CS 776: Evolutionary Computing -Assignment 2 Fall 2018

Full Name: Batyr Charyyev
Email address: bcharyyev@nevada.unr.edu

First I started with simple HillClimber which starts from 0^{th} digit and flips bits one by one and compares obtained fitness values. If fitness value decreases it reverts changes. I tried to add(with small probability) some randomization by flipping random bits. I also tried to flip two or more digits at once however none of them helped and my fitness value on average was 32.

```
void simpleHillClimber(int iteration)
{
    int fitness1=0;
    int fitness2=0;

    for(int i=0; i<iteration; i++)
    {
        //for(int j = 0; j<150; j++)
        for(int j = 0; j<150; j=j+2)
        {
            flip(j);
            flip(j+1);
            fitness2=eval(vec);

            if(fitness1>fitness2)//revert
            {
                flip(j);
                flip(j+1);
            }
            else
                fitness1=fitness2;

            //if(1>(rand() % 20))
            //{
            //  flip(rand() % 150);
            //  fitness1=eval(vec);
            //}

            if (fitness2 == 100)
                break;

            //cout << "fitness = " << fitness2 << endl;
        }
        cout << "fitness = " << fitness1 << endl;
    }
}
```

Below I provided last portion of output for evalCPP.o (top) and evalCPP1.o (below).

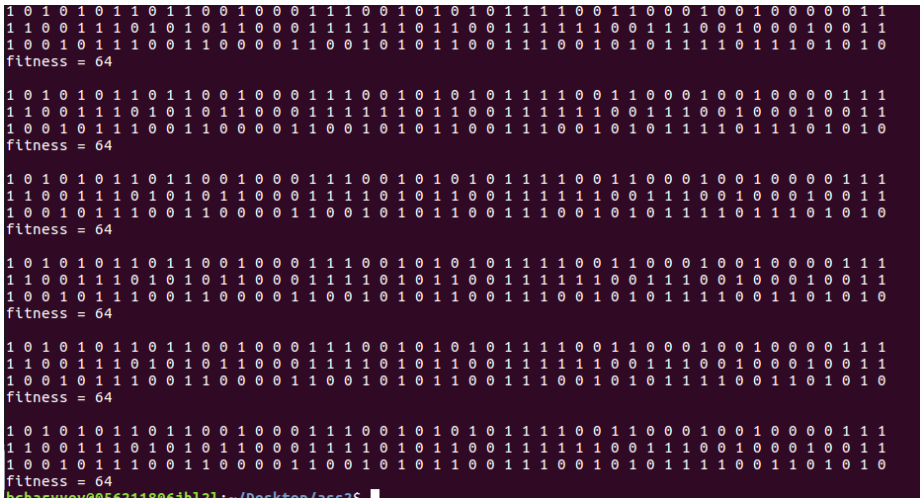
```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
fitness = 32

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
fitness = 32

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
fitness = 32

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
fitness = 32

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
fitness = 32

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
fitness = 32
fitness = 32
bcharyyev@056211806ibl21:~/Desktop/ass2$
```

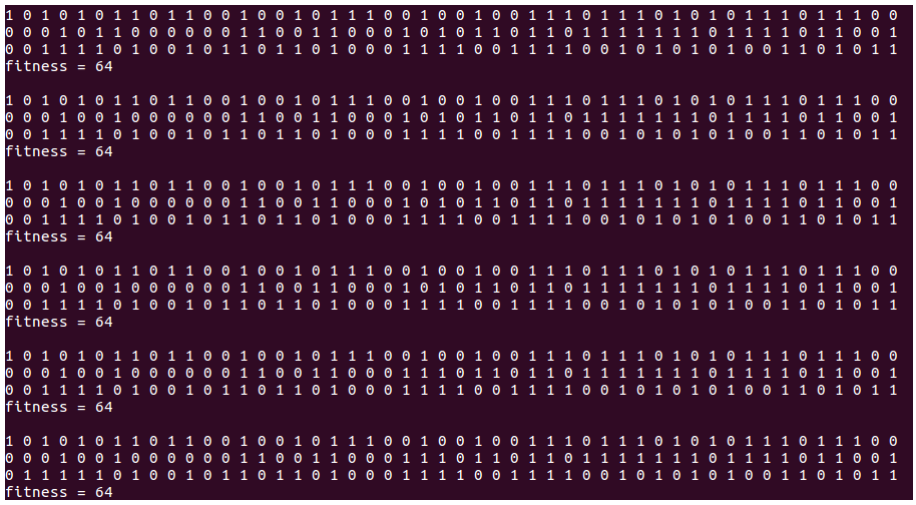
```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
fitness = 32

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
fitness = 32

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
fitness = 32

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
fitness = 32

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
fitness = 32
```

Then I tried randHillClimber shown in class with small probability I tried to make random jumps. And on average I was able to get 64.

Below I provided last portion of output for evalCPP.o (top) and evalCPP1.o (below).

```
void randomHillClimber(int iteration)
{
    int fitness1=0;
    int fitness2=0;

    for(int i=0; i<iteration; i++)
    {
        int r=rand() % 150;
        flip(r);
        fitness2=eval(vec);

        if(fitness1>fitness2) //&& (99>rand() % 100))//revert
        {
            flip(r);
        }
        else
            fitness1=fitness2;

        if (fitness2 == 100)
            break;

        cout << "fitness = " << fitness1 << endl;
    }
    printer(0,150);
}
```





Then I tried to start my HillClimber from random locations(pivot) and flip both sides of pivot and expand. I mean if my initial random pivot is index 65 it will flip both 66 and 64 and check fitness values for both if there is a decrease it will revert change and in next iteration it will check 67 and 63 and will go on like this. Again I was able to get on average 64 because this approach a little bit similar to randomHillClimber.

Below I provided last portion of output for evalCPP.o (top) and evalCPP1.o (below).

```
void locationHillClimbers(int number)
{

    int fitness1=0;
    int fitness2=0;
    for(int i=0; i<number; i++)
    {
        int initial=rand() % 150;

        int a=initial;
        int b=initial;

        for(; a<150 || b>-1; a++, b--)
        {

            if(a<150)
            {
                flip(a);
                fitness2=eval(vec);

                if(fitness1>fitness2)//revert
                    flip(a);
                else
                    fitness1=fitness2;
            }

            if(b>-1)
            {
                flip(b);

                fitness2=eval(vec);

                if(fitness1>fitness2)//revert
                    flip(b);
                else
                    fitness1=fitness2;
            }
            cout << "fitness = " << fitness1 << endl;
        }
    }
}
```

```
0 0 0 1 1 0 0 1 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1 0 1
0 0 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1
0 1 1 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 1 0
fitness = 64

0 0 0 1 1 0 0 1 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1 0 1
0 0 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1
0 1 1 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 1 0 1 0
fitness = 64

0 0 0 1 1 0 0 1 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1 0 1
0 0 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1
0 1 1 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 1 0
fitness = 64

0 0 0 1 1 0 0 1 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1 0 1
0 0 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1
0 1 1 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 0 0
fitness = 64

0 0 0 1 1 0 0 1 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1 0 1
0 0 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1
0 1 1 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 0 1
fitness = 64
```

```
1 0 0 1 0 1 1 0 1 1 1 0 1 0 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0
1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1
0 1 1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 1 1
fitness = 64

1 0 0 1 0 1 1 0 1 1 1 0 1 0 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0
1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1
0 1 1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 1 1
fitness = 64

1 0 0 1 0 1 1 0 1 1 1 0 1 0 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0
1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1
0 1 1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 1 1
fitness = 64

1 0 0 1 0 1 1 0 1 1 1 0 1 0 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0
1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1
0 1 1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 1 1
fitness = 64

1 0 0 1 0 1 1 0 1 1 1 0 1 0 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0
1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1
0 1 1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 1 1
fitness = 64
```

Then I tried to reduce dimension of search space. I mean, I assumed that I have search space with dimension of 150 and each dimension has range of [0,1]. Then I tried to combine *N dimensions* and obtain new search space with *150/N dimension.* For example if *N* is 10 I am assuming that digits between 0^{th} and 9^{th} indexes are one dimension and 10^{th} to19^{th} another dimension. And I make exhaustive search on one dimension and take the best fitting value then again I do exhaustive search on next dimension and take best possible value then if obtained configuration of bits in second dimension gives better results combined with configuration of first dimension I keep the changes otherwise I revert the changes for my second dimension. However this did not give me results expected so I was able to get on average 32 for N=5,10 and 15.

In next page I provided last portion of output for evalCPP.o and evalCPP1.o

```
void dimensionReduced(int offset, int size)
{
    double fitnessBefore=0,fitnessAfter=0;

    int partitions=150/size;
    int array[partitions];

    for(int i=offset,j=0; i<150; j++)
    {   array[j]=i;
        i=i+size;
    }
    int counter=0;
    for(int j=0; j<partitions; j++)
    {
        for(int k=0; k<partitions; k++)
        {
            //cout<<array[(k+j)%partitions]<<" ";
            int startingPoint=array[(k+j)%partitions];

            for(int c = 0; c < 150; c++)
                vec[c] = 0;

            for(int i=startingPoint; i<150; i=i+size)
            {
                partition(i,size);
                fitnessAfter=eval(vec);

                if(fitnessBefore>fitnessAfter)//revertchanges
                {
                    for (vector<int>::iterator it = maxdata.begin(); it != maxdata.end(); ++it)
                        flip(*it);
                }
                else
                {
                    fitnessBefore=fitnessAfter;
                }
            }
            for(int i=0; i<startingPoint; i=i+size)
            {
                partition(i,size);
                fitnessAfter=eval(vec);

                if(fitnessBefore>fitnessAfter)//revertchanges
                {
                    for (vector<int>::iterator it = maxdata.begin(); it != maxdata.end(); ++it)
                        flip(*it);
                }
                else
                {
                    fitnessBefore=fitnessAfter;
                }
            }

            if(fitnessBefore >100)
                return;
            printer(0,150);
            cout<<"counter="<<counter<<"\tfitnessBefore="<<fitnessBefore<<endl;
            counter++;
        }
        cout<<endl;
    }
}
```

```
void printCombination(int arr[], int n, int r)
{
    int data[r];
    double maxFitness=0;
    combinationUtil(arr, data, 0, n-1, 0, r,maxFitness);
}

void partition(int offset, int size)
{
    int array[size];

    for(int i=offset,j=0; i<offset+size; i++,j++)
        array[j]=i;


    for(int i=1; i<=size;i++)
    {
        printCombination(array,size,i);

        //cout<<"<<<<<<<<<<<<<>>>>>>>>>>>>>"<<endl;
    }
    for (vector<int>::iterator it = maxdata.begin(); it != maxdata.end(); ++it)
        flip(*it);
    //printer(offset,offset+size);

}
```

```
vector<int> maxdata;
void combinationUtil(int arr[], int data[], int start, int end, int index, int r, int maxFitness)
{
    if (index == r)
    {
        for (int j=0; j<r; j++)
        {
           // cout<<data[j]<<" ";
            flip(data[j]);
        }
        double fitness=eval(vec);

        if (fitness<maxFitness)
        {
            for (int j=0; j<r; j++)
                flip(data[j]);
        }
        else
        {
            maxdata.clear();
            for (int j=0; j<r; j++)
                maxdata.push_back(data[j]);

            for (int j=0; j<r; j++)
                flip(data[j]);

            //cout<<"maxFitness"<<fitness<<endl;
            maxFitness=fitness;
        }
        return;
    }
    for (int i=start; i<=end && end-i+1 >= r-index; i++)
    {
        data[index] = arr[i];
        combinationUtil(arr, data, i+1, end, index+1, r,maxFitness);
    }
}
```

Interestingly for all values of N=5, 10, 15 and for both evalCPP.o and evalCPP1.o output converges to output provided below.

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
fitness=32

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
fitness=32

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
fitness=32

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
fitness=32

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
fitness=32
```

Then started entering values manually hoping to find some pattern I was able to get 0 but was not able to get more than 64.

Then I remembered that in class professor mention that it is hard problem and you will see it in the future. So I started to check lecture slides for next classes and observed that discussion about "Designing parity checker" resembles over problem furthermore I remembered that professor mentioning it is design problem. Thus, I tried to understand how this problem is solved unfortunately I was not able to understand how we can solve it with HillClimber. I checked some research papers(which also points to Phd dissertation of professor) however I was not able to come up with solution.

I think my approaches failed because, it is design problem. I think we should consider whole 150 bits at once instead of finding best value for small portion of 150 bits and trying to come up with best value for rest of the bits based on value of that small portion (like I did in Hillclimber where I tried to reduce dimensions).

From my understanding in design problems there are lots of local optimum values, thus my hill climbers are not able to overcome this problem.

My hill-climbers have different strength and weaknesses. For example RandomHill climber is straight forward to implement and understand however if our problem is really easy then it may do excessive computations which are not necessary. SimpleHillclimber is good for problems with single hill however if there are multiple hills it may stuck in local optimal. But starting hill climber from multiple random locations can overcome this problem. Reducing dimension in hill-climber algorithm is good for problems where there exist some patterns however, if there are not any patterns it may give really bad results.