



Cohort Genetic Algorithms





Based on "Building Blocks, Cohort Genetic Algorithms, and Hyperplane-Defined Functions" by John H. Holland

presented by Jeff Wallace
CS 790R



Review: GA Crossover

- Parent genomes recombine to form 2 new genomes
- Crossover point(s) determined at random

Parents	Children
	
	



Review: GA Mutation

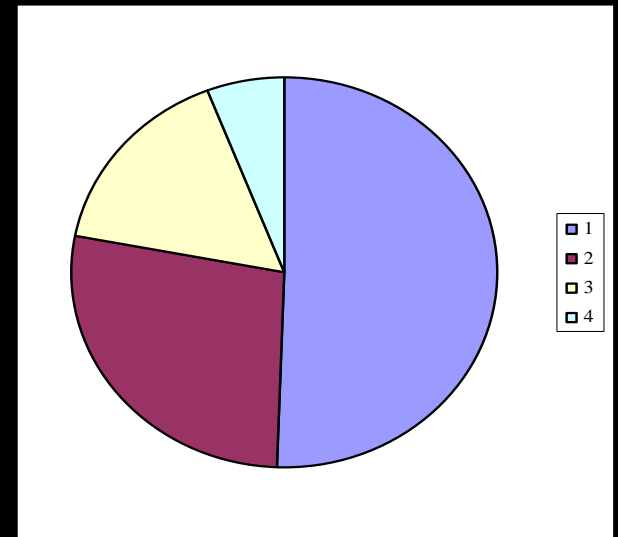
- Random gene mutates with (low) probability



Review: Selection

Roulette-wheel selection determines parents of next generation

	Score	Norm
p1	60	.504
P2	33	.277
P3	19	.160
p4	7	.059





Review: Basic Algorithm

choose initial population

repeat

- evaluate each individual's fitness

- select best-ranking individuals to reproduce

- mate pairs at random

- apply crossover operator

- apply mutation operator

until terminating condition



Schema

- aka similarity templates
- describes how strings are similar at specific positions
- uses wildcards ('*') to describe parts of the template that are not relevant
- e.g. *111* describes {01110, 01111, 11110, 11111}
- "order" is the number of fixed positions
 - high order is more specific/defining than low order
- "defining length" is the distance between the first/last fixed positions
- schemata aka hyperplanes
- USEFUL schemata are "building blocks"



Schema Theorem

- short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations.
- corollary: below-average schemata receive exponentially fewer trials in subsequent generations.



Genetic Algorithms

- number of schemata manipulated is much larger than the number of strings explicitly processed ($\sim n^3$, n =pop size). aka “implicit parallelism”
- not good for finding best individual
- not best approach for highly correlated landscapes
- good for finding improvements in uncorrelated landscapes

Problem with Basic GA:

Hitchhiking

- bits that are near useful building blocks tend to persist by virtue of proximity — not utility.
- hitchhiking loci tend to be under-explored.
- solution: scale reproduction rate for best string downwards (toward 1.0)



Problem with Hitchhiking

Solution: Fractional Offspring

e.g. best string = reproduction rate of 1.2

- second offspring created with $p = 0.2$
- after 4 generations, expect to find 2 copies of individual ($1.2^4 = 2$)
- variance is large, however. The probability of only one copy is 40%, meaning that useful schemata are lost.



Solution to the Problem of the Solution: Cohort GAs?

- designed to allow low scaling of reproduction rates without high variance caused by stochastic approach to fractional offspring
- idea: fitness determines how long a string has to wait before reproducing
 - higher-fit strings reproduce more quickly, thus having a higher number of progeny over time.



Cohort GA Implementation

- divide population into ordered set of non-overlapping subpopulations (cohorts)
- Reproduction function cycles through cohorts (in order)

cGA Reproduction (within cohort)



- each string produces 2 offspring
- each string is evaluated and scaled to low reproductive rate (e.g. 1.2). Average reproductive rate = 1.0
- calculate doubling time* (DT) at this reproductive rate (e.g. =4 @ 1.2).
 - * at low rates, a linear function may be used
- place this offspring in the cohort DT steps "ahead" of current cohort.



cGA Tweaks

- cohorts will contain different pop sizes.
- mating produces 4 offspring (2 each parent)
- if population bounded, strings must be deleted. You want to delete low-fitness strings without emptying distant cohorts
- preserve diversity by scaling fitness according to commonality. Common alleles reduce fitness. Unique alleles increase it.



Issues

- biological analogue?
- why doesn't downscaling fitness of common alleles punish strong building blocks?
- reduces premature convergence, but this isn't always desirable.