Introduction to Genetic Algorithms

A Tutorial by Erik D. Goodman

Professor, Electrical and Computer Engineering
Professor, Mechanical Engineering
Co-Director, Genetic Algorithms Research and Applications Group (GARAGe)
Michigan State University
goodman@egr.msu.edu
Executive Committee Member, ACM SIGEVO
Vice President, Technology
Red Cedar Technology, Inc.

2009 World Summit on Genetic and Evolutionary Computation
Shanghai, China

Thanks to:

Much of this material is based on:

- **David Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989 (still one of the best introductions!)**
- **Darrell Whitley, "Genetic Algorithm Tutorial" – on the web at** **www.cs.colostate.edu/~genitor/MiscPubs/tutorial.pdf**

Overview of Tutorial

- Quick intro – What IS a genetic algorithm?
  - Classical, binary chromosome
- Where used, & when better to use something else
- A little theory – why a GA works
- GA in Practice -- some modern variants

Genetic Algorithms:

- **Are a method of search, often applied to optimization or learning**
- **Are stochastic – but are *not* random search**
- **Use an evolutionary analogy, "survival of fittest"**
- **Not *fast* in some sense; but sometimes more robust; scale relatively well, so can be useful**
- **Have extensions including Genetic Programming (GP) (LISP-like function trees), learning classifier systems (evolving rules), linear GP (evolving "ordinary" programs), many others**

The Canonical or Classical GA

- **Maintains a set or "population" of <u>strings</u> at each stage**
- **Each string is called a chromosome, and encodes a "candidate solution"– CLASSICALLY, encodes as a *binary string* (but now in almost any conceivable representation)**

Criterion for Search

- Goodness ("fitness") or optimality of a string's solution determines its FUTURE influence on search process -- survival of the fittest
- Solutions which are good are used to generate other, similar solutions which may also be good (even better)
- The POPULATION at any time stores ALL we have learned about the solution, at any point
- Robustness (efficiency in finding good solutions in difficult searches) is key to GA success

Classical GA: The Representation

`1011101010` – a possible 10-bit string ("CHROMOSOME") representing a possible solution to a problem

Bits or subsets of bits might represent choice of some feature, for example. Let's represent choice of shipping container for some object:

<u>bit position</u>	<u>meaning</u>
1-2	steel, aluminum, wood or cardboard
3-5	thickness (1mm-8mm)
6-7	fastening (tape, glue, rope, plastic wrap)
8	stuffing (paper or plastic "peanuts")
9	corner reinforcement (yes, no)
10	handles (yes, no)

Terminology

Each position (or each set of positions that encodes some feature) is called a LOCUS (plural LOCI)

Each possible value at a locus is called an ALLELE

We need a simulator, or evaluator program, that can tell us the (probable) outcome of shipping a given object in any particular type of container

- may be a COST (including losses from damage) (for example, maybe 1.4 means very low cost, 8.3 is very bad on a scale of 0-10.0), or
- may be a FITNESS, or a number that is larger if the result is BETTER

How Does a GA Operate?

- **For ANY chromosome, must be able to determine a FITNESS (measure of performance toward an objective) using a simulator or analysis tool, etc.**
- **Objective may be maximized or minimized; usually say *fitness* is to be maximized, and if objective is to be minimized, define fitness from it as something to maximize**

GA Operators: Classical Mutation

- Operates on ONE “parent” chromosome
- Produces an “offspring” with changes.
- Classically, toggles one bit in a binary representation
- So, for example: `1101000110` could mutate to: `1111000110`
- Each bit has same probability of mutating

Classical Crossover

- Operates on two parent chromosomes
- Produces one or two children or offspring
- Classical crossover occurs at 1 or 2 points:
- For example: (1-point) (2-point)

```
           111|1111111  or   111|11111|11
   X       000|0000000       000|00000|00
   -------------------       ------------
           111|0000000       111|00000|11
and        000|1111111       000|11111|00
```

Selection

- ***Traditionally*, parents are chosen to mate with probability proportional to their fitness: *proportional selection***
- **Traditionally, children replace their parents**
- **Many other variations now more commonly used (we'll come back to this)**
- **Overall principle: survival of the fittest**

Synergy – the KEY

Clearly, selection alone is no good …

Clearly, mutation alone is no good …

Clearly, crossover alone is no good …

Fortunately, using all three simultaneously is sometimes spectacular!

Contrast with Other Search Methods

- “indirect” -- setting derivatives to 0
- “direct” -- hill climber
- enumerative – search ‘em all
- random – just keep trying, or can avoid resampling
- simulated annealing – single-point method, reals, changes all loci randomly by decreasing amounts, mostly keeps the better answer, …
- Tabu (another common method)

BEWARE of Claims about ANY Algorithm's Asymptotic Behavior – "Eventually" is a LONG Time

- **LOTS of methods can guarantee to find the best solution, with probability 1, eventually…**
  - **Enumeration**
  - **Random search (better without resampling)**
  - **SA (properly configured)**
  - **Any GA that avoids "absorbing states" in a Markov chain**
- **The POINT: you can't afford to wait that long, if the problem is anything interesting!!!**

When Might a GA Be Any Good?

- Highly multimodal functions
- Discrete or discontinuous functions
- High-dimensionality functions, including many combinatorial ones
- Nonlinear dependencies on parameters (interactions among parameters) -- “epistasis” makes it hard for others
- Often used for approximating solutions to NP-complete combinatorial problems
- DON’T USE if a hill-climber, etc., will work well

The Limits to Search

- **No search method is best for all problems – per the No Free Lunch Theorem**
- **Don't let anyone tell you a GA (or THEIR favorite method) is best for all problems!!!**
- **Needle-in-a-haystack is just *hard*, in practice**
- **Efficient search must be able to EXPLOIT correlations in the search space, or it's no better than random search or enumeration**
- **Must balance with EXPLORATION, so don't just find nearest local optimum**

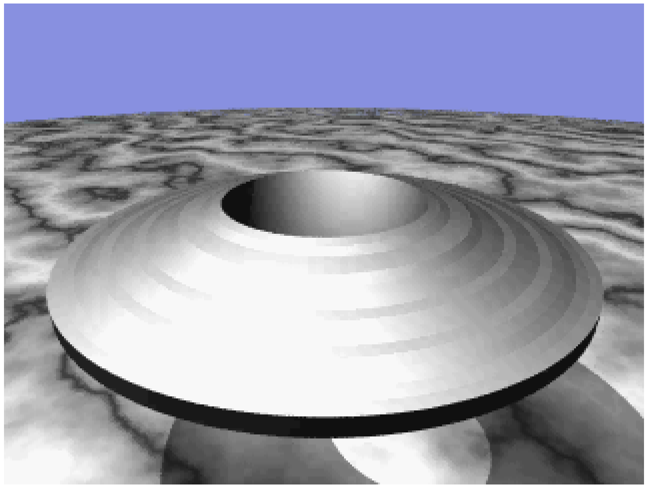
Examples of Successful Real-World GA Application

- Antenna design
- Drug design
- Chemical classification
- Electronic circuits (Koza)
- Factory floor scheduling (Volvo, Deere, others)
- Turbine engine design (GE)
- Crashworthy car design (GM/Red Cedar)
- Protein folding
- Network design
- Control systems design
- Production parameter choice
- Satellite design
- Stock/commodity analysis/trading
- VLSI partitioning/ placement/routing
- Cell phone factory tuning
- Data Mining

EXAMPLE!!!
Let's Design a Flywheel

GOAL: To store as much energy as possible (for a given diameter flywheel) without breaking apart

- On the chromosome, a number specifies the thickness (height) of the "ring" at each given radius
- Center "hole" for a bearing is fixed
- To evaluate: simulate spinning it faster and faster until it breaks; calculate how much energy is stored just before it breaks



Flywheel Example

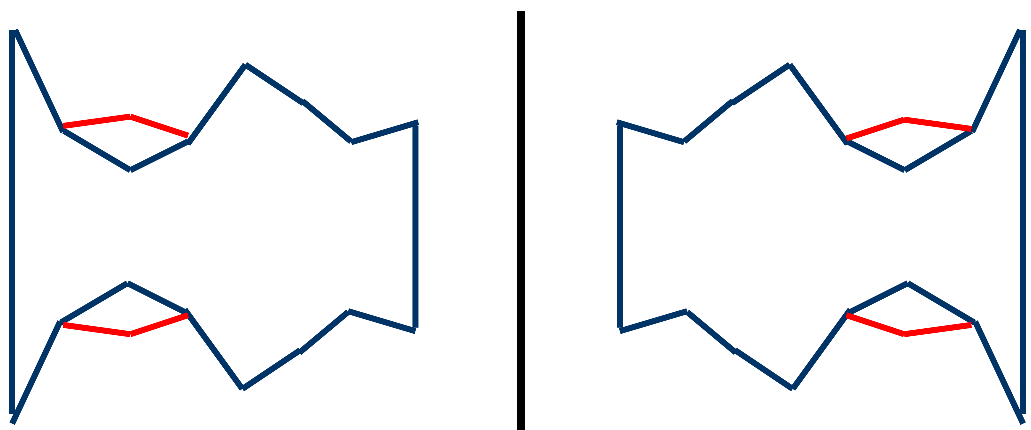
So if we use 8 rings, the chromosome might look like:

6.3 3.7 2.5 3.5 5.6 4.5 3.6 4.1

If we mutate HERE, we might get:

6.3 3.7 4.1 3.5 5.6 4.5 3.6 4.1

And that might look like (from the side):



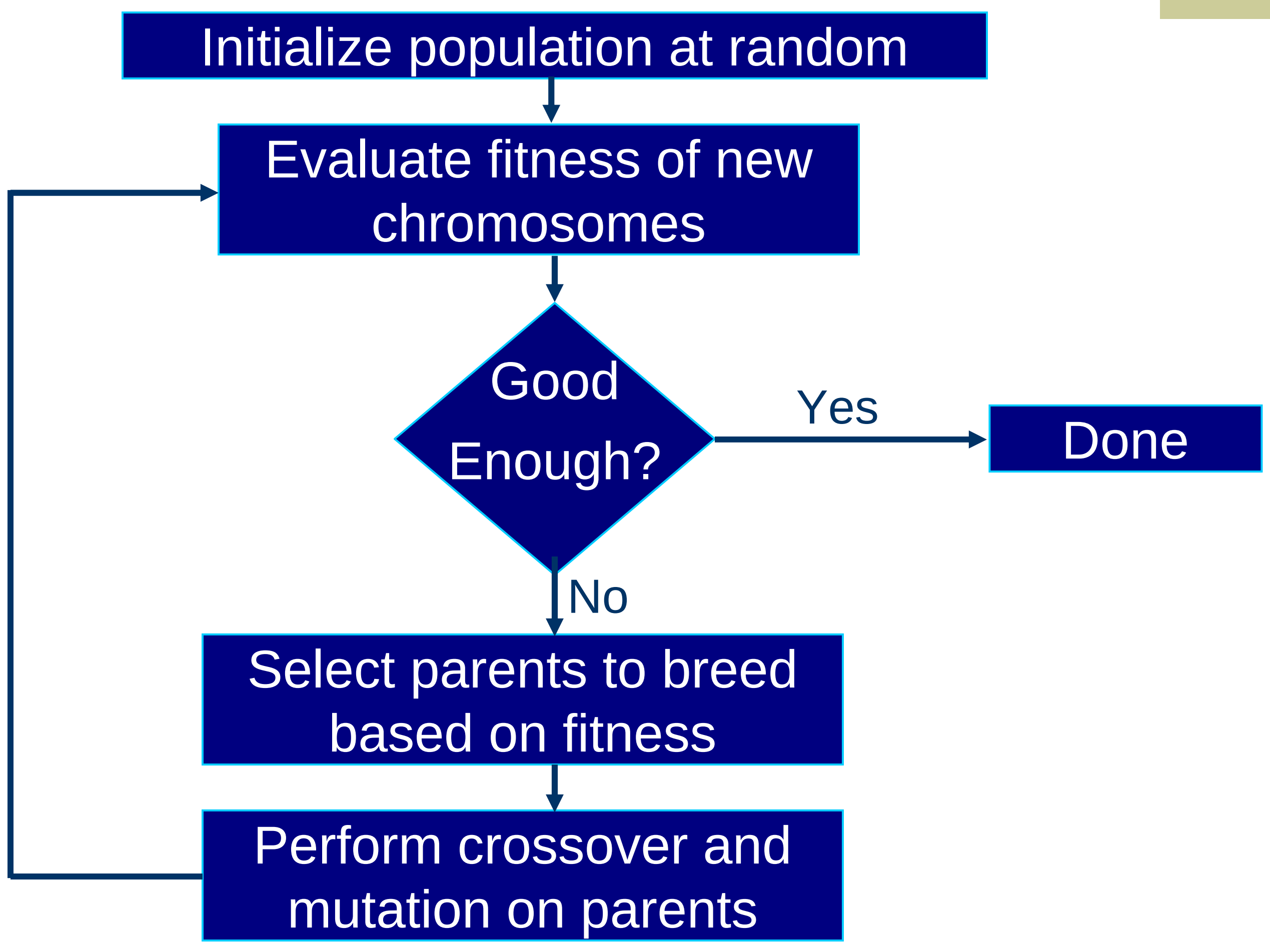
Recombination ("Crossover")

If we recombine two designs, we might get:

6.3	3.7	2.5	3.5	5.6	4.5	3.6	4.1
				x			
3.6	5.1	3.2	4.3	4.4	6.2	2.3	3.4
3.6	5.1	3.2	3.5	5.6	4.5	3.6	4.1

This new design might be BETTER or WORSE!

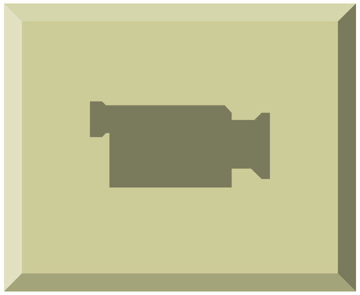
Typical GA Operation -- Overview



A GA Evolves the Flywheel:

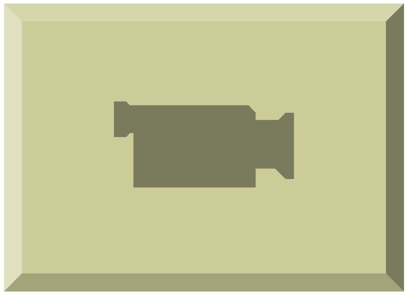
One

Material



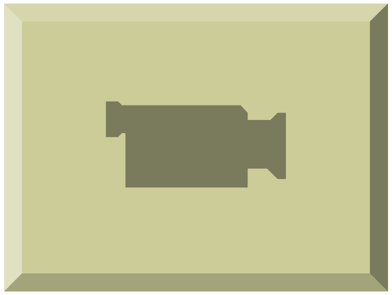
Choice of

Materials



Choice

(side view)



"Genetic Algorithm" -- Meaning?

- "classical or canonical" GA -- Holland (taught in '60's, book in '75) -- binary chromosome, population, selection, crossover (recombination), low rate of mutation
- More general GA: population, selection, (+ recombination) (+ mutation) -- may be hybridized with LOTS of other stuff

Representation Terminology

- Classically, binary string: individual or chromosome
- What's on the chromosome is GENOTYPE
- What it *means* in the problem context is the PHENOTYPE (e.g., binary sequence may map to integers or reals, or order of execution, or inputs to a simulator, etc.)
- Genotype and problem environment determine phenotype, but phenotype may *look* very different

In PRACTICE – GAs Do a JOB

- **DOESN'T mean necessarily finding *global optimum***
- **DOES mean *trying* to find *better* approximate answers than other methods do, within the time available!**
- **People use any "dirty tricks" that work:**
  - **Hybridize with local search operations**
  - **Use multiple populations/multiple restarts, etc.**
  - **Use problem-specific representations and operators**
- **The GOALS:**
  - **Minimize # of function evaluations needed**
  - **Balance exploration/exploitation so get best answer can during time available (AVOIDING *premature convergence*)**

Other Forms of GA

Generational vs. "Steady-State"

- **"Generation gap": 1.0 means replace ALL by newly generated "children"**
- **at lower extreme, generate 1 (or 2) offspring per generation (called "steady-state") – no real "generations" – children ready to become parents on next operation**

More Forms of GA

Replacement Policy:

1. **Offspring replace parents**
2. **K offspring replace K worst ones**
3. **Offspring replace random individuals in intermediate population**
4. **Offspring are "crowded" in**
5. **"Elitism" – always keep best K**

How Do GAs Go Bad?

- **Premature convergence**
- **Unable to overcome deception**
- **Need more evaluations than time permits**
- **Bad match of representation/mutation/crossover, making operators destructive**
- **Biased or incomplete representation**
- **Problem too hard**
- **(Problem too easy, makes GA *look* bad)**

So, in Conclusion…

- GAs can be easy to use, but not necessarily easy to use WELL
- Don’t use them if something else will work – it will probably be faster
- GAs can’t solve every problem, either…
- GAs are only one of several strongly related “branches” of evolutionary computation – and they all commonly get hybridized
- There’s lots of expertise at GECCO – talk to people for ideas about how to address YOUR problem using evolutionary computation