of humans to intuitively grasp time scales that are much greater than their own lifetimes. Another reason is that it is very difficult to appreciate how random changes can lead to emergent complex structures. Genetic algorithms provide one way of understanding the nature of evolution. Their principal utility at present is in optimization problems, but they are also being used to model biological and social evolution.

Historically, developments in physics, such as x-ray crystallography and quantum mechanics, have lead to developments in biology. In recent years developments in biology as well as in computer science and other areas have had a direct impact on developments in physics. Genetic algorithms are an example of the influence of ideas in biology impacting ideas in physics.

The idea of genetic algorithms is to model the process of evolution by natural selection. This process involves two steps: random changes in the genetic code during reproduction and selection according to some fitness criteria. In biological organisms the genetic code is stored in the DNA. We will store the genetic code as a string of 1s and 0s. The genetic code constitutes the *genotype*. The conversion of this string to the organism or *phenotype* depends on the problem. The selection criteria is applied to the phenotype.

First we describe how change is introduced into the genotype. Typically, nature changes the genetic code in two ways. The most obvious, but less often used method, is mutation. Mutation corresponds to changing a character at random in the genetic code string from 0 to 1 or from 1 to 0. The other much more powerful method is associated with sexual reproduction. We take two strings, remove a piece from one string, and exchange it with the same length piece from the other string. For example, if string $A = 0011001010$ and string $B = 0001110001$, then exchanging the piece from position 4 to position 7 leads to two new strings, $A' = 0011110010$ and $B' = 0001001001$. This type of change is called recombination or crossover.

At each generation we produce changes using recombination and mutation. We then select from the enlarged population of strings (including strings from the previous generation) a new population for the next generation. Usually, a constant population size is maintained from one generation of strings to the next.

We next have to choose a selection criterion. If we want to model an actual ecosystem, we can include a physical environment and other sets of populations corresponding to different species. The fitness could depend on the interaction of the different species with one another, the interaction within each species, and the interaction with the physical environment. In addition, the behavior of the populations might change the environment from one generation to the next. For simplicity, we will consider only a single population of strings, a simple phenotype, and a simple criteria for fitness.

The phenotype we consider is a variant of the Ising model considered in Problem 14.14. We consider a square lattice of linear dimension L occupied by $N = L^2$ spins that have the values $s_i = \pm 1$. The energy of the system is given by

$$E = -\sum_{i,j=nn(i)} J_{ij} s_i s_j, \tag{14.7}$$

where the sum is over all pairs of spins that are nearest neighbors. The energy function in (14.7) assumes that only nearest neighbor spins interact, in contrast to the energy function in (14.6) which assumes that every spin interacts with every other spin. The coupling constants J_{ij} are either $+1$, -1, or distributed according to some probability distribution. If

we assume that $|J_{ij}| = 1$, then the minimum energy equals $-2N$ and the maximum energy is $2N$. Because we want the fitness to be positive, we choose $2N - E$ as the measure of fitness and take the probability of selecting a particular string with energy E for the next generation to be proportional to the fitness $2N - E$.

How does a genotype become "expressed" as a phenotype? A genotype consists of a string of length N with 1s and 0s. The lattice site (i, j) corresponds to the nth position in the string, where $n = jL + i$. If the character in the string at position n is 0, then the spin at site (i, j) equals -1. If the character is 1, then the spin equals $+1$. Note that in this case, the representation of the genotype is very similar to that of the phenotype. In particular, they have the same size N, and each "piece" can have only two values. In general, the expression of the genotype in the phenotype is much more complicated. Usually, a sequence within the genotype corresponds to one value in the phenotype, which in biological systems is related to the coding for a specific protein. Such a sequence is what we call a gene.

We now have all the ingredients we need to apply the genetic algorithm. The `GeneticApp` class obtains the various parameters, initializes the population of genotypes, and calls the various methods needed to evolve the gene pool (see the `doStep` method). The `GenePool` class carries out the evolution. In method `recombine` two genotypes are chosen at random, and a random piece of one is exchanged for the equivalent piece of the other. In method `mutate` a random position in a randomly selected genotype is changed. We use a boolean array to represent the genotype, so that a change represents converting `true` to `false` or vice versa. In both methods we do not replace the original genotype but instead add a new genotype to the population. The `Phenotype` class determines the fitness of each member of the population by computing the energy of the lattice of spins corresponding to each member of the population. Members of this population are selected for the new generation by generating a discrete nonuniform probability distribution as discussed in Section 11.5.

Listing 14.10 The `GeneticApp` class.

```
package org.opensourcephysics.sip.ch14.genetic;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.frames.*;

public class GeneticApp extends AbstractSimulation {
   GenePool genePool = new GenePool();
   Phenotype phenotype = new Phenotype();
   DisplayFrame frame = new DisplayFrame("Gene pool");

   public void initialize() {
      phenotype.L = control.getInt("Lattice size");
      genePool.populationNumber = control.getInt("Population size");
      genePool.recombinationRate =
            control.getInt("Recombination rate");
      genePool.mutationRate = control.getInt("Mutation rate");
      genePool.genotypeSize = phenotype.L*phenotype.L;
      genePool.initialize(phenotype);
      phenotype.initialize();
      frame.addDrawable(genePool);
      frame.setPreferredMinMax(-1.0, genePool.genotypeSize+5, -1.0,
            genePool.populationNumber+2);
      frame.setSize(phenotype.L*phenotype.L*10,
            genePool.populationNumber*20);
   }
```