

in moving away from local optima and diversifying the search space. Typically, a rate of 0.02 to 0.05 is used for mutation. Higher rates are used in problem-specific scenarios. The popular stopping criteria is a lack of solution improvement and the average fitness of population not improving over a predetermined number of generations. The decision maker can observe the entire population of the final generation and choose the solution with the best fitness value or the one that provides the most intangibles. In contrast to SA, GA works on multiple solutions simultaneously.

Problem-specific descriptions of these steps are described in the following discussion, using a product-line pricing and positioning example.

EXAMPLE 2

Product-Line Selection and Pricing with Genetic Algorithms

Acme Widget Company (AWC) has the possibility of offering five substitutable products for 10 customer segments. Each segment is assumed to be of size 20. For each segment, AWC has estimated the segment preference for the products. This information is given in Table 5. The variable cost per unit of manufacturing the product and the fixed-cost production are given in Table 6.

TABLE 5

Segment/Prod	1	2	3	4	5
1	184	149	126	148	187
2	198	182	125	167	152
3	155	132	192	165	167
4	141	197	133	173	182
5	136	128	117	200	192
6	172	152	171	182	133
7	184	167	138	119	142
8	123	156	163	176	159
9	114	191	179	168	126
10	110	184	139	149	185

TABLE 6

Fixed cost	710	1,203	665	1,224	1,391
Variable cost	50	55	56	69	58

The segment surplus is preference minus the price of the product. A segment will choose a product with highest segment surplus if it is not negative. If the highest surplus for a segment is negative, then the segment will buy nothing. AWC is faced with the task of identifying what products to offer and at what price.

Solution

The model is a nonlinear integer programming problem and can be classified as NP-hard. We will use a genetic approach to solve this problem. The first step is to determine a suitable encoding. The decision variables for the GA are whether a product is offered or not and, if offered, at what price. The offer-or-not variables are encoded as a binary string with five members. Each element with a 1 means the corresponding product is offered. Each element with a 0 means the product is not offered. The price is a continuous variable and can be encoded using a real representation. There will be five continuous variables for each product (each variable represents the price charged for a particular product). Because

the minimum variable cost is 50 and the maximum preference is 200, each price will be between 50 and 200. A representation of the string is as follows:

String 1 0 1 1 1 0 121 128 93 200 51

String 1 implies that products 2, 3, and 4 are offered, and the prices are 121, 128, 93, 200, and 51 for the five products. Another example string follows:

String 2 1 1 0 0 1 92 116 87 96 95

A single point *crossover* will pick a position in a set of two strings (call it position p) to split the strings into two. Then, assuming there are k alleles in each string, we combine the split strings as follows to form two new offspring:

New string 1 Left p alleles from string 1 followed by right $k - p$ alleles from string 2.

New string 2 Left p alleles from string 2 followed by right $k - p$ alleles from string 1.

For example, if we choose $p = 3$, applying single point crossover to strings 1 and 2 will result in two new offspring as follows:

Original strings

0	1	1		1	0	121	128	93	200	51
1	1	0		0	1	92	116	87	96	95

New strings

0	1	1	0	1	92	116	87	96	95
1	1	0	1	0	121	128	93	200	51

Another alternative is to use a two-point crossover. Suppose we choose the two crossover positions at the second and eighth positions. Then we create two new strings by swapping the first two and last two alleles in each string. If we apply this two-point crossover to strings 1 and 2, we obtain the following strings:

Result of two-point crossover on strings 1 and 2

1	1	1	1	0	121	128	93	96	95
0	1	0	0	1	92	116	87	200	51

A mutation can pick one of the positions in the string and then can modify the contents. If the position is between 1 and 5, then we change the content from 0 to 1 or 1 to 0. If the mutation position is 3 in the last string, then the string will be modified as follows:

0 1 1 0 1 92 116 87 200 51

If the mutation position is between 6 and 10, then the content can be modified by an amount that can be randomly determined to be positive or negative. For example, we can alter the price by, say, 20. Let us specify that position 8 from the previous string is chosen for mutation and the sign is positive. Then the string will be modified as follows:

0 1 1 0 1 92 116 107 200 51

Once the price and the product offerings are set, the consumer surplus for each segment can be calculated by subtracting price from preference and multiplying it by whether or not a product is offered. The maximum surplus can then be calculated; if it is positive, then the segment will be assigned to the corresponding product. Segment size multiplied by the product margin less the fixed cost of offered products would provide the evaluation or fitness profit value.

Let us assume a population of 20 for each generation, a mutation rate of 0.1, and a single-point crossover after position 2. We will also define a crossover rate that specifies what percentage of the population can be considered for crossover. We choose a crossover rate of 80%.

Step 1 Generation. Set the number of generations to 50. Population = 20, mutation = 0.1, and crossover rate = 0.8. Suppose our randomly chosen population of 20 strings is as shown in Table 7.

TABLE 7

0	0	0	1	1	91	108	100	116	118
1	1	1	0	0	128	87	110	114	83
1	1	0	1	1	135	67	114	98	95
1	1	1	0	1	76	108	77	126	135
0	1	1	0	0	94	100	105	81	62
0	1	1	0	1	115	104	97	74	106
1	1	1	0	0	110	91	88	86	73
0	1	0	0	1	103	140	80	131	111
1	1	0	0	1	97	98	99	93	103
1	1	0	0	1	60	81	94	89	118
1	1	0	1	0	88	103	74	76	120
1	1	0	0	1	126	105	98	79	107
1	0	0	1	1	111	94	105	112	110
0	1	0	1	0	100	114	93	117	100
1	0	1	0	0	97	112	85	104	119
1	1	0	0	0	98	100	89	85	136
1	0	1	0	0	135	103	86	87	71
1	0	1	1	0	54	56	138	94	127
1	1	1	0	0	105	135	93	93	98
1	1	0	1	0	115	87	78	105	127

Step 2 Evaluation. To determine the fitness of each string, we determine which segments will buy each produced product (given our product prices). Note that if the maximum consumer surplus is non-negative, then a segment buys the offered product yielding the maximum consumer surplus. If a segment's maximum surplus is negative, then the segment makes no purchases. Now our profit is computed by

$$\sum_{\text{all segments}} (\text{Size of segment}) * (\text{price of product bought} - \text{variable cost of product bought}) - \sum_{\text{all products offered}} (\text{product fixed cost}).$$

For example, for the string shown in Table 8, we find that eight segments purchase product 5 and two segments purchase product 2; thus, we find the profit of \$9,286 as follows:

$$20(2) * (140 - 55) + 20(8) * (111 - 58) - 1203 - 1391 = \$9,286.$$

TABLE 8

P1	P2	P3	P4	P5	Price					Profit (\$)
					1	2	3	4	5	
0	1	0	0	1	103	140	80	131	111	9,286

Then we can determine the sorted strings in descending order-of-fitness values as shown in Table 9.

TABLE 9

P1	P2	P3	P4	P5	Price					Profit (\$)
					1	2	3	4	5	
0	1	0	0	1	103	140	80	131	111	9,286
0	1	0	1	0	100	114	93	117	100	8,493
1	1	1	0	0	105	135	93	93	98	7,982
0	0	0	1	1	91	108	100	116	118	7,825
1	1	0	0	0	98	100	89	85	136	7,747
0	1	1	0	0	94	100	105	81	62	7,372
1	0	1	0	0	135	103	86	87	71	6,825
1	0	0	1	1	111	94	105	112	110	6,715
1	1	0	0	1	126	105	98	79	107	6,616
0	1	1	0	1	115	104	97	74	106	6,021
1	0	1	0	0	97	112	85	104	119	5,865
1	1	0	0	1	97	98	99	93	103	5,696
1	1	1	0	0	110	91	88	86	73	4,862
1	1	1	0	0	128	87	110	114	83	4,262
1	1	0	1	0	115	87	78	105	127	4,243
1	1	1	0	1	76	108	77	126	135	2,011
1	1	0	1	0	88	103	74	76	120	943
1	1	0	0	1	60	81	94	89	118	-24
1	0	1	1	0	54	56	138	94	127	-539
1	1	0	1	1	135	67	114	98	95	-788

Step 3 With a crossover rate at 0.8, the first 16 strings will be chosen to be eligible parents. The 16 strings' associated probabilities of selection and cumulative probabilities are as shown in Table 10.

For example, the probability of choosing string 1 equals

$$9,286/(\text{total fitness of all strings}) = 9,286/101,821 = 9.12\%.$$

The cumulative fitness percentage for the second string is $9.12 + 8.34 = 17.46\%$.

Steps 4 and 5 We now use crossovers and mutations to produce a new generation. Because the "fitter" strings have more chance of being used for crossover, we would expect the next generation to have improved fitness. This is the way GAs implement the biological concept of survival of the fittest.

We now illustrate how to create the strings for the next generation. We begin by generating two random numbers that will be used to choose the two strings used in the first crossover. Suppose random number 1 = 0.070826 and random number 2 = 0.245279. Any random number less than or equal to .0912 yields string 1; any random number greater than .09 and less than .1746 yields string 2; and so on. This ensures that the prob-

TABLE 10

P1	P2	P3	P4	P5	Price					Profit (\$)	Fitness (%)	Cumulative Fitness (%)
					1	2	3	4	5			
0	1	0	0	1	103	140	80	131	111	9,286	9.12	9.12
0	1	0	1	0	100	114	93	117	100	8,493	8.34	17.46
1	1	1	0	0	105	135	93	93	98	7,982	7.84	25.30
0	0	0	1	1	91	108	100	116	118	7,825	7.69	32.99
1	1	0	0	0	98	100	89	85	136	7,747	7.61	40.59
0	1	1	0	0	94	100	105	81	62	7,372	7.24	47.83
1	0	1	0	0	135	103	86	87	71	6,825	6.70	54.54
1	0	0	1	1	111	94	105	112	110	6,715	6.59	61.13
1	1	0	0	1	126	105	98	79	107	6,616	6.50	67.63
0	1	1	0	1	115	104	97	74	106	6,021	5.91	73.54
1	0	1	0	0	97	112	85	104	119	5,865	5.76	79.30
1	1	0	0	1	97	98	99	93	103	5,696	5.59	84.90
1	1	1	0	0	110	91	88	86	73	4,862	4.78	89.67
1	1	1	0	0	128	87	110	114	83	4,262	4.19	93.86
1	1	0	1	0	115	87	78	105	127	4,243	4.17	98.02
1	1	1	0	1	76	108	77	126	135	2,011	1.98	100.00

ability of choosing a string is proportional to the string's fitness. Based on our two random numbers, the original selected strings are 1 and 3. The strings and crossover are:

0	1		0	0	1	103	140	80	131	111
0	0		0	1	1	91	108	100	116	118

With the crossover operation (after position 2), the new strings are:

0	1	0	1	1	91	108	100	116	118
0	0	0	0	1	103	140	80	131	111

Suppose a mutation for string 2 occurs only in position 3. The strings after mutation are:

0	1	0	1	1	91	108	100	116	
0	0	1	0	1	103	140	80	131	

In a similar fashion, we would perform nine more reproductive operations to generate 18 more strings to create a new population of 20 strings for the next generation.

Step 6 Perform steps 2 through 5 for 50 generations and declare the best solution found so far as the heuristic solution.

After 50 generations we found the following solution:

1 1 1 1 0 135 67 114 98 95

Profit for this solution is \$21,118. Note that the randomness involved in selecting strings for crossover and for implementing mutation implies that two different people performing the same number of GA iterations may obtain different solutions!