What is GA?

What are Genetic Algorithms?

Genetic Algorithms (GAs) are search algorithms based on mechanics of natural selection and natural genetics.

GAs combine survival of the fittest among string structures with a structured yet randomized information exchange.

In every generation a new set of strings is created using bits and pieces of the old.

GAs efficiently exploit historical information to speculate on new search points.

Developed by John Holland (1975)

Robustness of Search Methods

Conventional methods are not robust

Calculus-based methods

- use of local gradient (hill-climbing)
- local in scope
- depends on the existence of derivatives

Enumerative schemes

- simple and attractive
- lack of efficiency
- dynamic programming

Random search methods

- in the long run not better than the enumerative schemes
- lack of efficiency

How Are GAs Different From Traditional Methods?

GAs work with coding of the parameter set

GAs search from a population of points, not a single point

GAs use payoff information, not derivatives or other knowledge

GAs use probabilistic transition rules, not deterministic rules

A Simple Genetic Algorithm

Three operators

- Reproduction
- Crossover
- Mutation

Reproduction

Individual strings are copied according to their fitness values

Parents are selected with probability biased toward chromosomes with better evaluations

Reproduction

Fitness-proportional reproduction

Roulette wheel selection

Expected value selection (stochastic sampling without replacement)

Remainder stochastic sampling without replacement

Rank-proportional reproduction

Tournament

Sharing/Deterministic crowding

Expected value model (R3)

Designed to reduce the stochastic errors of roulette wheel selection.

Compute the expected number of offspring for each string (f/f_{avg}):

Each time a string is selected for mating and crossover, its offspring count is decreased by 0.5.

When an individual string is selected for reproduction without mating and crossover, its offspring count is decreased by 1.0.

In either case an individual whose offspring count fell below zero is no longer available for selection.

In this model the actual number of offspring is generally less than $f/f_{avg} + 0.5$.

R3 outperforms R1 and R2 in both on-line and off-line performance measures over the environment E (functions F1-F5).

Crossover

Children in the mating pools are mated at random and each pair of strings are crossed over

Crossover causes exchange of genetic material between two parents

Greatly accelerates search early in the evolution of a population

It is common in recent GA applications to use either two-point crossover or parameterized uniform crossover with $p_c \approx 0.7\text{-}0.8$

Mutation

Causes local alteration in a single chromosome (secondary role)

Restores lost information to the population

Empirically one mutation per thousand bit transfers

Genetic Algorithms by Hand

max f(x) = x^2 subject to x ∈ [0, 1]

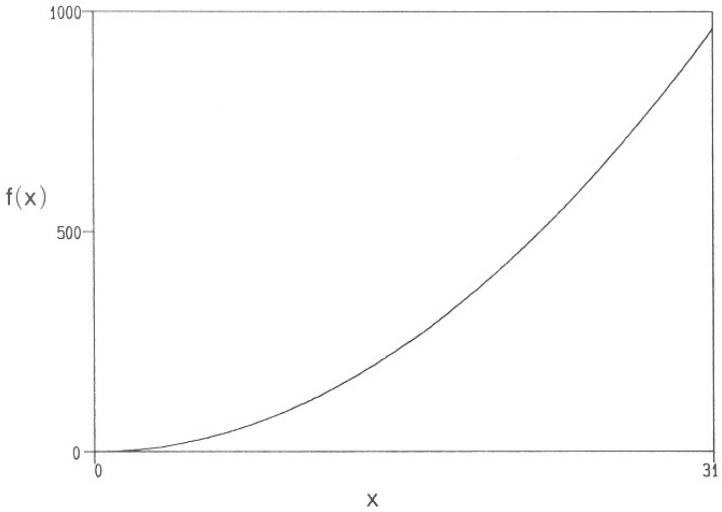

FIGURE 1.5 A simple function optimization example, the function $f(x) = x^2$ on the integer interval [0, 31].

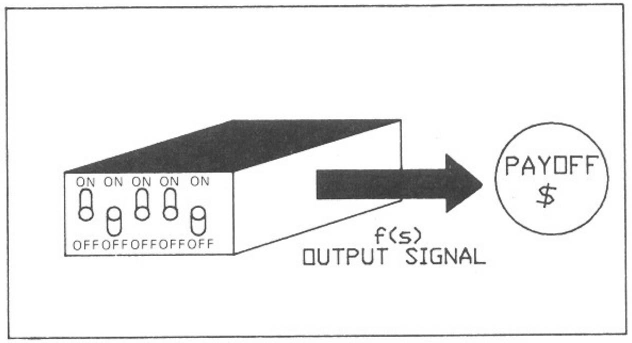

FIGURE 1.6 A black box optimization problem with five on-off switches illustrates the idea of a coding and a payoff measure. Genetic algorithms only require these two things: they don't need to know the workings of the black box.

TABLE 1.1 Sample Problem Strings and Fitness Values

No.	String	Fitness	% of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

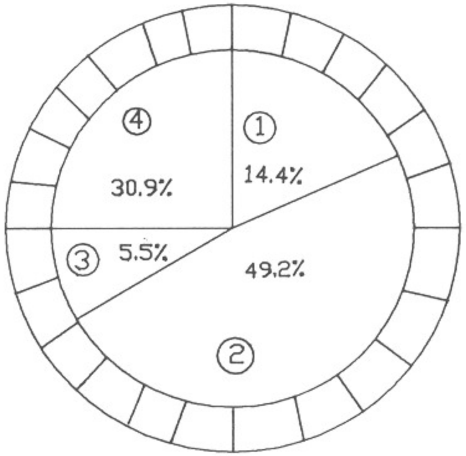

FIGURE 1.7 Simple reproduction allocates offspring strings using a roulette wheel with slots sized according to fitness. The sample wheel is sized for the problem of Tables 1.1 and 1.2.

TABLE 1.2 A Genetic Algorithm by Hand

String No.	Initial Population (Randomly Generated)	x Value (Unsigned Integer)	$f(x)$ x^2	pselect$_i$ $\frac{f_i}{\Sigma f}$	Expected count $\frac{f_i}{\bar{f}}$	Actual Count from Roulette Wheel
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4.0
Average			293	0.25	1.00	1.0
Max			576	0.49	1.97	2.0

TABLE 1.2 (*Continued*)

Mating Pool after Reproduction (Cross Site Shown)	Mate (Randomly Selected)	Crossover Site (Randomly Selected)	New Population	x Value	$f(x)$ x^2
0 1 1 0 | 1	2	4	0 1 1 0 0	12	144
1 1 0 0 | 0	1	4	1 1 0 0 1	25	625
1 1 | 0 0 0	4	2	1 1 0 1 1	27	729
1 0 | 0 1 1	3	2	1 0 0 0 0	16	256
					1754
					439
					729

Similarity Templates (Schemata)

What information is contained in a population of strings and their objective function values to help guide a directed search for improvement?

Similarities among strings in the population

Relationships between the similarities and high fitness

A *schema* is a similarity template describing a subset of strings with similarities at certain string positions

The Schema *111* describes a subset with four members

{01110, 01111, 11110, 11111}

Similarity Templates (Schemata)

Total number of possible schemata: $(k+1)^l = 3^5$

k: number of alphabet

l: string length

Total number of strings: $k^l = 2^5$

Why make matters more difficult by enlarging the space of concern?

When we consider the strings, their fitness values, and the similarities among the strings in the population, we admit a wealth of new information to help direct our search.

Similarity Templates (Schemata)

In general, a particular string contains (is a member of) 2^l schemata of length l

The upper bound on the total number of schemata in a population of size n is $n2^l$

Effect of operators on a particular schemata:

1. Reproduction: More highly fit strings have higher probabilities of selection
2. Crossover: Schemata of short defining length survive
3. Mutation at low rates does not disrupt a particular schema very frequently

Similarity Templates (Schemata)

Highly fit, short defining length schemata (***building block***) are propagated generation to generation by giving exponentially increasing samples to the observed best - ***Implicit Parallelism***