An Introduction to Genetic Algorithms

Rajkumar Pant
Aerospace Engineering Department
IIT Bombay

OUTLINE

- Historical background
- Working Principles of GA
- GA operators
- Concept of schemata in GA
- GA v/s traditional methods
- Drawbacks of GA
- GA for constrained optimization
- Modern developments

OPTIMIZATION TECHNIQUES

$\mathbf{x} = x_i,\ i = 1, n$	'n' Design Variables
$f(\mathbf{x})$	Objective function
$U_i \leq x_i \leq L_i,\ i = 1,n$	Bounds on Design Variables
$g_k(\mathbf{x}) \leq 0,\ k = 1,m$	'm' inequality Constraints
$h_j(\mathbf{x}) = 0,\ j = 1,p$	'p' equality Constraints

Trivial Solution — Exhaustive Search

Initial Point >>>>>>>>>>>>>> Trial Point

Deterministic Methods	Direct mathematical link
Stochastic Methods	Element of randomness

Historical Background

- Mechanics of natural genetics and evolution
  - Survival of the fittest
  - Evolution of the species
- John Holland, University of Michigan
  - *Adaptation in natural and artificial systems*, 1975
- David Goldberg, University of Illinois
  - *Genetic Algorithms in Search, Optimization and Machine Learning*,1989
- ICGA, 1985, 87, 89, …….

Working Principles of GA

- Unconstrained Optimization
  - Maximize f(**x**), **x** = x_i, i = 1, n, $U_i \le x_i \le L_i$, i = 1,n

- Coding of **x** in string structures
  - Not absolutely essential (Real GA)
  - Usually Binary Coding
    - each x_i coded in k sub-strings of length l_i, k = $2^{\{l_i\}}$
    - sub-string element $s_k(m) \in (0,1)$, m = 0, l_i-1
      - $s_k = s_{l-1}s_{l-2} \ldots\ldots s_2 s_1 s_0$
    - Discrete values of x_i mapped
      - $x_i = L_i$, $DV(s_i) * (U_i - L_i) / (2^{\{l_i\}} - 1)$, where
        $DV(s_k)$ = Decoded value of $(s_k) = \sum 2^{m} \cdot s_k(m)$, m = 0, l-1

Example of Binary Coding

- Decoding Procedure
  - $i=1,\ l_1 = 4$
  - k = total number of discrete values of $x_1 = 2^{\{l_1\}} = 2^4 = 16$
  - string $s_k = s_3 s_2 s_1 s_0$
  - $s_k(m) \in (0,1),\ m = 0, 3$
  - $DV(s_k) = \sum 2^{m \cdot} s_k(m),\ m = 0, 3$
- Accuracy
  - $\approx (U_i - L_i) / \ 2^{\{l_i\}}$
  - Exponential increase with string length

k	s_k	DECODING OF s_k	$x_1(k)$
1	0000	$(0)*2^3 + (0)2^2 + (0)*2^1 + (0)*2^0$	0
2	0001	$(0)*2^3 + (0)2^2 + (0)*2^1 + (1)*2^0$	1
3	0010	$(0)*2^3 + (0)2^2 + (1)*2^1 + (0)*2^0$	2
4	0011	$(0)*2^3 + (0)2^2 + (1)*2^1 + (1)*2^0$	3
5	0100	$(0)*2^3 + (1)2^2 + (0)*2^1 + (0)*2^0$	4
6	0101	$(0)*2^3 + (1)2^2 + (0)*2^1 + (1)*2^0$	5
7	0110	$(0)*2^3 + (1)2^2 + (1)*2^1 + (0)*2^0$	6
8	0111	$(0)*2^3 + (1)2^2 + (1)*2^1 + (1)*2^0$	7
9	1000	$(1)*2^3 + (0)2^2 + (0)*2^1 + (0)*2^0$	8
10	1001	$(1)*2^3 + (0)2^2 + (0)*2^1 + (1)*2^0$	9
11	1010	$(1)*2^3 + (0)2^2 + (1)*2^1 + (0)*2^0$	10
12	1011	$(1)*2^3 + (0)2^2 + (1)*2^1 + (1)*2^0$	11
13	1100	$(1)*2^3 + (1)2^2 + (0)*2^1 + (0)*2^0$	12
14	1101	$(1)*2^3 + (1)2^2 + (0)*2^1 + (1)*2^0$	13
15	1110	$(1)*2^3 + (1)2^2 + (1)*2^1 + (0)*2^0$	14
16	1111	$(1)*2^3 + (1)2^2 + (1)*2^1 + (1)*2^0$	15

Working Principles of GA

- Fitness Function **F**(**x**)
  - Directly related to f(**x**)
    - **F**(**x**) = f(**x**) for maximization
    - **F**(**x**) = 1/ (1, f(**x**)) for minimization
- Steps in GA
  - Create initial population (randomly) by concantening sub-strings
  - Evaluate **F**(**x**) of each member, identify best member(s)
  - Create next generation, using GA operations
    - Reproduction
    - Crossover
    - Mutation
  - Iterate till convergence

Reproduction

- Insertion of strings with higher **F**(**x**) in mating pool
  - Strings with low **F**(**x**) may also get selected

- Selection strategies
  - Proportionate
    - probability of selection proportional to **F**(**x**)
      - Roulette-Wheel selection
  - Ranking
    - best of a few strings selected each time
      - Tournament selection

Crossover

- Two strings from mating pool selected (randomly)
- Location(s) of crossover location determined (randomly)
- Bits of Strings interchanged between crossover location(s)

- Single point Crossover

	Parent Strings		Child Strings	
(0)	**0 0 0 | 0**	=====>	**0 0 0 | 1**	**(1)**
(11)	**1 0 1 | 1**	=====>	**1 0 1 | 0**	**(10)**

- Crossover can be beneficial or detrimental
  - Crossover Probability $p_c \approx 0.7$ to 0.9

Mutation

- Flipping of bits of strings at random
  - local search around current solution
  - maintain genetic diversity of population
  - enable search to climb towards global optimum
  - mutation probability $p_m \approx 0.005$ to 0.015

- Illustration with $p_m = 0.10$
  - Original population

0 0 1 0 , 1 0 0 1 , 0 0 0 1 , 1 1 1 1 , 0 1 1 1 , 1 1 0 1 , 1 1 1 1 , 1 0 0 1

(2) (9) (1) (16) (7) (13) (16) (9)

  - After Mutation

0 0 1 0 , 1 0 0 1 , 0 1 0 1 , 1 1 1 1 , 0 1 1 0 , 1 1 0 1, 1 1 1 1 , 1 1 0 1

(2) (9) (5) (16) (6) (13) (16) (13)

GA Operators - Summary

- Reproduction
  - Select ***good*** strings from population (eliminate ***bad*** strings)
- Crossover
  - Recombine strings and (hopefully) create better strings
- Mutation
  - Once in a while, create even better/worse strings

Concept of Schemata in GA

- Schemata
  - number of strings with similarities at certain string positions
  - total schemata for binary representation = 3^l, l = string length
  - defining length $\delta(H)$ = difference in outermost defining points
  - order o(H) = number of fixed point
  - Example for H = 1*0*
    - $\delta(H) = o(H) = 2$
    - strings = 1101, 1100, 1000, 1001
    - occupies 25% of search space

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

- population = 16, but total schemata = 3^4 = 81

Schemata Theorem

$$m(H,t+1) \geq m(H,t)\frac{F(H)}{F_{avg}}\left[1 - p_c\frac{d(H)}{l-1} - p_m o(H)\right]$$

Where,

- $m(H,t)$ = Number of strings of schema H in generation t
- $F(H)$ = Average fitness of strings of schema H
- F_{avg} = Average fitness of the population
- p_c = Probability of crossover
- p_m = Probability of mutation
- l = String length
- $d(H)$ = Defining length of schema H
- $o(H)$ = Order of schema H

Building Block Hypothesis

- Building Blocks
  - Schema with low $d(H)$ & $o(H)$, and $F(H) > F_{avg}$
    - Represent different large, good regions in search space
    - Propagate exponentially with generations
    - Combine with each other vigorously
    - Lead to
      - optimum (near-optimal) solutions

- Implicit Parallelism
  - In each generation
    - n function evaluations
    - n^3 schemata processed
    - without any specific book-keeping !

GA v/s Traditional methods

- Direct Methods
  - Gradients or any auxiliary information not required
  - Can handle discontinuous, Multi-modal functions
- Stochastic formulation
  - Independent of starting point
- Work on coding of variables
  - Domain Discretization
  - General purpose & robust code possible
- Work on a population of points
  - Better chance of converging to global minimum
  - Can identify multiple optimal solutions

Drawbacks of GA

- Coding scheme should be meaningful & appropriate
  - premature convergence
- GA parameters have to be tuned before starting
  - population size, max. generations, p_c, p_m
  - crossover type, selection strategy
- Constraints cannot be implicitly handled
  - penalty function approach (usually)
- Large number of function evaluations
  - Orders of magnitude higher than traditional algorithms

GA for constrained optimization

Maximize F ($\boldsymbol{x}$) = 1/(1+ ***P***($\boldsymbol{x}$)) or

Minimize F($\boldsymbol{x}$) = ***P***($\boldsymbol{x}$), where

$$P(x) = f(x) + \sum_{j=1}^{J} u_j \langle g_j(x) \rangle^2 + \sum_{k=1}^{K} v_k \langle h_k(x) \rangle^2$$

$f(x)$ = objective function

u_j = penalty coefficients for J inequality constraints g_j

v_k = penalty coefficients for K equality constraints h_k

u_j & v_k are usually kept constant during one GA run

Large values can be assigned to u_j & v_k

Other GA operators

- Fitness Function scaling
  - To avoid premature convergence due to
    - dominance by one string
    - presence of several average fit strings
      - transform ***F*** (**x**) to **S**(**x**) = a***F***(**x**) + b ; choose a & b such that
      - best string has a predefined number of copies
      - each string with average fit has only one copy
- Multi-point Crossover
  - Single point crossover is biased in favor of right-most bits
- Uniform Crossover
  - A bit at any location chosen either parent with p = 0.5
  - for e.g. if 1st and 3rd bits are changed, then

(0) 0 0 0 0 =====> 1 0 1 0 (10)

(11) 1 0 1 1 =====> 0 0 0 1 (1)

Recent development in GAs

- Real Coded GA
  - work directly on design variables
- Simultaneous multiple solutions of multi-modal functions
  - divide population using sharing functions
- Multi-criteria optimization
  - pareto-optimal solutions

- Reported applications of GA in literature
  - Gas pipeline layout optimization
  - Job shop scheduling, Routing, Travelling salesperson
  - Aircraft Design and Structural / Aerodynamic optimization
  - Gas Turbine blade design
  - Laminate stacking in Composite structures

Happy computing with GA !!