11.11.09 - 2PM GENETIC ALGORITHMS (M) (1)

* GA → Search Algos. based on the mechanics of natural selection and natural genetics.

↳ Survival of fittest among string structures + structured, Randomized Information Exchange - to result in a Search Algorithm with human search flair.

↳ In Every generation, new creatures / strings are created using bits and pieces of old - Tested for a good measure.

↳ GA → Exploit historical information to speculate on new search points with expected improved performance.

↳ Developed by John Holland et.al - Univ. of Michigan
  ↳ Abstract adaptive process of natural systems
  ↳ Create s/w retaining mechanisms " "

↳ GA - Thrust on robustness - Balance b/w efficiency and efficacy - Capacity for Beneficial change

↳ Robust Systems - Avoid costly redesigns - Longevity of existing systems

↳ Biological systems - self Repair, self Guidance and Reproduction } Key features

↳ Natural systems "more Robust" than Artificial systems
  ↳ 'Adaptation and survival' features

↳ GA finds application where efficient and effective search is required.

↳ Computationally simple, powerful search for improvement

Traditional optimization, Search Methods:

* 3 key search methods are 'Calculus based', 'Enumerative' and 'random'.

* Calculus – Indirect and Direct

* Indirect Method → seek 'local extrema' by solving the usually nonlinear set of equations resulting from setting the gradient of the function (objective) to 'zero'.

* Direct search Methods – 'seek local optima', move in direction related to local gradient.
– 'Hill climbing' – find local best and climb in the steepest possible direction.

Limitations: Both methods are 'local in scope' optima sought is best in a neighborhood of the current point.

↳ Calculus method suits 'single peak' fns. with multi Peak functions the direction (Hill) to climb is not straightforward. 'Global optima can be lost in the local (Higher peak) / (Lower peak) optima Search Strategy.

↳ Calculus methods require existence of derivatives which might not be feasible in all real life situations – Methods based on 'Continuity' and 'Derivative Existence' might not suit all application domains.

- Calculus Methods inherently 'local scope of search' biased.

Enumerative schemes – In a finite search space (or) Discretized Infinite search space – objective function values are evaluated @ every point in space

↳ such schemes are useful only for small (reduced) search spaces – "Generally not preferred for lack of efficiency". eg. Dynamic programming

↳ Random searches over Calculus / Enumerative schemes

↳ Different from 'Randomized' search Technique

eg: GA – search procedure that uses random choice as a tool to guide an exploitative search. (Random choice used as a tool in a directed search strategy)

↳ Simulated Annealing uses random processes in its search for minimal energy states.

"Conventional search Methods are not Robust"

Peak performance could be sacrificed to achieve relatively high performance across a spectrum of problems. Hybrid schemes – Combine local search schemes with ~~random~~ robust schemes is also used.

↳ Achieve relatively higher efficiency across broad spectrum of problems (Combinatorial – Unimodal – Multimodal) as opposed to peak high/low performances.

Goals of optimization:- optimization theory is quantitative study of optima and methods to find them.

"Describe and Attain what is Best"

↳ "Alter Good or Bad"

* optimization → "improve performance towards" some "optimal point / points"

↳ Distinction b/w 'process of improvement' and 'destination / optimum' itself.

↳ Generally optimization focusses on 'Convergence' (optimum) forgetting interim performance. ⟹ Based on Calculus

* But in ~~relatt~~ real life optimum / Goodness is a relative Measure. — Judged in relation to Competition Thus In Real life "Doing better relative to others" is Preferred over "attainment of Best / Convergence to optimum". Goal is to achieve "satisficing" results

GA's V/s Traditional Methods:

(1) GA's work with Coding of parameter set and not the original parameter themselves

(2) GA's search from a population of points and not a single point

(3) GA's use payoff (objective function) information and not derivatives / other auxiliary knowledge

(4) GA's use probabilistic transition rules and not deterministic rules.

①→ Natural parameter set of the optimization Problem is "coded as a finite length string over some finite alphabet".

Assume problem is to max $f(x) = x^2$ over $[0, 31]$. Traditional methods x is directly manipulated to test for maximum.

GAs – Black Box View – Five switches (i/p) and output signal is $f(s)$; s is a particular switch settings. New problem "set switches to maximize f". Binary coding scheme – 1 for on, 0 for off. One setting → 11110

②→ Traditional methods move from a single point to a next point based on a transition rule. Could result in 'false peaks'. GAs search from a DB of points – climb many peaks in parallel. For the $f(x) = x^2$ problem; Traditional methods could move from one setting to another by some transition. With GAs: assume we start with 'random population of 4 strings' (using coin tosses for deciding i/o of a string)

Random population of size 4:

01101
11000
01000
10011

} will be referred as population X throughout.

Succesive populations generated using "Genetic Algorithm"

③ → Traditional search techniques require auxillary information such as derivative for gradient techniques. GA's are "BLIND" and require no other auxillary information. "They require only payoff values". More canonical natural than other schemes. However lack of auxillary knowledge can impede performance in competition GA's using nonpayoff information — Knowledge Directed GA's - explored later in Course.

④ → Probabilistic transition rules are used in GA to guide their search. Random choice is used as a tool to guide the search towards regions of search space.

Key aspects of GA:-
- Direct use of coding
- Search from population
- Blindness to Auxillary Info
- Randomized operators

+

## SIMPLE GENETIC ALGORITHM :

↳ Two Basic operations in GA "copying strings" "swapping partial strings"

↳ simplicity of operation and power of Effect - Key aspects of GA.

Generally simple GA's Comprised of 3 operators: {(1) Reproduction (2) crossover (3) Mutation}

① Reproduction → Individual strings are copied acurding to objective function values

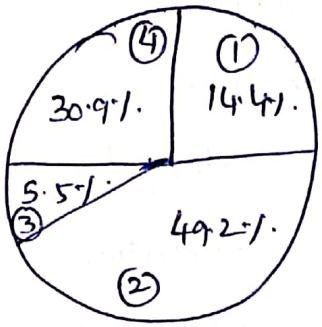
↳ function can be thought of as a profit measure, utility value → strings with higher value have a higher prob. of contributing its offspring (1/more) in the next generation.

↳ Artificial Version of Natural selection - Darwin Survival of fittest among Creatures. In GA objective function decides life/death of a string. (Solution)

Implementation of Reproduction - Roulette wheel (Biased slot sized in proportion to the fitness of the strings. For the sample population x considered earlier the roulette wheel is arrived @ as follows.

| No. | String | Dec | Fitness | % of total |
|---|---|---|---|---|
| 1 | 01101 | (13) | 169 | 14.4 |
| 2 | 11000 | | 576 | 49.2 |
| 3 | 01000 | | 64 | 5.5 |
| 4 | 10011 | | 361 | 30.9 |
| | | Sum | 1170 | |

Roulette wheel:-




To reproduce, Spin the wheel 4 times.

String 1 in each spin prob of 0.144

∴ Highly fit strings have higher number of offsprings in succeeding generation. Reproduced string (identical to selected one) is subject to the other genetic operators.

Next operation is crossover:

Pair of strings undergo crossing at an integer position k along string; k = 1 to $\ell-1$

$\ell \rightarrow$ length of string. Two new strings created by swapping all chars b/w positions (k+1) and ($\ell$) inclusive.

Assume reproduced strings: $A_1 = 0110|0$

$A_2 = 1100|0$

Random No b/w 1 to 4; assume k = 4 (crossover pt)

Resultant strings: $A_1' = 01100$

$A_2' = 11001$

crossover

(N)

# Genetic Algorithms

## Part 2: What is a Genetic Algorithm?

Fall 2009

*Instructor: Dr. Masoud Yaghini*



## Outline



(0)

# Definition



## Definition

- A population of individuals exists in an environment with limited resources
- **Competition** for those resources causes selection of those **fitter individuals** that are better adapted to the environment
- These individuals act as seeds for the generation of new individuals through recombination and mutation
- The new individuals have their fitness evaluated and compete for survival.
- Over time **natural selection** causes a rise in the fitness of the population

simple GA [schemata, Truel, BB claim/✓]
My Notes

## Definition

- **Genetic Algorithms** are
  - **Bio-Inspired artificial intelligence class,**
  - **stochastic,**
  - **population-based** algorithms

- Typically applied to:
  - hard problems with a large search space
  - discrete optimization
- Developed by **John Holland**, USA in the 1970's

## Genetic Algorithm Phases

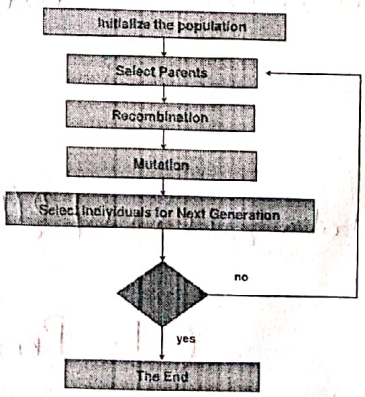


## Pseudo-code for typical GA

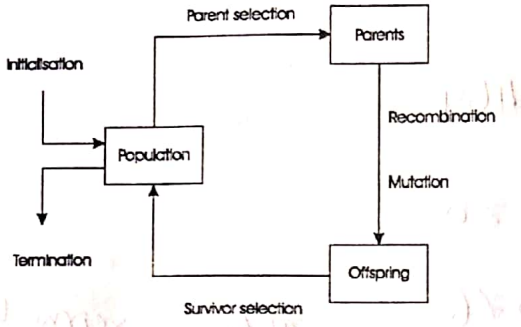
```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```



## GA Algorithmic Phases



## General Scheme of GA




## Simple Genetic Algorithm (SGA)



## Simple Genetic Algorithm (SGA)

- Holland's original GA is now known as the **simple genetic algorithm** (SGA)
- Other GAs use different:
  - Representations
  - Mutations
  - Crossovers
  - Selection mechanisms



## SGA summary

| | |
|---|---|
| Representation | Binary strings |
| Recombination | 1-point crossover |
| Mutation | bit-flipping with fixed probability |
| Parent selection | Fitness-Proportionate |
| Survivor selection | All children replace parents |
| Speciality | Emphasis on crossover |

## Simple example – f(x) = x²

- **Finding the maximum of a function:**
  - f(x) = x²
  - Range [0, 31] → Goal: find max ($31^2 = 961$)
  - Binary representation: string length 5 = 32 numbers (0-31)

| | | |
|---|---|---|
| genotype | 0 0 1 0 1 | |
| mapping | $2^4$ $2^3$ $2^2$ $2^1$ $2^0$<br>16 8 4 2 1 | |
| phenotype | $0*16+0*8+1*4+0*2+1*1 = 5$ | |
| fitness | 25 | = f(x) |



## x² example

- **x² example:**
  - Representation: Binary code
  - Population size: 4
  - Recombination: 1-point crossover
  - Mutation: Bit-flipping with fixed probability
  - Parent selection: Fitness-Proportionate
  - Initialization: Random
- We show one generational cycle done by hand



## x² example: selection

| String no. | Initial population | $x$ Value | Fitness $f(x) = x^2$ | $Prob_i$ | Expected count | Actual count |
|---|---|---|---|---|---|---|
| 1 | 0 1 1 0 1 | 13 | 169 | 0.14 | 0.58 | 1 |
| 2 | 1 1 0 0 0 | 24 | 576 | 0.49 | 1.97 | 2 |
| 3 | 0 1 0 0 0 | 8 | 64 | 0.06 | 0.22 | 0 |
| 4 | 1 0 0 1 1 | 19 | 361 | 0.31 | 1.23 | 1 |
| Sum | | | 1170 | 1.00 | 4.00 | 4 |
| Average | | | 293 | 0.25 | 1.00 | 1 |
| Max | | | 576 | 0.49 | 1.97 | 2 |



## X² example: crossover

| String no. | Mating pool | Crossover point | Offspring after xover | $x$ Value | Fitness $f(x) = x^2$ |
|---|---|---|---|---|---|
| 1 | 0 1 1 0 \| 1 | 4 | 0 1 1 0 0 | 12 | 144 |
| 2 | 1 1 0 0 \| 0 | 4 | 1 1 0 0 1 | 25 | 625 |
| 2 | 1 1 \| 0 0 0 | 2 | 1 1 0 1 1 | 27 | 729 |
| 4 | 1 0 \| 0 1 1 | 2 | 1 0 0 0 0 | 16 | 256 |
| Sum | | | | | 1754 |
| Average | | | | | 439 |
| Max | | | | | 729 |

## $X^2$ example: mutation

| String no. | Offspring after xover | Offspring after mutation | $x$ Value | Fitness $f(x) = x^2$ |
|---|---|---|---|---|
| 1 | 0 1 1 0 0 | 1 1 1 0 0 | 26 | 676 |
| 2 | 1 1 0 0 1 | 1 1 0 0 1 | 25 | 625 |
| 2 | 1 1 0 1 1 | 1 1 0 1 1 | 27 | 729 |
| 4 | 1 0 0 0 0 | 1 0 1 0 0 | 18 | 324 |
| Sum | | | | 2354 |
| Average | | | | 588.5 |
| Max | | | | 729 |



## The simple GA

- SGA Shows many **shortcomings**:
  - Representation is too restrictive
  - Mutation & crossovers only applicable for bit-string & integer representations
  - Selection mechanism sensitive for converging populations with close fitness values
  - Generational population model (step 5 in SGA) can be improved with explicit survivor selection

## References



## References


- Eiben and Smith. **Introduction to Evolutionary Computing**, Springer-Verlag, New York, 2003.
- J. Dreo A. Petrowski, P. Siarry E. Taillard, **Metaheuristics for Hard Optimization**, Springer-Verlag, 2006.

Defining length (H) - diff between 1st & last fixed points.

Order (H) = No of fixed points.

$$m(H_1, t+1) = m(H_1, t) \cdot \frac{f(H)}{\bar{f}}$$

Ai string gets selected with $P_i = f_i / \Sigma f_i$

$$= f_i / \Sigma f_i$$

$$m(H_1, t) \cdot \frac{f(H)}{\Sigma f_i} \cdot n \longrightarrow \text{Non overlapping Popn of Size.}$$

[Sel'n with replacement].

$$= m(H_1, t) \cdot \frac{f(H)}{\Sigma f_i / n} = m(H_1, t) \cdot \frac{f(H)}{\bar{f}}$$

$$m(H_1, t+1) = m(H_1, t) \frac{(\bar{f} + c\bar{f})}{\bar{f}} = \boxed{m(H_1, t)(1+c)}$$

$f(H)$ above average by $c$

$$m(H_1, 1) = m(H_1, 0)(1+c)$$

$$m(H_1, 2) = m(H_1, 1)(1+c) = (1+c)^2 m(H_1, 0)$$

$$\boxed{m(H_1, t) = (1+c)^t m(H_1, 0)}$$ → ✓ Compound Interest eqn

"exp incr. trials to above avg H ✓ Geometric progression

" decr " " below " "

$A =$ 0 1 1 | 1 | 0 0 0

$H_1 =$ * 1 * * | * * 0 — $H_1$ defined by 1

$H_2 =$ * * * 1 | 0 * * — $H_2$ survives crossover

$H_1 = \delta = 5$ $\quad$ $H_2 = \delta = 1$

$O(H_1) = 2$ $\quad$ $O(H_2) = 2$.

Short defining length survives X over

Possible X over $\quad$ No's $= l-1$ $\quad$ if length$(H) = l$.

$$P_d = \frac{\delta(H)}{l-1}$$

$$\therefore P_s = 1 - \frac{\delta(H)}{l-1} \qquad \therefore P_s \geq 1 - \frac{P_c \cdot \delta(H)}{l-1}.$$

$P_c \geq 0.5$; $P_c = 1 \Rightarrow \delta(H)/l-1$.

$$\therefore m(H, t+1) = m(H,t) \frac{f(H)}{\bar{f}} \left\{ 1 - \frac{P_c \delta(H)}{l-1} \right\}.$$

$P_{\text{Mutation survival}} \quad P_r = 1 - P_m$.

$(1-P_m) \times \cdots \cdots$ $O(H)$ times

$= (1-P_m)^{O(H)}$

$= 1 - P_m O(H)$

$$m(H, t+1) = m(H,t) \frac{f(H)}{\bar{f}} \left\{ 1 - \frac{P_c \delta(H)}{l-1} - P_m O(H) \right\}$$

"Short, low order, above average schemata receive exp. inv trials in subsq Sens

Schemata Illustration :-

Earlier Trial :-

| | | SR'N | $\bar{H}$ |
|---|---|---|---|
| $H_1$ | 1 * * * * | 2,4 | 469 |
| $H_2$ | * 1 0 * * | 2,3 | 320 |
| $H_3$ | 1 * * * 0 | 2 | 576 |

after regression

Soln :-

$m(H_1) = 2 \cdot \frac{469}{293} = 3.2 \approx 3$ (failures)

$m(H_2) = 2 \cdot \frac{320}{293} = 2.18 \approx 2$

$m(H_3) = 1 \cdot \frac{576}{293} = 1.97 \approx 2$

after x'over :-

$m(H_2) = 2.18 \times \frac{1}{1-\frac{1}{4}} = \frac{2.18}{4} = 1.64$ ✓ (2) ✓

after x'over and

$2.18 \left(1 - \frac{1}{4}\right)$

$m(H_3) = 1.97 \times \frac{4}{4} = 1.97 * (1-1)$

$= 0$ // No surviving offspring

no mutation.

actual Ans :

16 8 4 2 1

Generation 0 [Initial population].

$f(x) = x^2$

| Idx | x | Encoding |
|---|---|---|
| 1 | 13 | 0 1 1 0 1 |
| 2 | 24 | 1 1 0 0 0 |
| 3 | 8 | 0 1 0 0 0 |
| 4 | 19 | 1 0 0 1 1 |

[Repr Mod].

# Bucket Brigade classifier

* (1) Rule + Message System; Apportionment of Credit System;
* Genetic Algorithm

(1) → Prodn system → if <Condn> then <action>

Activation of rules < Serial Manner [Expert Systems domain]
parallel [NL - classifiers]

* This choice of best rule to be activated is done using GA for Competition based firing of rule

<Message> = $\{0,1\}^*$ ; <classifier> = <Condn> : <message>

<Condn> = $\{0,1,\#\}^*$

Initial Msg:- 0111 (E)

| Index | Classifier | Strength | Msg (t=0) | Match | Bid | S | Msg (t=1) | Match | Bid | S | Msg (t=2) | Match | Bid |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 01##: 0000 | 200 | | E | 20 | 180 | 0000 | | | 220 | | | |
| 2 | 00#0: 1100 | 200 | | | | 200 | | 1 | 20 | 180 | 1100 | | |
| 3 | 11##: 1000 | 200 | | | | 200 | | | | 200 | | 2 | 20 |
| 4 | ##00: 0001 | 200 | | | | 200 | | 1 | 20 | 180 | 0001 | 2 | 18 |
| | Env:- | 0 | 0111 (t=0) | | | (20) | (t=1) | | | (20) | (t=2) | | |
| 1 | 01##: 0000 | 220 | | | | 220 | | | | | | | |
| 2 | 00#0: 1100 | 218 | | | | 218 | | | | | | | |
| 3 | 11##: 1000 | 180 | 1000 | | | 196 | | | | | | | |
| 4 | ##00: 0001 | 162 | 0001 | 3 | 16 | 146 | 0001 | | | | | | |
| | | (20) | | | | (20) | | | | | | | |