Optimisation in COPASI

Pedro Mendes

School of Computer Science Univ. Manchester

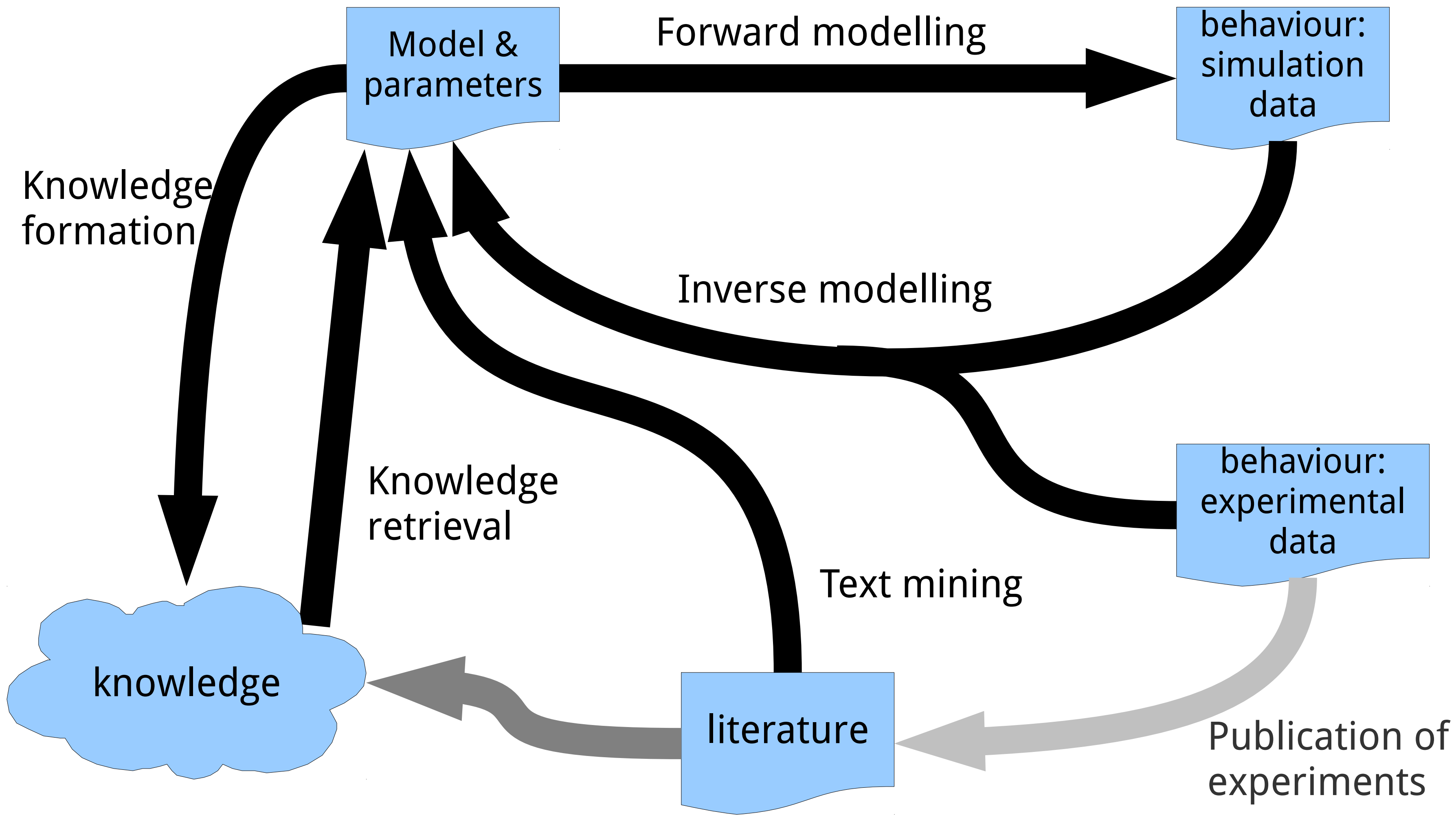
Manchester Centre for Integrative Systems Biology
Virginia Bioinformatics Institute, Virginia Tech

http://www.comp-sys-bio.org

pedro.mendes@manchester.ac.uk

@gepasi on twitter

Modelling cycle

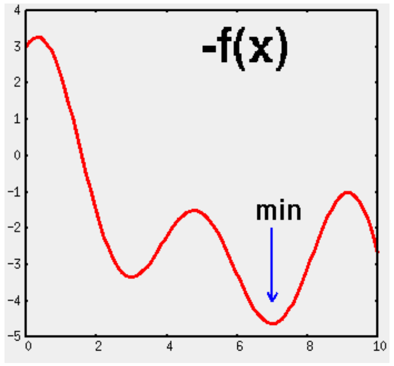


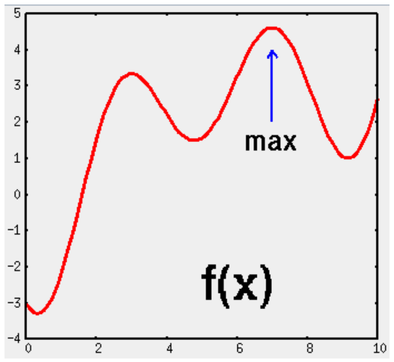
Optimization

given a real-valued scalar function $f(\mathbf{x},\mathbf{k})$
of n parameters $\mathbf{k}=(k_1, \ldots, k_n)$

- find a minimum of $f(\mathbf{x},\mathbf{k})$ such that
- $g_i(\mathbf{x})\geq 0$ with $i=1,\ldots, m$ (inequality constraints)
- $h_j(\mathbf{x})=0$ with $j=1,\ldots, m'$ (equality constraints)

- Note that max f = min $-f$



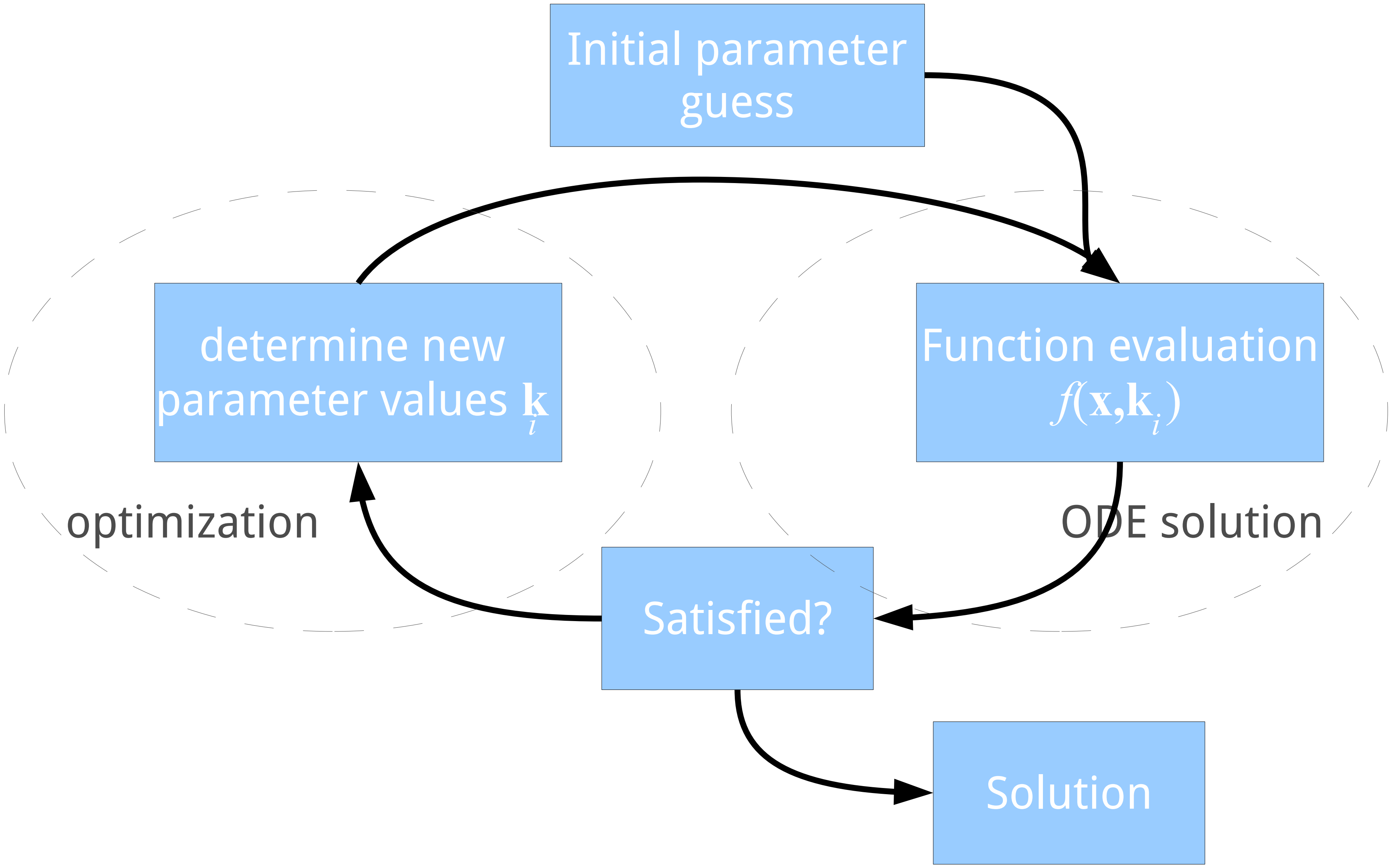


- Optimization methods attempt to maximize or minimize an objective function
- Optimization is able to find the parameter values that result in some property
- Properties of interest must be expressed as minima or maxima

Applications of optimisation in biochemical network modeling

- In addition to their role in **parameter estimation**, optimisation methods are also important in other applications of modeling
- **Exploratory modeling**
  - Explore the properties of a model in a wide range of conditions
  - Similar objective to parameter scans and bifurcation analysis
- **Metabolic engineering**
  - Rational improvement of biotechnological processes through modeling
- **Evolutionary studies**
  - Study of basic principles of biochemical evolution through modeling

Numerical optimization cycle



Initial parameter guess
determine new parameter values $\mathbf{k}_i$
Function evaluation $f(\mathbf{x},\mathbf{k}_i)$
optimization
ODE solution
Satisfied?
Solution

Optimization methods

Based on derivatives:

- Levenberg-Marquardt
- Steepest descent
- Truncated Newton

Direct search:

- Hooke & Jeeves
- Nelder & Mead (simplex)
- Praxis

Evolutionary:

- Genetic algorithm
- GA w/ stochastic ranking
- Evolutionary programming
- Evolution strategy w/ stochastic ranking

Other stochastic:

- Simulated annealing
- Particle swarm
- Random search

Numerical optimization methods

- **Gradient search**
    - Steepest descent
    - Newton and quasi-Newton
    - Levenberg-Marquardt
    - Conjugate gradient

These methods rely on derivatives of the objective function. They are, therefore applicable only to differentiable functions. Some methods require the functions to have second derivatives. Strictly speaking, these methods require analytical expressions of the derivatives, but can be used with numerical estimates.

- **Direct search**
    - Hooke and Jeeves
    - Nelder and Mead (simplex)
    - Powell
    - Brent's *praxis*

These methods do not rely on derivatives, not even on their (numeric) estimates. They rely on memorizing previous estimates of parameters, and on heuristics. They only require the objective function to be continuous, not differentiable.

Gradient search

Using derivatives to find good search directions

Steepest descent

- Considering that the function is differentiable:
  - It decreases at maximum speed in the direction of -∇:

$$b = a - \alpha \nabla f(a)$$

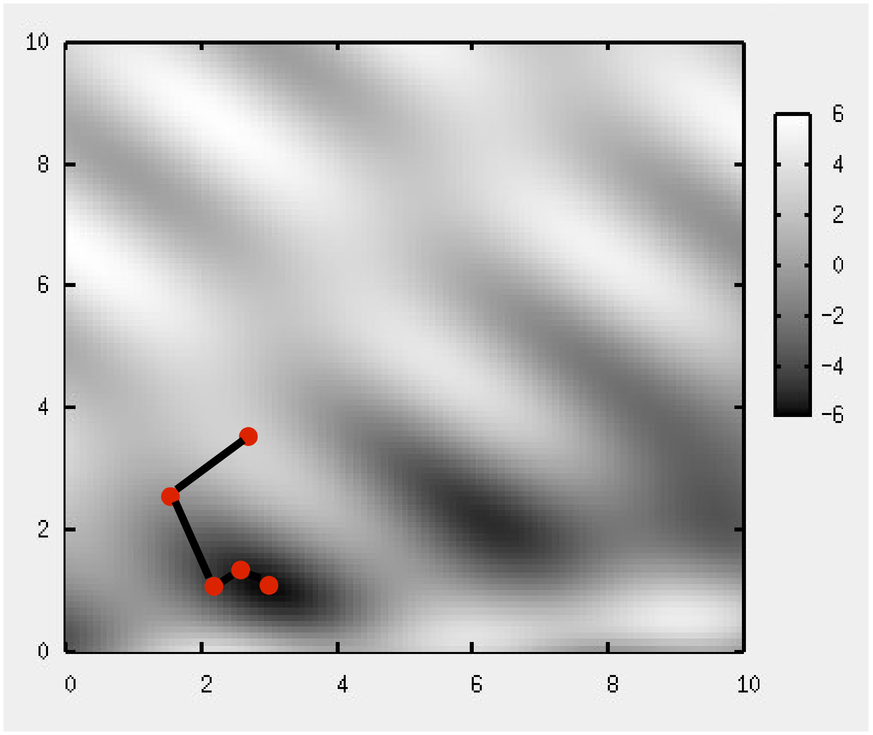
$$f(a) \geq f(b)$$

  - For a small enough α
  - Then iterate as:

$$x_{i+1} = x_i - \alpha_i \nabla F(x_i)$$

$$f(x_0) \geq f(x_1) \geq f(x_2) \ldots \geq f(x_n)$$

- Rather than using a small constant α, carry out a line search in the direction of -∇ at each step



Newton method for minimisation

- If F(x) is twice-differentiable, then:

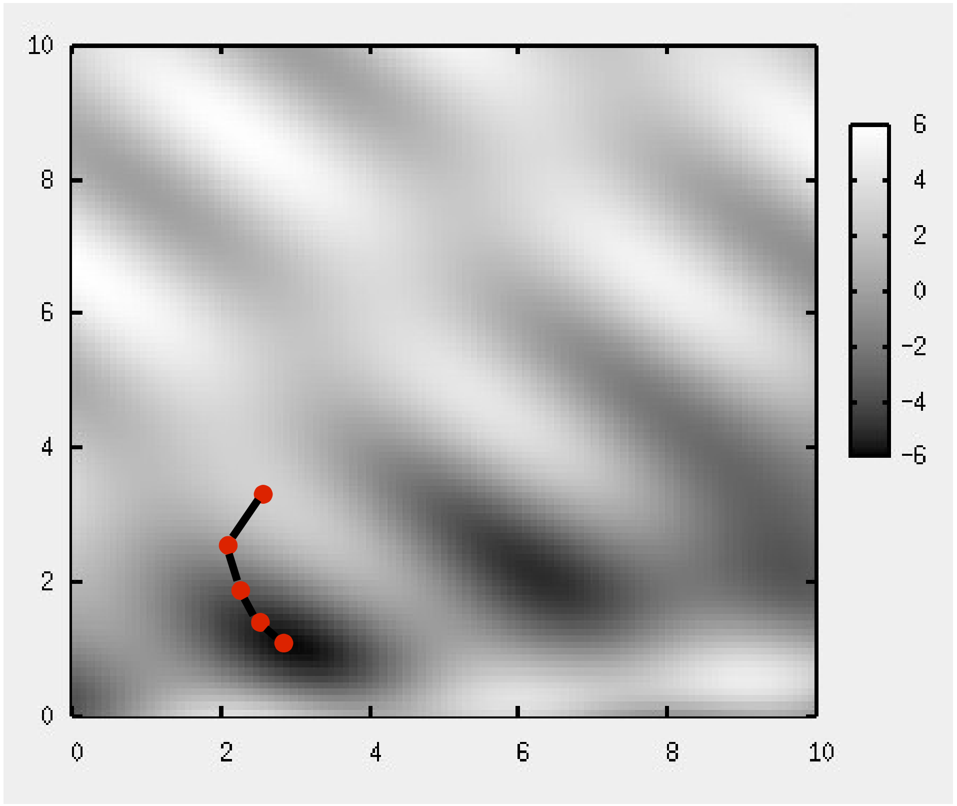
$$x_{i+1} = x_i - \alpha_i \frac{\nabla f(x_i)}{H f(x_i)}$$

$$H f(a) = \left[\frac{\partial^2 f}{\partial a_i a_j} \right]$$

$$f(x_0) \geq f(x_1) \geq f(x_2) \ldots \geq f(x_n)$$

- Method only converges if initial point is close to solution
- Hessian matrix has large memory requirements

- Carry out a line search in the direction of -∇/*H* at each step



Practical variants of Newton method

- Since Newton method is not garanteed to converge, but steepest descent is:
  - ***Levenberg-Marquardt*** uses an adaptive linear combinantion of the steepest and Newton directions
  - ***L-M*** is very robust and in practice the method of choice for least-squares

- ***Quasi-Newton*** methods avoid recalculating the Hessian, using instead some approximation, e.g. the ***BFGS*** algorithm

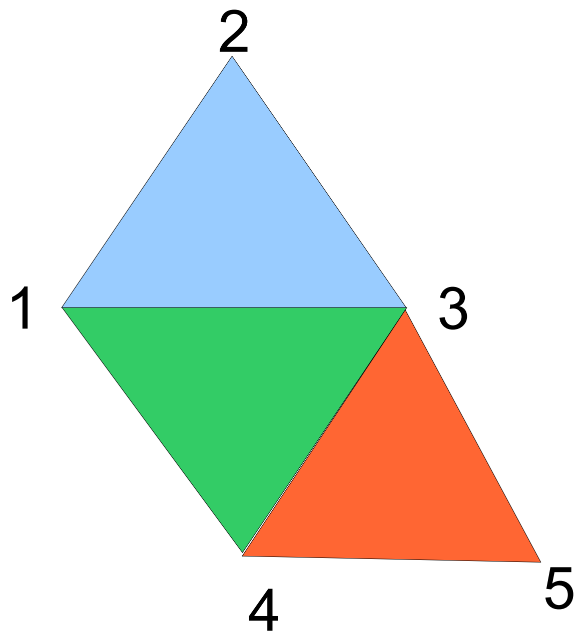
Direct search

Using memory to find good search directions

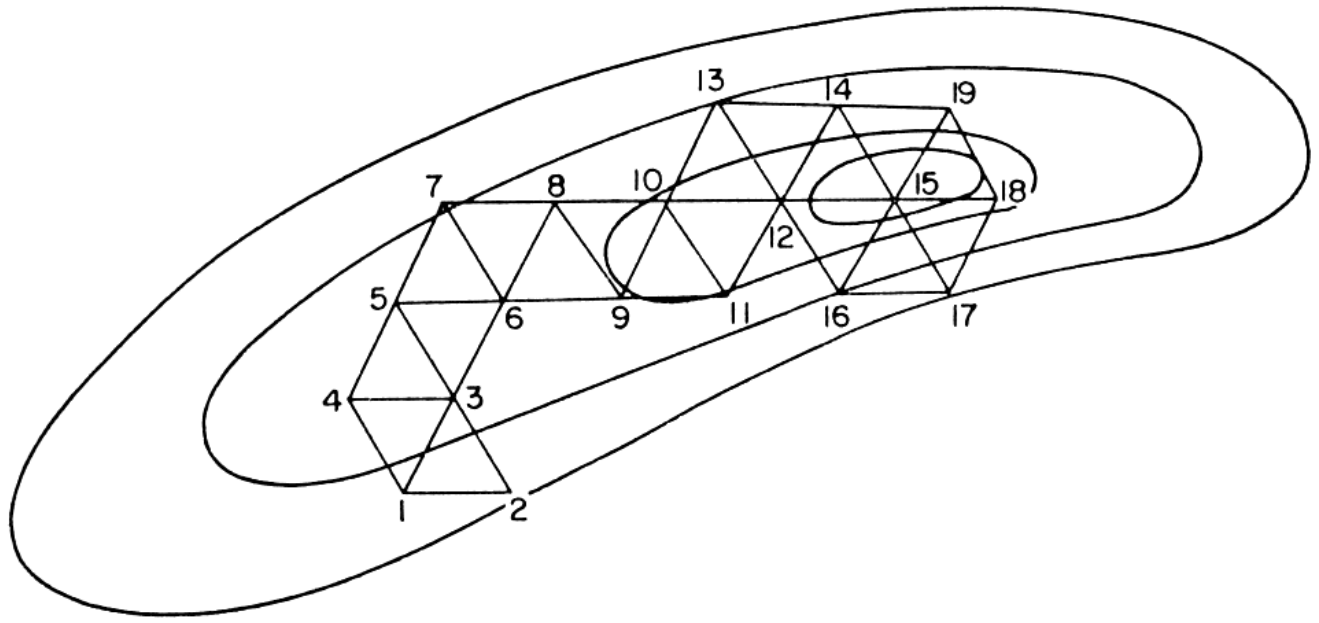
Simplex methods

- A simplex is an object consisting of $n+1$ vertices (in a space of n dimensions)
- A new simplex can always be formed on any face of a given simplex by the addition of only a single new point

In two dimensions:



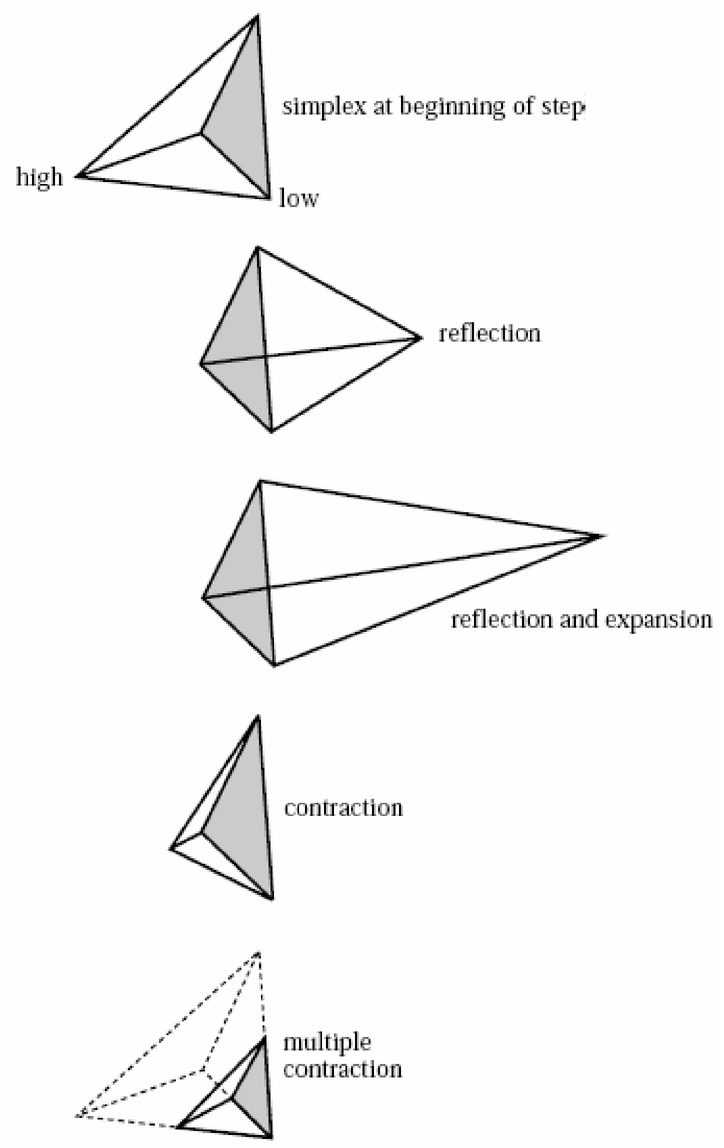
The simplex method of Spendley, Hext and Himsworth (1962)



The simplex method for a function of two variables.

The downhill simplex method of Nelder and Mead (1965)

- Nelder and Mead proposed an important variant, where the simplices are no longer necessarily regular
- They created rules that expand and contract the simplex, in addition to reflections
- This allows the method to be adaptive, quicker, and results in better approximations to the solution

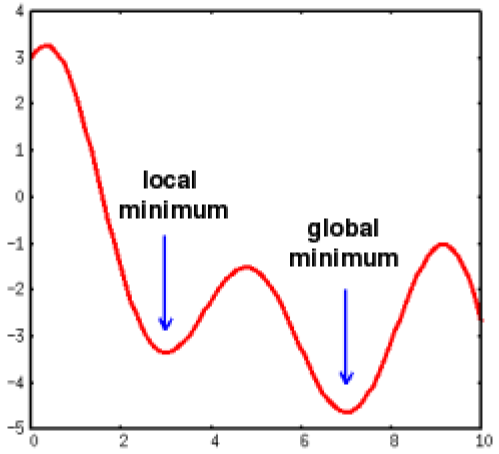


Stochastic methods

Using probabilistic methods to find good search directions

Global optimization

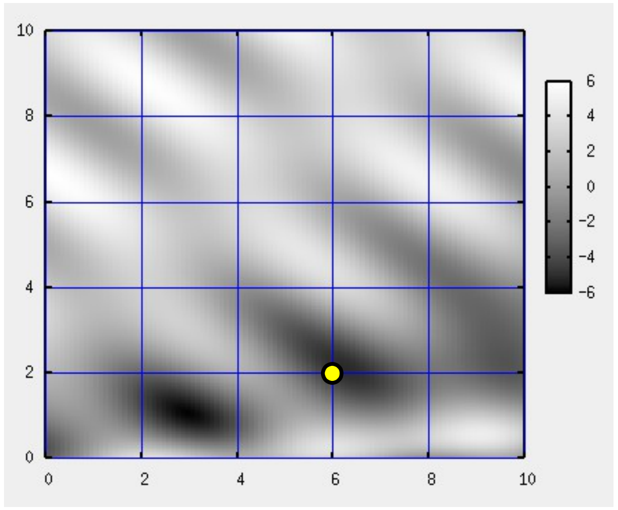
- Global optimization is the task of finding the absolutely best set of admissible conditions to achieve an objective under given constraints, assuming that both are formulated in mathematical terms (Neumaier, 2004)



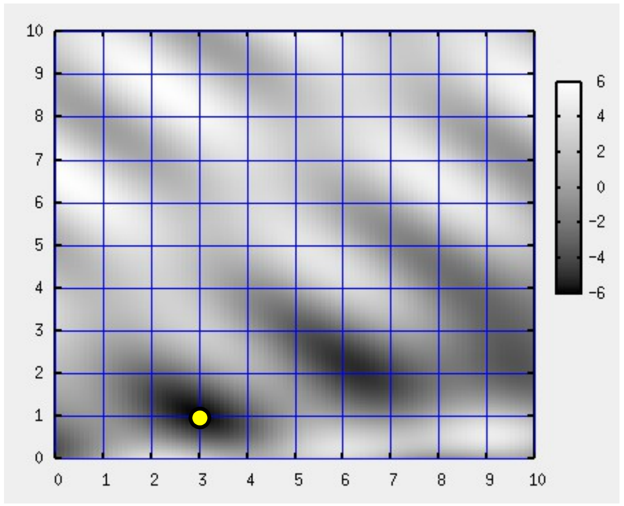
- Examples:
  - Protein folding
  - Traveling salesman
  - Scheduling tasks and slots
  - Chemical equilibrium
  - Least squares
  - Packing (Kepler's problem)
- Classification:
  - Incomplete
  - Asymptotycally complete
  - Complete
  - rigorous

Grid search

- In **grid search** all points on a fine grid (in parameter space) are tested, and the best retained
- Algorithm scales as O((n^p))
  - n: number of grid points per dimension
  - p: number of parameters
- Solution quality depends on grid density
- Impractical even for small dimensions



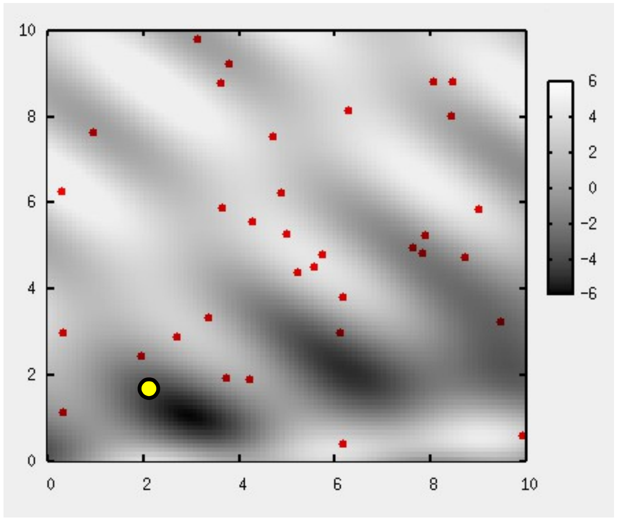
p=2, n=6, total = 6^2 = 36



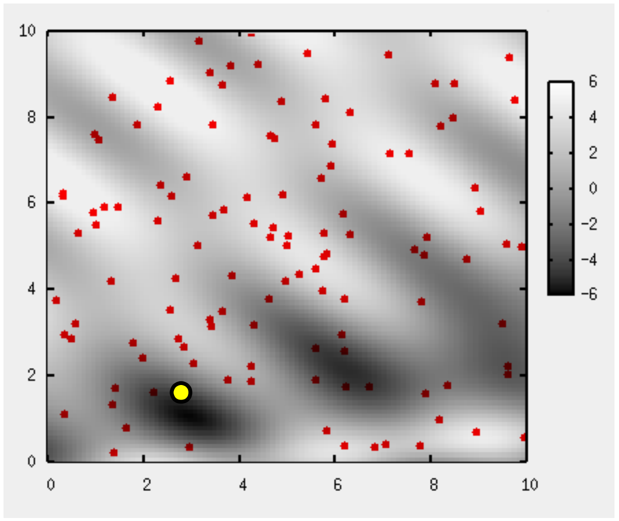
p=2, n=11, total = 11^2 = 121

Random search

- In **random search** a fixed number of random points (in parameter space) are tested and the best retained
- Algorithm scales with O(n)
    - n: total number of points
- Solution quality depends on grid density
- Usually performs very badly
- … but sometimes can outperform all other methods



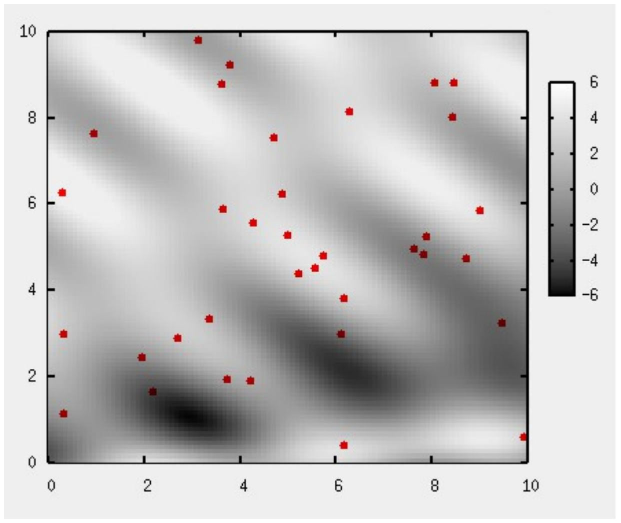
total = 36



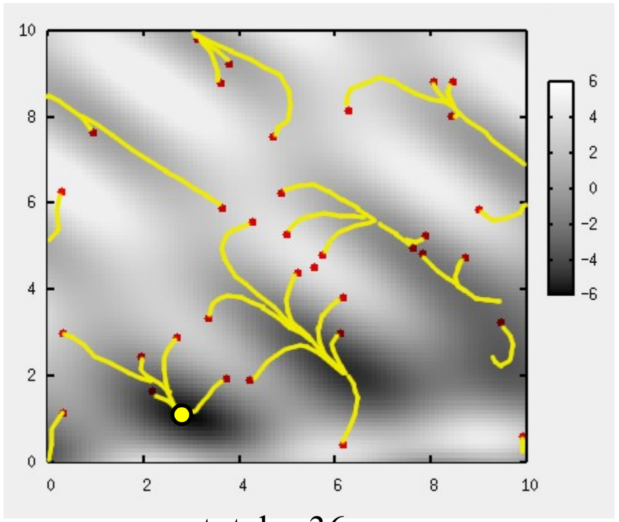
total = 121

Multistart

- An improvement over random search
- Carry out a local minimization for each random parameter guess
- Improves solution quality
- Performance is still bad since local min. May be too costly (time)
- A further improvement is to cluster points so as not to visit clusters 2nd time



total = 36



total = 36

Minimization by analogy with Nature (statistical mechanics)

- “Perfect” crystals are formed by melting, and then cooling down slowly, allowing the material to reach equilibrium at each temperature
- If cooling is fast, the crystal will have imperfections; if it is too fast it will become amorphous (glass)
- **Local optimization algorithms are similar to cooling materials too fast**

- Metropolis *et al.* (1953) proposed an algorithm to simulate an ensamble of particles in equilibrium at a certain temperature
- The Boltzmann probability density function:

$$p.d.f. = e^{\frac{-E}{k_B T}}$$

probability that a certain particle configuration with energy E has at a certain temperature T

Metropolis algorithm

1. Start with an arbitrary position for one atom, $\boldsymbol{k}^{(0)}$
2. Create a small random displacement to obtain $\boldsymbol{k}^{(1)}$ and calculate the difference in energy $\Delta E = E^{(1)} - E^{(0)}$
3. If $\Delta E < 0$ accept the new position $\boldsymbol{k}^{(1)}$, otherwise accept it only with probability

$$P(\Delta E) = e^{\frac{-\Delta E}{k_B T}}$$

4. Iterate algorithm a large number of times, simulating the thermal motion of particles in a heat bath of temperature T
5. This choice of probability $P(\Delta E)$ evolves the system to a Boltzmann distribution
6. Note that the Metropolis algorithm allows the energy to increase (though, with probability of decreasing with T)

Simulated annealing
Kirkpatrick, Gelatt, Vecchi (1983)

- The energy of a particle configuration is similar to the value of an objective function
- The atom coordinates are similar to the parameters of the objective function
- The temperature is a control parameter with the same units as the objective function
- Simulated annealing starts by “melting” the objective function to a high enough temperature
- It uses the Metropolis algorithm to calculate the equilibrium of the objective at a certain temperature
- A cooling schedule must be defined (*i.e.* how the energy will be decreased)

Evolutionary algorithms

Using populations and selection to optimize functions

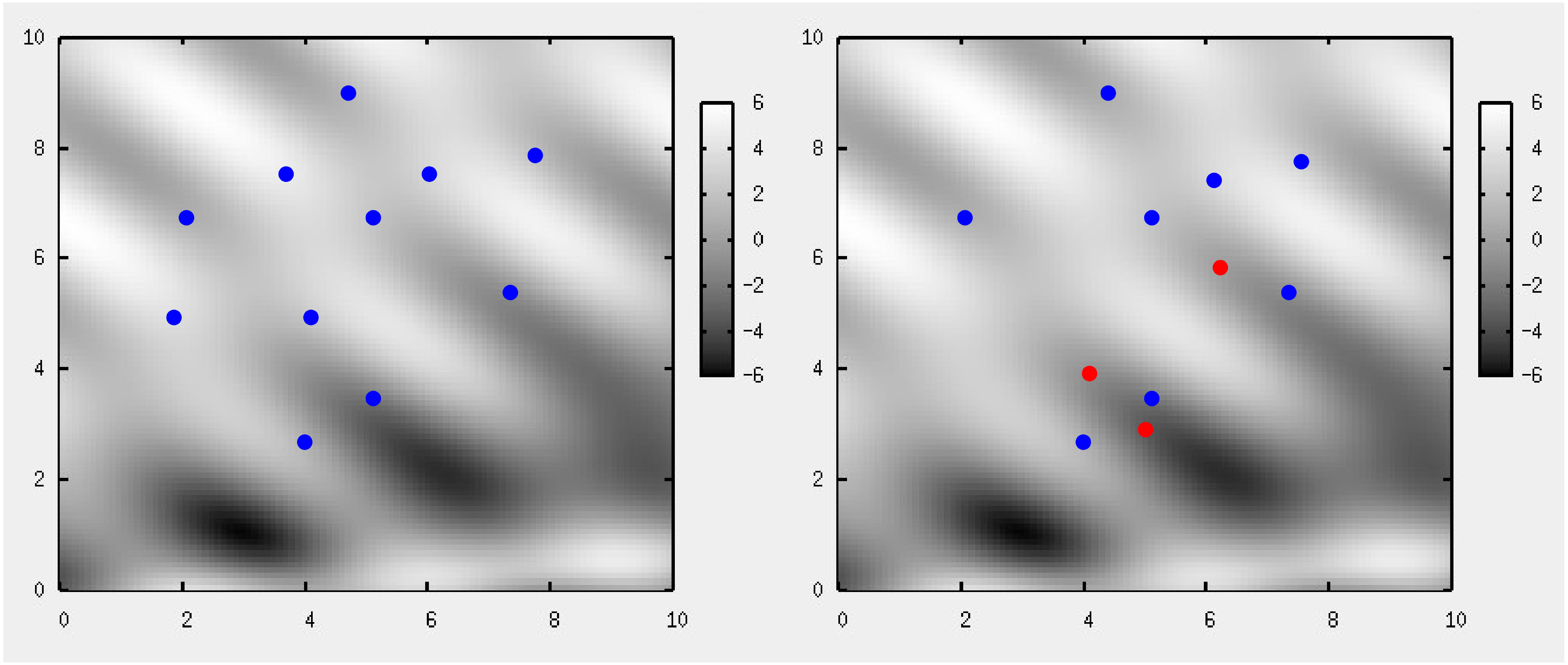
Minimization by analogy with Nature (evolutionary algorithms)

- Populations evolve by the action of *variation* and *selection*
- Evolutionary algorithms are a class of optimization methods that are based on **evolving candidate solutions** as an ensemble, rather than one at a time
- Algorithms differ in method of variation, selection and how numeric values are represented

- Gene ↔ parameter
- Chromosome ↔ all parameters
- Individual ↔ candidate solution
- Generation ↔ iteration
- Fitness ↔ objective function value
-

Evolutionary algorithms evolve populations of solutions

Generation *n* → Generation *n*+1



Evolutionary programming (Fogel et al., 1966)

- **Parameters are encoded as real numbers: genes are numbers**

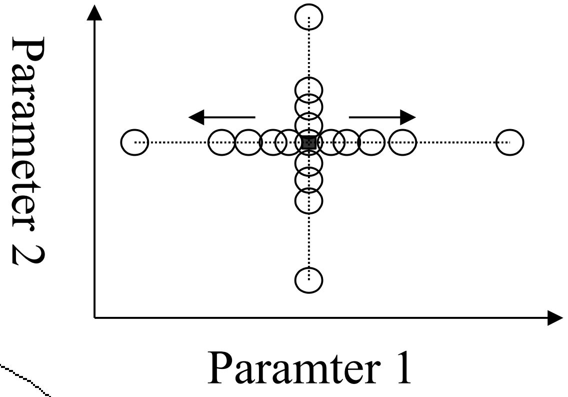
1. Generate a random initial population of *n* individuals
2. Calculate the fitness of each individual in the population
3. Each individual from the current population generates an offspring by copying its own genes
4. *Mutate* each locus in the offspring with a small variance
5. Put the offspring in the new population (now 2*n*)
6. *Select* *n* individuals probabilistically as a function of fitness to be removed (back to *n*)
7. Go to step 2 with the new population, or stop if satisfied

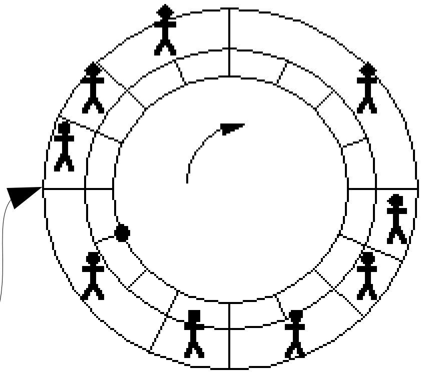
Selection and mutation operators

- **Mutation operator**: add one small normal random number to the original value:

$$Mut(x) = x + N(0, \sigma)$$

- **Selection operator**: find the *n* best individuals in a probabilistic way, *i.e.* may not be exactly the best...
  - Roullete wheel
  - Tournament selection
  - Stochastic ranking





Each individual is compared with a small number of others (random) and it receives a score that is the number of those others that are less fit that itself. The *n* individuals with the best scores are chosen.

A stochastic sort algorithm is used such that the individuals are almost sorted by fitness, but not exactly.

Genetic algorithm
Holland (1975) De Jong (1975)

- **Parameters are encoded in binary: genes are strings of binary digits**

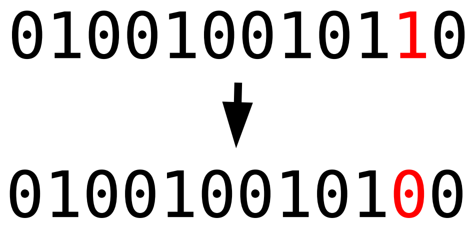
1. Generate a random initial population of *n* individuals
2. Calculate the fitness of each individual in the population
3. Choose two parent individuals from the current population probabilistically as a function of fitness
4. *Cross them over* at a randomly chosen locus to produce two offspring
5. *Mutate* each locus in the offspring with a small probability
6. Put the offspring in the new population
7. Go to step 2 with the new population, or stop if satisfied

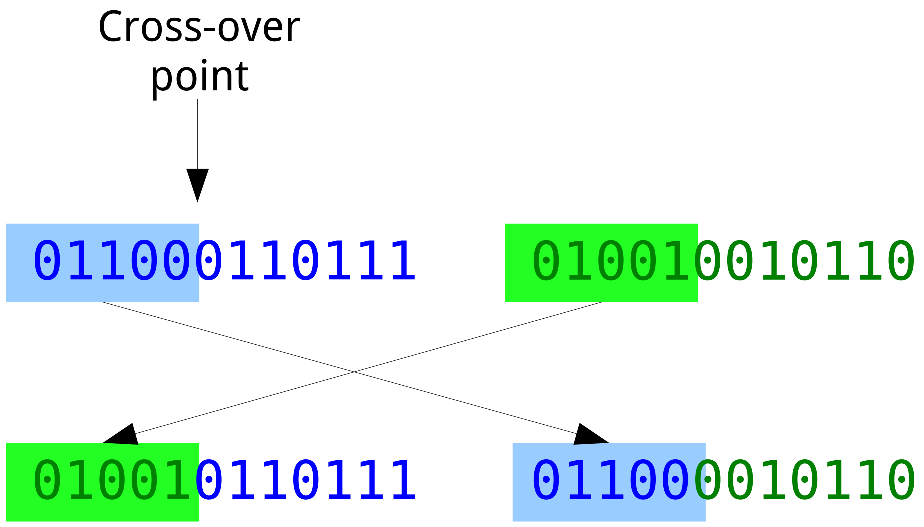
Mutation and cross-over operators

- **Mutation operator**: change one bit:

  $$Mut(x) = x + 1, x \in \{0,1\}$$

- **Cross-over operator**: from two parents, produce two offspring with genetic recombination
  - Cross-over can happen at one or more points





Selection *vs.* variation

- Selection is responsible for keeping improvements in the population
- Mutation and cross-over are responsible for introducing variation in the population, i.e. drift in the parameter values
- Very strong selection results in uniform populations that are have many copies of a good individual
- Very strong variation results in that good solutions do not progress but are constantly replaced by new random ones

Termination criteria

- After a prespecified number of generations
  - Not easy to guess how many generations are required, usually requires some trial and error
- When best solution reaches a prespecified level of fitness
  - Appropriate if the required level of fitness is known a priori, but this is often not possible
- When the variation of individuals from one generation to the next reaches a prespecified level of stability
  - Given that convergence can be punctuated, this is generally not a good criterion

Parameter estimation with COPASI

Pedro Mendes

School of Computer Science Univ. Manchester

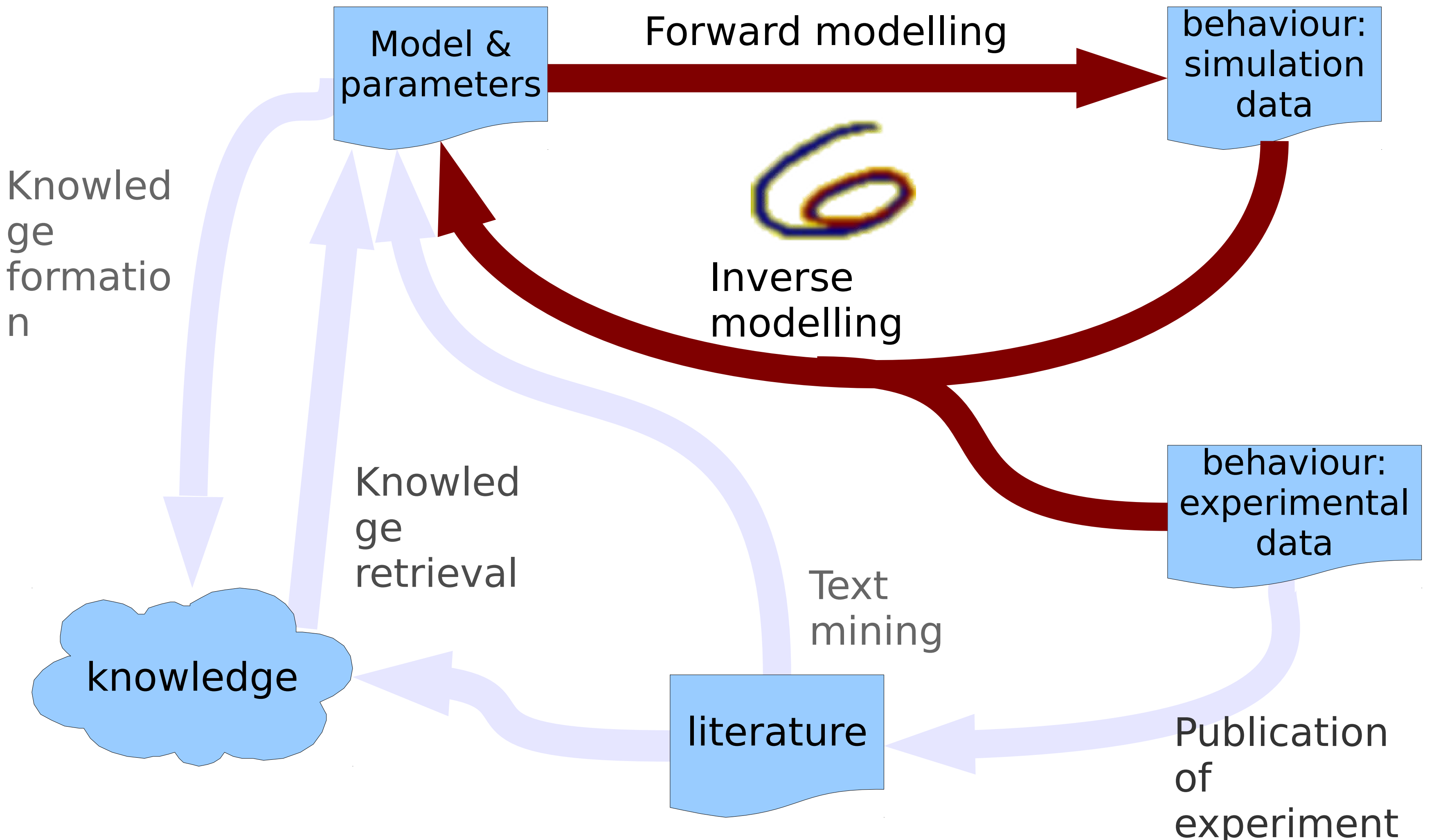
Manchester Centre for Integrative Systems Biology
Virginia Bioinformatics Institute, Virginia Tech

http://www.comp-sys-bio.org

pedro.mendes@manchester.ac.uk

@gepasi on twitter

Modelling cycle



Parameter Estimation

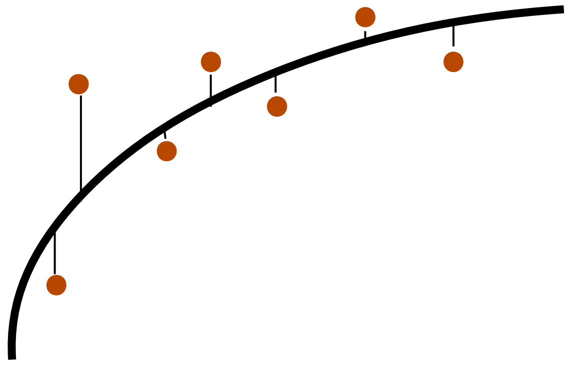
- ***Given a set of data, adjust a model's parameter values such that the distance between the model behaviour and the data is minimal***
- An essential part of parameter estimation consists on the application of numerical optimisation algorithms.
- In particular, many parameter estimation applications rely on either of the following

  minimisation of a least squares function

  minimisation of other distance measure

Least-squares methods

- Given a nonlinear function $f(\boldsymbol{x}, \boldsymbol{k})$
- And a set of observations χ Find the minimum of $\sum(\chi - f(\boldsymbol{x}, \boldsymbol{k}))^2$
- By changing $\boldsymbol{k}$ (parameters)



Scale problem

- Some of the model variables being fit are of very different scales
- Each variable trajectory (or steady state) is then multiplied by a weight
- Weights rescale the importance of each variable in the fit
- In COPASI weights can be changed

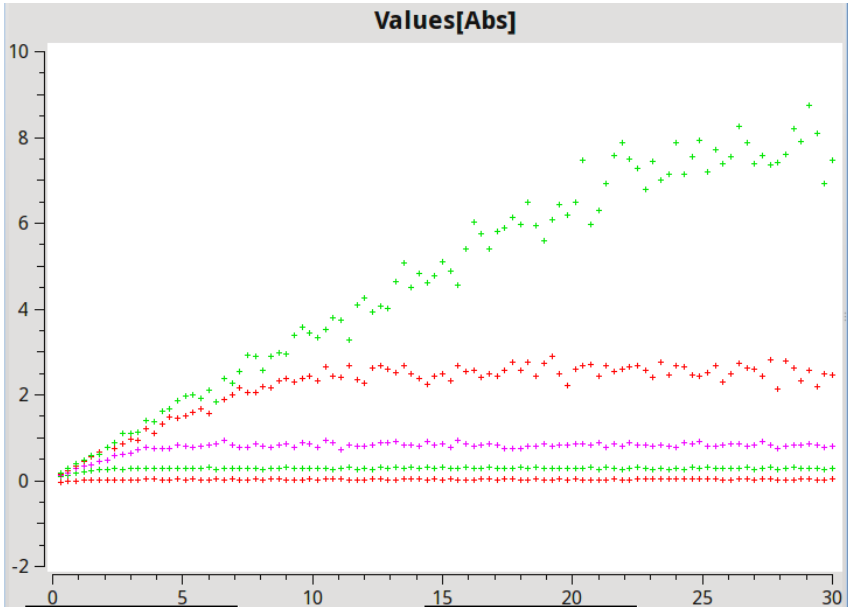
Enzyme kinetics

Michaelis-Menten mechanism

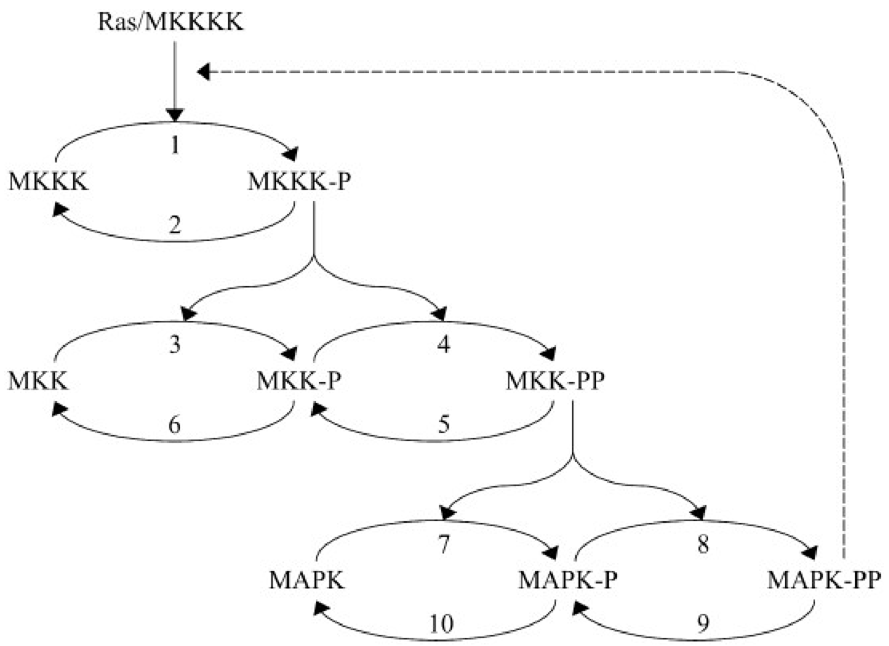
$S + E \rightleftarrows ES \quad k_{1,1}, k_{1,2}$

$ES \rightarrow E + S \quad k_2$

Data from spectrophotometer measurements

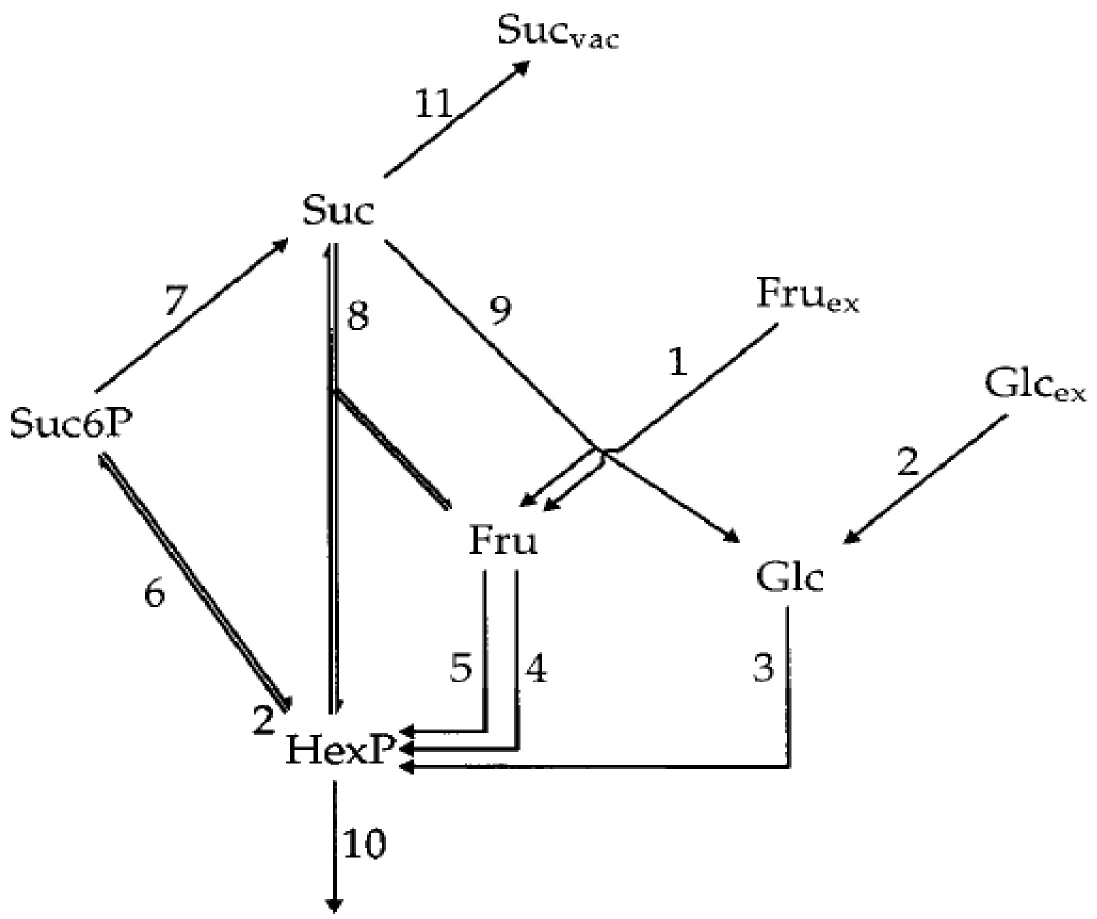


Biomodels 10



Kholodenko BN. (2000) Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. *Eur J Biochem.* 267(6):1583-8

Biomodels 23



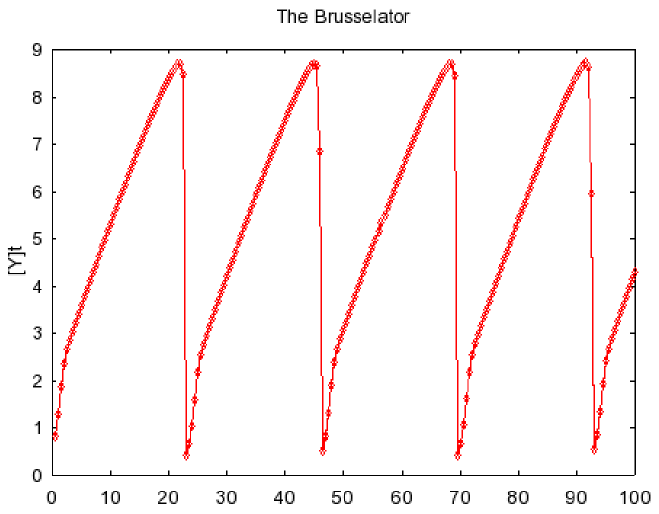
Rohwer JM, Botha FC. (2001) Analysis of sucrose accumulation in the sugar cane culm on the basis of in vitro kinetic data. *Biochem J.* 358(Pt 2):437-45

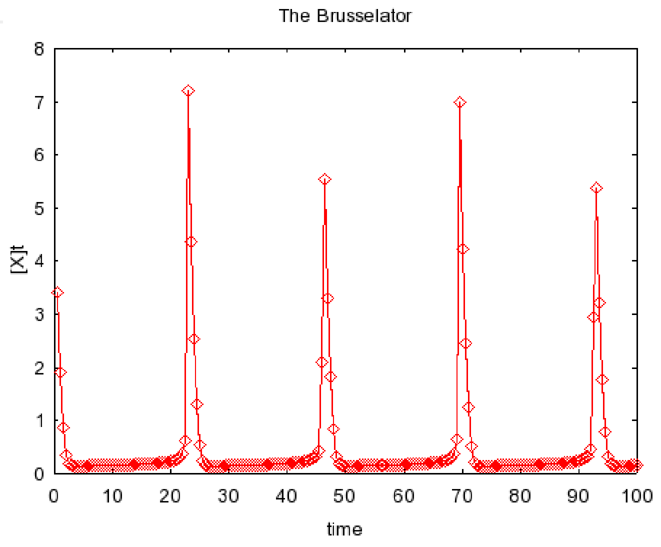
Brusselator

$A \rightarrow X$	$r1(k) = 1$
$2X + Y \rightarrow 3X$	$r2(k) = 1$
$X + B \rightarrow Y + D$	$r3(k) = 1$
$X \rightarrow E$	$r4(k) = 1$

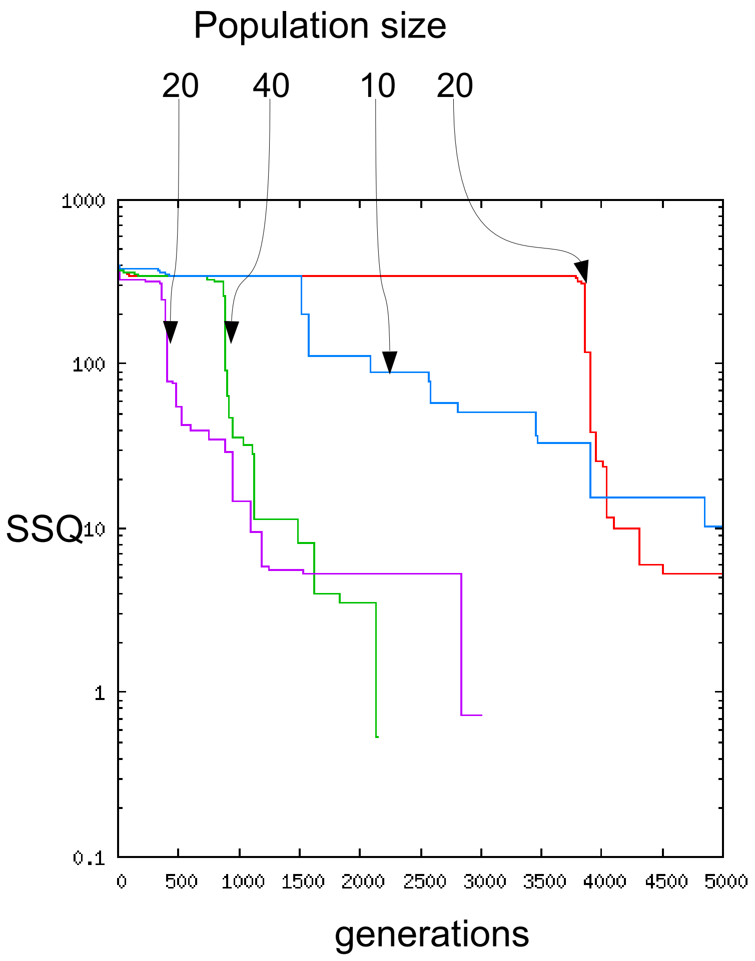
$$\frac{dX}{dt} = A \cdot r1(k) + X^2 \cdot Y \cdot r2(k) - X \cdot B \cdot r3(k) - X \cdot r4(k)$$
$$\frac{dY}{dt} = X \cdot B \cdot r3(k) - X^2 \cdot Y \cdot r2(k)$$

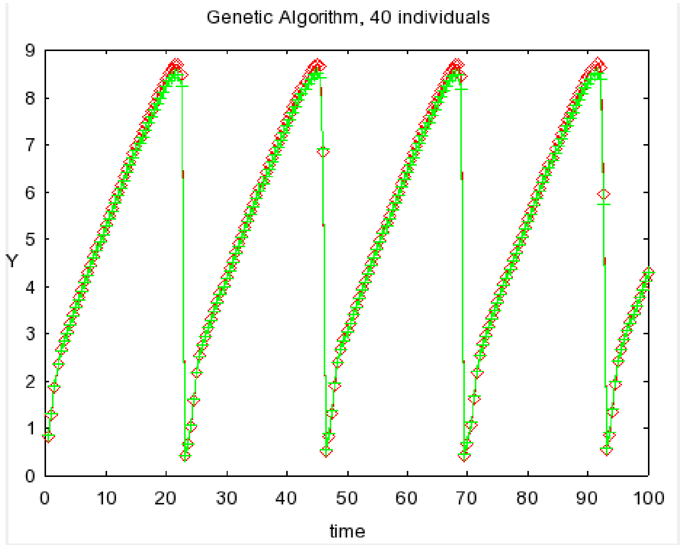
- X, Y are the variables
- A, B, D and E are fixed concentrations
- All reactions follow mass action kinetics





Fitting the Brusselator model to the data





```
PARAMETER FITTING
method: Genetic algorithm
        Generations = 5000
        Population = 40
        iterations  = 2151
        simulations = 87488
        time        = 999.959 s
        speed       = 87.4916 simulation/s
 BEST SOLUTION:
 Sum of squares = 0.539357
 Std. deviation = 0.0524578
 RMS error      = 0.0519306
 Parameter     Value        Std.Deviation
 -----------------------------------------------
 R1(k)          1.028        1.709e-005  (0.00%)
 R2(k)          0.889        2.083e-005  (0.00%)
 R3(k)          0.9169       0.0004926   (0.05%)
 R4(k)          1.028        0.00056     (0.05%)
```