Genetic Algorithms

Tribhuvan University

Central Department of Computer Science

Compiled by

RK DAHAL

Table of Contents

References

- Davis, L. (ed.)
  Handbook of Genetic Algorithms.
  New York: Van Nostrand Reinhold, 1996
- Goldberg, David E.
  Genetic Algorithms in Search, Optimization, and Machine Learning.
  Reading, MA: Addison-Wesley, 1989
- Michalewicz, Zbigniew
  Genetic Algorithms + Data Structures = Evolution Programs.
  Berlin; Heidelberg; New York: Springer, 3rd edition 1997
- Falkenauer, Emanuel
  Genetic Algorithms and Grouping Problems.
  Chichester: John Wiley & Sons, 1998

History

Genetic Algorithms

John von Holland,
K. A. De Jong
(Univ. of Michigan, 1975)

Evolutionsstrategien

Ingo Rechenberg
Hans P. Schwefel
(Technical Univ. of Berlin, 1973)

Evolutionary Algorithms

(since 1985)

Part I
Optimisation Problems

Travelling Salesman Problem

The travelling salesman must visit every city in his territory exactly once and then return back to the starting point.

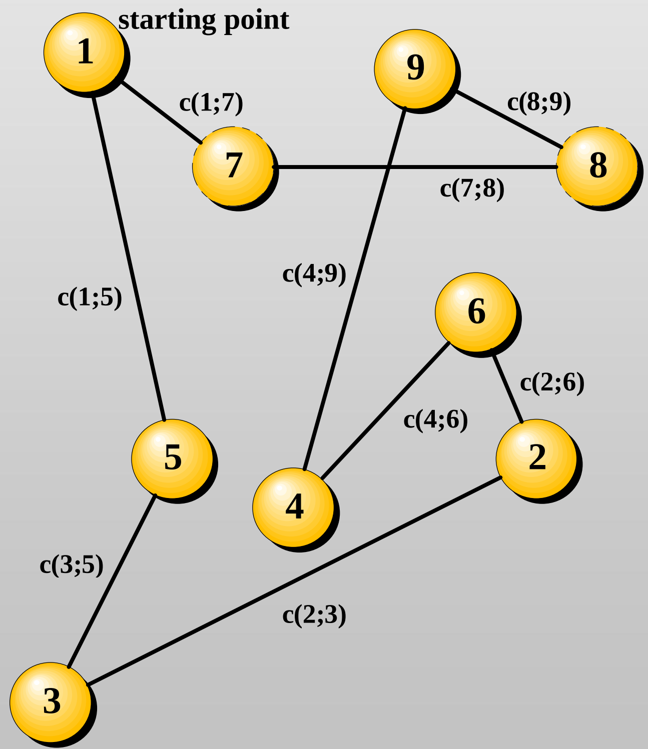
Given the cost of travel between all cities, how should he plan his itinerary for minimum total cost?

Total Cost

f(x) = c(1;7) + c(7;8) + ... + c(1;5)

Search Space

Given n cities, there are ½·(n-1)! different routes.



Combinatorial Optimisation

Bin Packing Problem	optimal allocation of items to 'bins', minimal number of bins
Vehicle Routeing Problem	optimal routeing, minimal number of vehicles and idle runs
Task Allocation Problem	optimal allocation of tasks to parallel processors, minimal execution time
Job-Shop Scheduling Problem	maximal machine utilisation, minimal service time, minimal cost
VLSI Design Problem	optimal placing of logic modules, minimal wire-length

Search Space & Objective Function

- The search space S is the *finite* set of possible solutions.
- Each solution $x \in S$ is also called an individual.

- An objective function

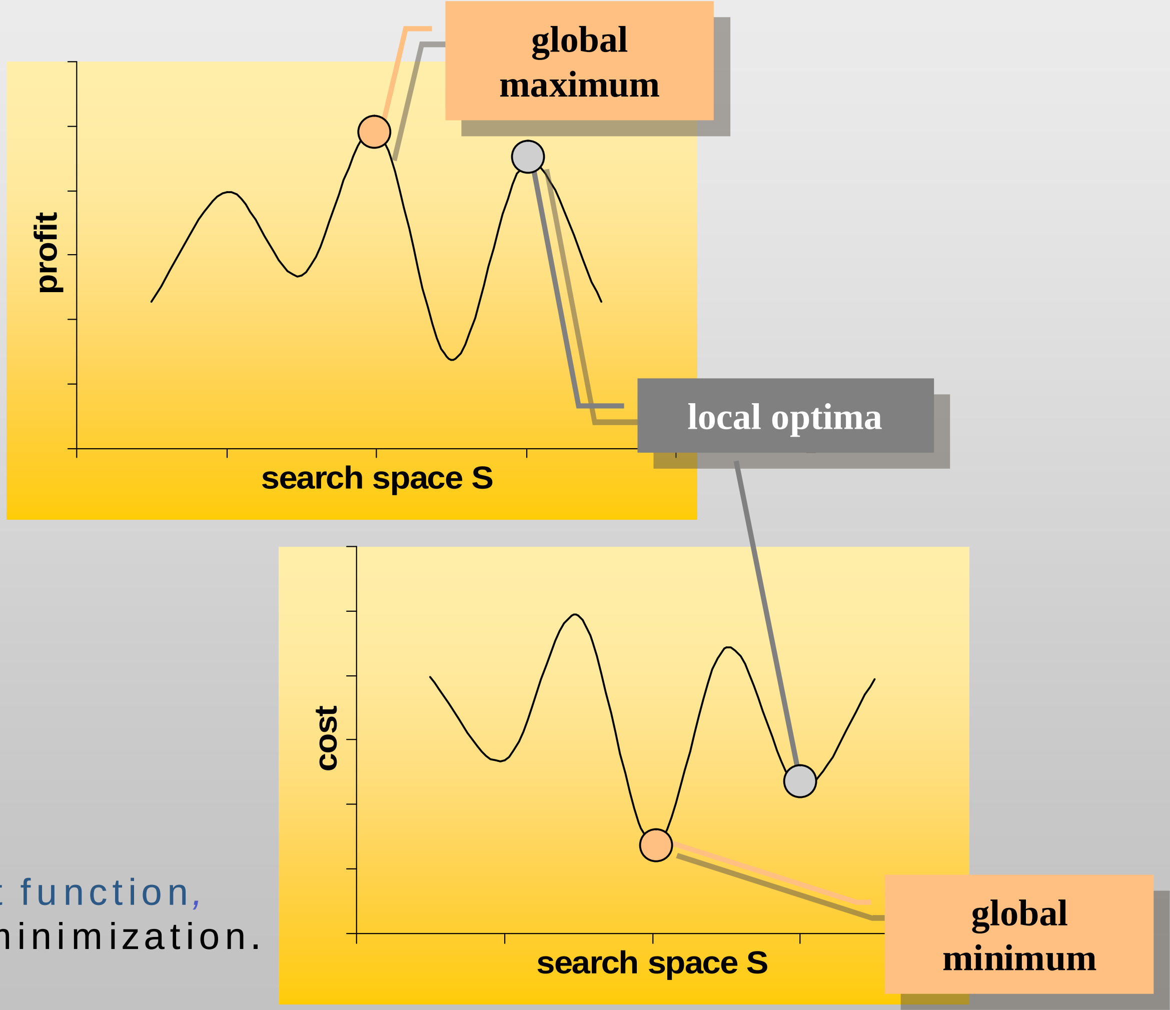
$$f: S \rightarrow \mathrm{IR}$$

  must be defined.
- The objective is to maximize or minimize this function.

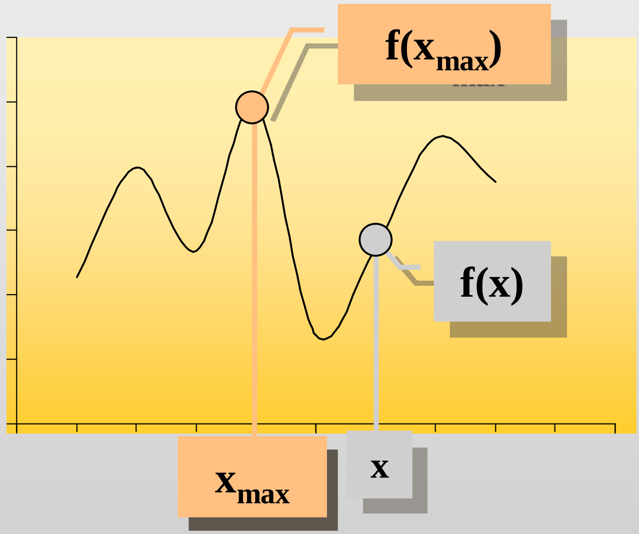
Objective Function

The objective function f
is also called
profit function,
in case of maximization.

...or cost function,
in case of minimization.



Fitness Function



In what follows,
we usually assume that ...

- ... the objective function has been mapped to a *nonnegative* function f.

  In that case f is called fitness function,
  and the value f(x) is the fitness of the individual $x \in S$.

- .... we shall focus upon maximizing f,
  i. e. searching for an individual $x_{max} \in S$ with maximal fitness.

Hard Problems

- Combinatorial Explosion

  The search space S is finite, but extremely large.

- The optimisation problem is NP-hard,

  i.e. the time required to solve it is expected to increase *exponentially* with the size of the problem.

As a consequence,
an *exhaustive* search for the optimal solution cannot be performed within reasonable time.

Heuristics, Probabilistic Operators

- Given a hard optimisation problem, it is often possible to find an efficient heuristic approach.
- For some hard problems probabilistic operators, which use random choices as a tool to guide the search, can be applied as well.

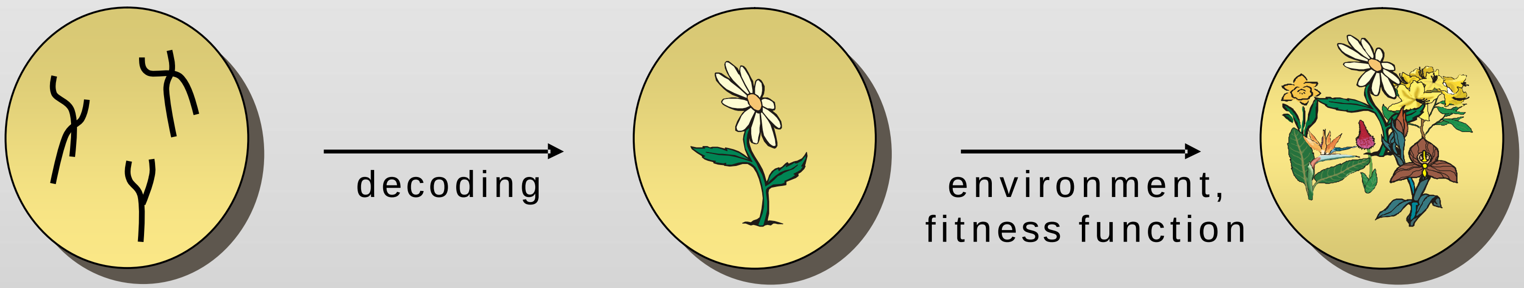
Such algorithms do *not* guarantee to find an optimal solution.

They may only find near optimal solutions.
These, however, may be acceptable for most applications.

Part II
How Do Genetic Algorithms Work?

The Paradigm

genotype | phenotype | fitness

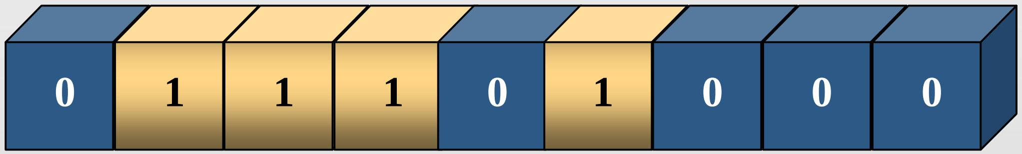


chromosomes, strings (encoded solutions)

problem solutions, individuals (decoded solutions)

value of the solution

Binary Coding



binary string (chromosome) composed of genes

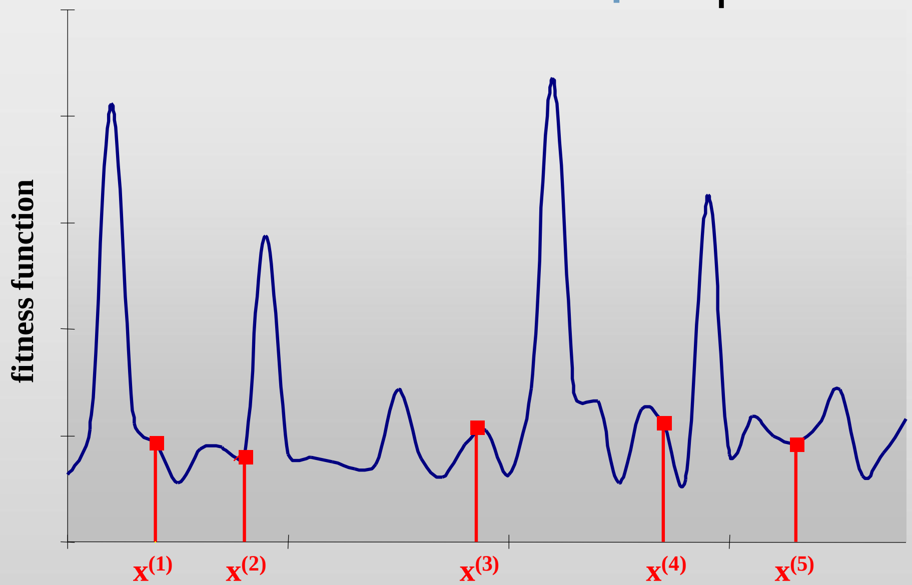
- In classic genetic algorithms, binary strings of fixed length m are used.

- In order to be able to encode each solution of the search space S in a one-to-one way, the inequality

$$2^m \geq \mathrm{card}(S)$$

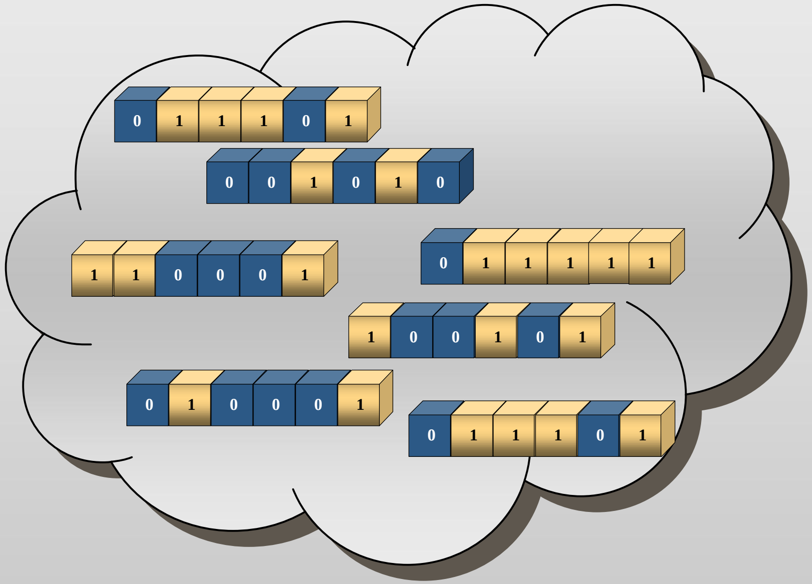
must hold.

Population



- Genetic algorithms work from a population P, i. e. a series of chromosomes.
- The initial population P(0), the first generation, is created randomly.
- In an iterative process, populations P(t) at generation t, (t = 1,2,...) are constituted.

Population Size

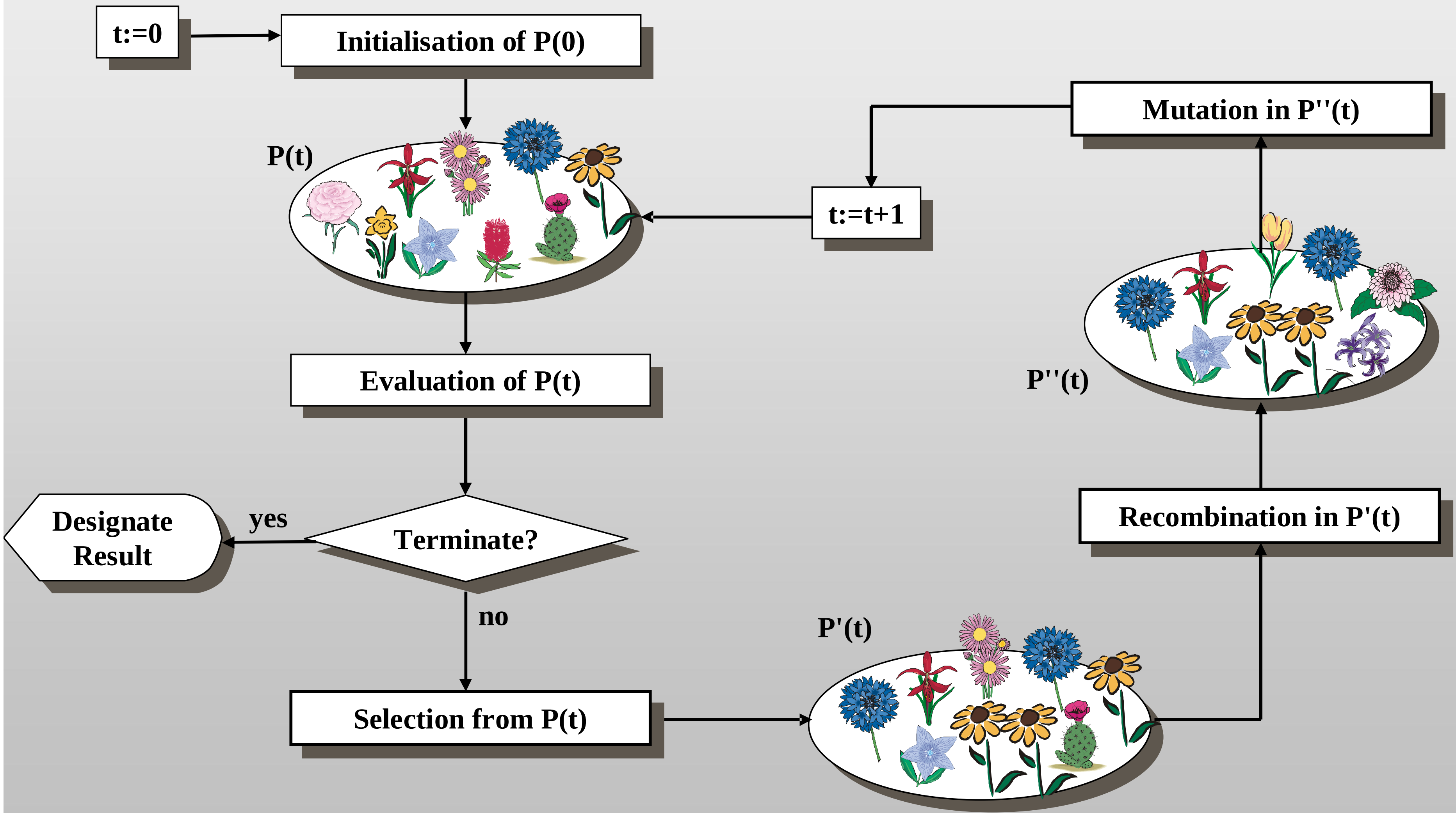


The constant population size is one of the parameters of a genetic algorithm.

Typical values are N=20, N=50, N=100, ...

In classic genetic algorithms, the population size N remains unchanged from one generation to the next.

Flowchart


t:=0
Initialisation of P(0)
P(t)
Mutation in P''(t)
t:=t+1
Evaluation of P(t)
P''(t)
Designate Result
yes
Terminate?
Recombination in P'(t)
no
P'(t)
Selection from P(t)

The Genetic Operators

- Reproduction (Selection)

  Copy existing chromosomes, chosen at random, to the new population.

- Recombination (Crossover)

  Create new chromosomes by recombining randomly chosen substrings from existing chromosomes.

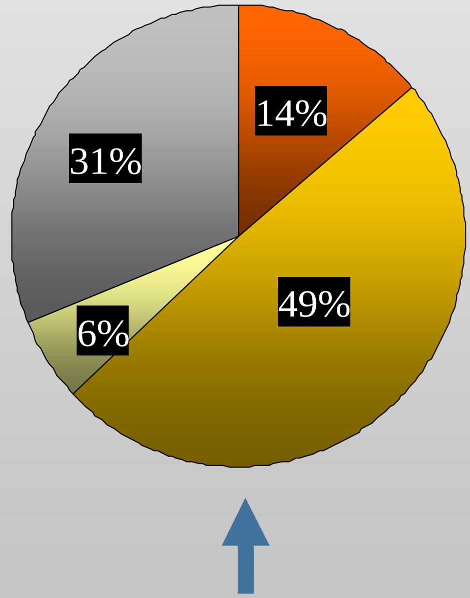
- Mutation

  Create a new chromosome from an existing one by performing small random changes.

Roulette Wheel Selection (fitness-proportional selection; stochastic sampling with replacement) is an instance of a reproduction operator:

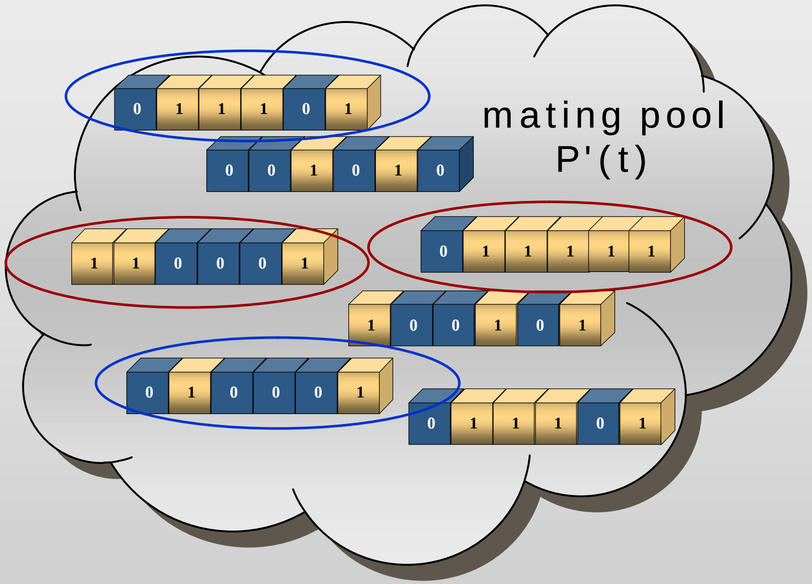
sample population

No. i	fitness f_i	probability p_i
1	169	169/1170 = 0.14
2	576	576/1170 = 0.49
3	64	64/1170 = 0.06
4	361	361/1170 = 0.31
Total	1170	1,00



spin the weighted roulette wheel N times

Recombination



The cross-over probability p_c is another parameter of the genetic algorithm.

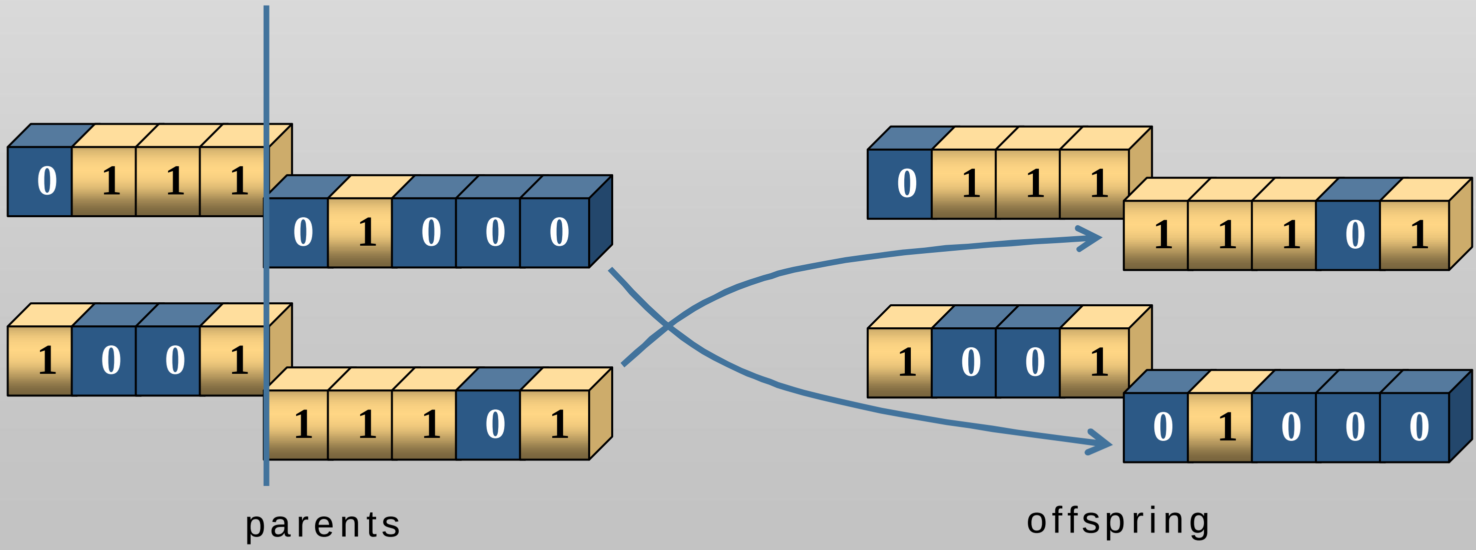
Typical values are between 60% and 90%.

- After reproduction, a specified percentage p_c of chromosomes in the mating pool P'(t) is chosen at random.
- The selected chromosomes are mated at random, and each pair of parents undergoes a crossover operation, such as the following one ....

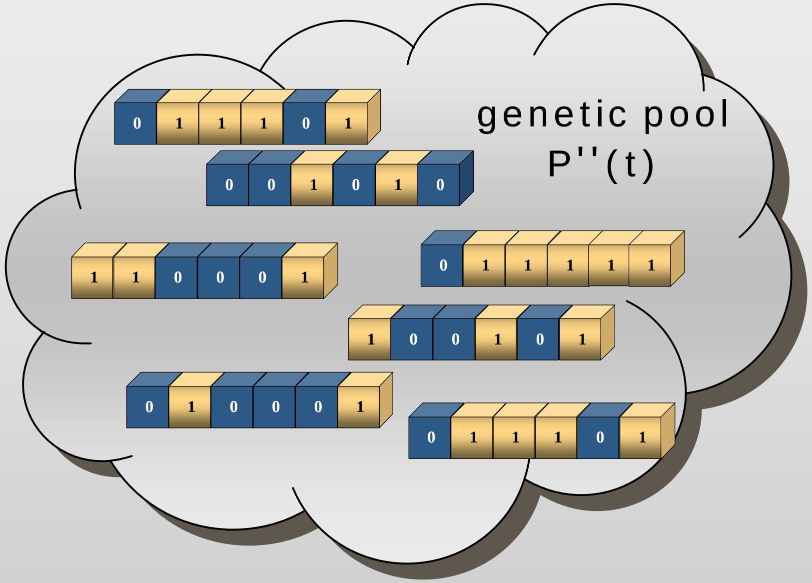
Recombination

A classical recombination operator is the One-point Crossover:

- Randomly select a cross-over site between 1 and m-1.
- Each parent is then split at this point into two fragments.
- Offspring are obtained by joining the non-corresponding fragments of each parent.



Mutation



The mutation probability p_m is another parameter of the genetic algorithm.

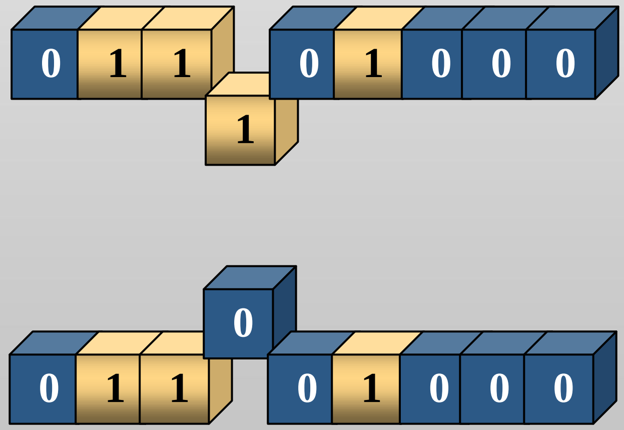
Typical values are below 1%.

- After recombination, a specified percentage p_m of genes in the pool P''(t) is chosen at random.
- A selected parent chromosome undergoes a mutation operation, such as the following one ...

Mutation

The classical mutation operator is the Bit-flip Mutation:

• The value of the selected gene is simply inverted.



parent

offspring

Steady State vs Classical GAs

Generational Replacement

In classical genetic algorithms,
the progeny
obtained by crossover or mutation
replaces the parent chromosomes.

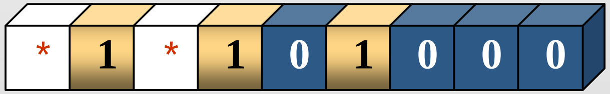
Steady State Model

In each generation,
just a few chromosomes,
namely the worst ones,
are replaced by the progeny.

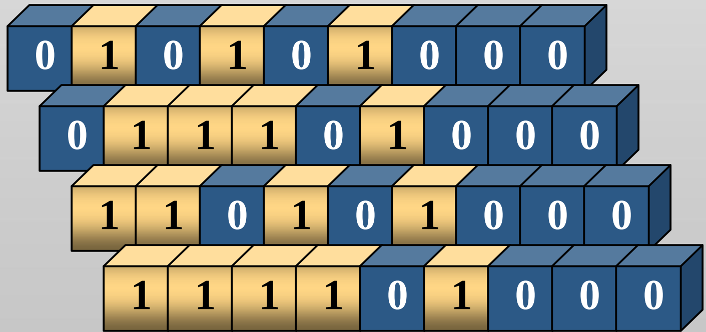
Part III
Mathematical Foundations

Schemata

The schema

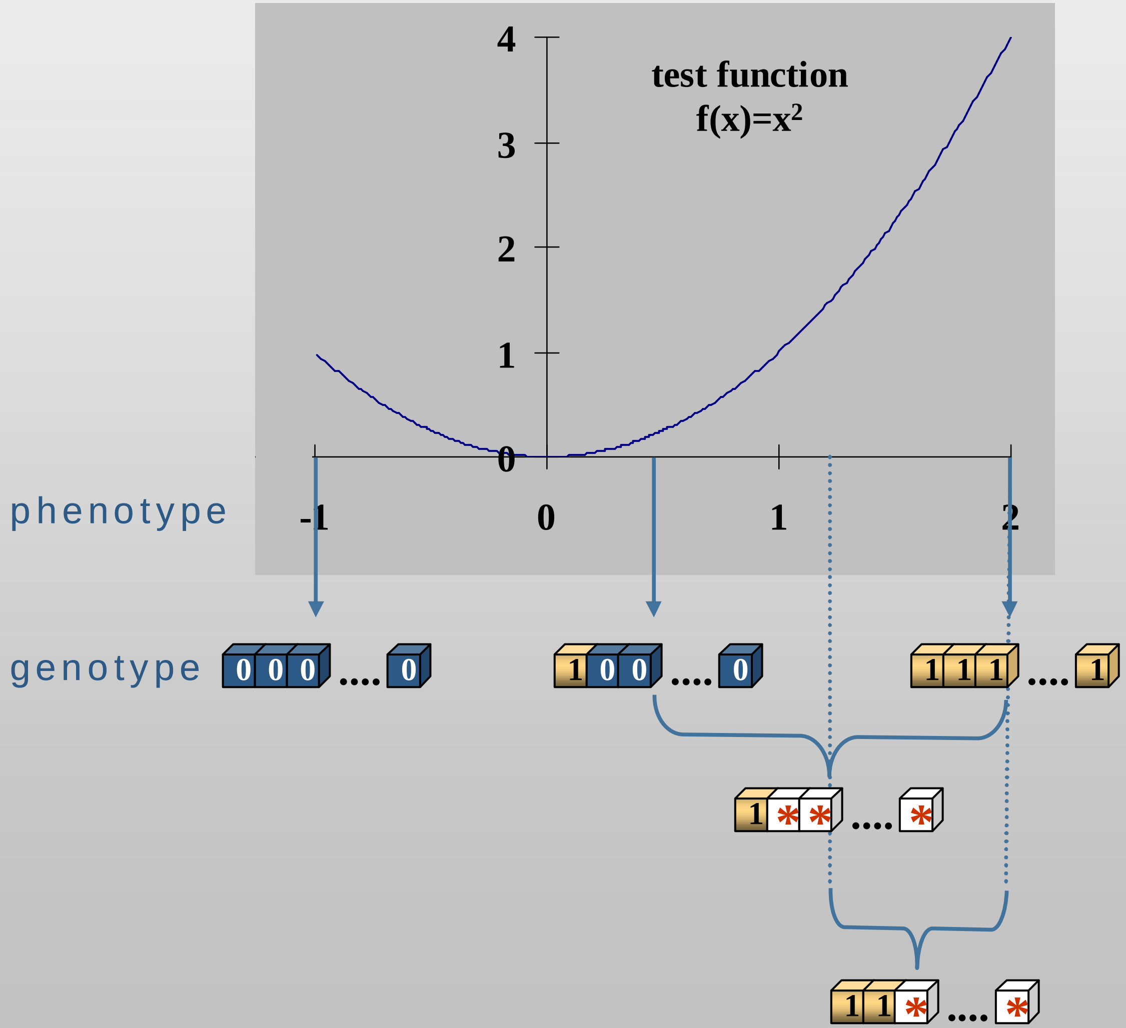


is a similarity template
describing the following four similar binary strings:



- The don’t care symbol * matches either a 0 or a 1 at its particular position.
- A schema with k don’t care symbols represents a set of 2^k binary strings.

Search for Promising Patterns



Above-Average Schemata

S: * * * * 0 1 0 * *

Suppose, schema S represents good solutions.

population P(t)	fitness values	
0 1 0 1 0 1 0 0 0	40	40
0 1 1 1 0 1 0 1 0	30	30
1 1 0 1 0 0 0 0 0	20	
1 1 0 1 0 0 0 0 0	20	
1 0 1 1 0 1 1 0 0	30	
1 1 1 1 0 1 0 0 0	50	50
1 1 0 1 0 0 0 0 0	20	

average fitness $\bar{f}(t) = \frac{210}{7} = 30$ (whole population) $\bar{f}_S(t) = \frac{120}{3} = 40$ (strings matched by schema S)

$$\boxed{\frac{\bar{f}_S(t)}{\bar{f}(t)} = \frac{4}{3} > 1}$$

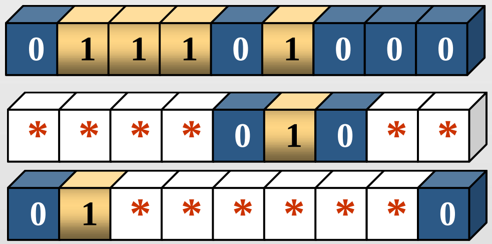
Schema S is above average.

Schemata Surviving Crossover

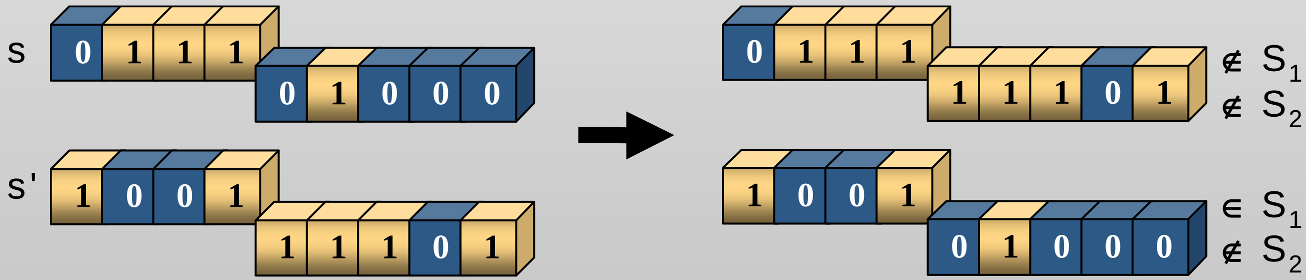
Binary string s

is matched by schema S_1

... as well as schema S_2



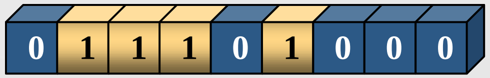
Schema S_1 survives the following one-point crossover.
Schema S_2 , however, would be destroyed!



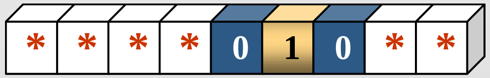
Note:
In the example, the defining length of schema S_1,
i.e. the distance between its first and its last fixed position,
is relatively small: $\delta(S_1) = 2$. But $\delta(S_2) = 8$ is maximal!

Schemata Surviving Mutation

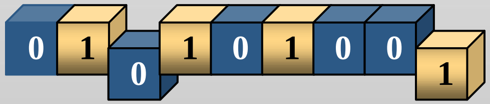
Binary string s



is matched by schemas S:



Schema S survives bit-flip mutations, if and only if the mutated genes do not belong to the fixed positions of schema S:



∈ S

Note:
In the example, the order of schema S,
i.e. its number of fixed positions,
is relatively small: $o(S) = 3$.

The Schema Theorem

John H. Holland proved this theorem for classic genetic algorithms, i.e. for GAs using

- roulette wheel selection,
- one-point crossover
- and bit-flip mutation.

$$\xi_S(t+1) \geq \xi_S(t) \cdot \frac{\bar{f}_S(t)}{\bar{f}(t)} \cdot \left(1 - p_c \cdot \frac{\delta(S)}{m-1} - o(S) \cdot p_m\right)$$

Schemata S with
above-average fitness,
short defining length
and low order,
receive an increasing expected number ξ_S of copies
in subsequent generations.

Building Block Hypothesis

"(.....) instead of building high-performance strings
by trying every conceivable combination,
we construct better and better strings
from the best partial solutions of past samplings." *)

"Just as a child creates magnificent fortresses
through the arrangement of simple blocks of wood,
so does a genetic algorithm seek near optimal performance
through the juxtaposition of ..... building blocks." *)

Building Blocks:
short, low-order, above-average schemata

*) Goldberg 1989, p.41

The Principal of Minimal Alphabets

> “Select the smallest alphabet that permits a natural expression of the problem.” *)

*) Goldberg 1989, p.80

Binary vs Nonbinary Codings

binary	decimal x	alphabet of cardinality 26	fitness $f(x)=x^2$
01101	13	N	169
11000	24	Y	576
01000	08	I	64
10011	19	T	361

Schema 11***
leads to good solutions

no coding similarities
to exploit

The binary alphabet offers
the maximum number of schemata
per bit of information of any coding.

"Natural" Data Structures

Nevertheless, ...

"Select the smallest alphabet
that permits a *natural* expression of the problem."
Goldberg 1989

"What should one do
when elements in the space to be searched
are most naturally represented by more complex data structures
such as arrays, trees, digraphs, etc.
Should one attempt to 'linearize' them into a string
representation ..."
De Jong 1985

"Adapt the Genetic Operators.
Create crossover and mutation operators for the new type of encoding
by analogy with bit string crossover and mutation operators.
Incorporate domain-based heuristics as operators as well. "
Davis 1991

Part IV
The Travelling Salesman Problem

Travelling Salesman Problem

The travelling salesman must visit every city in his territory exactly once and then return back to the starting point.

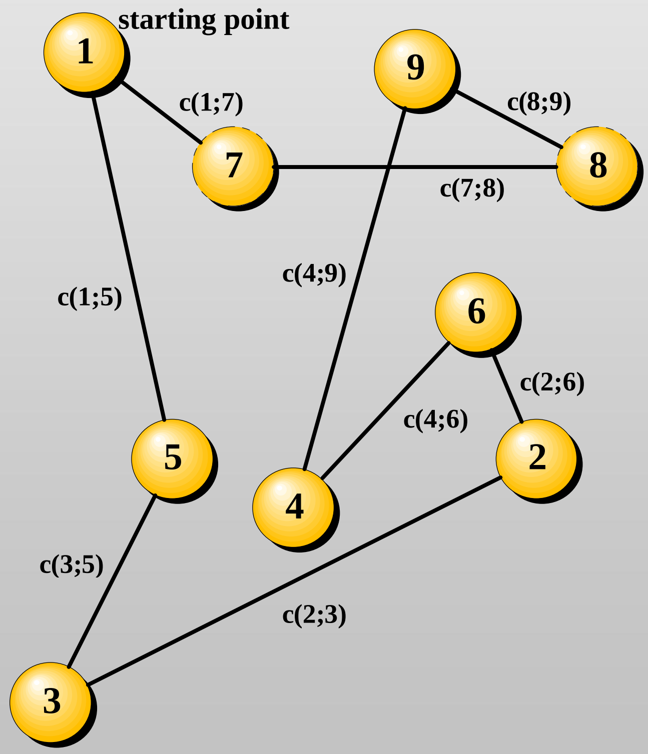
Given the cost of travel between all cities, how should he plan his itinerary for minimum total cost?

Total Cost

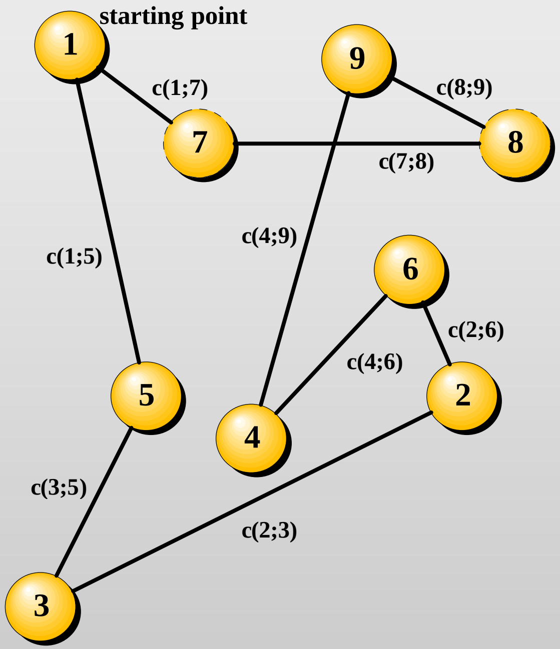
f(x) = c(1;7) + c(7;8) + ... + c(1;5)

Search Space

Given n cities, there are ½·(n-1)! different routes.

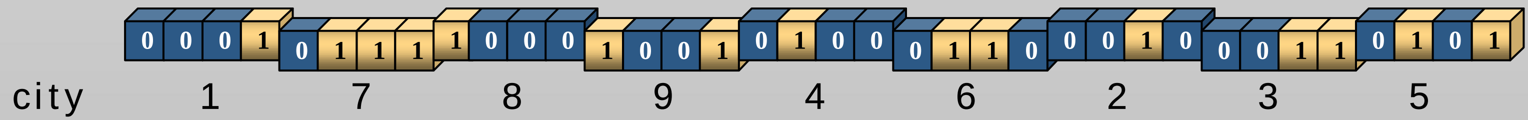


Binary Representation

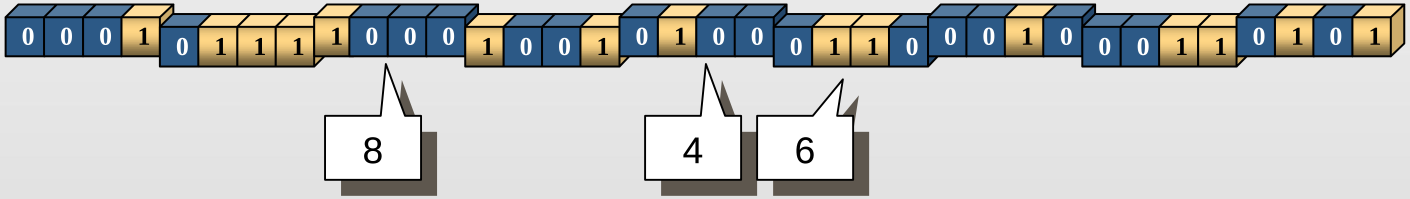


The routes could be represented as binary strings of length

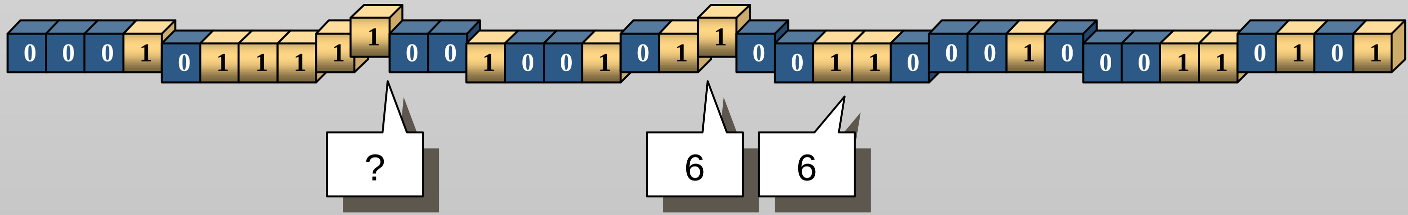
$$n \cdot \lceil \log_2 n \rceil = 9 \cdot \lceil \log_2 9 \rceil = 9 \cdot 4 = 36$$



Problems with this Coding



One-point crossover or bitflip mutation
can produce
illegal chromosomes :



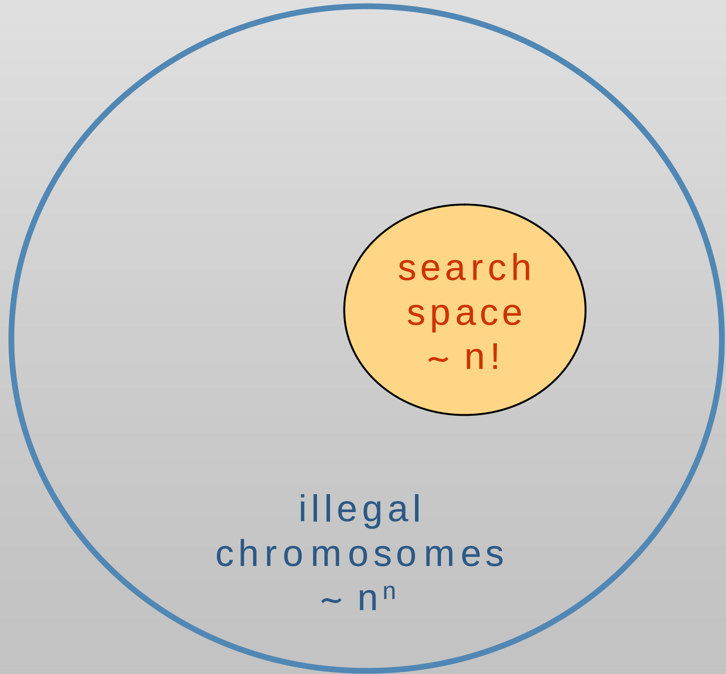
City no 12
does not exist!

City no 6
appears twice!

Constraints that must not be violated (hard constraints)
can be implemented by imposing penalties
on individuals that violate them.

Disadvantages

- In heavily constraint problems, one runs the risk of spending most time evaluating illegal chromosomes.
- If *high penalties* are imposed, premature convergence to legal but mediocre chromosomes is possible.
- If the penalties are *too moderate*, the GA may evolve illegal chromosomes that are rated better than those that do not violate the constraints.

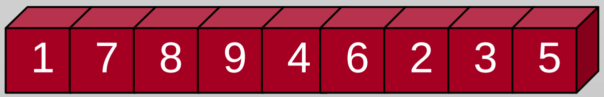


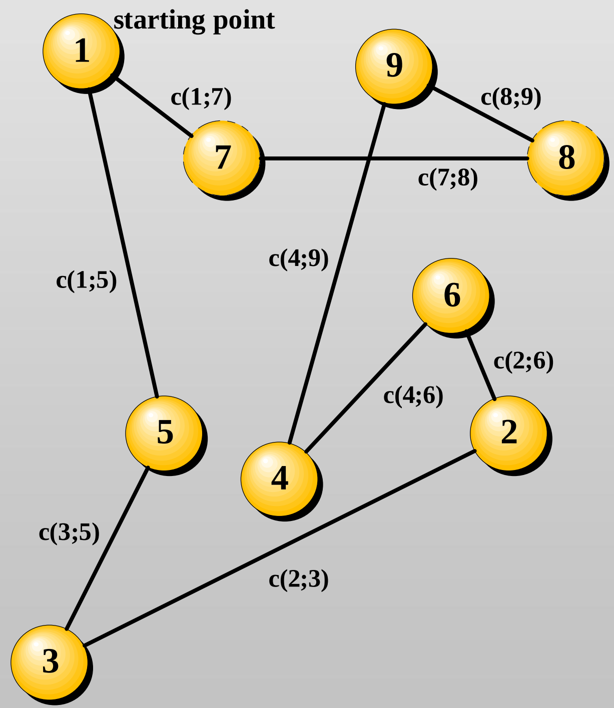
Nonbinary Representation

Alternative:

- Look for the most natural expression of the problem.
- Create genetic operators that avoid building illegal chromosomes.

Path Representation

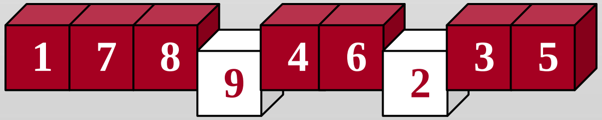




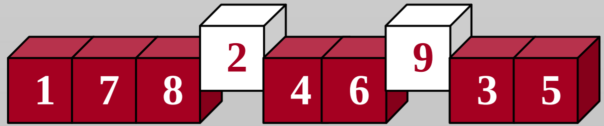
Swap Mutation

The following mutation operator
is adapted to the path representation:

Select two cities at random ...

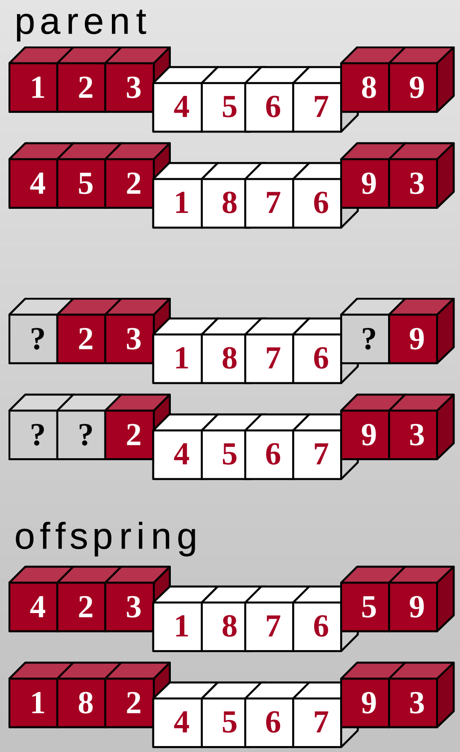


.... and swap their positions.



PMX-Crossover

Also Partially Matched Crossover (PMX)
avoids building illegal chromosomes:



Select 2 crossing points at random.

Swap the segments between the 2 points ("matching section").

Fill further cities for which there is no conflict.

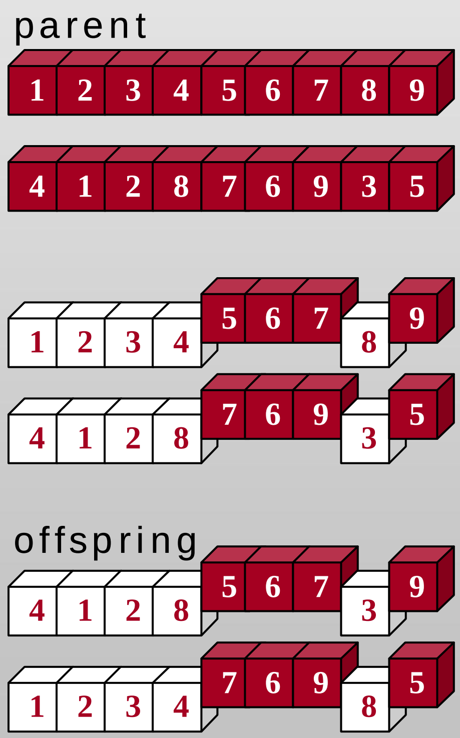
The matching section defines the mappings

$1 \leftrightarrow 4$, $8 \leftrightarrow 5$, $7 \leftrightarrow 6$ und $6 \leftrightarrow 7$.

Fill the ?-gaps accordingly.

CX-Crossover

And also Cycle Crossover (CX) is adapted to the path representation and produces only valid chromosomes.



The parents define a one-to-one mapping (permutation):

$1 \rightarrow 4, \; 2 \rightarrow 1, \; 3 \rightarrow 2 \; \ldots.$

Select a city of the first parent at random, for instance city 4. The sequence

$4 \rightarrow 8, \; 8 \rightarrow 3, \; 3 \rightarrow 2, \; 2 \rightarrow 1, \; 1 \rightarrow 4$

is a cycle of this permutation.

Swap the segments belonging to this cycle. Leave the other cities unchanged.