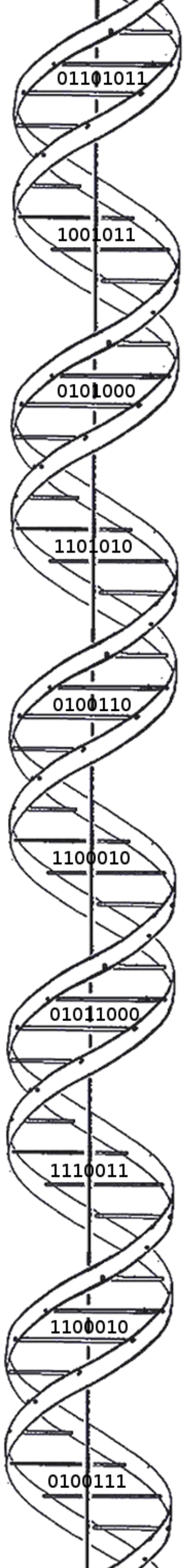
Genetic Algorithms

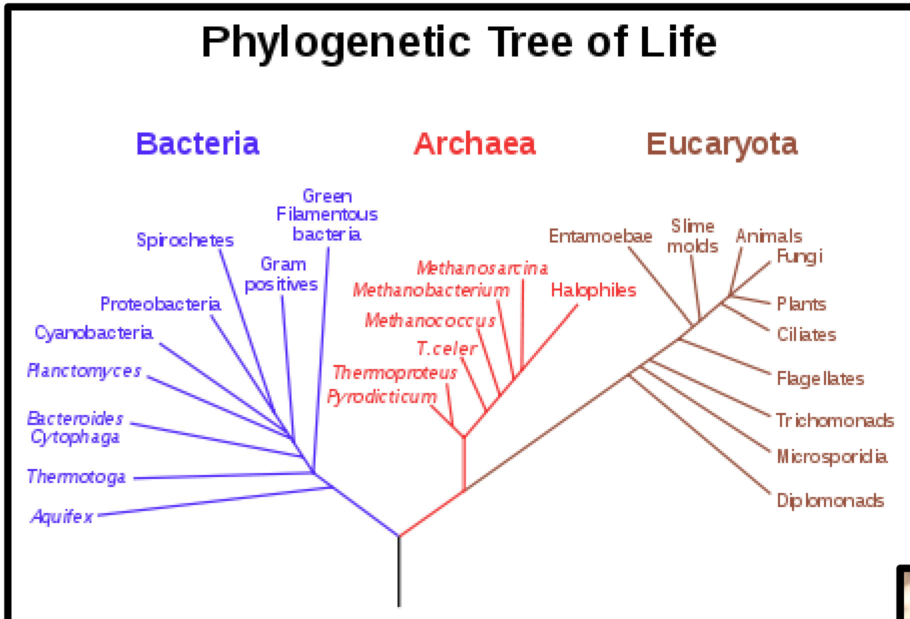
Forrest Stonedahl

EECS 349: Machine Learning

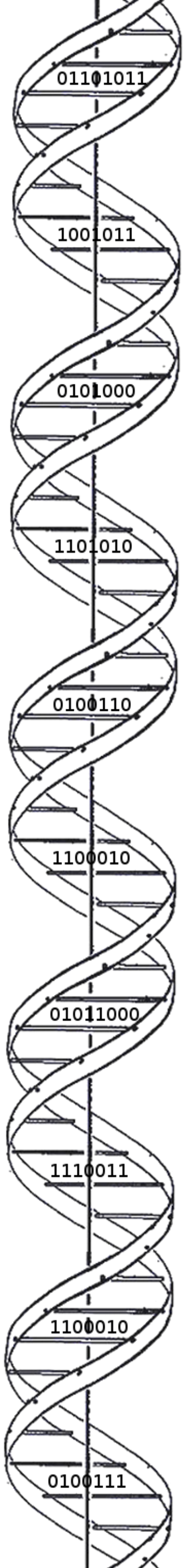
October 16, 2009



Evolution (a non-biologist's guide...)

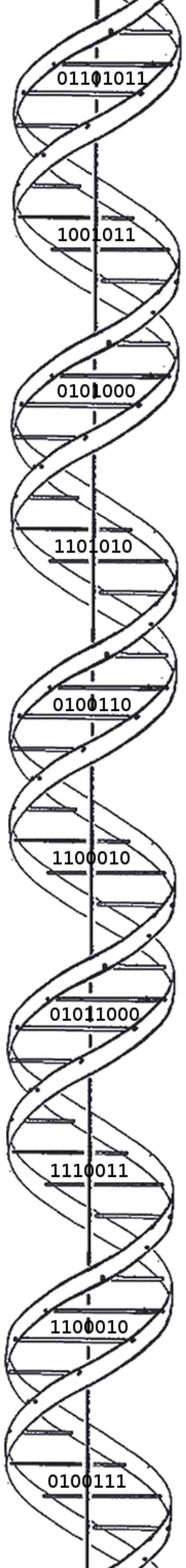


(image credits: public domain)



Evolution (a non-biologist's guide...)

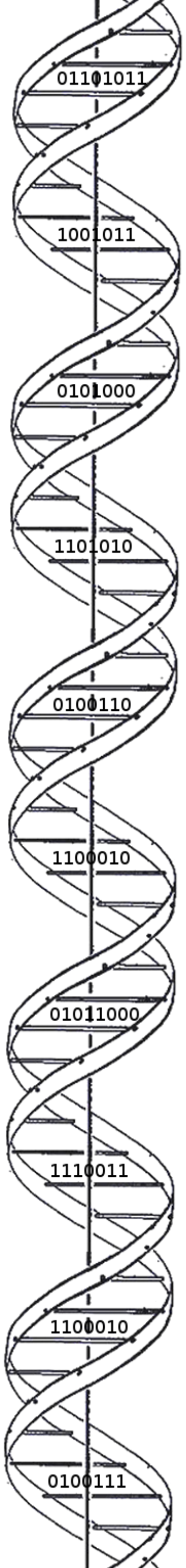
- There is a population of creatures
- Creatures reproduce
  - Children are like parents, but different.
- Creatures die
  - Some animals are more likely to survive and reproduce. They are considered “more fit”.
- Over time the population will resemble the successes more than the failures



Evolution (a non-biologist's guide...)

Key ideas:

- Variation
  - Mutation & recombination
  - Allows new favorable (or unfavorable) features to appear in children.
- Selection
  - Causes the population to adapt to its environment.



Genetics (a non-biologist's guide...)

- **Gene:** basic hereditary unit (information)
- **Chromosome:** sequence of genes
- **Genotype:** chromosome-level genetic information about a creature
  - (e.g. genes that influence growth-rate)
- **Phenotype:** a creature's actual traits
  - (e.g. short, tall, or blue-eyed).
- The mapping between genotype and phenotype is often very complex, involving cell development, etc.

Harnessing Evolution/Genetics

Siamese
(my cat)

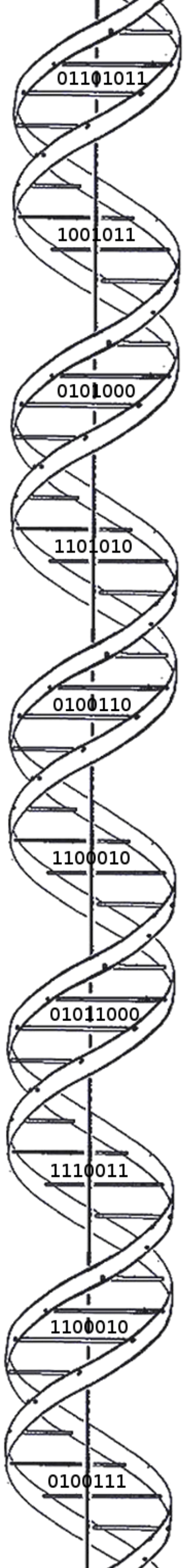
Pekingese
(thankfully not my dogs)

≠

Wolf
(just for comparison)

- Selective breeding
- More recently, genetic engineering

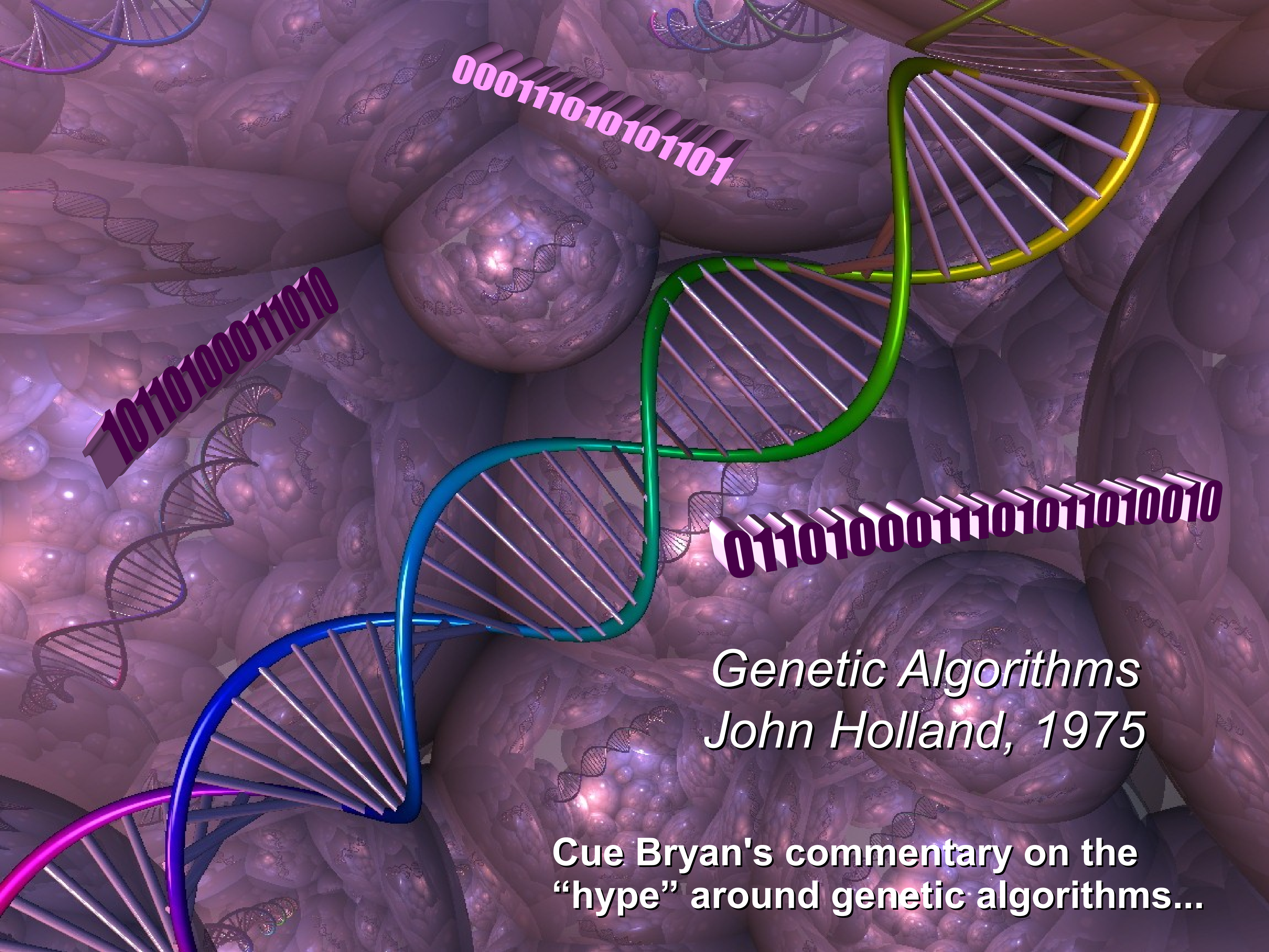
image credits: cat - Susa Stonedahl (2007), dogs & wolf - public domain



A key insight

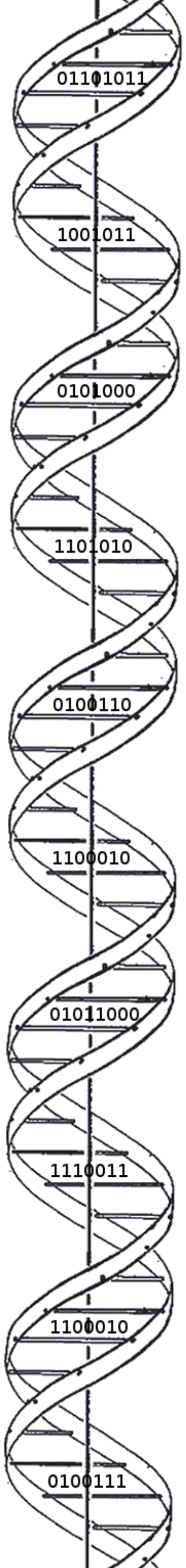
Evolution is a powerful “problem-solving technique”...

We can apply this nature-inspired approach to solve all sorts of problems, by simulating evolution in a computer algorithm.



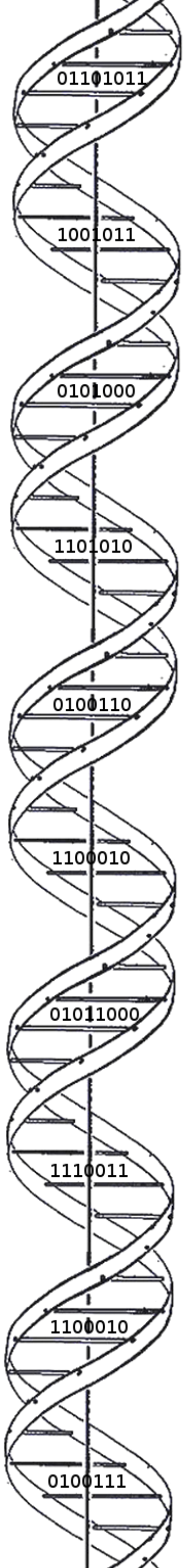
Genetic Algorithms
John Holland, 1975

Cue Bryan's commentary on the "hype" around genetic algorithms...



Evolutionary Algorithms

- How can we simulate evolution on the computer to solve problems?
- Virtual population of “candidate” solutions.
- Some form of reproduction, to create new different candidates from the existing ones (for **variation**)
- Some way of measuring the “fitness” of a candidate (for **selection**)

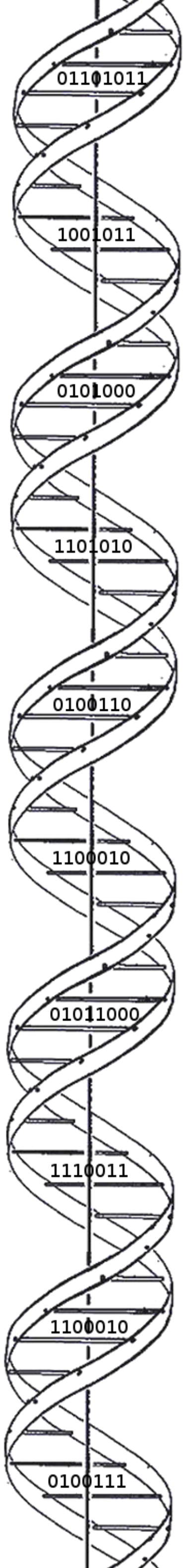


Genetic Algorithm Ingredients

- an encoding for candidate solutions
- an initial population
- “fitness” function
  - for phenotype *selection*
- genetic operators
  - for genotype *variation*
- reproduction model
  - to put it all together

GA schematic

- Start with random population
- Loop until "good enough" solution found
  - Evaluate fitness on each individual
  - Choose parents from this population, preferentially selecting "fitter" ones
  - Create children from the chosen parents
    - Using sexual & asexual reproduction, and some amount of mutation
  - Replace (at least part of) the old population with these children



Example: elephant bath time

Name	9:00	9:30	10:00	10:30	11:00
Allie	X				
Bubs		X			
Candy	X				
Dumbo				X	
Elle		X			

One candidate schedule. (Phenotype)

`10000 01000 10000 00010 01000`
`Ally  Bubs  Candy Dumbo Elle`

(Genotype)

One possible encoding for this schedule.

Photo credit: Susa Stonedahl (2008)

Example: elephant bath time

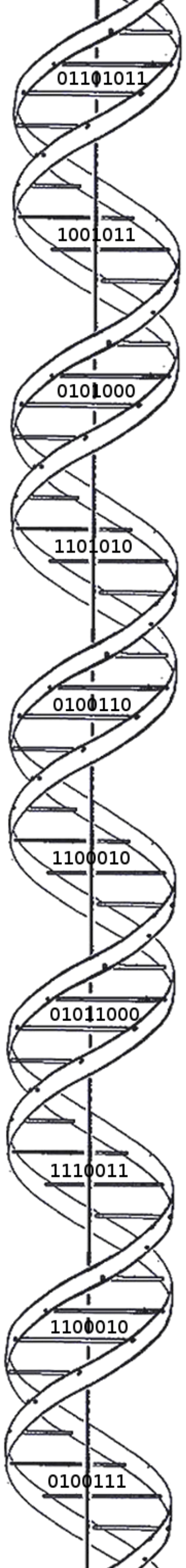
Name	9:00	9:30	10:00	10:30	11:00
Allie	X				
Bubs		X			
Candy	X				
Dumbo				X	
Elle		X			

```
10000 01000 10000 00010 01000
Ally  Bubs  Candy Dumbo Elle
```

Q: Does every genotype map to a sensible phenotype?

Q: And is every phenotype representable?

Q: What other encodings could we choose?



Genotype Representations

- Bit-string encoding for candidate solutions

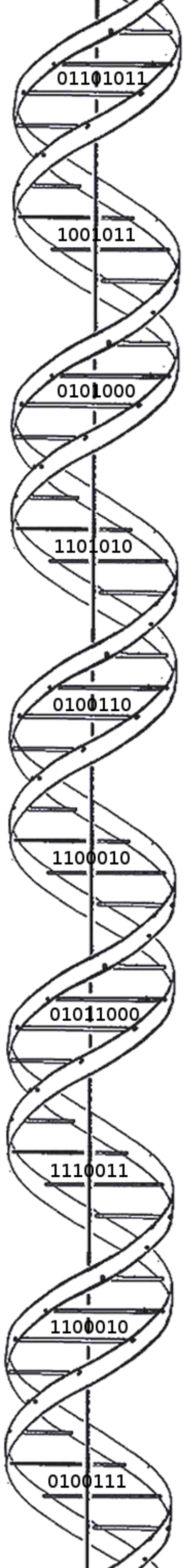
Allie & Candy, then Bubs & Elle, & Dumbo alone

= 1000001000100000001001000

Allie & Dumbo, then Bubs alone, & Candy & Elle

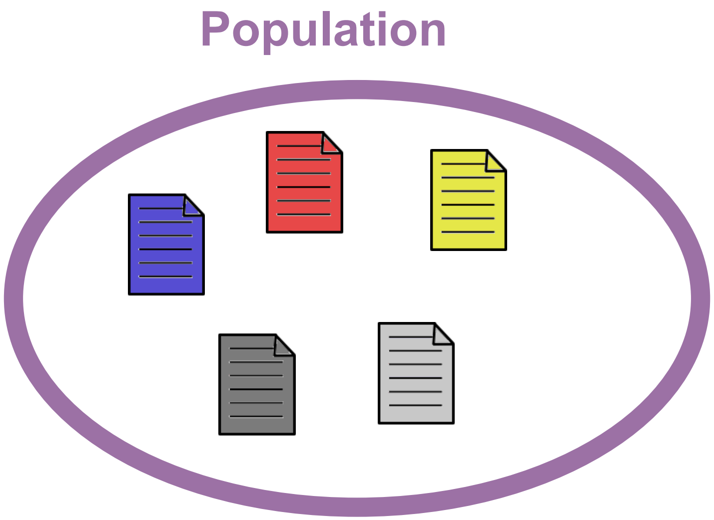
= 1000000100000011000000001

- (alternative integer encoding)
  - <AliceTime,BubsTime,CandyTime,DumboTime,...>
  - <1, 3, 1, 2, 4>



Initial Population

- Start with randomly generated genotypes
- Population size usually at least 50, could be 1000s
- PopSize is a GA parameter to vary.

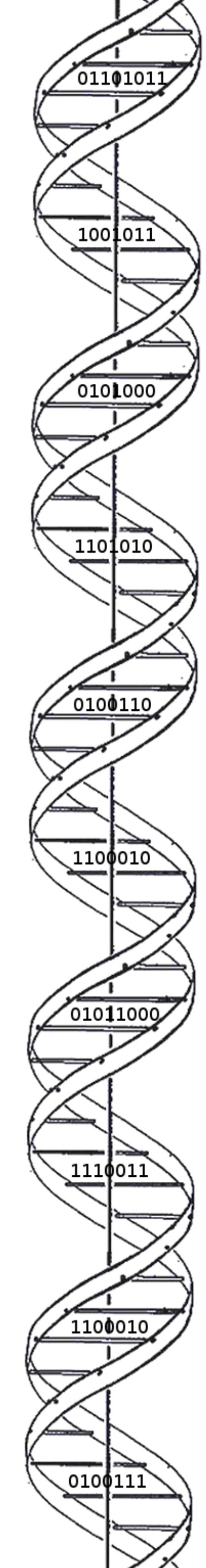


 = 10000010001000000001001000

 = 1000000100000011000000001

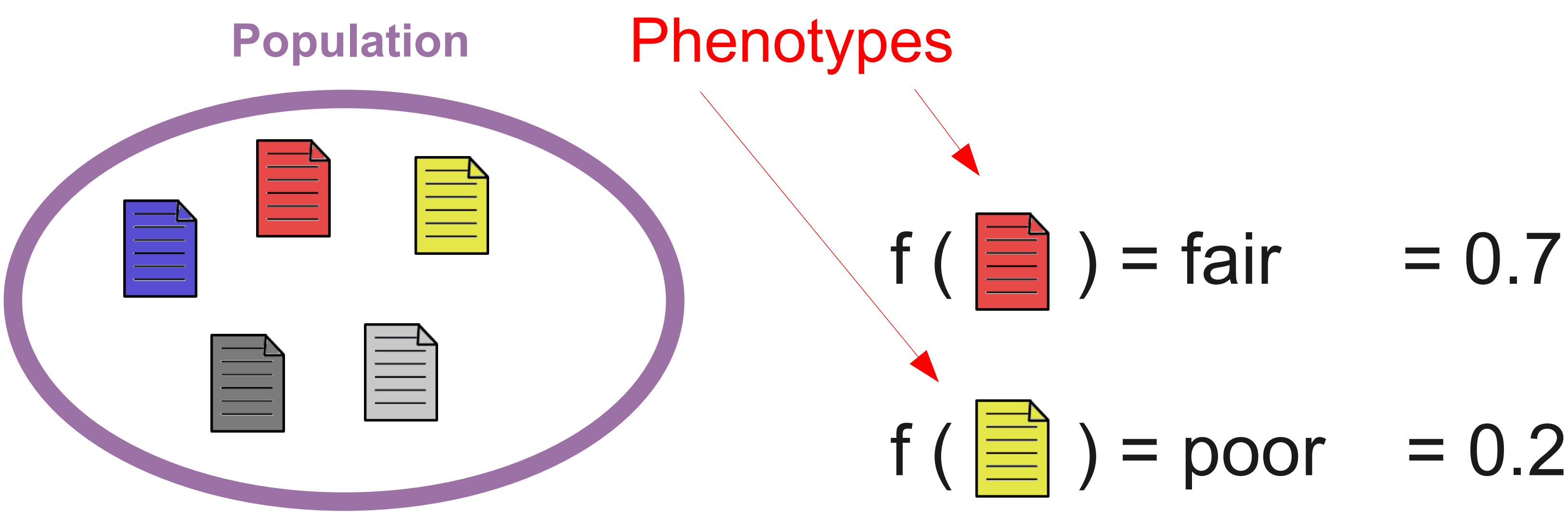
 = 0110101000110100001010000

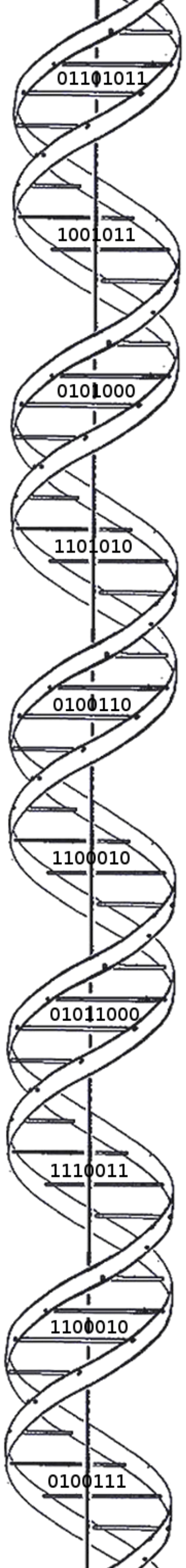
etc...



Fitness Function

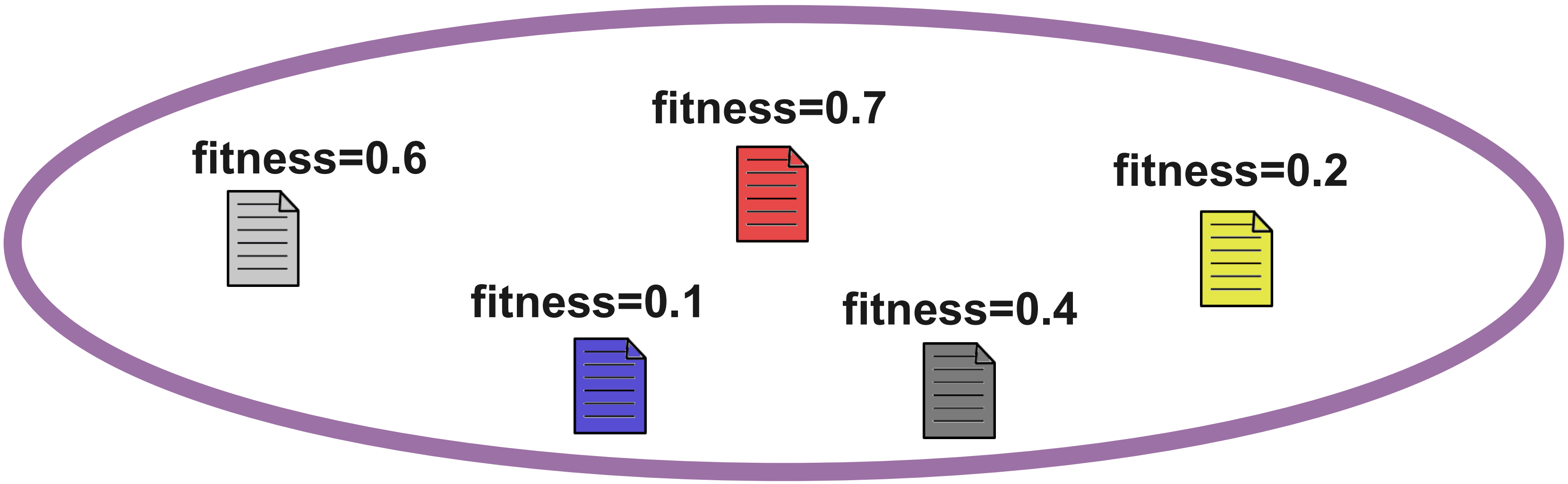
- How good is a given bath schedule?
  - **simple** = efficiency – penalty for constraints
  - **complex** = also consider P(conflict) given elephant personality matrix + bonus for elephants bathing with friends



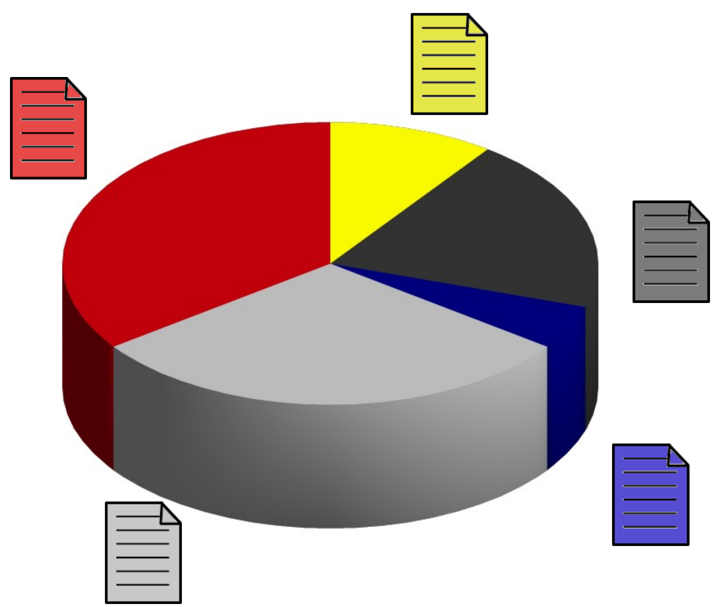


Fitness-proportional selection

a.k.a. “roulette selection”

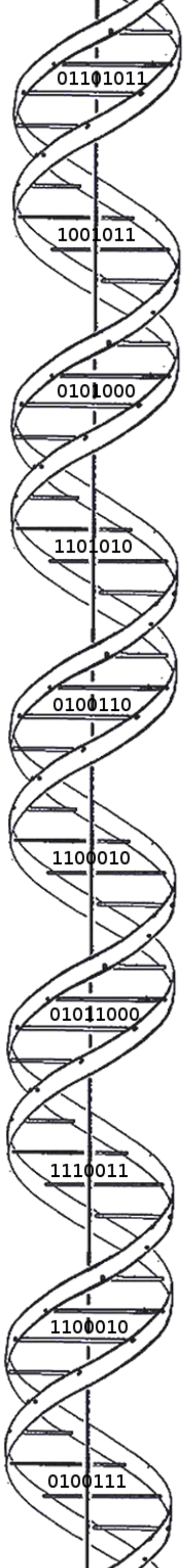


Choose individuals for reproduction with probability proportional to fitness.

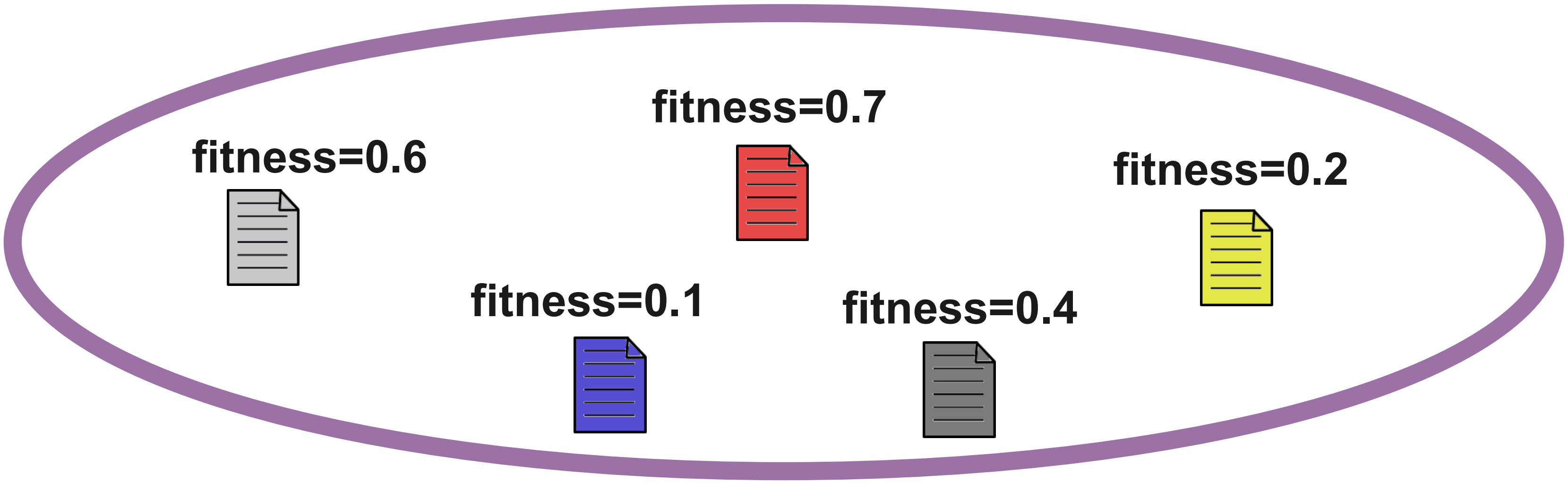


Potential Issues

- Premature population convergence
  - diversity is important!
- Loss of “selection pressure”.
  - At the end of the run, e.g. 99 isn't much more likely to be selected than 98...
- Affected by function transposition
  - f(x) = 2 - (# of errors) or 100 - (# of errors)
- Windowing & scaling can help
  - e.g. fitness relative to worst-in-population



Rank & tournament selection



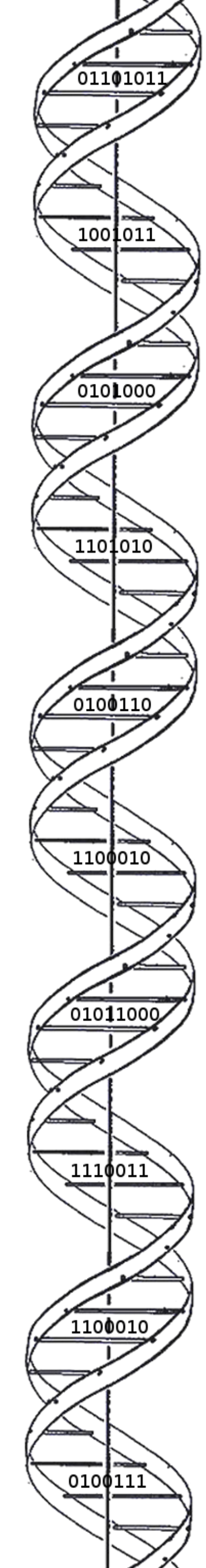
Rank order:

Requires sorting (but usually running time is dominated by fitness evaluation.)

Tournament: sample *k* at random from the population, and select the best.

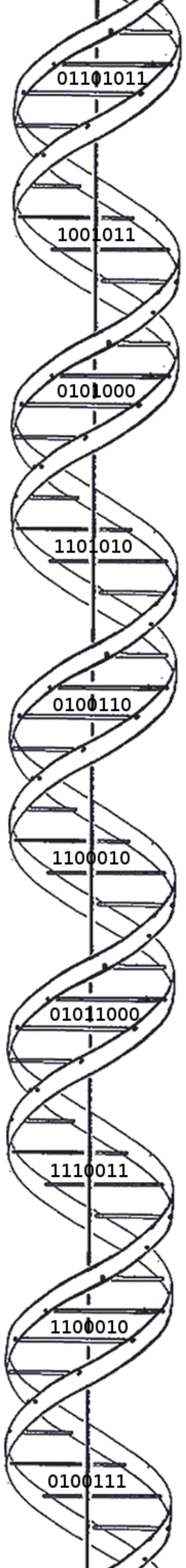
e.g. Best of ()

Doesn't need global information!

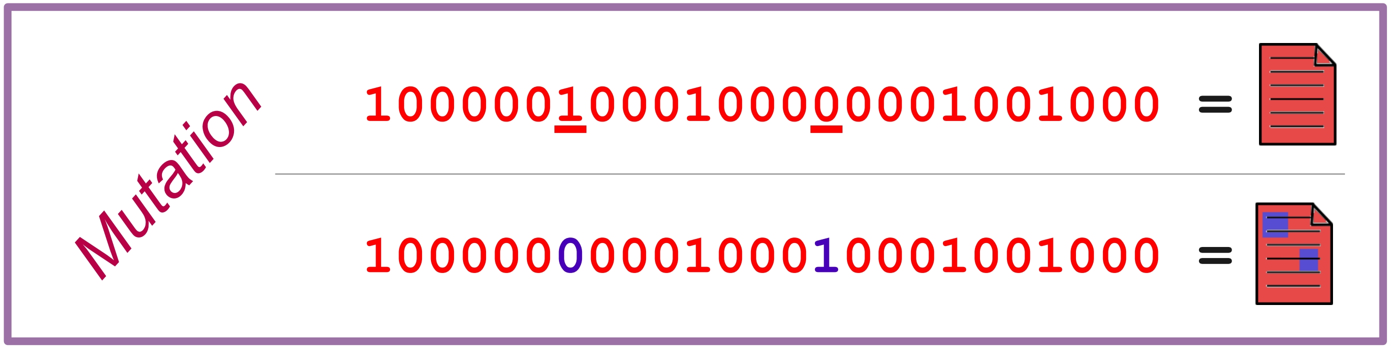


Recombination operators

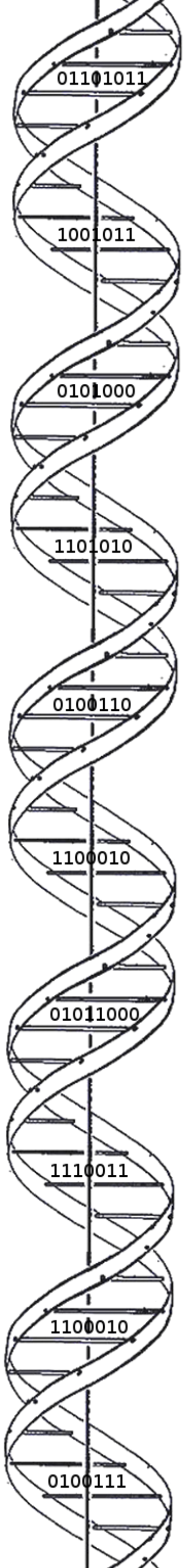
- Variants: 2-point, n-point, uniform
- Crossover is a hallmark of GAs
- Intuition: combine building blocks
  - BUT, does the representation suit well?



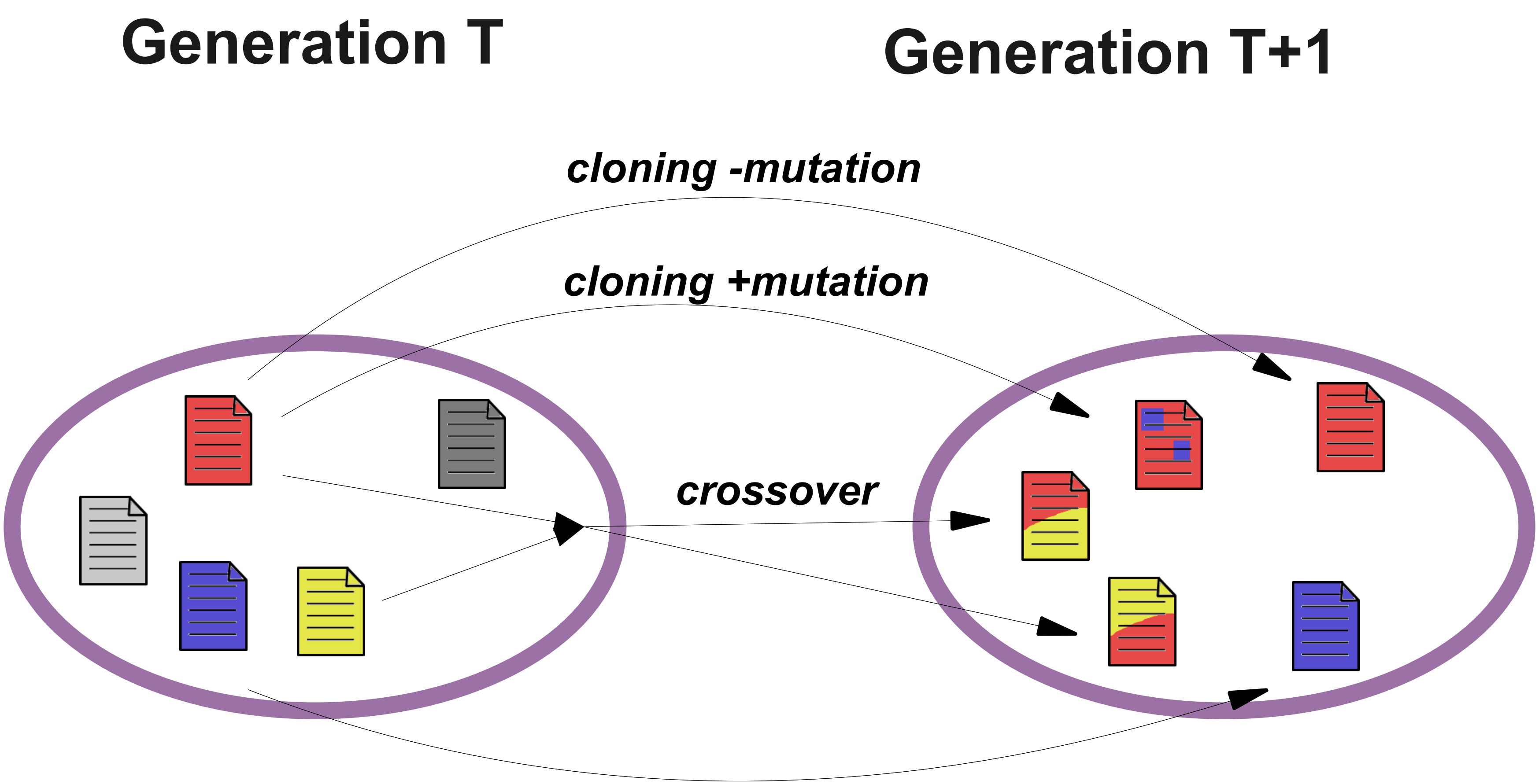
Mutation operators



- Per-bit mutation
  - For each bit, P(flip) = k
- Common mutation rates:
  - 1 / (2L) where L = bit string length
  - Sometimes fixed at < 1%
- Source of “new” information in the GA.



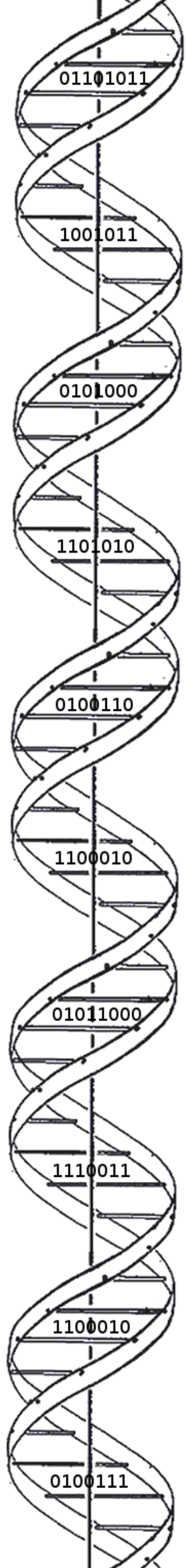
“The Next Generation”



Lather, rinse, and repeat until satisfied...

Population-replacement models

- Generational (classic, simple GA)
  - replace everyone
- Generational gap model
  - Replace X% of population
- Steady-state model
  - Choose someone to remove
  - Create one individual to add
- “Elitism”
  - Guarantee the best Y% will survive.

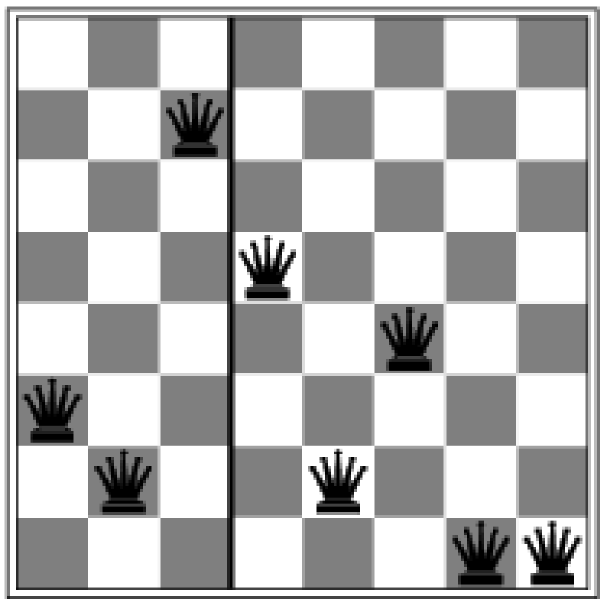


Example 2: NQueens

Genotype

the encoding operated on by mutation and inheritance

3, 2, 7, 5, 2, 4, 1, 1



Phenotype

the “real” thing, (ideally) operated on by the fitness function

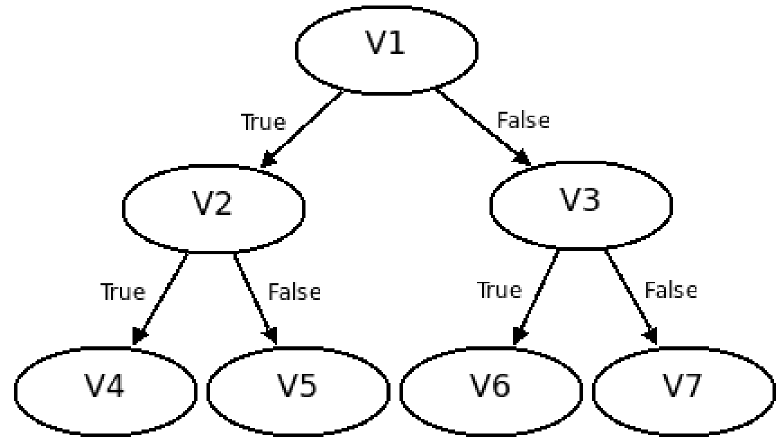
Photo credit: public domain

How else could we encode the genotype for chess positions?

This slide adapted from Bryan Pardo, EECS 349 Fall 2007

Example 3: Decision Trees

Genotype representation:
$<V_1,V_2,V_3,V_4,V_5,V_6,V_7>$



Where each V_i =

- 0 if the node is a FALSE leaf
- 1 if the node is a TRUE leaf
- K for splitting on the (K-1)st attribute.

Question: What about the bottom tree layer ($V_4...V_7$)?

Example Attribute Set: {IsSmoker, Exercises}

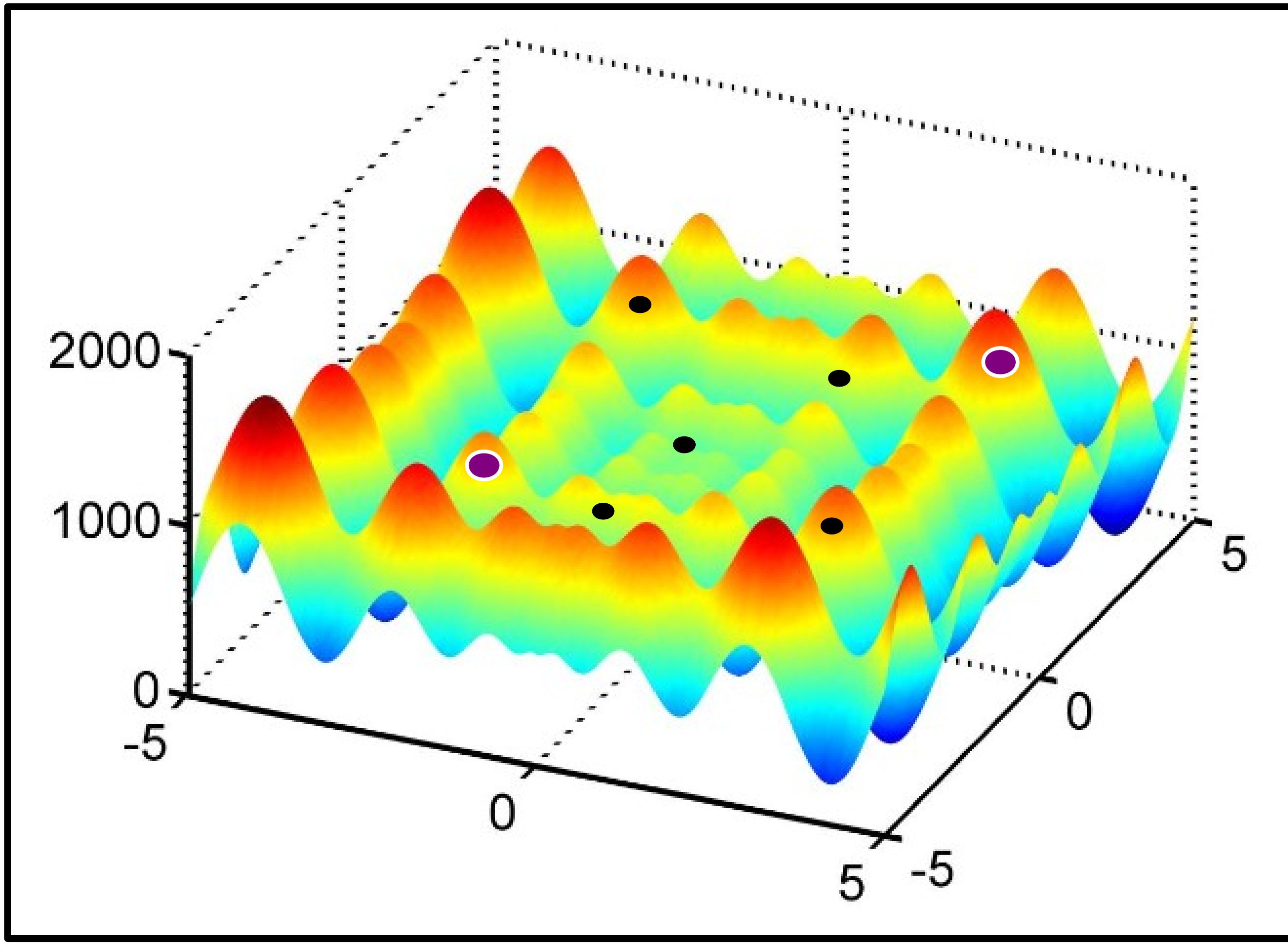
What is the phenotype for: A) $<0,2,3,1,0,2,1>$?
B) $<2,2,2,2,2,2,2>$?

More Genotype Representations

- Real-valued <3.729, 0.21, 11.9…>
  - Gaussian mutation
- Permutation-based
  - swapping mutations
  - permutation crossover
- Nonlinear
  - Trees, 2D arrays, graphs

Photo credit: http://www.swissarmy.com/

Fitness landscapes



Usually very high-dimensional, not 2D.

Image credit: Susa Stonedahl (2009)

--continuing from Friday--

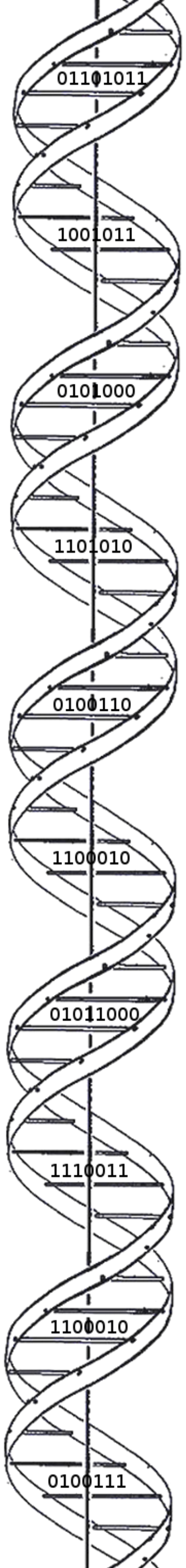
First, a quick review...

Review:GA Ingredients

- an encoding for candidate solutions
- an initial population
- “fitness” function
  - for phenotype *selection*
- genetic operators
  - for genotype *variation*
- reproduction model
  - to put it all together

Tennis Predictor Example

- Outlook = {Sunny, Overcast, Rain}
- Wind = {Weak, Strong}
- Given 100 training examples like:
    - Sunny, Strong, YES
    - Rain, Weak, NO
- Should you play tennis?
- How can we design a GA to learn the PlaysTennis concept?



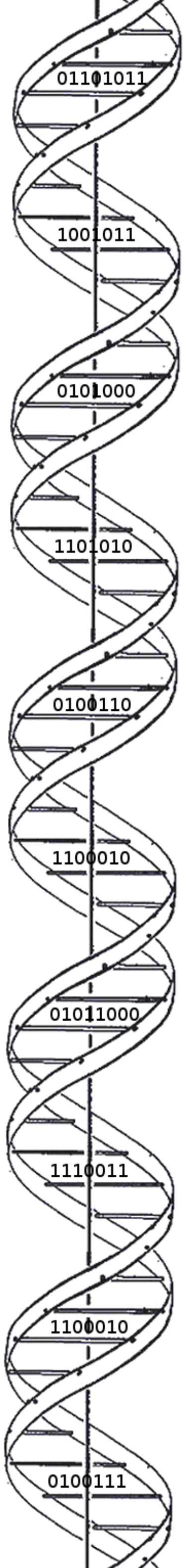
Representing simple rules

— — — — — — —

Outlook Wind PlaysTennis

- If (Outlook=X or Y or Z) AND Wind=(A or B) Then PlaysTennis = YES or NO.
- Outlook = {Sunny, Overcast, Rainy}
- Wind = {Weak, Strong}
- Classification: PlaysTennis = {YES, NO}

- What does 011 01 01 mean?
- What does 100 11 10 mean?
- What does 000 00 00 mean?

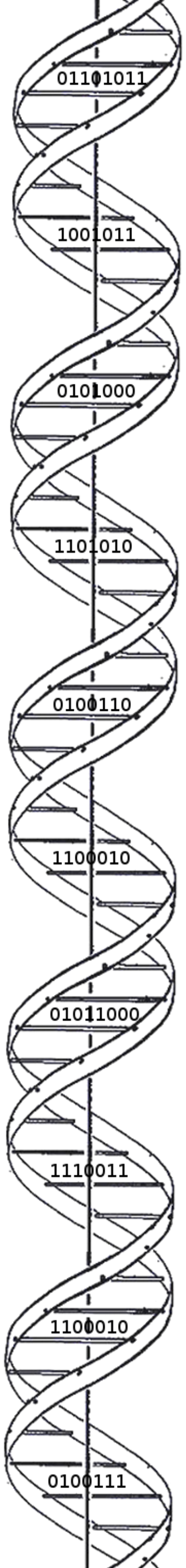


Review: GA Ingredients

- an encoding for candidate solutions **DONE**
- an initial population **Random bit strings will do. Try PopSize=200...**
- “fitness” function **Discuss!**
  - for phenotype *selection*
- genetic operators
  - for genotype *variation*
- reproduction model
  - to put it all together

Fitness & Selection

- One possibility:
  - Fitness F = % correct on training set
- Select who will reproduce using:
  - Tournament selection
    - Look at 3 random individuals and select the best.



Genetic Operators

- 2-point crossover

Parents

0110101 (don't play in non-sunny strong wind)

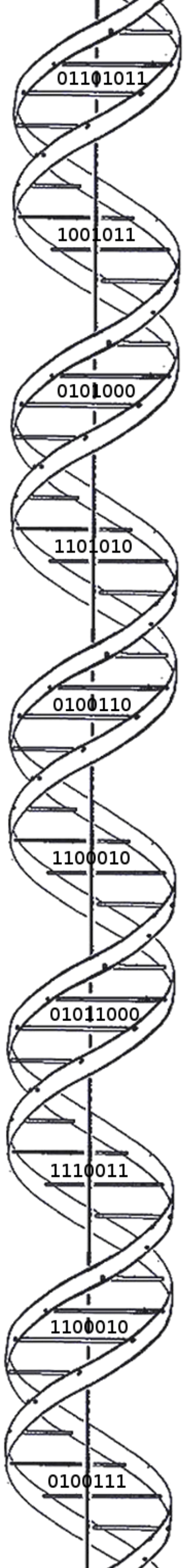
1001110 (do play when sunny, in any wind)

Children

0111101 (don't play when non-sunny in any wind)

1000110 (do play in sunny strong wind)

- Per-bit mutation, perhaps rate = 1%
  - 1% chance of flipping each bit in the children.



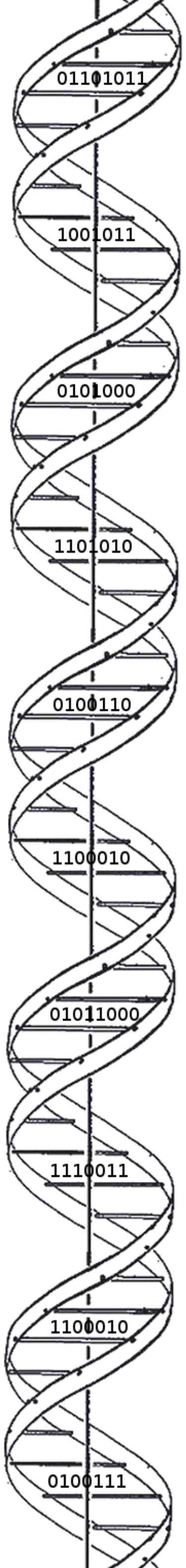
Review: GA Ingredients

- an encoding for candidate solutions — **DONE**
- an initial population — **Random bit strings will do.**
- "fitness" function
  - for phenotype *selection*

  F = training set score.
  Tournament selection.
- genetic operators
  - for genotype *variation*

  2-pt crossover
  & mutation
- reproduction model
  - to put it all together

  Generational



Review: GA schematic

- Start with random population
- Loop until "good enough" solution found
  - Evaluate fitness on each individual
  - Choose parents from this population, preferentially selecting "fitter" ones
  - Create children from the chosen parents
    - Using sexual & asexual reproduction, and some amount of mutation
  - Replace (at least part of) the old population with these children

A leading question...

If genetic algorithms are “evolving” solutions, that sounds really flexible...

Are there any optimization problems that GAs aren't good at solving?

“No Free Lunch” Theorem

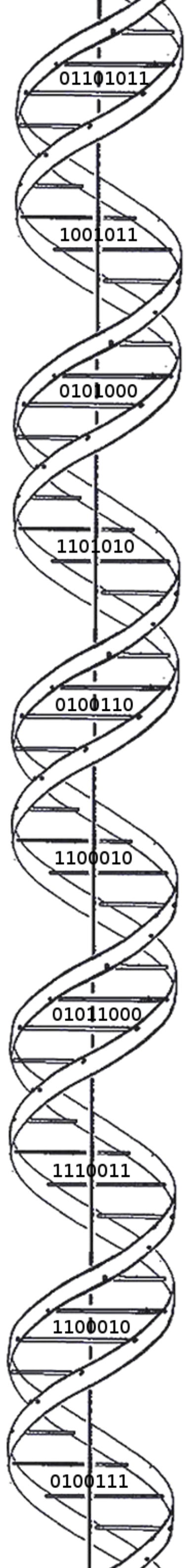
- All search algorithms are biased.
  - If they perform better on one function, it is at the cost of performing worse on another.
- No search algorithm is any better than random search, across the set of all fitness functions.
- (I'm glossing over details...)

NFL due to: Wolpert & Macready (1997)

Image credit: http://www.townofrosendale.com/images/lunch_sign.jpg

Before applying a GA?

- Is there a domain-specific approach you could try?
  - GA is a "black box" optimizer
  - Can incorporate domain-specific operators into the GA as well...
- Do greedy/local algorithms fail?
  - Do they get stuck on local optima?
- Think hard about search space representation!



Thoughts on using GAs

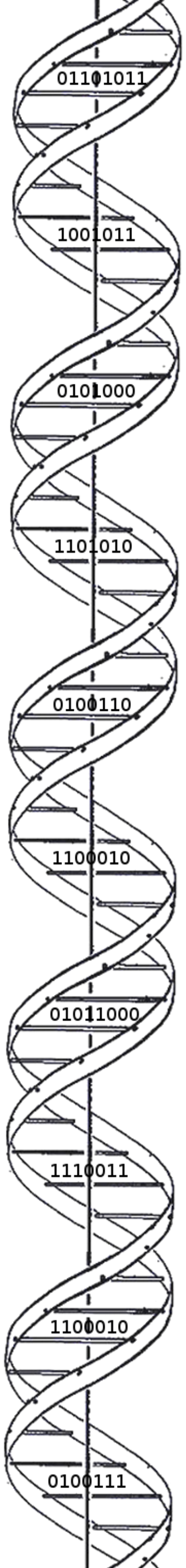
- The chromosomal representation should encourage recombination of useful “building blocks.” Can the solution be built from subcomponents?
- The fitness function must provide sufficient search gradient.
  (Won't find a “needle in a haystack”.)
- Biological evolution is not really an “optimization” process. Rather, it is a complex adaptive system. This can also be helpful for thinking about GAs.

GAs and speed

- GAs are often slow
  - (in their defense, the problems are often pretty challenging.)

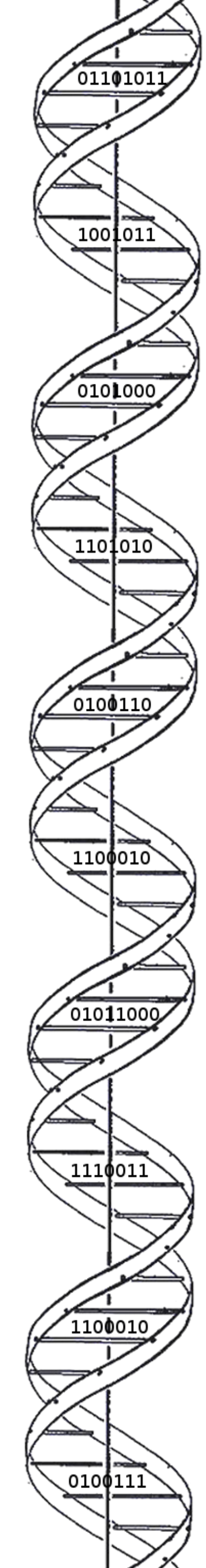
Photo credit: public domain

- One response: parallelization
  - island-migration models (“demes”)
  - fine-grained parallelization



"Evolutionary Computation"

- GAs fall into a larger family of evolutionary algorithms, including
  - Genetic Programming ← Coming up!
  - Evolutionary Strategies
  - Evolutionary Programming
  - EDAs, DE, GE, Harmony Search...
- Artificial life ("Alife")
  - Simulating (or creating?) virtual life

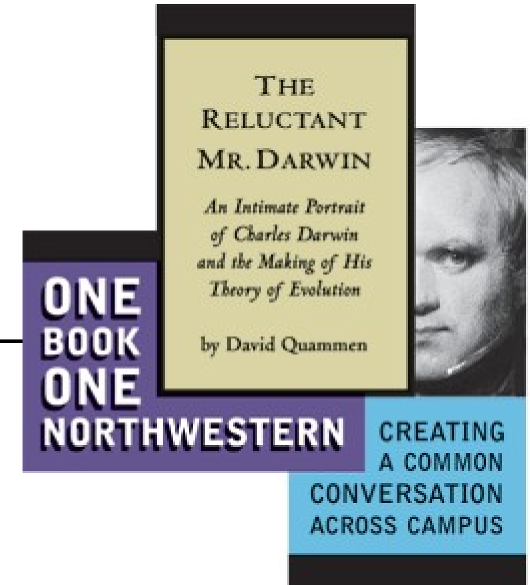


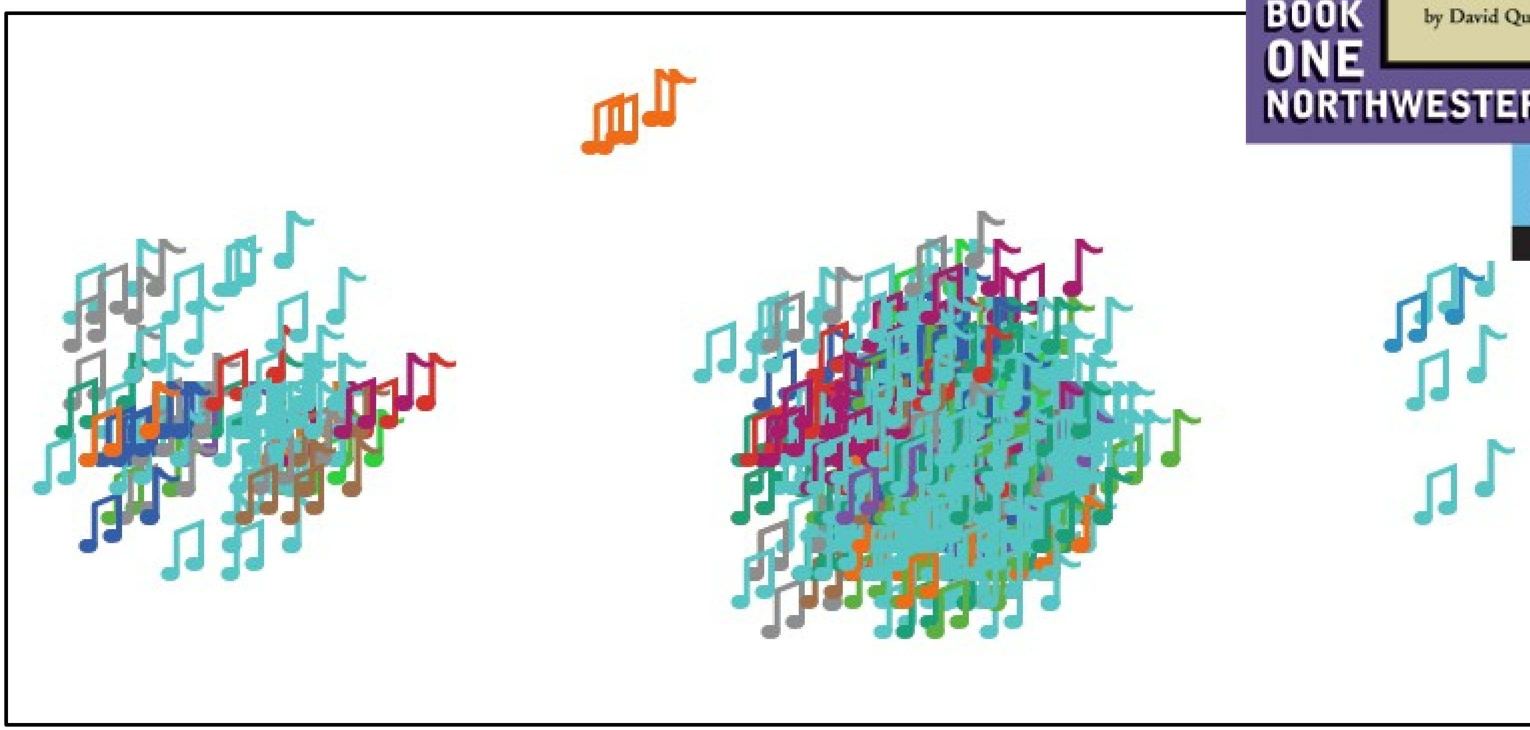
GA Demo

1st place prize

"Art of Evolution" Exhibit

February 12, 2009





(Switch Slides)

to Genetic Programming

A few fun topics

Photo credit: public domain

Interactive GAs

- Require human interaction & feedback for the fitness function
- Can be used to evolve art, music...
- Example: online banner ads
  - Try different combinations of fonts, background/foreground colors, sizes, accompanying photos, etc.

Example artwork created by an IGA. *Image credit: kandid.sourceforge.net*

Coevolution

- Consider two populations, each evaluating fitness based on the other

Example:

- 1 population of parallel sorting networks
- 1 population of "input sequences"

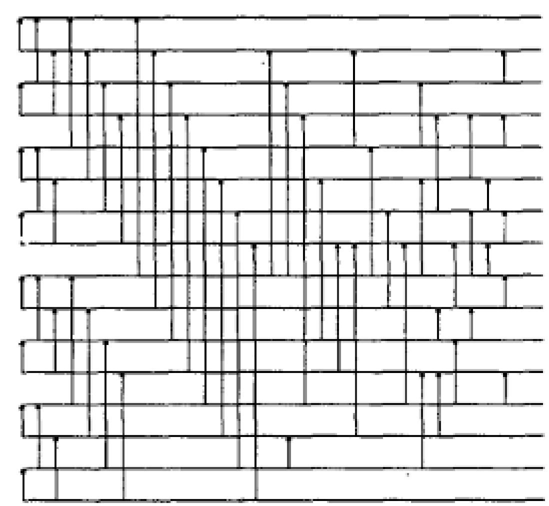


Fig. 3. 61 exchanges.

This figure courtesy of:
Hillis, W.D. (1990)
"Coevolving parasites..."

Learning Classifier Systems

- evolve a population of rules
  - rules can trigger other rules based on message passing
- the whole population = the classifier
- use “credit assignment” to reward useful rules with good fitness

- (combines GAs with reinforcement learning)

In conclusion...

- GAs are fun...
  - So you should do your homework!
- Evolutionary algorithms can evolve creative and unexpected solutions to difficult problems.
- But you only get intelligence out, if you put some intelligence in!
  - well-designed problem representation
  - fitness function
  - appropriate parameter settings