Evolutionary Algorithms: Introduction

Jiří Kubalík
Department of Cybernetics, CTU Prague

http://cw.felk.cvut.cz/doku.php/courses/a4m33bia/start

Contents

4. **Real-coded EAs**
   - Evolution strategies,
   - Crossover operators for real-coded representation,
   - Differential evolution.
5. **EAs for dynamic optimization problems**
6. **Swarm Intelligence**
   - Ant Colony Optimization,
   - Particle Swarm Optimization.

Evolutionary Programming

Lawrence J. Fogel in 1960: Evolutionary programming (1960)

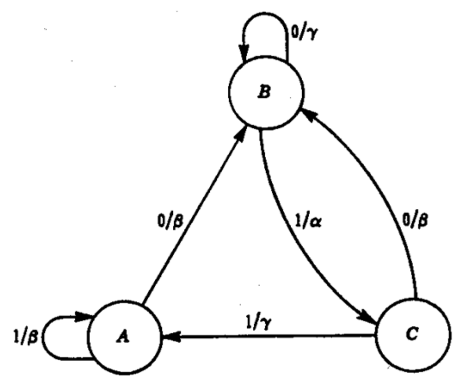
The goal is to evolve an "intelligent behavior" that would exhibit the ability to (1) predict one's environment, coupled with (2) a translation of the predictions into a suitable response in light of the given goal.

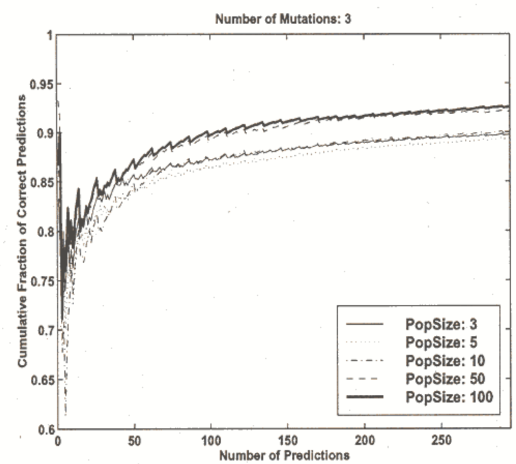
- the environment was described as a sequence of symbols taken from a finite alphabet,
- finite state machines (FSMs) were used for representing the required behavior.

Five modes of mutation

- add a state,
- delete a state,
- change the initial state,
- change an output symbol,
- change a next-state transition.

FSM





D. B. Fogel and K. Chellapilla: Revisiting Evolutionary Programming, Aerospace/Defense Sensing and controls.

Evolution Strategies: Optimization of a Two-Phase Nozzle

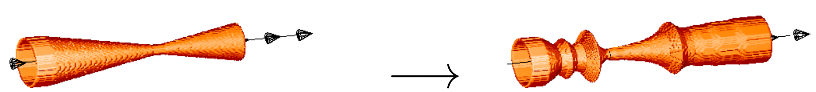
Ingo Rechenberg and Hans-Paul Schwefel: Evolution Strategy (early 1960s)

The task was to determine the internal shape of a two-phase jet nozzle with maximum thrust under constant starting conditions.

- The nozzles were built of conical pieces such that no discontinuity within the internal shape was possible.
- Every nozzle shape could be represented by its overall length and the inside diameters at the borders between the segments (every 10mm).

(1+1) Evolution Strategy using mutations of the following forms:

- Add new segment to the nozzle at positions chosen at random.
- Duplicate an existing segment.
- Delete a randomly chosen segment.
- Vary diameters of a randomly chosen segment.



http://ls11-www.cs.uni-dortmund.de/people/kursawe/Demos/Duese/dueseGIFE.html

Evolutionary Algorithms: Characteristics

:: **EA are stochastic optimization algorithms**

- **Stochastic** – but not random search,
- **Use an analogy of natural evolution**
  - genetic inheritance (J.G. Mendel) – the basic principles of transference of hereditary factors from parent to offspring – genes, which present hereditary factors, are lined up on chromosomes.
  - strife for survival (Ch. Darwin) – the fundamental principle of natural selection – is the process by which individual organisms with favorable traits are more likely to survive and reproduce.
- **Not fast in some sense** – population-based algorithm,
- **Robust** – efficient in finding good solutions in difficult searches.

EA: Vocabulary

:: Vocabulary borrowed from natural genetics

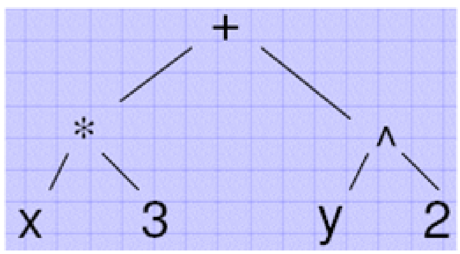
- **Individual** (chromosome + its quality measure "fitness value") – a solution to a problem.
- **Chromosome** – entire representation of the solution.
- **Fitness** – quality measure assigned to an individual, expresses how well it is adapted to the environment.
- **Gene** (also features, characters) – elementary units from which chromosomes are made.
  - each gene is located at certain place of the chromosome called locus (pl. loci),
  - a particular value for a locus is an allele.

    example: the "thickness" gene (which might be at locus 8) might be set to allele 2, meaning its second-thinnest value.
- **Genotype** – what's on the chromosome.
- **Phenotype** – what it means in the problem context (e.g., binary sequence may map to integers or reals, or order of execution, etc.).

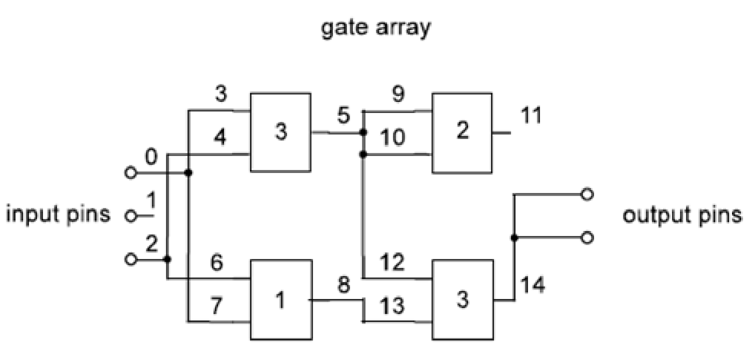
Representation

:: Problem can be represented as

- **binary string** – 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1
- **real-valued string** – 3,24 1,78 -2,61
- **string of chars** – D→E→A→C→B
- or as a **tree**



- or as a **graph**, and others.



Evaluation Function

:: **Objective (Fitness) function**

- the only information about the sought solution the algorithm dispose of,
- must be defined for every possible chromosome.

:: **Fitness function may be**

- multimodal,
- discrete,
- multidimensional,
- nonlinear,
- noisy,
- multiobjective.

:: **Fitness does not have to be define analytically**

- simulation results,
- classification success rate.

:: **Fitness function should not be too costly!!!**

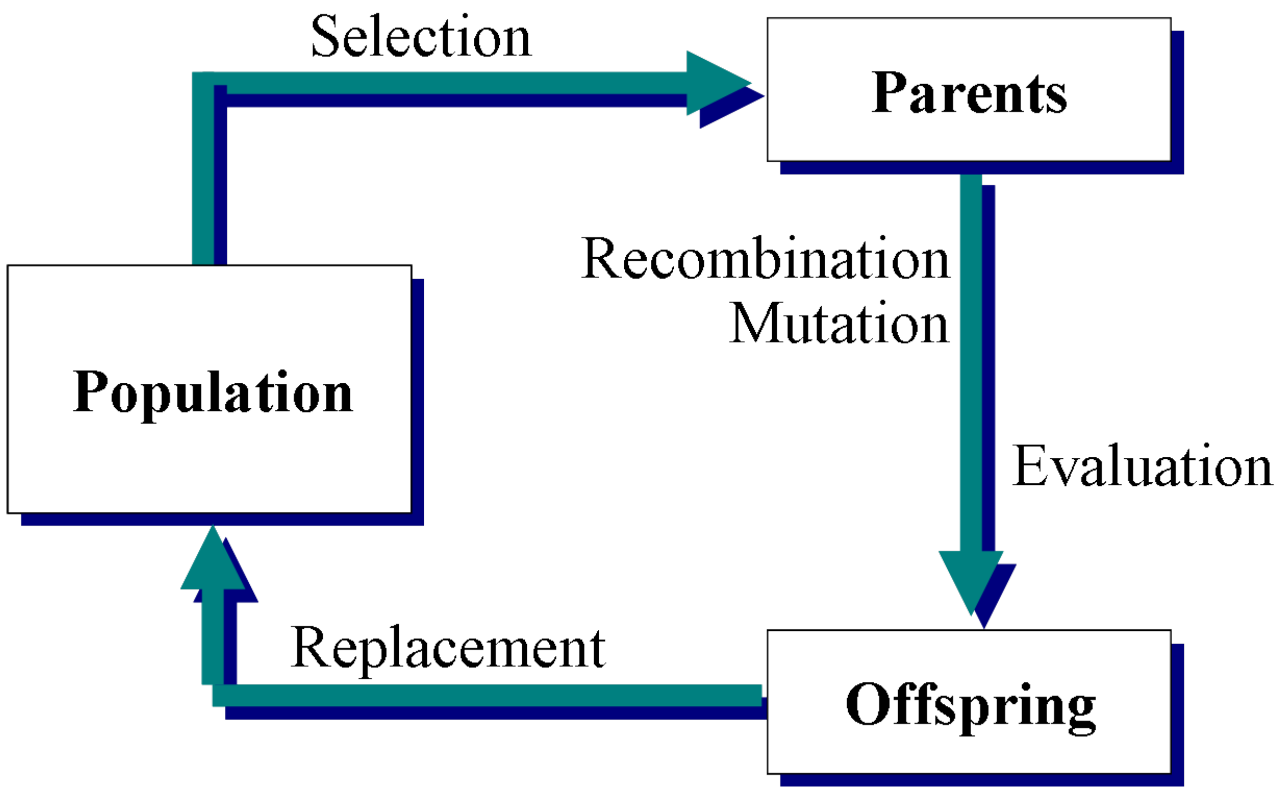
Example: Coding & Evaluation

:: **Function optimization**

- maximization of $f(x, y) = x^2 + y^2$,
- parameters x and y take on values from interval $< 0, 31 >$,
- and are code on 5 bits each.

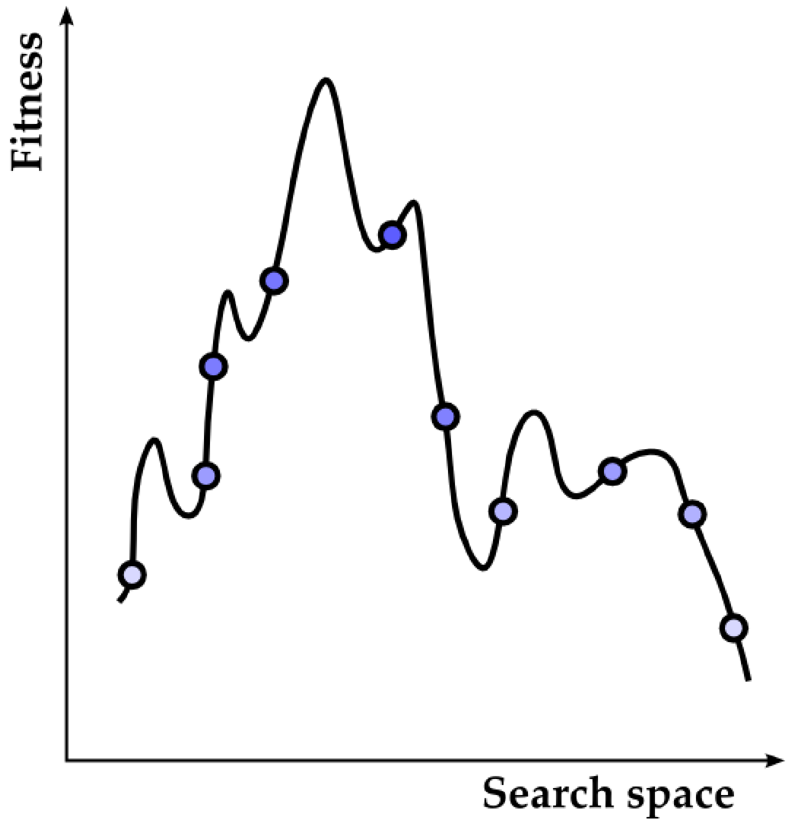
genotype	phenotype	fitness
0 0 0 0 0, 0 1 0 1 0	**0, 10**	**100**
0 0 0 0 1, 1 1 0 0 1	**1, 25**	**625 + 1 = 626**
0 1 0 1 1, 0 0 0 1 1	**11, 3**	**121 + 9 = 130**
1 1 0 1 1, 1 0 0 1 0	**27, 18**	**729 + 324 = 1053**

Evolutionary Cycle

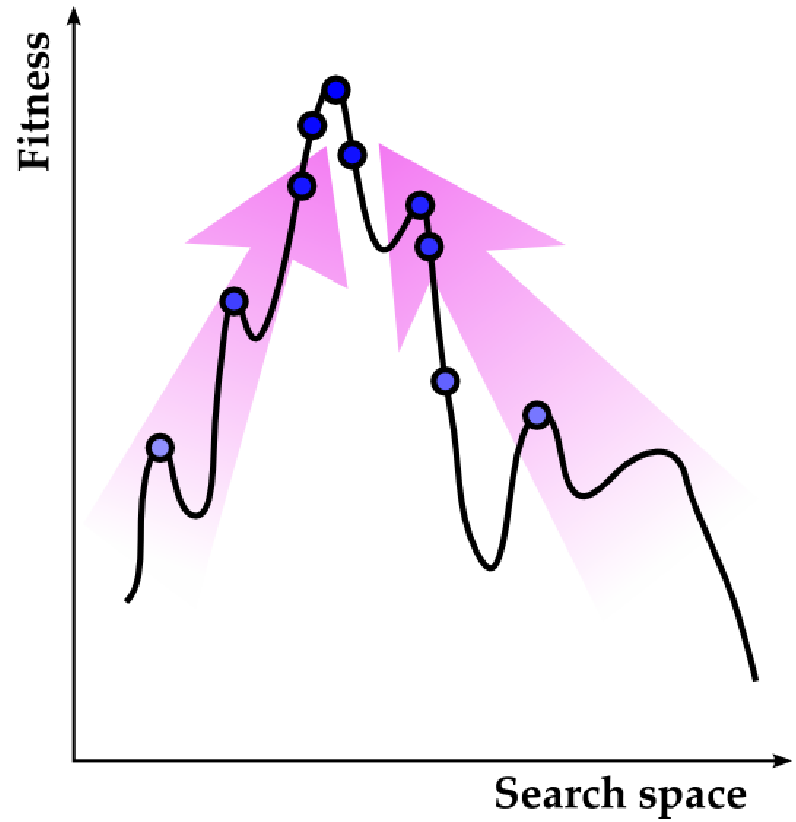


Idealized Illustration of Evolution

- Uniform sampled population.



- Population converged to promising regions.



Initialization

:: Random

- randomly generated solutions,
- no prior information about the shape of the sought solution,
- relies just on "lucky" sampling of the whole search space by a finite set of samples.

:: Informed (pre-processing)

- (meta)heuristic routines used for seeding the initial population,
- biased random generator sampling regions of the search space that are likely to contain the sought solutions,
  + may help to find better solutions,
  + may speed up the search process,
  – may cause irreversible focusing of the search process on regions with local optima.

Reproduction

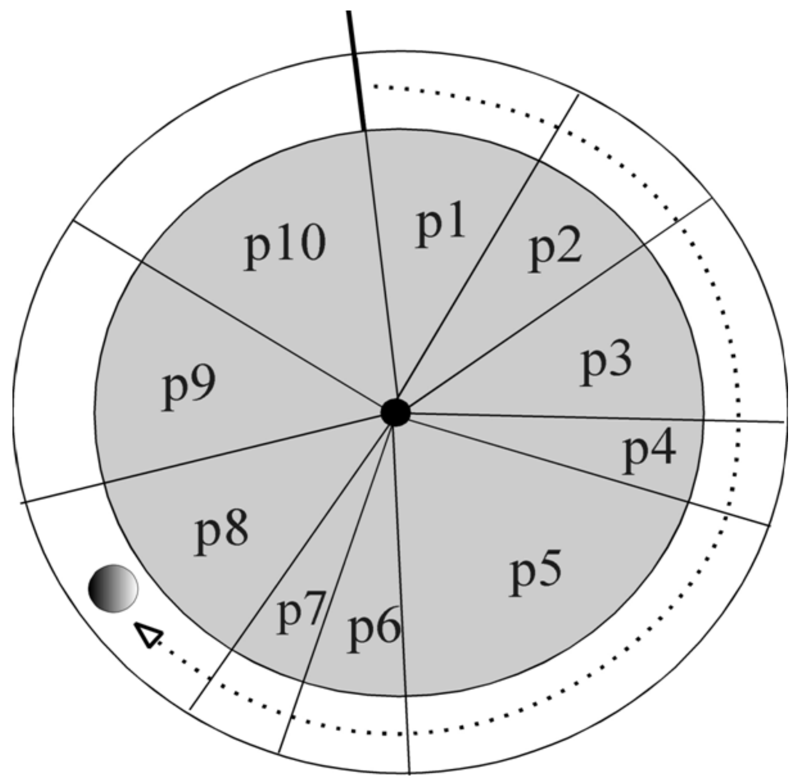
:: Models nature's survival-of-fittest principle

- prefers better individuals to the worse ones,
- still, every individual should have a chance to reproduce.

:: **Roulette wheel**

- probability of choosing some solution is directly proportional to its fitness value

$$P_i = \frac{f_i}{\sum_{j=1}^{PopSize} f_j}$$



:: Other methods

- Stochastic Universal Sampling,
- Tournament selection,
- Reminder Stochastic Sampling.

Reproduction: Premature Convergence & Stagnation

:: Two (strongly related) important issues in the evolution process

- **population diversity**,
- **selective pressure**.

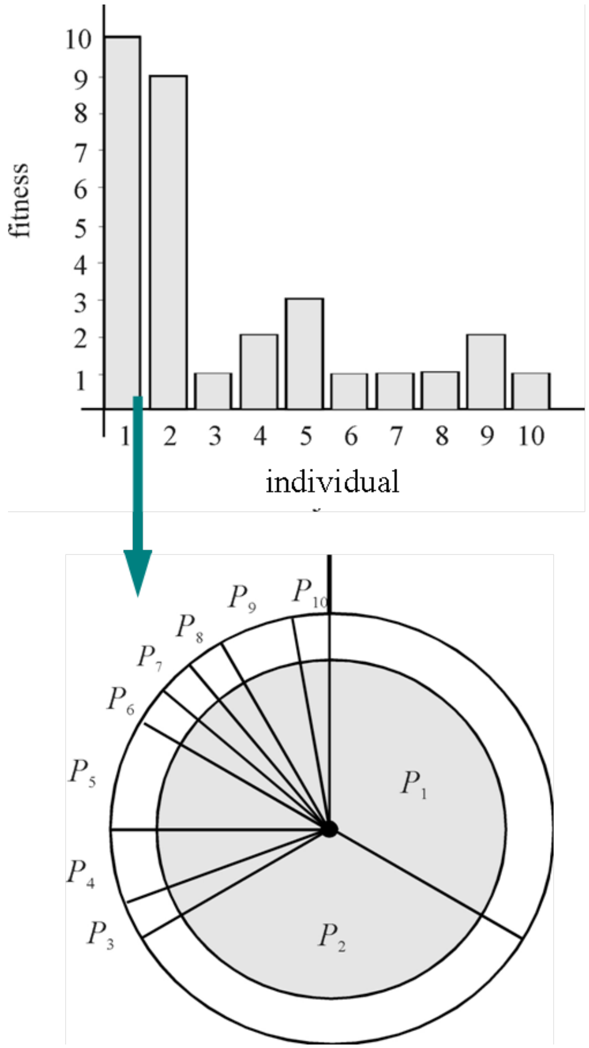
:: **Premature convergence** – a premature loss of diversity in the population with the search converging to a sub-optimal solution.

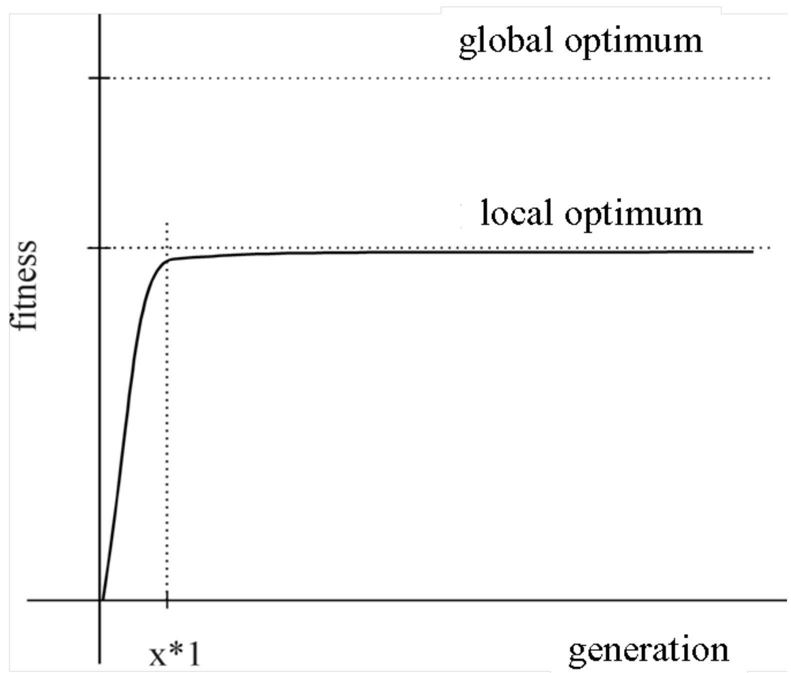
- early stages of the evolution search process.

:: **Stagnation** – ineffective search due to a weak selective pressure.

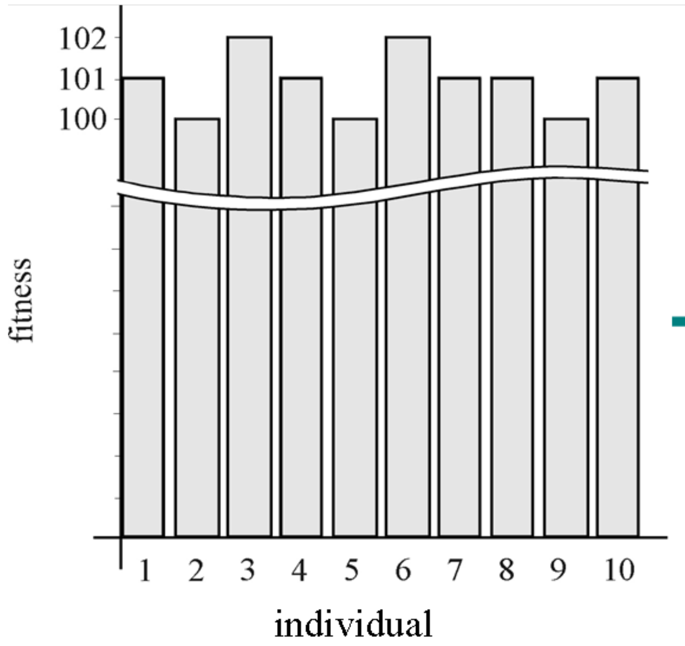
- later stages of the evolution search process.

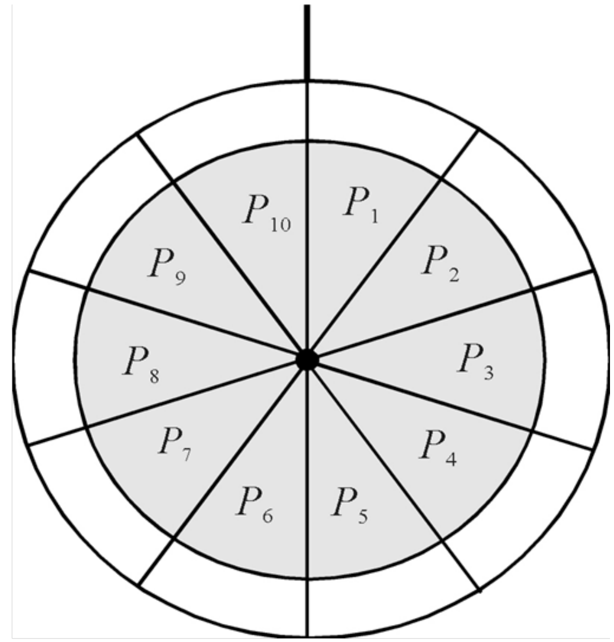
Premature Convergence





Stagnation


102
101
100
fitness
1
2
3
4
5
6
7
8
9
10
individual


P_{10}
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
P_9

How to Deal with it?

:: Balance between **exploration** and **exploitation**.

- How to achieve the optimal selective pressure during the whole evolution search?

:: Options

- scaling techniques,
- proper selection mechanisms,
- fitness sharing and crowding,
- . . . .

Scaling

:: **Linear scaling** – adjustment of the fitness values distribution in order to get desired selection pressure

$$\sigma = f_{max}/f_{avg}$$

The actual chromosomes' fitness is scaled as

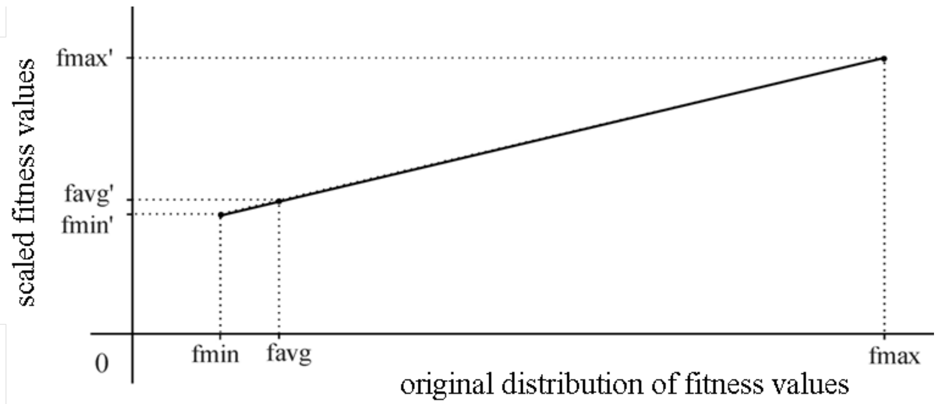
$$f_i' = a \cdot f_i + b$$

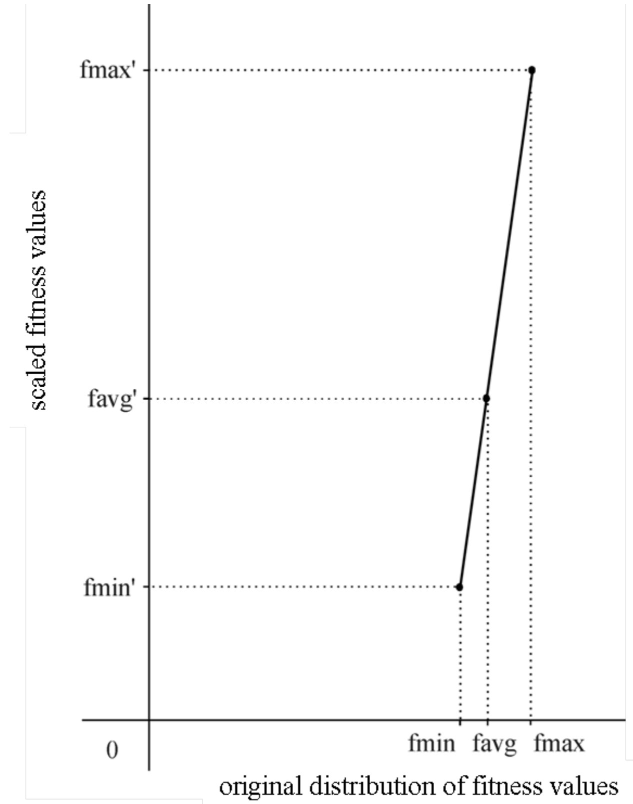
Parameters a and b are selected so that

- the average fitness is mapped to itself, and
- the best fitness is increased by a desired multiple of the average fitness.

  Typical value of σ is from $(1.5, 2.0)$

Effect of Linear Scaling

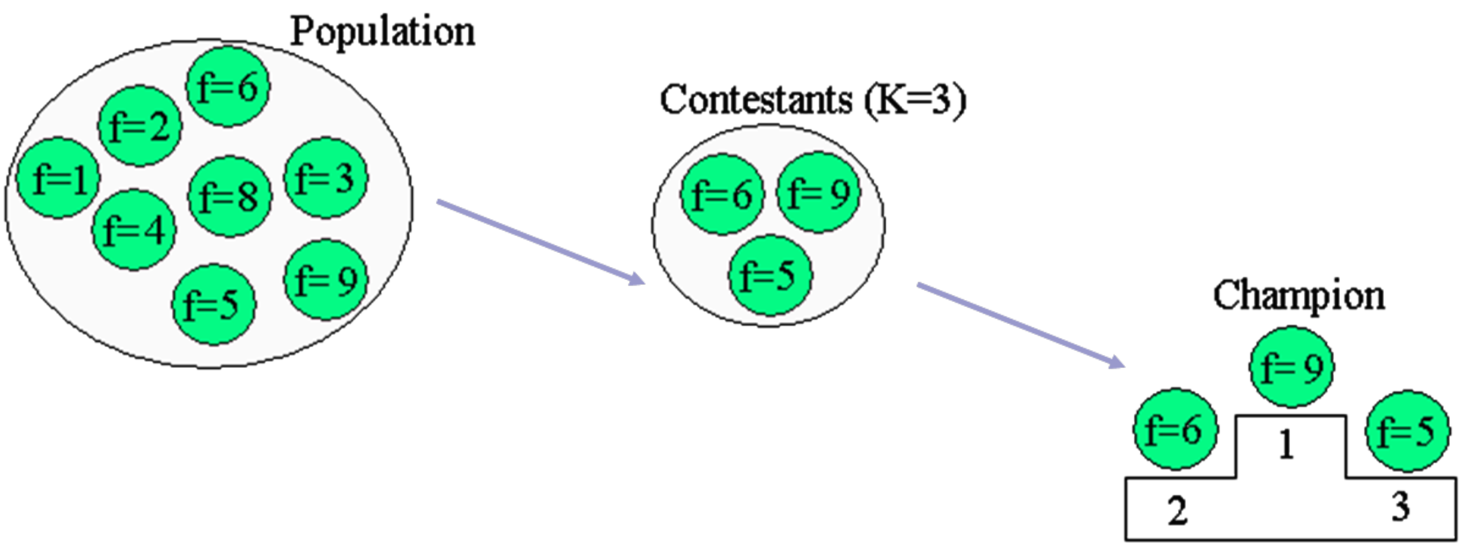




Tournament Selection

:: **Tournament selection** – the best out of n randomly chosen individuals is selected.

- n is the size of the tournament,
- rank-based method – absolute differences among individuals do not count.

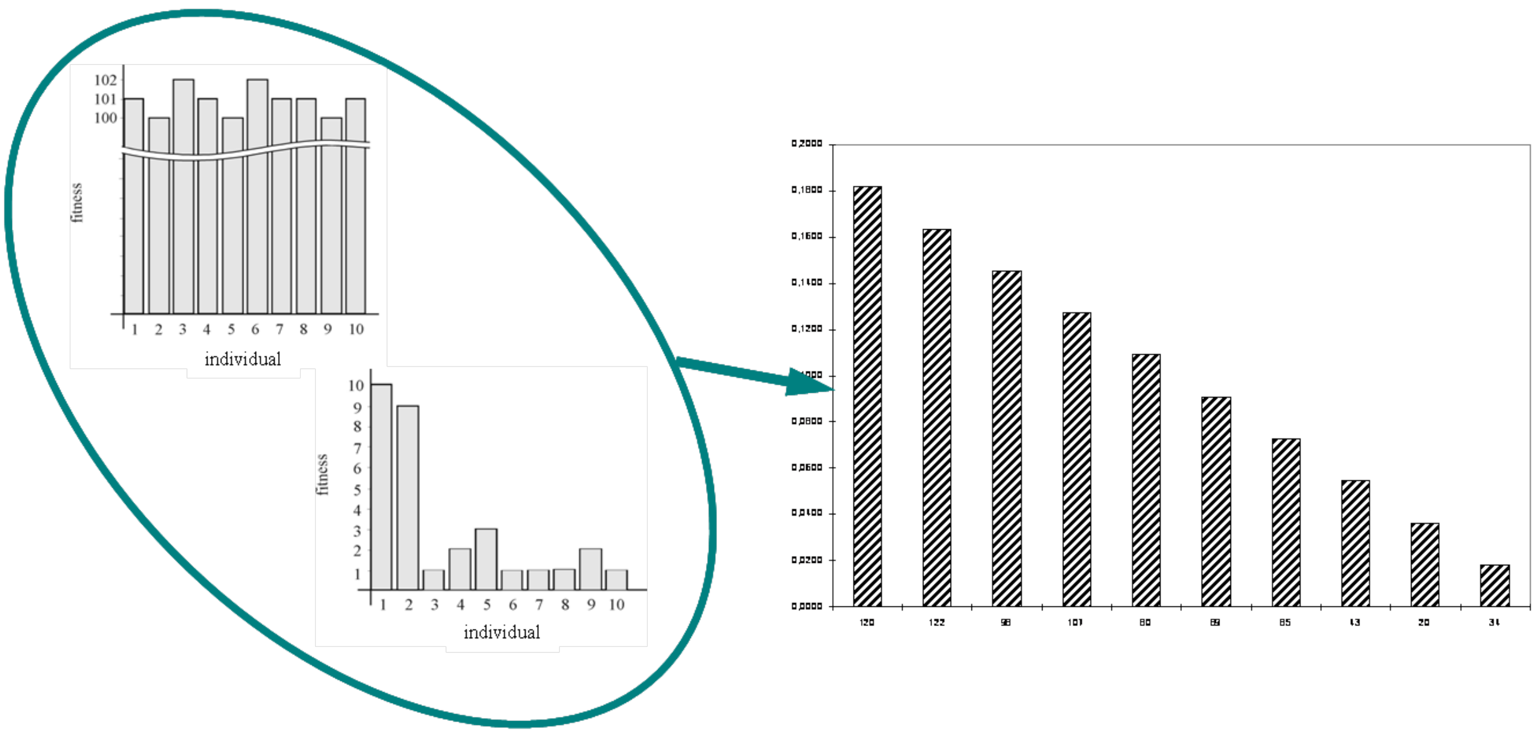


Rank Selection

:: **Rank selection** – fitness of the individual is calculated based on the rank of the individual in the population according to the formula

$$f_i' = PopSize - rank(i) + shift$$

where $shift$ is the fitness of the worst individual in the population.



Genetic Operators: Crossover

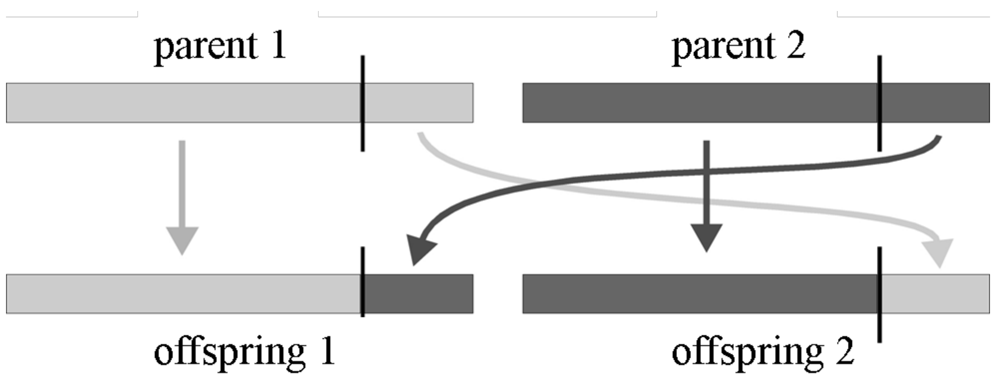
:: Idea

- given two well-fit solutions to the given problem, it is possible to get a new solution by properly mixing the two that is even better than both its parents.

:: Role of crossover

- sampling (exploration) of the search space

Example: 1-point crossover

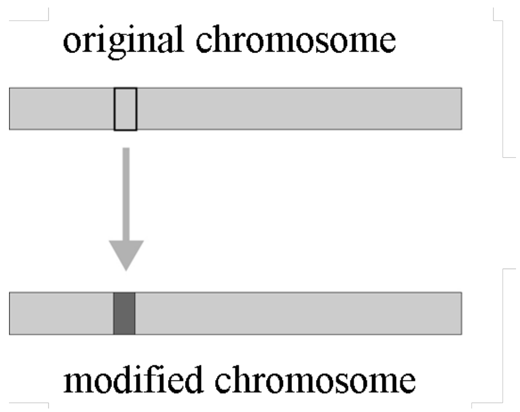


Genetic Operators: Mutation

Role of mutation

- preservation of a population diversity,
- minimization of a possibility of loosing some important piece of genetic information.

Single bit-flipping mutation



Population with missing genetic information

0	0	1	1	0	0	0	1	1	0
0	1	1	0	0	1	0	1	0	0
0	0	0	1	1	0	1	0	1	1
0	1	0	0	1	0	0	1	1	1
0	1	1	0	0	0	0	1	0	1
			. . .						
0	1	0	0	1	1	0	1	0	0

Replacement Strategy

Replacement strategy defines

- how big portion of the current generation will be replaced in each generation, and
- which solutions in the current population will be replaced by the newly generated ones.

Two extreme cases

- **Generational** – the whole old population is completely rebuild in each generation (analogy of short-lived species).
- **Steady-state** – just a few individuals are replaced in each generation (analogy of longer-lived species).

Application Areas of Evolutionary Algorithms

EAs are popular for their

- **simplicity**,
- **effectiveness**,
- **robustness**.

Holland: *"It's best used in areas where you don't really have a good idea what the solution might be. And it often surprises you with what you come up with."*

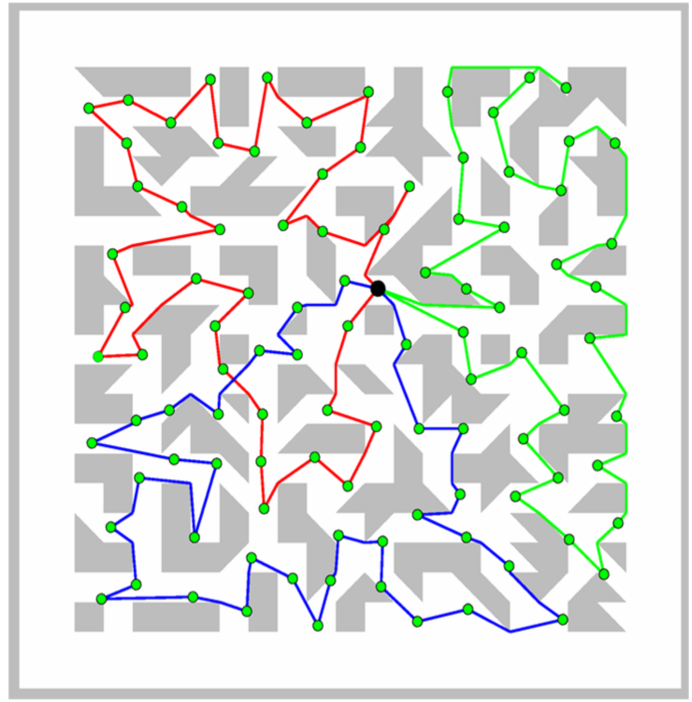
Applications

- control,
- engineering design,
- image processing,
- planning & scheduling,
- VLSI circuit design,
- network optimization & routing problems,
- optimal resource allocation,
- marketing,
- credit scoring & risk assessment,
- and many others.

Multiple Traveling Salesmen Problem

Rescue operations planning

- Given N cities and K agents, find an optimal tour for each agent so that every city is visited exactly once.
- A typical criterion to be optimized is the overall time spent by the squad (i.e., the slowest team member) during the task execution.



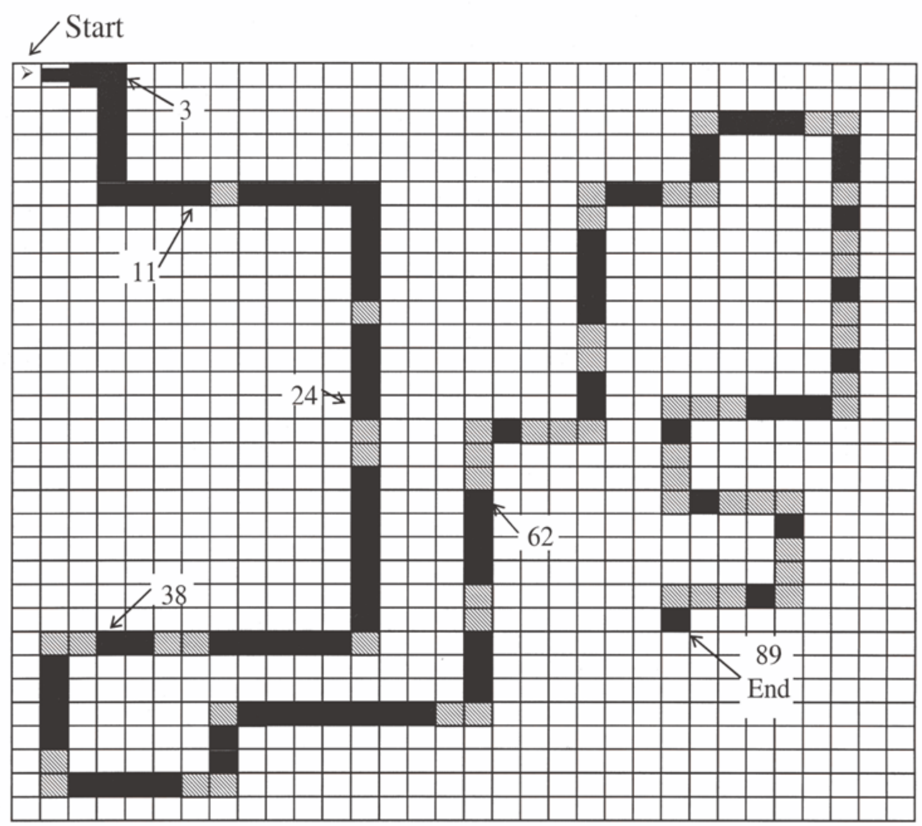
Artificial Ant Problem

Santa Fe trail

- 32×32 **grid** with 89 food pieces.
- **Obstacles**
  - $1\times, 2\times$ strait,
  - $1\times, 2\times, 3\times$ right/left.

Ant capabilities

- **detects** the food right in front of him in direction he faces.
- **actions** observable from outside
  - MOVE – makes a step and eats a food piece if there is some,
  - LEFT – turns left,
  - RIGHT – turns right,
  - NO-OP – no operation.



Goal is to find a strategy that would navigate an ant through the grid so that it finds all the food pieces in the given time (600 time steps).

Artificial Ant Problem: GA Approach

Collins a Jefferson 1991, standard GA using binary representation

Representation

- strategy represented by finite state machine,
- table of transitions coded as binary chromosomes of fixed length.

Example: 4-state FSM, 34-bit long chromosomes $(2 + 4 \times 8)$

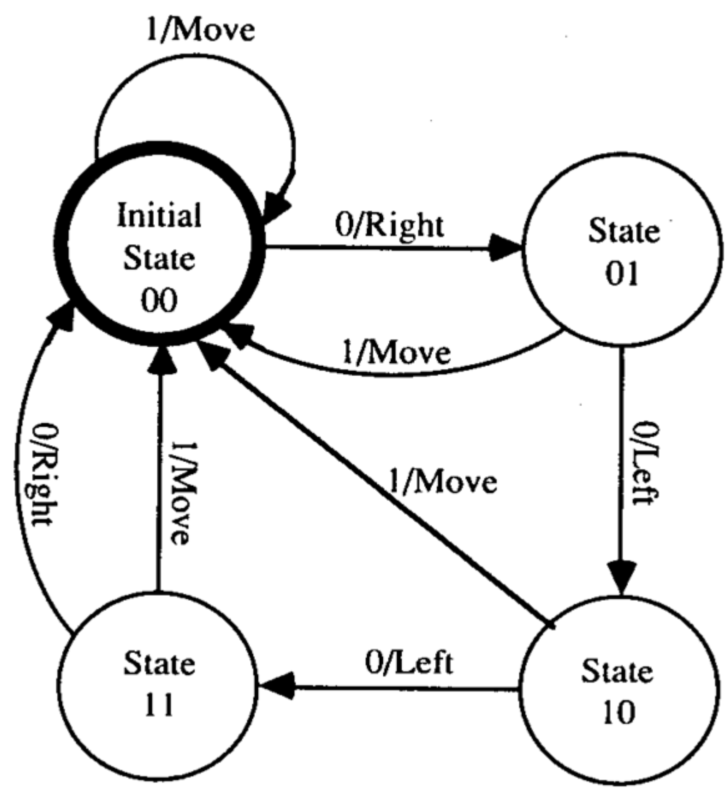
	Current state	Input	New state	Operation
1	00	0	01	10 = Right
2	00	1	00	11 = Move
3	01	0	10	01 = Left
4	01	1	00	11 = Move
5	10	0	11	01 = Left
6	10	1	00	11 = Move
7	11	0	00	10 = Right
8	11	1	00	11 = Move

00	0110	0011	1001	0011	1101	0011	0010	0011

Artificial Ant Problem: Example cont.

Ant behavior

- What happens if the ant "hits" an obstacle?
- What is strange with transition from state 10 to the initial state 00?
- When does the ant succeed?
- Is the number of states sufficient to solve the problem?
- Do all of the possible 34-bit chromosomes represent a feasible solution?



Artificial Ant Problem: GA result

Representation

- 32 states,
- $453 = 64 \times 7 + 5$ bits !!!

Population size: 65.536 !!!

Number of generations: 200

Total number of samples tried: 13×10^6 !!!

Schema Theory

Schema theory (J. Holland, 1975) – tries to analyze effect of selection, crossover and mutation on the population's genotype in order to answer the question: **"Why and How Evolutionary Algorithms Work?"**

In its original form the schema theory assumes:

- binary representation,
- proportionate roulette wheel selection,
- 1-point crossover and bit-flip mutation.

Schema theory

:: **Schema** – a template, which defines set of solutions from the search space with certain specific similarities.

- consists of 0s, 1s (fixed values) and wildcard symbols * (any value),
- covers 2^r strings, where r is a number of $*$ used in the schema.

  Example: schema $S =${11*0*} covers strings 11000, 11001, 11100, and 11101

:: **Schema properties**

- **Defining length** $\delta(S)$ (compactness) – distance between first and last non-* in a schema ($=$ number of positions where 1-point crossover can disrupt it).
- **Order** $o(S)$ (specificity) – a number of non-*'s ($=$ number of positions where simple bit swapping mutation can disrupt it).
  - Chromosomes are order l schemata, where l is length of chromosome (in bits or loci).
  - Chromosomes are instances (or members) of lower-order schemata.
  - How many schemata are matched by a string of length l?
- **Fitness** $f(S)$ (quality) – average fitness computed over all covered strings.

  Example: $S =${**1*01*0**}: $\delta(S) = 5$, $o(S) = 4$

Schema Properties: Example

:: **8-bit Count Ones problem** – maximize a number of ones in 8-bit string.

string	fitness		string	fitness
00000000	0		11011111	7
00000001	1	...	10111111	7
00000010	1		01111111	7
00000100	1		11111111	8

Assume schema S_a ={1*1**10*} vs. S_b ={*0*0****}:

- **defining length**: $\delta(S_a) = 7 - 1 = 6$, $\delta(S_b) = 4 - 2 = 2$
- **order**: $o(S_a) = 4$, $o(S_b) = 2$
- **fitness of** S_a: S_a covers 2^4 strings in total

1 string of fitness 3
4 string of fitness 4 $\quad f(S_a) = (1 \cdot 3 + 4 \cdot 4 + 6 \cdot 5 + 4 \cdot 6 + 1 \cdot 7)/16$
6 string of fitness 5 $\quad f(S_a) = 80/16 = 5$
4 string of fitness 6
1 string of fitness 7

fitness of S_b: $S_b = (1 \cdot 0 + 6 \cdot 1 + 15 \cdot 2 + 20 \cdot 3 + 15 \cdot 4 + 6 \cdot 5 + 1 \cdot 6)/2^6 = 192/64 = 3$

Question: What would be a fitness of S ={*0*1****} compared to S_b?

Schema Theorem Derivation: Effect of Reproduction

Let $m(S,t)$ be number of instances (strings) of schema S in population of size n at time t.

Question: How do schemata propagate? What is a lower bound on change in sampling rate of a single schema from generation t to $t+1$?

Effect of fitness-proportionate roulette wheel selection

A string a_i is copied according to its fitness; it gets selected with probability

$$p_i = \frac{f_i}{\sum f_j}.$$

After picking n strings with replacement from the population at time t, we expect to have $m(S,t+1)$ representatives of the schema S in the population at time $t+1$ as given by the equation

$$m(S,t+1) = m(S,t) \cdot n \cdot \frac{f(S)}{\sum f_j},$$

where $f(S)$ is the fitness of schema S at time t.

Schema Theorem Derivation: Effect of Reproduction

Let $m(S,t)$ be number of instances (strings) of schema S in population of size n at time t.

Question: How do schemata propagate? What is a lower bound on change in sampling rate of a single schema from generation t to $t+1$?

Effect of fitness-proportionate roulette wheel selection

A string a_i is copied according to its fitness; it gets selected with probability

$$p_i = \frac{f_i}{\sum f_j}.$$

After picking n strings with replacement from the population at time t, we expect to have $m(S,t+1)$ representatives of the schema S in the population at time $t+1$ as given by the equation

$$m(S,t+1) = m(S,t) \cdot n \cdot \frac{f(S)}{\sum f_j},$$

where $f(S)$ is the fitness of schema S at time t.

The formula can be rewritten as

$$m(S,t+1) = m(S,t) \cdot \frac{f(S)}{f_{avg}},$$

where f_{avg} is the average fitness of the population.

Schema Theorem Derivation: Effect of Crossover and Mutation

Effect of 1-point Crossover

- Survival probability p_s – let's make a conservative assumption that crossover within the defining length of S is always disruptive to S, and ignore gains.
- Crossover probability p_c – fraction of population that undergoes crossover.

$$p_s \geq 1 - (p_c \cdot \delta(S)/(L-1))$$

Example: Compare survival probability of $S = (11****)$ and $S = (1****0)$.

Schema Theorem Derivation: Effect of Crossover and Mutation

Effect of 1-point Crossover

- Survival probability p_s – let's make a conservative assumption that crossover within the defining length of S is always disruptive to S, and ignore gains.
- Crossover probability p_c – fraction of population that undergoes crossover.

$$p_s \geq 1 - (p_c \cdot \delta(S)/(L-1))$$

Example: Compare survival probability of $S = (11 * * * *)$ and $S = (1 * * * * 0)$.

Effect of Mutation

Each fixed bit of schema (o(S) of them) changes with probability p_m, so they all stay unchanged with probability

$$p_s = (1 - p_m)^{o(S)}$$

that can be approximated as

$$p_s = (1 - o(S) \cdot p_m)$$

assuming $p_m \ll 1$.

Schema Theorem Derivation (cont.)

:: Finally, we get a "classical" form of the **reproductive schema growth equation**:

$$m(S, t+1) \geq m(S, t) \cdot \frac{f(S)}{f_{avg}} \cdot [1 - p_c \cdot \frac{\delta(S)}{L-1} - o(S) \cdot p_m].$$

What does it tell us?

Schema Theorem Derivation (cont.)

:: Finally, we get a "classical" form of the **reproductive schema growth equation**:

$$m(S, t+1) \geq m(S,t) \cdot \frac{f(S)}{f_{avg}} \cdot [1 - p_c \cdot \frac{\delta(S)}{L-1} - o(S) \cdot p_m].$$

What does it tell us?

:: **Schema theorem**: Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm.

:: **Building Block Hypothesis**: A genetic algorithm seeks near-optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks.

David Goldberg: "*Short, low-order, and highly fit schemata are sampled, recombined, and resampled to form strings of potentially higher fitness... we construct better and better strings from the best partial solutions of the past samplings.*"

:: Y. Davidor: "*The whole GA theory is based on the assumption that one can state something about the whole only by knowing its parts.*"

Corollary: The problem of coding for a GA is critical for its performance, and that such a coding should satisfy the idea of short building blocks.

EA Materials: Reading, Demos, Software

:: Reading

- D. E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
- Z. Michalewicz: Genetic Algorithms + Data Structures = Evolution Programs, Springer, 1998.
- Z. Michalewicz: How to solve it? Modern heuristics. 2nd ed. Springer, 2004.

:: Demos

- M. Obitko: Introduction to genetic algorithms with java applets,
  http://cs.felk.cvut.cz/ xobitko/ga/

:: Software

- ECJ 16 – A Java-based Evolutionary Computation Research System
  http://cs.gmu.edu/ eclab/projects/ecj/
- PISA – A Platform and Programming Language Independent Interface for Search Algorithms
  http://www.tik.ee.ethz.ch/sop/pisa/?page=selvar.php