Name – Gautam Singla
Roll No. – 19MA20017

PROJECT ON GENETIC ALGORITHM

MINIMIZATION PROBLEM USING SINGLE OBJECTIVE GENETIC ALGORITHM

INTRODUCTION

Genetic Algorithms are playing very important role in today's world. These are very effective techniques which are used to solve very important real-life problems. These are used to solve optimization problems where other mathematical techniques are very difficult to be applied. Permutation problem can also be solved by genetic algorithms. Here also, we will take up a permutation problem which can have a huge application.

In a genetic algorithm, we basically create an analogy of some natural process or biological process. In this approach, at the initial point we initialize random population. Each member of this population is known as *individuals*. We have many type-of operators such as mutation, crossover which are also the analogies of some processes. These operators played very important role. These operators are used to create next generation. The search procedure is quasi random; in each generation using the population members that covers a very significant portion of the total search space. This is however done in an intelligent fashion by designing a biologically inspired fitness for the individual solutions. The algorithms are designed in such a way that from one generation to the next the fitness values of the individuals tend to increase. The search therefore, is carefully driven towards the direction of higher fitness, rendering the procedure far more efficient than a fully random or a brute force search.

PROBLEM DESCRIPTION

The problem which we will be solving here is a permutation problem. Let us discuss it in detail with its application.

Let us suppose we have 'x' different things (We have many pieces of one particular thing). There are 'n' people. Let us suppose some people may know each other and others may not. For this, we are given a matrix of size (n x n) which will tell us which persons know each other assuming diagonal elements to be zero. Let us suppose there are four persons.

r = row number

$1 \leq r \leq 4$

c = column number

$1 \leq c \leq 4$

Table 1

r \ c	P1	P2	P3	P4
P1	0	1	0	1
P2	1	0	1	0
P3	0	1	0	0
P4	1	0	0	0

Diagonal elements are assumed to be zero. Any other block value will be one if corresponding to that block the two members know each other.
(r,c) = 1 this implies that Pr and Pc (rth person knows cth person) know each other. And if (r,c) = 1 then (c,r) = 1 as if rth person knows cth person then cth person will also know rth person. So, this matrix is a symmetric matrix.
Our aim is to give some fix **'k' different things** to each person in such a way, so that people who knows each other should get different things, not same. So, our aim is to minimize the sharing if they know each other. **One important condition is n*k > x. If condition is not satisfied, then we can simply distribute all n persons with k different things.** There can be many solutions to above problem but we are interested in only one.

Let us take one application which we will solve and understand the things more easily.

Example 1
Let us suppose we are organizing an online examination. Let us suppose there are 6 students in the exam. I have total 10 questions. Each student should get 5 questions. So, we have n=6, k=5, x=10. And clearly condition is satisfied(n*k>x).

Table 2

r \ c	P1	P2	P3	P4	P5	P6
P1	0	1	0	0	1	0
P2	1	0	0	1	0	1
P3	0	0	0	0	1	1
P4	0	1	0	0	0	0
P5	1	0	1	0	0	0
P6	0	1	1	1	0	0

So, our aim is to set the question papers in such a way that people who knows each other should get different questions. Our aim is to minimize the cheating in the exam.

SIMPLE GENETIC ALGORITHM

Flowchart of Simple Genetic Algorithm –

Pseudocode SGA
Begin
generation of random binary population

begin
selection
crossover
mutation
repeat until (happy)
end
end

CONSTRUCTION OF THE ALGORITHM

The algorithm we will use is not exactly simple genetic algorithm. Many modifications are made in technique according to the requirement.

INDIVIDUALS
Each member of a population is known as individual. Every individual is a capable solution of the given problem. In our problem population members are not bit strings as in case of simple genetic programming. Here in our problem, we have population of matrices. Each individual is represented by matrix of size (6 x 5). Values in the matrix can be any number between 1 to x (in below example 1 to 10). **Any row cannot have two same members (as questions cannot be same in exam).**
For example -

10	9	7	3	5
1	3	10	2	6
1	4	7	5	10
3	10	4	5	1
6	10	7	4	5
3	9	8	6	2

Here each row tells us the question no. given to a particular student. Rth row gives question number for Pr student.

FITNESS
Here fitness values are calculated in a very unique way. Here for a particular individual, we calculate fitness by counting the number of common questions of friends and then multiplying it with -1. Let us take example shown above to calculate the fitness value. Let us use the relation matrix as stated in Table 2.

P1 is a friend of P2. So, number of common questions are 2.
P1 is a friend of P5. So, number of common questions are 3.
P2 is a friend of P4. So, number of common questions are 3.
P2 is a friend of P6. So, number of common questions are 3.
P3 is a friend of P5. So, number of common questions are 4.
P3 is a friend of P6. So, number of common questions are 0.
So, fitness value is -15.

SELECTION

For selection, we have used tournament selection of size 2. Here we randomly select two members from the population and one with the higher fitness value wins. All the members get two chances for getting selected. This is very helpful in maintaining the diversity in the population. Diversity in population is very important as it leads to fast convergence and prevents us from stucking at local optima.

MUTATION

Process of mutation is little bit different. Here we select one parent on which we have to perform mutation. Then we have some fixed number p(μ). We will perform mutation only if 0<rn< p(μ).

rn = random number generated

p(μ) = mutation probability

0<rn< p(μ)

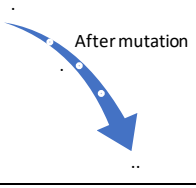
Here we will generate random number 'rn' for every row and if above condition is satisfied then we randomly select one location of that row and mutate it. As already stated above, any row cannot have the same numbers. This thing is taken into account and after mutation we will get a different number from that which are already present. Mutation here in our case is made self adjusting as it decreases p(μ) value after some generations, which helps in fast convergence. One example of mutation is shown below –

p(μ) = 0.2

Parent selected for mutation rn

10	9	7	3	5	0.1
1	3	10	2	6	0.5
1	4	7	5	10	0.63
3	10	4	5	1	0.15

6	10	7	4	5	0.9
3	9	8	6	2	0.6



10	9	7	4	5
1	3	10	2	6
1	4	7	5	10
3	2	4	5	1
6	10	7	4	5
3	9	8	6	2

Parent after mutation

In above example, for 1st and 4th row '**rn**' lies between 0 and p(μ). So, one member of both rows is randomly changed.

CROSSOVER

Crossover is a very important operator as it helps in global search. Here we select two parents on which we have to perform crossover. Then we have some fixed number p(c). We will perform crossover only if 0<rn< p(c).

rn = random number generated

p(c) = crossover probability

0<rn< p(c)

Here we will generate random number 'rn' for two selected parents and if above condition is satisfied then we randomly select one row and exchange it. As already

stated above, any row cannot have the same numbers. This condition is not violated in this mechanism. One example of mutation is shown below –

Parent selected for crossover p(c) = 0.6 rn = 0.5

10	9	7	3	5		7	5	2	10	6
1	3	10	2	6		1	9	8	5	6
1	4	7	5	10		9	2	6	3	4
3	10	4	5	1		3	5	7	10	1
6	10	7	4	5		5	10	4	2	6
3	9	8	6	2		6	7	1	10	5
10	9	7	3	5		7	5	2	10	6
1	3	10	2	6		1	9	8	5	6
9	2	6	3	4		1	4	7	5	10
3	10	4	5	1		3	5	7	10	1
6	10	7	4	5		5	10	4	2	6
3	9	8	6	2		6	7	1	10	5

Parent after crossover

In above example, condition is satisfied (0<rn< p(c)). So, we randomly generate one number from 1 to 6 (3rd in above example) and exchange that row from both the parents.

ELITISM

Elites E is a proper nonempty subset of the population set P at any generation t, having high fitness such that $Et \subset Pt$. Usually the number of elites is taken as small compared to the population size. In our case, it has only one member. Also, the fitness (F) of any elite member i needs to be better than any other member j in the population that is not an elite.

Parameters used in our algorithm

Number of members in elite set	1

Tournament selection size	2
Crossover Probability	0.6
Mutation probability	0.4 after some generation changes to 0.2

RESULTS AND DISCUSSION

Results for example 1 (Page 2)

INITIAL POPULATION

Initial population is randomly generated. Here population size is 40.

Fitness= -15	Fitness= -15	Fitness = -15	Fitness = -14
10 9 7 3 5	7 5 2 10 6	6 10 4 3 9	7 5 2 10 6
1 3 10 2 6	1 9 8 5 6	1 5 9 2 4	1 9 8 5 6
1 4 7 5 10	9 2 6 3 4	6 3 2 9 10	7 9 1 8 5
3 10 4 5 1	3 5 7 10 1	3 10 6 5 1	4 2 7 9 10
6 10 7 4 5	5 10 4 2 6	6 9 1 2 4	7 5 3 10 4
3 9 8 6 2	6 7 1 10 5	10 8 1 4 2	2 5 9 3 1

Fitness = -14	Fitness = -13	Fitness = -13	Fitness = -14
7 8 4 9 5	3 6 2 10 9	5 9 8 6 10	5 9 4 10 8
4 3 7 6 8	10 5 2 6 7	1 6 2 3 10	7 5 6 9 2
8 5 6 10 9	6 2 9 5 1	9 10 2 8 4	6 2 10 8 5
4 1 5 7 8	6 1 8 3 2	4 1 8 7 3	6 5 3 4 7
2 6 5 1 3	5 10 1 4 6	9 10 5 3 1	9 1 7 4 10
7 1 5 8 4	1 8 7 9 4	9 6 4 1 7	5 6 7 3 4

Fitness = -15	Fitness = -13	Fitness = -14	Fitness = -14
9 7 8 4 2	1 7 10 8 9	9 1 8 4 2	4 3 5 2 10
6 8 9 4 3	7 5 9 10 4	1 8 4 5 10	1 9 8 10 6
5 10 2 9 8	7 8 2 5 1	7 6 2 1 3	7 9 1 6 5
4 5 3 1 7	7 2 3 5 10	10 7 9 4 5	3 2 7 1 10
3 8 1 5 2	6 3 4 8 2	9 7 10 3 6	1 5 3 10 9

1 4 8 10 6

6 3 4 2 7

6 4 3 5 9

2 5 9 3 1

Fitness = -13

10 3 4 8 2

6 9 2 10 3

5 2 6 10 1

7 5 6 4 1

9 10 2 4 5

8 5 7 4 6

Fitness = -10

6 10 4 3 1

8 5 9 2 4

6 3 7 9 10

3 10 6 5 1

7 9 1 2 4

3 8 1 4 2

Fitness = -14

3 10 2 5 9

7 5 9 10 4

6 2 9 5 1

6 5 8 3 2

5 10 1 4 6

1 8 7 9 4

Fitness = -17

9 3 8 4 7

1 5 10 7 4

7 3 5 6 9

9 2 4 5 10

9 7 5 6 10

10 9 4 7 3

Fitness = -18

4 3 5 8 10

6 8 5 4 1

7 9 1 8 5

4 2 7 1 10

1 5 3 10 4

2 5 9 8 1

Fitness = -15

9 10 3 7 1

2 5 9 10 4

6 10 8 9 1

2 5 8 3 1

10 3 7 5 6

1 5 6 2 10

Fitness = -15

4 3 5 2 10

10 8 5 4 1

9 2 6 3 4

3 5 7 10 1

5 10 4 1 6

6 7 1 10 5

Fitness = -15

9 6 8 4 2

1 9 2 10 3

5 10 2 6 8

4 5 3 1 2

3 8 1 5 2

1 4 8 10 6

Fitness = -16

3 1 8 4 6

4 5 1 7 8

6 1 2 10 7

5 6 3 2 7

1 4 2 5 10

3 1 8 7 6

Fitness = -13

4 3 5 2 10

1 9 8 5 6

7 9 1 8 5

4 2 7 1 10

1 5 3 10 4

2 5 9 3 6

Fitness = -13

4 3 5 2 10

1 9 8 5 6

7 9 1 8 5

4 2 7 1 10

5 10 4 2 6

2 5 9 3 1

Fitness = -15

7 8 4 10 5

10 3 7 6 8

8 9 2 6 7

4 1 3 7 8

2 6 5 9 3

7 1 2 8 4

Fitness = -15

3 6 2 5 9

7 5 9 10 4

Fitness = -14

5 4 2 3 9

8 5 3 10 4

Fitness = -11

7 5 2 9 6

6 8 5 4 1

Fitness = -12

2 9 7 3 1

7 3 10 2 8

6 2 9 5 1	2 10 9 7 6	9 2 6 3 4	1 4 7 5 10
6 5 8 3 2	1 3 9 7 5	3 6 7 10 1	3 10 4 5 1
5 9 1 4 6	9 5 10 3 1	1 5 3 10 4	6 10 8 4 5
1 8 7 9 4	1 7 2 4 9	6 7 1 10 5	1 9 6 8 10
Fitness = -15	Fitness = -16	Fitness = -16	Fitness = -14
7 8 4 9 5	9 1 8 4 2	9 3 8 2 7	3 9 8 1 4
3 10 9 8 6	1 8 4 5 10	2 6 3 8 5	2 6 3 4 5
9 4 10 5 7	7 6 2 1 10	7 3 5 6 9	9 3 8 7 1
10 9 6 7 1	10 7 9 4 5	7 2 3 5 10	4 8 3 9 1
8 3 7 5 2	9 7 8 1 10	9 3 5 6 10	8 3 6 10 2
4 3 2 7 9	6 7 8 5 9	10 9 4 7 3	3 9 8 6 2

Fitness = -14	Fitness = -12	Fitness = -13	Fitness = -13
1 4 5 2 3	5 1 8 6 10	3 9 2 1 4	7 4 2 10 1
5 3 7 6 8	1 6 2 3 10	1 5 10 7 4	7 9 3 6 2
8 5 6 10 9	9 10 2 8 7	9 3 4 7 6	8 2 7 1 3
4 1 3 7 8	10 7 9 4 5	4 6 3 9 1	5 10 1 3 2
2 6 5 1 3	9 6 5 3 1	8 9 6 10 2	6 5 3 1 2
7 1 2 5 4	6 8 4 1 7	1 9 4 8 10	3 8 10 6 5

Fitness = -15	Fitness = -14	Fitness = -13	Fitness = -13
5 4 2 3 9	9 1 8 4 2	1 9 7 2 10	6 7 2 5 3
8 9 3 10 4	1 8 4 5 10	9 10 5 3 1	6 7 10 2 3
2 10 9 7 6	7 6 2 1 5	8 5 6 10 9	1 2 3 8 7
1 2 9 7 5	4 1 8 7 6	2 8 5 6 9	8 3 5 4 7
9 7 2 1 10	9 5 10 3 6	7 3 6 4 10	3 9 4 5 7
1 6 2 4 9	6 4 8 5 9	4 5 7 2 9	9 5 8 4 10

Best solution

Fitness= -10

6 10 4 3 1

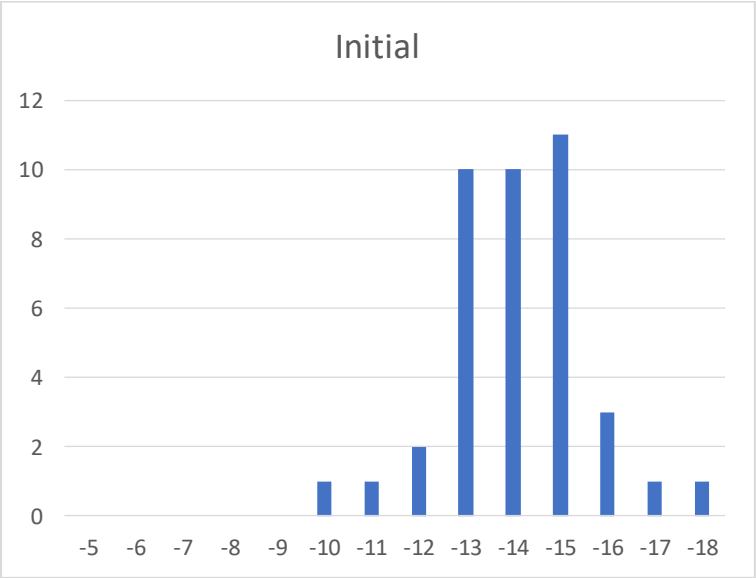
8 5 9 2 4

6 3 7 9 10

3 10 6 5 1

7 9 1 2 4

3 8 1 4 2



Above is the distribution of fitness values.

AFTER 100 GENERATIONS

Fitness = -11	Fitness = -7	Fitness = -9	Fitness = -8
7 9 10 4 1	7 8 10 5 1	7 9 10 5 4	7 9 10 4 1
8 7 2 6 3	8 7 2 6 3	8 10 2 6 3	8 7 2 6 3
6 1 10 7 8	6 1 10 7 8	6 1 10 7 8	6 1 10 7 2
8 1 5 4 7	9 4 5 6 10	9 7 5 4 10	8 1 5 4 10

4 3 8 9 1
3 4 5 8 9

4 3 8 9 2
3 4 5 9 10

5 3 8 4 1
3 4 5 9 10

4 3 8 9 2
3 4 5 8 10

Fitness = -6
7 9 10 4 1
8 7 2 6 3
6 1 10 7 8
9 1 5 4 10
4 3 8 9 2
2 4 5 9 10

Fitness = -6
7 9 10 5 4
8 1 2 6 3
6 1 10 7 8
9 7 5 4 10
5 3 8 9 1
3 4 5 9 10

Fitness = -7
7 9 10 2 4
8 1 2 6 3
6 1 10 7 2
9 7 5 3 10
5 3 8 10 1
3 4 5 9 10

Fitness = -6
7 9 10 5 1
8 7 2 6 3
6 1 10 7 8
9 4 5 6 10
4 3 8 9 2
3 4 5 9 10

Fitness = -9
7 9 8 5 4
8 1 2 6 3
6 1 10 7 4
9 7 5 4 10
5 3 8 9 1
3 4 7 9 10

Fitness = -6
7 9 10 5 4
8 1 2 6 3
6 1 10 7 8
9 7 5 4 10
5 3 8 4 1
7 4 5 9 10

Fitness = -7
7 9 10 5 4
8 1 2 6 3
6 1 10 7 4
9 7 5 4 10
5 3 8 9 1
3 4 7 9 10

Fitness = -8
7 9 10 5 4
8 1 2 6 3
6 1 10 7 8
9 7 5 4 10
5 10 8 4 1
3 4 5 9 10

Fitness = -6
7 9 10 5 4
8 1 2 6 3
6 1 10 7 8
9 7 5 4 10
5 3 8 9 1
3 4 5 9 10

Fitness = -9
3 9 10 5 4
8 7 2 6 3
6 1 10 7 8
9 1 8 4 10
5 3 8 9 1
3 4 5 9 10

Fitness = -9
3 9 10 5 4
8 7 2 6 3
6 1 10 7 8
9 1 8 4 10
5 3 8 9 1
3 4 5 9 10

Fitness = -9
3 9 10 5 4
8 7 2 6 3
6 1 10 7 8
9 1 8 4 10
5 3 8 9 1
3 4 5 9 10

Fitness = -9

Fitness = -8

Fitness = -6

Fitness = -8

7 6 10 5 4	7 9 2 4 1	7 9 10 5 4	7 9 10 4 1
8 1 7 6 3	8 7 10 6 3	8 1 2 6 3	8 7 2 6 3
6 1 10 7 8	6 1 5 7 8	6 1 10 7 8	6 4 10 7 8
9 7 5 4 10	9 1 8 4 10	9 7 5 4 10	9 1 5 4 10
5 3 8 4 1	4 3 8 10 2	5 3 8 4 1	4 3 8 9 2
3 4 5 9 10	2 4 5 9 1	3 4 5 9 10	2 4 5 9 10

Fitness = -6	Fitness = -6	Fitness = -5	Fitness = -6
7 9 10 5 1	7 9 10 5 4	7 9 10 5 4	7 9 10 5 1
8 7 2 6 3	8 1 2 6 3	8 1 2 6 3	8 7 2 6 3
6 1 10 7 8	6 1 10 7 8	6 1 10 7 8	6 1 10 7 2
9 4 5 6 10	9 7 5 4 10	9 7 5 4 10	9 1 5 4 10
4 3 8 9 2	5 3 8 9 1	4 3 8 9 2	4 3 8 9 2
3 4 5 9 10	3 4 5 9 10	3 4 5 9 10	3 4 5 8 10

Fitness = -11	Fitness = -6	Fitness = -9	Fitness = -7
7 9 10 4 1	7 9 10 5 1	7 9 10 4 1	7 9 10 5 4
8 7 2 6 3	8 7 2 6 3	8 7 2 6 3	8 1 2 6 3
3 4 10 7 8	6 1 10 7 8	6 1 10 7 8	6 1 10 7 8
9 4 5 7 10	9 4 5 6 10	9 1 8 4 10	9 7 5 4 10
4 3 8 9 2	4 3 8 9 2	5 3 8 4 1	5 3 8 9 1
3 4 5 9 10	3 4 5 9 10	2 4 5 9 7	3 4 7 9 10

Fitness = -10	Fitness = -6	Fitness = -8	Fitness = -8
7 9 10 5 4	7 9 10 5 1	7 9 10 5 1	7 9 10 5 1
8 1 9 6 3	8 7 2 6 3	8 7 2 6 3	8 7 2 6 3
6 1 10 7 4	6 1 10 7 8	6 1 10 9 8	6 1 10 7 4
9 7 5 4 10	9 4 5 6 10	9 4 5 6 10	9 4 8 6 10
5 3 8 9 1	4 3 8 9 2	4 3 8 9 2	6 3 8 9 2
3 4 7 9 10	3 4 5 9 10	3 4 5 9 10	3 4 5 9 10

Fitness = -11	Fitness = -8	Fitness = -8	Fitness = -9
7 9 10 4 1	8 9 10 5 4	7 9 10 5 4	7 9 10 3 4
8 7 2 6 3	8 1 2 6 3	8 1 2 6 9	8 7 2 6 3
6 1 10 7 8	6 1 10 7 8	6 1 10 7 8	6 1 10 7 8
9 1 8 4 7	9 7 5 4 10	9 7 5 4 10	9 7 5 4 10
7 1 8 9 2	5 3 8 4 1	5 3 8 9 1	5 3 8 9 1
2 4 5 9 3	3 4 5 9 10	3 4 5 9 10	3 4 5 9 10

Fitness = -12	Fitness = -8	Fitness = -7	Fitness = -8
7 9 10 5 4	7 9 4 5 1	7 9 10 4 1	7 9 10 4 1
5 1 2 6 3	8 7 2 6 3	8 7 2 6 3	8 1 2 6 3
6 1 10 5 8	6 1 10 7 8	6 1 10 7 8	6 1 10 7 8
2 7 5 4 10	8 1 5 4 10	9 4 5 6 10	9 1 8 4 10
5 3 8 4 1	4 3 8 9 5	4 3 8 9 2	4 3 8 9 2
2 4 5 9 10	3 4 5 9 10	2 4 5 9 10	2 4 5 9 7

Best solution

Best Solution Fitness = -5

7 9 10 5 4

8 1 2 6 3

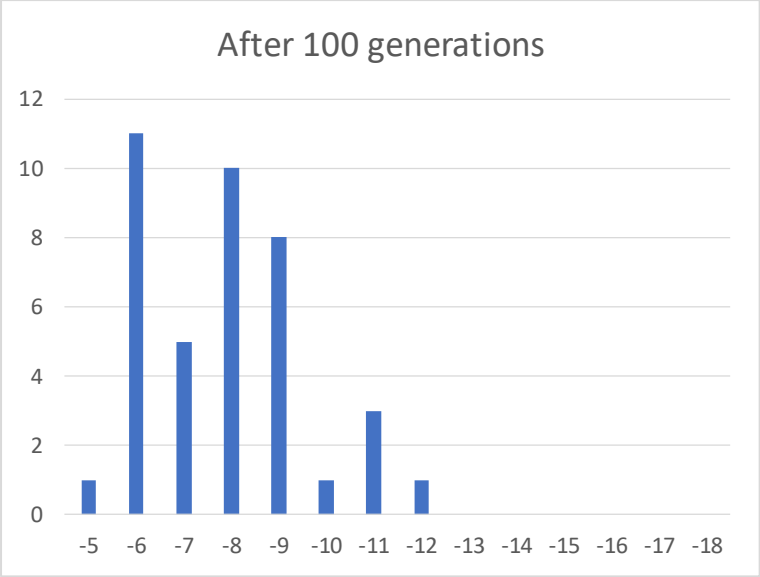
6 1 10 7 8

9 7 5 4 10

4 3 8 9 2

3 4 5 9 10

As we are proceeding, population should converge to its optimum. Now we can see the distribution of fitness values has changed vastly. Initially, population was concentrated in the region of fitness value from -10 to -18. Now population is converging towards its optimum. Now population fitness values lies between -5 and -12. This directly shows the power of genetic algorithms.



AFTER 1000 GENERATIONS

Fitness = -5	Fitness = -5	Fitness = -6	Fitness = -9
6 1 9 7 10	6 1 9 7 10	6 1 9 7 10	6 1 9 3 10
3 4 5 2 8	3 4 5 2 8	3 4 5 2 8	3 4 5 2 8
8 5 9 1 7	2 5 9 3 8	2 5 9 3 8	2 5 9 3 8
7 9 10 6 1	7 9 10 6 1	7 9 10 6 1	7 3 10 6 8
10 3 6 4 2	10 3 6 4 2	10 3 6 4 2	10 3 6 4 2
6 9 10 1 4	6 7 10 1 4	6 9 10 1 4	6 7 10 1 4

Fitness = -8	Fitness = -6	Fitness = -7	Fitness = -7
6 1 9 7 10	6 1 9 7 10	6 1 9 7 10	6 1 9 7 10
3 4 6 2 8	3 4 5 2 8	3 4 5 2 8	3 4 5 2 8
2 5 9 8 1	2 5 9 3 8	8 5 9 1 7	8 5 9 1 7
7 9 10 6 1	7 9 10 6 1	3 9 10 6 1	2 9 10 6 1

10 3 6 4 2
6 7 10 1 4

10 3 6 4 2
6 9 10 1 4

10 3 6 4 2
6 7 10 1 5

10 3 6 4 2
6 7 10 1 5

Fitness = -9
6 8 9 7 10
3 4 5 2 8
2 5 9 3 1
7 9 10 6 1
9 3 6 4 5
6 7 10 1 5

Fitness = -9
6 1 9 7 10
3 4 6 2 8
8 5 9 3 7
7 9 10 6 1
10 3 6 4 5
6 5 10 1 4

Fitness = -6
6 1 9 7 10
3 4 5 2 8
8 5 9 1 3
7 9 10 6 4
7 3 10 4 2
6 7 10 1 4

Fitness = -5
6 1 9 7 10
3 4 5 2 8
8 5 9 1 7
7 9 10 6 1
10 3 6 4 2
6 9 10 1 4

Fitness = -6
6 1 9 7 10
3 4 5 2 8
8 5 9 1 7
2 9 10 6 1
10 3 6 4 2
6 7 10 1 4

Fitness = -7
6 1 9 7 10
3 4 5 2 8
2 5 4 3 8
7 9 10 6 1
10 3 6 4 2
6 9 10 1 4

Fitness = -6
8 1 9 7 10
3 4 5 2 8
2 5 9 8 1
7 9 10 6 1
10 3 6 4 2
6 7 9 1 4

Fitness = -8
6 1 9 7 10
3 4 5 2 9
8 5 9 1 7
7 9 10 6 1
10 3 6 4 2
6 9 10 1 4

Fitness = -8
6 1 9 7 10
3 4 5 7 8
2 5 9 3 8
7 9 10 6 1
10 3 6 4 5
6 9 10 1 4

Fitness = -5
6 1 9 7 10
3 4 5 2 8
8 5 9 1 7
7 9 10 6 1
10 3 6 4 2
6 9 10 1 4

Fitness = -6
6 1 9 7 10
3 4 5 2 8
2 5 9 3 8
7 9 10 6 5
10 3 6 4 2
6 7 10 1 4

Fitness = -5
6 1 9 7 10
3 4 5 2 8
2 5 9 3 8
7 9 10 6 1
10 3 6 4 2
6 7 10 1 4

Fitness = -9

Fitness = -5

Fitness = -5

Fitness = -6

4 1 9 7 10	6 1 9 7 10	6 1 9 7 10	6 1 9 7 10
3 4 6 2 8	3 4 5 2 8	3 4 5 2 8	3 4 5 2 8
2 5 9 8 1	8 5 9 1 7	2 5 9 3 8	2 5 9 8 1
7 9 10 6 3	7 9 10 6 1	7 9 10 6 1	7 9 10 6 1
10 3 6 4 2	10 3 6 4 2	10 3 6 4 5	10 3 6 4 2
6 7 10 1 4	6 9 10 1 4	6 7 10 1 4	6 7 9 1 4

Fitness = -8	Fitness = -5	Fitness = -7	Fitness = -6
8 1 9 7 10	6 1 9 7 10	8 1 9 7 10	6 1 9 7 10
3 4 5 2 8	3 4 5 2 8	3 4 5 2 8	3 4 5 2 8
2 5 9 8 1	2 5 9 8 1	2 5 9 3 1	8 5 9 1 7
7 9 10 6 1	7 9 10 6 1	7 9 10 6 1	2 9 10 6 1
10 3 6 4 2	10 3 6 4 2	9 3 6 4 7	10 3 6 4 2
8 7 9 1 4	6 7 10 1 4	6 7 10 1 4	6 7 10 1 4

Fitness = -6	Fitness = -6	Fitness = -5	Fitness = -11
6 1 9 7 10	6 1 9 7 10	6 1 9 7 10	6 1 9 7 10
3 4 5 2 8	3 4 5 2 8	3 4 5 2 8	9 4 5 2 8
8 5 9 1 7	8 5 9 1 7	2 5 9 3 8	2 5 9 3 1
7 9 10 6 1	7 9 10 6 1	7 9 10 6 1	7 9 10 6 1
10 3 6 4 5	10 3 6 4 8	10 3 6 4 2	9 3 6 4 5
6 7 10 1 4	6 9 10 1 4	6 7 10 1 4	6 7 3 1 5

Fitness = -6	Fitness = -5	Fitness = -6	Fitness = -6
6 1 9 7 10	6 1 9 7 10	8 1 9 7 10	4 1 9 7 10
3 4 5 2 8	3 4 5 2 8	3 4 5 2 8	3 4 5 2 8
2 5 9 3 8	2 5 9 3 8	8 5 9 1 7	8 5 9 1 7
7 9 10 6 1	7 9 10 6 1	7 3 10 6 1	7 9 10 6 1

10 3 6 4 5	10 3 6 4 2	10 3 6 4 2	10 3 6 4 2
6 9 10 1 4	6 7 10 1 4	6 9 10 1 4	6 9 10 1 4

Fitness = -7	Fitness = -5	Fitness = -6	Fitness = -9
6 1 9 7 10	6 1 9 7 10	6 1 9 7 10	6 1 2 7 10
3 4 5 2 8	3 4 5 2 8	3 4 5 2 8	3 4 5 2 8
8 5 9 1 7	2 5 9 8 1	2 5 9 8 1	2 5 9 3 8
2 9 10 6 1	7 9 10 6 1	7 9 10 6 1	7 9 10 6 1
10 3 6 4 2	10 3 6 4 2	10 3 6 4 2	10 7 6 4 2
6 7 10 1 5	6 7 10 1 4	6 7 9 1 4	6 9 10 1 8

Best solution

Best Solution Fitness = -5

6 1 9 7 10

3 4 5 2 8

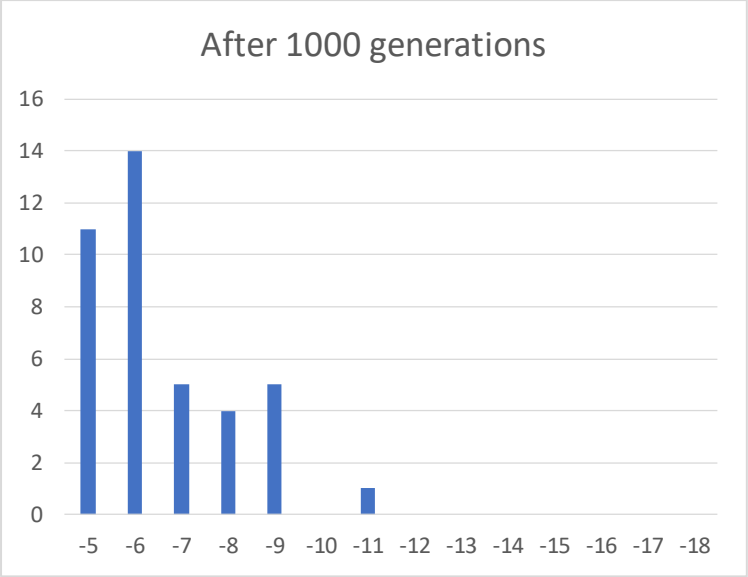
2 5 9 3 8

7 9 10 6 1

10 3 6 4 2

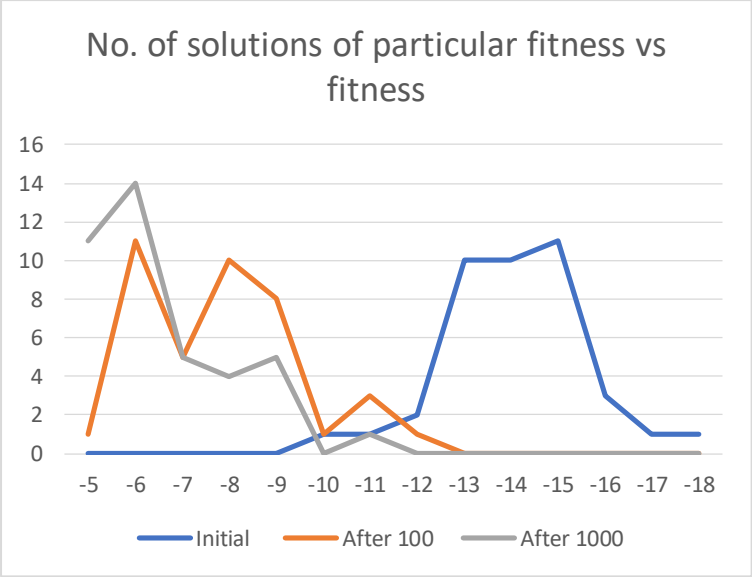
6 7 10 1 4

After 1000 generations, we can see our population has converged to fitness value -5 which is basically optimum value.



CONCLUDING REMARKS

From the problem solved above, we can see the power of genetic algorithms. For example solved above there are total (10C5)^6 (of order 10^14) possible permutations. There are many possible solutions of lowest fitness but they are very less in number. So, by any other approach it is impossible to get the optimum solution. But we get it through genetic algorithms. We can see the below graph which shows the comparative study of fitness value distribution after different number of generations. We can see initially population density is higher towards the low fitness value (range of -10 to -18). After 100 generations, population fitness density is going towards higher fitness value. After 1000 generations, population density vs fitness graph has increasing density towards -5, which is the optimum fitness value. Density is also maintained in the population.



REFERENCES

- Class notes
- Introduction to genetic algorithms by S.N. Sivanandam and S.N. Deepa