Evolutionary Algorithms

Part 2

Sources

Martijn Schut, schut@cs.vu.nl,

Jaap Hofstede, Beasly, Bull, Martin

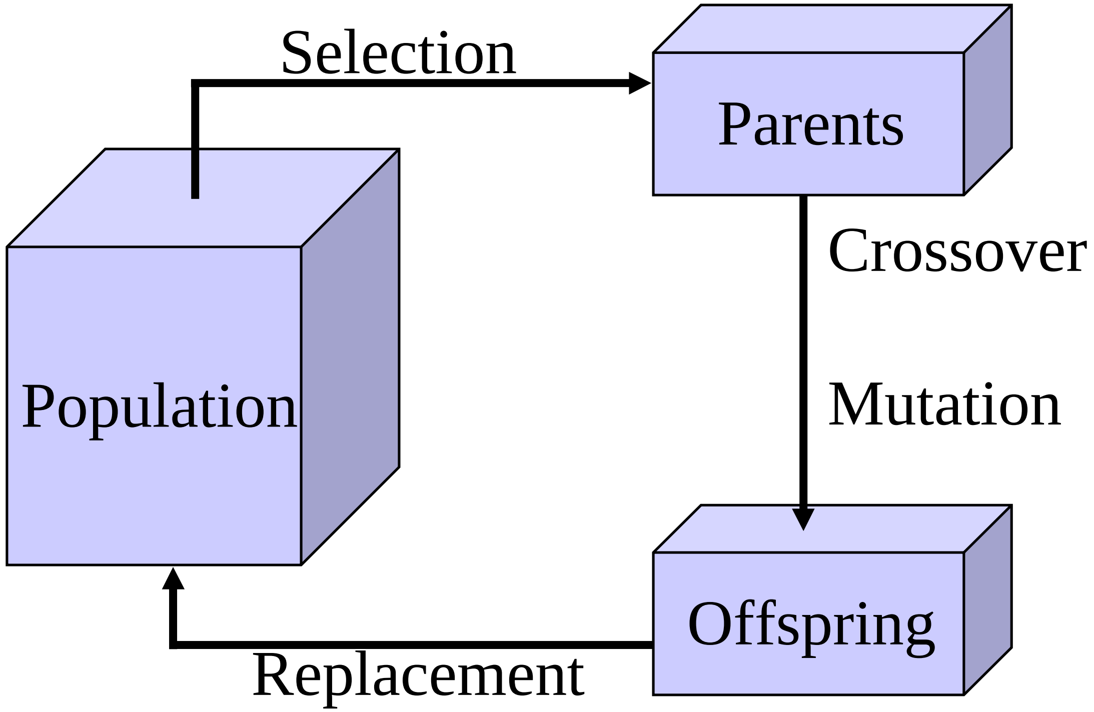
Andrea G. B. Tettamanzi

Joost N. Kok and Richard Spillman

KMT Software, Inc. -- http://www.kmt.com

Evolutionary Algorithm Variants, Theory and Ideas

Review – EA Structure



- GA Structure
- GA Operators
- GA Applications
- GA Examples

Remember: EA is a philosophy of solving problems not a single method

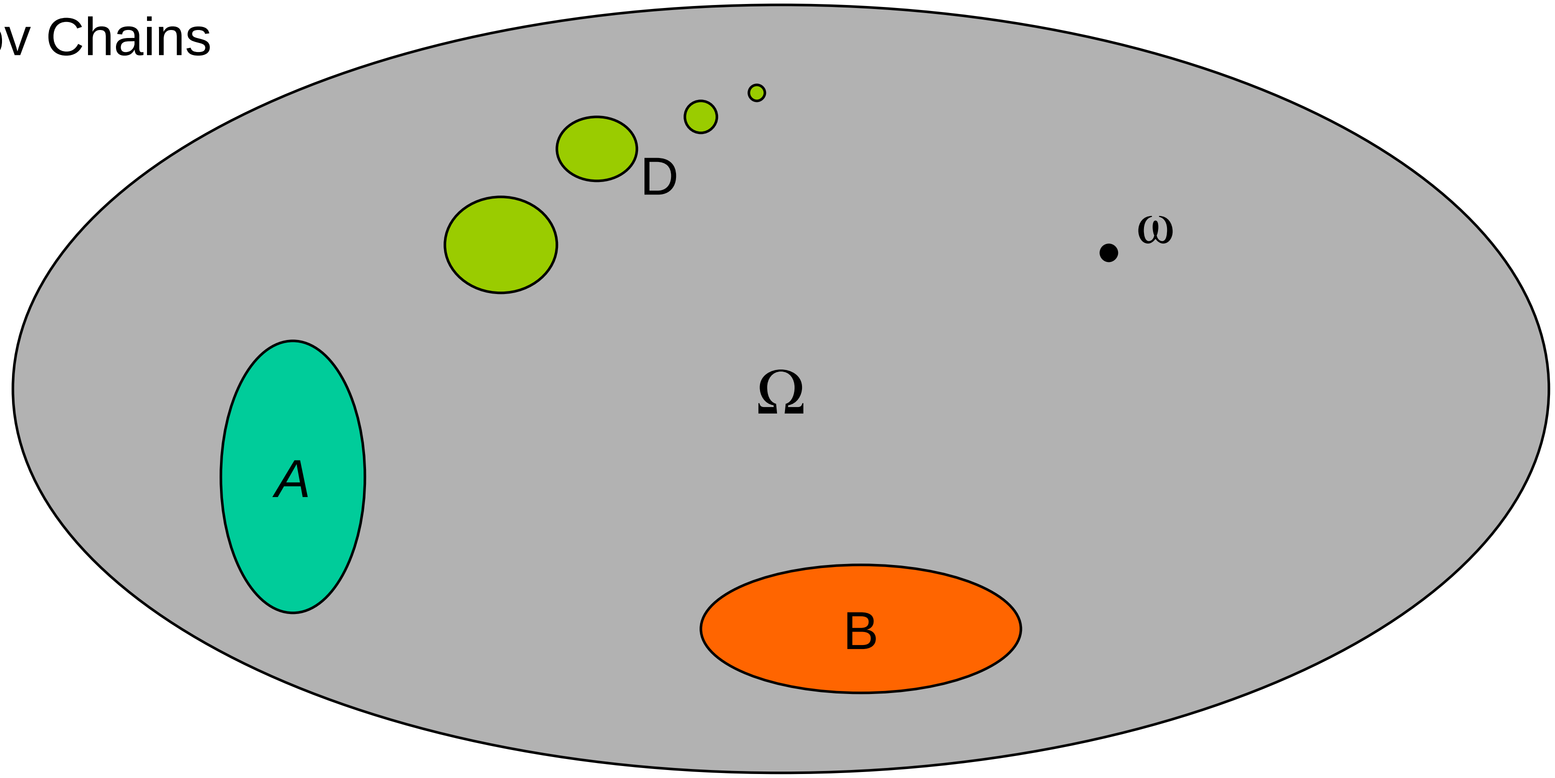
Alternative EA Structures

Theoretical Background

- Theory of random processes;
- Convergence in probability;
- Open question: rate of convergence.

Events and Stochastic Processes

Banach Fixpoint Theorem
Walsh Functions
Schema Theorem
Markov Chains



Markov Chains

Stochastic Process $\left\{ X_t(\omega) \right\}_{t=0,1,\ldots}$

Markov chain - depends on state. Probabilities. Inputs

$$P[X_t = x | X_0, X_1, \ldots, X_{t-1}] = P[X_t = x | X_{t-1}]$$

$$\begin{tikzcd}
A \arrow[loop left, "0.6"] \arrow[r, bend left, "0.4"] & B \arrow[l, bend left, "0.3"] \arrow[r, bend left, "0.7"] & C \arrow[loop right, "0.75"] \arrow[l, bend left, "0.25"]
\end{tikzcd}$$

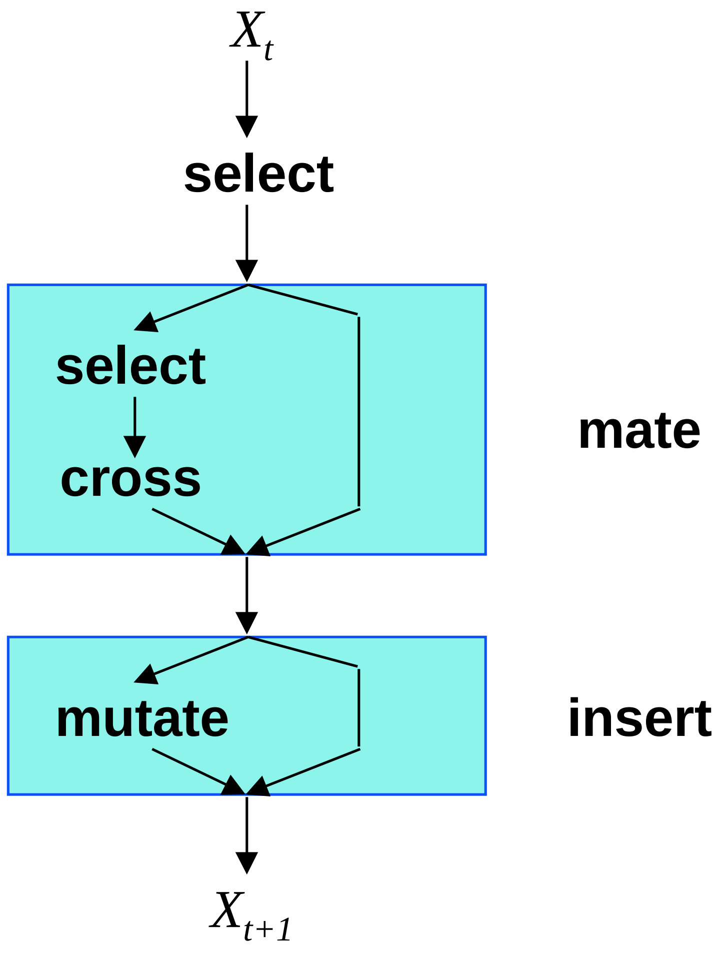
Abstract Evolutionary Algorithm

Stochastic functions:

select: $\Gamma^{(n)} \times \Omega \to \Gamma$
cross: $\Gamma \times \Gamma \times \Omega \to \Gamma$
mutate: $\Gamma \times \Omega \to \Gamma$
mate: $\Gamma \times \Gamma \times \Omega \to \Gamma$
insert: $\Gamma \times \Omega \to \Gamma$

Transition function:

$$X_{t+1}(\omega) = T_t(\omega)\left(X_t(\omega)\right)$$



Convergence to Optimum

Theorem: if $\{X_t(\omega)\}_{t = 0, 1, \ldots}$ is *monotone, homogeneous*, x_0 is given, $\forall y$ in **reach**(x_0) $\exists \gamma \in \Gamma^{(n)}{}_O$ reachable, then

$$\lim_{t\to\infty} P[X_t \in \Gamma_O^{(n)} \mid X_0 = x_0] = 1.$$

Theorem: if **select**, **mutate** are *generous*, the neighborhood structure is *connective*, transition functions $T_t(\omega)$, $t = 0, 1, \ldots$ are i.i.d. and *elitist*, then

$$\lim_{t\to\infty} P[X_t \in \Gamma_O^{(n)}] = 1.$$

Introduction to GA Theory

- Reproduction, crossover, and mutation are all surprisingly simple, computationally trivial operations - *so, how do they lead to an effective search procedure*?
  - The answer is found in the concept that the strings in the population encode more than just a single solution
  - Substrings within each string contain notions of what is important to the solution

$$1\ 0\ 1\ 1\ 0\ 0\ 0\ \underline{0\ 1\ 1\ 0\ 1\ 0}\ 1\ 0\ 0\ 1\ 0\ 0\ 0$$

Substring - maybe all the best answers have this substring

Schemata

- Individual strings in the population are not the focus
  - **similarities** between highly fit strings guide the search
- DEFINITION: A **schema** is a **similarity template** describing a subset of strings with similarities at certain string positions
  - EXAMPLE: What is the schema for these strings?

```
1 0 1 0 1 0 1 0
1 1 0 0 1 0 0 1   ────────→   * * * 0 1 * * *
0 0 1 0 1 1 0 0
1 1 0 0 1 0 1 1
```

Schemata Structure

- The schemata symbols are 0, 1 and *
- A schema matches a given string if at every location in the schema:
  - a 1 matches a 1 in the string
  - a 0 matches a 0 in the string
  - a * matches either 0 or 1
- For example
  - *0000 matches (00000, 10000)

Schemata

Don't care symbol: $*$

Example of schema

*	*	*	1	0	*	1	*	*	*

O(s)=3

order of a schema: $o(S)$ = # fixed positions

defining length $\delta(S)$ = distance between first and last fixed position

- a schema S matches $2^{l - o(S)}$ strings
- a <u>string of length l</u> is matched by 2^l schemata

Counting Schemata

- If the string is of length *v* then there are 3^v different similarity templates because each of the *v* positions could be 0, 1 or *

- If the alphabet has k characters (k=2 for binary strings) then the number of similarity templates is **$(k+1)^v$**

- A population of n strings of length *l* contains **2^l** to **$n2^l$** schemata depending upon the population diversity

- **RESULT:** even a small population contains a large number of important similarities

Implicit Parallelism of EA

- In a population of *n* individuals of length *l*
- $2^l \leq$ **# schemata processed** $\leq n2^l$
- n^3 of which are <u>processed usefully</u> (Holland 1989)
  - (i.e. are not disrupted by crossover and mutation)

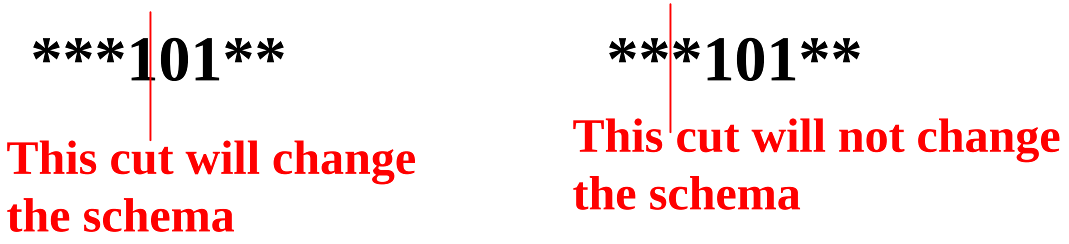
But see Bertoni & Dorigo (1993)

"Implicit Parallelism in Genetic Algorithms"

Artificial Intelligence ***61****(2), p. 307–314*

Reproduction / Crossover

- Observe that highly fit strings have higher probabilities of being selected
- Thus, on the average, more samples in the next population will have the best similarity patterns
- Crossover leaves ***some schema unchanged*** **while it disrupts others**
  - a schema is unchanged if it is not cut by the crossover
  - a schema is changed if it is cut by the crossover point


***101**
This cut will change the schema
***101**
This cut will not change the schema

So how schema affect solutions?

- Some schema are more likely to be disrupted than others
- For example, 1***0 is more likely to be disrupted by crossover than **11*
- Therefore, schemata that have a short defining length tend to be left alone by the crossover
  - thus they are *reproduced at a higher sampling rate*

CONCLUSION: Highly fit, short-defining length schemata (called *building blocks*) are propagated generation to generation by giving exponentially increasing samples to the observed best.

… let us be more formal….

Fitness of a schema

***f*(γ)**: fitness of string γ

q*$_x$(γ)**: fraction of strings equal to γ in population ***x

q*$_x$(S)**: fraction of strings matched by **S** in population ***x

$$f_x(S) = \frac{1}{q_x(S)} \sum_{\gamma \in S} q_x(\gamma) f(\gamma)$$

Fitness of schema S

The Schema Theorem

$\{X_t\}_{t=0,1,\ldots}$ populations at times *t*

suppose that $\dfrac{f_{X_t}(S) - f(X_t)}{f(X_t)} = c$ is constant

$$E[q_{X_t}(S) \mid X_0] \geq q_{X_0}(S)(1+c)^t \left(1 - p_{cross}\frac{\delta(S)}{l-1} - o(S)\,p_{mut}\right)^t$$

i.e. above-average individuals increase exponentially!

The Schema Theorem (proof)

$$E[q_{X_t}(S)|X_{t-1}] \geq q_{X_{t-1}}(S)\frac{f_{X_{t-1}}(S)}{f(X_{t-1})}P_{surv}[S] = q_{X_{t-1}}(S)(1+c)P_{surv}[S]$$

$$P_{surv}[S] = 1 - p_{cross}\frac{\delta(S)}{1-l} - p_{mut}o(S)$$

The Building Blocks Hypothesis

“An evolutionary algorithm seeks near-optimal performance through the juxtaposition of short, low-order, high-performance schemata — the building blocks”

Deception

i.e. when the building block hypothesis does not hold:

for some schema *S*, $\gamma^* \in S$ but $f(S) < f(\bar{S})$

Example:

S_1 = 111*******

S_2 = ********11

γ^* = 1111111111

S = 111*****11

$\bar{S}$ = 000*****00

Remedies to deception

$$\begin{tikzcd}
\text{Prior knowledge of the objective function} \arrow[dr] & \\
& \text{Non-deceptive encoding}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Inversion} \arrow[dr] & \\
& \text{Semantics of genes not positional}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Underspecification \& overspecification} \arrow[dr] & \\
& \text{“Messy Genetic Algorithms”}
\end{tikzcd}$$

Alternative Selection Methods

- The process of selecting the chromosomes to contribute to the next generation has a significant impact on the performance of a GA

- Rather than use a roulette wheel approach, there are many other selection methods
  - Tournament Selection
  - Greedy Overselection
  - . . .

REVIEW: Tournament Selection

- With tournament selection a specified group of individuals (usually 2) are chosen at random from the population
  - the one with the better fitness is selected to be the parent
- If more than 2 are used, the process is to select the best from the set

Greedy Overselection

- **CONCEPT**: Greedily over-select the fitter individuals in a population

- **PROCESS**: Create a high fitness group, H, and a low fitness group, L
  - 80% of the time select the parent from group H
  - 20% of the time select the parent from group L

Steps of Greedy Overselection

- **Step 1**: Sort the chromosomes in decreasing order of fitness
- **Step 2**: Place the top 32% of the chromosomes in group H, the rest in group L
- **Step 3**: 80% of the time select a parent from group H and 20% of the time select a parent from group L

NOTE: Also use greedy overselection to create the initial population

Alternative Crossover Methods

- Rather than use single point crossover there are several other methods:
  - Two Point Crossover (already discussed)
  - Constrained Crossover
  - Shuffle Crossover
  - Uniform Crossover
  - Selective Crossover
  - Arithmetical Crossover
  - . . .

Constrained Crossover

- **GOAL:** Select a crossover point that *always produces variations*
- **PROCESS**: Constrain crossover points to occur within non-matching alleles
- **EXAMPLE:**

```
0 | 1 0 1 1 | 0 1 0 0
0 | 0 1 0 0 | 0 1 0 0
```

Find a non-matching segment and select a random point within the segment

Shuffle Crossover

- Three Step Process
  - randomly shuffle the bit positions of the two strings in tandem
  - cross the strings using any crossover process
  - unshuffle the strings
- EXAMPLE

```
START:   0 1 0 1 0 1 0 1 0 1
         1 1 1 1 0 0 0 0 1 1
SHUFFLE: (1 to 3, 3 to 6, 6 to 2, 2 to 9, 9 to 10, 10 to 1)
         1 1 0 1 0 0 0 1 1 0
         1 0 1 1 0 1 0 0 1 1
CROSSOVER (say random point 5) and UNSHUFFLE:
         0 1 1 1 0 1 0 0 1 1
         1 1 0 1 0 0 0 1 0 1
```

Arithmetical Crossover

- If the chromosomes are made up of floating point numbers instead of binary bits, then other types of crossover may be considered

- Arithmetical crossover is defined as a linear combination of two vectors
  - if v^t and w^t are to be crossed, the resulting offspring are:

    $$v^{t+1} = aw^t + (1-a)v^t$$
    $$w^{t+1} = av^t + (1-a)w^t$$

  - if the parameter a is a constant then this is called uniform arithmetical crossover

Important Issues / Ideas

- Selective Crossover – Is the increase in the dominance value for the higher fit child anything like an ant algorithm – laying down the pheromone trail?

Literature:

- "Selective Crossover in Genetic Algorithms" K. Vekaria and C. Clack

Selective Crossover

- Selective Crossover associates a real-valued vector with each chromosome such that each gene has an associated "dominance" value

- EXAMPLE

The dominance values are randomly selected at the start

Dominance	.1	.4	.2	.2	.8	.2	.1	.5
Chromosome	1	0	0	1	1	1	0	0

Selective Crossover Procedure

- During crossover the first child receives all dominant alleles and their dominance values.
  - The second child receives all the recessive alleles
  - The fitness of the children are compared to the parents.
    - When an increase occurs, *the dominance values in the fitter child* whose genes have changed from that of the fitter parent are increased.

Example of selective crossover

average

Parent 1: Fitness .36

.1	.4	.2	.2	.8	.2	.1	.5
1	0	0	1	1	1	0	0

Parent 2: Fitness .30

.3	.2	.5	.1	.3	.6	.2	.2
0	1	0	0	1	1	1	0

Child 1: Fitness .46

.3	.4	.5	.2	.8	.6	.2	.5
0	0	0	1	1	1	1	0

Child 2: Fitness .20

.1	.2	.2	.1	.3	.2	.1	.2
1	1	0	0	1	1	0	0

Increase these dominance values because they are better than in the fitter parent

Advantages of selective crossover

- Selective crossover is not biased against schema with high defining length

- For example, interaction genes at the two extremes of the chromosome can be propagated as easily as those located adjacent to each other.

Alternative Mutations

- Rather than complementing a random bit based on a low probability action:
  - Non Uniform Mutation
  - Inversion
  - . . .

Non-Uniform Mutation

- Standard mutation changes one bit of a string at a time - it is using only local knowledge (that is, knowledge of that single bit)
  - however, if the string encodes a number then we know
    - ✓ bits on the left side of the sequence represent significant values
    - ✓ bits on the right side of the sequence represent small values
  - Non-uniform mutation uses this global knowledge
    - ✓ as the population ages, the probability of mutation for bits on the right increases
    - ✓ as the same time, the probability of mutation on the left decreases

Inversion of bit order as mutation

- Rather than selecting a single bit to mutate, inversion finds two random points in the string and reverses the order of the bits between those points.

- EXAMPLE

```
1 0 0 1|0 0 1 1|1 0

1 0 0 1|1 1 0 0|1 0
```

Alternative GA structures

- There are several variations on the GA concept
  - CHC Algorithm(see below)
  - Breeder Genetic Algorithm
  - Various Hybrid Architectures
  - Evolution Algorithms
  - . . .

- These variations illustrate the use of alternative selection, crossover, and mutation schemes

CHC Adaptive Search Algorithm

- Fundamentally, **CHC** is a genetic algorithm but it differs from the standard GA in several ways
  - Unbiased parent selection - cross generational selection
  - HUX crossover (Half, Uniform X-over)
  - Maintains diversity

Cross Generational Selection

- Chromosomes are randomly selected from the parent population (high fitness is not favored)
  - Offspring are held in a temporary population
  - A survival competition is held where the best N chromosomes from the parent and the offspring populations are selected to form the next generation

Heterogeneous Recombination

- Incest prevention is used to guide the combination of parents
  - only chromosomes which differ from each other by some <u>fixed number of bits</u> are allowed to crossover
  - the threshold is initially L/4 where L is the length of a chromosome
  - when no offspring are created, then the <u>threshold is reduced by 1</u>

Diversity

- When no offspring can be inserted into the population and the threshold is 0, the population is reinitialized using cataclysmic mutation
  - the best chromosome is saved and placed in the new population
  - the remaining elements are formed by a random mutation of about 35% of the bits in the best element

HUX Crossover

- Uniform crossover over half the bits that differ between the two parents
  - find all the bit positions that differ between the two parents
  - randomly select a position and swap the bits
  - repeat this process until half of the differing bits have been swapped

GA Flexibility and your "Inventing New Game of Life Class Project"

- The real beauty of genetic algorithms is their adaptability
- You are free to try any reasonable approach to selection, crossover, and mutation
- This opens up a wide range of possible innovative class projects

Possible Other Project Topics

- Check my KAIST web site for Fall Quarter for a GA project in quantum computing
- Some topics that might be interesting:
  - Developing GA’s to break cipher systems
  - Developing GA’s to solve any of the class of NP-Complete problems
  - Developing GA’s to create neural networks or cellular automata (more on these later in the course) - Brain Building - check on my WebPage and Andrzej Buller’s WWW Page in ATR Japan.