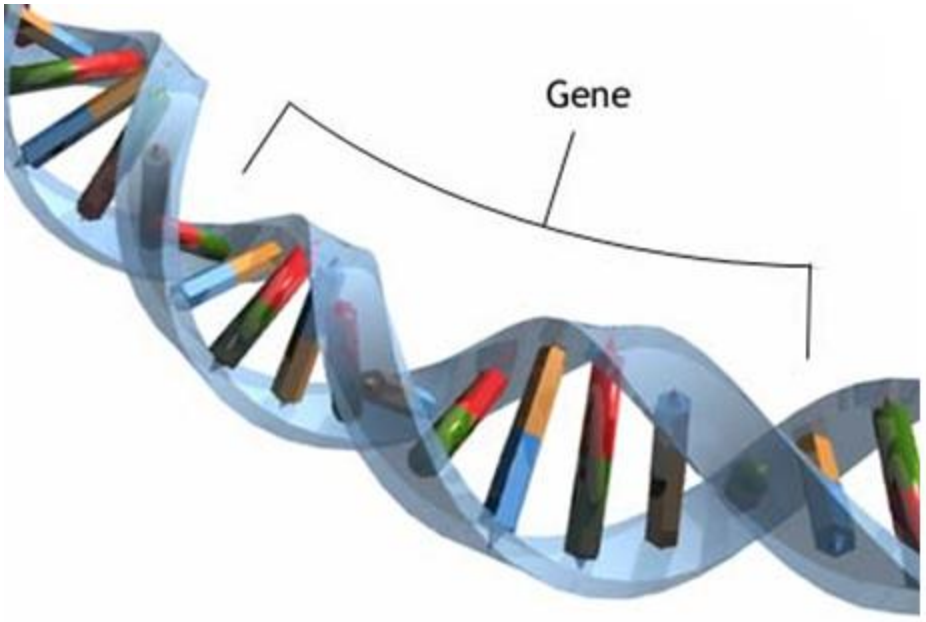
SA1098/2098 Energy System Optimization

Chapter 5. Basics in Genetic Algorithm



HYUNSOO LEE

References

- Two Viewpoints
  - Tom M. Mitchell,"Machine Learning", McGraw-Hill,1997
    - Chapter 9
    - From "Dr. Choe's teaching material"
  - Mitsuo Gen and Rinrin, "Network model to Tamokuteki Genetic Algorithm", Kyoritsu Shuppan, 2008

Contents

- Introduction
- Genetic Algorithm
- GABIL
- Schema and Schema Theorem
- Genetic Programming
- Baldwin effect

Introduction

- Genetic Algorithm
  - Based on “simulated evolution” loosely
  - Hypotheses
    - Described in bit strings
  - Search
    - Population of hypothesis, refined through mutation and crossover
  - Application
    - Optimization, learning topology
    - Parameter turning in neural network, etc

Biological Evolution

- Lamarck and others
  - Species “transmute” over time / Inheritance of acquired trait
  - Believed individual genetic makeup was altered by lifetime evidence’
  - But, current evidence contradicts this view
- Darwin and Wallace
  - Consistent, heritable variation among individuals in population
  - Natural selection of the fittest
- Mendel and genetics
  - A mechanism for inheriting traits
  - Genotype → phenotype

Motivation

- Motivation of G.A.
  - Mutation and Crossover of hypotheses in the current population
  - "Generate – and - test" approach
  - Motivating factors
    - Evolution is known to be successful
    - Gas can search hypotheses containing complex interacting parts
    - Easily parallelizable

Population and Fitness

- Population
  - Set of current hypotheses
- Fitness
  - Predefined measure of success
- Elements of GA
  - Fitness test → selection → reproduction (Mutation, crossover)

Genetic Algorithm Procedure (1)

- Terms in Mitchell's book
  - Fitness
    - A function that assigns an evaluation score, given a hypothesis
  - Fitness_threshold
    - A threshold specifying the termination criterion
  - p
    - The number of hypothesis to be included in the population
  - r
    - The fraction of the population to be replaced by Crossover
  - m
    - The mutation rate

Genetic Algorithm Procedure (2)

GA($Fitness, Fitness\_threshold, p, r, m$)

- *Initialize:* $P \leftarrow p$ random hypotheses
- *Evaluate:* for each h in P, compute $Fitness(h)$
- While $[\max_h Fitness(h)] < Fitness\_threshold$
  1. *Select:* Probabilistically select $(1-r)p$ members of P to add to P_s.

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

  2. *Crossover:* Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from P. For each pair, $\langle h_1, h_2 \rangle$, produce two offspring by applying the Crossover operator. Add all offspring to P_s.
  3. *Mutate:* Invert a randomly selected bit in $m \cdot p$ random members of P_s
  4. *Update:* $P \leftarrow P_s$
  5. *Evaluate:* for each h in P, compute $Fitness(h)$
- Return the hypothesis from P that has the highest fitness.

Genotype Conversion (1)

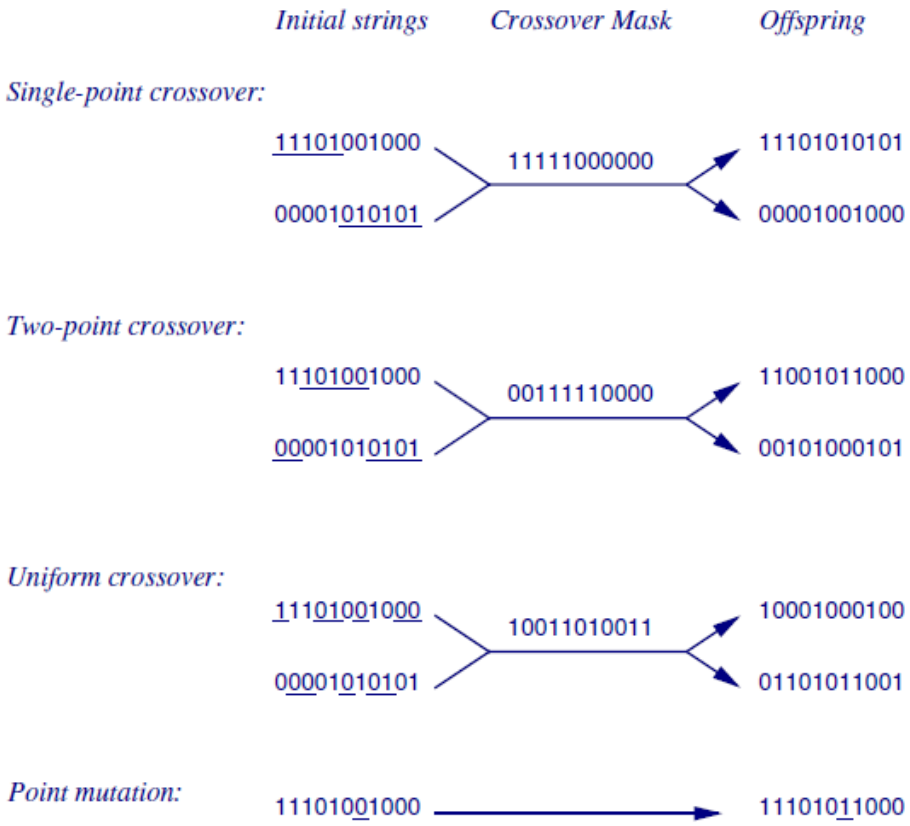
- Outlook
  - Sunny → 001
  - Overcast → 010
  - Rain → 010
  - Overcast Λ Rain →011
  - 111 ← ?

Genotype Conversion (2)

- Similarly
  - Outlook Wind → 011 10
  - If Wind=Strong Then PlayTennis = yes
    ➔111 10 10

Genetic Operator

- Crossover and Mutation



Selecting Most Fit Hypothesis

- Fitness Proportionate Selection

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_i)}$$

  - How to select "h" with the proportions
    - Tournament Selection
      - Uniformly select h1 & h2
      - With Pr, select the more fit
    - Rand Selection
      - Sort all hypothesis by Pr
      - Select with the rank

GABIL (1)

- DeJong et al. 1993

Fitness:

$$Fitness(h) = (percent\_correct(h))^2$$

Representation:

IF $a_1 = T \wedge a_2 = F$ THEN $c = T$; IF $a_2 = T$ THEN $c = F$

represented by

a_1	a_2	c	a_1	a_2	c
10	01	1	11	10	0

Genetic operators: ???

- want variable length rule sets (as number of attributes can change)
- want only well-formed bitstring hypotheses

GABIL (2)

Start with

	a_1	a_2	c	a_1	a_2	c
h_1 :	10	01	1	11	10	0
h_2 :	01	11	0	10	01	0

1. choose crossover points for h_1, e.g., after bits 1, 8
2. now restrict points in h_2 to those that produce bitstrings with well-defined semantics, e.g., $\langle 1, 3\rangle$, $\langle 1, 8\rangle$, $\langle 6, 8\rangle$.

if we choose $\langle 1, 3\rangle$, result is

	a_1	a_2	c
h_3 :	11	10	0

	a_1	a_2	c	a_1	a_2	c	a_1	a_2	c
h_4 :	00	01	1	11	11	0	10	01	0

GABIL (3)

- <1,8> cases

GABIL (4)

- <6,8> cases

GABIL (5)

- Lesson
  - Picking a representation for the hypotheses can be tricky
  - Genetic operators need to be preserve the semantics of the genetic encoding

GABIL (6)

- Introduction to another operators
  - AA : AddAlternative
    - Generalize constraint on a_i by changing a 0 to 1
  - DC : DropCondition
    - Generalize constraint on a_i by changing every 0 to 1

a_1	a_2	c	a_1	a_2	c	AA	DC
01	11	0	10	01	0	1	0

GABIL (7)

- Results
  - GABIL without AA and DC operators : 92.1 % accuracy
  - GABIL with AA and DC operators : 95.2% accuracy
- Needs more accurate operators

Characterizing Evolution (1)

- Schema in Genotype
  - 0,1,* (“don’t care”)
    - Typical schema : 10**0*
    - Instance : 101101, 100000, …
  - Another schema
    - An instance of length 4, say “0010” can have 2^4 matching schema
  - m(s,t) = number of instances of schema s in pop at time t
  - Want to estimate m(s,t+1) given m(s,t) and other factors

Characterizing Evolution (2)

- Factors influencing change in m(s,t)
  - M(s,t) can change as t changes, due to the following factors:
    - Selection
    - Crossover
    - Mutation
- Schema theorem
  - Gives E[m(s,t+1)]

Influence Selection (1)

- Definition
  - Average fitness of pop. at time t : $\bar{f}(t)$
  - Instances of schema s in pop at time t: $m(s,t)$
  - Average fitness of instances of s at time t: $\hat{u}(s,t)$
  - Instances of schema s in the pop at time t: $h \in s \bigcap p_t$

$$\Pr(h) = \frac{f(h)}{\sum_{i=1}^{n} f(h_i)} = \frac{f(h)}{n \cdot \bar{f}(t)}$$

$$\hat{u}(s,t) = \frac{\sum_{h \in s \bigcap p_t} f(h)}{m(s,t)}$$

Influence Selection (2)

$$\Pr(h) = \frac{f(h)}{\sum_{i=1}^{n} f(h_i)} = \frac{f(h)}{n \cdot \bar{f}(t)}$$

$$\hat{u}(s,t) = \frac{\sum_{h \in s \cap p_t} f(h)}{m(s,t)}$$

$$\Pr(h \in s) = \sum_{h \in s \cap p_t} \frac{f(h)}{n \cdot \bar{f}(t)} = \frac{\hat{u}(s,t)}{n \cdot \bar{f}(t)} m(s,t)$$

$$E[m(s,t+1)] = \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t)$$

Schema Theorem

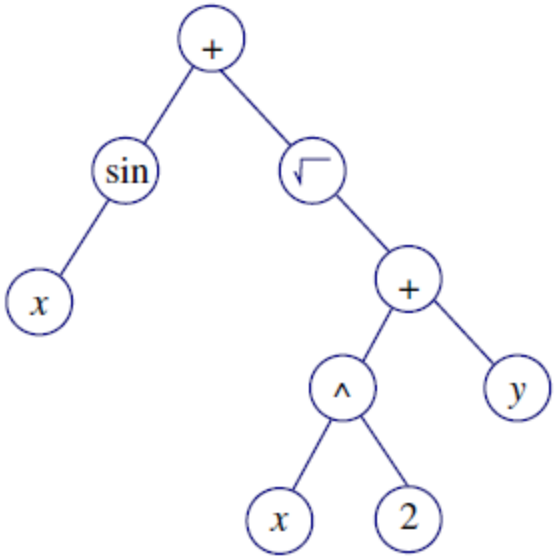
$$E[m(s,t+1)] \geq \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t) \left(1 - p_c \frac{d(s)}{l-1}\right) (1-p_m)^{o(s)}$$

- $m(s,t) =$ instances of schema s in pop at time t
- $\bar{f}(t) =$ average fitness of pop. at time t
- $\hat{u}(s,t) =$ ave. fitness of instances of s at time t
- $p_c =$ probability of single point crossover operator
- $p_m =$ probability of mutation operator
- $l =$ length of single bit strings
- $o(s)$ number of defined (non "*") bits in s
- $d(s) =$ distance between leftmost, rightmost defined bits in s

Genetic Programming (1)

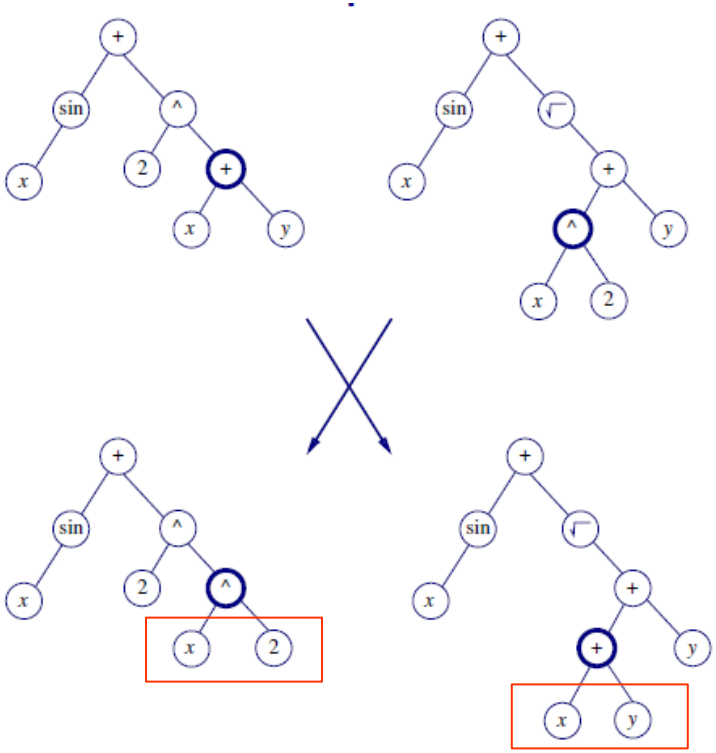
- Population of programs
  - represented by trees

$$\sin(x) + \sqrt{x^2 + y}$$



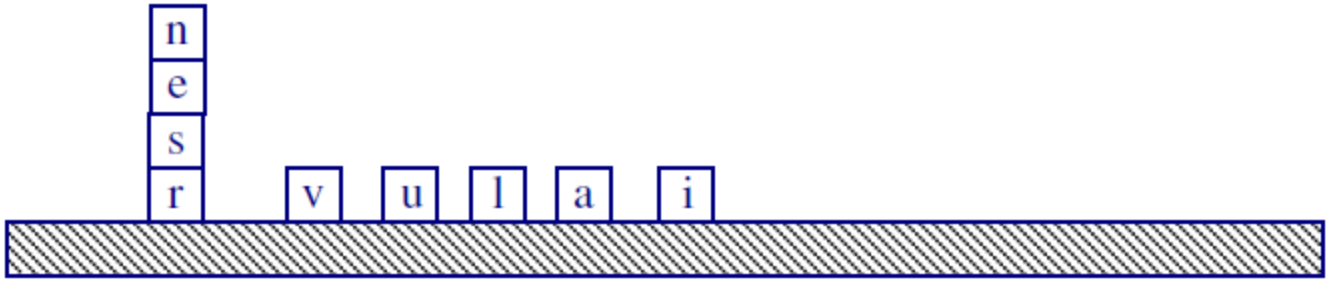
Genetic Programming (2)

- Crossover
  - Swap whole subtrees



Block Problem (1)

- "Universal"



  - CS (Current Stack) : name of the top block on stack or "F"
  - TB (Tob corrected Block) : name of topmost correct block on stack
  - NN (Next Necessary) : name of the next block needed above TB in the stack

Block Problem (2)

- Several Primitive functions

(MS x): ("move to stack"), if block x is on the table, moves x to the top of the stack and returns the value T. Otherwise, does nothing and returns the value F.

(MT x): ("move to table"), if block x is somewhere in the stack, moves the block at the top of the stack to the table and returns the value T. Otherwise, returns F.

(EQ x y): ("equal"), returns T if x equals y, and returns F otherwise.

(NOT x): returns T if $x = F$, else returns F

(DU x y): ("do until") executes the expression x repeatedly until expression y returns the value T

Block Problem (3)

- After 166 tests using GA
  - Using population of 300 Programs, found this after 10 generations:

"(EQ(DU(MT CS)(NOT CS))(DU(MS NN)(NOT NN)))"

Baldwin effect (1)

- Assume
  - Individual learning has no direct influence on individual DNA
  - But, ability to learn reduces need to “hard wire” traits in DNA
- Then
  - Ability of individuals to learn will support more diverse gene pool
    - Because learning allows individuals with various “hard wired” traits to be succcessful
  - More diverse gene pool will support faster evolution of gene pool

➔ **Individual learning increases rate of evolution, indirectly**

Baldwin effect (2)

- Examples
  - 1. New predator appears in environments
  - 2. Individuals who can learn (to avoid it) will be selected
  - 3. Increase in learning individuals →more diverse gene pool
  - 4. Resulting in faster evolution
  - 5. Possibly resulting in new non-learned traits such as instinctive fear of predator

Miscellaneous (1)

- Consideration
  - Coevolution
    - Escalating effect or complementary dependence (insects and flowering plants) between two or more species
  - Cultural Transmission
    - Memes Vs. Genes

Miscellaneous (2)

- Evolutionary Learning
  - Conduct randomized, Parallel, Hill-climbing search through H
  - Approach learning as optimization problem (Optimize fitness)
  - Nice feature : evaluation of Fitness can be very indirect
    - Consider learning rule set for multistep decision making
    - No issue of assigning credit / blame to individual Steps