```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  current ← problem.INITIAL
  for t = 1 to ∞ do
    T ← schedule(t)
    if T = 0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE(current) – VALUE(next)
    if ΔE > 0 then current ← next
    else current ← next only with probability e^(ΔE/T)
```

Figure 4.5 The simulated annealing algorithm, a version of stochastic hill climbing where some downhill moves are allowed. The *schedule* input determines the value of the "temperature" T as a function of time.

all the probability is concentrated on the global maxima, which the algorithm will find with probability approaching 1.

Simulated annealing was used to solve VLSI layout problems beginning in the 1980s. It has been applied widely to factory scheduling and other large-scale optimization tasks.

4.1.3 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** algorithm keeps track of k states rather than 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.

Local beam search

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.

Local beam search can suffer from a lack of diversity among the k states—they can become clustered in a small region of the state space, making the search little more than a k-times-slower version of hill climbing. A variant called **stochastic beam search**, analogous to stochastic hill climbing, helps alleviate this problem. Instead of choosing the top k successors, stochastic beam search chooses successors with probability proportional to the successor's value, thus increasing diversity.

Stochastic beam search

4.1.4 Evolutionary algorithms

Evolutionary algorithms can be seen as variants of stochastic beam search that are explicitly motivated by the metaphor of natural selection in biology: there is a population of individuals (states), in which the fittest (highest value) individuals produce offspring (successor states) that populate the next generation, a process called **recombination**. There are endless forms of evolutionary algorithms, varying in the following ways:

Evolutionary algorithms

Recombination