Evolutionary Algorithms: Introduction

Jiří Kubalík
The Czech Institute of Informatics, Robotics and Cybernetics
CTU Prague

http://cw.felk.cvut.cz/doku.php/courses/a0m33eoa/start

Contents

- Representation
- Operators
- Population diversity preservation
- Areas of EA's applications
- SGA example: Evolving strategy for an artificial ant problem
- Schema theory

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 in 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** – 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1

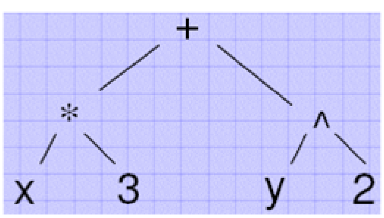
Representation

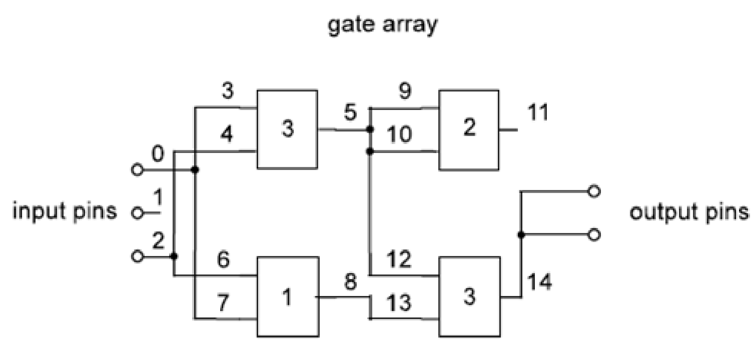
In conventional GAs, the solution is represented by a binary string

- **binary string** – 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1

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→E→A→C→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,
- 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!!!

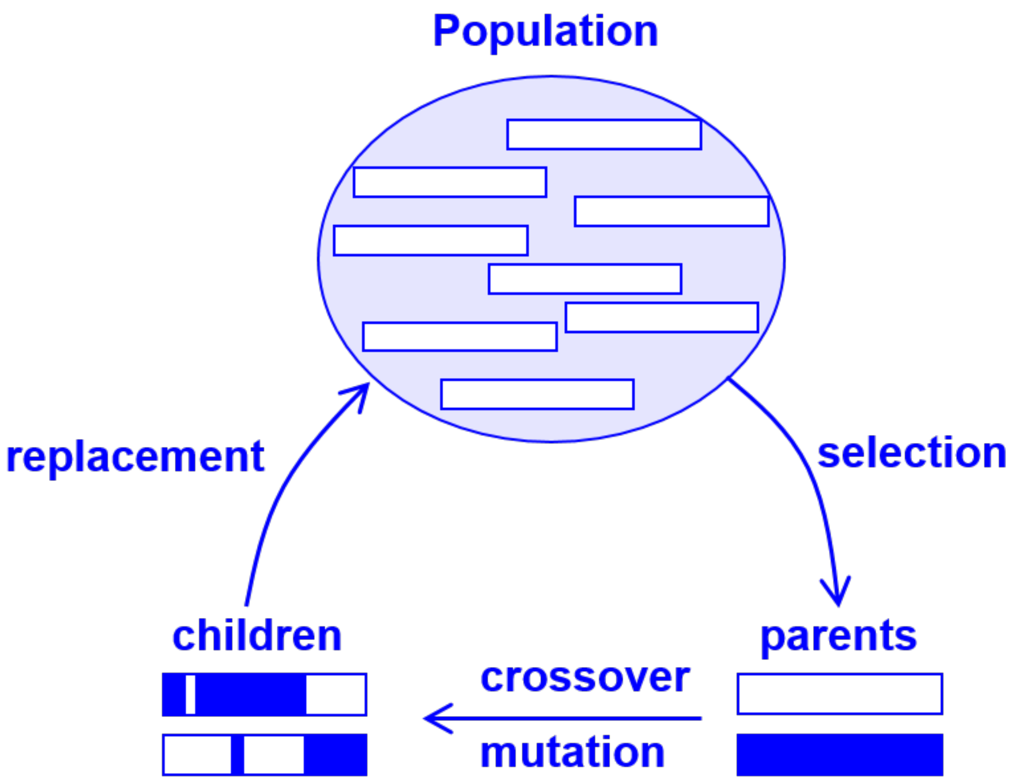
Example: Coding & Evaluation

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.

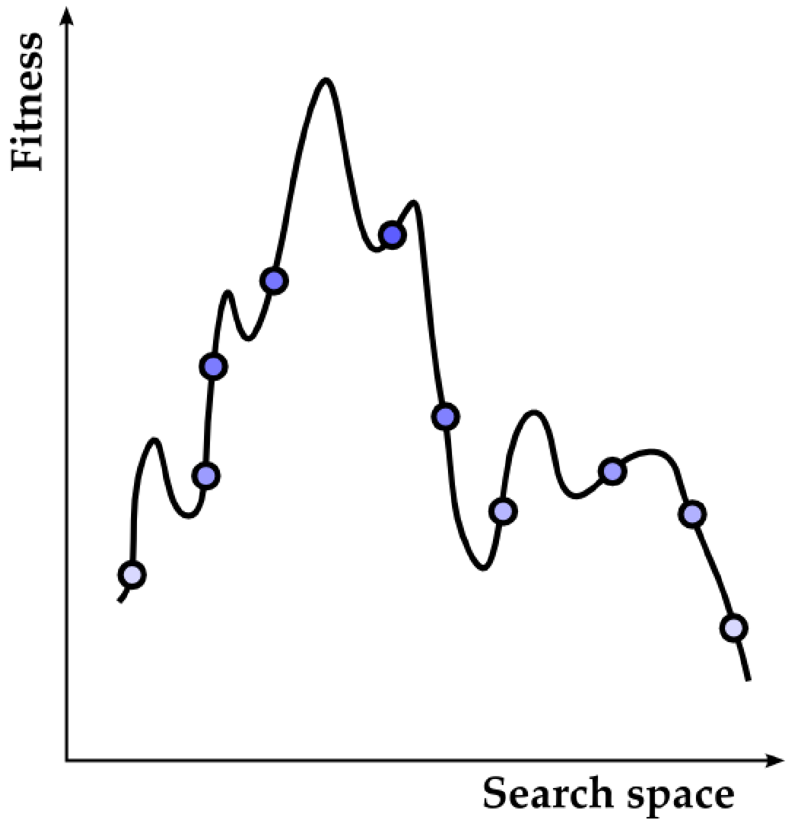
genotype	phenotype	fitness
0 0 0 0 0, 0 1 0 1 0	**0, 10**	**100**
0 0 0 0 1, 1 1 0 0 1	**1, 25**	**625 + 1 = 626**
0 1 0 1 1, 0 0 0 1 1	**11, 3**	**121 + 9 = 130**
1 1 0 1 1, 1 0 0 1 0	**27, 18**	**729 + 324 = 1053**

Evolutionary Cycle

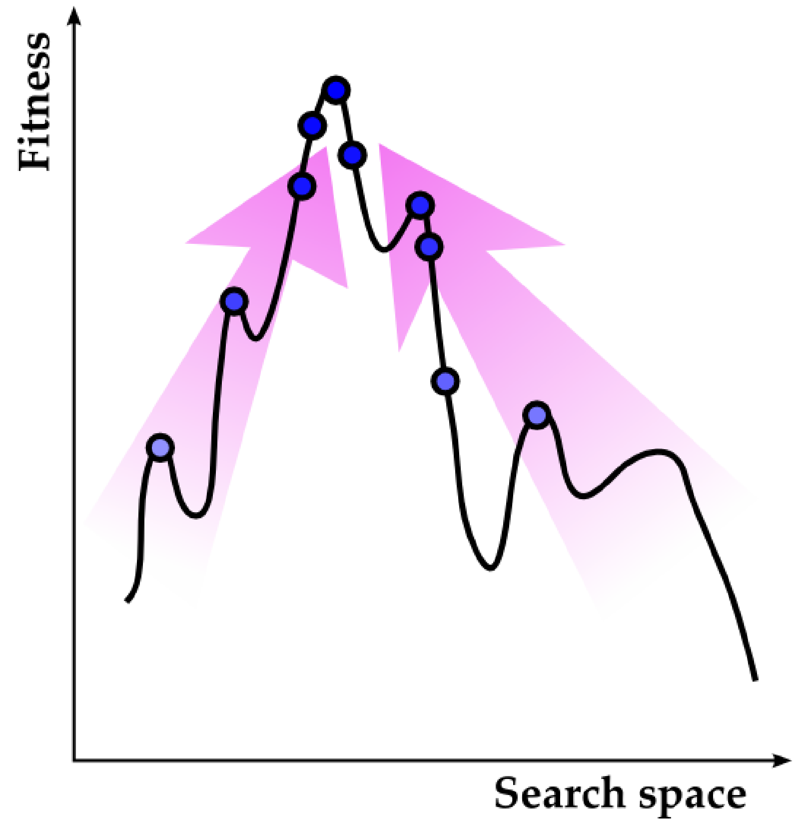


Idealized Illustration of Evolution

- Uniformly 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

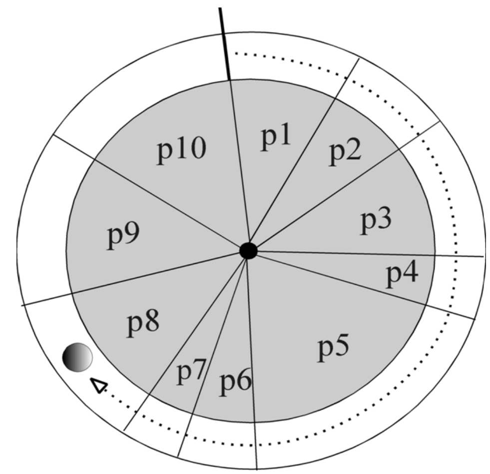
Reproduction (parental selection) models nature's survival-of-fittest principle

- 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

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



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

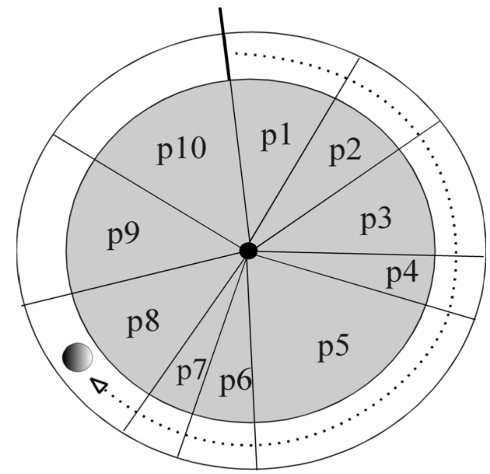
Reproduction (parental selection) models nature's survival-of-fittest principle

- 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

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



Expected frequencies vs. observed frequencies

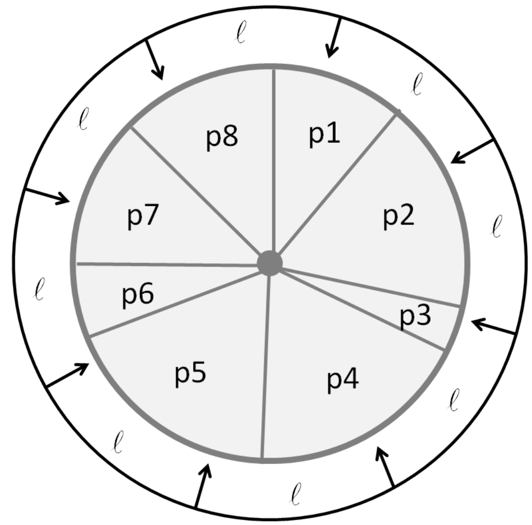
- *Expected selection frequency* – given selection probability of i-th individual, P_i, and $PopSize$ individuals to be selected, we expect to get on average $PopSize * P_i$ copies of individual i.
- *Observed selection frequency* can be anywhere between 0 and $PopSize$.

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
- an arbitrary rotation of the pointer wheel determines a whole set of $PopSize$ selected individuals
- every individual i receives a number of copies in the set of selected individuals from interval $(\lfloor expectedFrequency(i) \rfloor, \lceil expectedFrequency(i) \rceil)$

  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**.

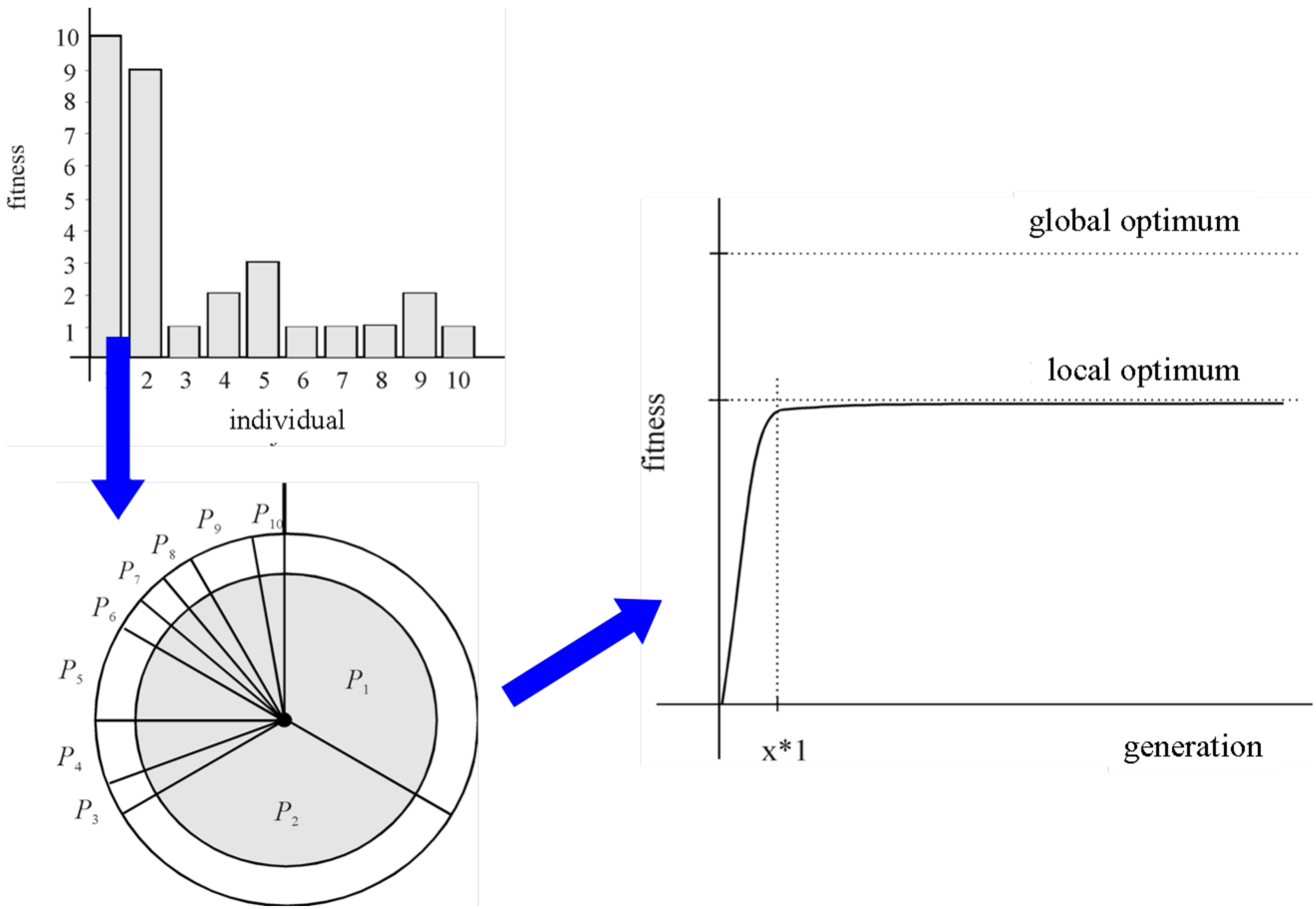
Premature convergence – a premature loss of diversity in the population with the search converging to a sub-optimal solution.

- early stages of the evolution search process.

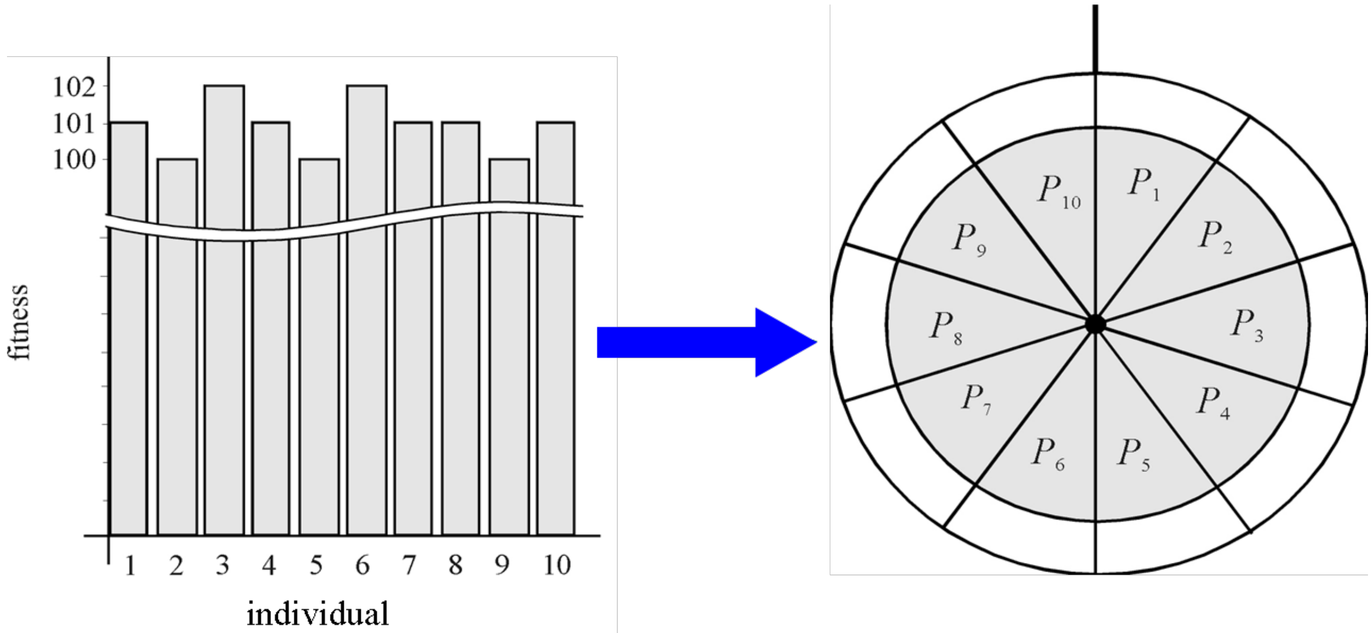
Stagnation – ineffective search due to a weak selective pressure.

- later stages of the evolution search process.

Premature Convergence



Stagnation


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

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

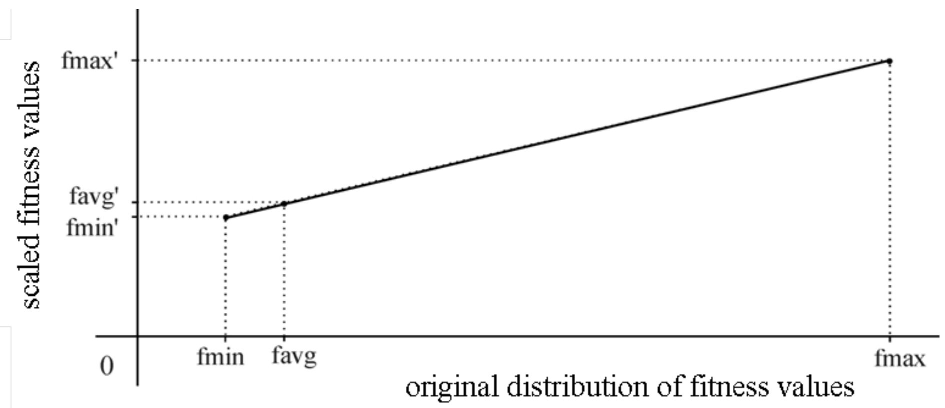
Parameters a and b are determined so that

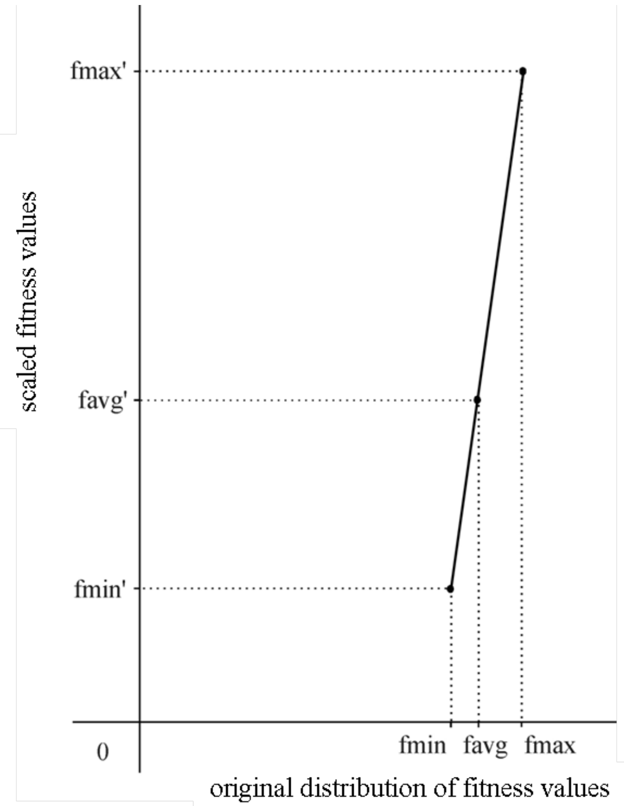
- the average fitness is mapped to itself, and
- the best fitness is increased by a desired multiple of the average fitness.

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

Effect of Linear Scaling

Linear scaling help to remedy both the premature convergence and stagnation.





Diversity Preservation: Fitness Sharing

Diversity preservation method proposed for solving multi-modal optimization problems so that GA is able to discover and evenly sample all optima.

Idea – diversity in the population is preserved by degrading the fitness of similar solutions

Algorithm for calculating the shared fitness value of i-th individual in population of size N

1. calculate *sharing function* value with all solutions in the population according to

$$Sh(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma_{share}})^{\alpha}, & \text{if } d_{ij} \leq \sigma_{share} \\ 0, & \text{otherwise.} \end{cases}$$

2. calculate niche count nc_i as follows

$$nc_i = \sum_{j=1}^{N} Sh(d_{ij})$$

3. calculate *shared fitness* as

$$f_i' = f_i / nc_i$$

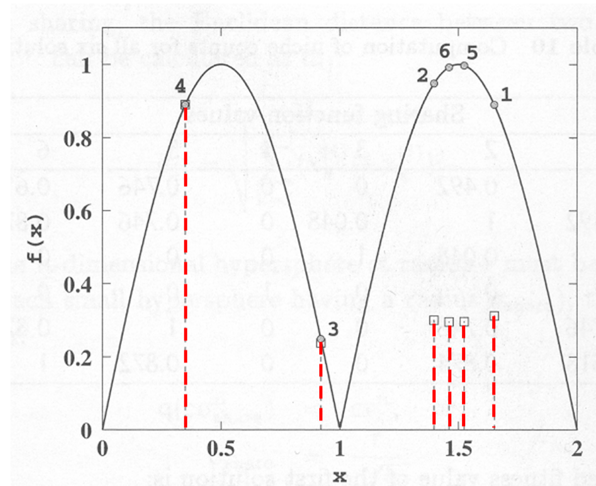
Remark: If $d = 0$ then $Sh(d) = 1$ meaning that two solutions are identical. If $d \geq \sigma_{share}$ then $Sh(d) = 0$ meaning that two solutions do not have any sharing effect on each other.

Fitness Sharing: Example

Bimodal function - six solutions and corresponding shared fitness functions

- $\sigma_{share} = 0.5$, $\alpha = 1$.

Sol. i	String	Decoded value	$x^{(i)}$	f_i	nc_i	f'_i
1	110100	52	1.651	0.890	2.856	0.312
2	101100	44	1.397	0.948	3.160	0.300
3	011101	29	0.921	0.246	1.048	0.235
4	001011	11	0.349	0.890	1.000	0.890
5	110000	48	1.524	0.997	3.364	0.296
6	101110	46	1.460	0.992	3.364	0.295



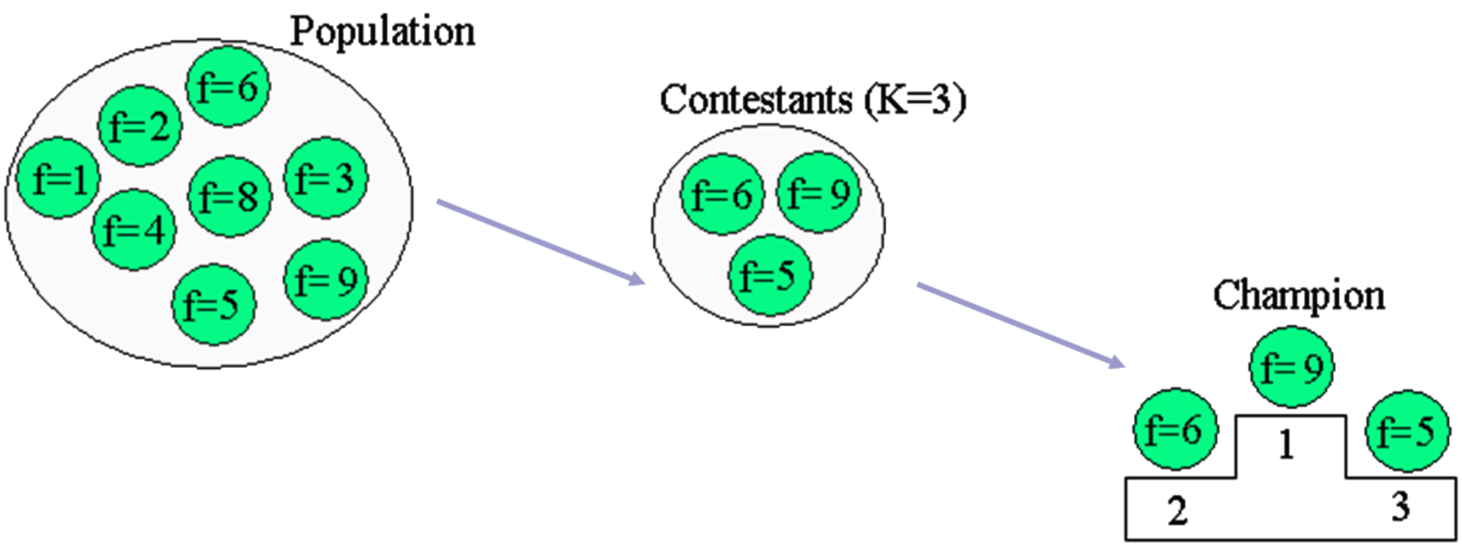
Let's calculate the shared fitness value of the first solution

- $d_{11} = 0.0$, $d_{12} = 0.254$, $d_{13} = 0.731$, $d_{14} = 1.302$, $d_{15} = 0.127$, $d_{16} = 0.191$
- $Sh(d_{11}) = 1$, $Sh(d_{12}) = 0.492$, $Sh(d_{13}) = 0$, $Sh(d_{14}) = 0$, $Sh(d_{15}) = 0.746$, $Sh(d_{16}) = 0.618$.
- $nc_1 = 1 + 0.492 + 0 + 0 + 0.746 + 0.618 = 2.856$
- $f'(1) = f(1)/nc_1 = 0.890/2.856 = 0.312$

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.



Genetic Operators: Crossover

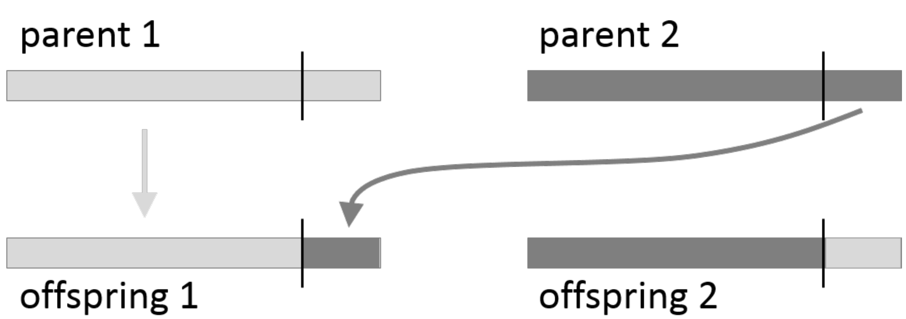
Role of crossover

- sampling (exploration) of the search space

Idea

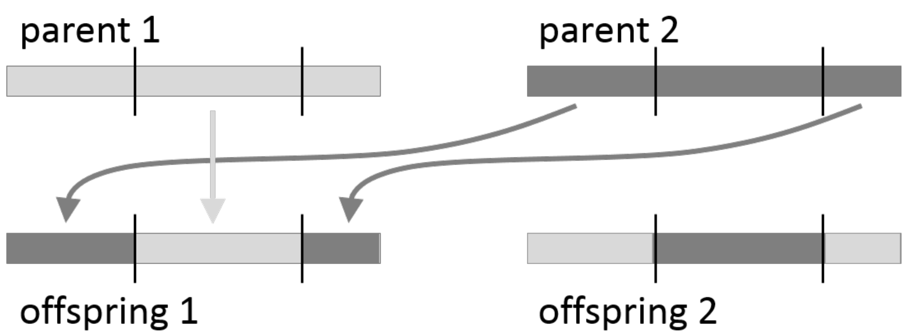
- given two well-fit solutions to the given problem, it is possible to get a new solution by properly mixing the two that is even better than both its parents.

1-point crossover



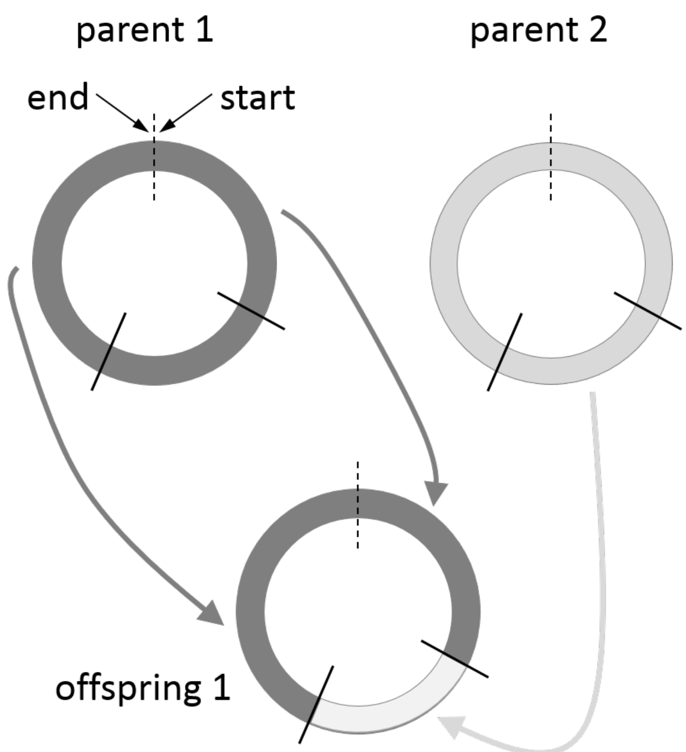
- mixing of two parents determined by a **single crossing point** chosen randomly for each pair of parents

Genetic Operators: 2-point crossover



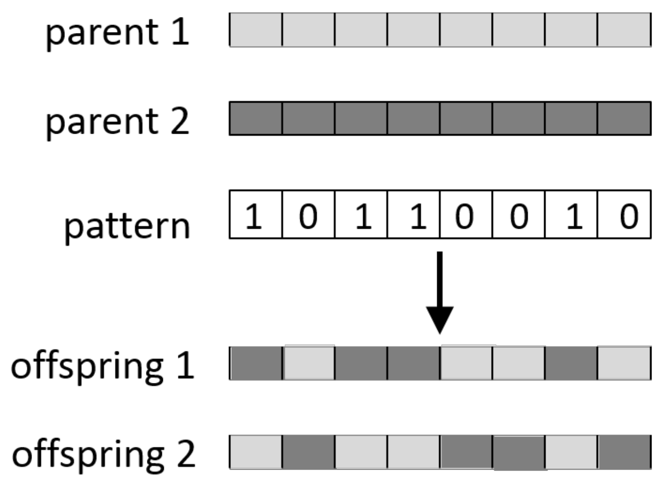
- mixing of two parents determined by **two crossing points** chosen randomly for each pair of parents

Genetic Operators: 2-point crossover



- mixing of two parents determined by **two crossing points** chosen randomly for each pair of parents
- higher exploration power than 1-point crossover

Genetic Operators: Uniform crossover



- mixing of two parents determined by an **n-point inheritance template** chosen randomly for each pair of parents
- highest exploration power

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.

$$\begin{array}{lcl} \text{parent1: } 4\ 8\ 1\ \mid\ 3\ 5\ 2\ 7\ 6 & & \text{offspring1: } 4\ 8\ 1\ \mid\ 1\ 3\ 6\ 5\ 4 \\ & \longrightarrow & \\ \text{parent2: } 2\ 7\ 8\ \mid\ 1\ 3\ 6\ 5\ 4 & & \text{offspring2: } 2\ 7\ 8\ \mid\ 3\ 5\ 2\ 7\ 6 \end{array}$$

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

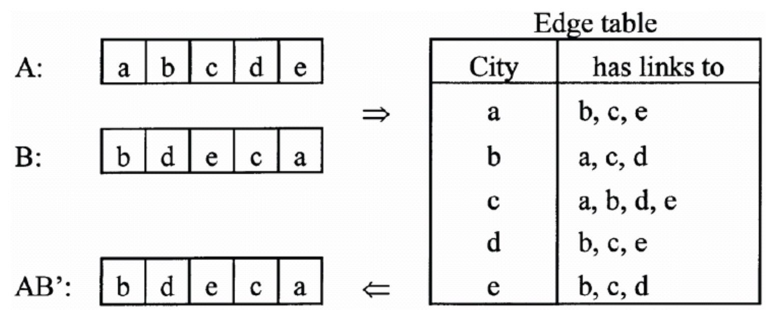
genotype: a e d b c

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

Edge recombination crossover

- 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 not yet in the created tour.

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

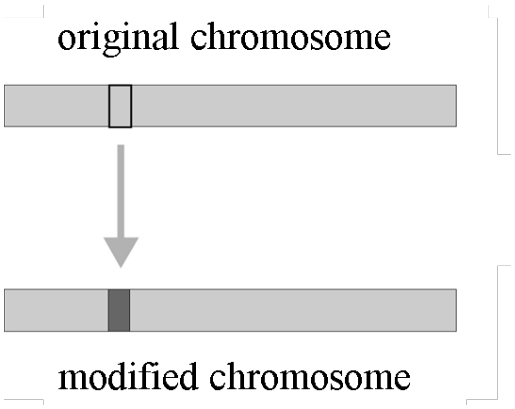
  Repeat this step until all cities have been added to the tour.

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 ← bestOf(oldPopulation)        // elitism
    while(newPopulation not full)
        parents ← select(oldPopulation)
        offspring ← crossover(parents)
        mutate(offspring)        // optional
        evaluate(offspring)
        newPopulation ← offspring
    swap(oldPopulation, newPopulation)
return bestOf(oldPopulation)
```

Steady-State Replacement Strategy

```
initialize(population)
evaluate(population)
while(not termination condition)
    parents ← select(population)
    offspring ← crossover(parents)
    mutate(offspring)            // optional
    evaluate(offspring)
    population ← offspring       // replacement rule
return bestOf(population)
```

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.

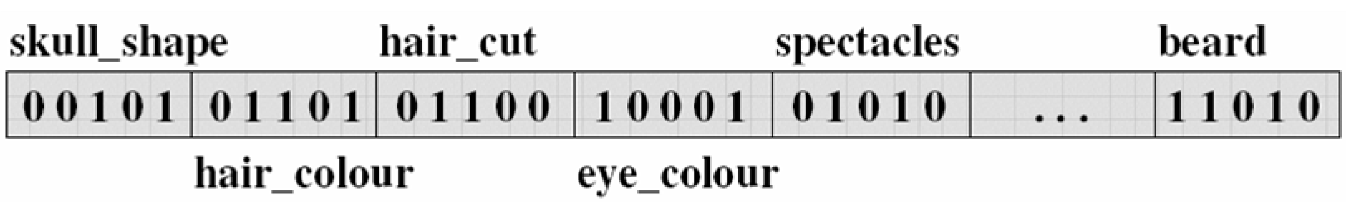
Tracking a criminal suspect

:: **Database of "criminals"**


. . .

:: **Goal** is to guide the witness through a huge database of faces in order to identify the criminal suspect.

:: **Chromosome structure**



:: **Human serves as an objective function evaluator**

? UK Home Office, Police Systems Research and Development Group ?

Job Shop Scheduling Problem

:: **Representation**: Pair-wise relative order of jobs on every machine

	machine sequence					
job1	3	1	2	4	6	5
job2	2	3	5	6	1	4
job3	3	4	6	1	2	5
job4	2	1	3	4	5	6
job5	3	2	5	6	1	4
job6	2	4	6	1	5	3

	processing times					
job1	3	6	1	7	6	3
job2	10	8	5	4	10	10
job3	9	1	5	4	7	8
job4	5	5	5	3	8	9
job5	3	3	9	1	5	4
job6	10	3	1	3	4	9

job1<job2: 1 1 0 1 0 0
job1<job3: 0 1 1 0 0 0
job1<job4: 1 1 0 0 1 0
job1<job5: 1 1 1 1 0 0
job1<job6: 1 1 0 0 0 0
job2<job3: 1 0 1 0 0 0
job2<job4: 1 1 1 1 0 0
job2<job5: 1 1 1 1 1 1
job2<job6: 1 1 1 0 0 0
job3<job4: 1 1 1 0 0 1
job3<job5: 1 1 1 1 0 0
job3<job6: 1 1 1 1 0 1
job4<job5: 1 1 0 1 0 0
job4<job6: 1 1 1 0 1 0
job5<job6: 1 0 1 0 0 0

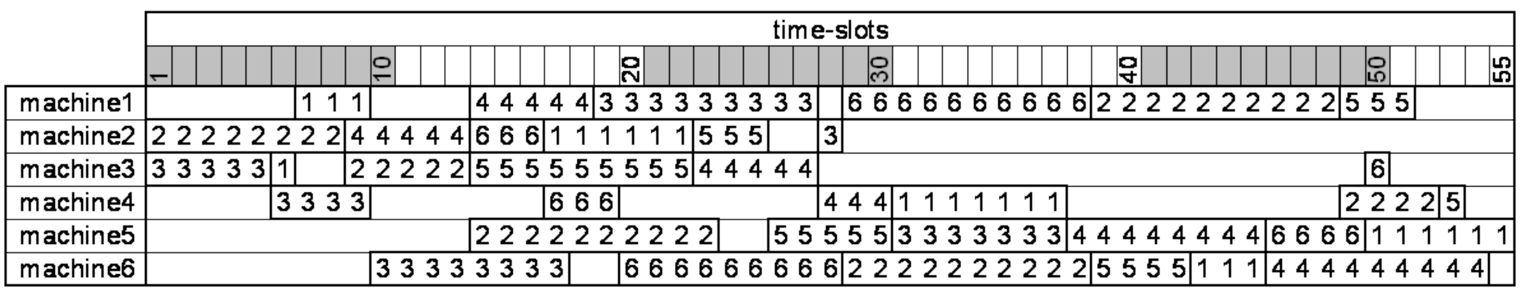
	job sequence					
machine1	1	4	3	6	2	5
machine2	2	4	6	1	5	3
machine3	3	1	2	5	4	6
machine4	3	6	4	1	2	5
machine5	2	5	3	4	6	1
machine6	3	6	2	5	1	4

Machines: M=6, Jobs: N=6

chromosome

110100 | 011000 | ... | 111010 | 101000

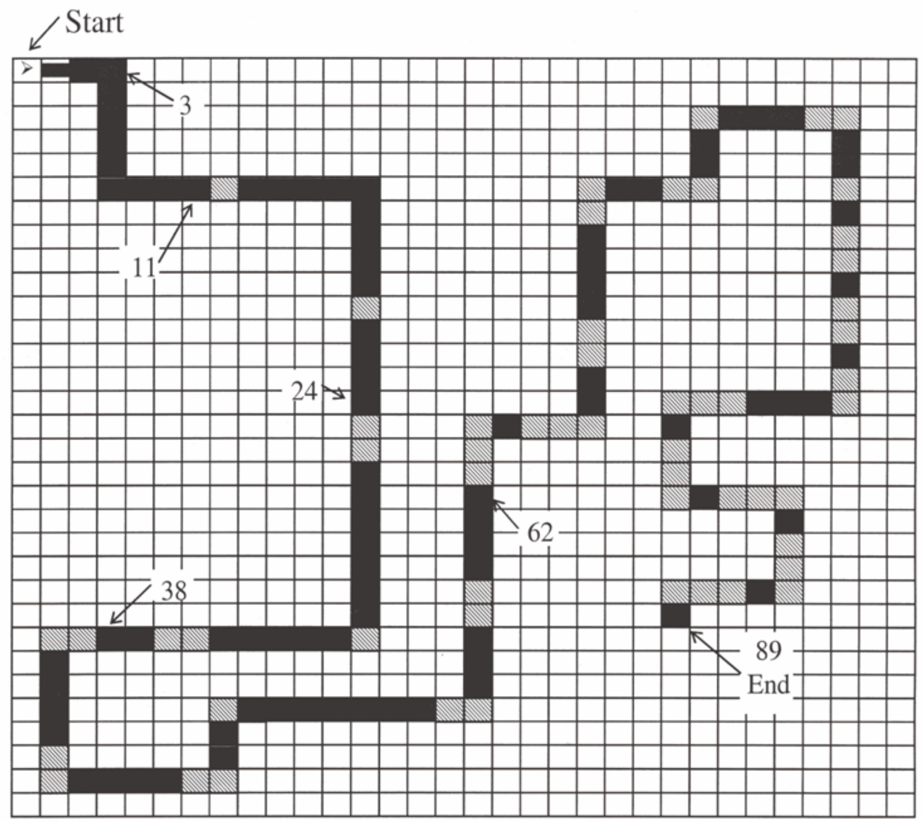
M*N*(N-1)/2 = 90 bits



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 $(2 + 4 \times 8)$

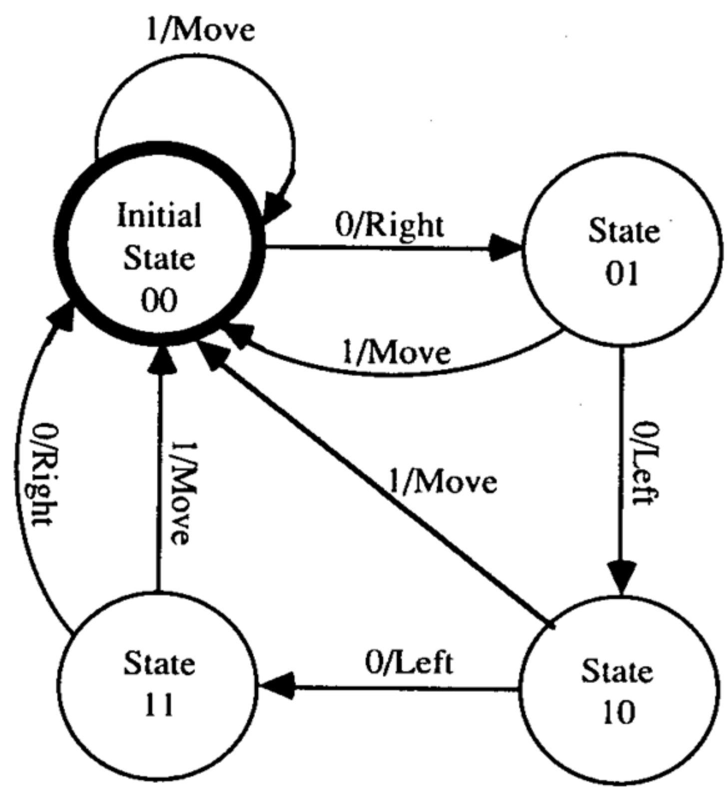
	Current state	Input	New state	Operation
1	00	0	01	10 = Right
2	00	1	00	11 = Move
3	01	0	10	01 = Left
4	01	1	00	11 = Move
5	10	0	11	01 = Left
6	10	1	00	11 = Move
7	11	0	00	10 = Right
8	11	1	00	11 = Move

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 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.
- **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
6 string of fitness 5
4 string of fitness 6
1 string of fitness 7

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

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 be the fitness of S ={*0*1****} compared 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

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

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

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

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

The formula can be rewritten as

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

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

Schema Theorem Derivation: Effect of Crossover and Mutation

Effect of 1-point Crossover

- Crossover probability p_c – fraction of population that undergoes 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.

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

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

Schema Theorem Derivation: Effect of Crossover and Mutation

Effect of 1-point Crossover

- Crossover probability p_c – fraction of population that undergoes 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.

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

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 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`