Università degli Studi di Milano

# *Evolutionary Algorithms*

Enrique Muñoz Ballester

Dipartimento di Informatica
via Bramante 65, 26013 Crema (CR), Italy
enrique.munoz@unimi.it

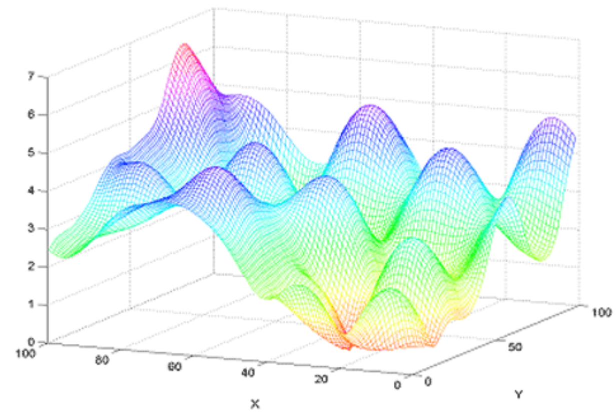
# Material

- Download slides data and scripts:

https://homes.di.unimi.it/munoz/teaching.html

# Mathematical optimization

- In general, optimization is the problem of finding the (global) minimum (or maximum) of an objective function
- Usually, but not always, the objective function is a specific, well-defined mathematical function
- $F=f(x_1,x_2,x_3,\ldots,x_n)$



# Evolutionary Algorithms

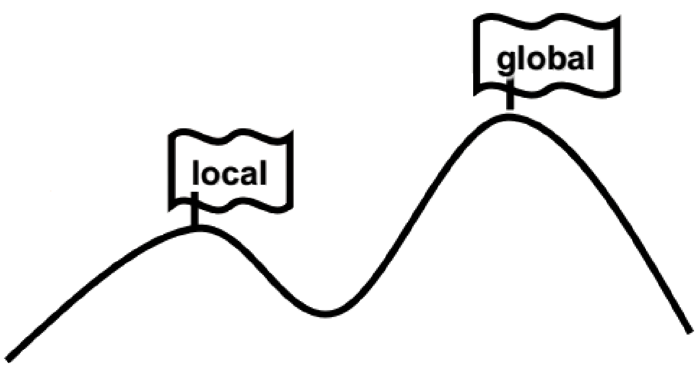
- ‘Evolutionary Algorithms’ (EA) constitute a collection of methods that originally have been developed to solve combinatorial optimization problems
- They adapt Darwinian principles to automated problem solving. Nowadays, Evolutionary Algorithms is a subset of Evolutionary Computation that itself is a subfield of Artificial Intelligence / Computational Intelligence
- Evolutionary Algorithms are those metaheuristic optimization algorithms from Evolutionary Computation that are population-based and are inspired by natural evolution. Typical ingredients are:
  - A population (set) of individuals (the candidate solutions)
  - A problem-specific fitness (objective function to be optimized)
  - Mechanisms for selection, recombination and mutation (search strategy)

# Motivation

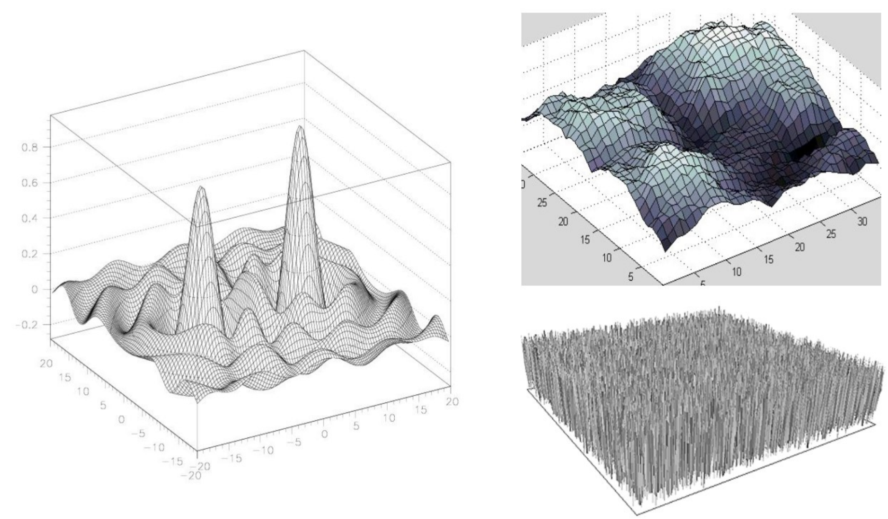
- Why might Evolution be an interesting model for computer algorithms?
  - Evolution has proven a powerful mechanism in ‘improving’ life-forms and forming ever more complex species
  - Driven by surprisingly simple mechanisms, nevertheless produced astonishing results
- Evolution is basically a random process, driven by evolutionary pressure:
  - Tinkering with genes (Genotype)
    - Mating: recombination of genes in descendants
    - Mutation: random changes (external influences, reproduction errors)
  - Testing (Phenotype), Competition (‘Survival of the fittest’)

# Motivation

- EA's are useful for solving multidimensional problems containing many local maxima (or minima) in the solution space



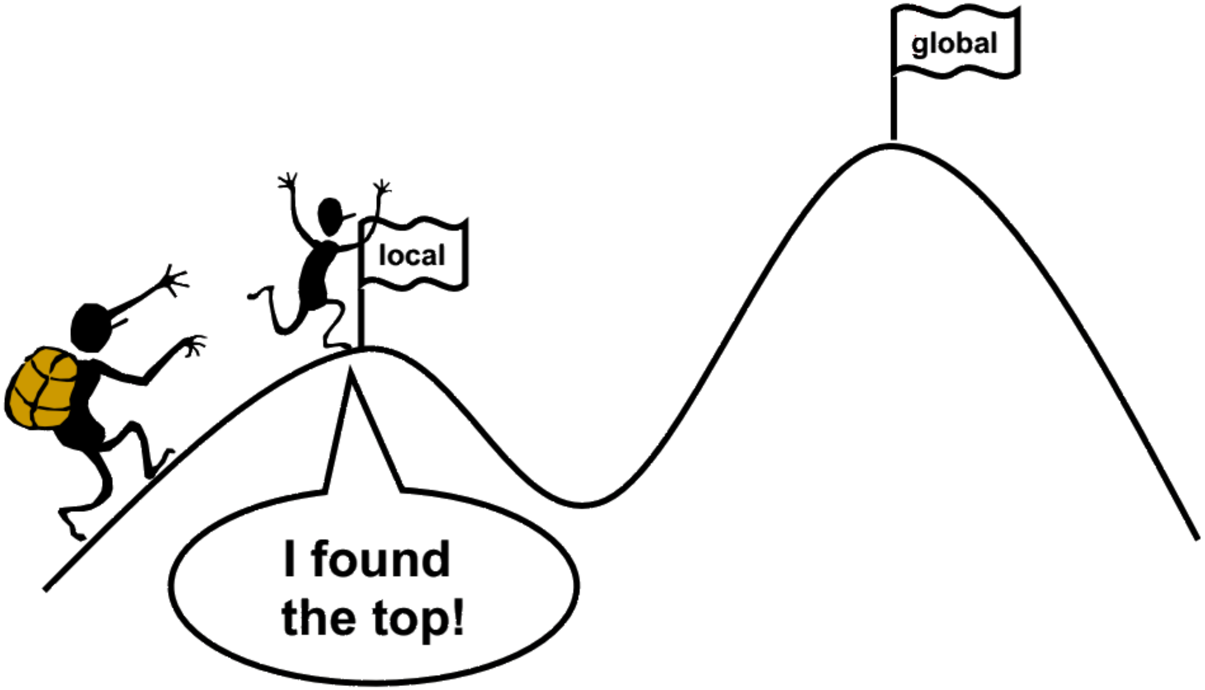

Simple optimization problem



Complex real optimization problems

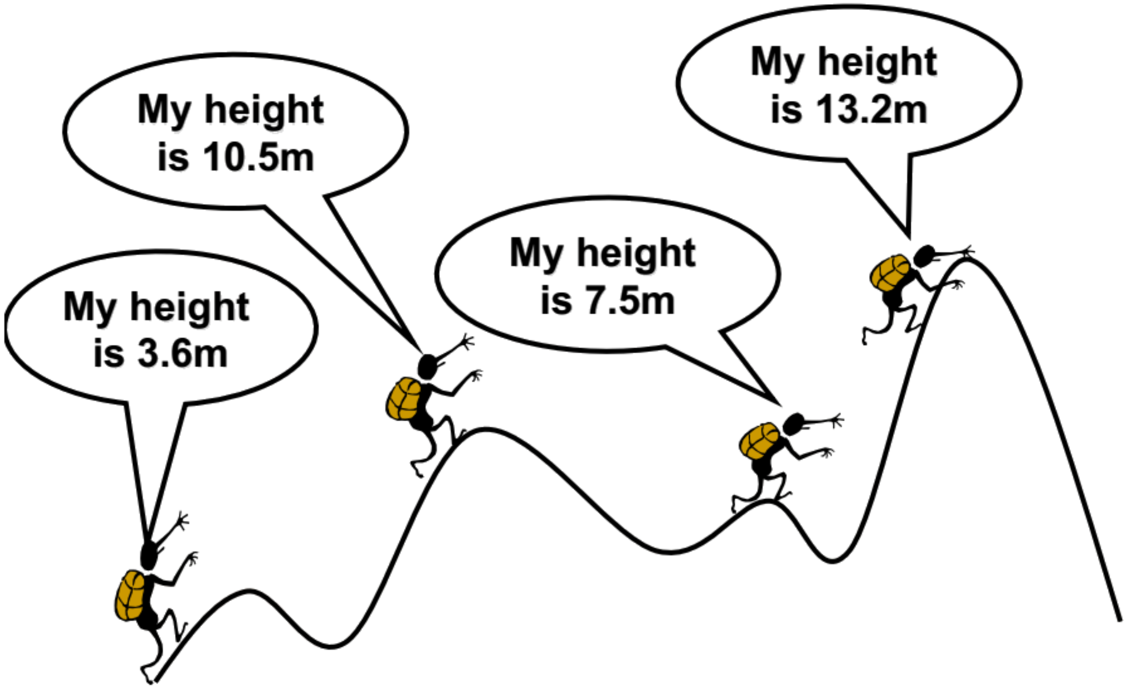
# General idea

- Traditional method (hill climbing, gradient ascent)
- Problem: may find only a local maxima



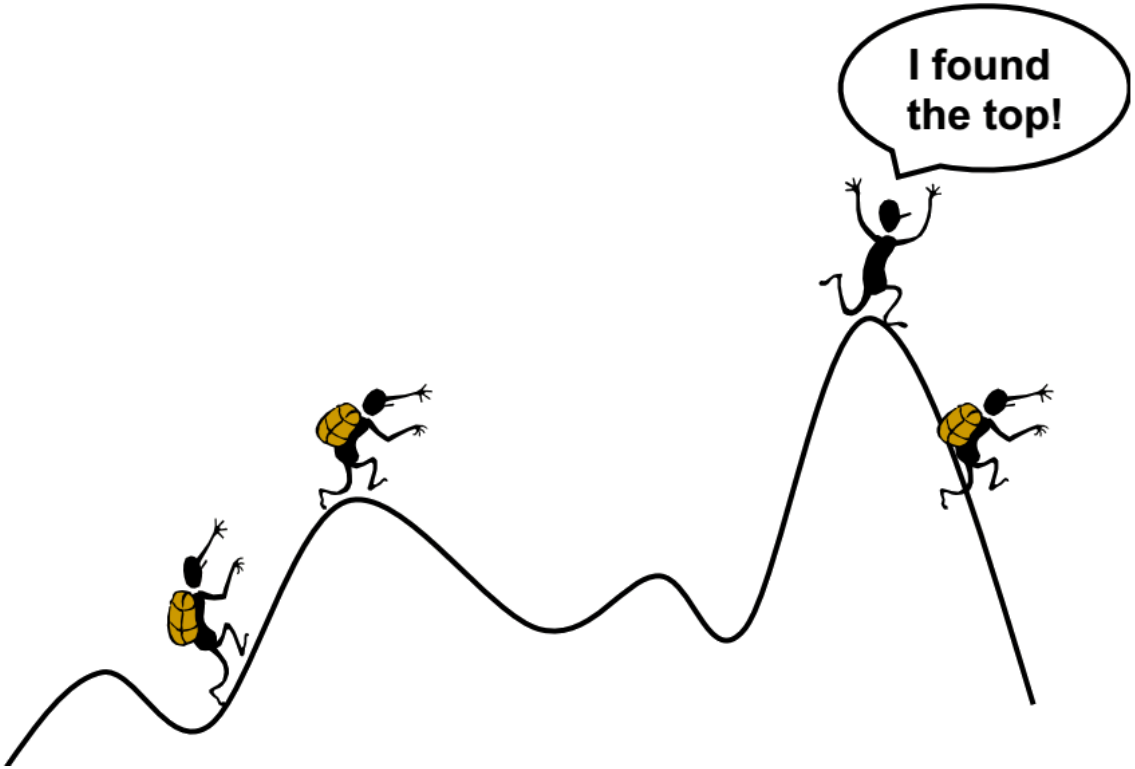
# General idea

- EAs use a population of searchers to find the global optimum



# General idea

- Some iterations later, a searcher has approached the global maximum



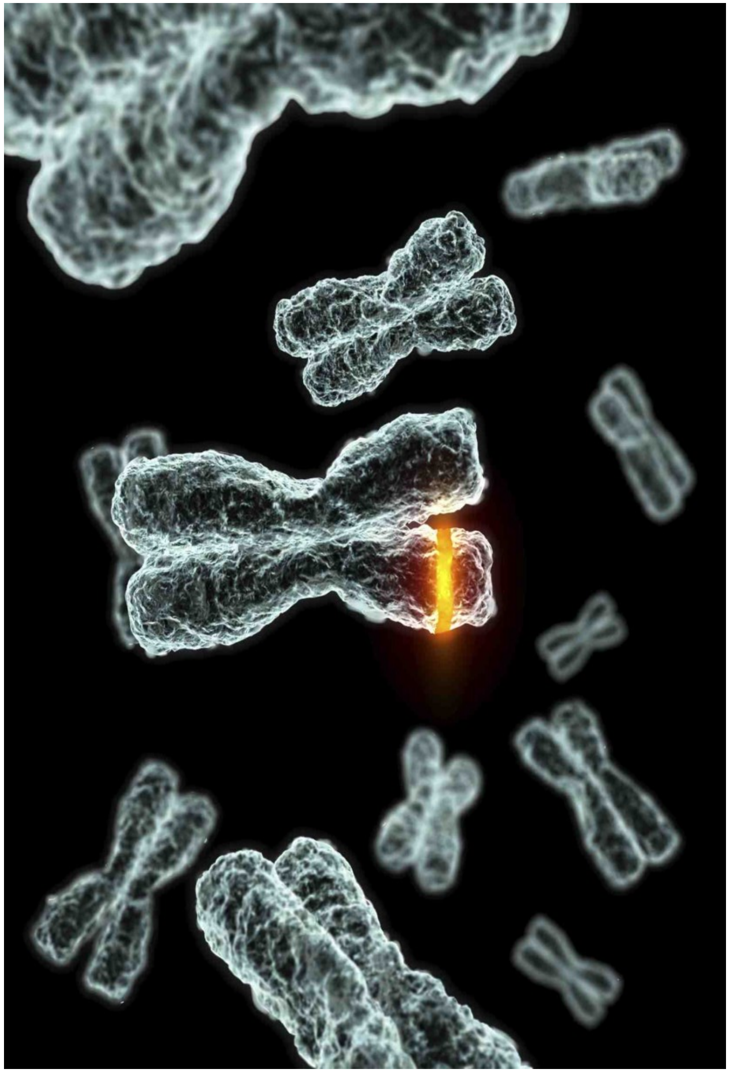
# Evolutionary algorithms: types

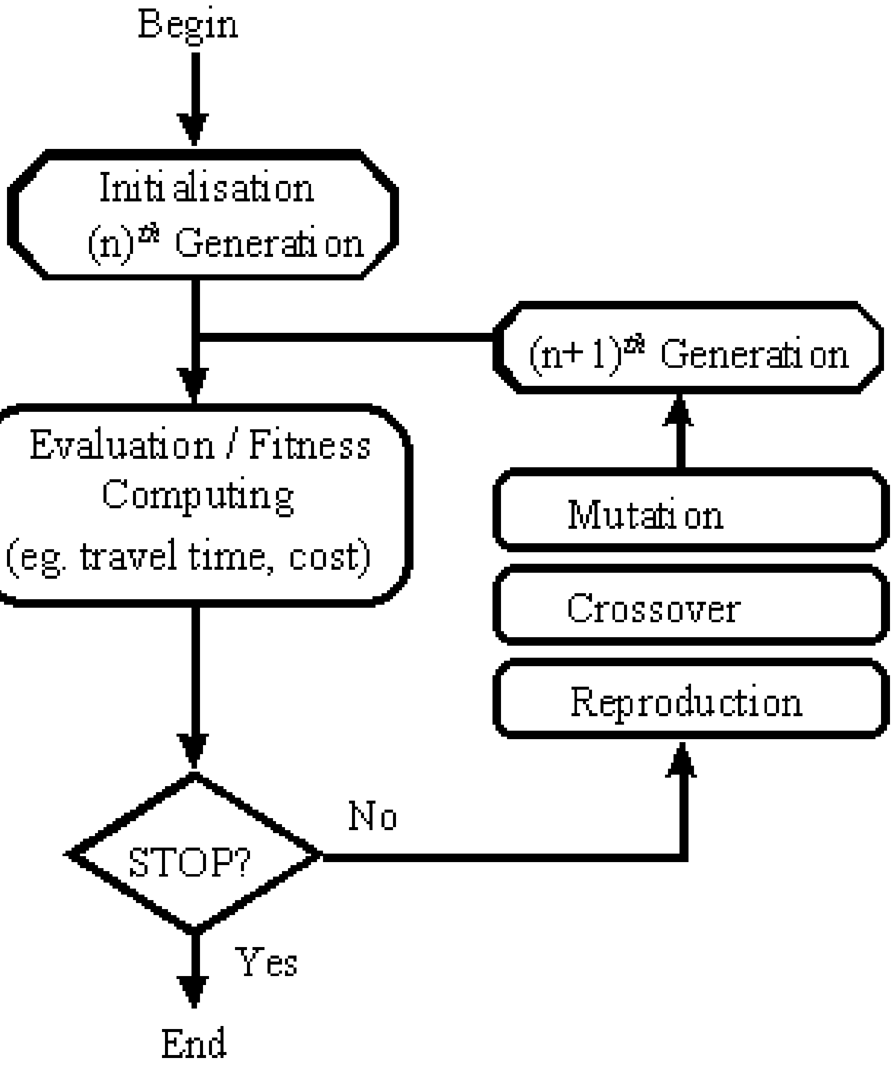
- **Genetic algorithm**
- Genetic programming
- Memetic algorithm
- Differential evolution
- Neuroevolution

# Evolutionary algorithms: applications

- Control
  - Gas pipeline, pole balancing, missile evaluation, pursuit
- Robotics
  - Trajectory planning
- Signal processing
  - Filter design
- Game playing
  - Chess, poker, checker, prisoner’s dilemma
- Scheduling
  - Manufacturing facility, resource allocation
- Design
  - Semiconductor layout, aircraft design, keyboard configuration, communication networks
- Combinatorial optimization
  - Set covering, travelling salesman problem, routing, bin packing, graph coloring or partitioning

# GA: the schema





# Example 1: optimization of a binary function, MAXONE

- Objective: maximize the number of ones in a string of x binary digits, e.g.: x=10
- Gene encoding: string of ***10*** binary digits, e.g., 0110110001
- Fitness function: number of ones in its genetic code, e.g. *f(*0110110001*) = 5*
- Start with a population of ***n*** random binary strings, e.g.: ***n*** = 6

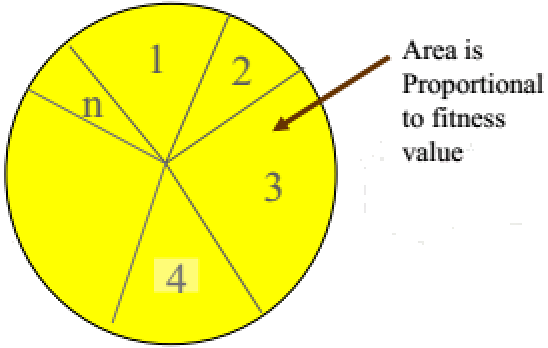
# Example 1: initialization

- Initial population of random parent genes:
  $s1 = 1001011101 \; f(s1) = 6$
  $s2 = 0110100101 \; f(s2) = 5$
  $s3 = 1101110110 \; f(s3) = 7$
  $s4 = 0101000011 \; f(s4) = 4$
  $s5 = 1101111101 \; f(s5) = 8$
  $s6 = 0000110010 \; f(s6) = 3$

# Example 1: selection

- Choose the best parent genes from the current population for breeding a new child population to focus the search in promising regions of the solution space
- Classical: roulette wheel

Individual $i$ will have a $\frac{f(i)}{\sum_i f(i)}$ probability to be chosen



# Example 1: crossover

- Combine two «parents» to obtain new offspring
- Probability to perform crossover $p_{cross}$
- Randomly generate a crossover point to mix parents
- E.g.: s1 x s2 e s5 x s6

  before crossover

  $s1' = 1001011101$ $s5' = 1101111101$

  $s2' = 0110100101$ $s6' = 0000110010$

  after crossover

  $s1'' = 1000100101$ $s5'' = 1101110010$

  $s2'' = 0111011101$ $s6'' = 0000111101$

# Example 1: mutation

- Switch a small number of bits
- Probability to perform mutation $p_{mut}$

| before mutation | after mutation |
|---|---|
| $s1''$ = 1000100101 | $s1'''$ = 1100100101 |
| $s2''$ = 0111011101 | $s2'''$ = 0111111001 |
| s5'' = 1101110010 | s5''' = 1101110010 |
| s6'' = 0000111101 | s6''' = 0000101101 |

# Matlab coding

- Call genetic algorithm

```
x = ga(fitnessfcn,nvars)
[x,fval,exitflag,output,population,scores] = ga(fitnessfcn,nvars,...)
x = ga(fitnessfcn,nvars,A,b,Aeq,beq,LB,UB,nonlcon,options)
```

- Specify binary problem

```
opts.PopulationType='bitstring';
```

- Indicate Selection function

```
opts.SelectionFcn=@selectionroulette;
```

- Indicate Cross function parameters

```
opts.CrossoverFcn=@crossoversinglepoint;
opts.CrossoverFraction=0.8;
```

- Indicate Mutation function

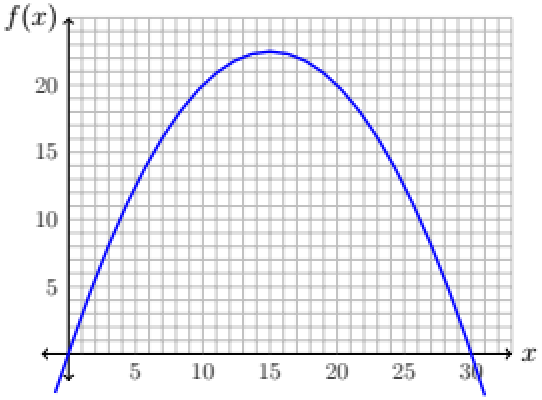
```
opts.MutationFcn= {@mutationuniform, 0.01};
```

# Stopping Conditions

- **Generations** — The algorithm stops when the number of generations reaches the value of Generations.
- Time limit — The algorithm stops after running for an amount of time in seconds equal to Time limit.
- Fitness limit — The algorithm stops when the value of the fitness function for the best point in the current population is less than or equal to Fitness limit.
- Stall generations — The algorithm stops when the weighted average change in the fitness function value over Stall generations is less than Function tolerance.
- Stall time limit — The algorithm stops if there is no improvement in the objective function during an interval of time in seconds equal to Stall time limit.
- Function Tolerance — The algorithm runs until the weighted average change in the fitness function value over Stall generations is less than Function tolerance.
- Nonlinear constraint tolerance — The Nonlinear constraint tolerance is not used as stopping criterion. It is used to determine the feasibility with respect to nonlinear constraints.

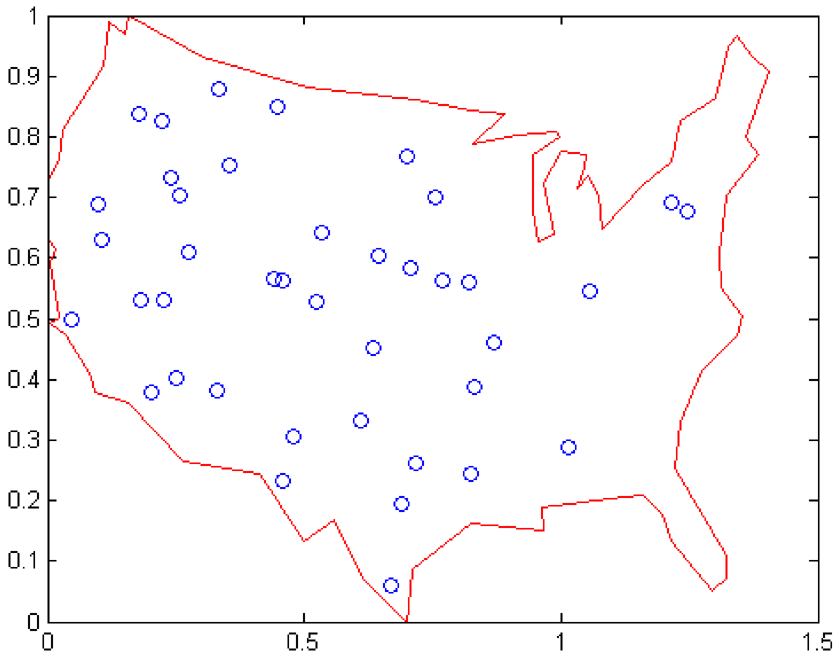
# Exercises

1. Maximize $y=-x^2/10 + 3x$ over the interval {0, 1, …, 31} (reuse code from example 1)
   – Use a binary coding (5 bits) e.g. 01101 -> 13
   – Define a fitness function (clue: bin2dec(int2str(x)) function)
   – Which value did you obtain?



# Real-world binary problems

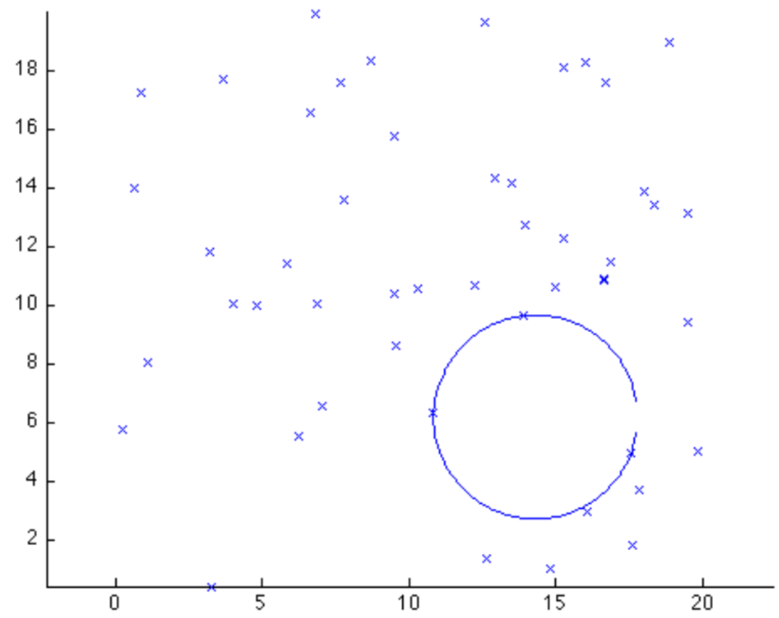
- Knapsack problem, financial applications
- Warehouse Location
- Scheduling
- Routing
- Register allocation
- …



# Example 2: optimization of a continuous function

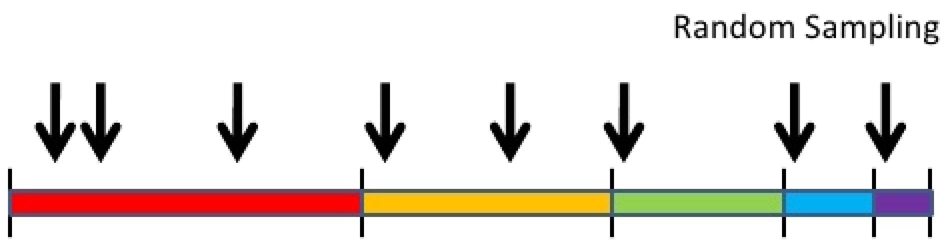
- Objective: searching the biggest circle that can be drawn without enclosing a set of points
- Gene encoding: string of ***2*** real values (coordinates)
- Fitness function: minimum distance to a star or the bounds
- Constraints

  0 <= x <= 20 and 0 <= y <= 20

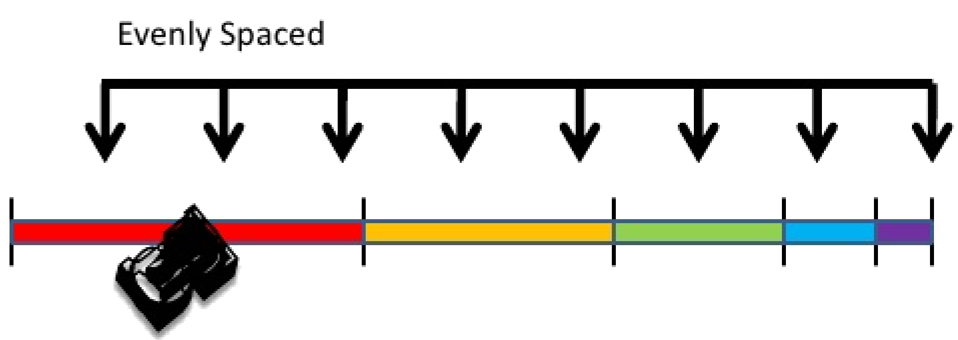


# Example 2: selection

- Roulette Wheel




- Stochastic uniform selection, only one roulette spin, then equally spaced selections, reduces selection pressure
  - Matlab code: opts.SelectionFcn=@selectionstochunif;



# Example 2: crossover

- Scattered crossover: creates a random binary vector and selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent

parent1 = [a b c d]      parent2 = [1 2 3 4]

child = [a 2 3 d]

# Example 2: mutation

- Uniform:
  1. Select a fraction of the elements of the gene to mutate using $p_{mut}$
  2. Replace values by a random number in the range of the entry

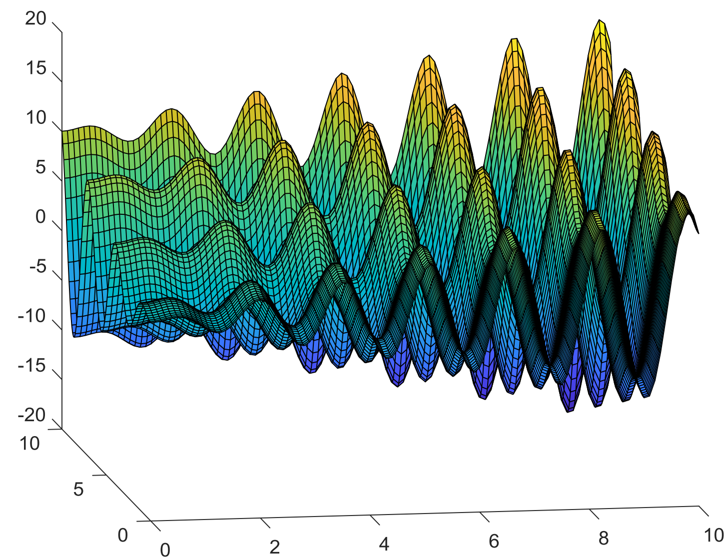
# Exercises

2. Lab4.mat contains 3 instances (star1, star2, star3) of the circle problem (to change instance, substitute param.star=star1, by the appropriate instance)
   - Try to find the global optimum of each instance by adjusting the parameters of the GA (may be different for each instance)
   - Global optimums:
     - Star1: x= 15.85 y=11.43 f=-3.7123
     - Star2: x=16.9 y=15,85 f=-3.0844
     - Star3: x=6.15 y=4.65 f=-2.86
   - You can adjust:
     - PopulationSize
     - Generations
     - CrossoverFraction
     - mutationFraction
     - EliteCount
     - Try also changing the selection to @selectionstochunif

# Exercises

3. Searching the lowest elevation on a topographical map (reuse code from example 2)
   - Create a fitness function using f(x,y) = x sin(4x) + 1.1 y sin(2y) (modify ObjFunGA_example2)
   - 0 <= x <= 10 and 0 <= y <= 10 Gene encoding: string of ***2*** real values
   - Adjust the parameters to obtain the global maximum (x=9, y=8.7, f=-18.426)

   (avoid the use of gaplotCircleVisualizer)

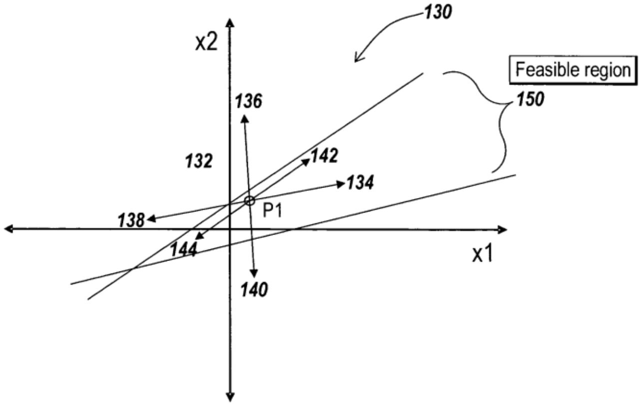


# Constrained optimization problem

- Constrains limit the feasible set of choices in an optimization problem
- A constrained optimication problem reflects a tension between what is desired and what is obtainable
- Types
  - Equality constrains: constrains that hold exactly, e.g. $h_i(x_1,x_2,\ldots,x_n)=0$
  - Inequality constrains: allow a function of one or more of the variables to be less than or greater than some level, e.g. $g_j(x_1,x_2,\ldots,x_n)\leq 0$

# Adaptive feasible mutation

- Randomly generates directions that are adaptive with respect to the last successful or unsuccessful generation
- The mutation chooses a direction and step length that satisfies bounds and linear constraints
- Matlab code:
  opts.MutationFcn= @mutationadaptfeasible;



Adaptive mutation (directions)

# Example 3

- Minimize the constrained optimization problem:

$$\min f(x) = 100 * (x1^2 - x2)^2 + (1 - x1)^2;$$

- such that the following two nonlinear constraints and bounds are satisfied

$$x1*x2 + x1 - x2 + 1.5 \leq 0,$$

$$10 - x1*x2 \leq 0,$$

$$0 \leq x1 \leq 1,$$

$$0 \leq x2 \leq 13$$

# Exercises

4. Minimize the function:

   $f(x,y) = (x - 0.8)^2 + (y - 0.2)^2$

   – Subject to the constraints:

     $g1(x,y)=((x - 0.2)^2 + (y - 0.5)^2) - 0.3 \leq 0,$

     $g2(x,y)=-((x + 0.5)^2 + (y - 0.5)^2) * 2.0 + 1.5 \leq 0,$

     $0 \leq x \leq 1$

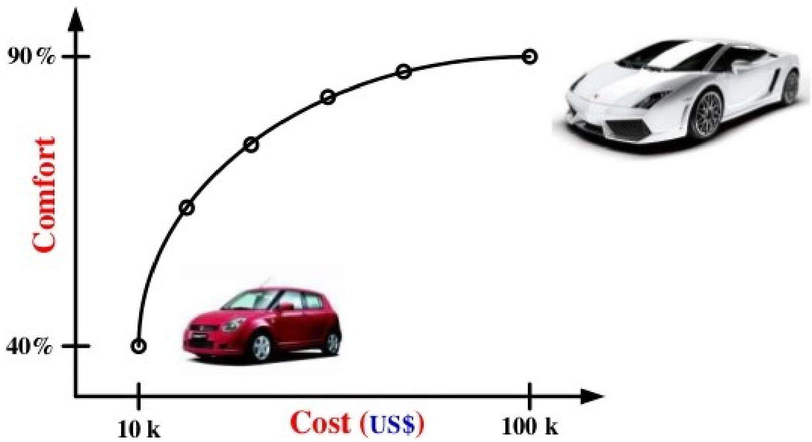
     $0 \leq y \leq 2$

   – Define the fitness function and constraints function according to the formulas

   – Try different parameters to get the optimum (f=0.0157, x=0.69, y=0.26)

# Multi-objective optimization

- Single Objective : Only one objective function
- Multi-Objective : Two or more and often conflicting objective functions
- e.g. Buying a car : minimize cost and maximize comfort



# Pareto front

- Dominated solutions: Set of design points performing worse than some other better points
- Domination criterion:

  A feasible solution $x_1$ dominates an other feasible solution $x_2$ (denoted as $x_1 < x_2$), if both of the following conditions are true:

  1) The solution $x_1$ is no worse than $x_2$ in all objectives, i.e. $f_i(x_1) \leq f_i(x_2)$
  2) The solution $x_1$ is strictly better than $x_2$ in at least one objective, i.e. $f_i(x_1) < fi(x_2)$

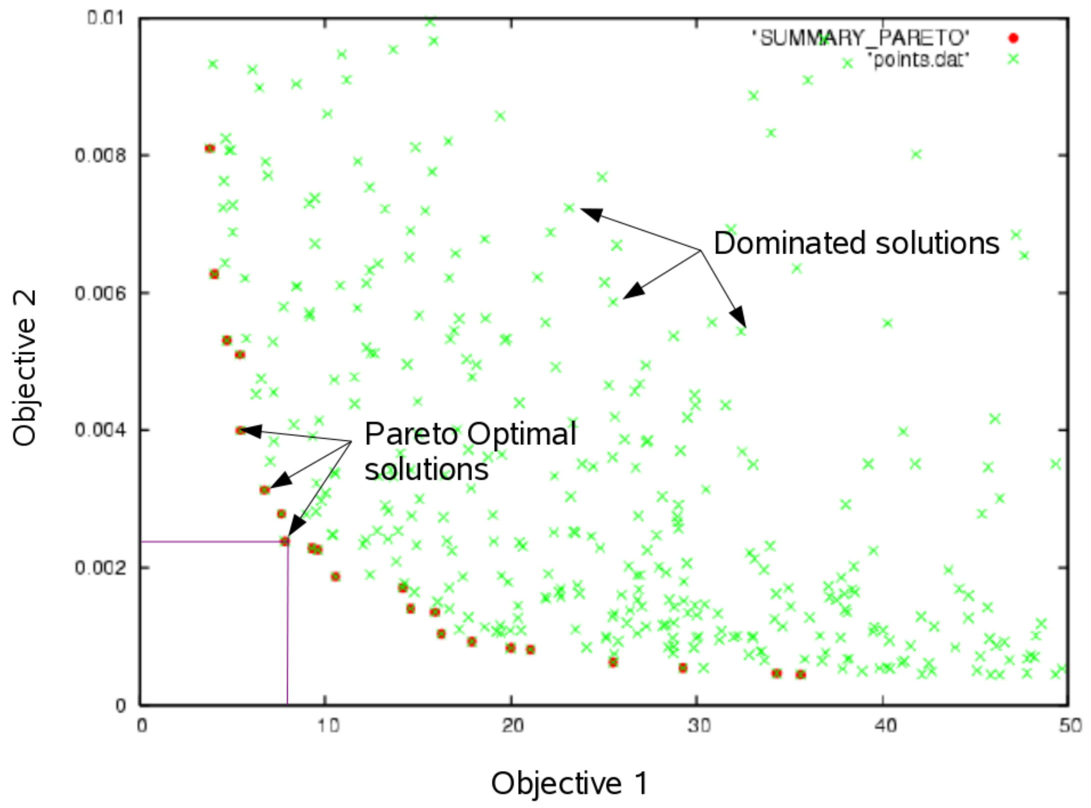
- Non-dominated solutions:

  If two solutions are compared, then the solutions are said to be non-dominated with respect to each other IF neither solution dominates the other

- Pareto optimal front :

  The function space representation of all the non-dominated solutions

# Pareto front



# Matlab coding

- Call genetic algorithm for multiobjective optimization

```
x = gamultiobj(fitnessfcn,nvars)
[x,fval,exitflag,output,population,scores] = gamultiobj(___)
x = gamultiobj(fitnessfcn,nvars,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

# Example 4

- min F(x) = [objective1(x); objective2(x)]

  where,

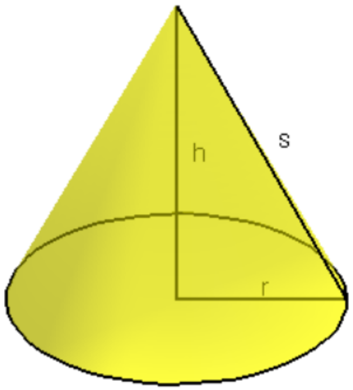
  objective1(x) = (x+2)^2 - 10, and

  objective2(x) = (x-2)^2 + 20

  with $-10 \leq x \leq 10$

# Exercises

5. Minimize the lateral surface area and total surface area of a right circular cone
   - Min f(r,h)=[S,T]
     with $0 \le r \le 10$
     $0 \le h \le 20$
     (use the formulas in the figure to define fitness function for S and T)
   - Constraint:
     $200 - V \le 0$
     (constraints in gamultiobj and ga are managed in an analogous way, define a constraint function)
   - Only plot 'objectives space'




$$s = \sqrt{r^2 + h^2}$$

$$V = \frac{\pi}{3} r^2 h$$

$$B = \pi r^2$$

$$S = \pi r s$$

$$T = B + S = \pi r (r + s)$$