U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence/ Inteligência Artificial

Lecture 3c: Optimization and Genetic Algorithms

(based on Løkketangen, 2019)

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence

Local Search

- Basic Idea: Improve the current solution
- Start with some solution
- Find a set of solutions (called neighbors) that are ”close” to the current solution
- If one of these neighbors is better than the current solution, move to that solution
- Repeat until no improvements can be made

Local Search (2)

- **Variations**
  - Best Improvement (always select the best neighbor)
  - First Improvement (select the first improving neighbor)
  - Random Descent (select neighbors at random)
  - Random Walk (move to neighbors at random)
- **Problem: Gets stuck in a local optimum**
  - Eexcept for Random Walk, which isn't a good method anyway...

Local Search Based Metaheuristics (1)

- **Main goal**
  - To avoid getting stuck in local optima
- **Additional goals**
  - Explore a larger part of the search space
  - Attempt to find the global (not just a local) optimum
  - Give an alternative to exact methods (especially for large/hard problem instances, and where the solution time is important)

Local Search Based Metaheuristics (2)

- **Different methods employ very different techniques in order to escape local optima**
  - Simulated Annealing -> controlled random movement
  - Tabu Search -> memory structures, recording enough information to prevent looping between solutions
- **Different methods employ very different techniques in order explore a larger part of the search space**
  - Simulated Annealing -> controlled random movement
  - Tabu Search -> memory structures, recording enough information to guide the search to different areas of the search space (e.g., frequency based diversification)

Local Search Based Metaheuristics (3)

- **Which method is better?**
- **Depends on your needs**
  - SA is easier to implement
  - SA is easier to use/understand
  - TS is more flexible and robust
  - TS requires a better understanding of the problem
  - TS requires more ”tuning”
  - TS produces typically better overall results

Genetic Algorithms (GAs)

- Not based on the idea of Local Search
- Directed search algorithms based on the mechanics of biological evolution
- Developed by John Holland, University of Michigan (1970's)
  - To understand the adaptive processes of natural systems
  - To design artificial systems software that retains the robustness of natural systems
- Provide efficient, effective techniques for optimization and machine learning applications
- Widely-used today in business, scientific and engineering circles

GAs - Analogies with Biology

- Applications to Function Optimization and AI (Games, Pattern Recognition ...)
- Basic idea: intelligent exploration of the search space based on random search
- Representation of complex objects by a vector of simple components
- Analogies with Biology:
  - Chromosomes
  - Selective breeding
  - Darwinistic evolution
- Classical GAs: Binary encoding

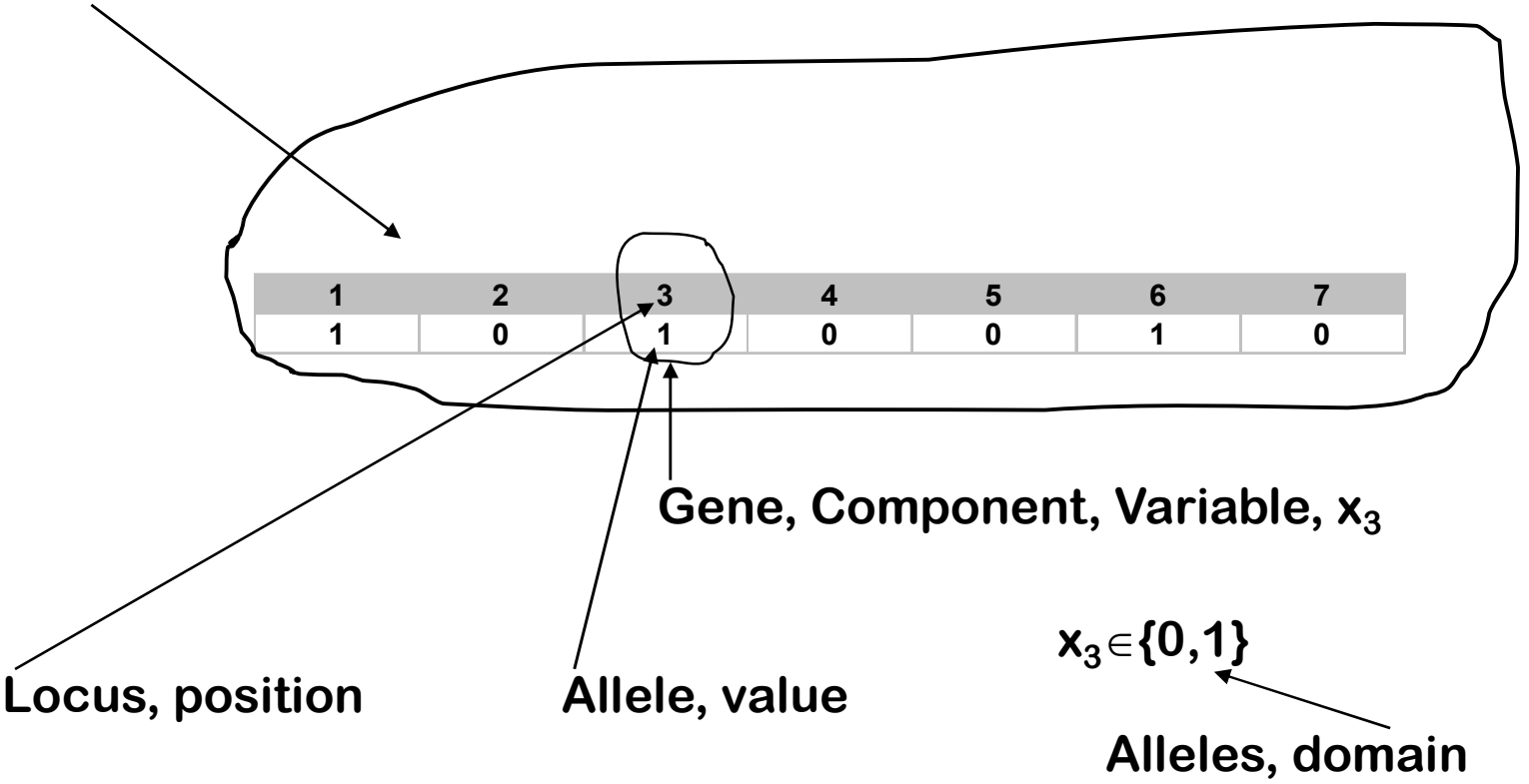
Components of a GA

A problem to solve, and ...

- **Encoding technique** (***gene, chromosome***)
- **Initialization procedure** (***creation)***
- **Evaluation function** ***(environment)***
- **Selection of parents** ***(reproduction)***
- **Genetic operators** ***(mutation, recombination)***
- **Parameter settings** ***(practice and art)***

Classical GA: Binary Chromosomes

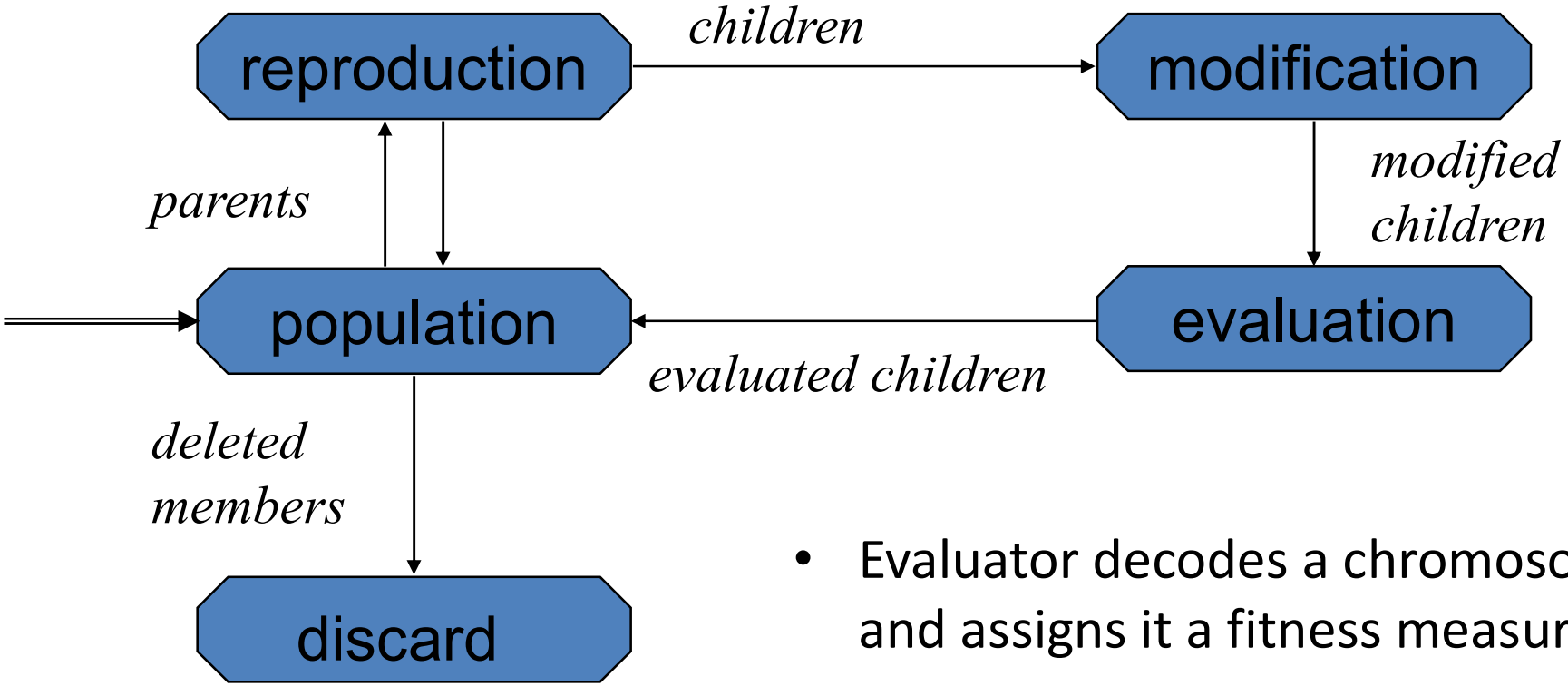
Chromosome, component vector, vector, string, solution, individual $x=(x_1, \ldots, x_7)$



Genotype, Phenotype, Population

- **Genotype**
  - chromosome
  - Coding of chromosomes
  - coded string, set of coded strings
- **Phenotype**
  - The physical expression
  - Properties of a set of solutions
- **Population – a set of solutions**

Gas Cycle of Reproduction



- Evaluator decodes a chromosome and assigns it a fitness measure
- Only link between a classical GA and the problem it is solving

Evaluation of Individuals

- **Adaptability – ”fitness”**
- **Relates to the objective function value for the problem**
- **Fitness is maximized**
- **Used in selection (”*Survival of the fittest*”)**
- **Often *normalized:*** $f : S \rightarrow [0,1]$

Genetic Operators and Evolution

Genetic Operators:

- Manipulates chromosomes/solutions
- Mutation: Unary operator - Inversions
- Crossover: Binary operator

GAs Evolution:

- **N generations of populations with fixed size M**
- **For every step in the evolution**
  - Selection of individuals for genetic operations
  - Creation of new individuals (reproduction)
  - Mutation
  - Selection of individuals to survive

Genetic Algorithm

```
1: Choose an initial population of chromosomes
2: while stopping criterion not met do
3:    while sufficient offspring has not been created do
4:       if condition for crossover is satisfied then
5:          Select parent chromosomes
6:          Choose crossover parameters
7:          Perform crossover
8:       end if
9:       if condition for mutation is satisfied then
10:         Choose mutation points
11:         Perform mutation
12:      end if
13:      Evaluate fitness of offspring
14:   end while
15: end while
```

Gas - Genetic Algorithms

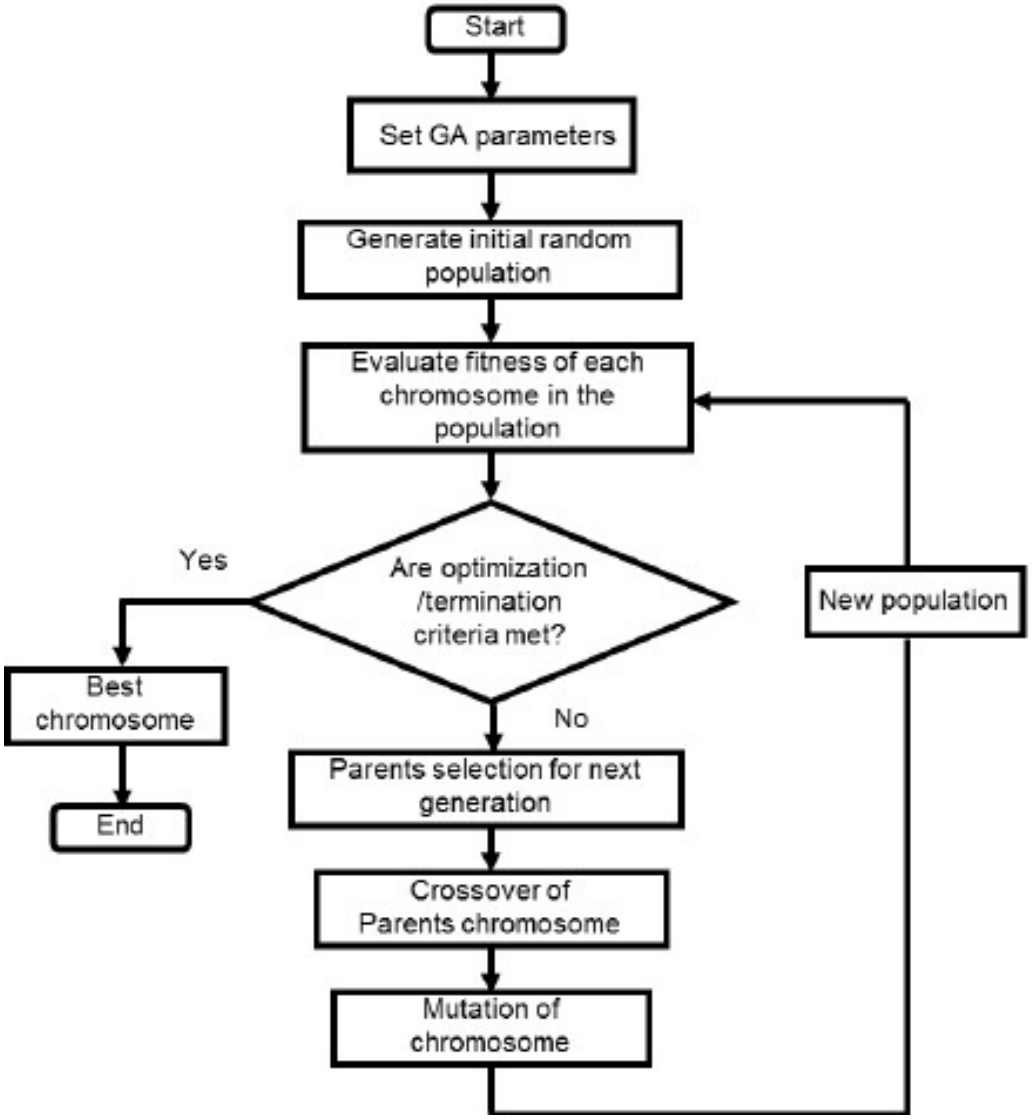
```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual

  repeat
    new_population ← empty set
    for i = 1 to SIZE(population) do
      x ← RANDOM-SELECTION(population, FITNESS-FN)
      y ← RANDOM-SELECTION(population, FITNESS-FN)
      child ← REPRODUCE(x, y)
      if (small random probability) then child ← MUTATE(child)
      add child to new_population
    population ← new_population
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to FITNESS-FN
```

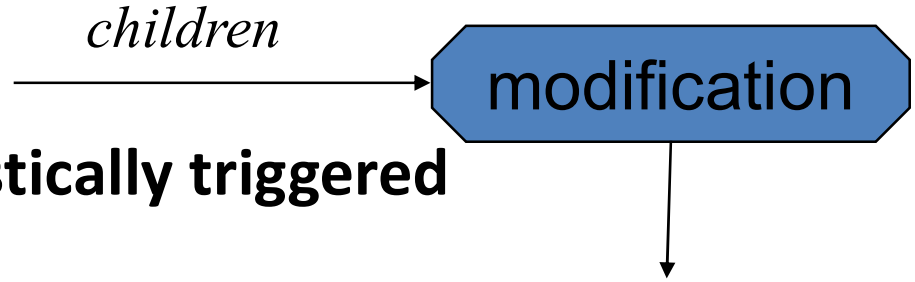
```
function REPRODUCE(x, y) returns an individual
  inputs: x, y, parent individuals

  n ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))
```

Fluxogram GAs



Mutation: Local Modification



- **Modifications are stochastically triggered**
- **Operator types are:**
  - **Mutation**
  - **Crossover (recombination)**

1	2	3	4	5	6	7
1	0	1	0	0	1	0

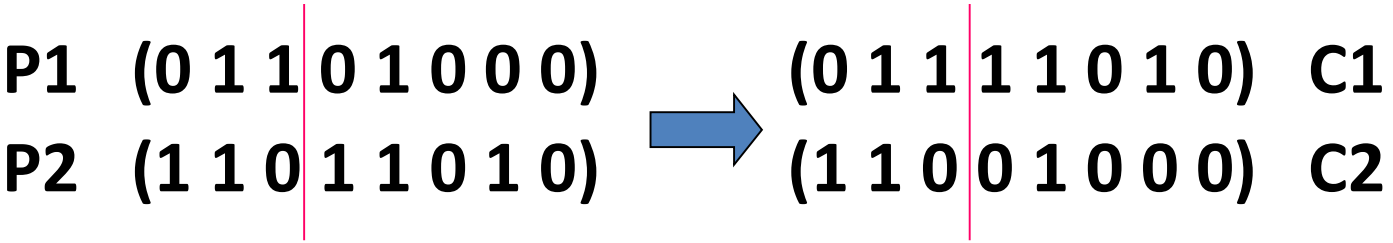
1	2	3	4	5	6	7
1	0	1	1	0	1	0

Mutation:

- Causes movement in the search space (local or global)
- Restores lost information to the population

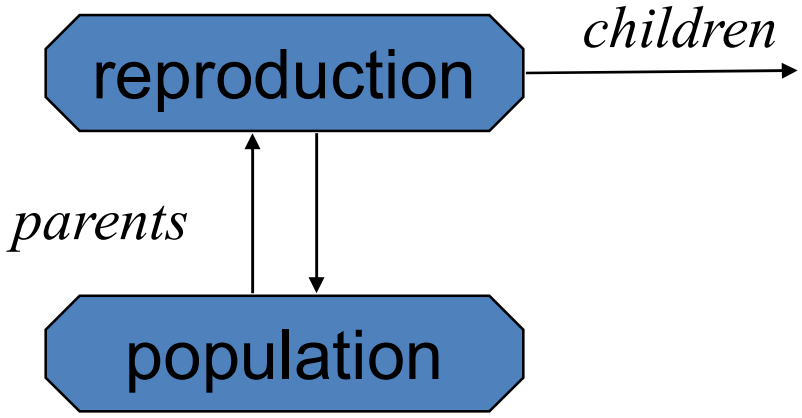
Crossover: Recombination

- Crossover combines parent chromosomes to generate child
- Example:



- **Crossover is a critical feature of genetic algorithms:**
  - It greatly accelerates search early in evolution of a population
  - It leads to effective combination of schemata (sub solutions on different chromosomes)

GAs – Selection/Reproduction

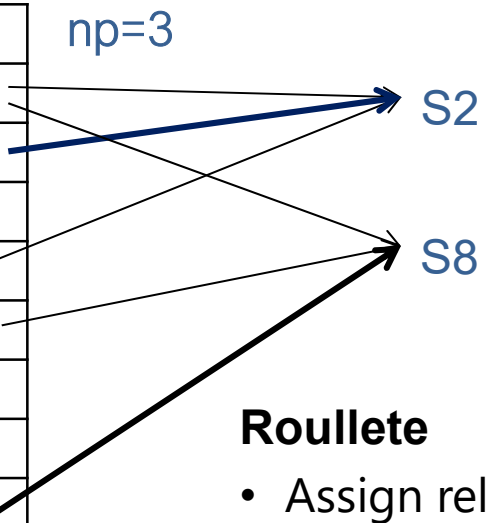


Parents are selected at random with selection chances biased in relation to chromosome evaluations – several methods

Selection Methods

- Select the best members of the current population for parents and reproduction
- Use ranking instead of objective function value?
- Tournament selection: Random choice of groups and the best in the group advances to reproduction
- Roullete: Selection based on probability given fitness

S	F(S)
1	14
2	17
3	17
4	11
5	8
6	23
7	22
8	20

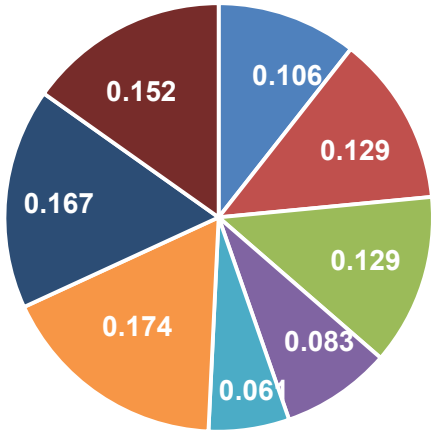


Tournament

- Set number of participants (np)
- Random selection of participants
- Best advances to reproduction

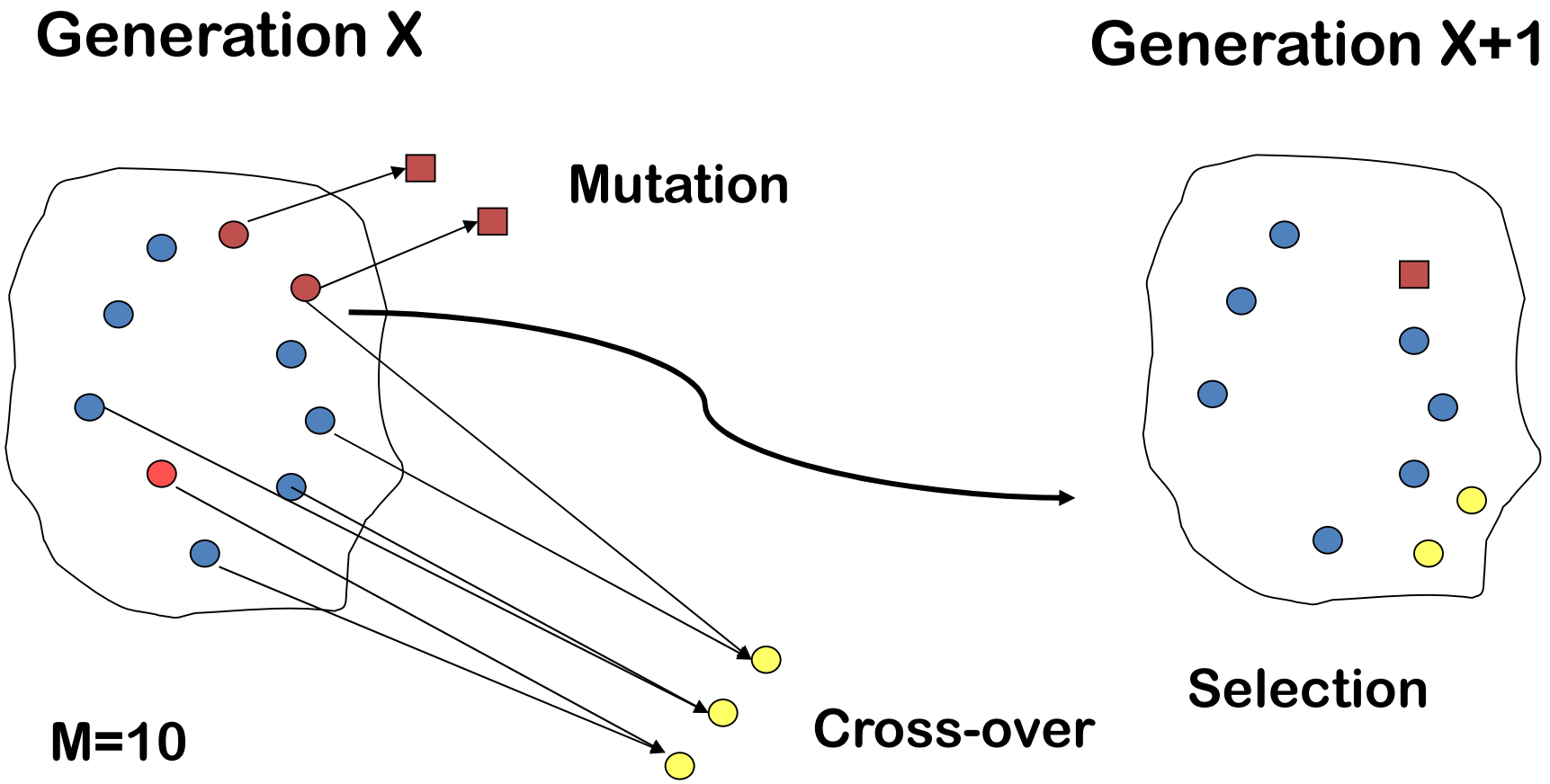
Roullete

- Assign relative value of the solution
- Selection of participants according to probability distribution



S	Prob(S)
1	0,106
2	0,129
3	0,129
4	0,083
5	0,061
6	0,174
7	0,167
8	0,152

GAs - Evolution



GAs - Population

Chromosomes could be:

- Bit strings (0101 ... 1100)
- Integer numbers (4, 7, 8, 6, … 8, 12, 24, 7)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- Program elements (genetic programming)
- ... any data structure ...

Classical GA: Binary chromosomes

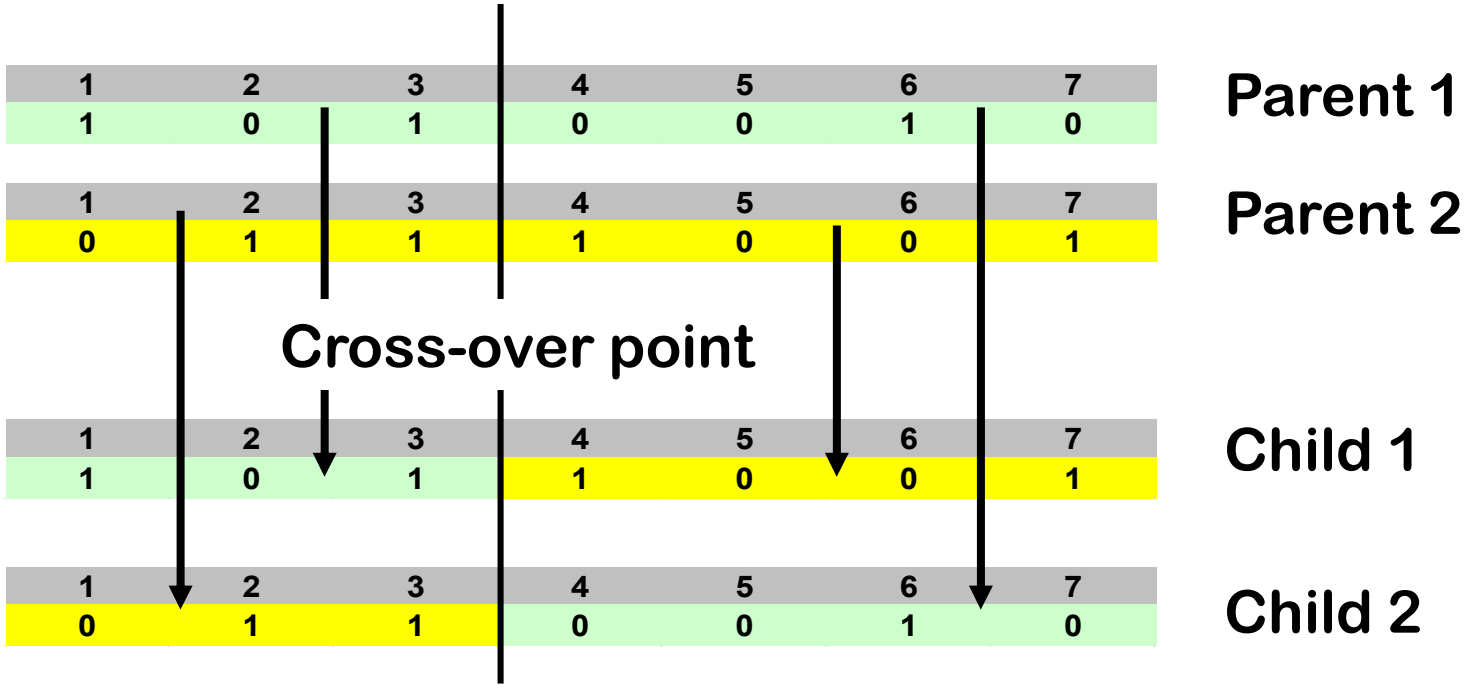
1	2	3	4	5	6	7
1	0	1	0	0	1	0

1	2	3	4	5	6	7
0	1	1	1	0	0	1

- **Functional optimization**
  - Chromosome corresponds to a binary encoding of a real number - min/max of an arbitrary function
- **COP, TSP as an example**
  - Binary encoding of a solution
  - Often better with a more direct representation (e.g. sequence representation)

GAs - Classical Crossover (1-point)

- One parent is selected based on *fitness*
- The other parent is selected randomly
- Random choice of cross-over point



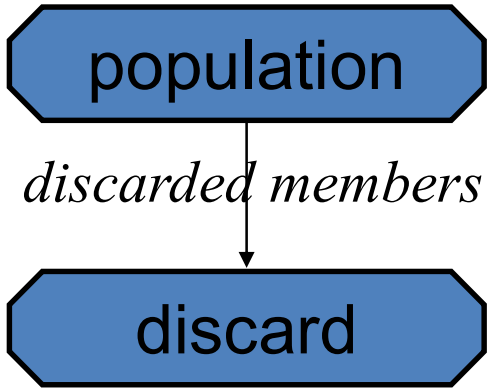
GAs – Classical Crossover

- Arbitrary (or worst) individual in the population is changed with one of the two offspring (e.g. the best)
- Reproduce as long as you want
- Can be regarded as a sequence of almost equal populations
- Alternatively:
  - One parent selected according to fitness
  - Crossover until (at least) M offspring are created
  - The new population consists of the offspring
- Lots of other possibilities ...
- Basic GA with classical crossover and mutation often works well

GAs – Standard Reproduction Plan

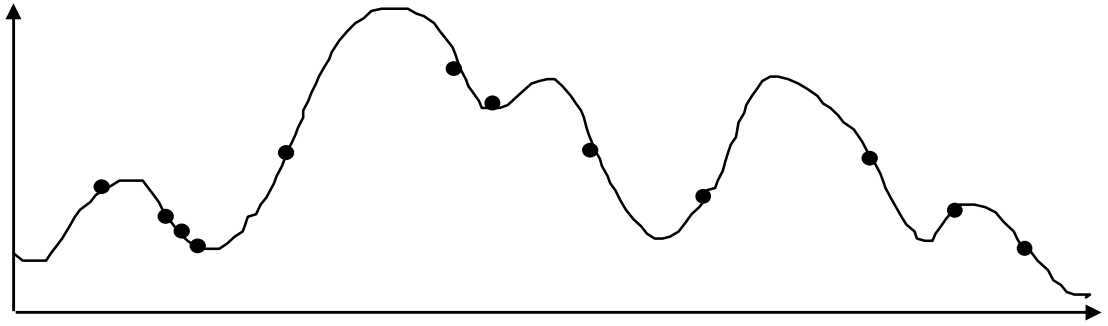
- **Fixed population size**
- **Standard cross-over**
  - One parent selected according to fitness
  - The other selected randomly
  - Random cross-over point
  - A random individual is exchanged with one of the offspring
- **Mutation**
  - A certain probability that an individual mutate
  - Random choice of which gene to mutate
  - Standard: mutation of offspring

GAs - Deletion

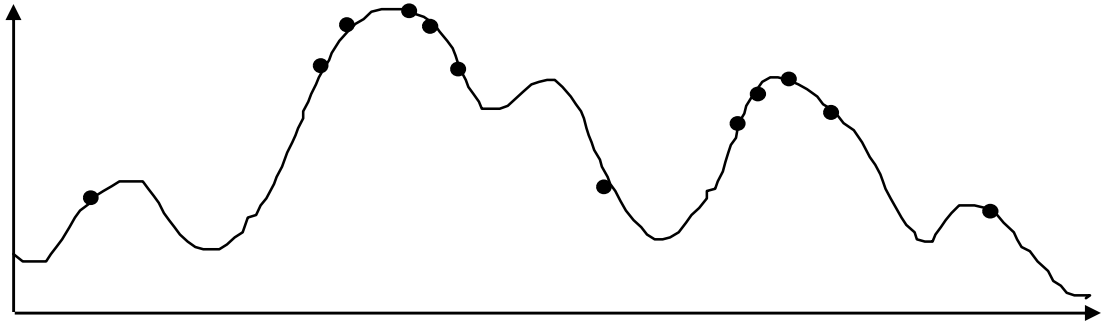


- ***Generational* GA:**
  entire populations replaced each iteration
- ***Steady-state* GA:**
  a few members replaced each generation

GAs Abstract Example



Distribution of Individuals in Generation 0



Distribution of Individuals in Generation N

Example: Traveling Salesman Problem

Find a tour of a given set of cities so that

- each city is visited only once
- the total distance traveled is minimized

Representation is an ordered list of city numbers known as an *order-based* GA.

1) London 3) Dunedin 5) Beijing 7) Tokyo
2) Venice 4) Singapore 6) Phoenix 8) Victoria

City List 1 (3 5 7 2 1 6 4 8)
City List 2 (2 5 7 6 8 1 3 4)

Crossover

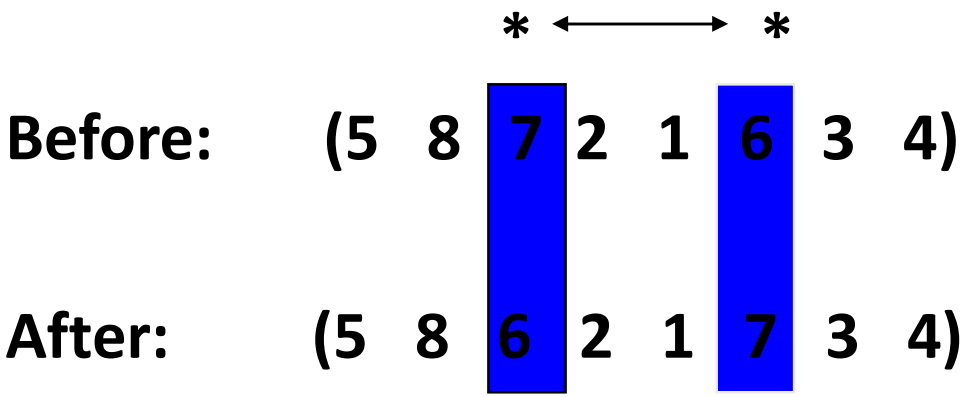
Crossover combines inversion and recombination:

			*			*		
Parent1	**(3**	**5**	**7**	**2**	**1**	**6**	**4**	**8)**
Parent2	**(2**	**5**	**7**	**6**	**8**	**1**	**3**	**4)**
Child	**(5**	**8**	**7**	**2**	**1**	**6**	**3**	**4)**

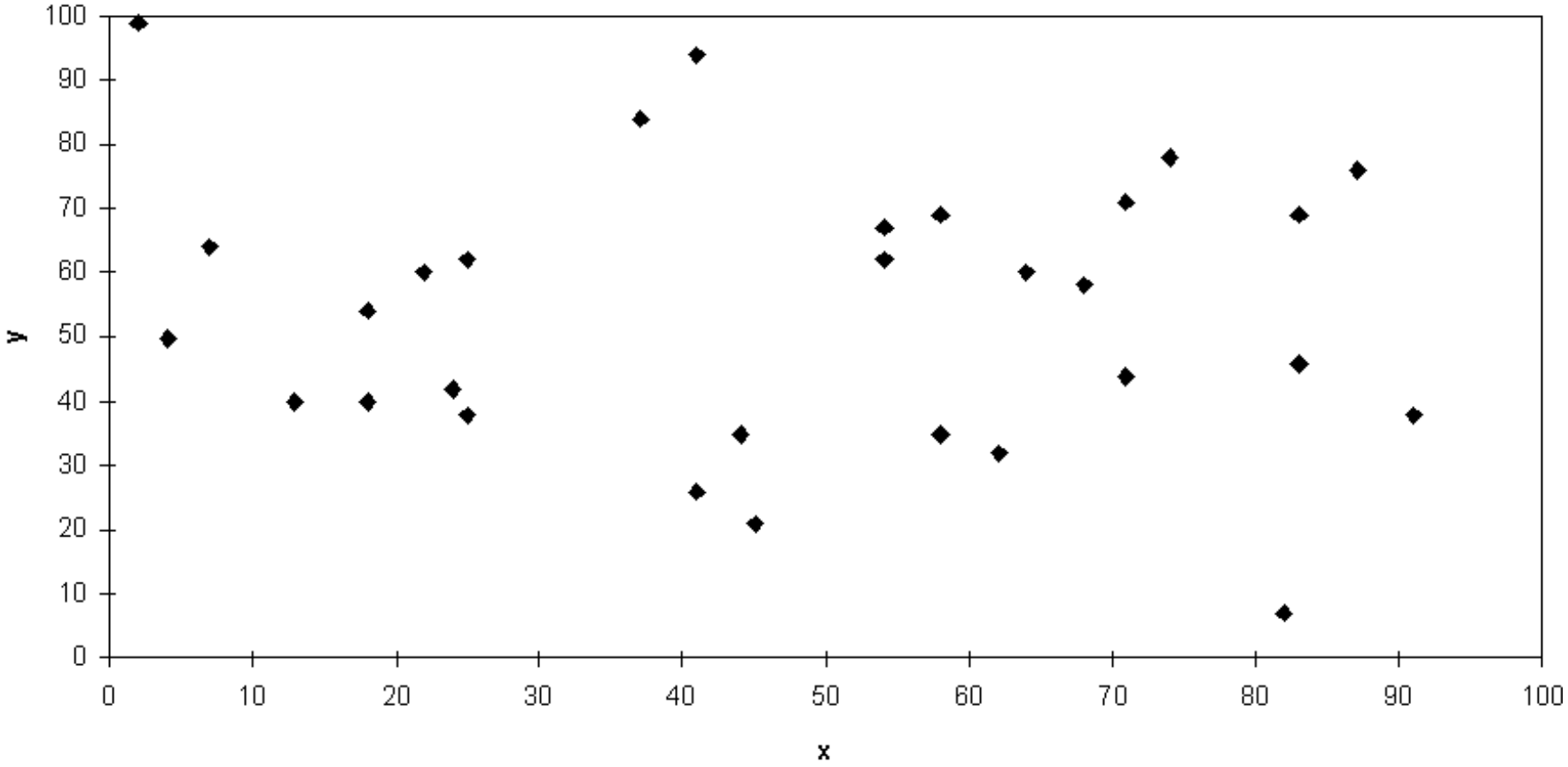
This operator is called order-based crossover.

Mutation

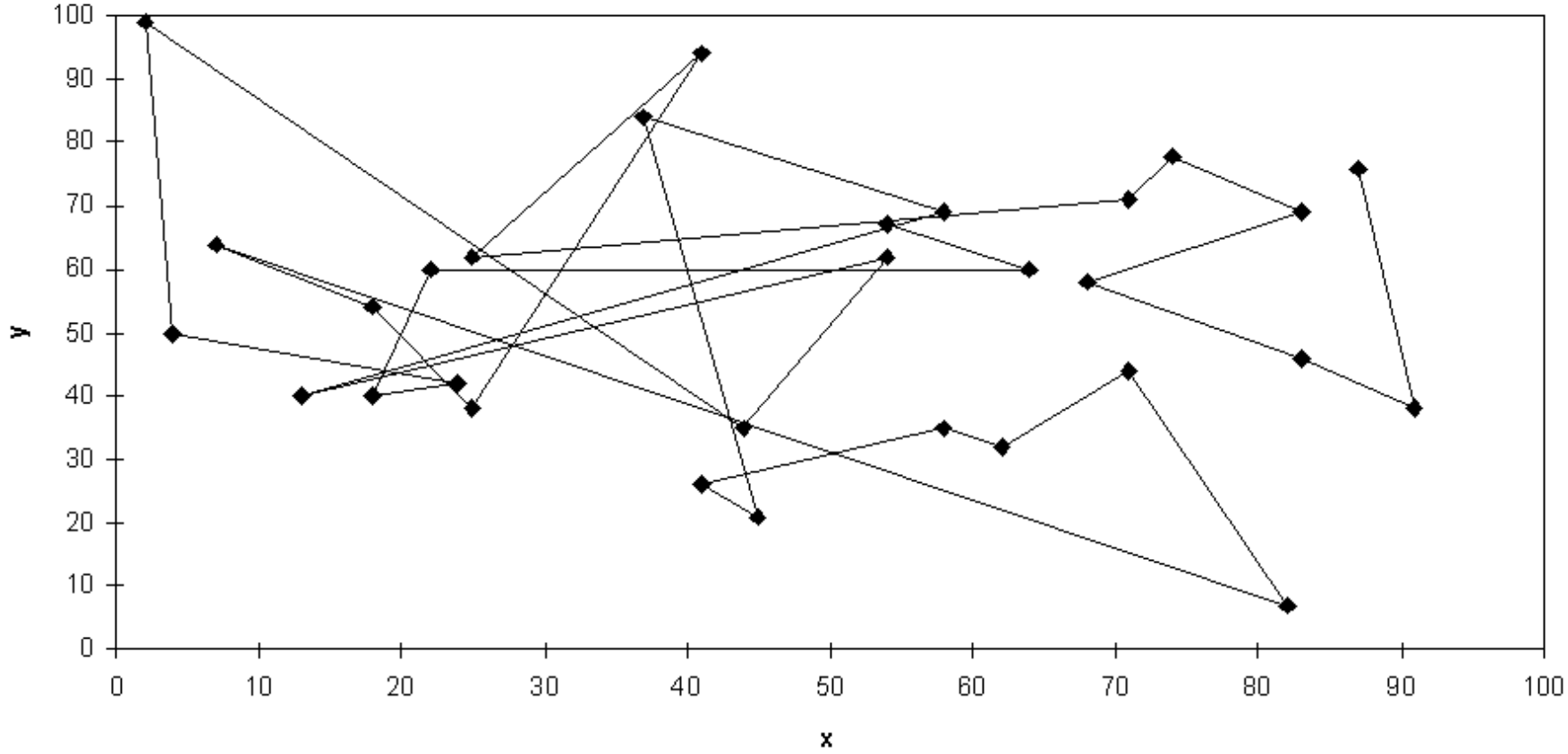
Mutation involves reordering of the list:



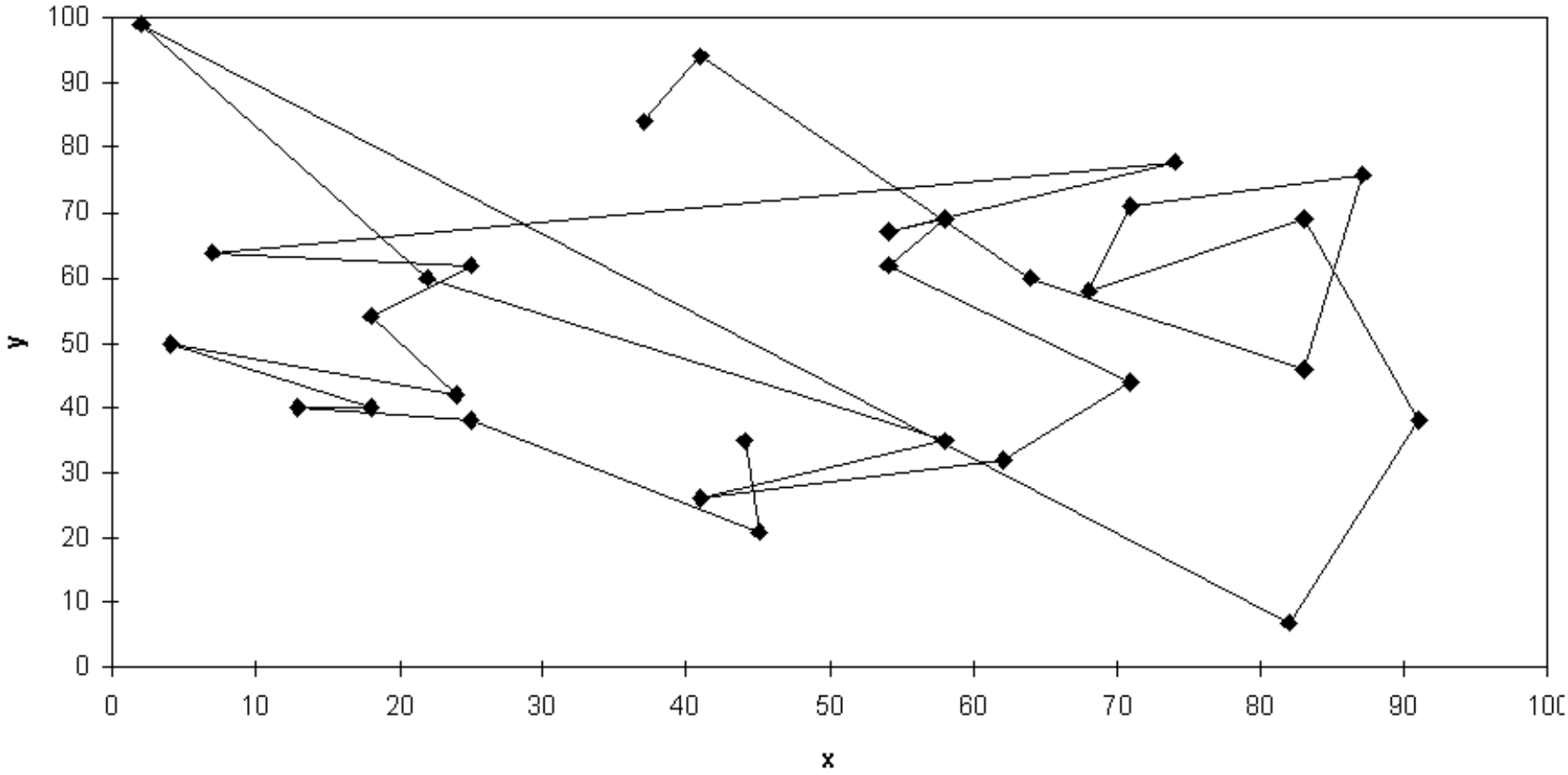
TSP Example: 30 Cities



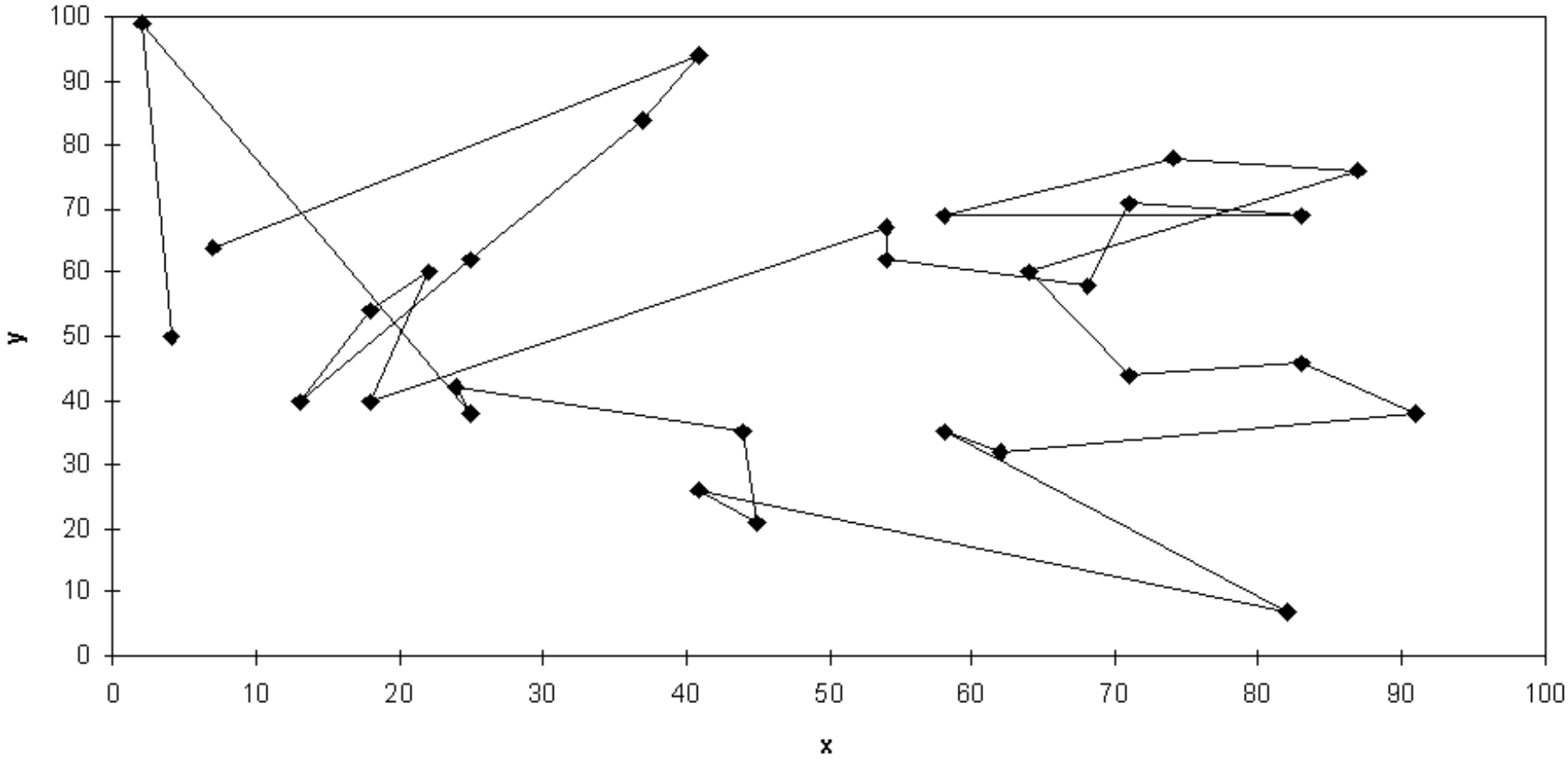
Solution $_1$ (Distance = 941)



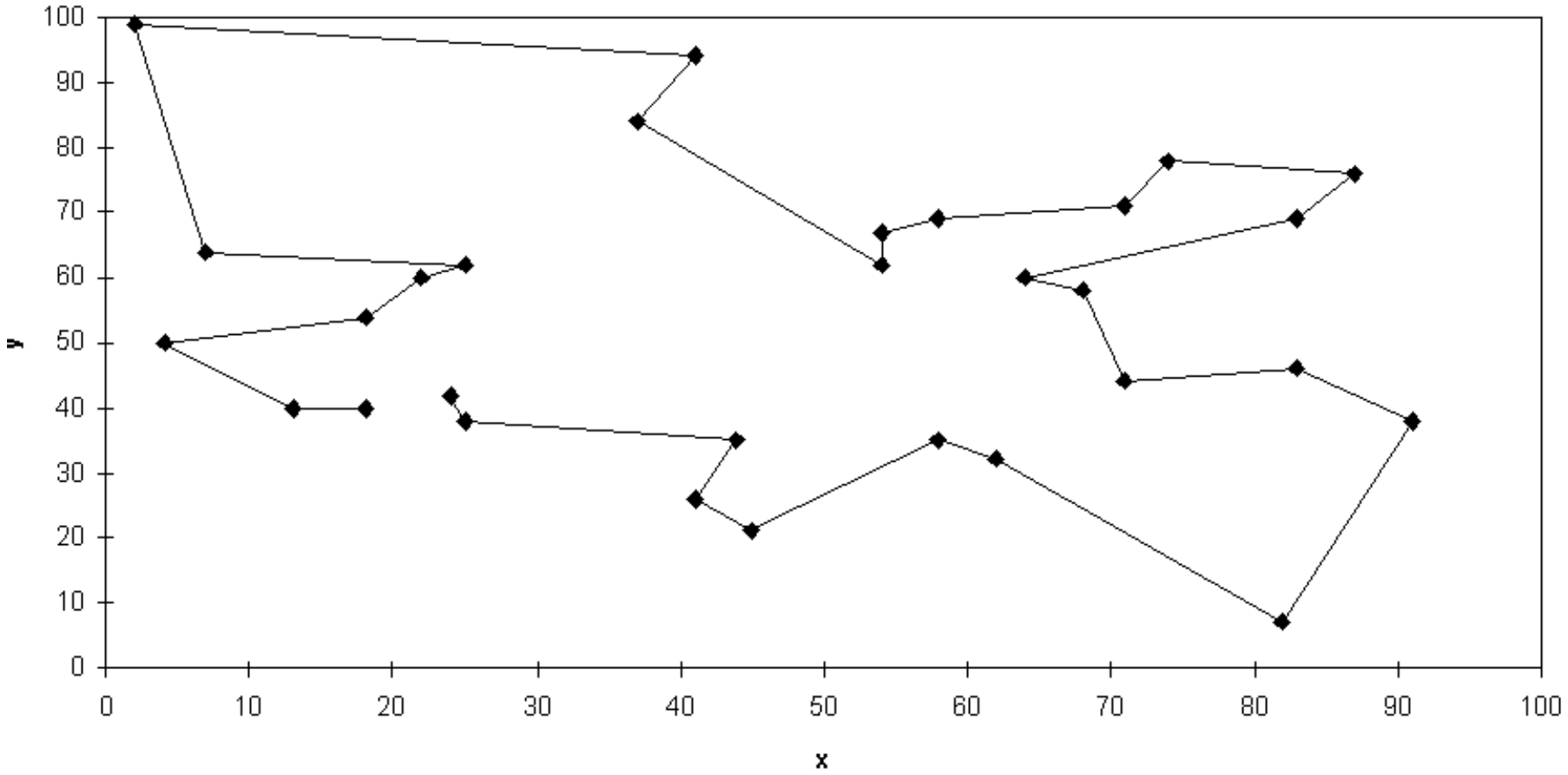
Solution $_2$ (Distance = 800)



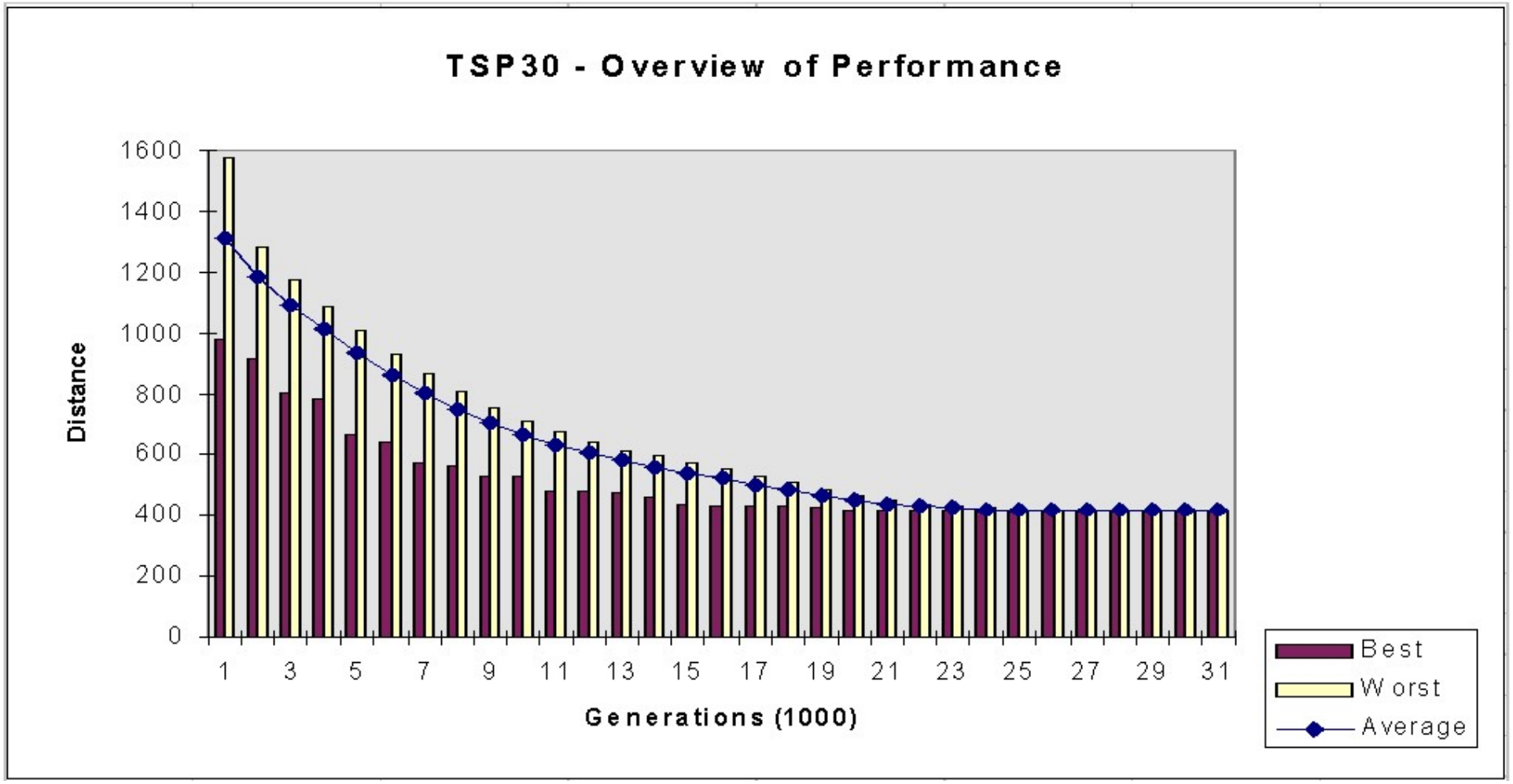
Solution $_3$ (Distance = 652)



Best Solution (Distance = 420)



Overview of Performance



Theoretical Analysis of GAs (Holland)

- Schema: subsets of chromosomes that are similar
- The same alleles in certain locations

1	2	3	4	5	6	7
1	0	1	0	0	1	0

1	2	3	4	5	6	7
0	1	1	1	0	0	1

1	2	3	4	5	6	7
*	*	1	*	0	*	*

- A given chromosome can be in many schema

Fitness Ratio

- The relation between the average fitness of a schema and the average fitness of the population
- *F(S)* is the fitness of schema *S*
- *F(t)* is the average fitness of the population at time *t*
- Let *f(S,t)* be the fitness ratio for schema *S* at time *t*

GA - Extensions and Modifications

- **Many possibilities for variations**
- **A lot of literature, chaotic**
- **Unclear terminology**
- **Modifications regarding:**
  - Population
  - Encoding
  - Operators
  - Hybridization, parallellization

GAs: Population

Population Size:

- Small populations: undercoverage
- Large population: computationally demanding
- Optimal size increases exponentielly with the string-length in binary encodings
- A size of 30 can often work (OK with 10-100)
- Between N and 2N (Alander)

Initial Popultation:

- Usually: random strings
- Alternative: seed with good solutions
  - faster convergence but sometimes premature convergence
- Sophisticated statistical methods
- Often problems with infeasibility

GAs: Population Updates

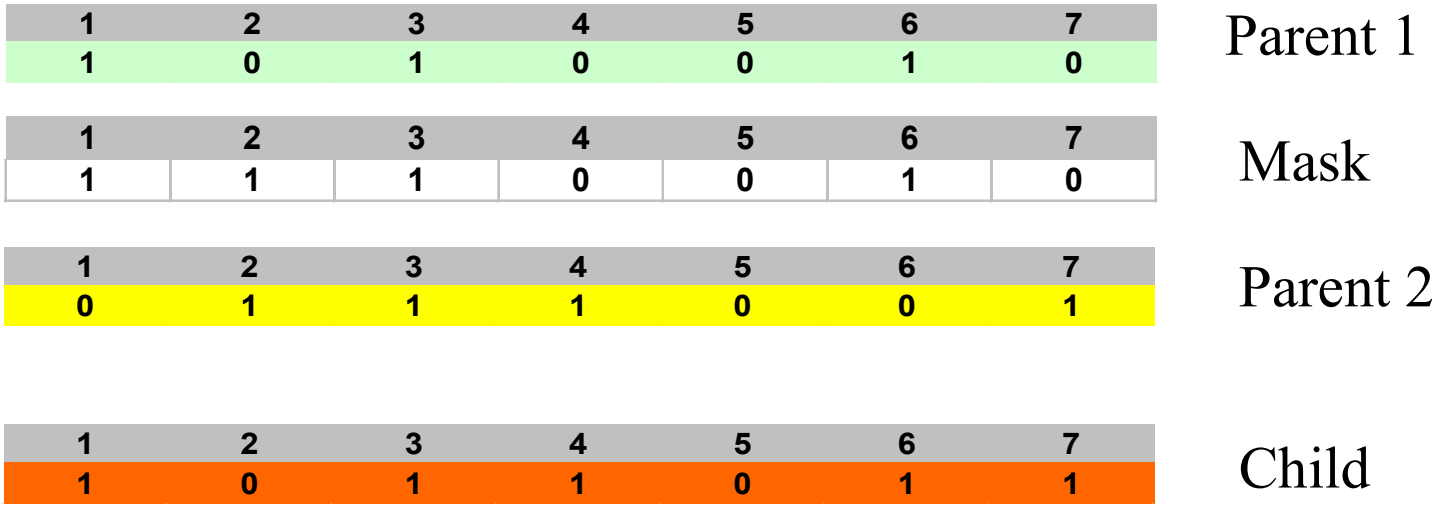
- **Generation gap**
  - Replace the whole population each iteration
- **Steady state**
  - Add and remove one individual each generation
  - Only use part of the population for reproduction
  - The offspring can replace
    - Parents
    - Worst member of population
    - Randomly selected individuals (doubtful if this works better)
- **Avoid duplicates**
- **Uncertain if the best solution so far will survive**
- **Elitism – e.g. have a small set of ”queens”**
- **Selective death**

GAs: Operators

- **Mutation – upholds diversity**
  - Choice of mutation rate not critical
- **Crossover – often effective**
  - Late in the search: crossover has smaller effect
  - Selective choice of crossover point
- **N-point crossover**
  - 2-points has given better performance
  - 8-point crossover has given best results

GAs: Generalized Crossover

- Bit-string specifies which genes to use



GA: Hybridization and Parallelization

- **GAs strengths and weaknesses:**
  - Domain independence
- **Hybridization**
  - Seed good individuals in the initial population
  - Combine with other Metaheuristics to improve some solutions
- **Parallelization**
  - Fitness-evaluation
  - Sub-populations
  - The Island Model

Issues for GAs Developers

- **Basic implementation issues:**
  - Representation
  - Population size, mutation rate, ...
  - Selection, deletion policies
  - Crossover, mutation operators
- **Termination Criteria**
- **Performance, scalability**
- **Solution is only as good as the evaluation function (often hardest part)**

Benefits of Genetic Algorithms

- Concept is easy to understand
- Modular, separate from application
- Supports multi-objective optimization
- Good for “noisy” environments
- Always an answer; answer gets better with time
- Inherently parallel; easily distributed
- Many ways to speed up and improve a GA-based application as knowledge about the problem domain is gained
- Easy to exploit previous or alternate solutions
- Flexible building blocks for hybrid applications
- Substantial history and range of use

When to Use a GAs

- Alternate methods are too slow or overly complicated
- Need an exploratory tool to examine new approaches
- Problem is similar to one that has already been successfully solved by using a GA
- Want to hybridize with an existing method
- Benefits of the GA technology meet key problem requirements

Some GAs Application Types

Domain	Application Types
Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner's dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning

GAs: Overview

- **Important characteristics:**
  - Population av solutions
  - Domaine independence – encoding
  - Structure is not exploited
  - Inherent parallell – schema, vocabulary
  - Robust
  - Good mechanisms for intensification
  - Lacking in diversification
- Bit-string encoding is inappropriate for many combinatorial problems. In particular, crossover may lead to infeasible or meaningless solutions
- Pure GAs are usually not powerful enough to solve hard combinatorial problems
- Hybrid GAs use some form of local search as mutation operator to overcome this

Memetic Algorithms

- **Basically, a Memetic Algorithm is a GA with Local Search as improvement mechanism**
  - Also known under different names
  - An example of hybridization
- **A meme is a unit of cultural information transferable from one mind to another**
  - Sounds like gene: the unit carrying inherited information
- The experience is that GAs do not necessarily perform well in some problem domains
- Using Local Search in addition to the population mechanisms is an improvement
- In a sense this elevates the population search to a search among locally optimal solutions, rather than among any solution in the solution space

Summary

- **Local Search**
  - Short summary
- **Genetic Algorithms**
  - Population based Metaheuristic
  - Based on genetics:
    - Mutation
    - Combination of chromosomes from parents
  - Hybridization: Memetic Algorithm

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence/ Inteligência Artificial

Lecture 3c: Optimization and Genetic Algorithms

(based on Løkketangen, 2019)

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence