

populate the next generation according to its “value” (fitness).

4.1.4 Genetic algorithms

GENETIC ALGORITHM

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.

POPULATION INDIVIDUAL

Like beam search, GAs begin with a set of k randomly generated states, called the **population**. Each state, or **individual**, is represented as a string over a finite alphabet—most commonly, a string of 0s and 1s. For example, an 8-queens state must specify the positions of 8 queens, each in a column of 8 squares, and so requires $8 \times \log_2 8 = 24$ bits. Alternatively, the state could be represented as 8 digits, each in the range from 1 to 8. (We will see later that the two encodings behave differently.) Figure 4.6(a) shows a population of four 8-digit strings representing 8-queens states.

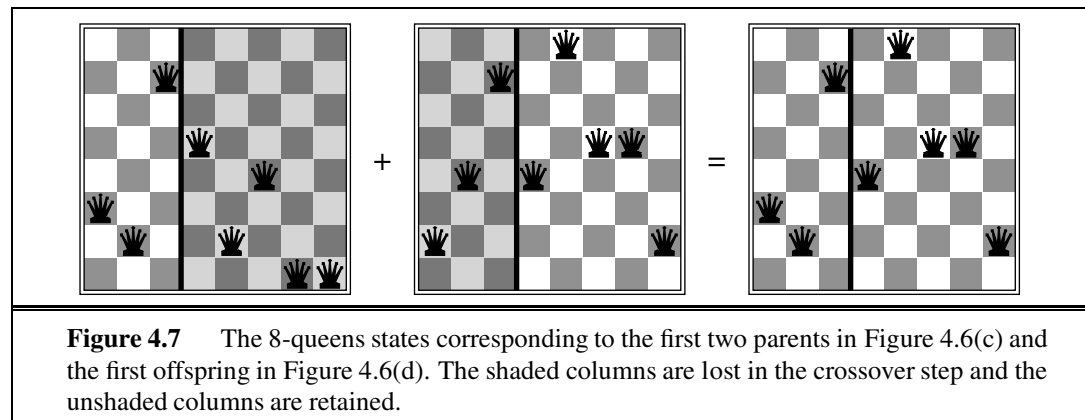
FITNESS FUNCTION

The production of the next generation of states is shown in Figure 4.6(b)–(e). In (b), each state is rated by the objective function or (in GA terminology) the **fitness function**. A fitness function should return higher values for better states, so, for the 8-queens problem we use the number of *nonattacking* pairs of queens, which has a value of 28 for a solution. The values of the four states are 24, 23, 20, and 11. In this particular variant of the genetic algorithm, the probability of being chosen for reproducing is directly proportional to the fitness score, and the percentages are shown next to the raw scores.

CROSSOVER

In (c), two pairs are selected at random for reproduction, in accordance with the probabilities in (b). Notice that one individual is selected twice and one not at all.⁴ For each pair to be mated, a **crossover** point is chosen randomly from the positions in the string. In Figure 4.6, the crossover points are after the third digit in the first pair and after the fifth digit

⁴ There are many variants of this selection rule. The method of **culling**, in which all individuals below a given threshold are discarded, can be shown to converge faster than the random version (Baum *et al.*, 1995).



in the second pair.⁵

In (d), the offspring themselves are created by crossing over the parent strings at the crossover point. For example, the first child of the first pair gets the first three digits from the first parent and the remaining digits from the second parent, whereas the second child gets the first three digits from the second parent and the rest from the first parent. The 8-queens states involved in this reproduction step are shown in Figure 4.7. The example illustrates the fact that, when two parent states are quite different, the crossover operation can produce a state that is a long way from either parent state. It is often the case that the population is quite diverse early on in the process, so crossover (like simulated annealing) frequently takes large steps in the state space early in the search process and smaller steps later on when most individuals are quite similar.

MUTATION

Finally, in (e), each location is subject to random **mutation** with a small independent probability. One digit was mutated in the first, third, and fourth offspring. In the 8-queens problem, this corresponds to choosing a queen at random and moving it to a random square in its column. Figure 4.8 describes an algorithm that implements all these steps.

Like stochastic beam search, genetic algorithms combine an uphill tendency with random exploration and exchange of information among parallel search threads. The primary advantage, if any, of genetic algorithms comes from the crossover operation. Yet it can be shown mathematically that, if the positions of the genetic code are permuted initially in a random order, crossover conveys no advantage. Intuitively, the advantage comes from the ability of crossover to combine large blocks of letters that have evolved independently to perform useful functions, thus raising the level of granularity at which the search operates. For example, it could be that putting the first three queens in positions 2, 4, and 6 (where they do not attack each other) constitutes a useful block that can be combined with other blocks to construct a solution.

SCHEMA

The theory of genetic algorithms explains how this works using the idea of a **schema**, which is a substring in which some of the positions can be left unspecified. For example, the schema 246***** describes all 8-queens states in which the first three queens are in

⁵ It is here that the encoding matters. If a 24-bit encoding is used instead of 8 digits, then the crossover point has a 2/3 chance of being in the middle of a digit, which results in an essentially arbitrary mutation of that digit.

function GENETIC-ALGORITHM(*population*, FITNESS-FN) **returns** an individual

inputs: *population*, a set of individuals

FITNESS-FN, a function that measures the fitness of an individual

repeat

new_population \leftarrow empty set

for $i = 1$ **to** SIZE(*population*) **do**

$x \leftarrow$ RANDOM-SELECTION(*population*, FITNESS-FN)

$y \leftarrow$ RANDOM-SELECTION(*population*, FITNESS-FN)

child \leftarrow REPRODUCE(x, y)

if (small random probability) **then** *child* \leftarrow MUTATE(*child*)

add *child* to *new_population*

population \leftarrow *new_population*

until some individual is fit enough, or enough time has elapsed

return the best individual in *population*, according to FITNESS-FN

function REPRODUCE(x, y) **returns** an individual

inputs: x, y , parent individuals

$n \leftarrow$ LENGTH(x); $c \leftarrow$ random number from 1 to n

return APPEND(SUBSTRING($x, 1, c$), SUBSTRING($y, c + 1, n$))

Figure 4.8 A genetic algorithm. The algorithm is the same as the one diagrammed in Figure 4.6, with one variation: in this more popular version, each mating of two parents produces only one offspring, not two.

INSTANCE

positions 2, 4, and 6 respectively. Strings that match the schema (such as 24613578) are called **instances** of the schema. It can be shown that, if the average fitness of the instances of a schema is above the mean, then the number of instances of the schema within the population will grow over time. Clearly, this effect is unlikely to be significant if adjacent bits are totally unrelated to each other, because then there will be few contiguous blocks that provide a consistent benefit. Genetic algorithms work best when schemata correspond to meaningful components of a solution. For example, if the string is a representation of an antenna, then the schemata may represent components of the antenna, such as reflectors and deflectors. A good component is likely to be good in a variety of different designs. This suggests that successful use of genetic algorithms requires careful engineering of the representation.

In practice, genetic algorithms have had a widespread impact on optimization problems, such as circuit layout and job-shop scheduling. At present, it is not clear whether the appeal of genetic algorithms arises from their performance or from their aesthetically pleasing origins in the theory of evolution. Much work remains to be done to identify the conditions under which genetic algorithms perform well.