Global Optimization
Genetic Algorithms

Olesya Peshko

Outline

- Evolution in biology
- Algorithm
- Pros and cons
- Applications
- Example
- Software
- Matlab toolboxes

Evolution in Biology

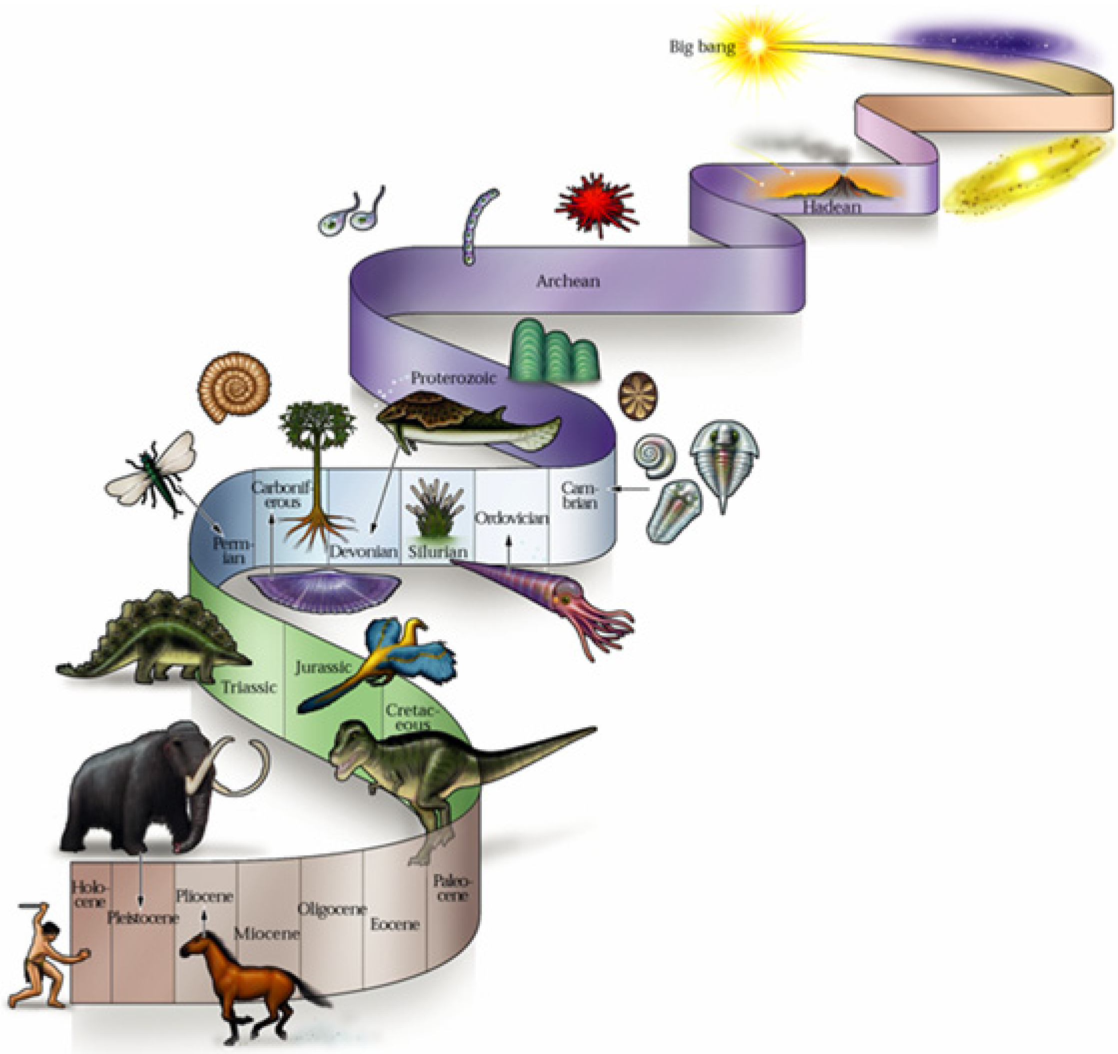
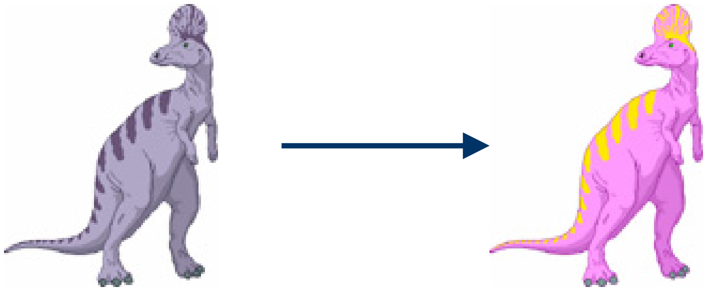


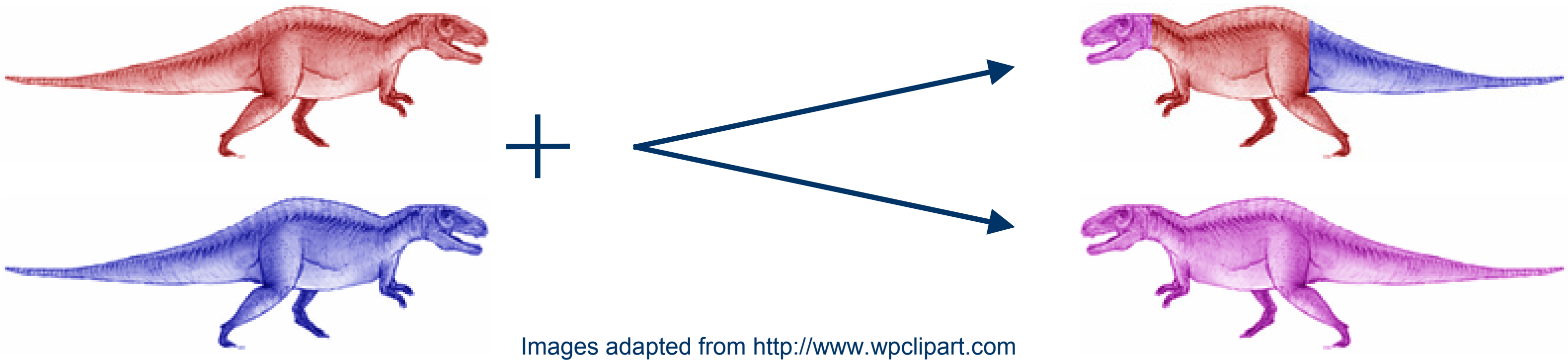
Image from http://www.geo.au.dk/besoegsservice/foredrag/evolution

Evolution in Biology I

- Organisms produce a number of offspring similar to themselves but can have variations due to:
  - Mutations (random changes)



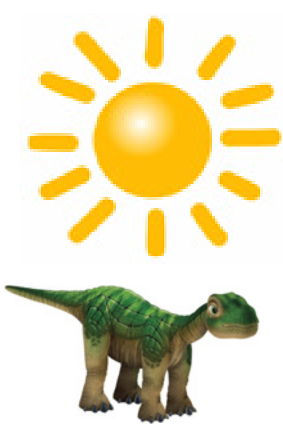
  - Sexual reproduction (offspring have combinations of features inherited from each parent)



Images adapted from http://www.wpclipart.com

Evolution in Biology II

- Some offspring survive, and produce next generations, and some don't:
  - The organisms adapted to the environment better have higher chance to survive
  - Over time, the generations become more and more adapted because the fittest organisms survive



Images adapted from http://www.wpclipart.com

The Genetic Algorithms

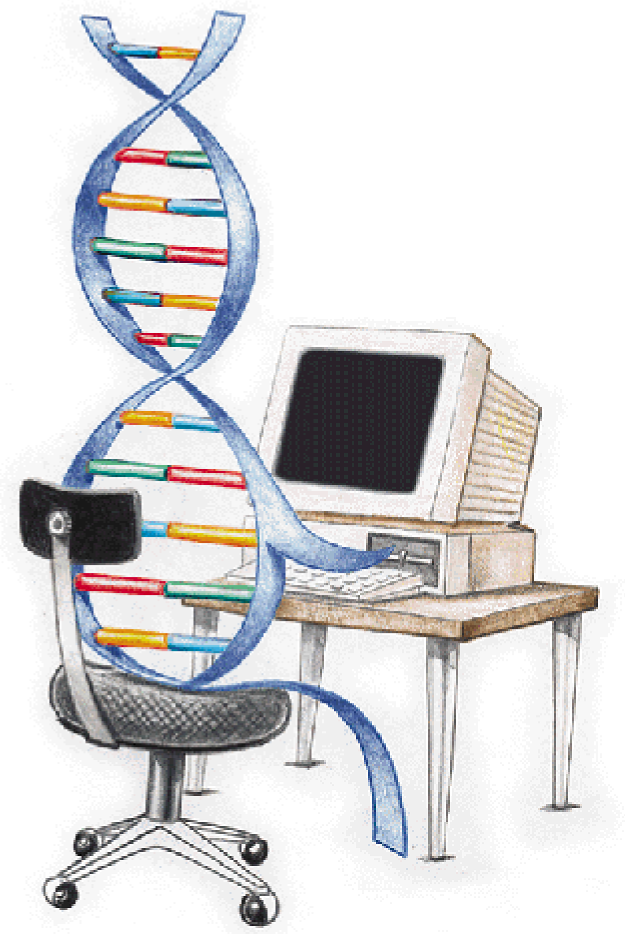
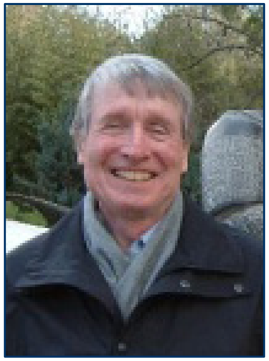


Image from http://www.genetic-programming.org

The Genetic Algorithms (GA)

- Based on the mechanics of biological evolution
- Initially developed by John Holland, University of Michigan (1970's)
  - To understand processes in natural systems
  - To design artificial systems retaining the robustness and adaptation properties of natural systems



- Holland's original GA is known as the simple genetic algorithm (SGA)
- Provide efficient techniques for optimization and machine learning applications
- Widely used in business, science and engineering

Image adapted from http://today.mun.ca/news.php?news_id=2376

Genetic Algorithms Techniques

- GAs are a particular class of evolutionary algorithms. The techniques common to all GAs are:
  - Inheritance
  - Mutation
  - Selection
  - Crossover (also called recombination)
- GAs are best used when the objective function is:
  - Discontinuous
  - Highly nonlinear
  - Stochastic
  - Has unreliable or undefined derivatives

Performance

- GAs can provide solutions for highly complex search spaces
- GAs perform well approximating solutions to all types of problems because they do not make any assumption about the underlying fitness landscape (the shape of the fitness function, or objective function)
- However, GAs can be outperformed by more field-specific algorithms

Biological Terminology

- Gene – a single encoding of part of the solution space, i.e. either single bits or short blocks of adjacent bits that encode an element of the candidate solution

1

- Chromosome – a string of genes that represents a solution

0	1	0	1	1

- Population – the number of chromosomes available to test

0	1	0	1	1
1	1	1	1	1

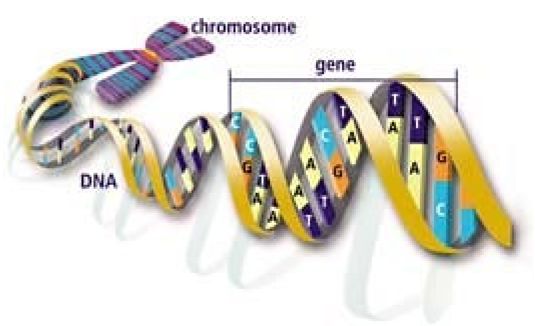
1	1	0	1	1
1	0	0	1	1

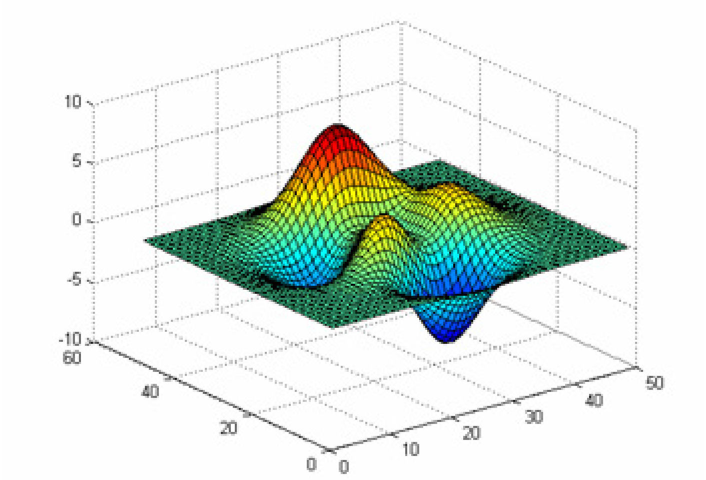
1	1	0	0	1
0	1	0	1	0

0	1	0	0	0
1	1	0	1	0

Biology vs. Optimization

- Candidate solutions to the optimization problem play the role of individuals in a population (or chromosomes)
- Cost/fitness/objective function determines the environment within which the solutions “live”





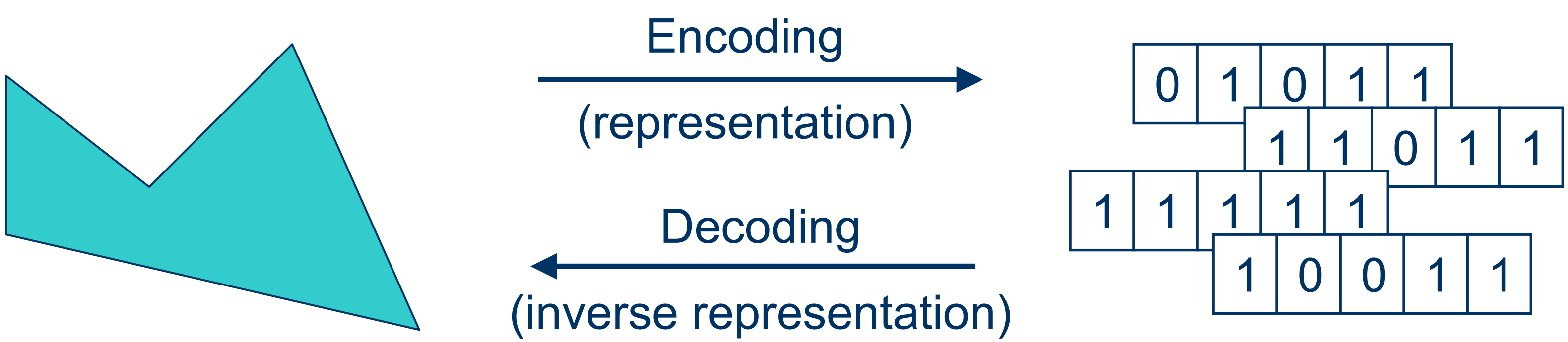
DNA image from http://static.howstuffworks.com/gif/cell-dna.jpg

Features of Genetic Algorithms

- Not too fast but cover large search space
  - Capable of quickly finding promising regions of the search space but may take a relatively long time to reach the optimal solution. Solution: hybrid algorithms
- Good heuristics for combinatorial problems
- Usually emphasize combining information from good parents (crossover)
- Different GAs use different
  - Representations
  - Mutations
  - Crossovers
  - Selection mechanisms

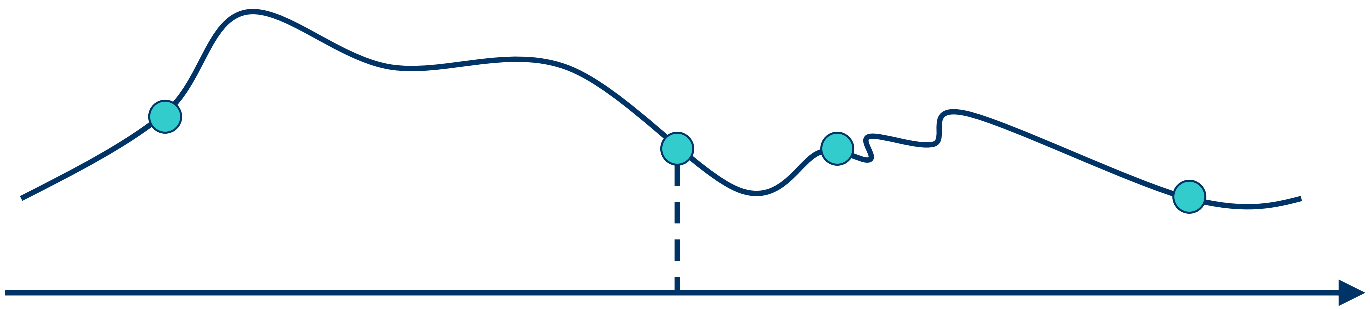
Representation

- Chromosomes can be:
  - Bit strings (0110, 0011, 1101, …)
  - Real numbers (33.2, -12.11, 5.32, …)
  - Permutations of element (1234, 3241, 4312, …)
  - Lists of rules (R1, R2, R3, … Rn …)
  - Program elements (genetic programming)
  - Any other data structure



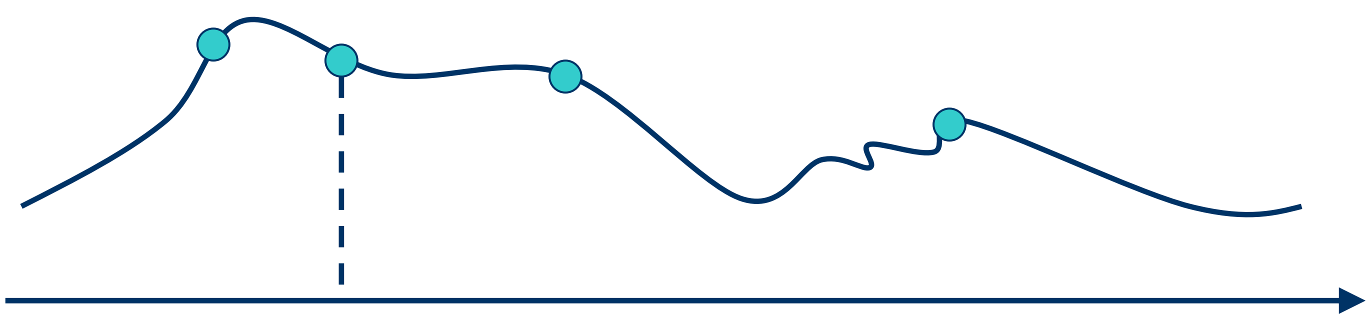
Representation Example

- Generation 0: 4 candidate solutions



34.57 ~ “1000010101101”

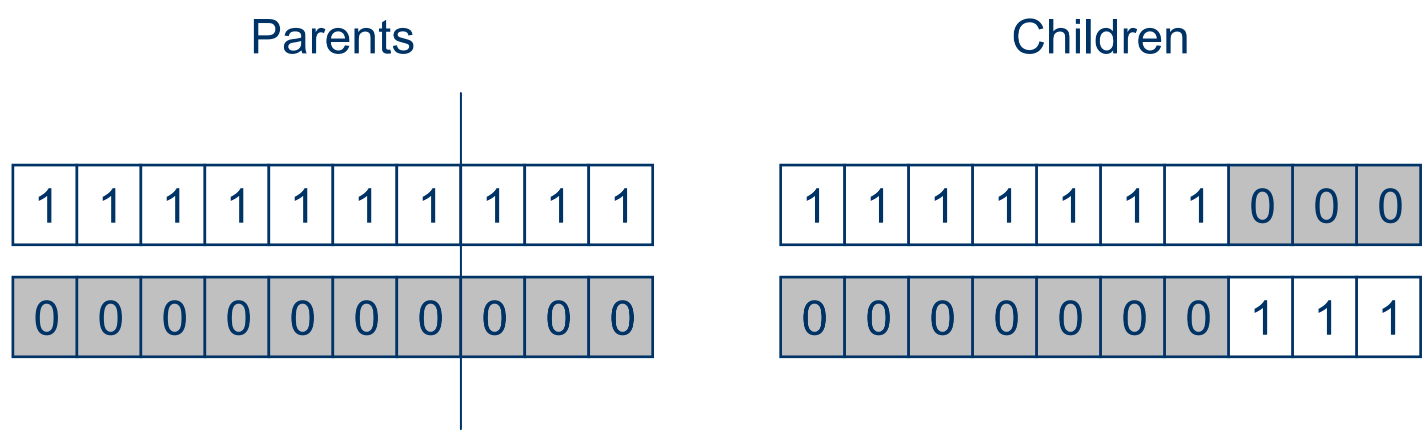
- Generation N



23.99 ~ “1000001111110”

1-Point Crossover

- Choose a random point
- Split parents at this crossover point
- Create children by exchanging tails
- Probability of crossover is typically in range (0.6, 0.9)



Other Crossover Types

- Two-point crossover

Parents

1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0

Children

0	0	1	1	1	1	1	0	0	0
1	1	0	0	0	0	0	1	1	1

- Uniform crossover: randomly generated mask

Mask:

1	1	0	1	0	0	0	1	1	0

Parents

1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0

Children

0	0	1	0	1	1	1	0	0	1
1	1	0	1	0	0	0	1	1	0

Mutation

- Alter each gene independently
- Mutation probability is typically in range (1/population_size, 1/chromosome_length)

Parent									
1	1	1	1	1	1	1	1	1	1

(-4.32 2.12 [-41.56] 9.99)

Child									
1	1	0	1	1	1	1	0	1	0

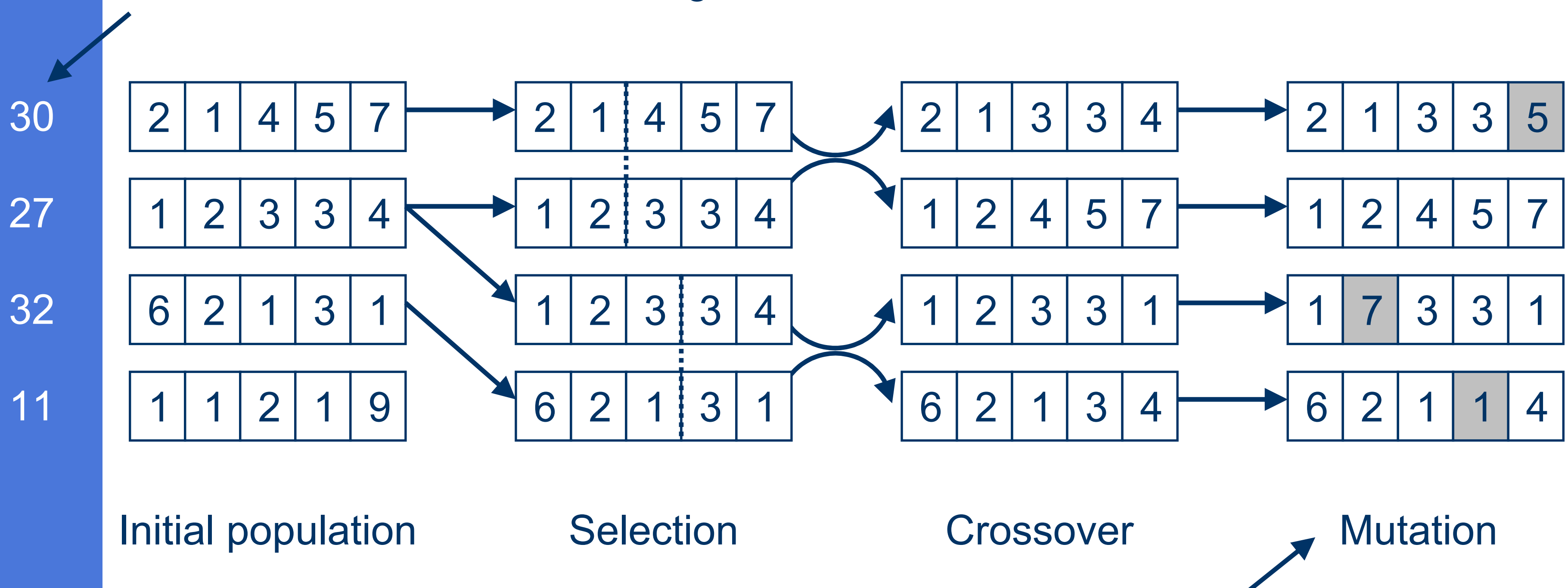
(-4.32 2.12 [-43.15] 9.99)

- Choose mutation with the best fit

Selection

- Parents with better fitness have better chances to produce offspring

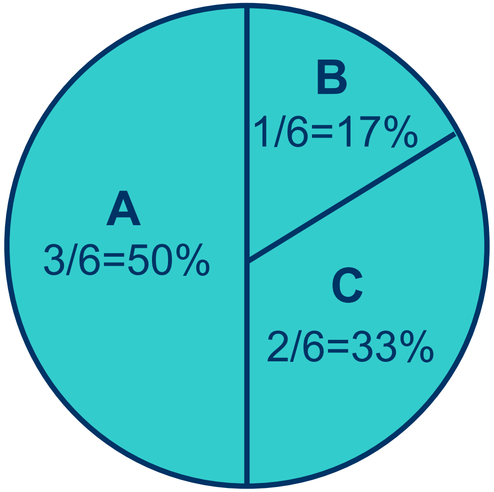
Fitness function – measure of goodness of the candidate solution



Mutation can be performed in a way that maximizes fitness function

Selection: Roulette Wheel

- Better solutions get higher chance to become parents for next generation solutions
- Roulette wheel technique:
  - Assign each individual part of the wheel
  - Spin wheel N times to select N individuals



fitness(A) = 3

fitness(B) = 1

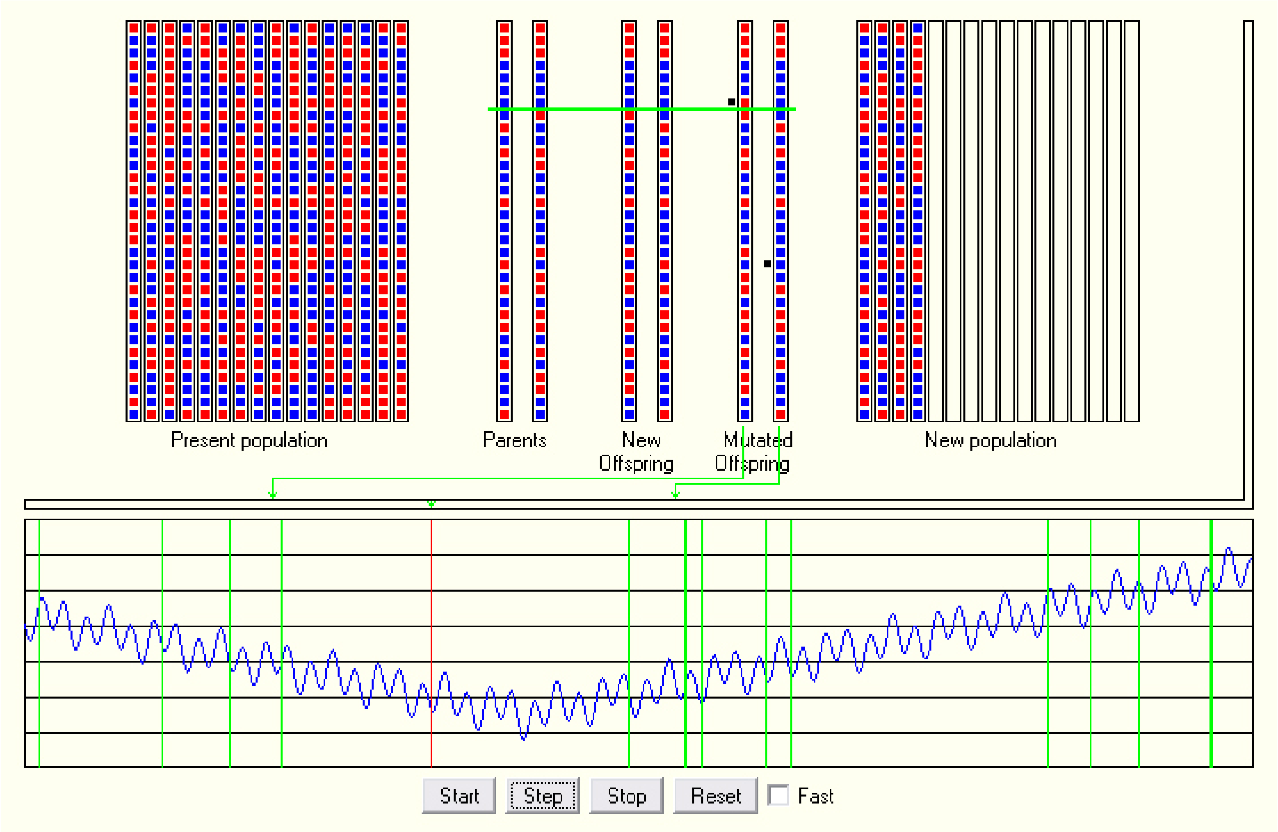
fitness(C) = 2

The Basic Genetic Algorithm

```
{   % Generate random population of chromosomes
    Initialize population;
    % Evaluate the fitness of each chromosome in the population
    Evaluate population;                                [Fitness]
    % Create, accept, and test a new population:
    while Termination_Criteria_Not_Satisfied
    {       % Select according to fitness
            Select parents for reproduction;            [Selection]
            % With a crossover probability perform crossover or copy parents
            Perform crossover;                          [Crossover]
            % With a mutation probability mutate offspring at each position in chromosome
            Perform mutation;                           [Mutation]
            Accept new generation;
            Evaluate population;                        [Fitness]
    }
}
```

Demo Java Applet

- A Java applet demonstrating genetic algorithm in action can be found at: http://cs.felk.cvut.cz/~xobitko/ga/



Benefits of Genetic Algorithms

- Concept is easy to understand
- Modular – separate from application (representation); building blocks can be used in hybrid applications
- Supports multi-objective optimization
- Good for "noisy" environment
- Always results in an answer, which becomes better and better with time
- Can easily run in parallel
- The fitness function can be changed from iteration to iteration, which allows incorporating new data in the model if it becomes available

Issues with Genetic Algorithms

- Choosing parameters:
  - Population size
  - Crossover and mutation probabilities
  - Selection, deletion policies
  - Crossover, mutation operators, etc.
  - Termination criteria
- Performance:
  - Can be too slow but covers a large search space
  - Is only as good as the fitness function

Applications

Image from http://www.uni-duisburg-essen.de/imperia/md/images/tul/lehre_vorschlag_studarbeiten_ctss_karte.gif

Applications of Genetic Algorithms

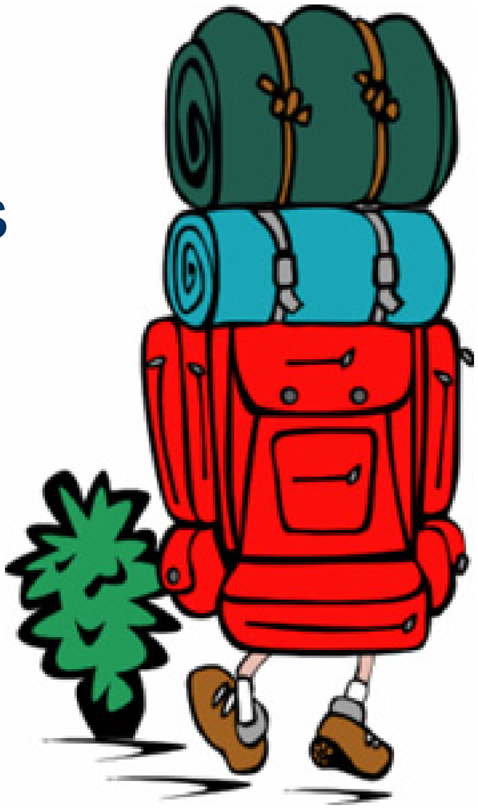
- **Optimization** – numerical and combinatorial optimization problems, e.g. traveling salesman, routing, graph colouring and partitioning
- **Robotics** – trajectory planning
- **Machine learning** – designing neural networks, classification and prediction, e.g. prediction of weather or protein structure,
- **Signal processing** – filter design
- **Design** – semiconductor layout, aircraft design, communication networks
- **Automatic programming** – evolve computer programs for specific tasks, design cellular automata and sorting networks
- **Economics** – development of bidding strategies, emergence of economics markets
- **Immune systems** – model somatic mutations
- **Ecology** – model symbiosis, resource flow
- **Population genetics** – “Under what condition will a gene for recombination be evolutionarily viable?”

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



*An example including results is adapted from [1]

Image from http://www.wpclipart.com

Representation

- Representation is an ordered list of city numbers:
  1. Kyiv
  2. Budapest
  3. Teheran
  4. Beijing
  5. Jerusalem
  6. Bucharest
  7. Hamilton
  8. Toronto
- Possible city lists as candidate solutions:
  - CityList1 (3 5 7 2 1 6 4 8)
  - CityList2 (2 5 7 6 8 1 3 4)

Crossover

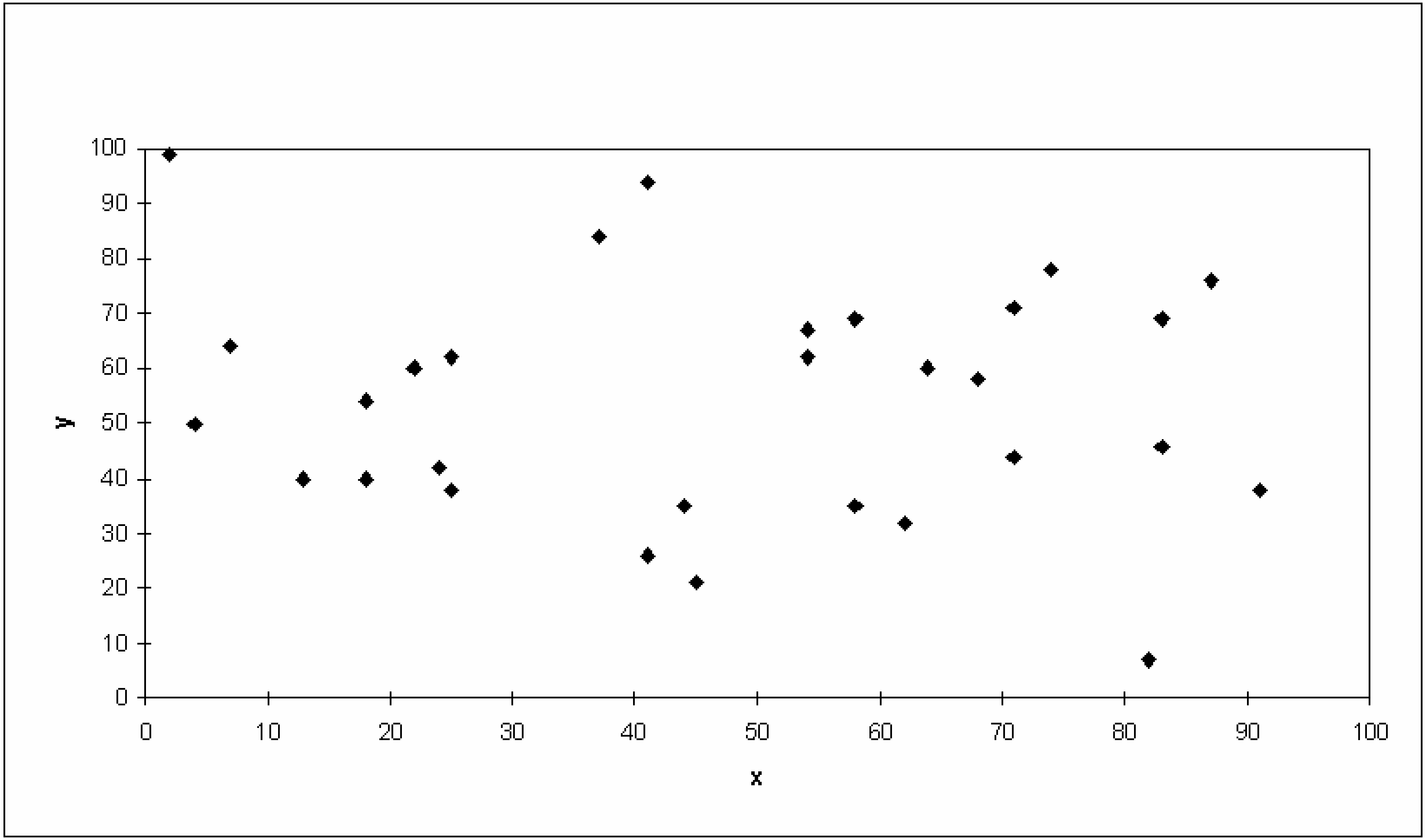
Parent1	(3	5	7	2	1	6	4	8)
Parent2	(2	5	7	6	8	1	3	4)
Child	(5	8	7	2	1	6	3	4)

Mutation

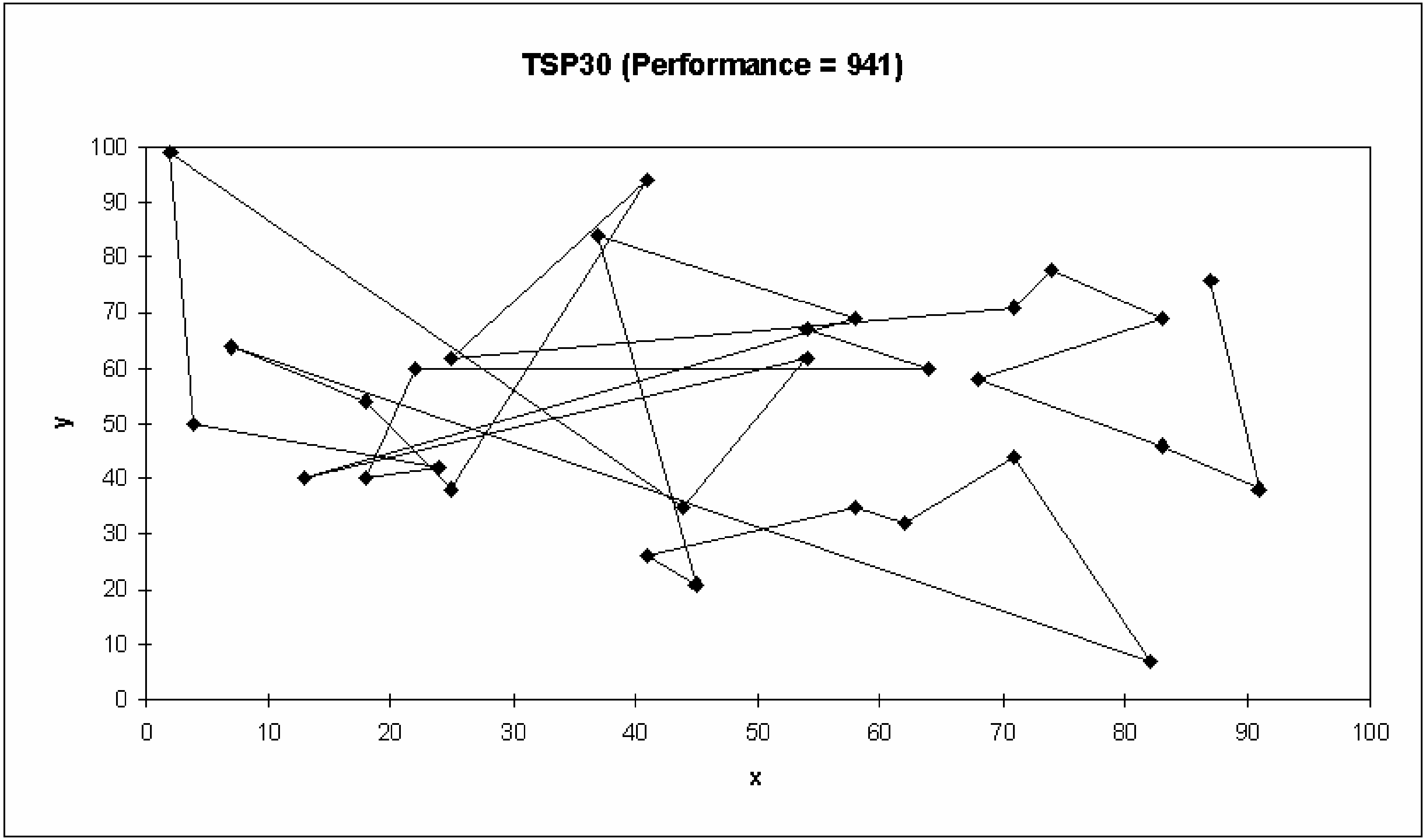
Mutation involves reordering of the list:

			*			*		
Before:	(5	8	7	2	1	6	3	4)
After:	(5	8	6	2	1	7	3	4)

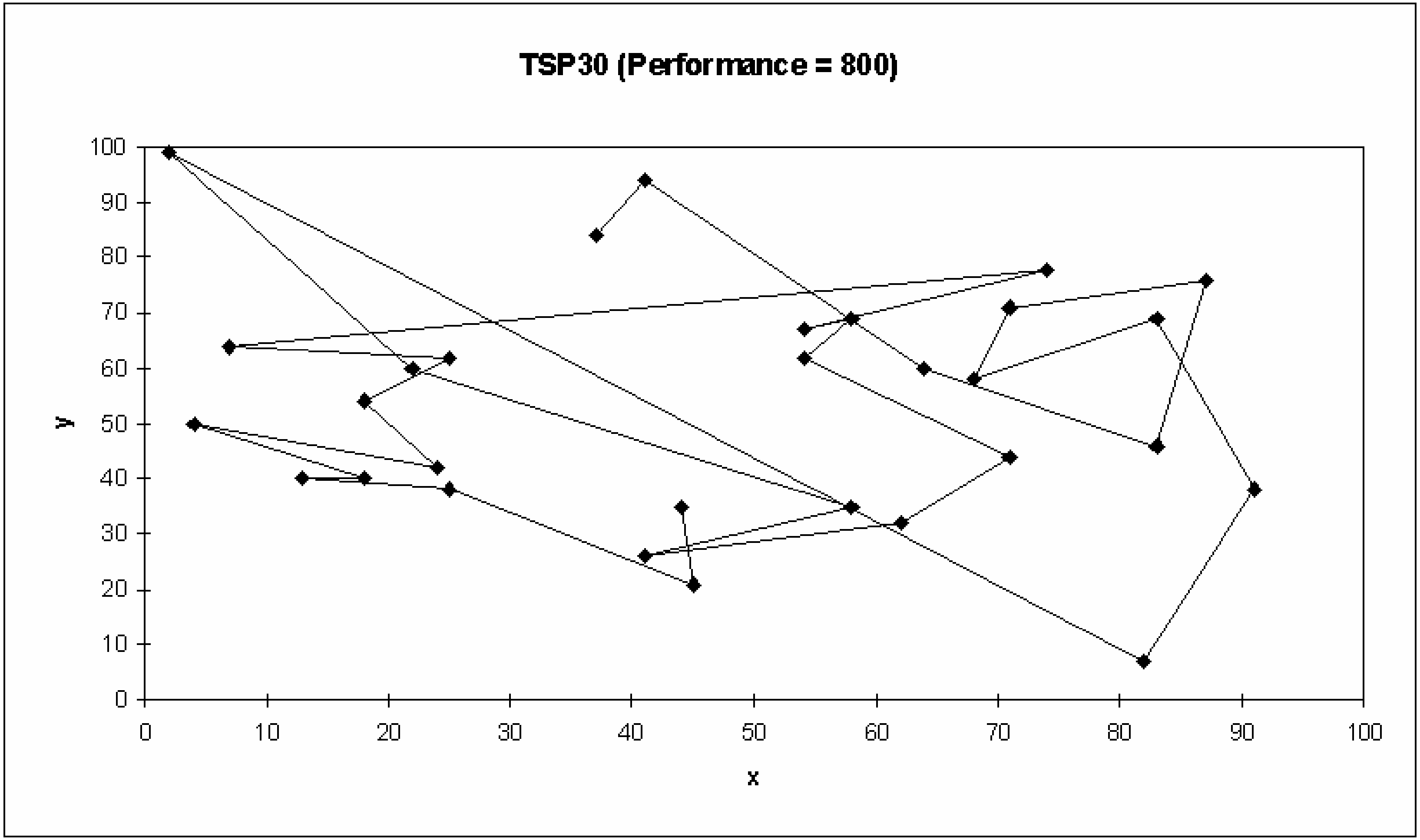
TSP Example: 30 Cities



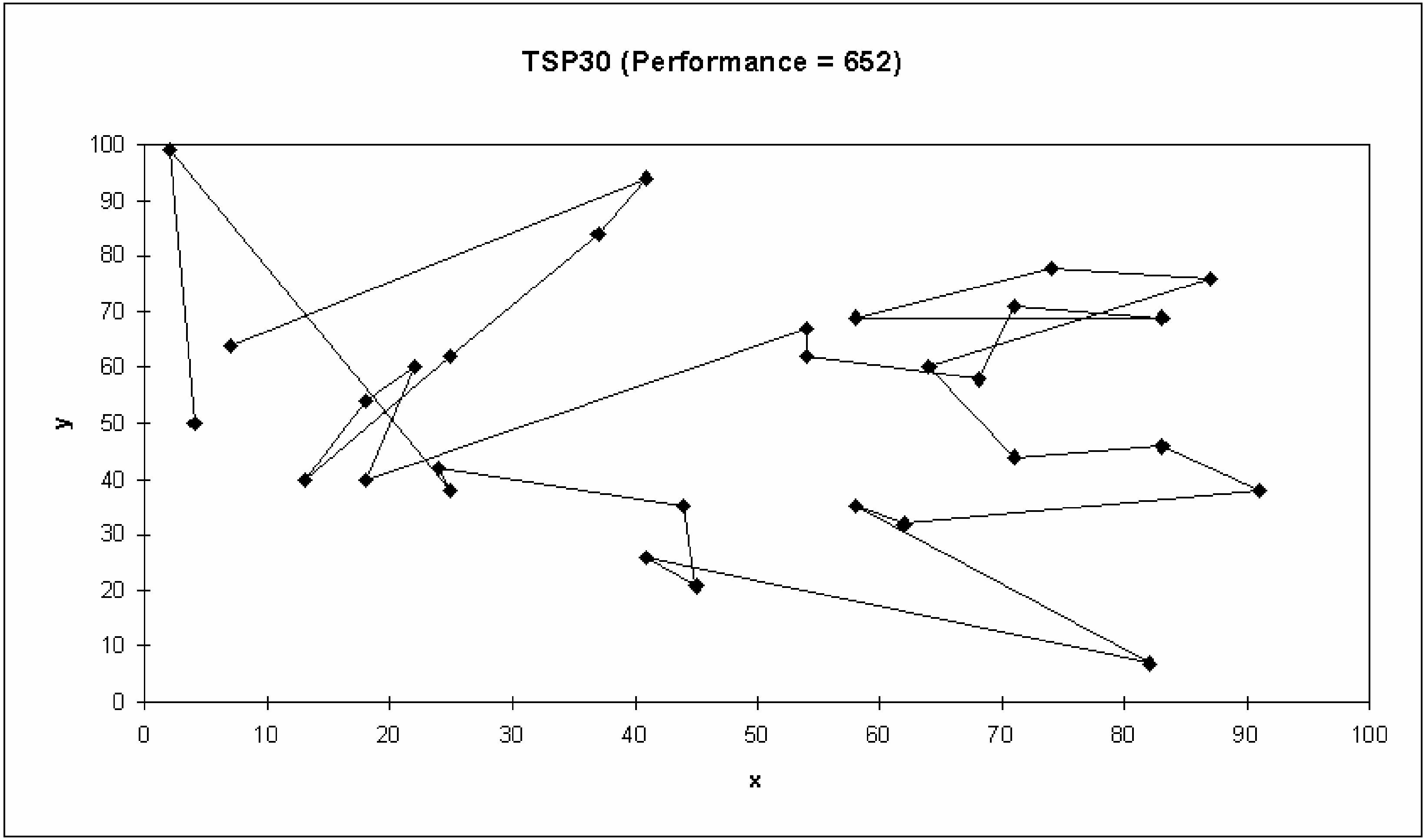
Solution i (Distance = 941)



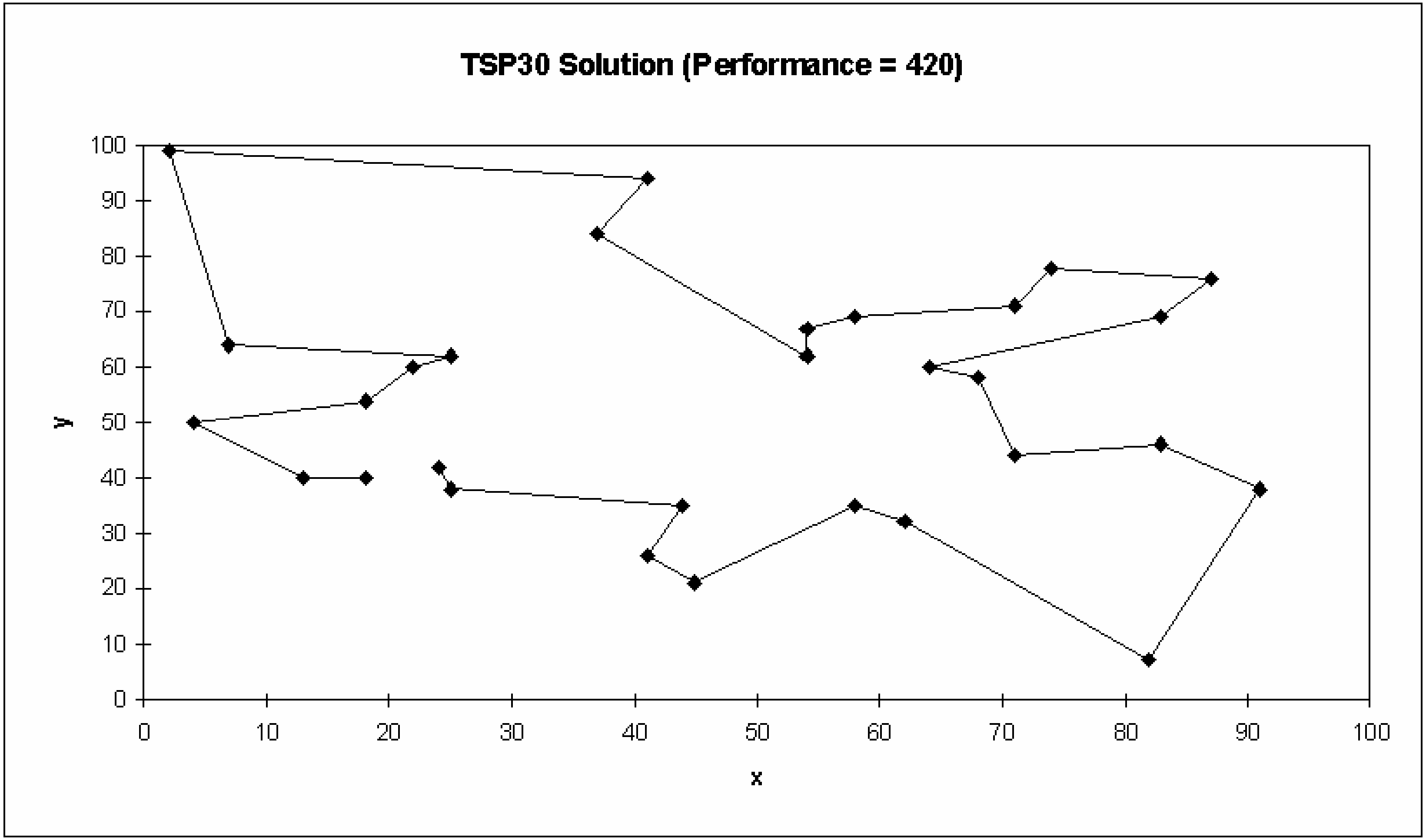
Solution j (Distance = 800)



Solution k (Distance = 652)



Best Solution (Distance = 420)



Genetic Algorithms References

1. W. Williams, Genetic Algorithms: A Tutorial, http://web.umr.edu/~ercal/387/slides/GATutorial.ppt
2. A. Eiben, J. Smith, Introduction to Evolutionary Computing, Genetic Algorithms, http://www.cs.vu.nl/~jabekker/ec0607/slides/Lecture03-Chapter3-GeneticAlgorithms.ppt
3. R. Horst and P.M. Pardalos (eds.), Handbook of Global Optimization, Kluwer, Dordrecht 1995.
4. M. Mitchell, An Introduction To Genetic Algorithms, Cambridge, MA: MIT Press, 1996.

Software

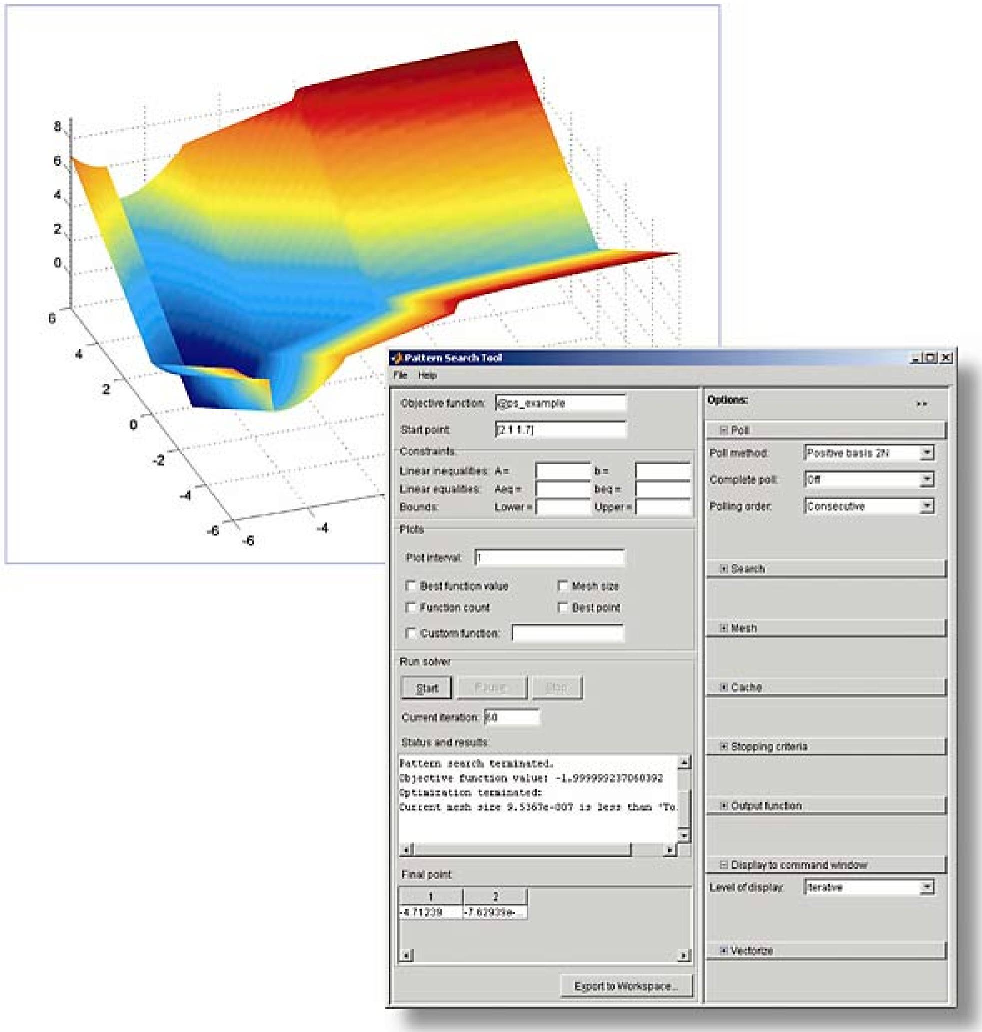


Image from http://www.mathworks.com/company/pressroom/articles/article7017.html

Genetic Algorithm Software

- **GENOCOP III** – Genetic Algorithm for Constrained Problems in C (by Zbigniew Michalewicz)
- **DE** – Differential Evolution Genetic Algorithm in C and Matlab (by Rainer Storn). DE has won the third place at the 1st International Contest on Evolutionary Computation on a real-valued function testsuite
- **PGAPack** – Parallel Genetic Algorithm in Fortran and C (from Argonne National Laboratory)
- **PIKAIA** – Genetic algorithm in Fortran 77/90 (by Charbonneau, Knapp and Miller)
- **GAGA** – Genetic Algorithm for General Application in C (by Ian Poole)
- **GAS** – Genetic Algorithm in C++ (by Jelasity and Dombi)
- **GAlib** – C++ Genetic Algorithm Library (by Matthew Wall)
- **Genetic Algorithm in Matlab** (by Michael B. Gordy)
- **GADS** – Genetic Algorithm and Direct Search Toolbox in Matlab (from MathWorks)
- **GEATbx** – Genetic and Evolutionary Algorithm Toolbox for Matlab (by Hartmut Pohlheim)
- **GAOT** – Genetic Algorithms Optimization Toolbox in Matlab (by Jeffrey Joines)

MathWorks GADS Toolbox

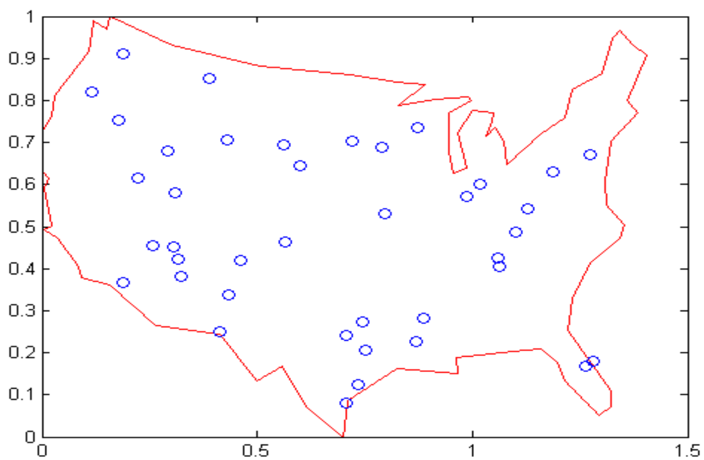
- The MathWorks offers Genetic Algorithm and Direct Search Toolbox 2.0.2
- GUI (gatool) and command line tools for setting up problems, algorithm options, and monitoring progress
- GA tools, such as creation of initial population, fitness scaling, parent selection, crossover and mutation
- Ability to solve optimization problems with nonlinear, linear, and bound constraints
- Support for automatic M-code generation

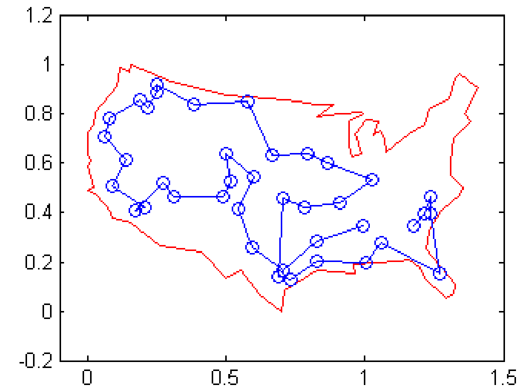
Prices

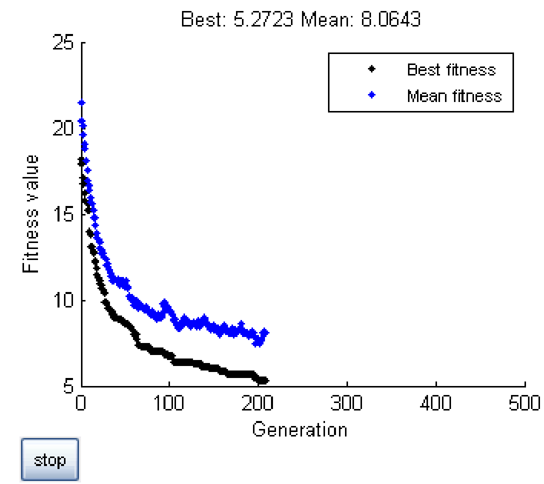
- No prices are quoted on the website for this toolbox, have to ask for a quote
  - Commercial
  - Academic
  - Student use
- Academic use, individual license: $200 U.S.
- Press release: “The Genetic Algorithm and Direct Search Toolbox requires MATLAB and the Optimization Toolbox and is available immediately for Windows, UNIX/Linux, and Macintosh systems. Pricing starts at $700 U.S.”
- http://www.mathworks.com/products/gads/index.html

Demo 1

- The traveling salesman problem gadsdemos\traveling_salesman_demo1.m:





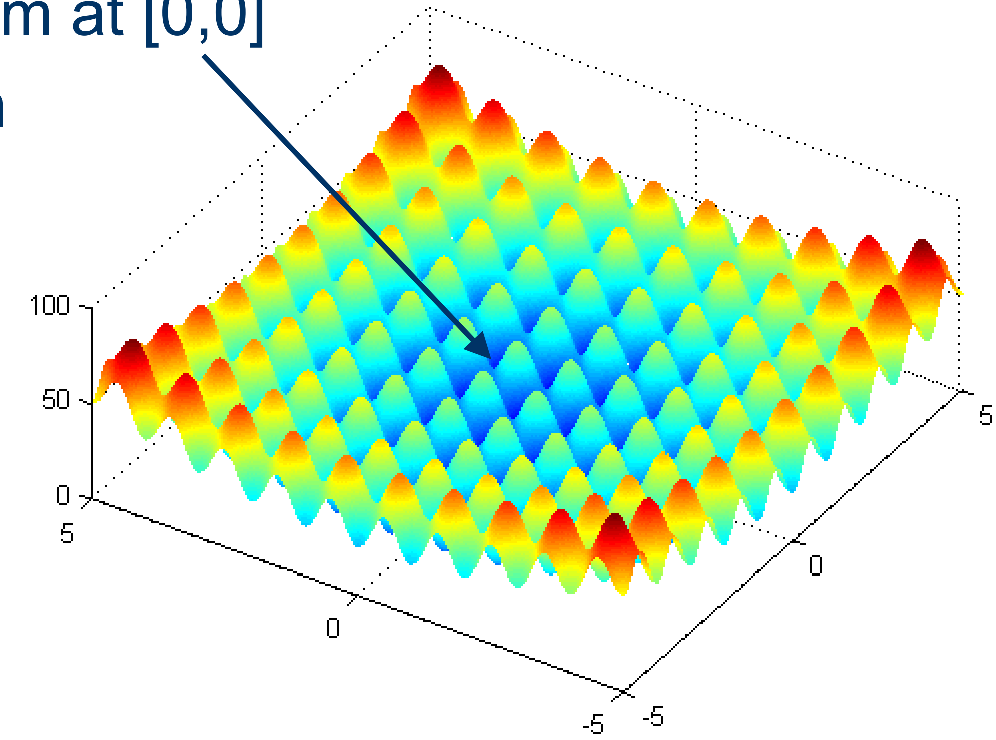


Demo 2: Function

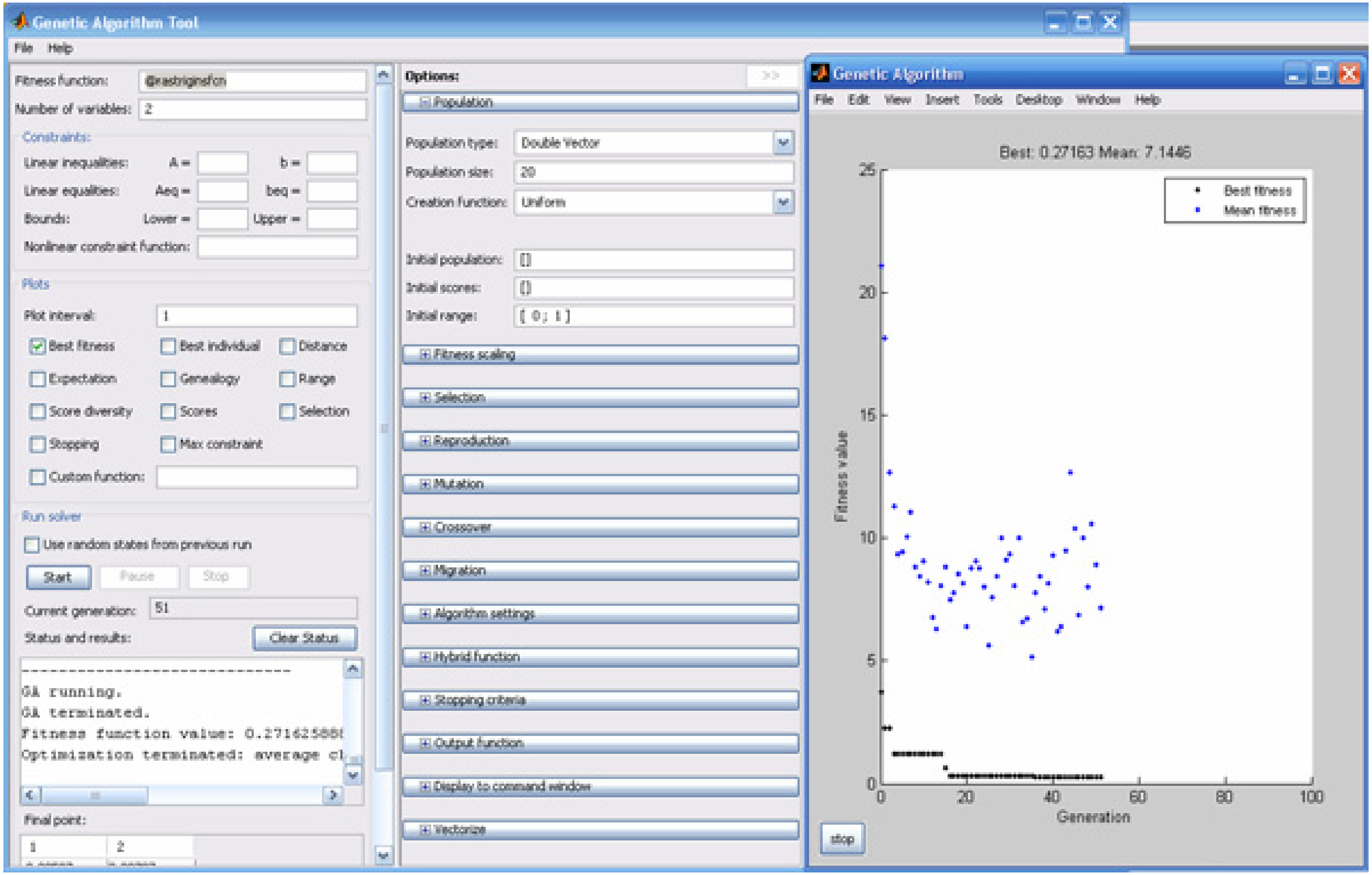
- Rastrigin's function :

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(cos2\pi x_1 + cos2\pi x_2)$$

- Global minimum at [0,0]
- rastriginsfcn.m



Demo 2: gatool

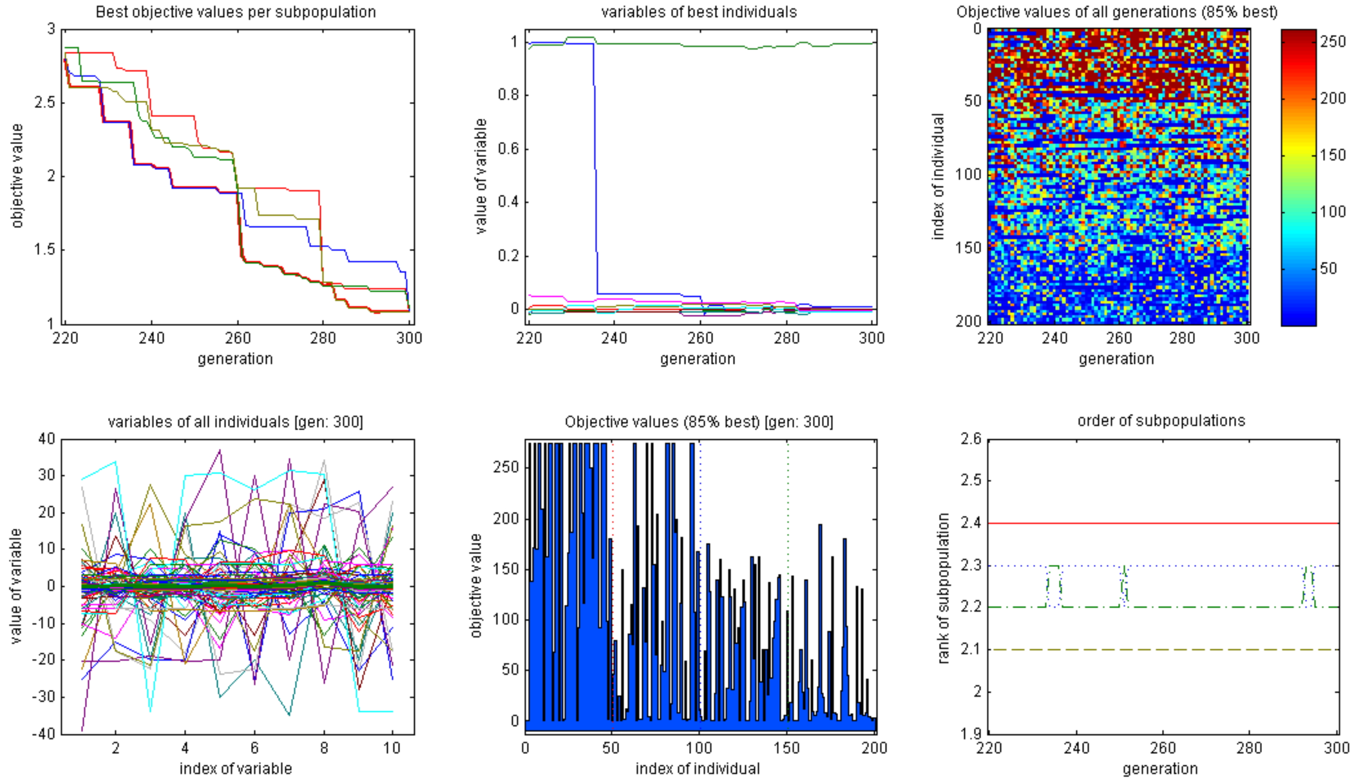


GEATbx: Features

- Broad class of operators for selection, crossover, mutation operations
- Complete support of multi-objective optimization
- Supplied with population models
- Real, integer, and binary variable representation
- Sophisticated visualization
- Comfortable monitoring and storing of results
- The GEATbx can be completely compiled into C/C++ code using the Matlab Compiler
- Author: Hartmut Pohlheim
- Can be bought at: http://www.geatbx.com/order.html
- Evaluation can be ordered

Demo

- demogeatbx.m



Prices

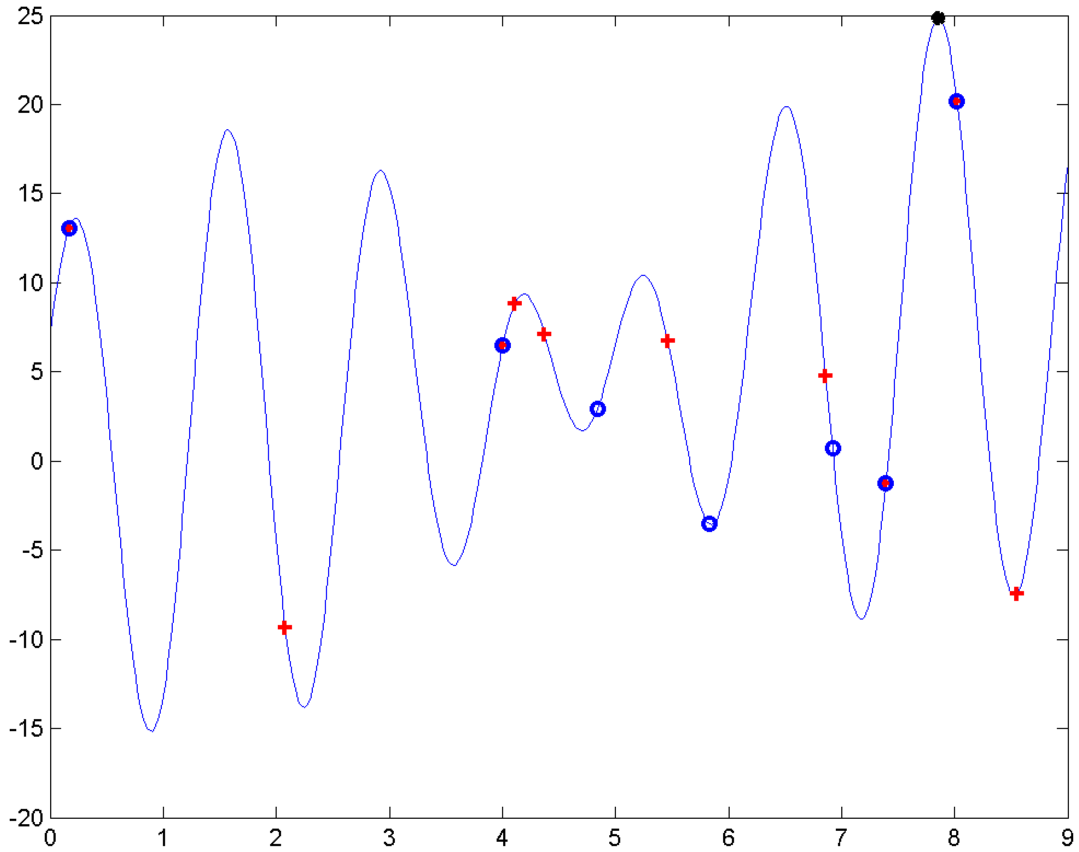
License	Price per license	Purchase at ShareIT
single user license (corporation)	400 Euro* 476 Euro incl. 19% VAT	Buy **single user** license (corporation) now.
single user license (students, university)	250 Euro* 297,50 Euro incl. 19% VAT	Buy single user license (students, university) now.
classroom license (university) info	250 Euro* 297,50 Euro incl. 19% VAT	Buy **classroom** license (university) now.
site license (corporation) info	2000 Euro* 2380 Euro incl. 19% VAT	Buy **site** license (corporation) now.
site license (students, university)	1250 Euro* 1487,50 Euro incl. 19% VAT	Buy **site** license (university) now.
multiple user licenses:		
2-3 copies (corporation)	300 Euro* 357 Euro incl. 19% VAT	Buy **multiple user** licenses (corporation) now.
4-5 copies (corporation)	240 Euro* 285,60 Euro incl. 19% VAT	
6-9 copies (corporation)	200 Euro* 238 Euro incl. 19% VAT	
multiple user licenses:		
2-3 copies (students, university)	190 Euro* 226,10 Euro incl. 19% VAT	Buy **multiple user** licenses (students, university) now.
4-5 copies (students, university)	150 Euro* 178,50 Euro incl. 19% VAT	
6-9 copies (students, university)	125 Euro* 148,75 Euro incl. 19% VAT	

GAOT – GA Optimization Toolbox

- Can be downloaded for free at: http://www.ise.ncsu.edu/mirage/GAToolBox/gaot/
- Author: Jaffrey Joines
- Real, binary, and order-based epresentations
- Paper (the same website):
  - Description of the toolbox
  - Numerical results:
    - Float genetic algorithm (FGA) outperformed binary GA (BGA) and simulated annealing (SA) algorithms in computational efficiency and solution quality
    - Corana testset: family of parameterized functions, very simple to compute and contain large number of local minima (reminds n-dimensional parabola)

Demo

- gademo1.m



GA Software References

1. Arnold Neumaier, Global Optimization Software, http://www.mat.univie.ac.at/~neum/glopt/software_g.html#ga_codes
2. GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with MATLAB, http://www.geatbx.com/links/ea_matlab.html
3. The MathWorks, Genetic Algorithm and Direct Search Toolbox, http://www.mathworks.com/products/gads/
4. GAOT – Genetic Algorithm Optimization Toolbox http://www.ise.ncsu.edu/mirage/GAToolBox/gaot/