/maxFitness;
  printf ("Global fitness on TEST trails is %d out of %d (%f\%)\n", globalFitness, maxFitness, ra
tio*100);

  printf ("Printing best ant... Done.\n");
  printf ("Starting the 'evolve multitrail' experiment... Done.\n");
  return;
}


/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
int main (void) {
  /* init */

  /* work */
  printf ("XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX\n");
  printf ("Welcome to Ant Farm!\n");
  main_evolve_multitrail();
  printf ("Exiting Ant Farm...\n");
  return 0;
}
```

1 10/10

2 3/3

3 ~~3/3~~ 3/3

4 2/3

5 5/5

6a 10/10

6b 10/10

6c 8/8

7 14/14

65/66