Lecture: Computational Systems Biology
Universität des Saarlandes, SS 2012

10 Optimisation

Dr. Jürgen Pahle

3.7.2012

Recap

- Biochemical systems show **(intrinsic) random fluctuations** in molecular numbers due to **stochastic timings** of **discrete** reactive events in the system ("firings of reactions")
- This can lead to **quantitative and qualitative different behaviour in stochastic models** (continuous-time Markov process described by a chemical master equation) compared to deterministic models → stochastic effects
- There exist several **exact stochastic simulation algorithms** (SSA, e.g. Gillespie's Direct Method) to capture these effects but they can be computationally demanding
- Therefore **approximate stochastic simulation algorithms**, such as τ-Leaping, are being developed (trade-off between accuracy and run time)
- **Hybrid algorithms**, seem particularly promising because they integrate different mathematical frameworks and can deal with the multi-scale nature of biochemical systems

Optimisation

Relation between parameter values and behaviour of models is usually not obvious →

How to change the parameters in a model/system to get a desired behaviour?

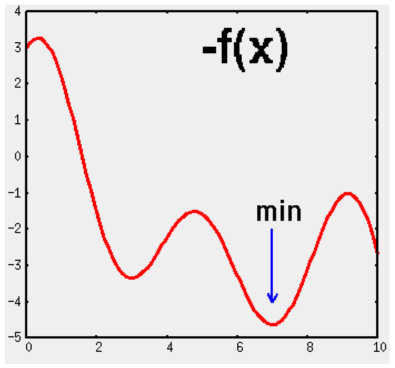
Optimisation

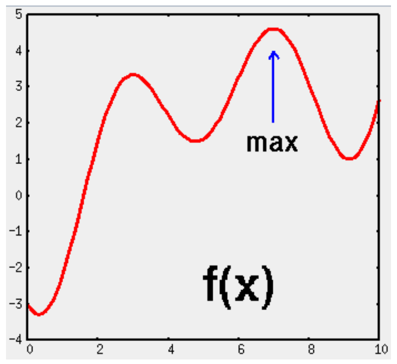
Optimisation methods try to maximize or minimize a so-called **objective function**

- **The properties of interest must be expressed as minima or maxima of this objective function**
- Then optimisation is able to find parameter values that result in that property

Note that *max(f(x)) = min(-f(x))*

Therefore, we can restrict ourselves to only finding minima





Problem formulation

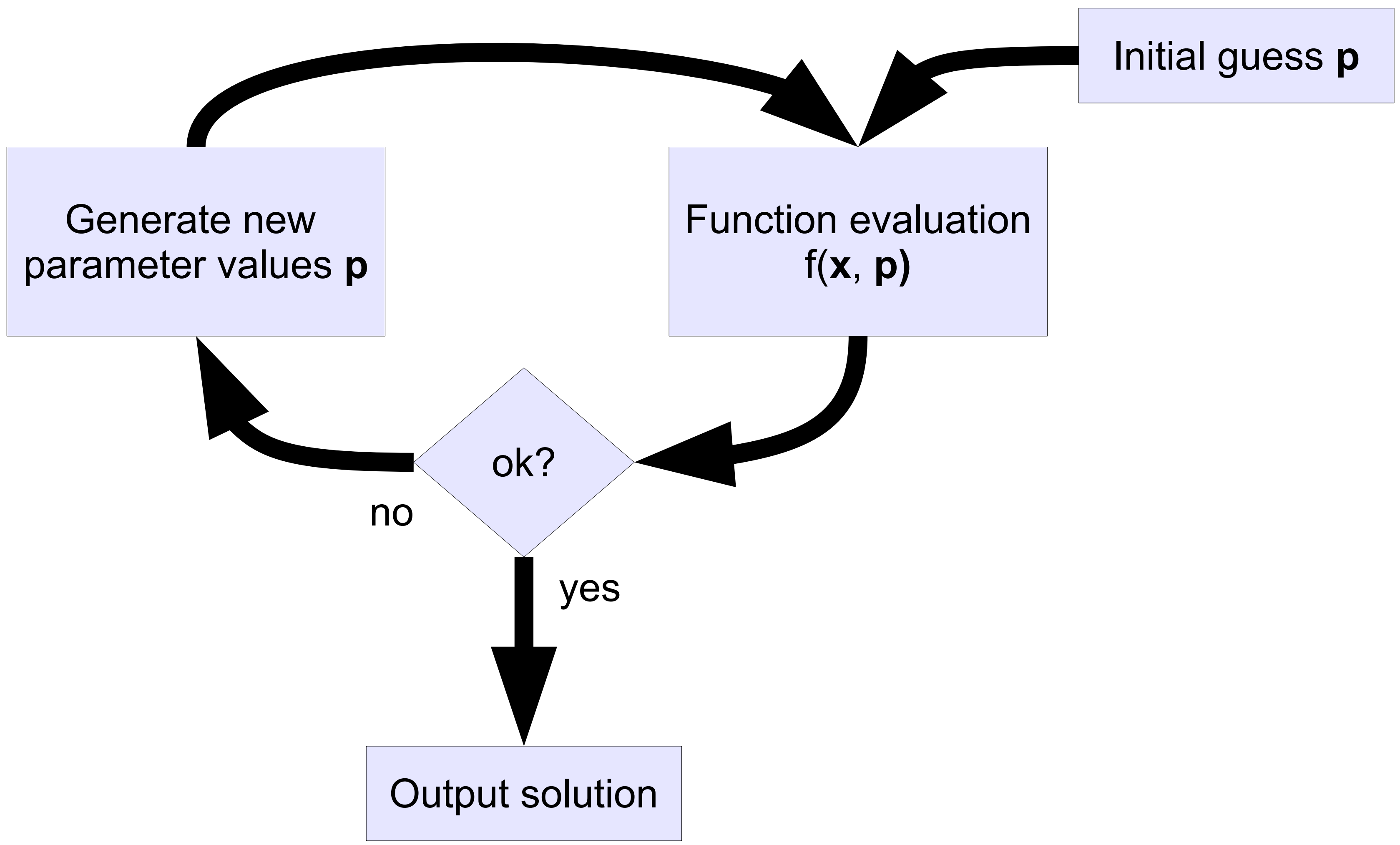
Given a real-valued scalar function ("objective function") $f(\mathbf{x}, \mathbf{p})$ of n parameters $\mathbf{p} = (p_1, ..., p_n)$ find a minimum of $f(\mathbf{x}, \mathbf{p})$ such that:

- $g_i(\mathbf{x}) \geq 0$ with $i = 1, ..., m$ **(inequality constraints)**
- $h_j(\mathbf{x}) = 0$ with $j = 1, ..., m'$ **(equality constraints)**

Applications of optimisation

- <u>**Parameter estimation**</u>
- **Exploratory modeling**
  - Explore the properties of a model in a wide range of conditions
  - Similar objective to parameter scans and bifurcation analysis
- **Metabolic engineering**
  - Optimise biotechnological processes, increase fluxes of reactions that produce desired products
- **Evolutionary studies**
  - Study of basic principles of biochemical evolution through modeling

Optimisation (schematic)


Initial guess **p**
Generate new parameter values **p**
Function evaluation f(**x**, **p**)
ok?
no
yes
Output solution

Local ↔ global optimisation methods

- **Local methods** only use information in the vicinity of a given point to find the next point (in parameter space)
  - Are usually fast
  - Might get stuck in local minima
- **Global methods** try to cover as much of the search space as possible
  - Are usually slower
  - Are less likely to get stuck in local minima
  - In the limit (of very many iterations) one can prove that, e.g., Simulated Annealing will find a globally optimal solution. However, this can take an extremely long time

Different families of optimisation methods

Based on derivatives

- Steepest descent
- Levenberg-Marquardt
- Truncated Newton

Direct search

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

Evolutionary

- Genetic algorithm (GA)
- GA with stochastic ranking
- Evolutionary programming
- Evolution strategy with stochastic ranking

Other stochastic

- Random search
- Simulated annealing
- Particle swarm

Numerical optimisation methods

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

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.

Minimisation of a univariate function (line search)

Line search is carried out assuming the function is quadratic:

- Considering three points a, b, c, an extremum of the line is at point d :

$$d = \frac{1}{2} \cdot \frac{(b^2 - c^2) f_a + (c^2 - a^2) f_b + (a^2 - b^2) f_c}{(b - c) f_a + (c - a) f_b + (a - b) f_c}$$

- This assumes a fit to the hyper-parabola

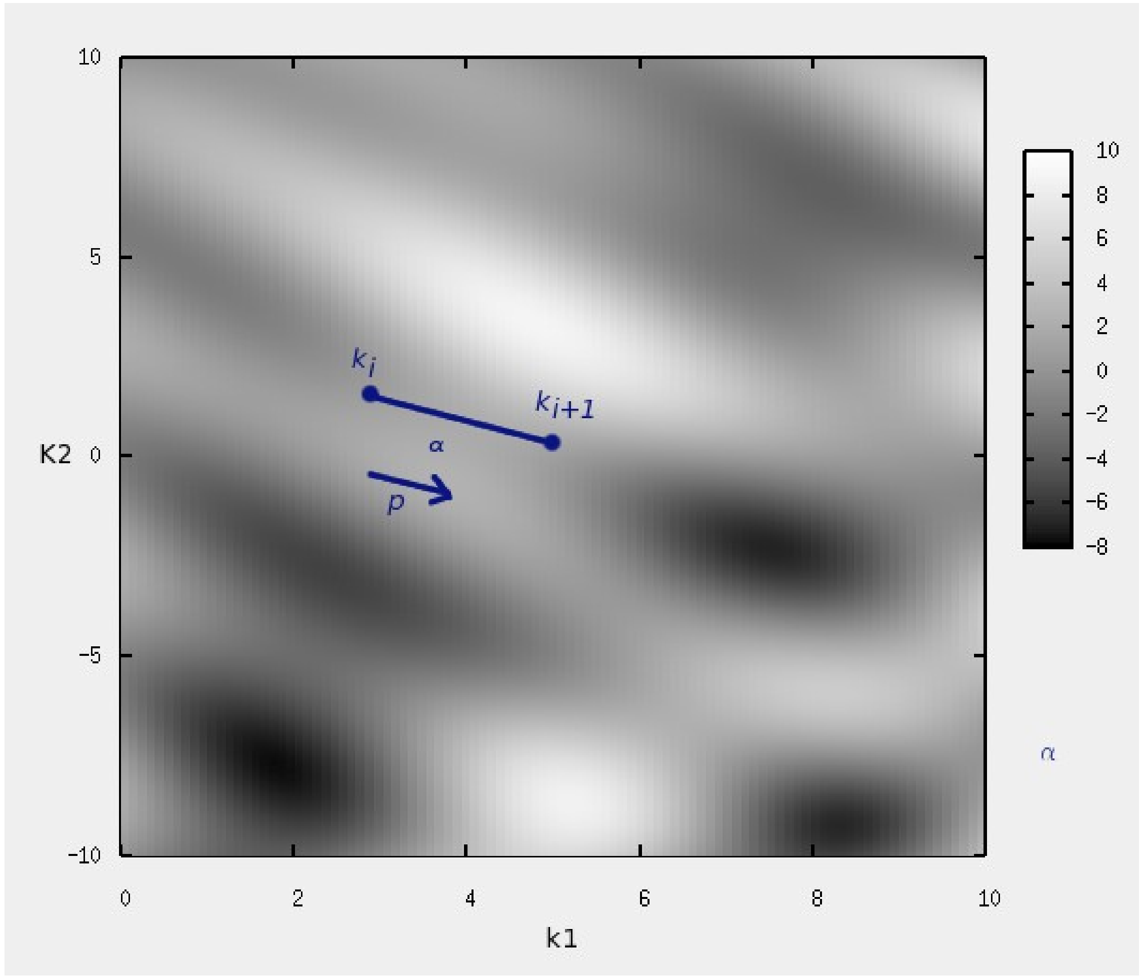
- The extremum is a minimum if

$$\frac{(b - c) f_a + (c - a) f_b + (a - b) f_c}{(a - b)(b - c)(c - a)} < 0$$

- If a maximum was found, move all the way through the longest distance and repeat the procedure again

Parameter update scheme

Given a point in parameter space k_i ,a direction p_i ,and a length α_i $k_{i+1}=k_i+\alpha_i p_i$



Gradient search

Using information from the derivative to find good search directions

Steepest descent

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

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

  - For a small enough α

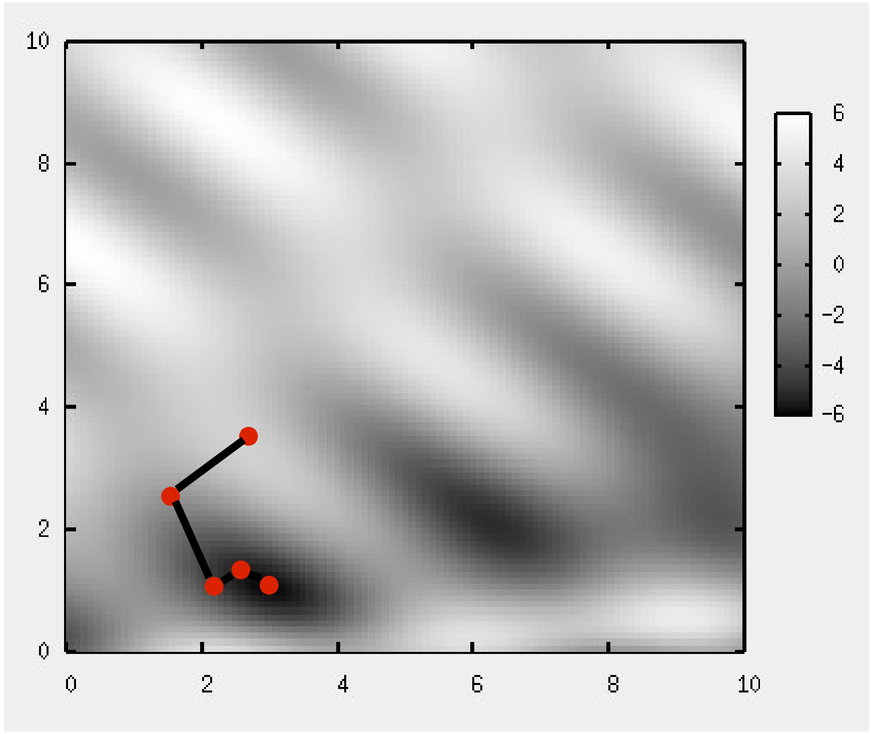
$$f(a) \geq f(b)$$

  - 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

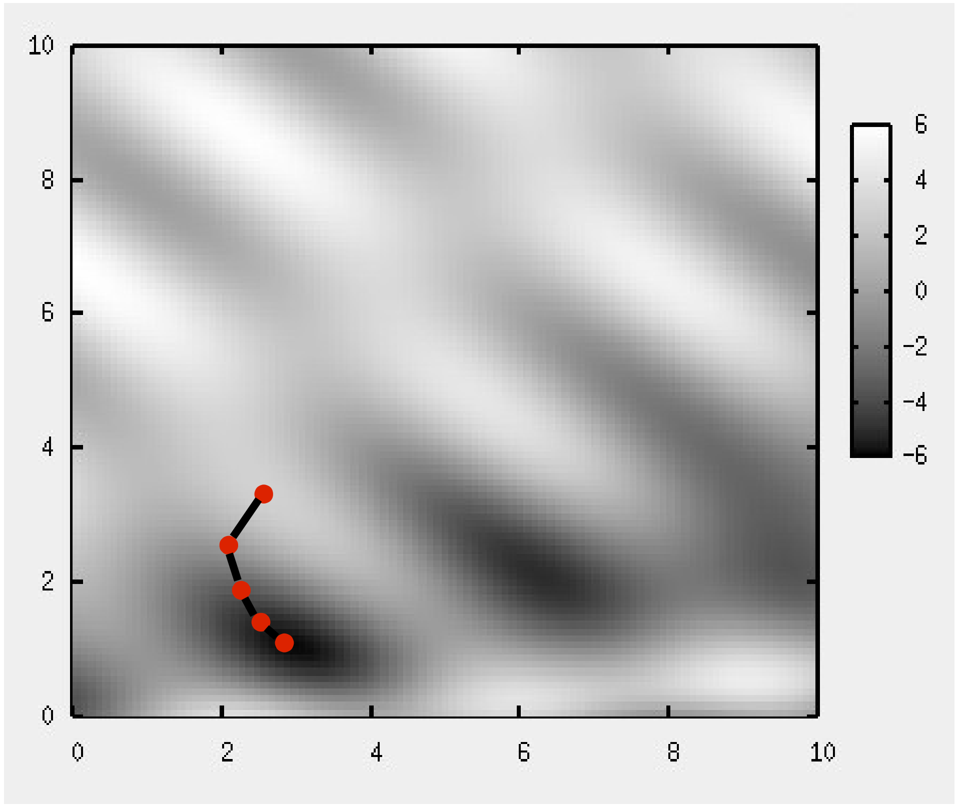
- If F(x) is twice-differentiable, then:

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

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

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

- 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 (L-M)** uses an adaptive linear combination of the steepest descent 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

Bibliography

- Fletcher, R. (1987). *Practical methods of optimization*. 2nd Edition, Chichester: John Wiley & Sons.
- Levenberg, K. (1944) A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math.* **2**, 164-168.
- Marquardt, D. W. (1963) An algorithm for least squares estimation of nonlinear parameters. *SIAM J.* **11**, 431-441.
- Zhu, C., Byrd, R.H. & Nocedal, J. (1997) Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Trans. Math. Soft.* **23**, 550-560.
- Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P. (1992) *Numerical Recipes in C*, Cambridge University Press, Cambridge, U.K.

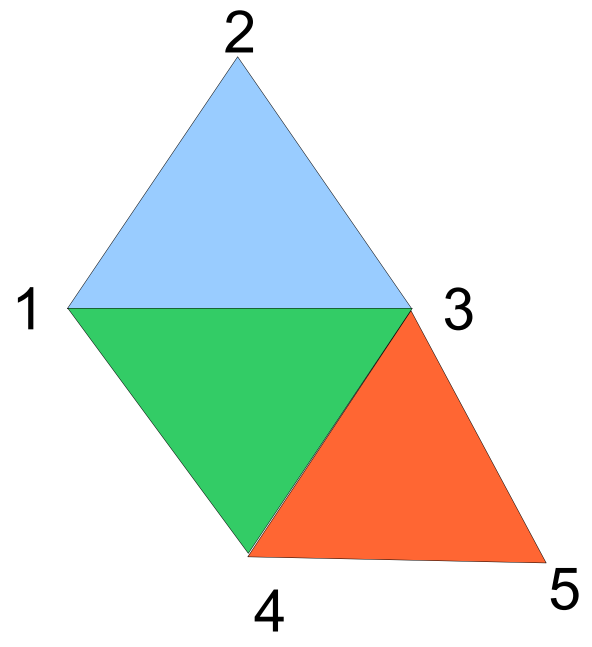
Direct Search

Using memory to find good search directions

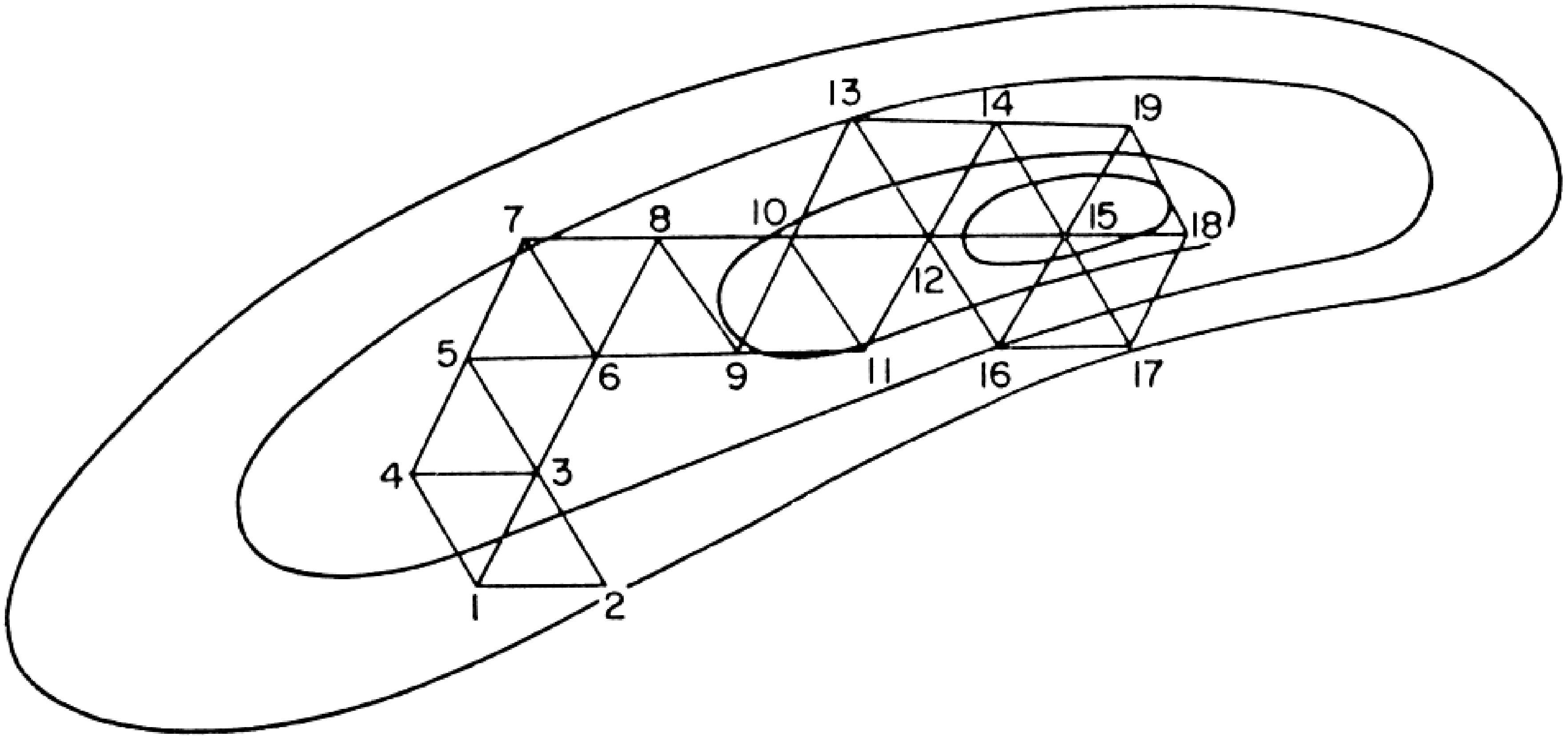
Simplex methods

- 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:



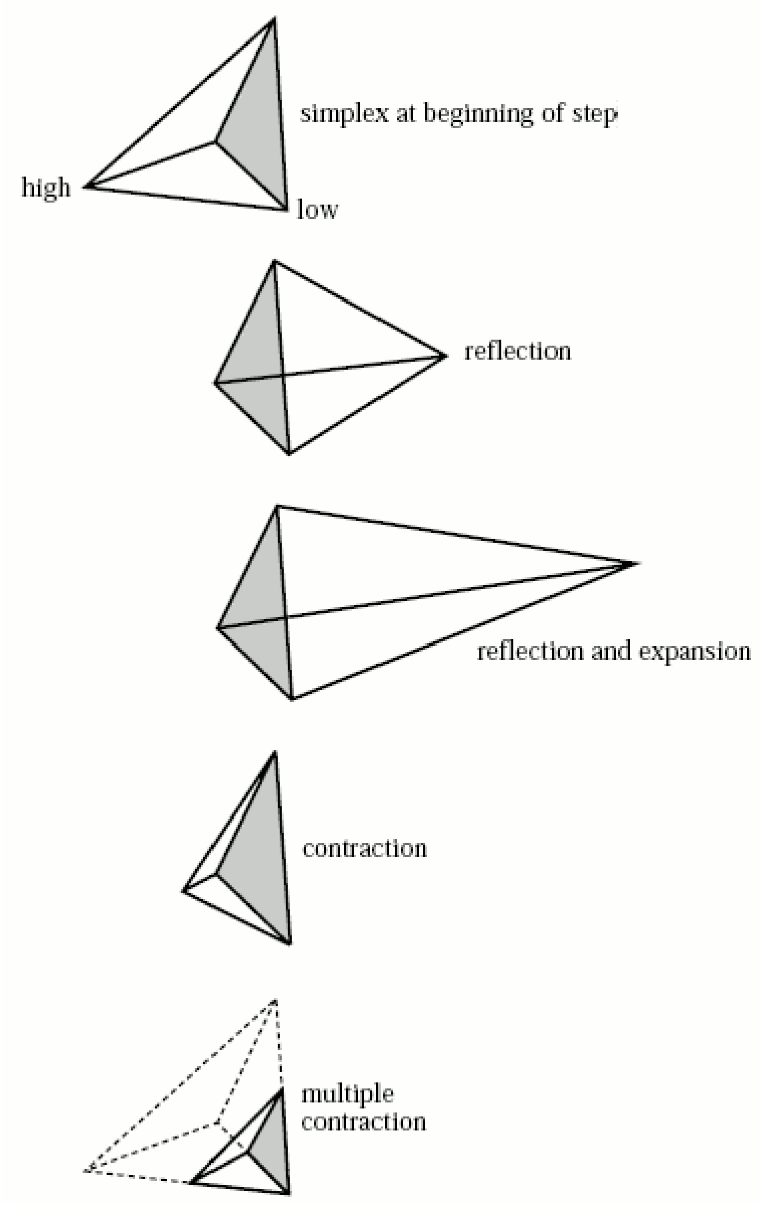
The simplex method of Spendley, Hext and Himsworth (1962)



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



Bibliography

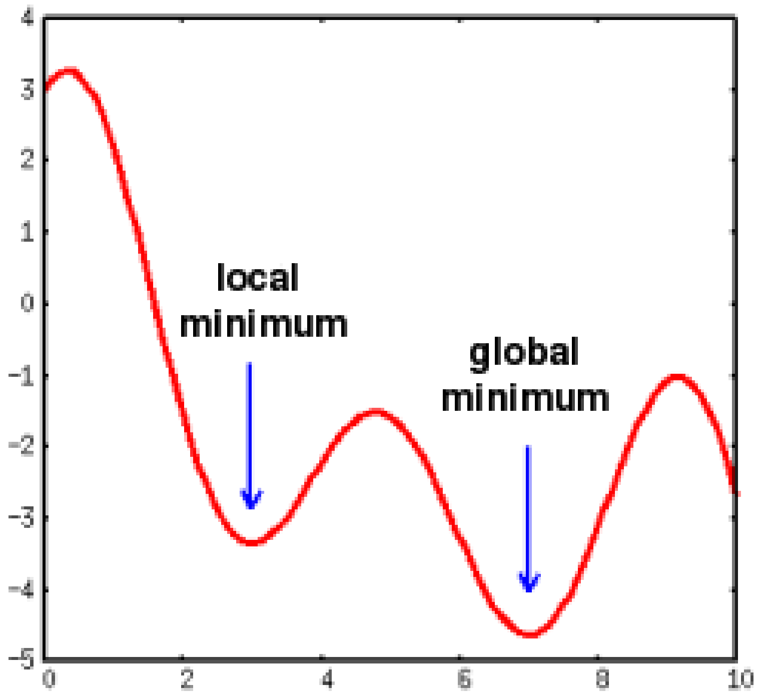
- Spendley, W., Hext, G.R. & Himsworth, F.R. (1962) “Sequential application of simplex designs in optimization and evolutionary operation” *Technometrics* **4**, 441-461
- Nelder, J.A. & Mead, R. (1965) “A simplex method for function minimization” *Computer J.* **7**, 308-313
- Swann, W.H. (1972) “Direct search methods” *in Numerical methods for unconstrained optimization* (W. Murray Ed.), pp. 13-28
- Olsson, D.M. (1975) “The Nelder-Mead simplex procedure for function minimization” *Technometrics* **17**, 45-51
- Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P. (1992) *Numerical Recipes in C*, Cambridge University Press, Cambridge, U.K.

Stochastic methods

Using probabilistic methods to find good search directions

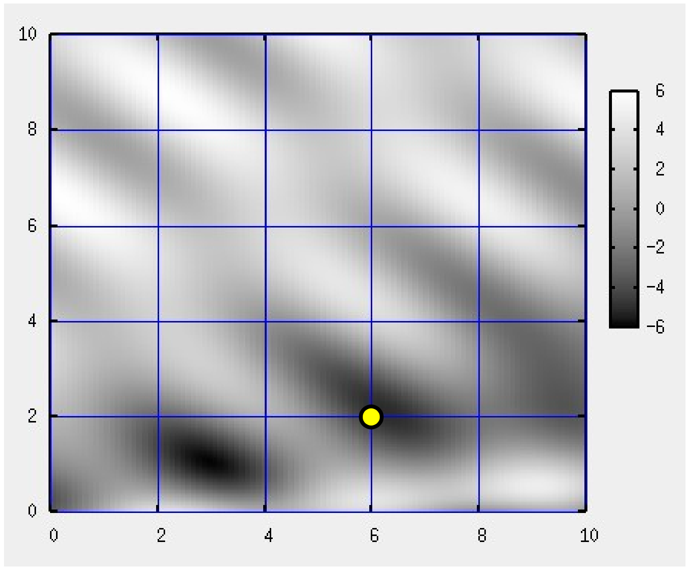
Global optimization

Global optimisation is the task of finding the absolutely best set of parameter values under a given **objective** in the admissible region of the parameter space (**constraints**)

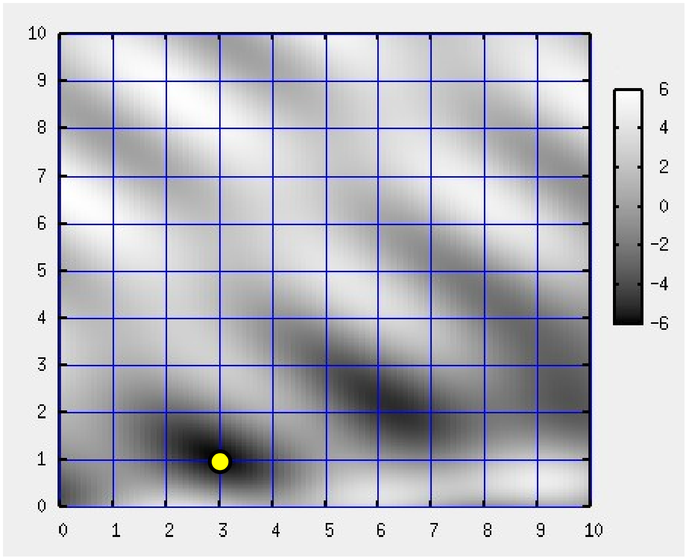


Grid search

- 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



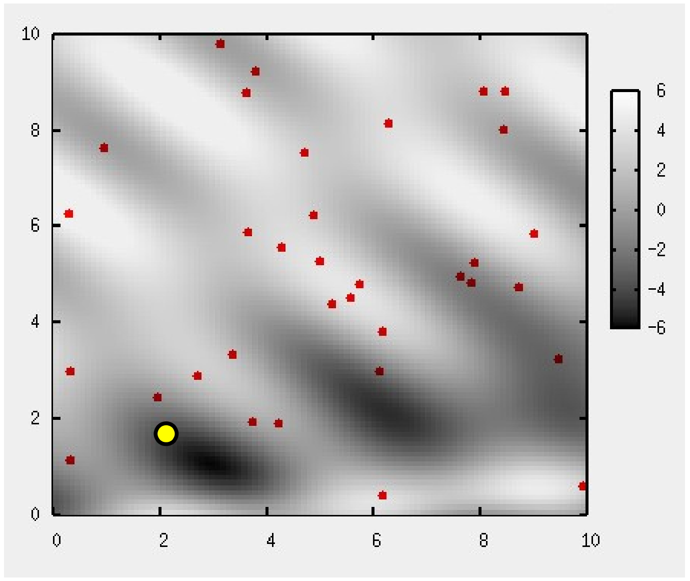
p=2, n=6, total = 6^2 = 36



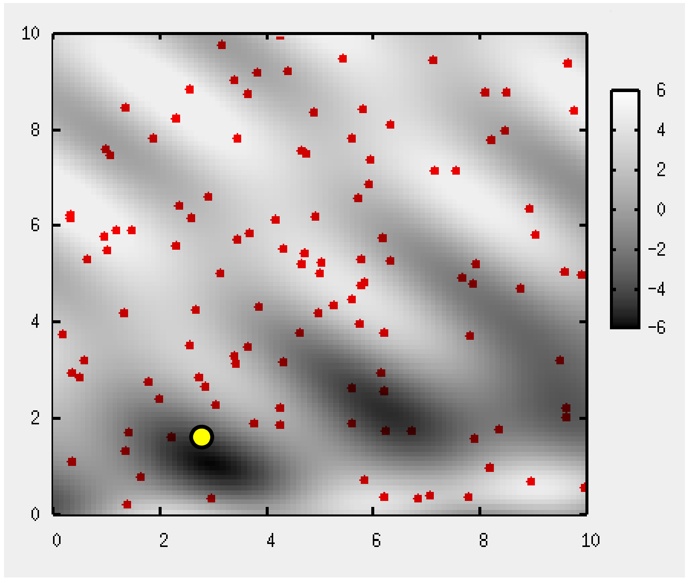
p=2, n=11, total = 11^2 = 121

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 number of tested points
- Usually performs very badly
- … but sometimes can outperform all other methods



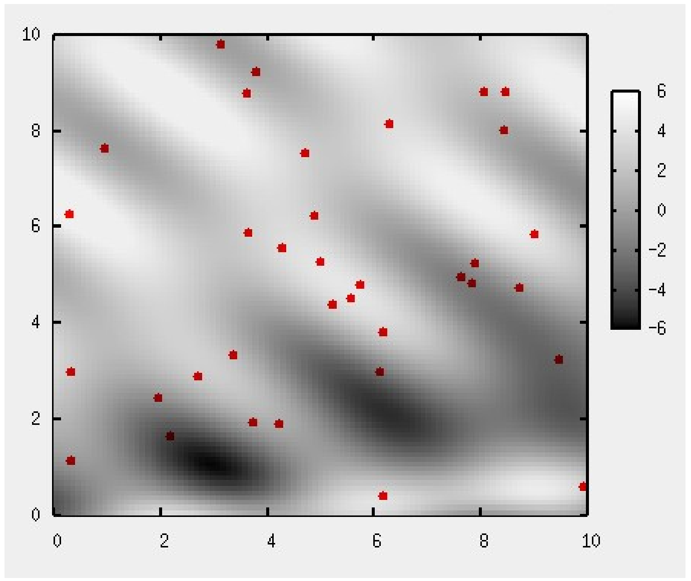
total = 36



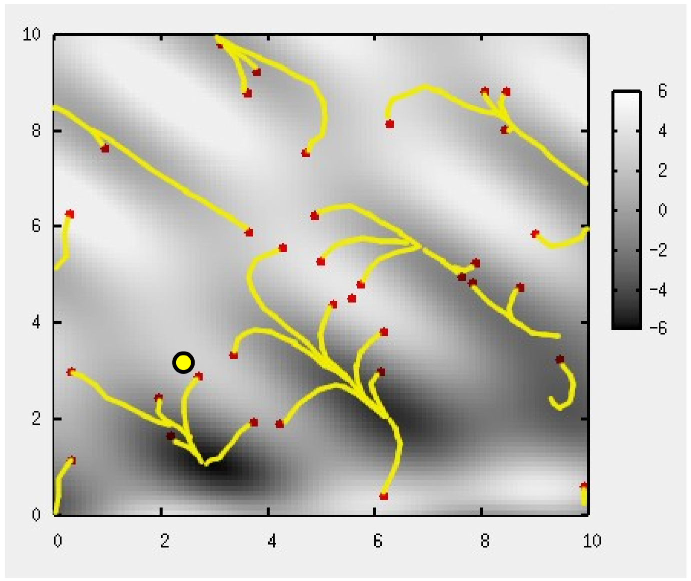
total = 121

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 minima are found. May be too costly (time)
- A further improvement is to cluster points so as not to visit clusters 2nd time



total = 36



total = 36

Minimisation 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 ensemble 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 exists 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

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

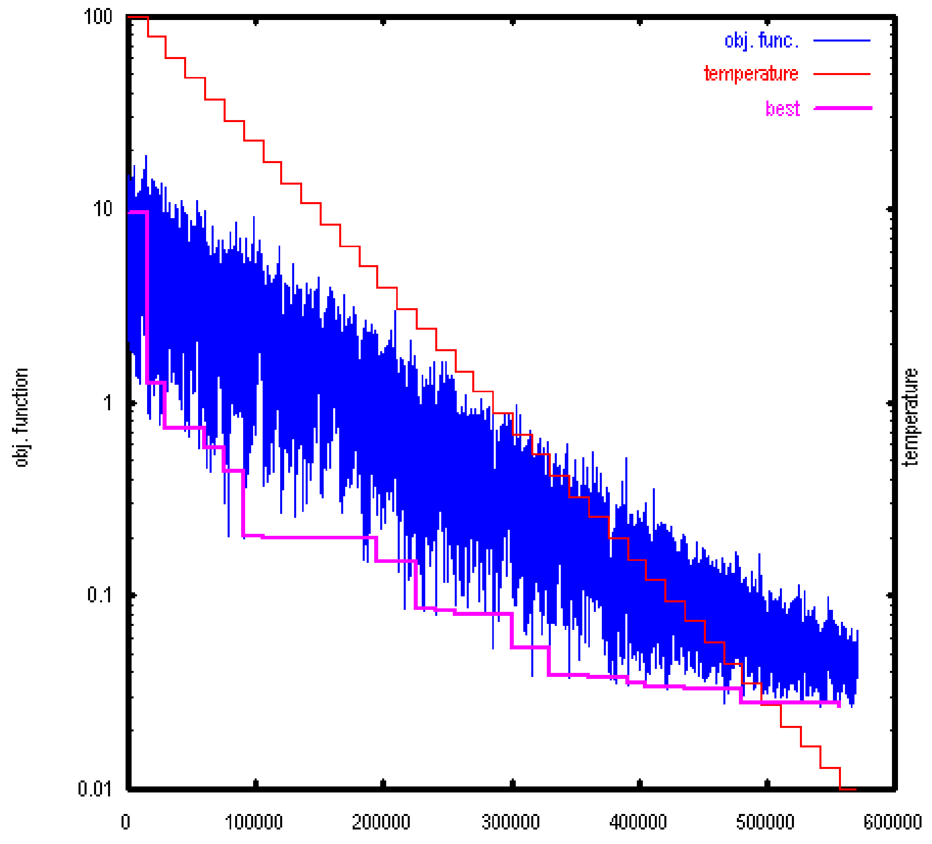
4. Iterate algorithm a large number of times, simulating the thermal motion of particles in a heat bath of temperature T
5. This choice of probability $P(\Delta E)$ evolves the system to a Boltzmann distribution
6. Note that the Metropolis algorithm allows the energy to increase (though, with probability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

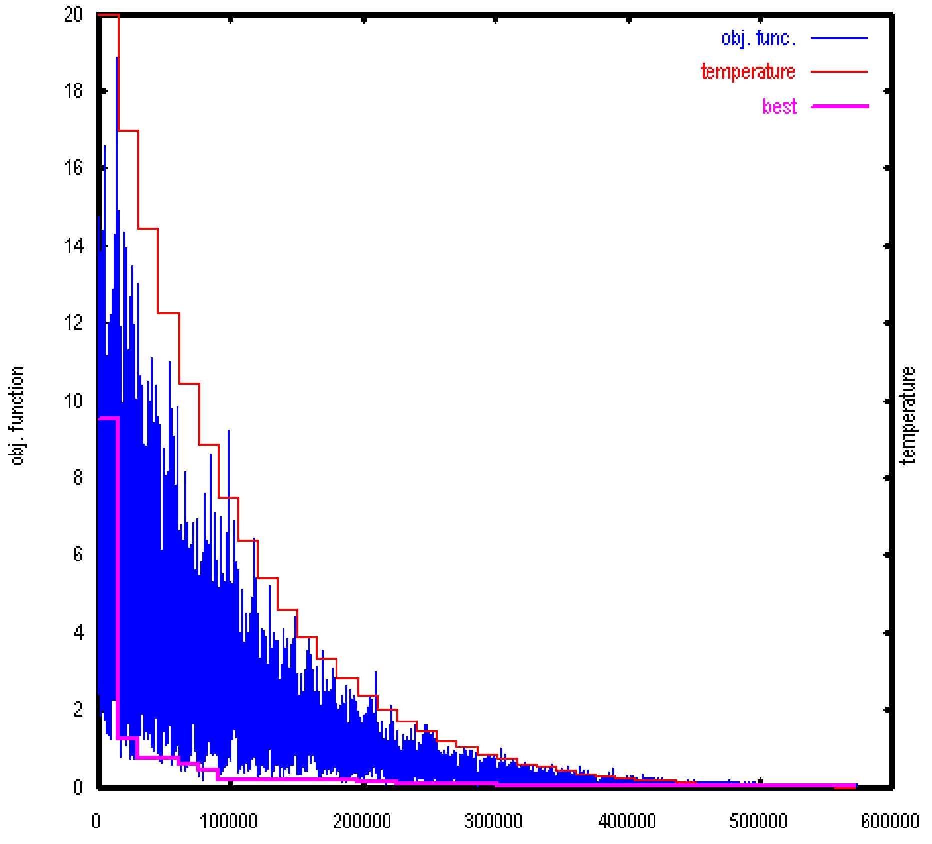
Simulated annealing

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)

Convergence of simulated annealing





Bibliography

- Bohachevsky, I.O., Johnson, M.E. & Stein, M.L. (1986) "Generalized simulated annealing for function optimization" *Technometrics* 28, 209-217.
- Corana, A., Marchesi, C. & Ridella, S. (1987) "Minimizing multimodal functions of continuous variables with the simulated annealing algorithm" *ACM Trans. Math. Softw.* 13, 262-280
- Kirkpatrick, S., Gelatt, C.D. & Vecchi, M.P. (1983) "Optimization by simulated annealing" *Science* 671-680
- Locatelli, M. (1996) "Convergence properties of simulated annealing for continuous global optimization" *J. Appl. Prob.* 33, 1127-1140.
- Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N. & Teller, A.H. (1953) "Equation of state calculations by fast computing machines" *J. Chem. Phys.* 21, 1087-1092
- Mitra, D., Romeo, F. & Sangiovanni-Vincentelli, A. (1986) "Convergence and finite-time behavior of simulated annealing" *Adv. Appl. Prob.* 18, 747-771
- Neumaier, A. (2004) "Complete Search in Continuous Global Optimization and Constraint Satisfaction" in *Acta Numerica* (A. Iserles, ed.), Cambridge University Press.
- Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P. (1992) *Numerical Recipes in C*, Cambridge University Press, Cambridge, U.K.

Evolutionary algorithms

Using populations and selection to optimize functions

Minimization by analogy with Nature
(evolutionary algorithms)

- Candidate solutions of the problem are formulated as **genotypes** (*e.g.* strings of numbers)
- Optimisation proceeds through repeated mutation and selection stages on the **population of candidate solutions**
- The **fitness function** (how good a solution is) influences the selection process

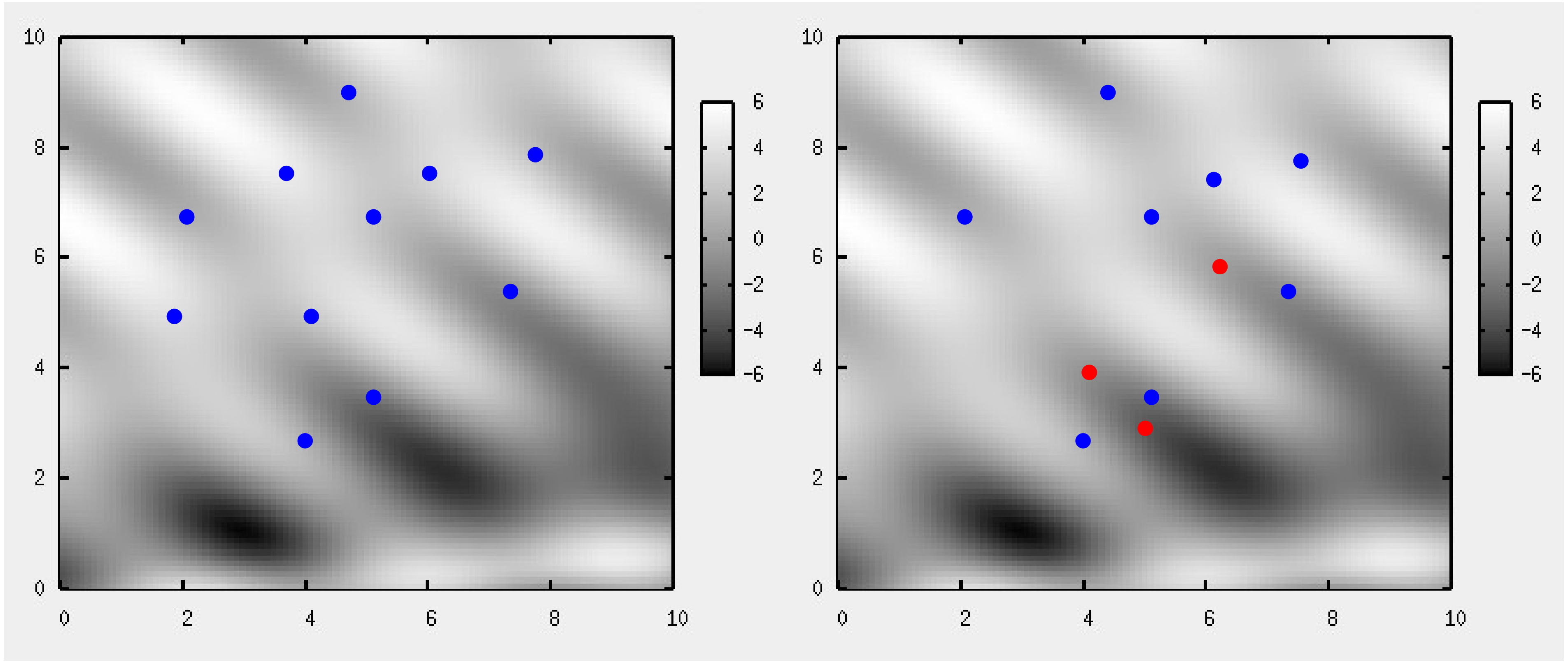
Evolutionary Algorithms

- Populations evolve by the action of **variation/mutation** 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 → set of all parameters
- Individual → candidate solution
- Generation → iteration
- Fitness → objective function value

Evolutionary algorithms evolve populations of solutions

Generation *n* → Generation *n*+1



Remark: direction-length paradigm is no longer appropriate

Evolutionary programming
(Fogel et al., 1966)

Parameters are encoded as real numbers: genes are numbers

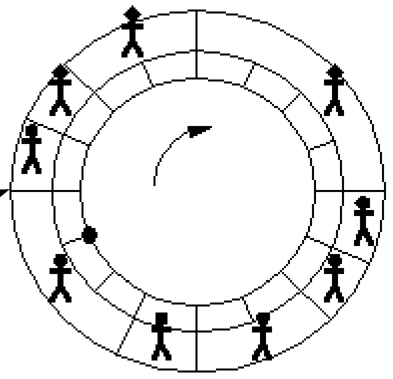
1. Generate a random initial population of *n* individuals
2. Calculate the fitness of each individual in the population
3. Each individual from the current population generates an offspring by copying its own genes
4. **Mutate** each locus in the offspring with a small variance
5. Put the offspring 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...
  - Roulette 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)

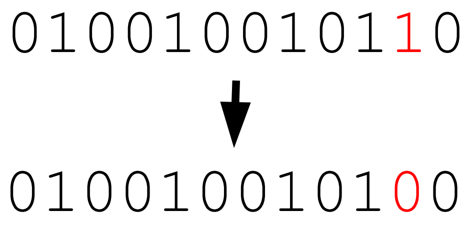
Parameters are encoded in binary: genes are strings of binary digits

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

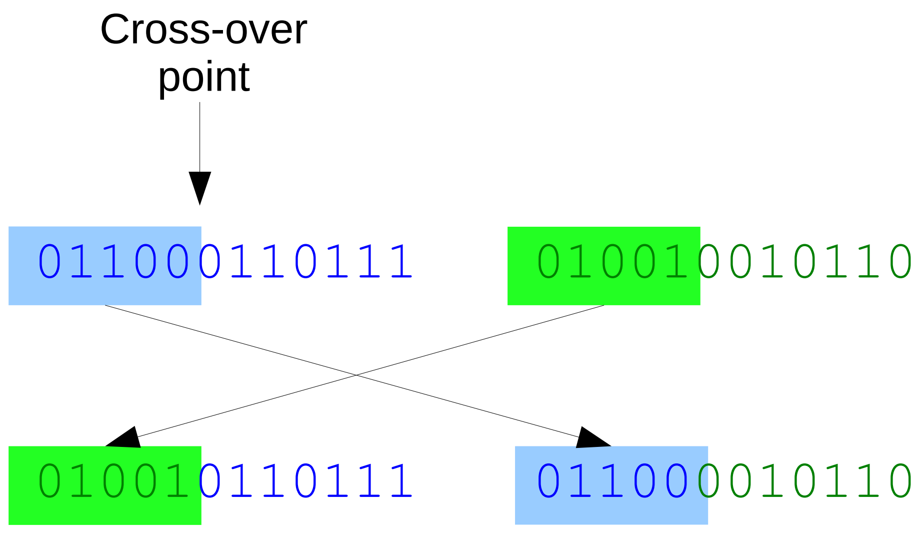
Mutation and cross-over operators

- **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



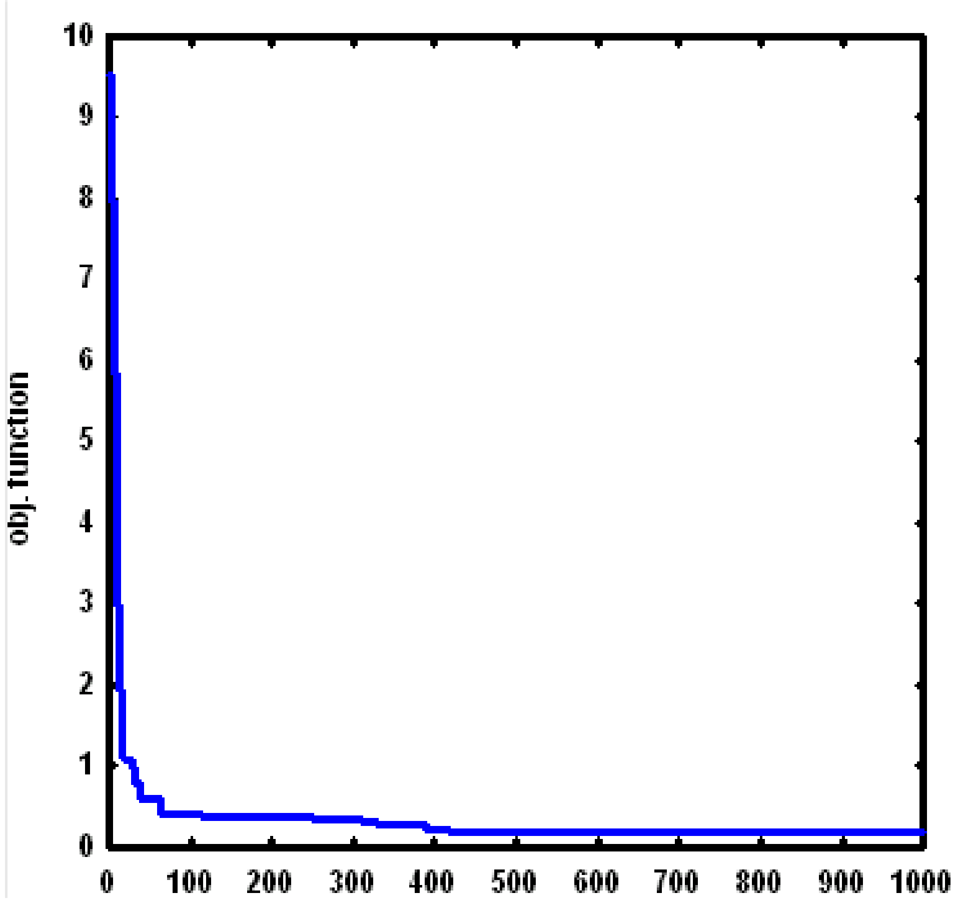
Selection *vs.* variation

- 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 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

Convergence properties of evolutionary algorithms

- It is possible to derive convergence properties of evolutionary algorithms applied to simple problems
- In general convergence properties do not exist
- Typically, evolutionary algorithms proceed as “punctuated evolution”, where there are long periods of stagnation and periods of very rapid progress

Typical convergence



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

Bibliography

- Bäck, T. (1996) *Evolutionary algorithms in theory and practice.* Oxford University Press, Oxford.
- De Jong, K. (1975) *An analysis of the behaviour of a class of genetic adaptive systems.* PhD thesis, The University of Michigan.
- Fogel, L.J., Owens, A.J. & Walsh, M.J. (1966) *Artificial intelligence through simulated evolution.* John Wiley
- Holland, J.H. (1975) *Adaptation in natural and artificial systems.* The University of Michigan Press, Ann Arbor, MI.
- M. Mitchell. (1996) *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.
- Rechenberg, I. (1973) *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog Verlag, Stuttgart
- Schwefel, H.-P. (1975) *Evolutionsstrategie und numerische Optimierung.* Ph.D. Thesis, Technische Universität Berlin

Administrative stuff

- There will be a (brief) exercise by Aleksandr Andreychenko on Thursday, 5th of July
- We will have no lecture on Tuesday, 10th of July
- The questions on optimisation will be covered in the exercise on Thursday, 12th of July