POLITECNICO DI MILANO

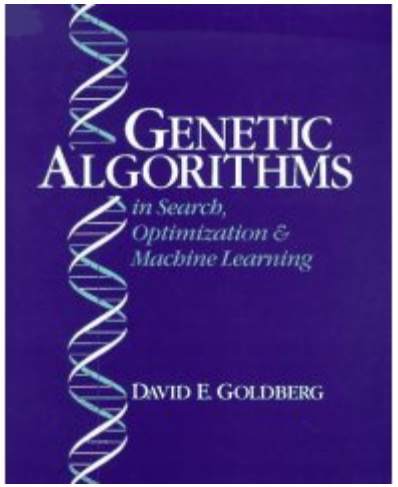
Lecture 4: Schemata & Schema Theorem

Genetic Algorithms and Other Evolutionary Techniques

Prof. Pier Luca Lanzi

References

- David E. Goldberg "Genetic Algorithms in Search, Optimization and Machine Learning" Addison Wesley (1 Jan 1989)



- David E. Goldberg "The Design of Innovation" Springer-Verlag 1992

Outline

- What does a genetic algorithm process?
- Schemata, the similarity templates
- Schema Theorem
- Building Block Hypothesis
- Implicit Parallelism

Let's Play The GA Game

x	$fitness$
10110111	1.00
00110100	4.00
11011011	1.00
00101000	5.00
11011111	1.00

Can you suggest a better individual?

Let's Play The GA Game (2nd Match)

x	$fitness$
10110111	6.00
00110100	4.00
11011011	6.00
00101000	2.00
11011111	7.00

Can you suggest a better individual?

What approach?

Define similarity among individuals

Study evolution of similarities
between candidate solutions

Study evolution of groups of
similar candidate solutions

How Can We Describe Similarity?

- To describe similarity among individuals in the population we use the concept of schema (schemata)

- What is a schema?
  - Similarity template describing a subset of strings with similarities at certain positions
  - String over the alphabet {0,1,*}
  - * is the don't care symbol

- The don't care symbol * is a metasymbol:
  it is never explicitly process by the genetic algorithm

Examples of Schema

- Schema ***11 matches,
  - 00011
  - 00111
  - 01011
  - …
  - 11111
- Schema 0***1 matches,
  - 00001
  - 00011
  - 00101
  - …
  - 01111

Schema and Schemata

- Order of schema H, o(H)
  the number of specific (non-*) positions in the schema

- Examples
  - o(**1*0) = 2
  - o(1**11) = 3

- Defining length of schema H, δ(H)
  distance between first and last non-* in a schema

- Examples
  - δ(**1*0) = 2
  - δ(1**11) = 4

Schema and Schemata

- Schema H1 is a competitor of schema H2 if,
  - H1 and H2 are specified in the same positions.
  - H1 and H2 differ in at least one specified position.

- Example
  - *111* is a competitor of *101*

Schema and Schemata

- A string of length l is an instance of 2^l-1 schemata
- But how many schemata are there in the whole search space? (how many choices each locus?)
- Since one string instances 2^l-1 schemata, how much does a population tell us about schemata of various orders?
- Implicit parallelism
  - One string's fitness tells us something about the relative fitness of more than one schema

Structural vs. Substructural Thinking

- We evaluate solutions at the level of full strings, solutions, or structures
- We modify a structure at the level of modules, substructures, components, etc.
- Trick of genetic optimization: Only get evaluation at the highest level.
- How can we get best substructures without explicit evaluation of them?

How a Simple Genetic Algorithm Works?

- A simple genetic algorithm
  - Population size N
  - Fixed-length binary strings of length l
  - Fitness-proportionate selection
  - One-point crossover with probability pc
  - Bit-flip mutation
  - Fitness is positive

- Notation
  - A(t) population at time t
  - A_i i-th individual in A(t)
  - f(H,t) fitness of schema H at time t
  - m(H,t) number of examples of H in A(t)
  - f_A average fitness in A(t)

- How selection, crossover, and mutation work?

How a Simple Genetic Algorithm Works? Proportionate Selection

- During reproduction, A_i is selected with probability p_i,

$$p_i = \frac{f_i}{\sum f_j}$$

- After reproduction, we expect,

$$m(H, t+1) = m(H, t)\frac{f(H, t)}{f_A}$$

What is the Effect of Selection Alone?

- The equation,

$$m(H, t+1) = m(H, t)\frac{f(H, t)}{f_A}$$

- can be rewritten as,

$$\begin{aligned} m(H, t+1) &= m(H, t)\frac{f_A + cf_A}{f_A} \\ &= m(H, t)(1+c) \\ &= m(H, 0)(1+c)^t \end{aligned}$$

Reproduction allocates exponentially increasing/decreasing number of trials of above/below average schemata

How a Simple Genetic Algorithm Works? Crossover

- Schema H survives crossover with probability p_s,

$$p_s = 1 - \delta(H)/(l-1)$$

- When crossover probability is taken into account,

$$p_s \geq 1 - p_c\delta(H)/(l-1)$$

- So that,

$$m(H, t+1) \geq m(H, t)\frac{f(H,t)}{f_A}\left[1 - p_c\frac{\delta(H)}{l-1}\right]$$

How a Simple Genetic Algorithm Works? Mutation

- Each allele survives with probability

$$(1 - p_m)$$

- The schema H survives when all the o(H) alleles survive

$$(1 - p_m)^{o(H)}$$

- When pm is very small,

$$(1 - p_m)^{o(H)} \approx 1 - o(H)p_m$$

- So that,

$$m(H, t+1) \geq m(H, t) \frac{f(H, t)}{f_A} \left[1 - p_c \frac{\delta(H)}{l-1} - p_m o(H) \right]$$

Selection, Crossover, and Mutation

- Define the probability of survival Ps(H,t),

$$P_s(H,t) = \left[1 - p_c \frac{\delta(H)}{l-1} - p_m o(H)\right]$$

- H gets more copies when $P_s(H,t)\frac{f(H,t)}{f_A} > 1$
- H gets less copies when $P_s(H,t)\frac{f(H,t)}{f_A} < 1$
- Expected number of representatives of H grows with $\frac{f(H,t)}{f_A}$
- P_s(H,t) decreases with increasing o(H) and δ(H)

The Schema Theorem

- Two factors influence the number of representatives of H
  - Over the average schema fitness, leads to higher m(H,t+1)
  - Greater o(H) and δ(H), lead to a smaller m(H,t)

- Short, low order, above-average schemata receive exponentially increasing trials in subsequent generations.

$$m(H,t+1) \geq m(H,t)\frac{f(H,t)}{f_A}\left[1 - p_c\frac{\delta(H)}{l-1} - p_m o(H)\right]$$

The Schema Theorem

- Generalization of the Schema Theorem

$$m(H, t+1) \geq m(H, t)\phi(H, t)\left[1 - \varepsilon(H, t)\right]$$

- Reproductive ratio $\phi(H, t)$
- Growth ratio $\gamma = \phi(H, t)\left[1 - \varepsilon(H, t)\right]$
- We want $\gamma \geq 1$

Building Blocks

Building blocks

Low order, low defining-length schemata with above average fitness.

Building Block Hypothesis

"Short, low-order, and highly fit schemata are sampled, recombined, and resampled to form strings of potentially higher fitness […] we construct better and better strings from the best partial solutions of the past samplings."

David Goldberg, 1989

How Many Schemata are Processed?

- A population with length l and size n, processes between 2^l to $n2^l$ schemata

- Consider only schemata that survive with a certain probability
- Count how many schemata are effectively processed

- Effective processing = selection + mixing (recombination).

Implicit Parallelism

A genetic algorithm with a population size n can usefully process on the order of $O(n^3)$ schemata in a generation despite the processing of only n solutions

Example I

No.	Old	$f(x)$	New	$f(x)$
1	11000	2	10010	2
2	10111	4	11011	4
3	01011	3	10111	4
4	00100	1	11001	3
avg		2.5		3.25

Example 1
Expected number of copies of 1****

- m(1****,0) = 2
- f(1****,0) = (2+4)/2 = 3 (3>2.5)

- Survival after mutation $1 - p_m o(H) = 1 - 0.2 \times 1 = 0.8$

- Survival after crossover and mutation

$$1 - p_m o(H) - p_c \frac{\delta(H)}{5-1} = 1 - 0.2 \times 1 - 1 \times 0/4 = 0.8$$

- After selection m(1****,1) = 2x3/2.5 = 2.4

- After selection, crossover, and mutation
  m(1****,1) ¸ 2.4 x 0.8 > 1.92

Example 2

No.	Old	$f(x)$	New	$f(x)$
1	11000	2	10010	2
2	10111	4	11011	4
3	01011	3	10111	4
4	00100	1	11001	3
avg		2.5		3.25

- m(***1*, 0) = 2
- f (***1*, 0) = (3 + 4)/2 = 3.5 (3.5 > 2.5)
- m(***1*, 1) = 3 (+1)
- f (***1*, 1) = (2 + 4 + 4)/3 = 3.33 (-0.17)

Example 3

No.	Old	$f(x)$	New	$f(x)$
1	11000	2	10010	2
2	10111	4	11011	4
3	01011	3	10111	4
4	00100	1	11001	3
avg		2.5		3.25

- m(****0, 0) = 2
- f (****0, 0) = (2 + 1)/2 = 1.5 (1.5 < 2.5)
- m(****0, 1) = 1 (-1)
- f (****0, 1) = 2 (+0.5)

Summary

- Schemata are similarity templates
- High performance, short, low-order schemata receive at least exponentially increasing number of trials in successive generations
- By processing similarities, a genetic algorithm reduces the complexity of an arbitrary problems
- Highly fit, short, low order schemata become the partial solutions to a problem (these are called building-blocks)