

The Genetic Algorithm

- 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

Genetic Algorithms

- Provide efficient, effective techniques for optimization and machine learning applications
- Widely-used today in business, scientific and engineering circles

Genetic Algorithms (GA)

- Function Optimization
- AI (Games, Pattern recognition ...)
- OR after a while
- Basic idea:
 - intelligent exploration of the search space based on random search
 - analogies from biology

GA - Analogies with biology

- Representation of complex objects by a vector of simple components
- Chromosomes
- Selective breeding
- Darwinistic evolution
- Classical GA: Binary encoding

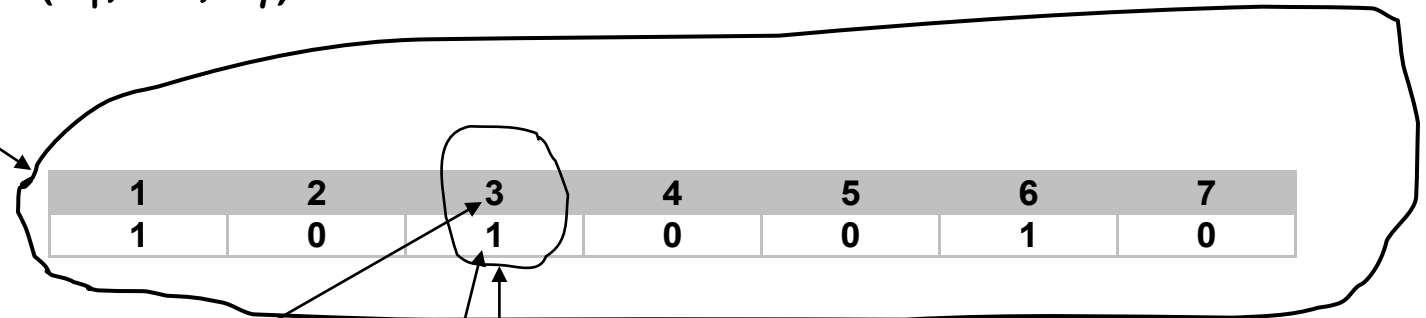
Components of a GA

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 $\mathbf{x}=(x_1, \dots, x_7)$



Gene, Component, Variable, x_3

Locus, position

Allele, value

$x_3 \in \{0,1\}$

Alleles, domain

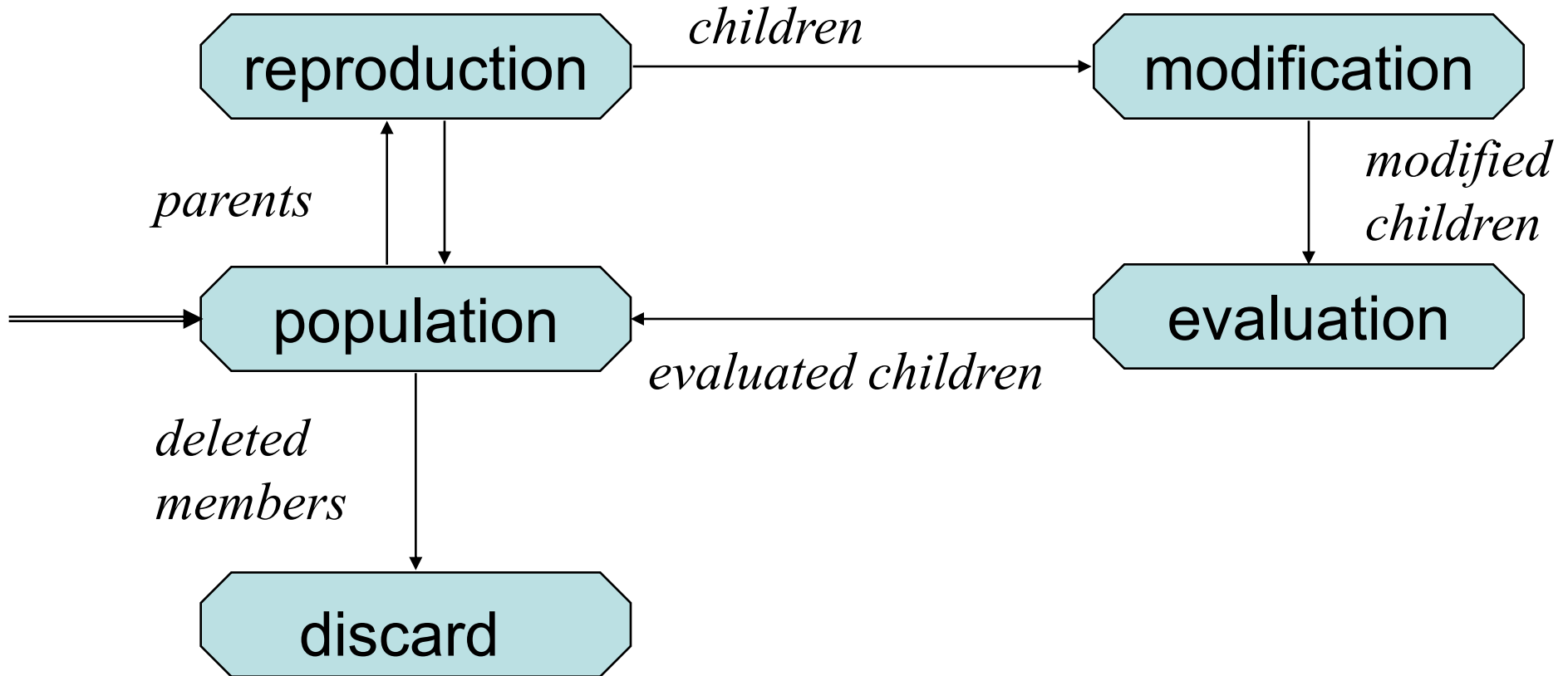
Genotype, Phenotype, Population

- 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

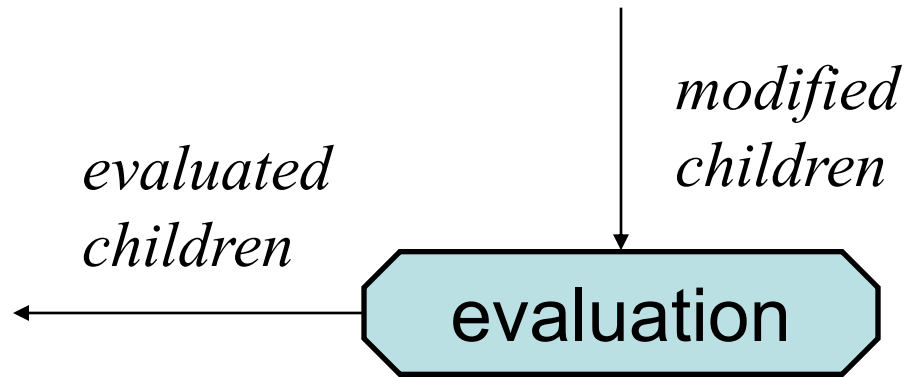
Genetic Algorithm

- 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

The GA Cycle of Reproduction



Evaluation



- The evaluator decodes a chromosome and assigns it a fitness measure
- The evaluator is the only link between a classical GA and the problem it is solving

Evaluation of Individuals

- Adaptability – “fitness”
- Relates to the objective function value for a DOP
- Fitness is maximized
- Used in selection (“*Survival of the fittest*”)
- Often *normalized*

$$f : S \rightarrow [0,1]$$

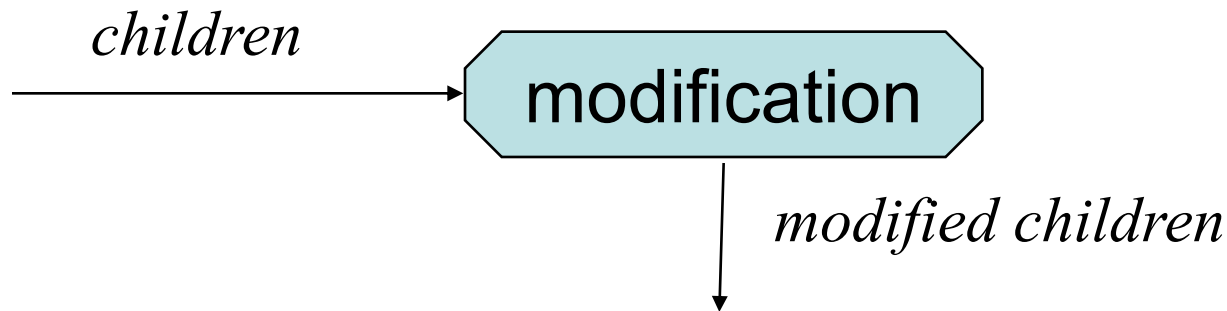
Genetic Operators

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

GA - Evolution

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

Chromosome Modification



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

GA - Mutation

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: Local Modification

Before: (1 0 1 **1** 0 1 1 0)

After: (1 0 1 **0** 0 1 1 0)

Before: (1.38 **-69.4** 326.44 0.1)

After: (1.38 **-67.5** 326.44 0.1)

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

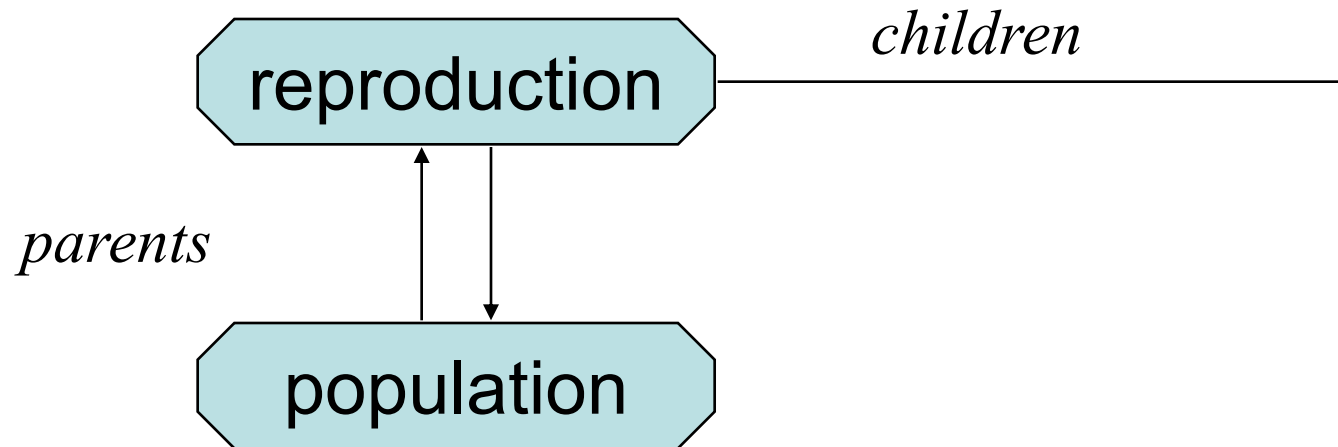
Crossover: Recombination

$$\begin{array}{lcl} \text{P1} & (0 \ 1 \ 1 \mid 0 \ 1 \ 0 \ 0 \ 0) & \longrightarrow (0 \ 1 \ 1 \mid 1 \ 1 \ 0 \ 1 \ 0) \quad \text{C1} \\ \text{P2} & (1 \ 1 \ 0 \mid 1 \ 1 \ 0 \ 1 \ 0) & (1 \ 1 \ 0 \mid 0 \ 1 \ 0 \ 0 \ 0) \quad \text{C2} \end{array}$$

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 (subsolutions on different chromosomes)

Reproduction

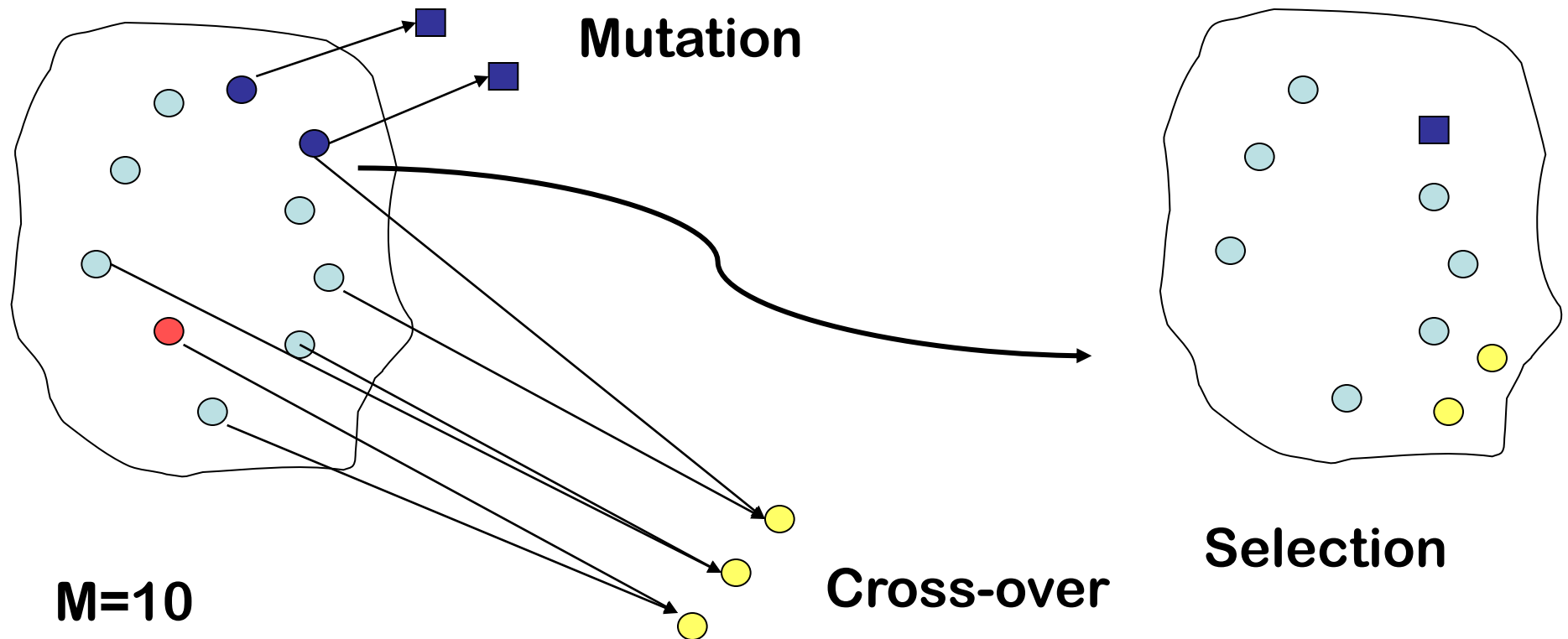


Parents are selected at random with selection chances biased in relation to chromosome evaluations

GA - Evolution

Generation X

Generation X+1



Population



Chromosomes could be:

- Bit strings (0101 ... 1100)
- 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

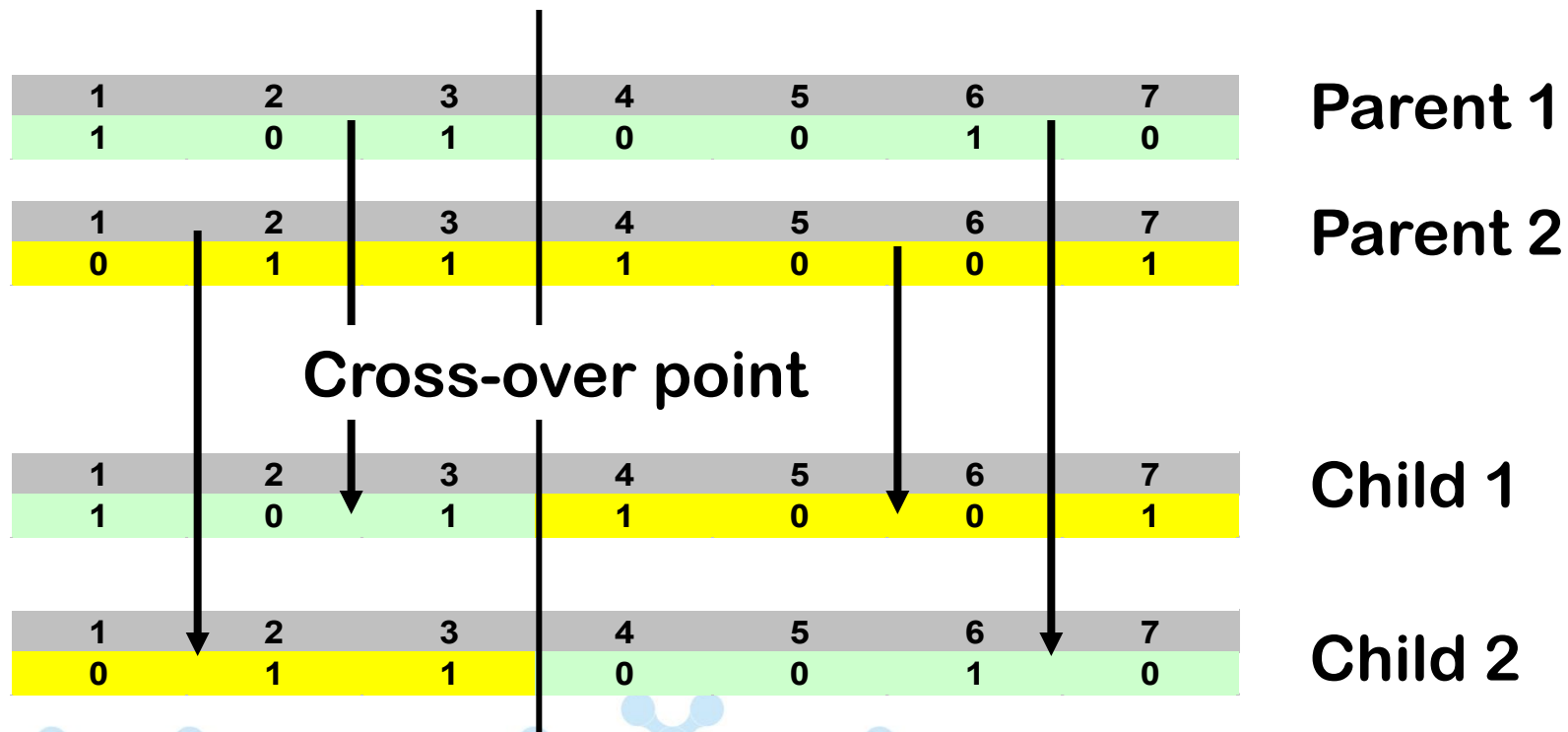
1	2	3	4	5	6	7
1	0	1	0	0	1	0

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)

GA - Classical Crossover (1-point)

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



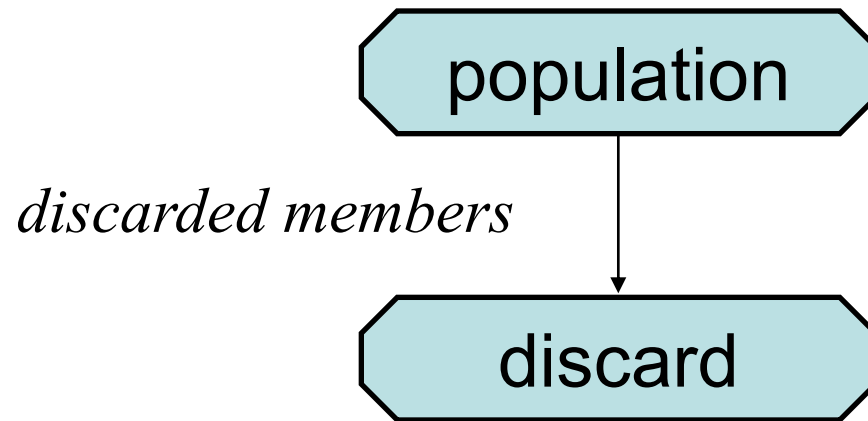
GA – Classical Crossover

- 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

GA – 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

Deletion



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

Methods of Selection

- *Roulette-wheel selection.*
- *Elitist selection.*
- *Fitness-proportionate selection.*
- *Scaling selection.*
- *Rank selection.*
- ...

Roulette wheel selection

- Conceptually, this can be represented as a game of roulette - each individual gets a slice of the wheel, but more fit ones get larger slices than less fit ones.

Roulette wheel selection

No.	String	Fitness	% Of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

Methods of Reproduction

- There are primary methods:
 - *Crossover*
 - *Mutation*

Methods of Reproduction:

Crossover

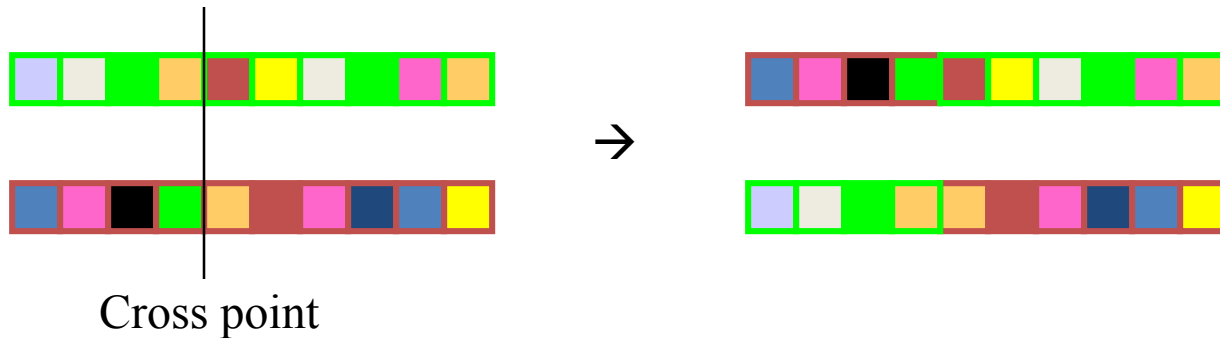
- Two parents produce two offspring
- Two options:
 1. The chromosomes of the two parents are copied to the next generation
 2. The two parents are randomly recombined (crossed-over) to form new offsprings

Several possible crossover strategies

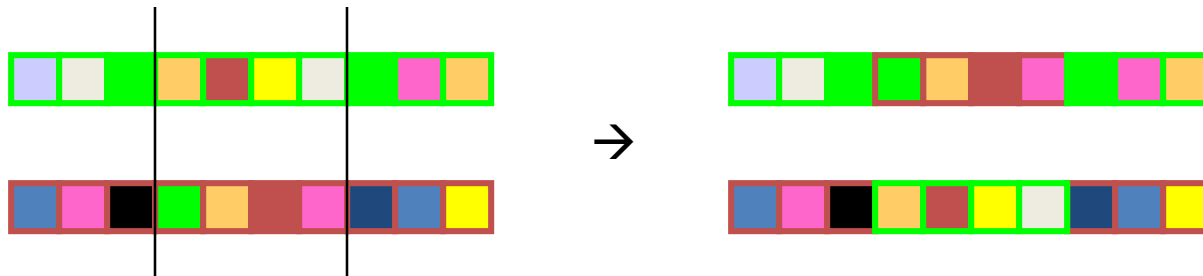
- Randomly select a single point for a crossover
- Multi point crossover
- Uniform crossover

Crossover

- Single point crossover

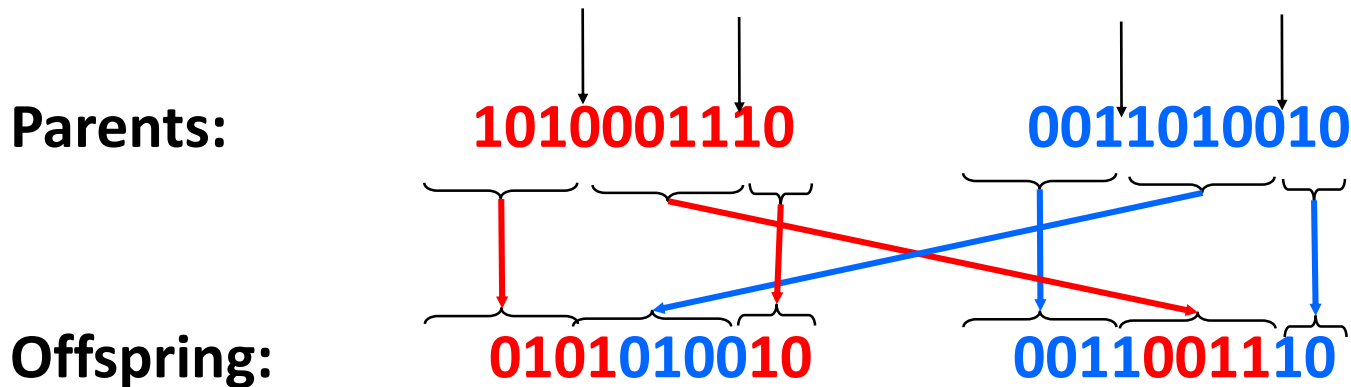


- Two point crossover (Multi point crossover)



Two-point crossover

- Avoids cases where genes at the beginning and end of a chromosome are always split



Uniform crossover

- A random subset is chosen
- The subset is taken from parent 1 and the other bits from parent 2.

Subset: **BAABBAABBB** **(Randomly generated)**

Parents: **1010001110** **0011010010**

Offspring: **0011001010** **1010010110**

A Simple Example

The 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

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

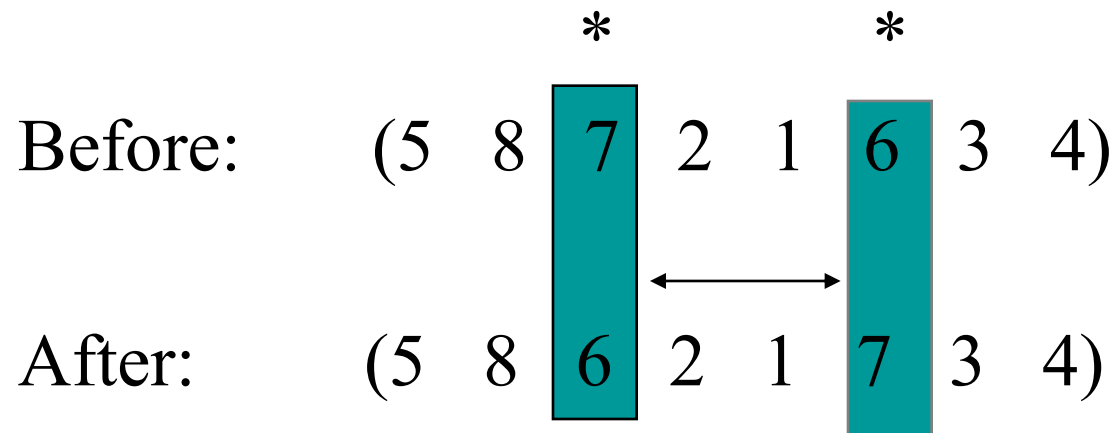
Crossover combines inversion and recombination:

		*		*			
Parent1	(3	5	7	2	1	6	4 8)
Parent2	(2	5	7	6	8	1	3 4)
<hr/>							
Child	(5	8	7	2	1	6	3 4)

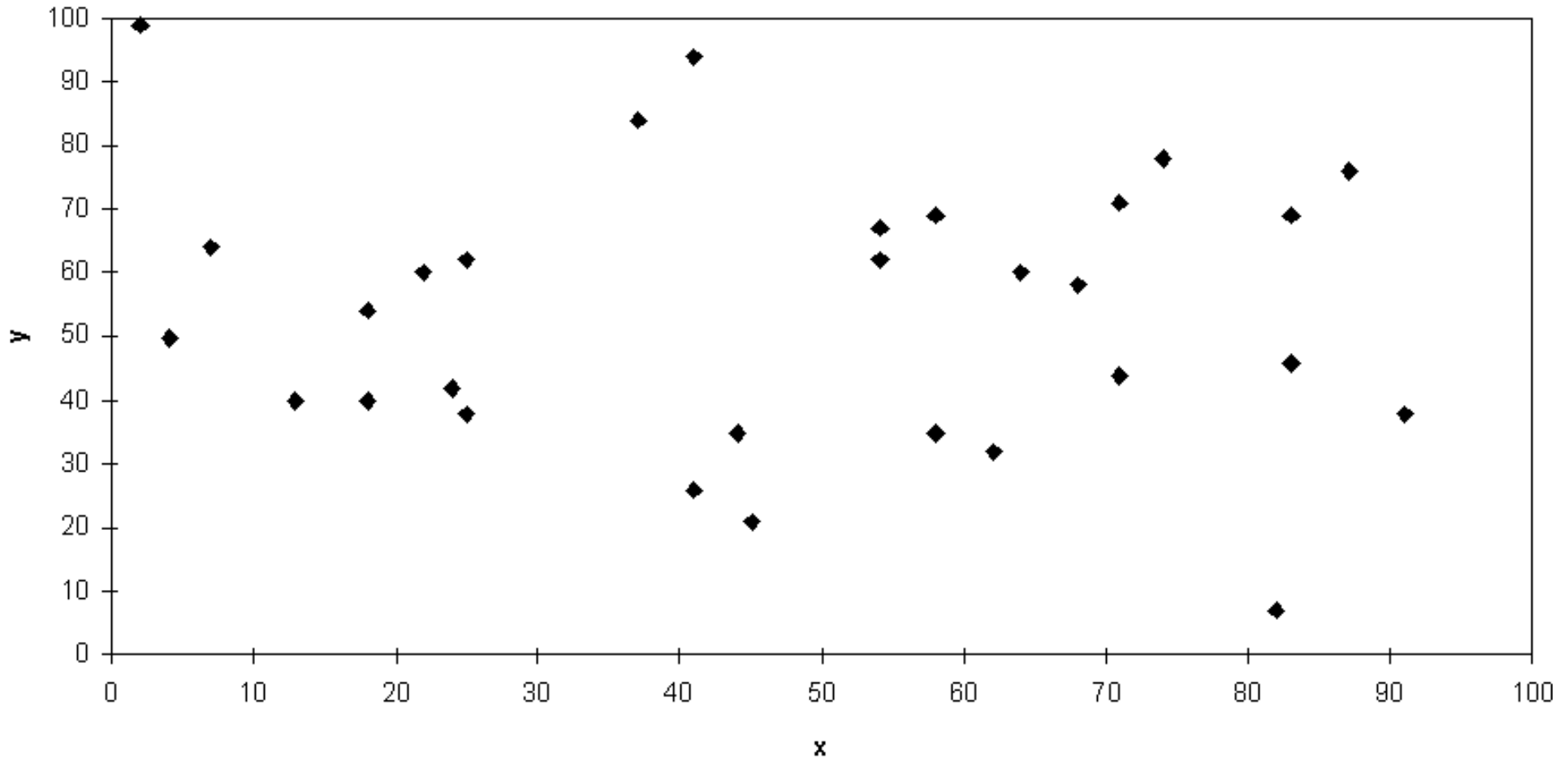
This operator is called order-based crossover.

Mutation

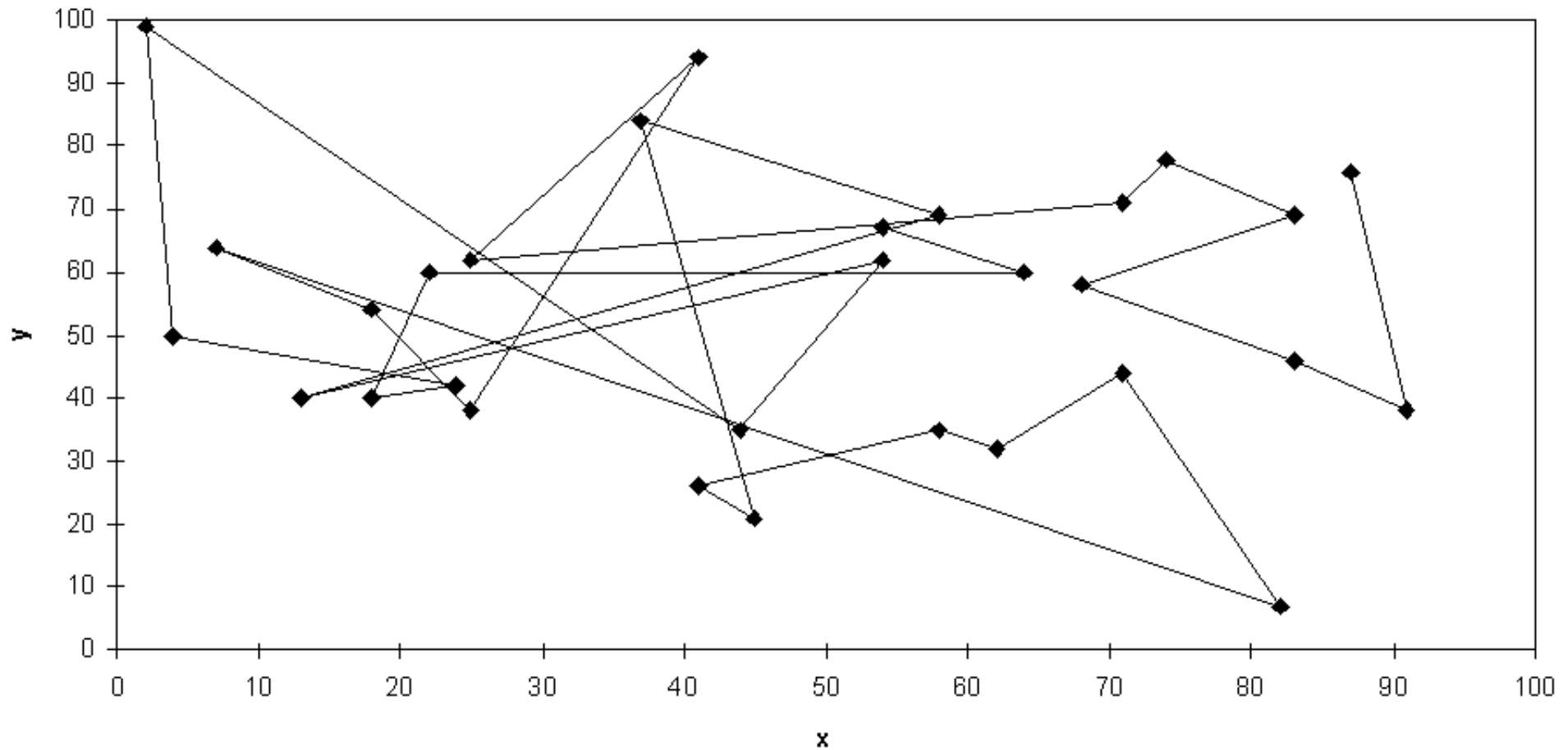
Mutation involves reordering of the list:



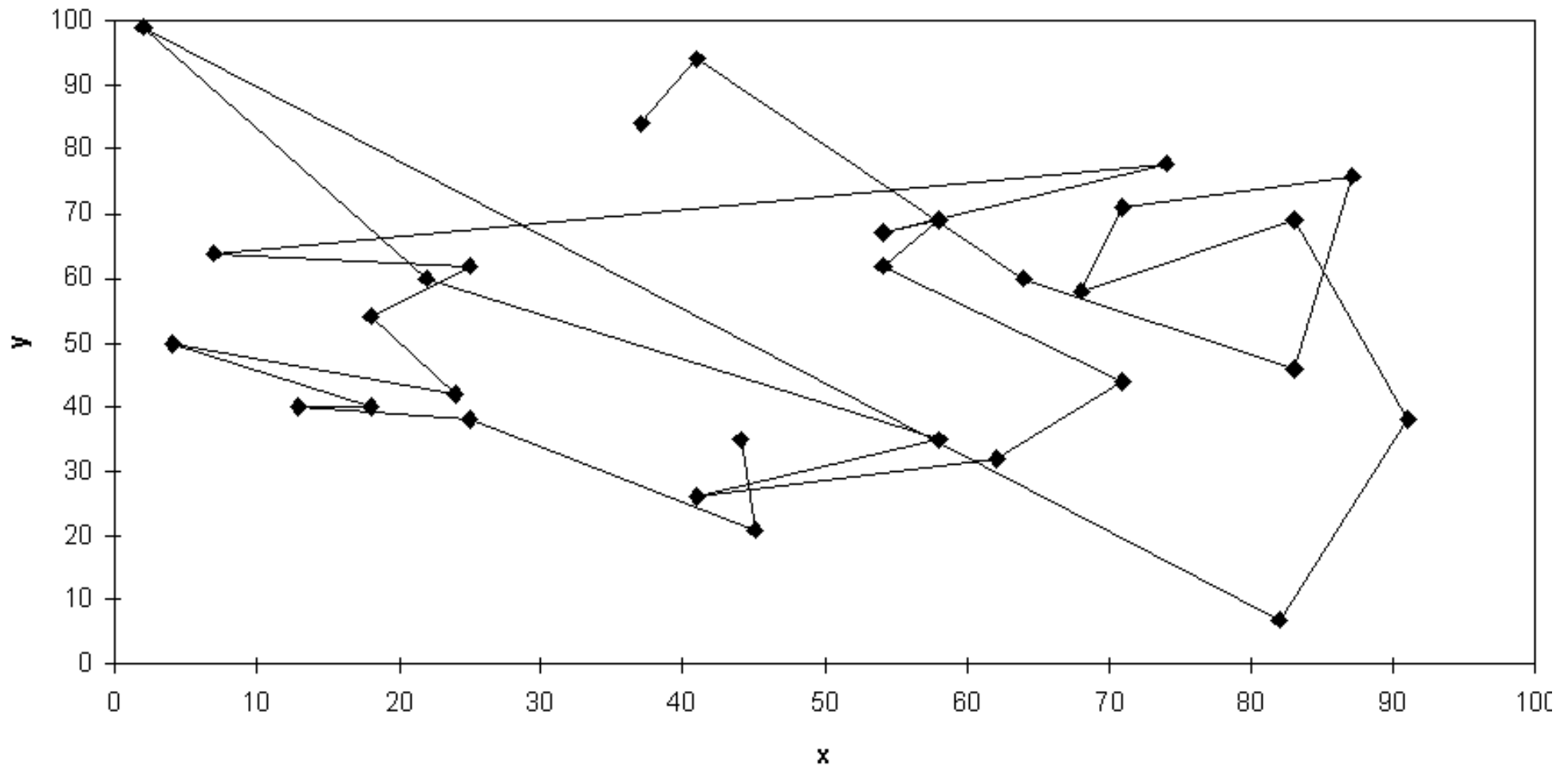
TSP Example: 30 Cities



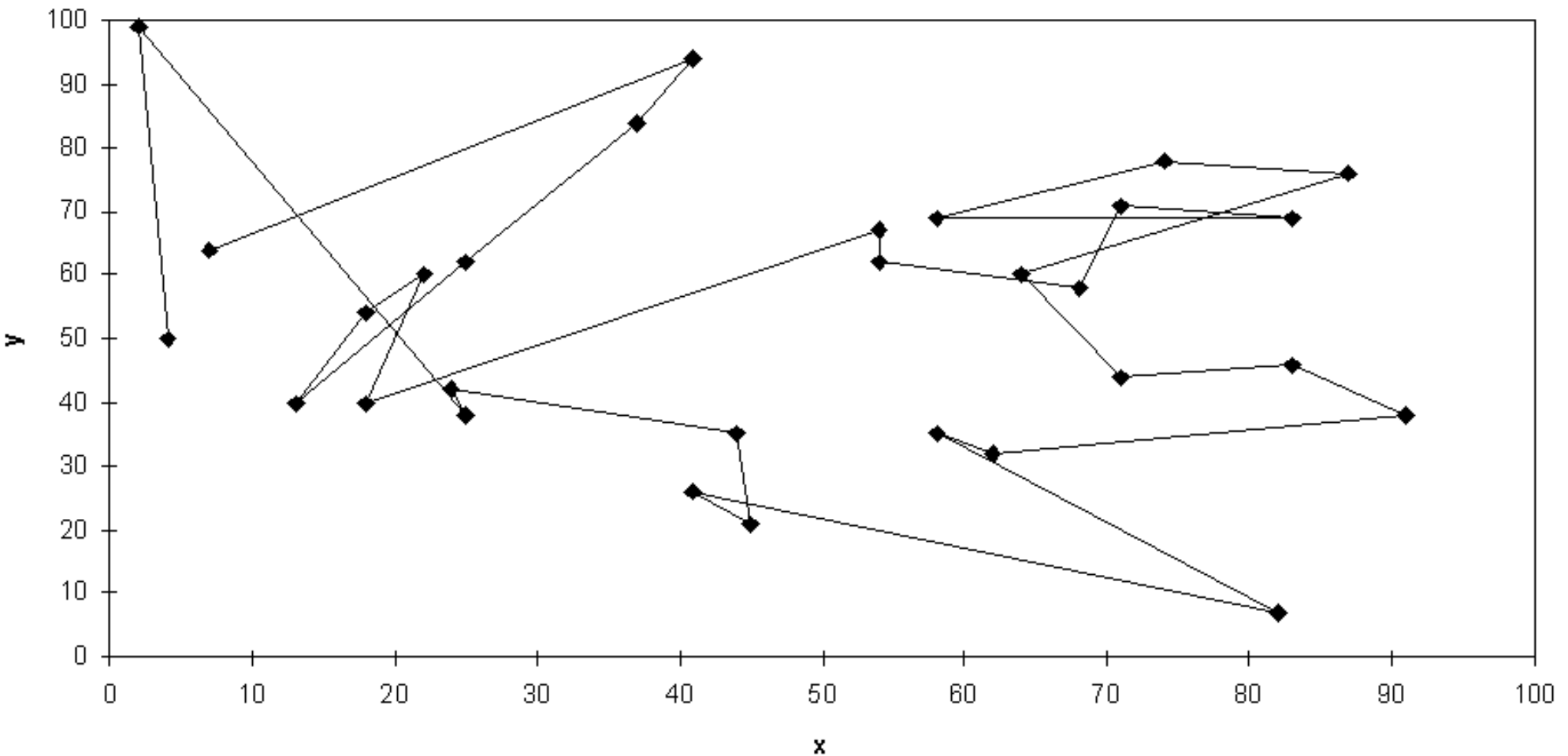
Solution i (Distance = 941)



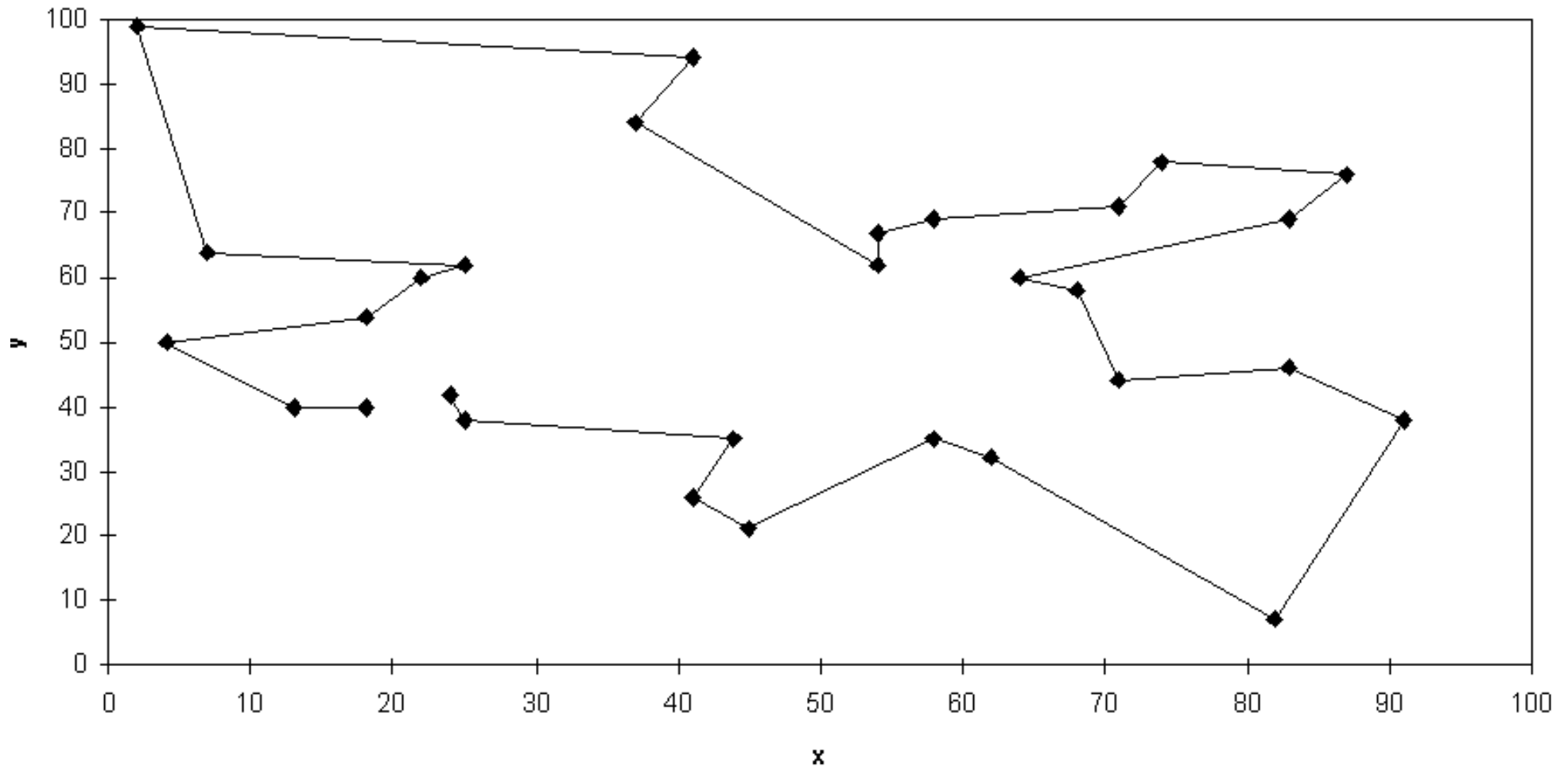
Solution j (Distance = 800)



Solution $_k$ (Distance = 652)



Best Solution (Distance = 420)



Overview of Performance

TSP30 - Overview of Performance

