Genetic Algorithms and Evolution Strategies

What is Evolutionary Computation?

- An abstraction from the theory of biological evolution that is used to create optimization procedures or *methodologies*,
  - usually implemented on computers,
  - that are used to **solve problems**.

The Argument

Evolution has optimized biological processes;

therefore

Adoption of the evolutionary paradigm to computation and other problems can help us find optimal solutions.

Example abound around you…..

Components of Evolutionary Computing

- **Genetic Algorithms**
  - invented by John Holland (University of Michigan) in the 1960's
- Evolution Strategies
  - invented by Ingo Rechenberg (Technical University Berlin) in the 1960's
- Started out as individual developments, but have begun to converge in the last few years

Presentation Agenda

- What is **evolutionary computing**?
- First we will present evolutionary theory from a **biological** perspective
- Why use evolution as a model for solving computational problems?
- When to use evolutionary computing strategies?

Presentation Agenda (2)

- Genetic Algorithms - in depth
- Evolution Strategies - in depth
- Drawbacks of GA's and ES's
- When to use GA's over ES's
- Questions

A primer on evolutionary theory from a biological perspective

The Concept of Natural Selection

- Limited number of resources
- Competition results in struggle for existence
- Success depends on fitness --
  - fitness of an individual:
    - how well-adapted an individual is to their environment.
    - This is determined by their genes (**blueprints** for their physical and other characteristics).
- Successful individuals are able to **reproduce and pass on** their genes

When changes occur …

- Previously “fit” (well-adapted) individuals will **no longer be best-suited** for their environment
- Some members of the population will have genes that confer different characteristics than “the norm”.
  - Some of these characteristics can make them more “fit” in the changing environment.

Major Agents of *Genetic Change* in Individuals

- **Mutation** in genes
  - may be due to various sources:
    - e.g. ultra-violet (UV rays),
    - chemicals, etc.

Start:

1001001001001001001001

After Mutation:

1001000001001001001001

Location of Mutation

Major Agents of Genetic Change in Individuals (2)

- Recombination (Crossing-Over)
  - occurs during reproduction -- sections of genetic material exchanged between two chromosomes

Recombination (Crossing-Over)

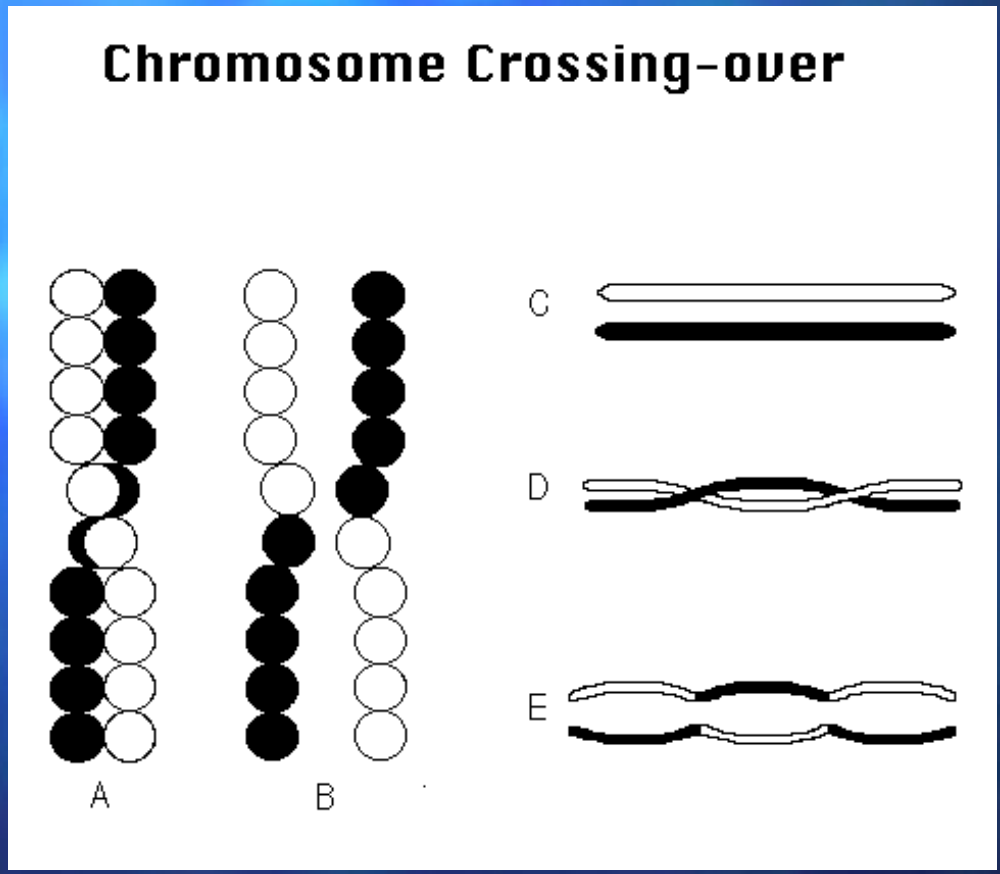


Image from http://esg-www.mit.edu:8001/bio/mg/meiosis.html

Example of Natural Selection and Micro-Evolution

- Peppered moth during Industrial Revolution (Manchester, England, 1848)
- Pre-Industrial Revolution -->

Image copyright Lawrence M. Cook (http://www.trueorigin.org/pepmoth1.htm)

Example of Natural Selection and Micro-Evolution (2)

- During Industrial Revolution -->

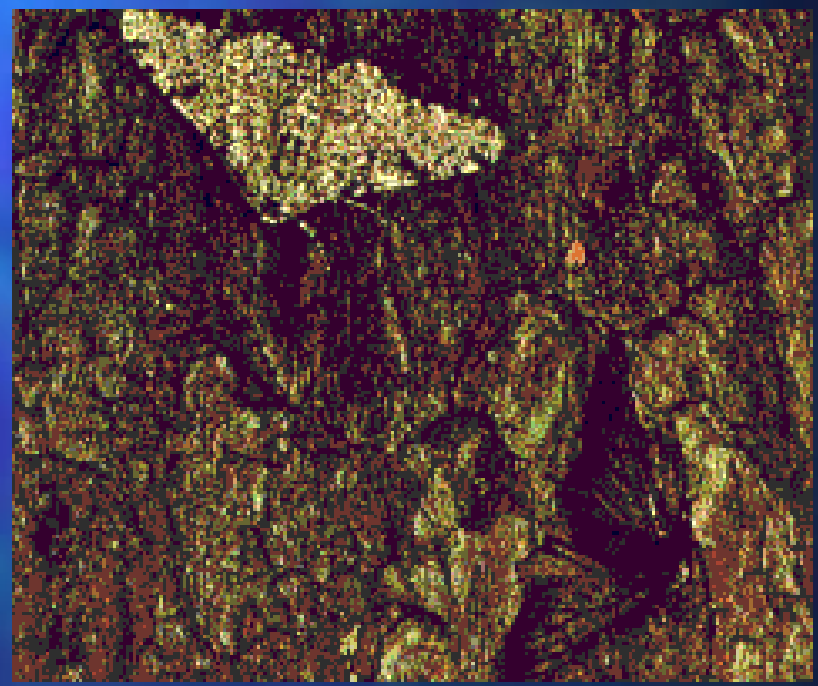


Image copyright Lawrence M. Cook (http://www.trueorigin.org/pepmoth1.htm)

Why use evolution as a model for solving computational problems?

The Nature of Computational Problems

- Require **search** through many possibilities to find a solution
  - (e.g. search through sets of rules for one set *that best predicts* the **ups and downs** of the financial markets)
  - **Search space too big** -- search won't return within our lifetimes
  - These types of problems are better solved using a **parallel approach**

The Nature of Computational Problems (2)

- Require algorithm to be adaptive
  - or to construct original solution
    - for instance, interfaces that must adapt to idiosyncrasies of different users

Why Evolution Proves to be a Good Model for Solving these Types of Problems?

- Evolution is in effect a method of searching for the best (optimal) solution from a great number of possibilities
  - Possibilities -- all individuals
  - Best solution -- the most “fit” or well-adapted individual
- Evolution is a parallel process
  - Testing and changing of *numerous species and individuals* occur at the same time (or, in parallel)

Why Evolution Proves to be a Good Model for Solving these Types of Problems?? (2)

- Evolution can be seen as a method that designs new (original) solutions to a changing environment

When to Use Evolutionary Computing Strategies?

- When space to be searched is large
- When the “best” solution is not necessarily required
- Approach to solving a problem **not well-understood**
- Problems with **many parameters** that need to be simultaneously optimized
- Problems that are difficult to describe mathematically

Genetic Algorithms

- **Closely follows a biological approach to problem solving**:
  - A simulated population of randomly selected individuals is generated
  - then allowed to evolve

Encoding the Problem

- Express the problem in terms of a **bit string**

x = (1001010101011100)

where the first 8 bits of the string represent the X-coordinate and the second 8 bits represent the Y-coordinate

Basic Genetic Algorithm

- **Step 1.** Generate a random population of *n* chromosomes
- **Step 2.** Assign a fitness to each individual
- **Step 3.** Repeat until *n* children have been produced
  - Choose 2 parents based on fitness proportional selection
  - Apply genetic operators to copies of the parents
  - Produce new chromosomes

Fitness Function

- For each individual in the population, evaluate its relative fitness
- For a problem with ***m*** parameters, the fitness can be plotted in an ***m*+1** dimensional space

Sample Search Space

- A randomly generated population of individuals will be randomly distributed throughout the search space

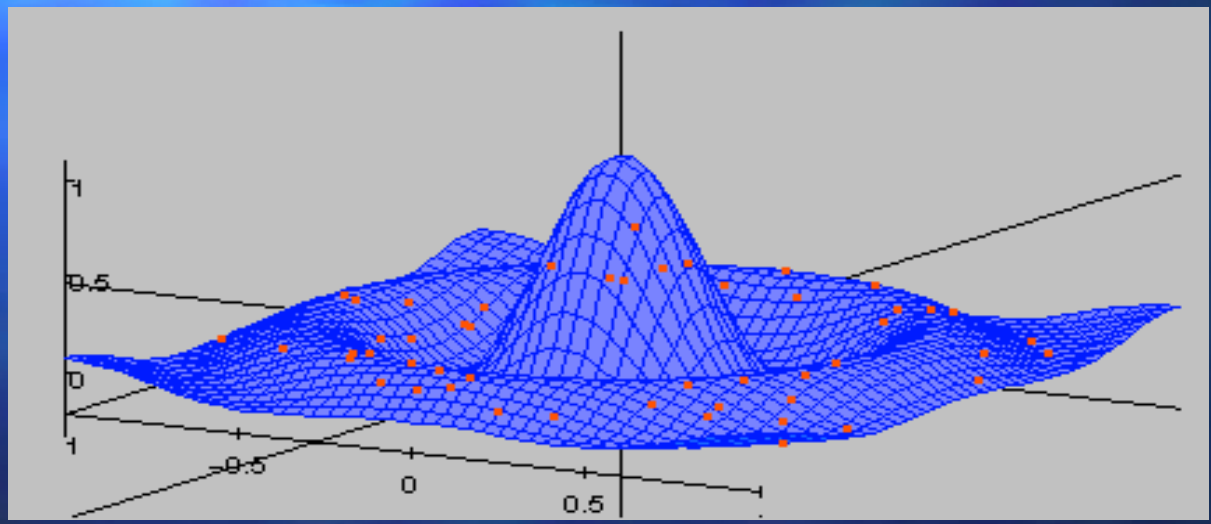


Image from http://www2.informatik.uni-erlangen.de/~jacob/Evolvica/Java/MultiModalSearch/rats.017/Surface.gif

Natural Selection

- The likelihood of any individual becoming a parent is directly *proportional* to its relative fitness

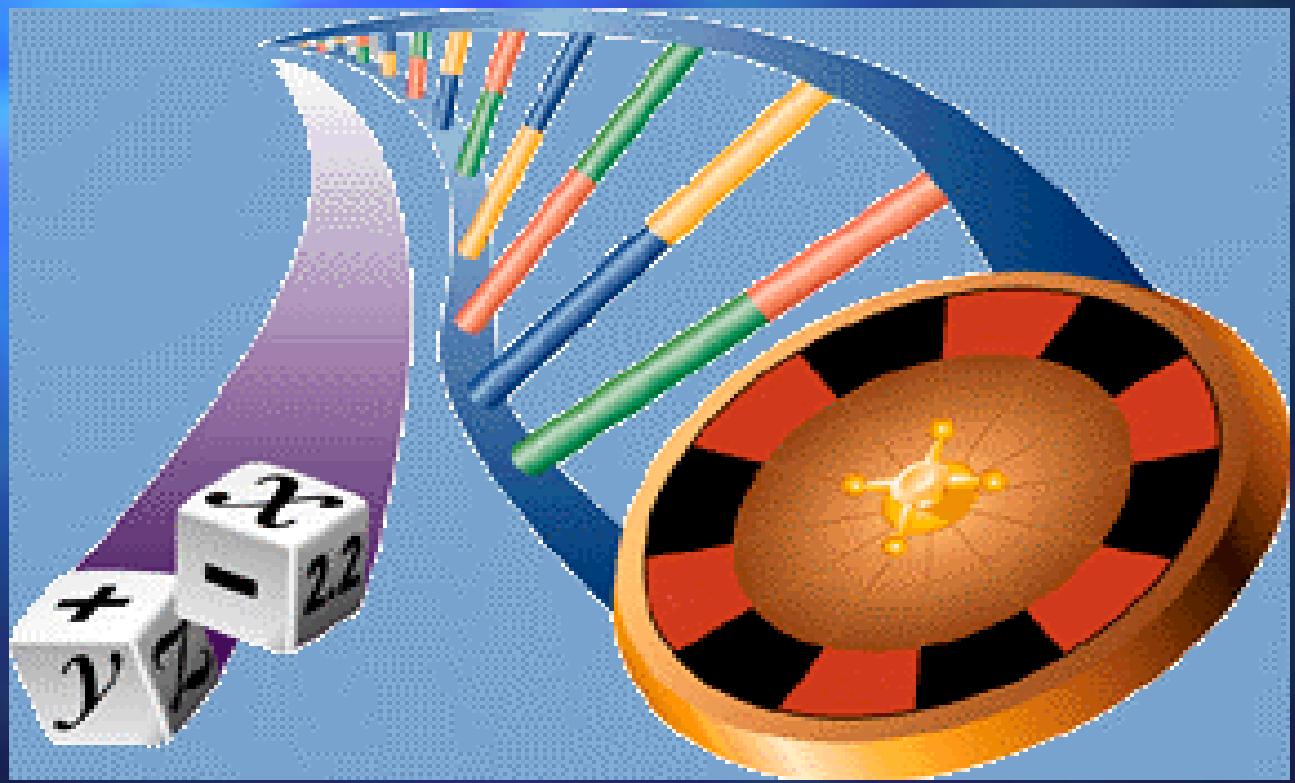
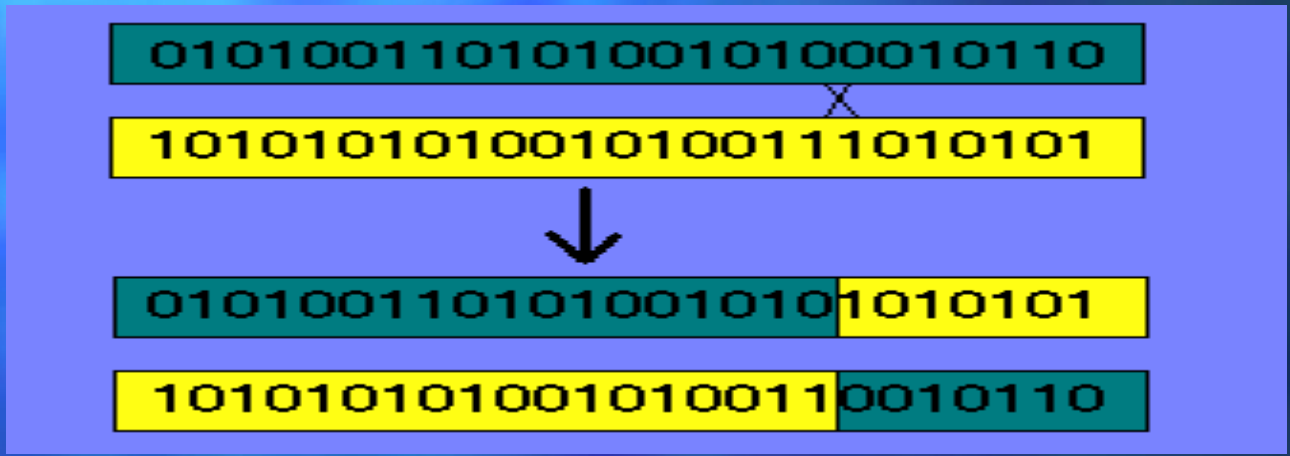


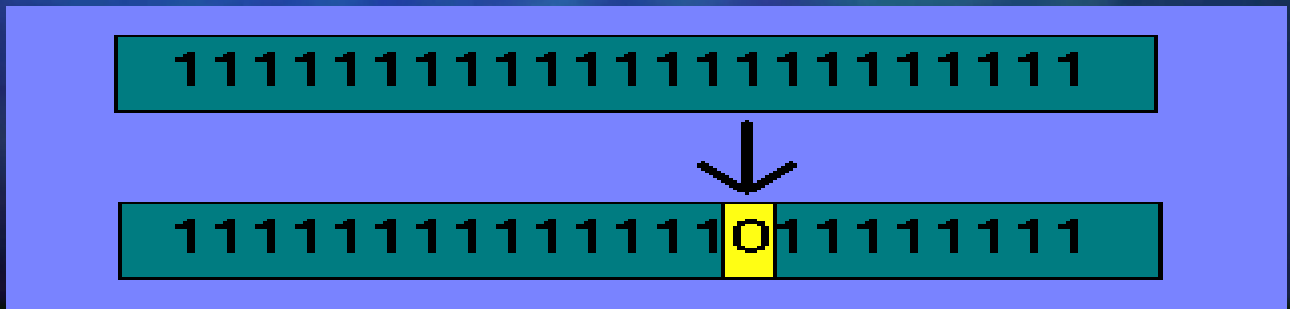
Image from http://www.genetic-programming.org

Genetic Operators

- Cross-over

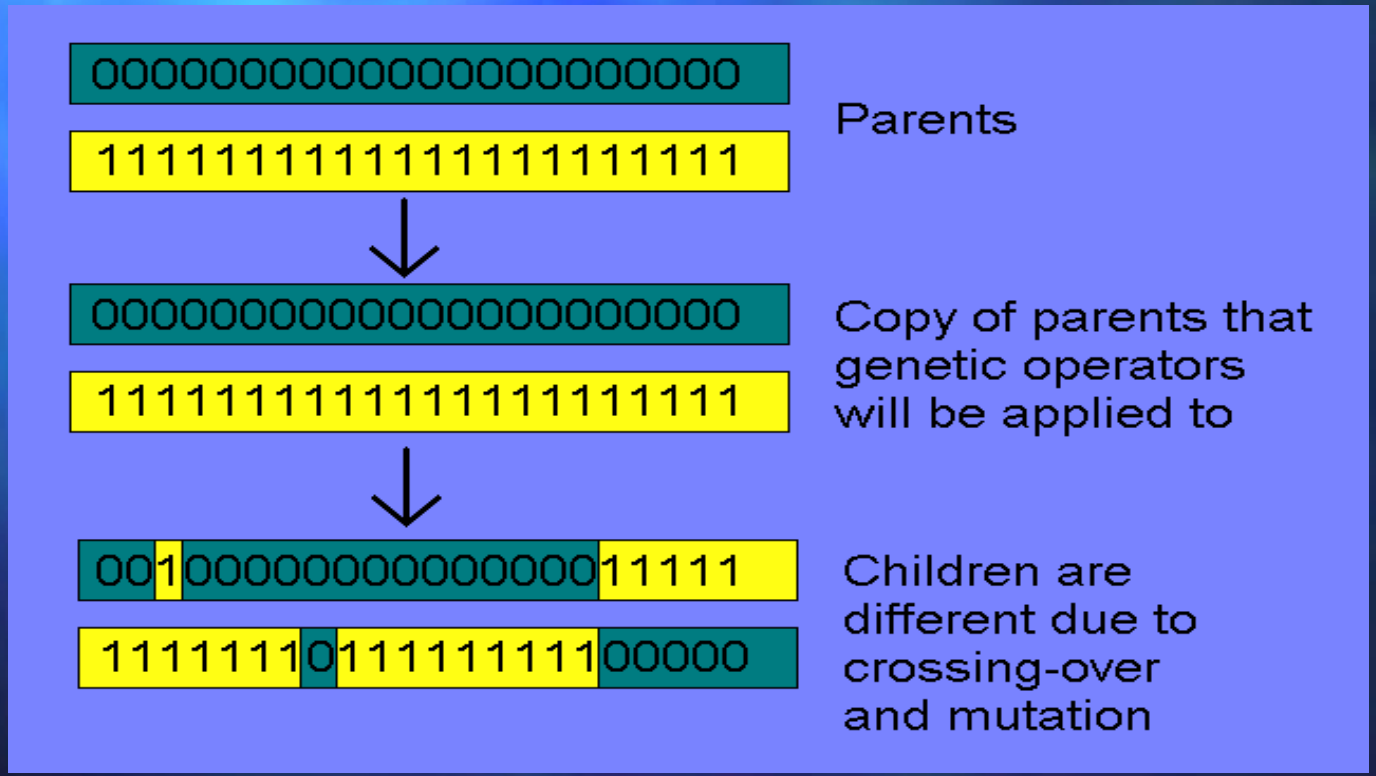


- Mutation



Production of New Chromosomes

- 2 parents give rise to 2 children



Generations

- As each new generation of ***n*** individuals is generated, they replace their parent generation

- To achieve the desired results, **500** to **5000** generations are required

Ultimate Goal

- Each subsequent generation will evolve toward the global maximum
- After sufficient generations a **near optimal solution** will be present in the population of chromosomes

Example

- Flywheel design

QuickTime™ and a
Video decompressor
are needed to see this picture.

Dynamic Evolution

- Genetic algorithms can adapt to a dynamically changing search space
- Seek out the moving maximum via a parasitic fitness function
  - as the chromosomes adapt to the search space, ***so does*** the fitness function

Example

- “Moving Optimum”

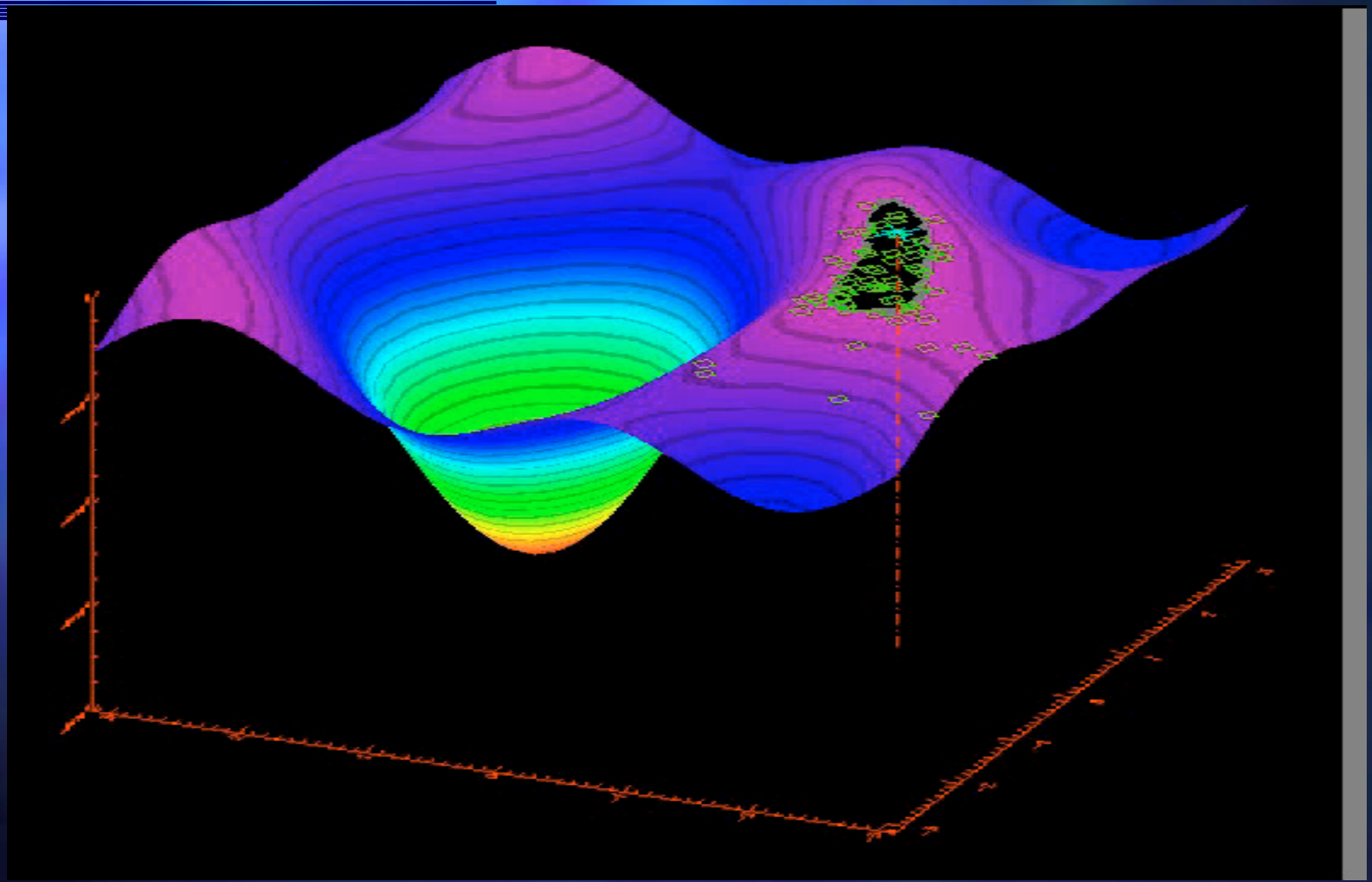


Image from http://www.wi.leidenuniv.nl/~gusz/ Flying_Circus/

another approach

Evolution Strategies

- Similar to Genetic Algorithms
  - find a (near-)optimal solution to a problem within a search space (all possible solutions)
- Developed by Ingo Rechenberg, independently from genetic algorithms
- Often used for empirical experiments
- Based on principal of strong causality: Small changes have small effects

Basic Evolution Strategy

1. **Generate** some random individuals
2. Select the ***p*** best individuals based on some selection algorithm (fitness function)
3. Use these ***p*** individuals to generate ***c*** children
4. Go to step 2, until the desired result is achieved
   - for instance, little difference between generations.

Encoding

- Individuals are encoded as vectors of real numbers (object parameters)
  - $\boldsymbol{op} = (o_1, o_2, o_3, \ldots, o_m)$
- The **strategy parameters** control the mutation of the object parameters
  - $\boldsymbol{sp} = (s_1, s_2, s_3, \ldots, s_m)$
- These two parameters constitute the individual's chromosome

Fitness Functions

- We need a method to determine if one solution is more optimal than another
- Mathematical formula
- Main difference from genetic algorithms is that *only the most fit individuals are allowed to reproduce* (**elitist selection**)

Forming the Next Generation

- Number of individuals selected to be parents (***p***)
  - **too many:** lots of persistent bad traits
  - **too few:** stagnant gene pool
- Total number of children produced (***c***)
  - limited by computer resources
  - more children $\Rightarrow$ faster evolution

Forming the Next Generation

- Similar operators as genetic algorithms
  - ***mutation*** is the most important operator (to uphold the principal of strong causality)
  - ***recombination*** needs to be used in cases where each child has multiple parents
- The parents can be included in the next generation
  - smoother fitness curve

Mutation

- Needed to ***add new genes*** to the pool:
  - optimal solution cannot be reached if a necessary gene is not present
  - bad genes filtered out by evolution
- Random changes to the chromosome:
  - **object** parameter mutation
  - **strategy** parameter mutation
    - changes the step size used in object parameter mutation

Object Parameter Mutation

- affected by the strategy parameters (another vector of real numbers)

op = (8, 12, 31, … ,5)

object parameter

$op_{mutated}$ = (8.2, 11.9, 31.3, …, 5.7)

sp = (.1, .3, .2, …, .5)

strategy parameter

Discrete Recombination

- Similar to crossover of genetic algorithms
- Equal probability of receiving each parameter *from each parent*

(8, 12, 31, … ,5) (2, 5, 23, … , 14)

(2, 12, 31, … , 14)

Intermediate Recombination

- Often used to adapt the strategy parameters
- Each child parameter is the mean value of the corresponding parent parameters

(8, 12, 31, … ,5) (2, 5, 23, … , 14)

(5, 8.5, 27, … , 9.5)

Evolution Process

- *p* parents produce *c* children in each generation
- Four types of processes:
  - *p,c*
  - *p/r,c*
  - *p+c*
  - *p/r+c*

p,c

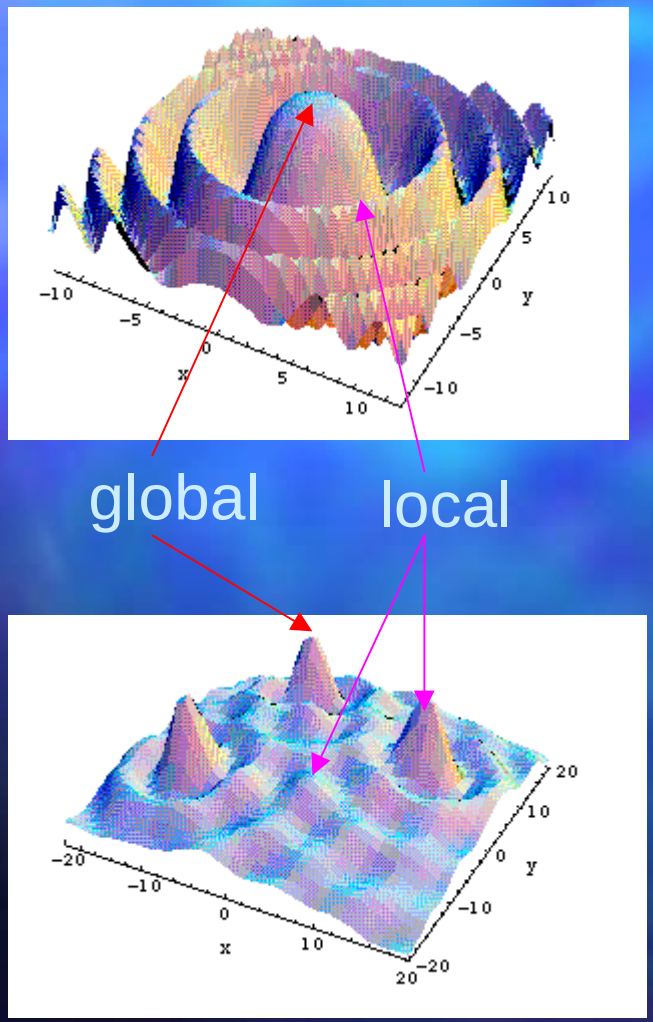
- ***p* parents produce *c* children using mutation only (no recombination)**
- The fittest ***p*** children become the parents for the next generation
- Parents are not part of the next generation
- $c \geq p$
- *p/r,c* is the above with recombination

p+c

- *p* parents produce *c* children using mutation only (no recombination)
- The fittest *p* individuals (parents or children) become the parents of the next generation

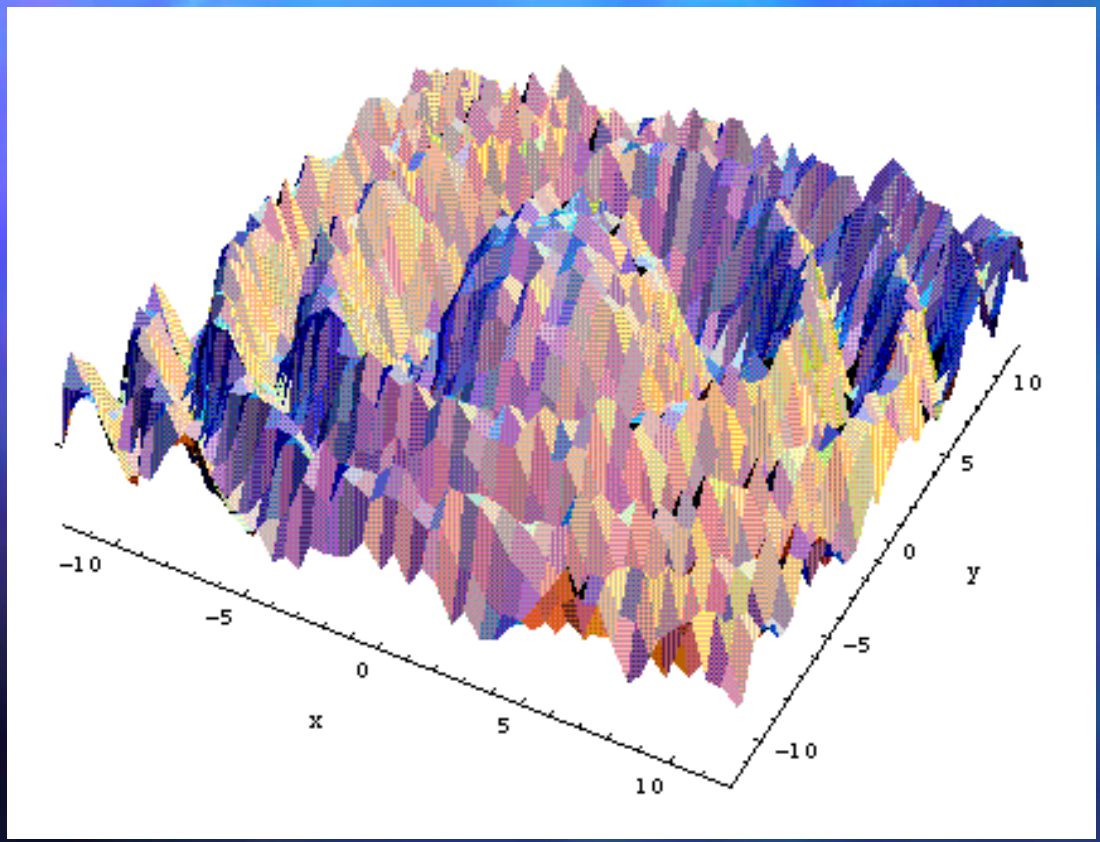
- ***p/r+c*** is the above with recombination

Example - Hill Climbing



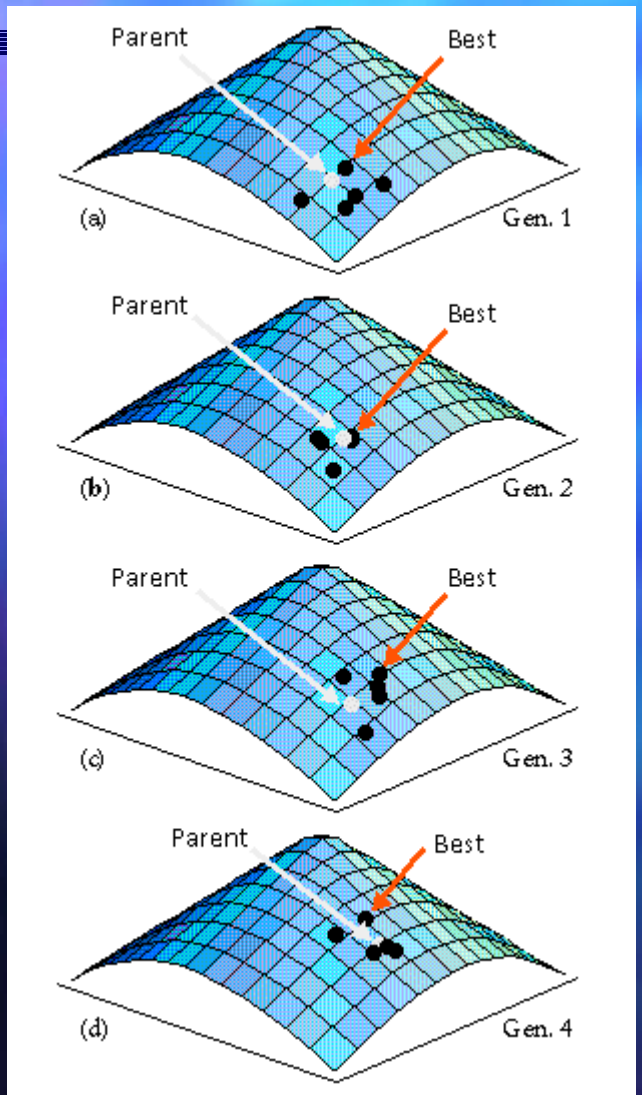
- 2-dimensional search space
- 3rd dimension is the fitness
- **Goal:** find the global maximum (most fit solution in the space)
- Avoid local maxima

Rugged Terrain



- More of a challenge to optimize
  - easy to get stuck in the many local maxima
- Need to adjust mutation and crossover rates

Climbing Strategy



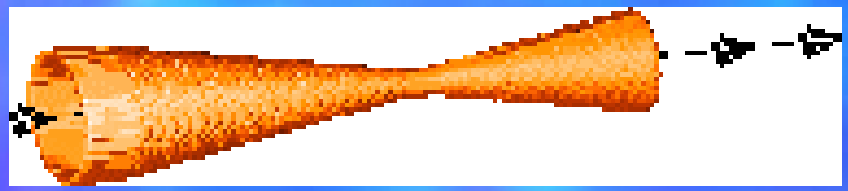
- 1,5 - ES

QuickTime™ and a
Animation decompressor
are needed to see this picture.

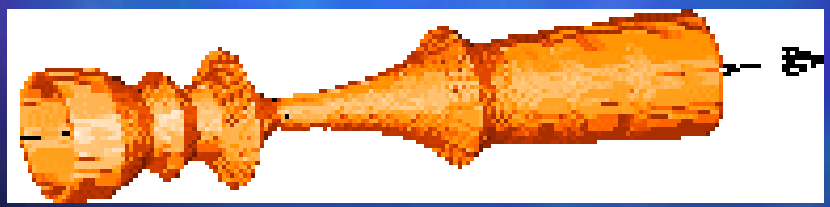
- Hill climbing in action…

Image from http://www2.informatik.uni-erlangen.de/IMMD-II/Lehre/SS99/NaturAlgorithmen/Presentations/05.EvolutionStrategies/Presentation/mutation.html

Evolution of a Two-Phase Jet Nozzle



Starting Nozzle (55%)



Optimized Nozzle (80%)

- Goal: obtain maximum thrust
- Nozzle represented by a series of conical segments
- Segments replaced by rules of 1+1-ES
- Unexpected outcome

Images from http://www.wi.leidenuniv.nl/~gusz/Flying_Circus/3.Demos/Movies/Duese/index.html

Evolution of a Two-Phase Jet Nozzle

- Each frame represents one generation

QuickTime™ and a
GIF decompressor
are needed to see this picture.

image from
http://www.wi.leidenuniv.nl/~gusz/Flying_Circus/3.Demos/Movies/Duese/index.html

Applications of Evolution Strategy

- Configuration of ventilation systems
- Piping systems
- Wing profiles and fuselage configurations
- Optical cable system layout

Drawbacks of GA's and ES's

- Difficult to find an **encoding** for the problem
- Difficult to define a valid fitness function
- May not return the global maximum

When to Use GA's vs. ES's

Genetic Algorithms

- More important to find optimal solution (GA's more likely to find global maximum; usually slower)
- Problem parameters can be represented as bit strings (computational problems)

Evolution Strategies

- "Good enough" solution acceptable (ES's usually faster; can readily find local maximum)
- Problem parameters are real numbers (engineering problems)

Nothing tends so much to the advancement of knowledge as the application of a new instrument.

The native intellectual powers of men in different times are not so much the causes of the different success of their labours, as the peculiar nature of the means and artificial resources in their possession.

- Sir Humphrey Davy

quoted from Thomas Hager, *Force of Nature,* Simon and Schuster, New York, 1995, p 86.

Sources

Julie Leung

Keith Kern

Jeremy Dawson