# **Evolutionary Algorithms: Introduction**

# Jiří Kubalík Department of Cybernetics, CTU Prague



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

#### **Contents**

### 1. Introduction to Evolutionary Algorithms (EAs)

- Simple Genetic Algorithm (SGA),
- Areas of EA's applications,
- SGA example: Evolving strategy for an artificial ant problem.
- Schema theory a schema, its properties, exponential growth equation and its consequences.

### 2. Genetic Programming (GP) and Grammatical Evolution (GE)

- Tree representation, closure condition, 'strong typing',
- Application of GP to the artificial ant problem,
- Other examples.

### 3. Multi-objective EAs (MOEAs)

- Multi-objective optimization
- Concept of dominance and Pareto-optimality,
- NSGA, NSGA-II, SPEA, SPEA2.

### **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 Algorithms: Characteristics**

### **EAs** 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

- Gene (also features, characters) elementary units from which chromosomes are made.
  - each gene is located at certain place of the chromosome called locus,
  - 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.
- Chromosome entire representation of the solution.
- Fitness quality measure assigned to an individual, expresses how well it is adapted to the environment.
- Individual (chromosome + its quality measure "fitness value") a solution to a problem.
- Genotype what's on the chromosome.
- Phenotype interpretation of the genotype (e.g., binary sequence may map to integers or reals, or order of execution, etc.).

# Representation

In conventional GAs, the solution is represented by a binary string

■ binary string - 101101100101101

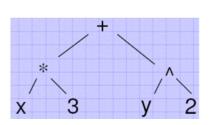
### Representation

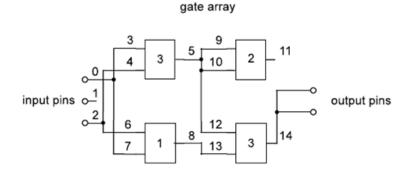
In conventional GAs, the solution is represented by a binary string

binary string - 101101100101101

However, other types of representation can be more suitable for the problem at hand

- real-valued string 3,24 1,78 -2,61
- string of chars  $D \rightarrow E \rightarrow A \rightarrow C \rightarrow B$
- tree or graph





### **Evaluation Function**

### **Objective (Fitness) function**

- the only information about the sought solution the algorithm is endowed with,
- must be defined for every possible chromosome.

### Fitness function may be

multimodal,

nonlinear,

discrete,

noisy,

multidimensional,

multiobjective.

### Fitness does not have to be define analytically

- simulation results,
- classification success rate.

Fitness function should not be too costly!!!

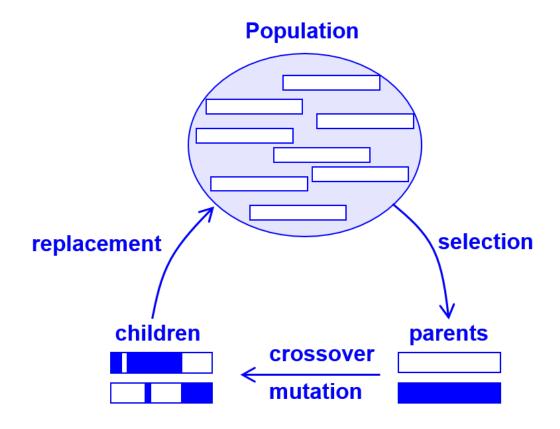
# **Example: Coding & Evaluation**

### :: Function optimization

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

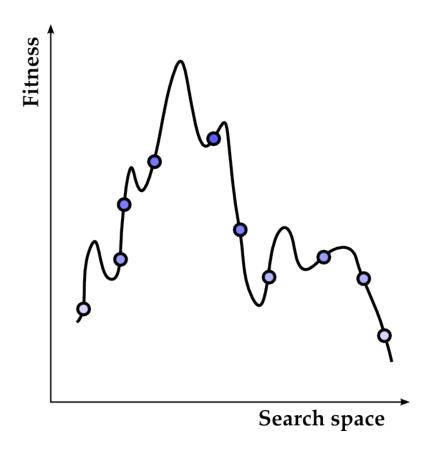
genotype	phenotype	fitness	
00000, 01010	0, 10	100	
00001, 11001	1, 25	625 + 1 = 626	
01011, 00011	11, 3	121 + 9 = 130	
11011, 10010	27, 18	729 + 324 = 1053	

# **Evolutionary Cycle**

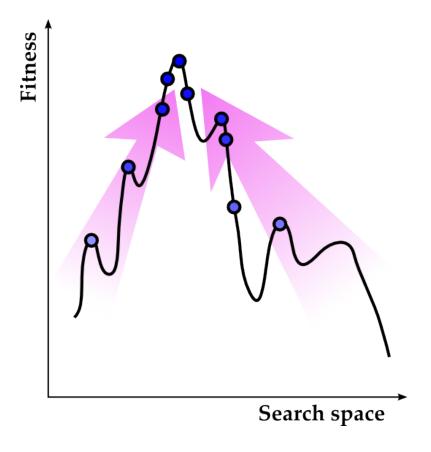


# **Idealized Illustration of Evolution**

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

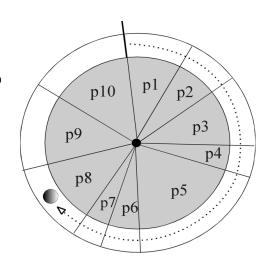
Reproduction (parental selection) models nature's survival-of-fittest principle

- 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

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



### **Expected frequencies vs. observed frequencies**

■ Expected selection frequency – given selection probability of i-th individual,  $P_i$ , and PopSize individuals to be selected, we expect to get on average  $PopSize * P_i$  copies of individual i.

# Reproduction

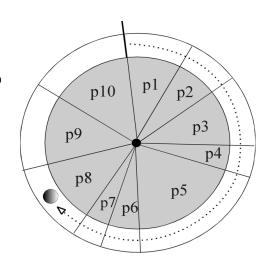
Reproduction (parental selection) models nature's survival-of-fittest principle

- 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

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



### Expected frequencies vs. observed frequencies

- Expected selection frequency given selection probability of i-th individual,  $P_i$ , and PopSize individuals to be selected, we expect to get on average  $PopSize * P_i$  copies of individual i.
- Observed selection frequency can be anywhere between 0 and PopSize.

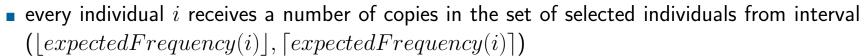
# **Stochastic Universal Sampling**

SUS ensures that the observed selection frequencies of each individual are in line with the expected frequencies:

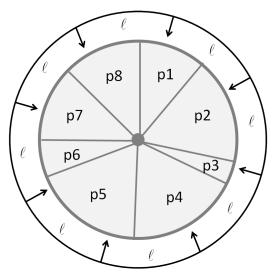
• extra wheel, let's denote it a *pointer wheel*, with equidistantly distributed PopSize pointers

Ex.: If we are selecting 8 individuals, the pointer wheel will have 8 pointers distributed with 360/8=45 degrees step size.

- works by making a single spin of the roulette wheel
- lacktriangleright an arbitrary rotation of the pointer wheel determines a whole set of PopSize selected individuals



Ex.: If we have an individual that occupies 4.5% of the roulette wheel and we select 100 individuals, we would expect on average 4.5 copies for that individual to be selected. Then, the individual will be selected either four or five times. Neither more, nor less.



# Reproduction: Premature Convergence & Stagnation

Two (strongly related) issues in the evolution process

- population diversity,
- selective pressure.

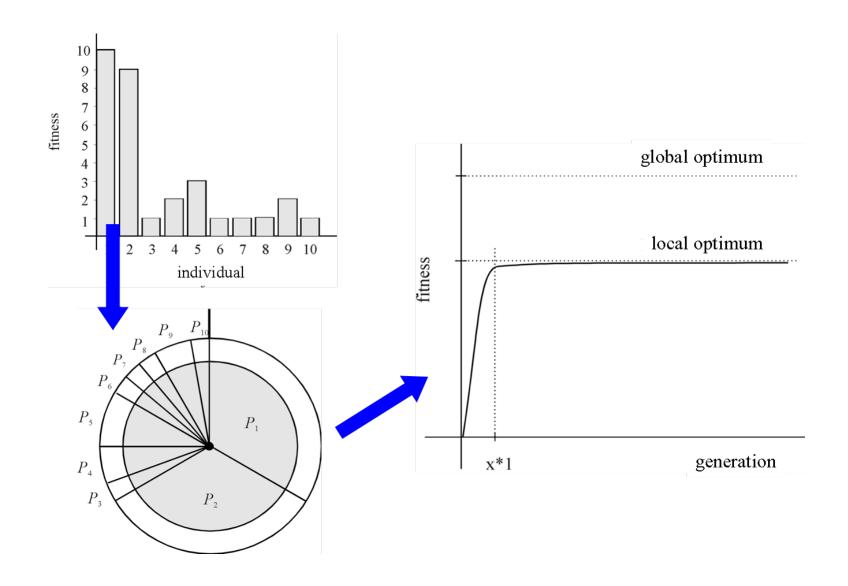
Premature convergence – a premature loss of diversity in the population with the search converging to a sub-optimal solution.

early stages of the evolution search process.

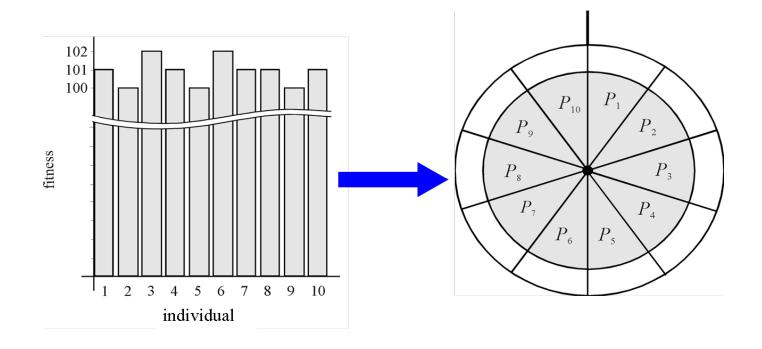
**Stagnation** – ineffective search due to a weak selective pressure.

later stages of the evolution search process.

# **Premature Convergence**



# **Stagnation**



# 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

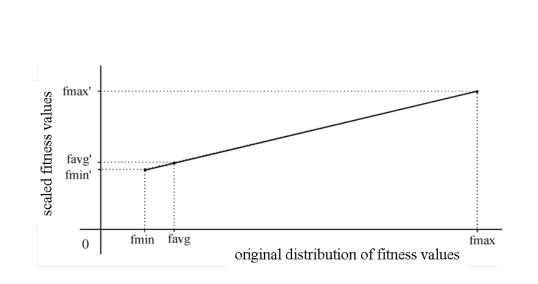
$$f_i' = a \cdot f_i + b$$

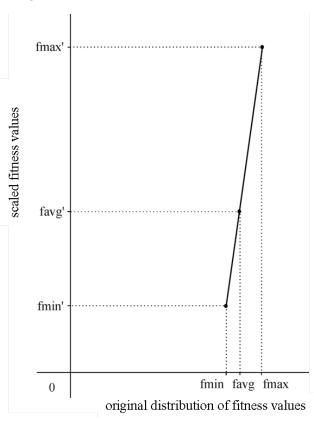
Parameters a and b are selected so that

- the average fitness is mapped to itself, and
- the best fitness is increased by a desired multiple of the average fitness. Typical value of  $\sigma$  is from (1.5,2.0)

# **Effect of Linear Scaling**

Linear scaling resolves both the premature convergence and stagnation.



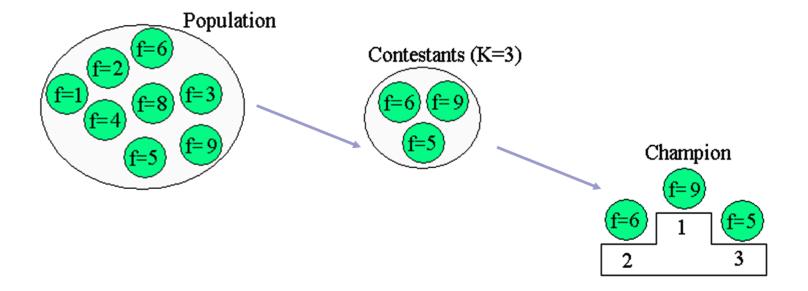


■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ Œvolutionary Algorithms

# **Tournament Selection**

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

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



# **Genetic Operators: Crossover**

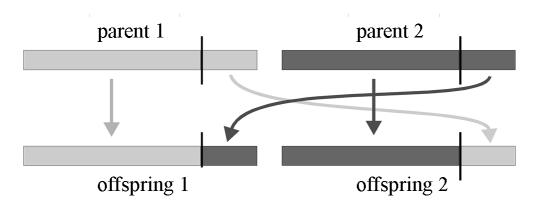
#### Role of crossover

sampling (exploration) of the search space

#### 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.

### Example: 1-point crossover



# **EAs and Optimization with Constraints**

**Search space with constraints** – contains both feasible and infeasible solutions.

Ex.: Assuming the Traveling Salesman Problem and a simple one-point crossover, it is easy to imagine a situation that an infeasible solution is produced by the operator, even when both parents are feasible solutions.

```
parent1: 4 8 1 | 3 5 2 7 6 offspring1: 4 8 1 | 1 3 6 5 4

parent2: 2 7 8 | 1 3 6 5 4 offspring2: 2 7 8 | 3 5 2 7 6
```

Neither of the two offspring represents a feasible solution – either some city is missing or is duplicated in the tour.

# **TSP: Edge-Recombination Operator**

### **Direct representation**

genotype: a e d b c

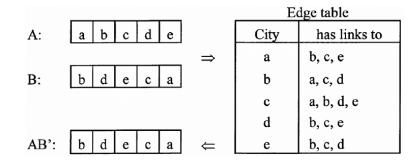
tour:  $a \rightarrow e \rightarrow d \rightarrow b \rightarrow c$ 

### Edge recombination crossover

not yet in the created tour.

- Create a table of neighbors (edge table) for each city i there is a list of cities that have a link to i in the parental tours.
- Start creating a tour in a randomly chosen city, currentCity.

Remove all occurrences of *currentCity* from the edge table.



- Choose a new *currentCity* among the unused neighbors of *currentCity* in the edge table.

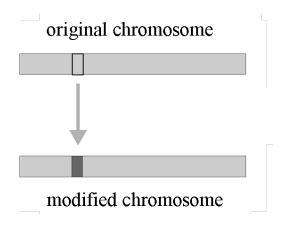
  If *currentCity* has already an empty list of unused neighbors, choose an arbitrary city that is
  - Remove all occurrences of currentCity from the edge table.
  - Repeat this step until all cities have been added to the tour.

# **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 general 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).

# **Generational Replacement Strategy**

```
initialize(oldPopulation)
   evaluate(oldPopulation)
   while(not termination condition)
        newPopulation \leftarrow \mathsf{bestOf}(oldPopulation)
4
                                                          // elitism
        while(newPopulation not full)
             parents \leftarrow select(oldPopulation)
             offspring \leftarrow crossover(parents)
             mutate(offspring) // optional
             evaluate(offspring)
10
             newPopulation \leftarrow offspring
11
        swap(oldPopulation, newPopulation)
12 return bestOf(oldPopulation)
```

# **Steady-State Replacement Strategy**

# **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**

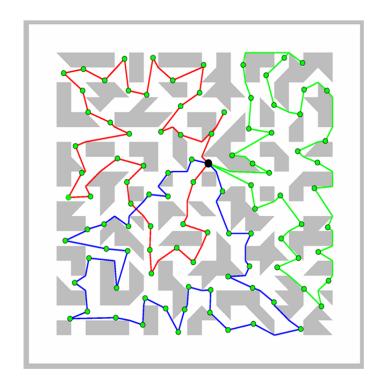
- control,
- engineering design,
- image processing,
- planning & scheduling,
- VLSI circuit design,

- 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.



□ □ □ □ □ □ □ □ □ Œvolutionary Algorithms

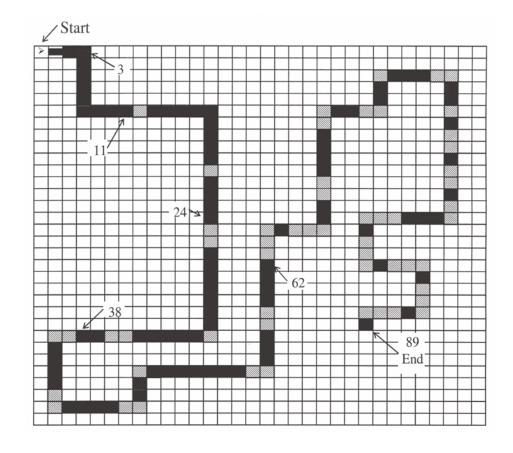
### **Artificial Ant Problem**

#### Santa Fe trail

- $32 \times 32$  **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).

□ □ □ □ □ □ □ □ Œvolutionary Algorithms

# **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  $(2 + 4 \times 8)$ 

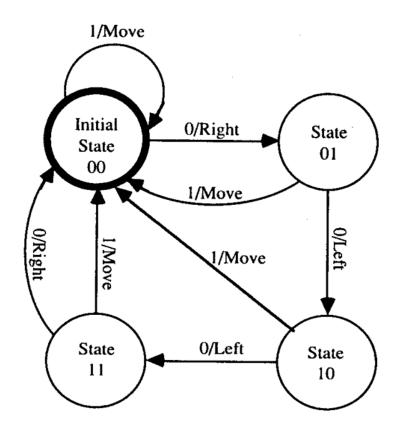
	Current state	Input	New state	Operation
1	00	0	01	10 = Right
2	00	1	00	11 = Move
3	01	0	10	01 = Left
4	01	1	00	11 = Move
5	10	0	11	01 = Left
6	10	1	00	11 = Move
7	11	0	00	10 = Right
8	11	1	00	11 = Move

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:  $13 \times 10^6$  !!!

# **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 the schema).
- Order o(S) (specificity) a number of non-\*'s (= number of positions where simple bit swapping mutation can disrupt the schema).
  - Chromosomes are order l schemata, where l is length of chromosome (in bits or loci).
  - Chromosomes are instances (or members) of lower-order schemata.
- ${\bf \blacksquare}$  Fitness f(S) (quality) average fitness computed over all covered strings.

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

□ □ □ Œvolutionary Algorithms

# **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  $f(S_a) = (1 \cdot 3 + 4 \cdot 4 + 6 \cdot 5 + 4 \cdot 6 + 1 \cdot 7)/16$ 

6 string of fitness 5  $f(S_a) = 80/16 = 5$ 

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: How will a fitness of  $S = \{*0*1****\}$  compare to  $S_b$ ?

# Schema Theorem Derivation: Effect of Reproduction

Let m(S,t) be a 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 a 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 a 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 a 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.

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

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.

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

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:

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

What does it tell us?

# Schema Theorem Derivation (cont.)

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

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

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.

**E**volutionary Algorithms

# **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 applet, http://cs.felk.cvut.cz/~xobitko/ga/

#### **Software**

- ECJ 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