Optimization of problem using Binary Coded Genetic Algorithm (BCGA)

ME674 Coding Assignment-2 Report

K. Sai Vishnu Vardhan Reddy

Roll No : 214103010

Department of Mechanical Engineering,

Indian Institute of Technology Guwahati,

Assam, India-781039

1. Problem Statement

Use a binary-coded GA to minimize the function f $(X_1, X_2) = X_1 + X_2 - 2X_1^2 - X_2^2 + X_1X_2$, in therange of $0.0 \leq X_1, X_2 \leq 0.5$. using a random population of size N=6, a single point crossover with probability p_c = 1.0, and assume 5 bits for each variable.

2. Procedure for Binary Coded Genetic Algorithm

a. A population of size N=6 is randomly initialized containing 6 strings of length 10 bits.5 bits for each variable
b. Then decoded values s1 and s2 is then calculated.
c. Then real values x1 and x2 are calculated and fitness values are also calculated using following formula.

$$x_i = x_i^{(L)} + \frac{x_i^{(U)} - x_i^{(L)}}{2^{\ell_i} - 1} DV(\mathbf{s}^i),$$

d. Then the minimization problem is converted into maximization problem using fitnessfunction of F(X) = -f(x)
e. Then reproduction is carried out through tournament selection of size 3 and winnersare obtained from reproduction.
f. Then for Crossover, Elitism method is used in which best 2 solutions are kept sameand Single Point Crossover method is carried out for 4 remaining solutions with probability of p_c = 1.0.
g. Then Mutation is carried out over the population with mutation probability of p_m =0.05.
h. Then this process is carried out for 10000 generations.

3. Results

After 10000 number of generations the population obtained is as follows:

1. 0 0 0 0 0 0 0 0 0 0
2. 0 1 0 0 1 0 0 0 1 0
3. 0 0 0 1 1 0 0 0 1 0
4. 1 0 0 1 1 0 0 1 1 0
5. 0 0 0 0 1 0 0 0 1 0
6. 0 1 0 1 0 0 0 0 1 0

- The final x1, x2 and fitness values are as follows:

Sr. No.	X1	X2	Fitness Value
1	0.00000	0.00000	0.00000
2	0.14516	0.03226	-0.13892
3	0.04839	0.03226	-0.07648
4	0.30645	0.09677	-0.23569
5	0.01613	0.03226	-0.04735
6	0.16129	0.03226	-0.14568

- Final Solution is:
  X1 = 0.00000
  X2 = 0.00000

```c
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>
#include<time.h>

double FN(double x1, double x2)
{
    double F;
    F = x1 + x2 - (2*x1*x1) - (x2*x2) + (x1*x2);
    return -F;
}

int main()
{
    //Initialization
    double FN(double , double);
    int S[7][11],i,j;
    int Gen = 1;
    srand((unsigned)time(NULL));

    //Generating some random Number
    for(i=1;i<=6;i++)
    {
        for(j=1;j<=10;j++)
        {
            S[i][j] = rand() % 2;

        }

    }

    //Decoding the values
    do
    {

        double D1[7],D2[7];
        for(i=1;i<=6;i++)
        {
            D1[i] = 0;
            D2[i] = 0;
            for(j=1;j<=10;j++)
            {
                if(j<=5)
                {
                    D1[i] = D1[i] + (pow(2, (5-j))*S[i][j]);
                }
                else
                {
                    D2[i] = D2[i] + (pow(2, (10-j))*S[i][j]);
                }
            }


        }
```

```
    //Calculalting x1,x2
    double x1min = 0.0, x1max = 0.5, x2min = 0, x2max = 0.5,x1[7],x2[7];
    for(i=1;i<=6;i++)
    {
        x1[i] = x1min + (((x1max - x1min)/((pow(2, 5))-1))*D1[i]);
        x2[i] = x2min + (((x2max - x2min)/((pow(2, 5))-1))*D2[i]);

    }

    //Finding the fitness values for each solution
    double f[7];
    for(i=1;i<=6;i++)
    {
        f[i] = FN(x1[i],x2[i]);

    }

    //Using tournament selection to choose mating pool
    int rn1,rn2,rn3;
    int MS[7][11];
    double max = f[1];

    //Directly taking a best solution
    for(i=1;i<=6;i++)
    {
        if(f[i]>max)
        {
            max = f[i];
        }
    }



    for(i=1;i<=6;i++)
    {
        if(f[i] == max)
        {
           for(j=1;j<=10;j++)
           {
               MS[1][j] = S[i][j];
           }

        }
    }

    for(i=2;i<=6;i++)
    {
        rn1 = (rand() %(6 - 3 + 1)) + 3;
        rn2 = (rand() %(6 - 3 + 1)) + 3;
        rn3 = (rand() %(6 - 3 + 1)) + 3;
        if(f[rn1]>f[rn2])
        {
            if(f[rn1]>f[rn3])
            {
```

```
            for(j=1;j<=10;j++)
            {
                MS[i][j] = S[rn1][j];
            }

        }
        else
        {
            for(j=1;j<=10;j++)
            {
                MS[i][j] = S[rn3][j];
            }

        }
    }
    else
    {
        if(f[rn2]>f[rn3])
        {
            for(j=1;j<=10;j++)
            {
                MS[i][j] = S[rn2][j];
            }

        }
        else
        {
            for(j=1;j<=10;j++)
            {
                MS[i][j] = S[rn3][j];
            }

        }
    }
}


for(i=1;i<=6;i++)
{
    for(j=1;j<=10;j++)
    {

    }

}

//Single point crossover
int CH[7][11], co1, co2, RNC;
RNC = (rand() %(3 - 1 + 1)) + 1;

for(j=1;j<=10;j++)
    {
        CH[1][j] = MS[1][j];
        CH[2][j] = MS[2][j];
    }
```

```
        if(RNC == 1)
        {
            co1 = (rand() %(9 - 1 + 1)) + 1;
            for(j=1;j<=co1;j++)
            {
                CH[3][j] = MS[3][j];
                CH[4][j] = MS[4][j];
            }
            for(j=co1+1;j<=10;j++)
            {
                CH[3][j] = MS[4][j];
                CH[4][j] = MS[3][j];
            }

            co2 = (rand() %(9 - 1 + 1)) + 1;
            for(j=1;j<=co2;j++)
            {
                CH[5][j] = MS[5][j];
                CH[6][j] = MS[6][j];
            }
            for(j=co2+1;j<=10;j++)
            {
                CH[5][j] = MS[6][j];
                CH[6][j] = MS[5][j];
            }
        }

        if(RNC == 2)
        {
            co1 = (rand() %(9 - 1 + 1)) + 1;
            for(j=1;j<=co1;j++)
            {
                CH[3][j] = MS[3][j];
                CH[5][j] = MS[5][j];
            }
            for(j=co1+1;j<=10;j++)
            {
                CH[3][j] = MS[5][j];
                CH[5][j] = MS[3][j];
            }

            co2 = (rand() %(9 - 1 + 1)) + 1;
            for(j=1;j<=co2;j++)
            {
                CH[4][j] = MS[4][j];
                CH[6][j] = MS[6][j];
            }
            for(j=co2+1;j<=10;j++)
            {
                CH[4][j] = MS[6][j];
                CH[6][j] = MS[4][j];
            }
        }

```

```
if(RNC == 3)
{
    co1 = (rand() %(9 - 1 + 1)) + 1;
    for(j=1;j<=co1;j++)
    {
        CH[3][j] = MS[3][j];
        CH[6][j] = MS[6][j];
    }
    for(j=co1+1;j<=10;j++)
    {
        CH[3][j] = MS[6][j];
        CH[6][j] = MS[3][j];
    }

    co2 = (rand() %(9 - 1 + 1)) + 1;
    for(j=1;j<=co2;j++)
    {
        CH[4][j] = MS[4][j];
        CH[5][j] = MS[5][j];
    }
    for(j=co2+1;j<=10;j++)
    {
        CH[4][j] = MS[5][j];
        CH[5][j] = MS[4][j];
    }
}



//Now mutation considering Pm = 0.05
double Pm = 0.05, Pb[7][11];
int RNm;
for(i=2;i<=6;i++)
{
    for(j=1;j<=10;j++)
    {
        RNm = (rand() %(100 - 0 + 1)) + 0;
        Pb[i][j] = (double)RNm/100;
        if(Pb[i][j]<=0.05)
        {
            if(CH[i][j] == 0)
            {
                CH[i][j] = 1;
            }
            else
            {
                CH[i][j] = 0;
            }
        }
    }

}
```

```

        // Restoring the child values back to S pool for next iteration
        for(i=1;i<=6;i++)
        {
            for(j=1;j<=10;j++)
            {
                S[i][j] = CH[i][j];
            }
        }

        Gen = Gen + 1;
    }while(Gen<=10000);

    printf("Final Values\n");
    //Decoding the values of x1 and x2

    for(i=1;i<=6;i++)
    {
        for(j=1;j<=10;j++)
        {
            printf("%d ",S[i][j]);
        }
        printf("\n");
    }

    double D1[7],D2[7];
    for(i=1;i<=6;i++)
    {
        D1[i] = 0;
        D2[i] = 0;
        for(j=1;j<=10;j++)
        {
            if(j<=5)
            {
                D1[i] = D1[i] + (pow(2, (5-j))*S[i][j]);
            }
            else
            {
                D2[i] = D2[i] + (pow(2, (10-j))*S[i][j]);
            }
        }

    }

    //Finding the actual x1 and x2
    double x1min = 0.0, x1max = 0.5, x2min = 0, x2max = 0.5,x1[7],x2[7];
    for(i=1;i<=6;i++)
    {
        x1[i] = x1min + (((x1max - x1min)/((pow(2, 5))-1))*D1[i]);
        x2[i] = x2min + (((x2max - x2min)/((pow(2, 5))-1))*D2[i]);
        printf("X1 = %lf and X2 = %lf\n",x1[i],x2[i]);
    }

    //Finding the fitness values for each solution
    double f[7];
```

```c
    printf("\nRespective fitness values\n");
    for(i=1;i<=6;i++)
    {
        f[i] = FN(x1[i],x2[i]);
        printf("F1 = %lf\n",f[i]);
    }

    //Printing the final values
    double max = f[1];
    for(i=1;i<=6;i++)
    {
        if(f[i]>max)
        {
            max = f[i];
        }
    }

    printf("\n\n Final Soultion is as follows\n");
    for(i=1;i<=6;i++)
    {
        if(f[i] == max)
        {
            printf("X1 = %lf , X2 = %lf\n",x1[i],x2[i]);
        }
    }


    return 0;
}
```