Bangladesh University of Engineering & Technology
Department of Computer Science & Engineering
CSE 402: Artificial Intelligence Sessional
Session July 2016

Expression Generation with Genetic Algorithm

---

1 Introduction

In this assignment you have to implement Genetic Algorithm to solve a problem of expression generation. An expression or mathematical expression is a finite combination of symbols that is well-formed according to rules that depend on the context. In this problem, we will only consider simple expressions that contain numbers and mathematical operators. For simplicity, the numbers are in range $[1, 9]$, and operator are $\{+, -, \times\}$. So, an expression can be evaluated to get a value in range $[-\infty, \infty]$. For example, consider a simple expression.

$$8 \times 7 - 2 \times 9 \times 4 - 1 + 7 \times 7 = 32$$

This expression evaluates to the value 32. It is easy to evaluate a value from an expression. In this assignment, the task is just the opposite. You have to generate an expression from a given value (i.e. given a value 32, generate an expression that evaluates to 32).

Note that, different expressions can evaluate to a same value. It will be mentioned how many digits and operators you need in the generated expression. In the above expression, there are 8 digits and 7 operators. The number of operator will be always 1 less than the number of digits. The input will contain the number of operators and the desired value. From this information, you have to generate the best expression using Genetic Algorithm that evaluates to the value. It is possible that an expression which evaluates to the target value is not generated in possible amount of time. Here, best expression means the expression that evaluates to the closest of the desired value.

2 Implementation

Genetic Algorithm is given in Figure **??**.

2.1 Representation

Each individual is a vector of length $l = d + o$, where d is the number of digits in the expression and o is the number of operators. You can keep two different

```
Algorithm  The Genetic Algorithm (GA)
popsize ← desired population size
P ← {}
for popsize times do
    P ← P ∪ {new random individual}
Best ← □
repeat
    for each individual P_i ∈ P do
        AssessFitness(P_i)
        if Best = □ or Fitness(P_i) > Fitness(Best) then
            Best ← P_i
    Q ← {}
    for popsize/2 times do
        Parent P_a ← SelectWithReplacement(P)
        Parent P_b ← SelectWithReplacement(P)
        Children C_a, C_b ← Crossover(Copy(P_a), Copy(P_b))
        Q ← Q ∪ {Mutate(C_a), Mutate(C_b)}
    P ← Q
until Best is the ideal solution or we have run out of time
return Best
```

Figure 1: Genetic Algorithm

arrays for keeping the digits and operators or use a string to keep them all for implementation purpose. But each individual is to considered as a vector including digits and operators in sequence as in the actual mathematical expression.

2.2 Initial Population

Let, the population size be N. Randomly generate N number of individuals of length l.

2.3 Fitness Function

Generally better fitness means a better individual that is closer to optimal solution. In this problem, fitness f_i for an individual i is the absolute difference between the desired value v and the value i evaluates.

$$f_i = -|v - evaluate(i)|$$

Here, better fitness means higher f_i value. Consider two individuals i and j. The fitness values are f_i and f_j, respectively. If $f_i > f_j$, then i is a better individual than j.

2.4 Selection

We will use Tournament Selection where you will choose t number of individuals from the population randomly, and select the best individual among the t individuals according to fitness function. Use $t = 5$ for the experiment.

Algorithm *Tournament Selection*

$P \leftarrow$ population
$t \leftarrow$ tournament size, $t \geq 1$

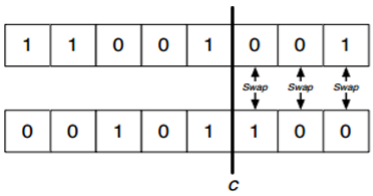
Best $\leftarrow$ individual picked at random from P with replacement
for i from 2 to t **do**
    Next $\leftarrow$ individual picked at random from P with replacement
    if Fitness(*Next*) $>$ Fitness(*Best*) **then**
        Best $\leftarrow$ *Next*
return *Best*

Figure 2: Tournament Selection

2.5 Crossover

Two individuals are selected from the population for crossover, and are called parents. Let, P_a, P_b are two parents. Two types of crossovers are taken into consideration with 50% probability for each.

- **One Point Crossover:** It randomly picks an integer c in the range $[1, l]$, inclusive, and swaps all the indexes greater than c.



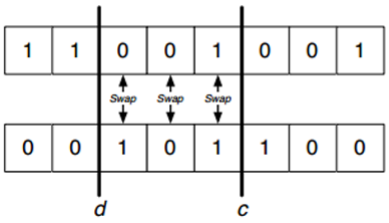
Algorithm *One-Point Crossover*

$\vec{v} \leftarrow$ first vector $\langle v_1, v_2, ...v_l \rangle$ to be crossed over
$\vec{w} \leftarrow$ second vector $\langle w_1, w_2, ...w_l \rangle$ to be crossed over

$c \leftarrow$ random integer chosen uniformly from 1 to l inclusive
for i from c to l **do**
    Swap the values of v_i and w_i
return $\vec{v}$ and $\vec{w}$

Figure 3: One Point Crossover

- **Two Point Crossover:** It randomly picks two numbers c and d in the range $[1, l]$, inclusive, and swap the indexes between them.



Algorithm *Two-Point Crossover*

$\vec{v} \leftarrow$ first vector $\langle v_1, v_2, ...v_l \rangle$ to be crossed over
$\vec{w} \leftarrow$ second vector $\langle w_1, w_2, ...w_l \rangle$ to be crossed over

$c \leftarrow$ random integer chosen uniformly from 1 to l inclusive
$d \leftarrow$ random integer chosen uniformly from 1 to l inclusive
if $c > d$ **then**
    Swap c and d
for i from c to $d - 1$ **do**
    Swap the values of v_i and w_i
return $\vec{v}$ and $\vec{w}$

Figure 4: Two Point Crossover

2.6 Mutation

For each element of the vector, flip a coin of a certain probability p (often $1/l$, where l is the length of the vector). Each time the coin comes up heads, check whether it is a digit or an operator. If it is a digit, then apply Random Walk Mutation as shown in Figure **??**. If it is an operator, change it to one of the other two operators with 50% probability.

Algorithm 42 *Random Walk Mutation*

$\vec{v} \leftarrow$ integer vector $\langle v_1, v_2, ...v_l \rangle$ to be mutated
$p \leftarrow$ probability of randomizing an integer ▷ Perhaps you might set p to $1/l$ or lower
$b \leftarrow$ coin-flip probability ▷ Make b bigger if you have many legal integer values so the random walks are longer
for i from 1 to l **do**
  if $p \geq$ random number chosen uniformly from 0.0 to 1.0 inclusive **then**
    repeat
      $n \leftarrow$ either a 1 or -1, chosen at random.
      if $v_i + n$ is within bounds for legal integer values **then**
        $v_i \leftarrow v_i + n$
      else
        $v_i \leftarrow v_i - n$
    until $b <$ random number chosen uniformly from 0.0 to 1.0 inclusive
return $\vec{v}$

Figure 5: Random Walk Mutation

3 Input

Input consists two integers. The first integer denotes the number of operators the expression contains. The second number denotes the desired value that the expected expression evaluates to.

4 Experiment and Results

Show the best individual (expression) and generate a table as shown for each of the input given for different N.

Table 1: Experiment

Target value	No. of operators	popsize, N	No. of generations	Best-Fitness

5 Conclusion

For any query or confusion please feel free to email me at ahmaadsabbir@gmail.com.

Prepared by-
Sabbir Ahmad
Lecturer
Department of CSE, BUET
ahmadsabbir@cse.buet.ac.bd
ahmaadsabbir@gmail.com
http://teacher.buet.ac.bd/ahmadsabbir