Problem 14.19 Ground state of Ising-like models

(a) Use the genetic algorithm we have discussed to find the ground state of the ferromagnetic Ising model for which $J_{ij} = 1$. In this case the ground state energy is $E = -2L^2$ (all spins up or all spins down). It will be necessary to modify method `Initialize` in class `Phenotype`. Choose $L = 4$ and consider a population of 20 strings, with 10 recombinations and 4 mutations per generation. How long does it take to find the ground state energy? You might wish to modify the program to show each new generation on the screen.

(b) Find the mean number of generations needed to find the ground state for $L = 4$, 6, and 8. Repeat each run several times. Use a population of 100, a recombination rate of 50, and a mutation rate of 20. Are there any general trends as L is increased? How do your results change if you double the population size? What happens if you double the recombination rate or mutation rate? Use larger lattices if you have sufficient computer resources.

(c) Repeat part (b) for the antiferromagnetic model for which $J_{ij} = -1$.

(d) Repeat part (b) for a spin glass for which $J_{ij} = \pm 1$ at random. In this case we do not know the ground state energy in advance. What criterion can you use to terminate a run? ■

One of the important features of the genetic algorithm is that the change in the genetic code is selected not in the genotype directly, but in the phenotype. Note that the way we change the strings (particularly with recombination) is not closely related to the two-dimensional lattice of spins. We could have used some other prescription for converting a string of 0s and 1s to a configuration of spins on a two-dimensional lattice. If the phenotype is a three-dimensional lattice, we could use the same procedure for modifying the genotype, but a different prescription for converting the genetic sequence (the string of 0s and 1s) to the phenotype (the three-dimensional lattice of spins). The point is that it is not necessary for the genetic coding to mimic the phenotypic expression. This point becomes distorted in the popular press when a gene is tied to a particular trait, because specific pieces of DNA rarely correspond directly to any explicitly expressed trait in the phenotype.

14.6 ■ LATTICE GAS MODELS OF FLUID FLOW

We now return to cellular automaton models and discuss one of their more interesting applications—simulations of fluid flow. In general, fluid flow is very difficult to simulate because the partial differential equation describing the flow of incompressible fluids, the Navier–Stokes equation, is nonlinear, and this nonlinearity can lead to the failure of standard numerical algorithms. In addition, there are typically many length scales that must be considered simultaneously. These length scales include the microscopic motion of the fluid particles, the length scales associated with fluid structures such as vortices, and the length scales of macroscopic objects such as pipes or obstacles. Because of these considerations, simulations of fluid flow based on the direct numerical solutions of the Navier–Stokes equation typically require very sophisticated numerical methods (cf. Oran and Boris).

Cellular automaton models of fluids are known as *lattice gas* models. In a lattice gas model the positions of the particles are restricted to the sites of a lattice, and the velocities

Table 14.1 Summary of the possible velocities and their representations.

Velocity Vector	Direction	Symbol	Abbreviation	Decimal	Binary
$\mathbf{v}_0$	$(1, 0)$	`RIGHT`	`RI`	1	00000001
$\mathbf{v}_1$	$(1, -\sqrt{3})/2$	`RIGHT_DOWN`	`RD`	1	00000010
$\mathbf{v}_2$	$-(1, \sqrt{3})/2$	`LEFT_DOWN`	`LD`	4	00000100
$\mathbf{v}_3$	$(-1, 0)$	`LEFT`	`LE`	8	00001000
$\mathbf{v}_4$	$(-1, \sqrt{3})/2$	`LEFT_UP`	`LU`	16	00010000
$\mathbf{v}_5$	$(1, \sqrt{3})/2$	`RIGHT_UP`	`RU`	32	00100000
$\mathbf{v}_6$	$(0, 0)$	`STATIONARY`	`S`	64	01000000
		`BARRIER`		128	10000000

are restricted to a small number of vectors corresponding to neighbor sites. A time step is divided into two substeps. In the first substep the particles move freely to their corresponding nearest neighbor lattice sites. Then the velocities of the particles at each lattice site are changed according to a collision rule that conserves mass (particle number), momentum, and kinetic energy. The purpose of the collision rules is not to accurately model microscopic collisions, but rather to achieve the correct macroscopic behavior. The idea is that if we satisfy the conservation laws associated with microscopic collisions, then we can find the correct physics at the macroscopic level, including translational and rotational invariance, by averaging over many particles.

We assume a triangular lattice, because it can be shown that this symmetry is sufficient to yield the macroscopic Navier–Stokes equations for a continuum. In contrast, the more limited symmetry of a square lattice is not sufficient. Three-dimensional models are much more difficult to implement and justify theoretically.

All the moving particles are assumed to have the same speed and mass. The possible velocity vectors lie only in the direction of the nearest neighbor sites, and hence there are six possible velocities as summarized in Table 14.1. A rest particle is also allowed. The number of particles at each site moving in a particular direction (channel) is restricted to be zero or one.

In the first substep all particles move in the direction of their velocity to a neighboring site. In the second substep the velocity vectors at each lattice site are changed according to the appropriate collision rule. Examples of the collision rules are illustrated in Figures 14.4–14.6. The rules are deterministic with only one possible set of velocities after a collision for each possible set of velocities before a collision. It is easy to check that momentum conservation for collisions between the particles is enforced by these rules.

As in Section 14.1, we use bit manipulation to efficiently represent a lattice site and the collision rules. Each lattice site is represented by one element of the integer array `lattice`. In Java each `int` stores 32 bits, but we will use only the first 8 bits. We use the first six bits from 0 to 5 to represent particles moving in the six possible directions with bit 0 corresponding to a particle moving with velocity $\mathbf{v}_0$ (see Table 14.1). If there are three particles with velocities $\mathbf{v}_0$, $\mathbf{v}_2$, and $\mathbf{v}_4$ at a site and no barrier, then the value of the lattice array element at this site is 00010101 in binary notation.

Bit 6 represents a possible rest (stationary) particle. If we want a site to act as a barrier that blocks incoming particles, we set bit 7. For example, a barrier site containing a particle with velocity $\mathbf{v}_1$ is represented by 10000010.