MA2000: OTML

Nachiketa Mishra

Indian Institute of Information Technology, Design & Manufacturing, Kancheepuram

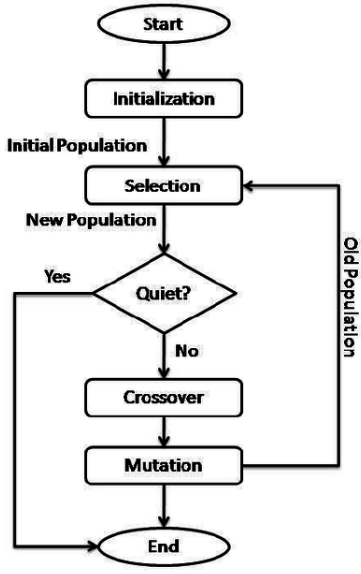
Genetic Algorithm

Motivation: Genetic Algorithms have the ability to deliver a "good-enough" solution "fast-enough". This makes genetic algorithms attractive for use in solving optimization problems.

NP-Hard: even the most powerful computing systems take a very long time (even years!) to solve that problem. In such a scenario, GAs prove to be an efficient tool to provide usable near-optimal solutions in a short amount of time.

- Since genetic algorithms are designed to simulate a biological process, much of the relevant terminology is borrowed from biology.
- The basic components common to almost all genetic algorithms are:
  1. a fitness function for optimization
  2. a population of chromosomes
  3. selection of which chromosomes will reproduce
  4. crossover to produce next generation of chromosomes
  5. random mutation of chromosomes in new generation

Flowchart of GA algorithm

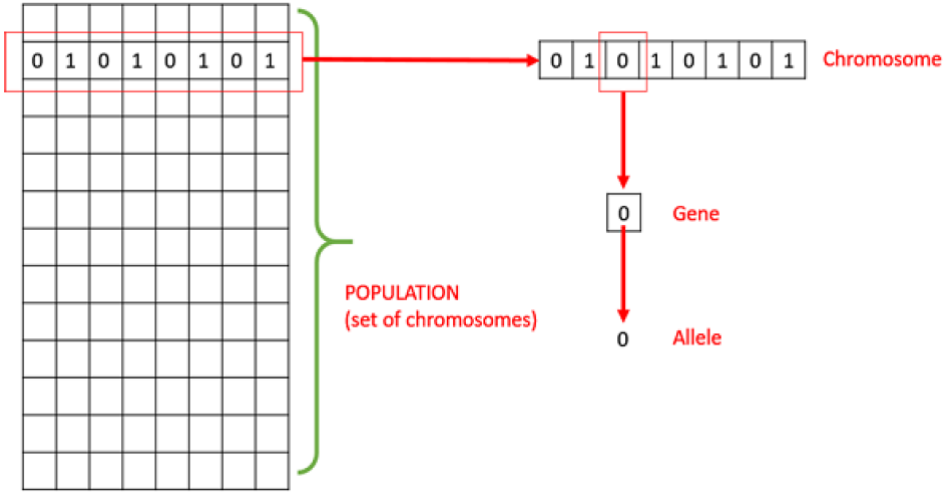


Genetic Algorithm

- **Fitness Function** is the function that the algorithm is trying to optimize. The word "fitness" is taken from evolutionary theory.
- **Chromosome** refers to a numerical value or values that represent a candidate solution to the problem that the genetic algorithm is trying to solve. Each candidate solution is encoded as an array of parameter values.
  - If a problem has N_{par} dimensions, then typically each chromosome is encoded as an N_{par}-element array

    $$\text{Chromosome} = [p_1, p_2, \ldots, P_{N_{par}}]$$

    where each p_i is a particular value of the ith parameter
  - One approach is to convert each parameter value into a bit string (sequence of 1's and 0's),


0 1 0 1 0 1 0 1
0 1 0 1 0 1 0 1
Chromosome
0
Gene
0
Allele
POPULATION
(set of chromosomes)

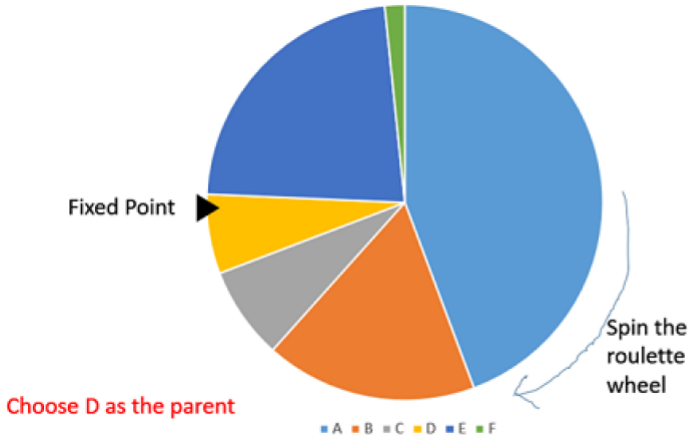
- A genetic algorithm begins with a randomly chosen assortment of chromosomes, which serves as the first generation (initial population)
- Then each chromosome in the population is evaluated by the fitness function to test how well it solves the problem.
- **Selection operator** chooses some of the chromosomes for reproduction based on a probability distribution defined by the user.
  - The fitter a chromosome is, the more likely it is to be selected.
  - For example, if f is a non-negative fitness function, then the probability that chromosome C_{53} is chosen to reproduce might be

$$P(C_{53}) = \left| \frac{f(C_{53})}{\sum_{i=1}^{N_{pop}} f(C_i)} \right|$$

  - How to select parent ? In this every individual can become a parent with a probability which is proportional to its fitness. Therefore, fitter individuals have a higher chance of mating and propagating their features to the next generation.
  - Therefore, such a selection strategy applies a selection pressure to the more fit individuals in the population, evolving better individuals over time.

Roulette Wheel Selection

Consider a circular wheel. The wheel is divided into n pies, where n is the number of individuals in the population. A fixed point is chosen on the wheel circumference as shown and the wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent. For the second parent, the same process is repeated.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

- It is clear that a fitter individual has a greater pie on the wheel and therefore a greater chance of landing in front of the fixed point when the wheel is rotated.
- Therefore, the probability of choosing an individual depends directly on its fitness.

Crossover operator

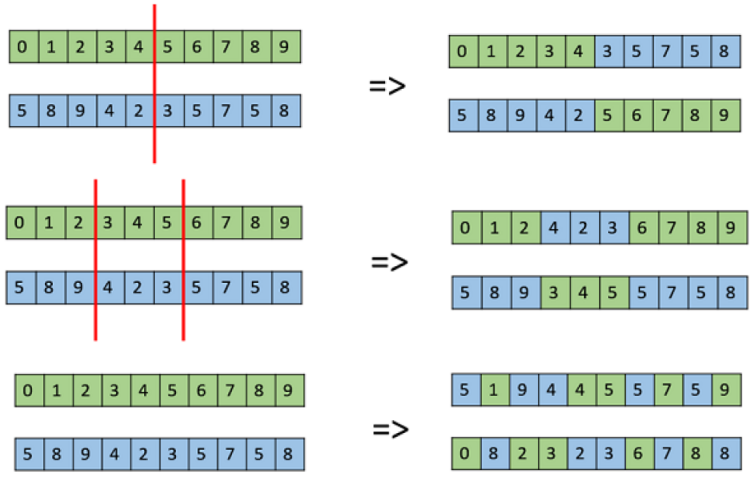
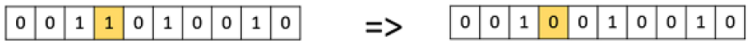


Figure: Top: One Point Crossover; Middle: Multi Point Crossover; Bottom: Uniform Crossover

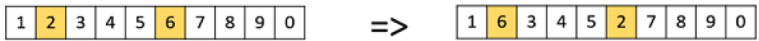
Mutation operator

Mutation may be defined as a small random tweak in the chromosome, to get a new solution. Typically mutation happens with a very low probability, such as 0.001.

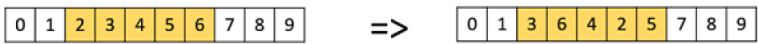
1. **Bit Flip Mutation:** we select one or more random bits and flip them.



2. **Random Resetting:** bit flip for the integer representation
3. **Swap Mutation** we select two positions on the chromosome at random, and interchange the values.



4. **Scramble Mutation:** a subset of genes is chosen and their values are scrambled or shuffled randomly.



5. **Inversion Mutation:** we select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset

Mutation operator

- Some algorithms implement the mutation operator before the selection and crossover operators; this is a matter of preference.
- At first glance, the mutation operator may seem unnecessary. In fact, it plays an important role, even if it is secondary to those of selection and crossover

- Selection and crossover maintain the genetic information of fitter chromosomes, but these chromosomes are only fitter relative to the current generation.
- This can cause the algorithm to converge too quickly and lose "potentially useful genetic material (1's or 0's at particular locations)"
- In other words, the algorithm can get stuck at a local optimum before finding the global optimum
- The mutation operator helps protect against this problem by maintaining diversity in the population, but it can also make the algorithm converge more slowly.
- Typically the selection, crossover, and mutation process continues until the number of offspring is the same as the initial population, so that the second generation is composed entirely of new offspring and the first generation is completely replaced.
- Now the second generation is tested by the fitness function, and the cycle repeats.

Advantages of GAs

1. Does not require any derivative information (which may not be available for many real world applications)
2. Is faster and more efficient as compare to the traditional method
3. Has very good parallel capabilities
4. Optimizes both continuous and discrete functions and also multi-objective problems
5. Provides a list of **good** solutions and not just a single solution
6. Always gets an answer to the problem, which gets better over time
7. Useful when the search space is very large and there are large number of parameters involved, etc,.

Limitations of GAs

1. GAs are not suited for all problems, especially problems which are simple for which derivative information is available
2. Fitness value is calculated repeatedly which might be computationally expensive for some problems
3. Being stochastic, there are no guarantees on the optimality or the quality of the solutions
4. If not implemented properly, the GA may not converge to the optimal solution, etc..

Example-1

Consider the problem of maximizing the function

$$f(x) = x^2$$

where x is allowed to vary between 0 and 31.

Selection:

String No.	Initial population (randomly selected)	x value	Fitness value $f(x) = x^2$	$Prob_i$	Percentage probability	Expected count	Actual count
1	01100	12	144	0.1247	12.47%	0.4987	1
2	11001	25	625	0.5411	54.11%	2.1645	2
3	00101	5	25	0.0216	2.16%	0.0866	0
4	10011	19	361	0.3126	31.26%	1.2502	1
Sum			1155	1.0000	100%	4.0000	4
average			288.75	0.2500	25%	1.0000	1
maximum			625	0.5411	54.11%	2.1645	2

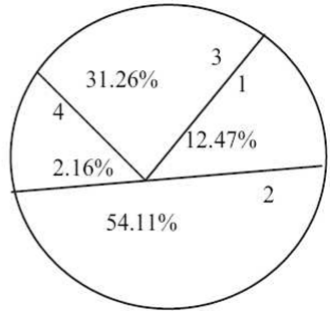
Selection

- Calculate probability: String 1 : $\frac{f(x_1)}{\sum_{i=1}^{4} f(x_i)} = \frac{144}{144+625+25+361} = \frac{144}{1155} = 0.1247$

- Percentage of probability = probability $\times$ 100
- Calculation of expected count: $\frac{f(x_i)}{Avg(f(x))}$, where

$$Avg(f(x)) = \frac{\sum_{i=1}^{n} f(x_i)}{n}$$

- Actual count is the round up integer
- $0.4987 \approx 1$



- Since the actual count of string no 3 is 0, it does not occur in the mating pool.

Crossover

String No.	Mating pool	Crossover point	Offspring after crossover	x value	Fitness value $f(x) = x^2$
1	0 1 1 0 0	4	0 1 1 0 1	13	169
2	1 1 0 0 1	4	1 1 0 0 0	24	576
2	1 1 0 0 1	2	1 1 0 1 1	27	729
4	1 0 0 1 1	2	1 0 0 0 1	17	289
Sum					1763
average					440.75
maximum					729

- single point crossover is performed

Mutation

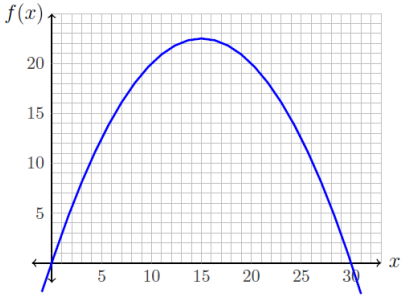
String No.	Offspring after crossover	Mutation chromosomes for flipping	Offspring after Mutation	X value	Fitness value $F(x) = x^2$
1	**0** 1 1 0 1	**1** 0 0 0 0	1 1 1 0 1	29	841
2	1 1 0 0 0	0 0 0 0 0	1 1 0 0 0	24	576
2	1 1 0 1 1	0 0 0 0 0	1 1 0 1 1	27	729
4	1 0 **0** 0 1	0 0 **1** 0 0	1 0 1 0 0	20	400
Sum					2546
average					636.5
maximum					841

Example-2

Consider the problem of maximizing the function

$$f(x) = 3x - \frac{x^2}{10}$$

where x is allowed to vary between 0 and 31.



- we must encode the possible values of x as chromosomes
- we will encode x as a binary integer of length 5.
- range from 0 (00000) to 31 (11111)

- To begin the algorithm, we select an initial population of 10 chromosomes at random.
- Next we take the x-value that each chromosome represents and test its fitness with the fitness function.

Table 1: Initial Population

Chromosome Number	Initial Population	x Value	Fitness Value $f(x)$	Selection Probability
1	01011	11	20.9	0.1416
2	11010	26	10.4	0.0705
3	00010	2	5.6	0.0379
4	01110	14	22.4	0.1518
5	01100	12	21.6	0.1463
6	11110	30	0	0
7	10110	22	17.6	0.1192
8	01001	9	18.9	0.1280
9	00011	3	8.1	0.0549
10	10001	17	22.1	0.1497
		Sum	147.6	
		Average	14.76	
		Max	22.4	

- We select the chromosomes that will reproduce based on their fitness values, using the following probability:

$$P(\text{chromosome i reproduces}) = \frac{f(x_i)}{\sum_{k=1}^{10} f(x_k)}$$

- Selecting the chromosomes, spinning a weighted roulette wheel can be used
- Since our population has 10 chromosomes and each 'mating' produces 2 offspring, we need 5 matings to produce a new generation of 10 chromosomes.
- To create their offspring, a crossover point is chosen at random, which is shown in the table as a vertical line.
- Note that it is possible that crossover does not occur, in which case the offspring are exact copies of their parents.

Table 2: Reproduction & Second Generation

Chromosome Number	Mating Pairs	New Population	x Value	Fitness Value $f(x)$
5	01|100	01010	10	20
2	11|010	11100	28	5.6
4	0111|0	01111	15	22.5
8	0100|1	01000	8	17.6
9	0001|1	01010	10	20
2	1101|0	11011	27	8.1
7	10110	10110	22	17.6
4	01110	01110	14	22.4
10	100|01	10001	17	22.1
8	010|01	01001	9	18.9
			Sum	174.8
			Average	17.48
			Max	22.5

- Lastly, each bit of the new chromosomes mutates with a low probability.
- For this example, we let the probability of mutation be 0.001. With 50 total transferred bit positions, we expect $50 \times 0.001 = 0.05$ bits to mutate.
- Thus it is likely that no bits mutate in the second generation.
- we see that both the maximum fitness and average fitness of the population have increased after only one generation.

Continuous Genetic Algorithm

- The last example allows for integer solutions
- What if you are attempting to solve a problem where the values of the variables are continuous and you want to know them to the full machine precision?
- In this case it makes more sense to make each chromosome an array of real numbers (floating-point numbers) as opposed to an array of just 0's and 1's
- In fact, this continuous genetic algorithm is faster than the binary genetic algorithm
- The primary difference is the fact that variables are no longer represented by bits of 0's and 1's, but instead by floating-point numbers

- If the chromosome has N_{par} variables (an N-dimensional optimization problem) given by

$$p_1, p_2, \ldots, p_{N_{par}}$$

then the chromosome is written as an array with $1 \times p_{N_{par}}$ elements so that

$$\text{chromosome} = [p_1, p_2, \ldots, p_{N_{par}}]$$

- Fitness value using fitness function $f(x)$

$$f(\text{chromosome}) = f(p_1, p_2, \ldots, p_{N_{par}})$$

Example:

Consider the fitness function $f(x, y) = x \sin(4x) + 1.1 \cos(2y)$ Subject to the constraints:

$$0 \leq x \leq 10,\ 0 \leq y \leq 10$$

- Since f is a function of x and y only, the clear choice for the variables is

$$\text{chromosome} = [x, y], \text{ where } N_{par} = 2.$$

- Our goal is to find the global minimum value of $f(x, y)$
- **Fix:** population size to be 8 chromosomes, the mutation rate to be 0.2, and the number of iterations to be 3.

- The initial population is generated randomly between 0 and 10 instead of just 1 or 0
- `pop = rand(`N_{pop}, N_{par}`)`
- A slightly different approach for constructing subsequent generations than we used in the previous examples.

TABLE 3.1 Example Initial Population of 8 Random Chromosomes and Their Corresponding Cost

x	y	Cost
6.9745	0.8342	3.4766
0.30359	9.6828	5.5408
2.402	9.3359	–2.2528
0.18758	8.9371	–8.0108
2.6974	6.2647	–2.8957
5.613	0.1289	–2.4601
7.7246	5.5655	–9.8884
6.8537	9.8784	13.752

- Instead of replacing the entire population with new offspring, we keep the fitter half of the current population, and generate the other half of the new generation through selection and crossover.
- Note that the fitness function takes on both positive and negative values, so we cannot directly use the sum of the chromosomes' fitness values to determine the probabilities of who will be selected to reproduce (AS WE DID IN PREV. EXAMPLE)
- Instead we use rank weighting

TABLE 3.2 Surviving Chromosomes after a 50% Selection Rate

Number	x	y	Cost
1	7.7246	5.5655	−9.8884
2	0.1876	8.9371	−8.0108
3	2.6974	6.2647	−2.8957
4	5.6130	0.12885	−2.4601

- the probability that the chromosome in n th place will be a parent is given by the equation

$$P_n = \frac{N_{keep} - n + 1}{\sum_{i=1}^{N_{keep}} i} = \frac{4 - n + 1}{1 + 2 + 3 + 4} = \frac{5 - n}{10}$$

- so the probability that the chromosome in first place will be selected is

$$P(C_1) = \frac{5 - 1}{10} = \frac{2}{5}$$

- ..... and the probability that the chromosome in sixth place will be selected is

$$P(C_6) = \frac{5 - 4}{10} = \frac{1}{10}$$

- Recall that each mating produces 2 offspring, so we need 2 pairs of parent chromosomes to produce the right number of offspring to fill the rest of the next generation.
- **Selection:** A random number generator produced the following two pairs of random numbers

$$(ma =) m_{ind} = [2, 3], \quad (pa =) d_{ind} = [3, 1]$$

- **Crossover:** There are several methods to achieve this; here we will use Haupt's method[1]
- Once we have two parent chromosomes

$$m = [x_m, y_m], \quad d = [x_d, y_d]$$

- we first randomly select one of the parameters to be the point of crossover.
- To illustrate, suppose x is the crossover point for a particular m and d. Then we introduce a random value between 0 and 1 represented by $\beta$and the x-values in the offspring are

$$x_{new1} = x_m - \beta(x_m - x_d)$$
$$x_{new2} = x_d + \beta(x_m - x_d)$$

- The remaining parameter (y in this case) is inherited directly from each parent, so the completed offspring are

$$\text{offspring}_1 = [x_{new1}, y_d]$$
$$\text{offspring}_2 = [x_{new1}, y_m]$$

[1]Randy L. Haupt Sue Ellen Haupt, Practical Genentic Algorithm, Wiley

- suppose that

$$\text{chromosome}_2 = [0.1876, 8.9371] \quad \text{chromosome}_3 = [2.6974, 6.2647]$$

  are selected as a pair of parents.
- Then with crossover at x and a $\beta = 0.0272$,(random)

$$\text{offspring}_1 = [0.18758 - 0.0272(0.18758 - 2.6974), 6.2647] = [0.2558, 6.2647]$$
$$\text{offspring}_2 = [2.6974 + 0.0272(0.18758 - 2.6974), 8.9371] = [2.6292, 8.9371].$$

- For the second set of parents: continuing this process once more with a $\beta = 0.7898$

$$\text{offspring}_3 = [6.6676, 5.5655]$$
$$\text{offspring}_4 = [3.7544, 6.2647].$$

TABLE 3.3 Pairing and Mating Process of Single-Point Crossover Chromosome Family Binary String Cost

2	ma(1)	0.18758	8.9371
3	pa(1)	2.6974	6.2647
5	$offspring_1$	0.2558	6.2647
6	$offspring_2$	2.6292	8.9371
3	ma(2)	2.6974	6.2647
1	pa(2)	7.7246	5.5655
7	$offspring_3$	6.6676	5.5655
8	$offspring_4$	3.7544	6.2647

Mutations

- we chose a mutation rate of 0.2 or 20%
- Multiplying the mutation rate by the total number of variables that can be mutated in the population gives $0.2 * (8 - 1) * 2 \approx 3$
- Next random numbers are chosen to select the row and columns of the variables to be mutated.
- A mutated variable is replaced by a new random variable.The following pairs were randomly selected:

$$mrow = [4, 4, 7], \quad mcol = [1, 2, 1]$$

- The first random pair is (4, 1). Thus the value in row 4 and column 1 of the population matrix is replaced with a uniform random number between one and 10:

$$5.6130 \Rightarrow 9.8190$$

- Mutations occur two more times

TABLE 3.4 Mutating the Population

Population after Mating		Population after Mutations		
x	y	x	y	cost
7.7246	5.5655	7.7246	5.5655	–9.8884
0.18758	8.9371	0.18758	8.9371	–8.0108
2.6974	6.2647	2.6974	6.2647	–2.8957
5.613	0.12885	*9.819*	*7.1315*	17.601
0.2558	6.2647	0.2558	6.2647	–0.03688
2.6292	8.9371	2.6292	8.9371	–10.472
6.6676	5.5655	*9.1602*	5.5655	–14.05
3.7544	6.2647	3.7544	6.2647	2.1359

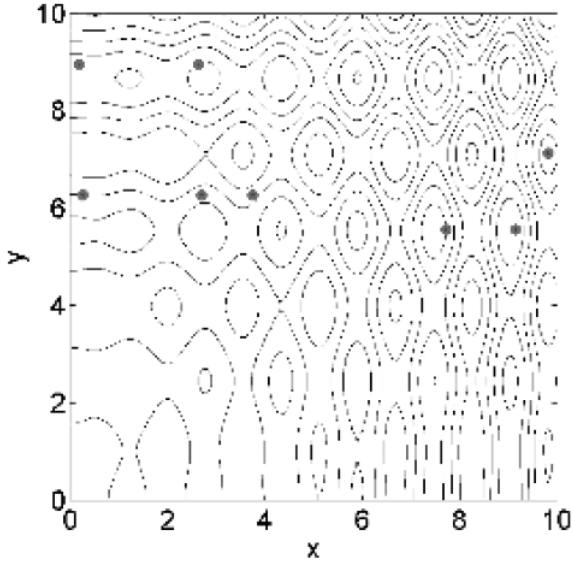


Figure: Contour plot of the cost function with the population after the first generation.

The Next Generation

TABLE 3.5 New Ranked Population at the Start of the Second Generation

x	y	Cost
9.1602	5.5655	–14.05
2.6292	8.9371	–10.472
7.7246	5.5655	–9.8884
0.18758	8.9371	–8.0108
2.6974	6.2647	–2.8957
0.2558	6.2647	–0.03688
3.7544	6.2647	2.1359
9.819	7.1315	17.601

TABLE 3.6 Population after Crossover and Mutation in the Second Generation

x	y	Cost
9.1602	5.5655	–14.05
2.6292	8.9371	–10.472
7.7246	6.4764	–1.1376
0.18758	8.9371	–8.0108
2.6292	5.8134	–7.496
9.1602	8.6892	–17.494
7.7246	8.6806	–13.339
4.4042	7.969	–6.1528

The 2nd Generation

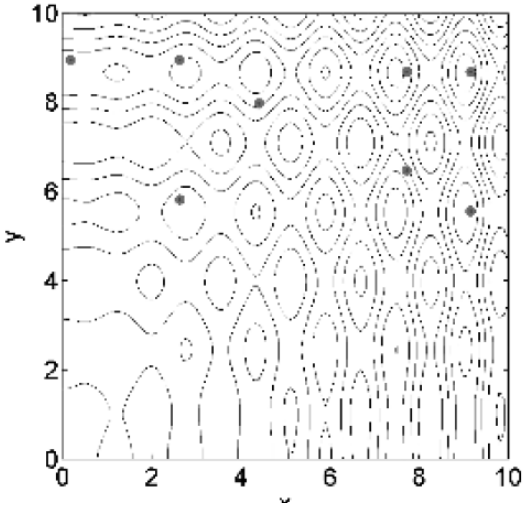


Figure: Contour plot of the cost function with the population after the second generation.

The 3rd Generation

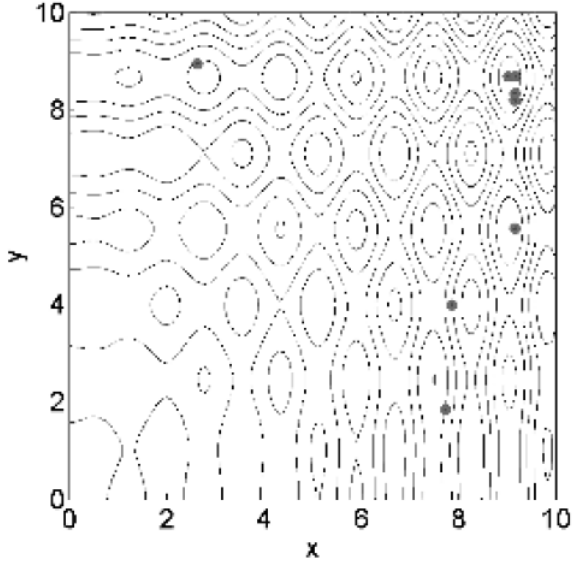
TABLE 3.7 New Ranked Population at the Start of the Third Generation

x	y	Cost
9.1602	8.6892	–17.494
9.1602	5.5655	–14.05
7.7246	8.6806	–13.339
2.6292	8.9371	–10.472
0.18758	8.9371	–8.0108
2.6292	5.8134	–7.496
4.4042	7.969	–6.1528
7.7246	6.4764	–1.137

TABLE 3.8 Ranking of Generation 3 from Least to Most Cost

x	y	Cost
9.0215	8.6806	–18.53
9.1602	8.6892	–17.494
9.1602	8.323	–15.366
9.1602	5.5655	–14.05
9.1602	8.1917	–13.618
2.6292	8.9371	–10.472
7.7246	1.8372	–4.849
7.8633	3.995	4.6471

The 3rd Generation



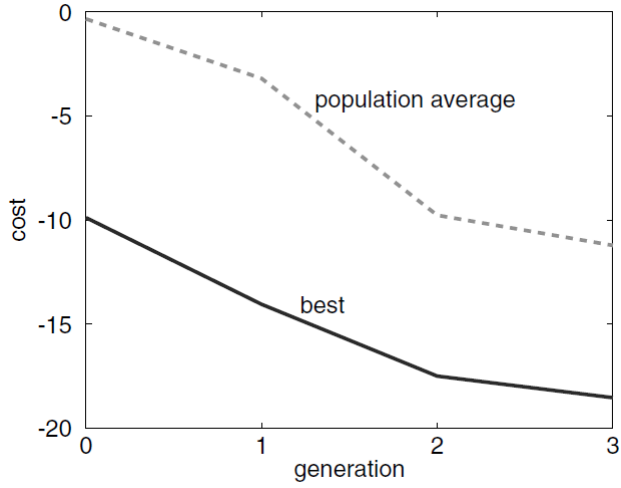


Figure: Plot of the minimum and mean costs as a function of generation.The algorithm converged in three generations.