

The success of hill climbing depends very much on the shape of the state-space landscape: if there are few local maxima and plateaux, random-restart hill climbing will find a good solution very quickly. On the other hand, many real problems have a landscape that looks more like a widely scattered family of balding porcupines on a flat floor, with miniature porcupines living on the tip of each porcupine needle, *ad infinitum*. NP-hard problems typically have an exponential number of local maxima to get stuck on. Despite this, a reasonably good local maximum can often be found after a small number of restarts.

4.1.2 Simulated annealing

SIMULATED
ANNEALING

GRADIENT DESCENT

A hill-climbing algorithm that *never* makes “downhill” moves toward states with lower value (or higher cost) is guaranteed to be incomplete, because it can get stuck on a local maximum. In contrast, a purely random walk—that is, moving to a successor chosen uniformly at random from the set of successors—is complete but extremely inefficient. Therefore, it seems reasonable to try to combine hill climbing with a random walk in some way that yields both efficiency and completeness. **Simulated annealing** is such an algorithm. In metallurgy, **annealing** is the process used to temper or harden metals and glass by heating them to a high temperature and then gradually cooling them, thus allowing the material to reach a low-energy crystalline state. To explain simulated annealing, we switch our point of view from hill climbing to **gradient descent** (i.e., minimizing cost) and imagine the task of getting a ping-pong ball into the deepest crevice in a bumpy surface. If we just let the ball roll, it will come to rest at a local minimum. If we shake the surface, we can bounce the ball out of the local minimum. The trick is to shake just hard enough to bounce the ball out of local minima but not hard enough to dislodge it from the global minimum. The simulated-annealing solution is to start by shaking hard (i.e., at a high temperature) and then gradually reduce the intensity of the shaking (i.e., lower the temperature).

The innermost loop of the simulated-annealing algorithm (Figure 4.5) is quite similar to hill climbing. Instead of picking the *best* move, however, it picks a *random* move. If the move improves the situation, it is always accepted. Otherwise, the algorithm accepts the move with some probability less than 1. The probability decreases exponentially with the “badness” of the move—the amount ΔE by which the evaluation is worsened. The probability also decreases as the “temperature” T goes down: “bad” moves are more likely to be allowed at the start when T is high, and they become more unlikely as T decreases. If the *schedule* lowers T slowly enough, the algorithm will find a global optimum with probability approaching 1.

Simulated annealing was first used extensively to solve VLSI layout problems in the early 1980s. It has been applied widely to factory scheduling and other large-scale optimization tasks. In Exercise 4.4, you are asked to compare its performance to that of random-restart hill climbing on the 8-queens puzzle.

4.1.3 Local beam search

LOCAL BEAM
SEARCH

Keeping just one node in memory might seem to be an extreme reaction to the problem of memory limitations. The **local beam search** algorithm³ keeps track of k states rather than

³ Local beam search is an adaptation of **beam search**, which is a path-based algorithm.

```

function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
           schedule, a mapping from time to “temperature”

  current  $\leftarrow$  MAKE-NODE(problem.INITIAL-STATE)
  for  $t = 1$  to  $\infty$  do
     $T \leftarrow$  schedule( $t$ )
    if  $T = 0$  then return current
    next  $\leftarrow$  a randomly selected successor of current
     $\Delta E \leftarrow$  next.VALUE – current.VALUE
    if  $\Delta E > 0$  then current  $\leftarrow$  next
    else current  $\leftarrow$  next only with probability  $e^{\Delta E/T}$ 

```

Figure 4.5 The simulated annealing algorithm, a version of stochastic hill climbing where some downhill moves are allowed. Downhill moves are accepted readily early in the annealing schedule and then less often as time goes on. The *schedule* input determines the value of the temperature T as a function of time.

just one. It begins with k randomly generated states. At each step, all the successors of all k states are generated. If any one is a goal, the algorithm halts. Otherwise, it selects the k best successors from the complete list and repeats.

At first sight, a local beam search with k states might seem to be nothing more than running k random restarts in parallel instead of in sequence. In fact, the two algorithms are quite different. In a random-restart search, each search process runs independently of the others. In a *local beam search*, useful information is passed among the parallel search threads. In effect, the states that generate the best successors say to the others, “Come over here, the grass is greener!” The algorithm quickly abandons unfruitful searches and moves its resources to where the most progress is being made.

In its simplest form, local beam search can suffer from a lack of diversity among the k states—they can quickly become concentrated in a small region of the state space, making the search little more than an expensive version of hill climbing. A variant called **stochastic beam search**, analogous to stochastic hill climbing, helps alleviate this problem. Instead of choosing the best k from the the pool of candidate successors, stochastic beam search chooses k successors at random, with the probability of choosing a given successor being an increasing function of its value. Stochastic beam search bears some resemblance to the process of natural selection, whereby the “successors” (offspring) of a “state” (organism) populate the next generation according to its “value” (fitness).

4.1.4 Genetic algorithms

A **genetic algorithm** (or **GA**) is a variant of stochastic beam search in which successor states are generated by combining *two* parent states rather than by modifying a single state. The analogy to natural selection is the same as in stochastic beam search, except that now we are dealing with sexual rather than asexual reproduction.



STOCHASTIC BEAM
SEARCH

GENETIC
ALGORITHM