

## Genetic Algorithm using MapReduce

The genetic algorithm program was designed to generate an expression which will represent a target number. Each digit and operator in the expression is represented in 4-bits and the series of bits form a chromosome. The program randomly generates an expression and performs the genetic transformations i.e. selection, crossover and mutation, to produce a chromosome with good fitness score. The solution chromosome is a valid chromosome where number is followed by an operator and the fitness score is 1, i.e. it is the equivalent to the target number.

Modifying the various factors of the reproduction phase of genetic algorithm, I analyzed –

- 1) Higher the size of population, less Generations required to find the solution chromosome.
- 2) Longer the Chromosome length, more generations
- 3) Lower mutation rate, fitness Score closer to 1
- 4) 100% Crossover rate, performance of genetic algorithm reduces

To design a parallelized approach for this problem via Hadoop, I have analyzed that I would need to split the reproduction phase of the algorithm in such a way that the reproduction steps are performed on the local data. After studying the references [1] and [2], I have decided that I would be implementing a compact genetic algorithm design based on Coarse-grained Parallelization Model or Island Model.

In this model, the whole population is divided into islands in the Mapper Phase depending on the number of reducers running and the Reducer performs the selection of two chromosomes, crossover between them, mutation of the genes and then identify the solution chromosome. In my code, the solution chromosome will be emitted in a solutionChromosome.txt. Multiple chains of map reduce job will be generated to achieve this goal.

The flow my MapReduce Algorithm is –

### GaDriver.java

- 1) Read the input arguments – TargetNumber, NumOfReducers, PopulationSize

```
Configuration conf = new Configuration();
GenericOptionsParser parser = new GenericOptionsParser(conf, args);
args = parser.getRemainingArgs();
int target = Integer.parseInt(args[1]);
conf.set("targetNumber" , args[0]); // targetNumber
int popSize = Integer.parseInt(args[1]); // Must be even, coz we select two chromosomes from this
conf.set("popSize" , args[1]);
int tasksNum = Integer.parseInt(args[2]); // Number of tasks
conf.set("tasks" , args[2]);
```

- 2) Since the population will be divided among the reducers must be even, validate the input arguments such that the division of population among reducers is satisfactory.

```
if(!validNumberOfTasks(popSize, tasksNum)){
    System.err.println("Number of Reducer Tasks will not evenly divide the population : " + (popSize/tasksNum));
    return;
}

private static boolean validNumberOfTasks(int popSize, int tasksNum) {
    if(popSize%tasksNum == 0)
        if((popSize/tasksNum)%2==0)
            return true;
    return false;
}
```

- 3) If valid, generate the initial pool of chromosomes equal to the population size. The file will have "ChromoNumber,binaryChromoString,fitnessScore"

```

public static boolean generatePopulation(int poolSize, int target) throws IOException, URISyntaxException{
    ArrayList<Chromosome> pool = initializePool(poolSize, target);

    File popFile = new File("ga/input-0/population.txt");
    if(popFile.exists()){
        popFile.delete();
    }

    try (Writer writer = new BufferedWriter(new OutputStreamWriter(
        new FileOutputStream(popFile)))) {
        for(Chromosome c: pool) {
            writer.write(poolSize-- + "," + c.getChromo() + "," + Double.toString(c.getFitnessScore())+"\n");
        }
        return true;
    }
    catch(Exception e){
        System.err.println("Exception : " + e.getMessage());
        return false;
    }
}

public static ArrayList<Chromosome> initializePool(int poolSize, int target){
    ArrayList<Chromosome> pool = new ArrayList<Chromosome>(poolSize);

    // Generate unique chromosomes in the pool
    for (int x=0;x<poolSize;x++)
        pool.add(new Chromosome(target));

    return pool;
}

```

- 4) Check if whether solutionChromosome.txt file exists or not. If not, continue with the generation steps otherwise stop the chaining of the map reduce jobs.

```

while(!solutionFound(conf)){

    Job job = new Job(conf, "genetic_algorithm_for_target_generation-"+gen);
    job.setJarByClass(GaDriver.class);
    // specify output types
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    job.setNumReduceTasks(tasksNum);

    FileInputFormat.setInputPaths(job, new Path("ga/input-"+gen));
    gen++;
    FileOutputFormat.setOutputPath(job, new Path("ga/input-"+gen));

    // specify a mapper
    job.setMapperClass(GaMapper.class);
    //specify a partitioner
    job.setPartitionerClass(GaPartitioner.class);
    // specify a reducer
    job.setReducerClass(GaReducer.class);

    System.out.println(job.waitForCompletion(true));

}

```

### GaMapper.java

In the Mapper function, I will decide the island for the chromosome. Chromosome belong to same island will have same key so that they can be processed by one reducer.

- 1) If the number of reduces tasks are greater than 1, then we will need to decide the islands. Otherwise it will always to the same single reducer.

- 2) To decide an island, I use the chromosome number to decide the island it belong to it.  
 For example – if we have 100 population size and 5 reduces then each island will be of size 20.  
 The island slots generated for this will be (0-19,20-39,40-59,60-79,80-99) Consider that we got chromosome number 61, then we would need to put in on island 3 since it will be processed by reducer number 3.

```
protected void map(LongWritable key, Text chromosomeString, Context context) throws IOException, InterruptedException {
    int popSize = Integer.parseInt(context.getConfiguration().get("popSize"));
    int numReduceTasks = Integer.parseInt(context.getConfiguration().get("tasks"));
    int outkey = 0;

    if (numReduceTasks > 1){
        ArrayList<String> slots = generateIslandSlots(numReduceTasks, popSize);
        int chromoNumber = Integer.parseInt(chromosomeString.toString().trim().split(",")[0].trim());
        outkey = findTheIslandReducer(chromoNumber, slots);
    }

    context.write(new Text(Integer.toString(outkey)), chromosomeString);
}
```

```
public ArrayList<String> generateIslandSlots(int numOfReducers, int popSize){
    int numberOfSlots = numOfReducers;
    int slotSize = popSize/numberOfSlots;
    ArrayList<String> slotList = new ArrayList<>();
    int lastLowerBound = -1;
    while(numberOfSlots > 0){
        slotList.add(Integer.toString(lastLowerBound + 1) + "-" + Integer.toString(lastLowerBound+slotSize));
        lastLowerBound = lastLowerBound + slotSize;
        numberOfSlots--;
    }
    return slotList;
}

public int findTheIslandReducer(int chromoNumber, ArrayList<String> slots){
    for(int i = 0 ; i < slots.size() ; i++){
        String slot = slots.get(i);
        int lowerBound = Integer.parseInt(slot.trim().split("-")[0]);
        int upperBound = Integer.parseInt(slot.trim().split("-")[1]);
        if(chromoNumber >= lowerBound && chromoNumber <=upperBound){
            return i;
        }
    }

    return 0;
}
```

- 3) The output of this Mapper will be  
 Key -> Reducer Number  
 Value -> "chromoNumber,binaryChromoString,fitnessScore"

### GaPartitioner.java

By default, Hadoop uses the value of (HashKey%numberOfReducers) to decide the reducer that will work on that set of values associated with the key. I have override written a custom Partitioner that will return the key number the reducer number, since I am setting the key number in the Mapper. Therefore, I am ensuring that the Reducer-0 processes all the chromosome associated with the Island-0 and nothing else.

```

public class GaPartitioner extends Partitioner<Text, Text> {

    @Override
    public int getPartition(Text key, Text value, int numReduceTasks) {

        return Integer.parseInt(key.toString());

    }

}

```

### GaReducer.java

- 1) In the reduce, I convert the chromosome string into an actual Chromosome Object and add it into the pool of chromosome which will be later used. Also, I have added the chromosome number in a different list as we need to emit the value from the reducer in the same format as the mapper generates it.

```

int target = Integer.parseInt(context.getConfiguration().get("targetNumber"));
int popSize = Integer.parseInt(context.getConfiguration().get("popSize"));
int numReduceTasks = Integer.parseInt(context.getConfiguration().get("tasks"));

ArrayList<Chromosome> pool = new ArrayList<Chromosome>(popSize/numReduceTasks);
ArrayList<String> chromoNumbers = new ArrayList<String>(popSize/numReduceTasks);

for(Text chromosomeString : arrayValues){
    String[] values = chromosomeString.toString().trim().split(",");
    String chromo = values[1].trim();
    double fitnessScore = Double.parseDouble(values[2].trim());
    Chromosome c = new Chromosome(chromo,fitnessScore);
    pool.add(c);
    chromoNumbers.add(values[0].trim());
}

```

The chromoString is like - 59,0100110100110001101011011101110001000011,0.010526315789473684  
 So I split it on "," and get the first part as chromoNumber, second as binaryString and third as fitnessScore. Also, I have added a new constructor in the Chromosome.java class with return a chromosome object when I give it second and third part.

```

public Chromosome(String chromo, double fitnessScore)
{
    this.chromo.append(chromo);
    this.fitnessScore = fitnessScore;
    this.total = addUp();
}

```

- 2) The Reducer then perform the steps of genetic algorithm on this pool of chromosomes. It randomly select two chromosomes from this pool. Crossover the bits in the selected chromosome. Mutation the bits in the child chromosomes and then validate if the new child score is valid and its total score is equal to target number.

```

// Loop until the pool has been processed
for(int x=pool.size()-1;x>=0;x--=2) {
    // Select two members
    Chromosome n1 = selectMember(pool);
    Chromosome n2 = selectMember(pool);

    // Cross over and mutate
    n1.crossOver(n2);
    n1.mutate();
    n2.mutate();

    // Re-score the nodes
    n1.scoreChromo(target);
    n2.scoreChromo(target);

    // Check to see if either is the solution
    if (n1.getTotal() == target && n1.isValid()) {
        writeSolution(n1);
    }
    if (n2.getTotal() == target && n2.isValid()) {
        writeSolution(n2);
    }

    // Add to the new pool
    context.write(new Text(), new Text(chromoNumbers.remove(chromoNumbers.size()-1).trim() + "," + n1.getChromo() + ",");
    context.write(new Text(), new Text(chromoNumbers.remove(chromoNumbers.size()-1).trim() + "," + n2.getChromo() + ",");
}

```

- 3) If the child chromosome is the solution chromosome, it writes the solution to solutionChromosome.txt. Otherwise, the reducer emits this new child chromosome as the output.

```

public static void writeSolution(Chromosome solutionChromosome){
    try{
        FileSystem fs = FileSystem.get(new URI("hdfs://localhost:9000"), new Configuration());
        Path popFile = new Path(fs.getWorkingDirectory() + "/ga/solutionChromo.txt");
        if ( fs.exists( popFile )) { fs.delete( popFile, true ); }
        BufferedWriter writer=new BufferedWriter(new OutputStreamWriter(fs.create(popFile,true)));

        writer.write(solutionChromosome.getChromo() + "|" +
            Double.toString(solutionChromosome.getFitnessScore()) +
            "|DecodedChromo" + solutionChromosome.decodeChromo() + "\n");

        writer.close();
        fs.close();
    }
    catch(Exception e){
        System.err.println("Exception : writeSolution" + e.getMessage());
    }
}

```

- 4) The output of the reducer servers as new pool of chromosomes and is the input to the next Mapper cycle.

### Chromosome.java

I have made a change in the scoreChromo function of this class. Since the fitness score the inverse of the difference between the target number and chromosome total, when the difference is 0 I have normalized it to fitness score of 1.

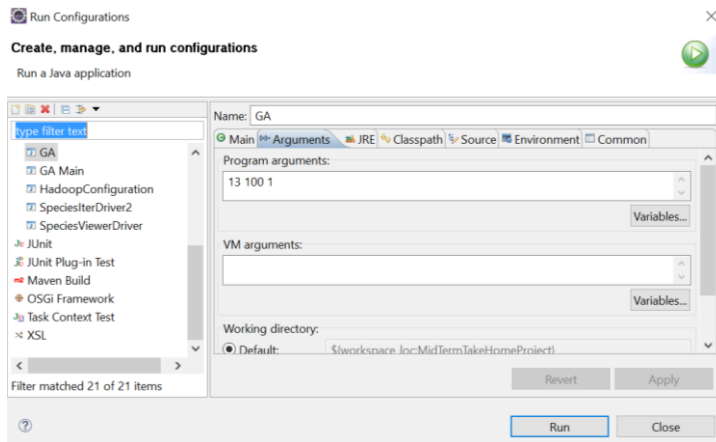
```

// Scores this chromo
public final void scoreChromo(int target) {
    this.total = addUp();
    if (this.total == target)
        this.fitnessScore = 1;
    else
        this.fitnessScore = (double)1 / (target - this.total);
}

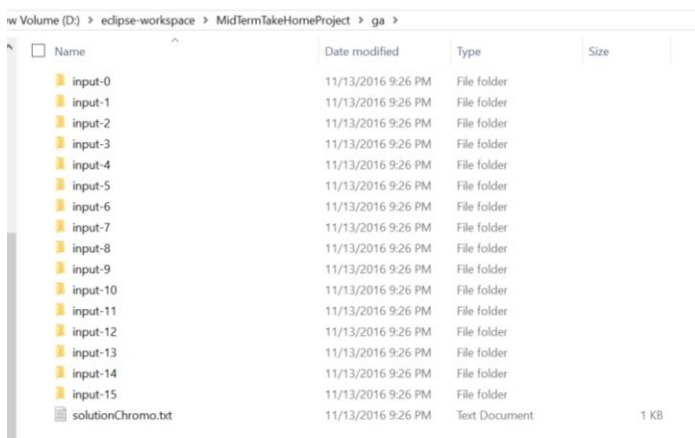
```

### StandAlone Mode –

In Stand Alone Mode, the number of reducers are always 1. For the following arguments –



TargetNumber=13, PopulationSize=100



The total generations were 15 to find solution Chromosome

solutionChromo.txt - Notepad

File Edit Format View Help

|00011010011110100101|1.0|DecodedChromo1+7+5

### PsuedoMode –

To actually parallelize my code, I decided to run my code in Psuedo Mode. I have first compiled my java files and generated the jar.

```
ManasiLaddha@Manasi-Desktop /cygdrive/c/hadoop
$ javac -classpath "lib/commons-cli-1.2.jar;hadoop-0.20.2-core.jar" -d ga/classes ga/java/*

ManasiLaddha@Manasi-Desktop /cygdrive/c/hadoop
$ jar cvf ga_target.jar -C ga/classes .
added manifest
adding: GA/(in = 0) (out= 0)(stored 0%)
adding: GA/Chromosome.class(in = 3232) (out= 1897)(deflated 41%)
adding: GA/GaDriver.class(in = 5996) (out= 3013)(deflated 49%)
adding: GA/GaMapper.class(in = 2704) (out= 1184)(deflated 56%)
adding: GA/GaPartitioner.class(in = 716) (out= 373)(deflated 47%)
adding: GA/GaReducer.class(in = 5879) (out= 2768)(deflated 52%)
```

I ran the command with target number is 13 and the population size 500 so each island will have 100 chromosomes.

```
ManasiLaddha@Manasi-Desktop /cygdrive/c/hadoop
$ bin/hadoop jar ga_target.jar GA.GaDriver 13 500 5
```

It took me 3 iterations to find the solution chromosome –

```
ManasiLaddha@Manasi-Desktop /cygdrive/c/hadoop
$ bin/hadoop-daemon.sh start namenode
starting namenode, logging to /cygdrive/c/hadoop/bin/../logs/hadoop-ManasiLaddha-namenode-Manasi-Desktop.out

ManasiLaddha@Manasi-Desktop /cygdrive/c/hadoop
$ bin/hadoop-daemon.sh start datanode
starting datanode, logging to /cygdrive/c/hadoop/bin/../logs/hadoop-ManasiLaddha-datanode-Manasi-Desktop.out


ManasiLaddha@Manasi-Desktop /cygdrive/c/hadoop
$ bin/hadoop-daemon.sh start secondarynamenode
starting secondarynamenode, logging to /cygdrive/c/hadoop/bin/../logs/hadoop-ManasiLaddha-secondarynamenode-Manasi-Desktop.out

ManasiLaddha@Manasi-Desktop /cygdrive/c/hadoop
$ bin/hadoop-daemon.sh start jobtracker
starting jobtracker, logging to /cygdrive/c/hadoop/bin/../logs/hadoop-ManasiLaddha-jobtracker-Manasi-Desktop.out

ManasiLaddha@Manasi-Desktop /cygdrive/c/hadoop
$ bin/hadoop-daemon.sh start tasktracker
starting tasktracker, logging to /cygdrive/c/hadoop/bin/../logs/hadoop-ManasiLaddha-tasktracker-Manasi-Desktop.out

ManasiLaddha@Manasi-Desktop /cygdrive/c/hadoop
$ bin/hadoop fs -ls ga/
Found 5 items
drwxr-xr-x - manasi-desktop\manasiladdha supergroup          0 2016-11-13 19:29 /user/manasi-desktop/manasiladdha/ga/input-0
drwxr-xr-x - manasi-desktop\manasiladdha supergroup          0 2016-11-13 19:30 /user/manasi-desktop/manasiladdha/ga/input-1
drwxr-xr-x - manasi-desktop\manasiladdha supergroup          0 2016-11-13 19:30 /user/manasi-desktop/manasiladdha/ga/input-2
drwxr-xr-x - manasi-desktop\manasiladdha supergroup          0 2016-11-13 19:31 /user/manasi-desktop/manasiladdha/ga/input-3
-rw-r--r-- 1 manasi-desktop\manasiladdha supergroup         44 2016-11-13 19:31 /user/manasi-desktop/manasiladdha/ga/solutionChromo.txt
```

The solutionChromosome.txt

 solChromo.txt - Notepad

File Edit Format View Help

|00101100001110100111|1.0|DecodedChromo2\*3+7

The JobTracker Sand –

maps	reduces	total submissions	nodes	map task capacity	reduce task capacity	avg. task time	distributed nodes
0	0	5	1	2	2	4.00	0

Scheduling Information

Queue Name	Scheduling Information
default	N/A

Filter (Jobid, Priority, User, Name)  
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
<a href="#">job_201611132235_0001</a>	NORMAL	manasi-desktop\manasiladdha	genetic_algorithm_for_target_generation-0	<div>100.00%</div>	1	1	<div>100.00%</div>	5	5	NA
<a href="#">job_201611132235_0002</a>	NORMAL	manasi-desktop\manasiladdha	genetic_algorithm_for_target_generation-1	<div>100.00%</div>	5	5	<div>100.00%</div>	5	5	NA
<a href="#">job_201611132235_0003</a>	NORMAL	manasi-desktop\manasiladdha	genetic_algorithm_for_target_generation-2	<div>100.00%</div>	5	5	<div>100.00%</div>	5	5	NA
<a href="#">job_201611132235_0004</a>	NORMAL	manasi-desktop\manasiladdha	genetic_algorithm_for_target_generation-3	<div>100.00%</div>	5	5	<div>100.00%</div>	5	5	NA

## **References**

- [1]Scaling Genetic Algorithms using MapReduce - Abhishek Verma, XavierLlor'a, David E. Goldberg, Roy H. Campbell
- [2] A Framework for Genetic Algorithms Based on Hadoop - Filomena Ferrucci, M-Tahar Kechadi, Pasquale Salza, Federica Sarro