

Computer simulations in Physics

Optimization

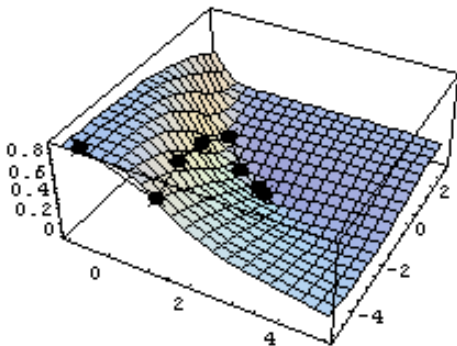
Janos Török

Department of Theoretical Physics

April 7, 2022

Optimization

- ▶ General problem of finding the ground state
- ▶ Phase-space:
- ▶ Arbitrary number of dimensions
- ▶ Methods:
 - ▶ Steepest Descent
 - ▶ Stimulated Annealing
 - ▶ Genetic algorithm



Optimization

- ▶ General problem of finding the ground state
- ▶ Phase-space:
- ▶ Arbitrary number of dimensions
- ▶ Methods:
 - ▶ Steepest Descent
 - ▶ Stimulated Annealing
 - ▶ Genetic algorithm
- ▶ Implementation
 - ▶ C: GSL
 - ▶ python: `scipy.optimize`
 - ▶ Both are very flexible and can be used with numerical or analytical derivatives

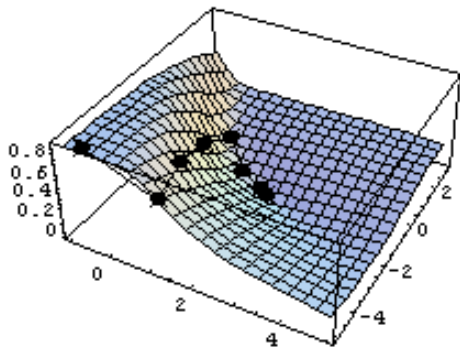
Gradient based optimization

- ▶ Given $f(x)$, with $x = \{x_1, x_2, \dots, x_n\}$
- ▶ Gradient $\nabla f(x) \equiv g(x) = \{\partial_1 f, \partial_2 f, \dots, \partial_n f\}$
- ▶ Second order partial derivatives: square symmetric matrix called the *Hessian matrix*:

$$\nabla^2 f(x) \equiv H(x) \equiv \begin{pmatrix} \partial_1 \partial_1 f & \dots & \partial_1 \partial_n f \\ \vdots & \ddots & \vdots \\ \partial_1 \partial_n f & \dots & \partial_n \partial_n f \end{pmatrix}$$

General Gradient Algorithm

1. Test for convergence
2. Compute a search direction
3. Compute a step length
4. Update x



Steepest descent algorithm

1. Start from x_0
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $\|g(x_k)\| \leq \varepsilon_g$ then stop, otherwise, compute normalized search direction
$$p_k = -g(x_k)/\|g(x_k)\|$$
3. Compute α_k such that $f(x_k + \alpha p_k)$ is minimized
4. New point: $x_{k+1} = x_k + \alpha p_k$
5. Test for $|f(x_{k+1}) - f(x_k)| \leq \varepsilon_a + \varepsilon_r |f(x_k)|$ and stop if fulfilled in two successive iterations, otherwise go to 2.

Conjugate Gradient Method

- ▶ The iteration

$$\mathbf{x}_{n+1} = \mathbf{x}_k - \gamma_n \nabla f(\mathbf{x}_k),$$

- ▶ We can select γ such that if the function is quadratic in all directions it goes immediately into the minimum
- ▶ Idea: almost all minima are quadratic close to the minimum

$$\gamma_n = \frac{|(\mathbf{x}_n - \mathbf{x}_{n-1})^T [\nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n-1})]|}{\|\nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n-1})\|^2}$$

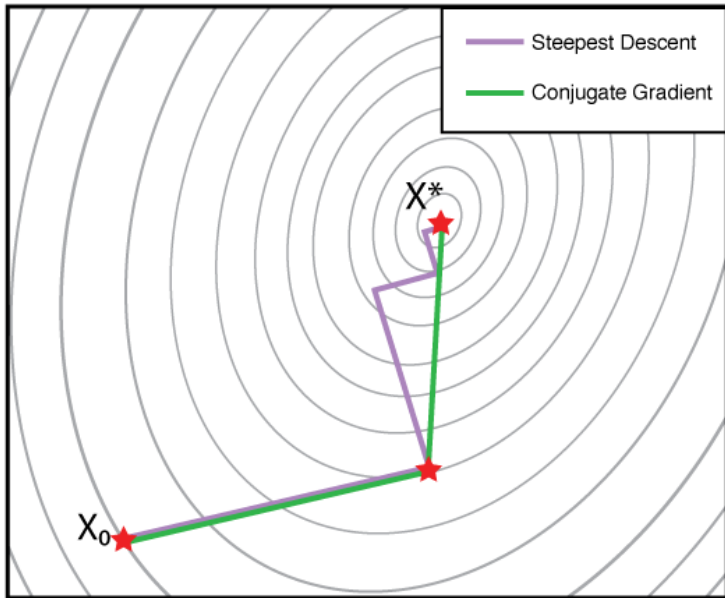
Conjugate Gradient Method

1. Start from x_0
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $\|g(x_k)\| \leq \varepsilon_g$ then stop, otherwise Go to 6
3. $p_0 = -g_0$
4. Compute $g(x_k) \equiv \nabla f(x_k)$. If $\|g(x_k)\| \leq \varepsilon_g$ then stop, otherwise continue
5. Compute the new conjugate gradient direction
 $p_k = -g_k + \beta_k p_{k-1}$, where

$$\beta = \left(\frac{\|g_k\|}{\|g_{k-1}\|} \right)^2$$

6. Compute α_k such that $f(x_k + \alpha p_k)$ is minimized
7. New point: $x_{k+1} = x_k + \alpha p_k$
8. Test for $|f(x_{k+1}) - f(x_k)| \leq \varepsilon_a + \varepsilon_r |f(x_k)|$ and stop if fulfilled in two successive iterations, otherwise go to 4.

Conjugate Gradient Algorithm



Modified Newton's method

Second order method

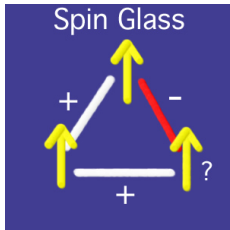
1. Start from x_0
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $\|g(x_k)\| \leq \varepsilon_g$ then stop, otherwise, continue
3. Compute $H(x_k) \equiv \nabla^2 f(x_k)$ and the search direction $p_k = -H^{-1}g_k$
4. Compute α_k such that $f(x_k + \alpha p_k)$ is minimized
5. New point: $x_{k+1} = x_k + \alpha p_k$
6. Go to 2.

Glassy behavior, frustration

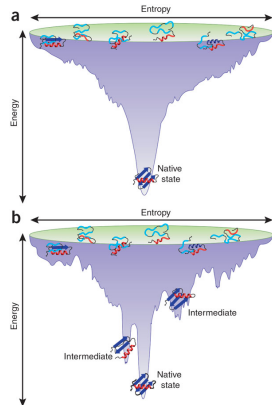
- Model glass: spin glass:

$$H = -\frac{1}{2} \sum_{\langle i,j \rangle} J_{ij} S_i S_j$$

- where J_{ij} are random quenched variables with 0 mean (e.g. $\pm J$ with probability half)

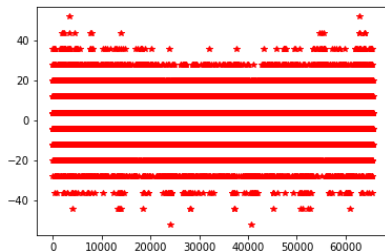
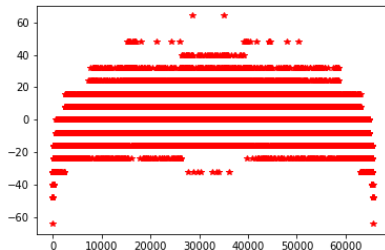


Rugged energy landscape.



Energy landscape

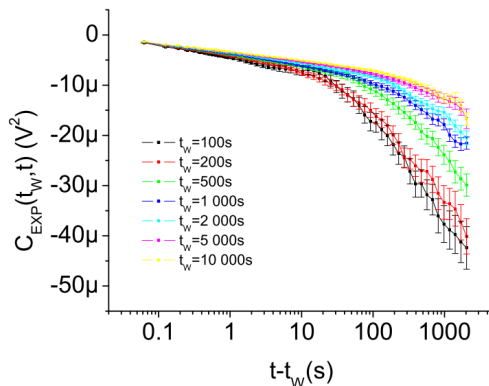
Ising vs. spin glass (X Axis: binary representation of number)



Spin glass: Aging

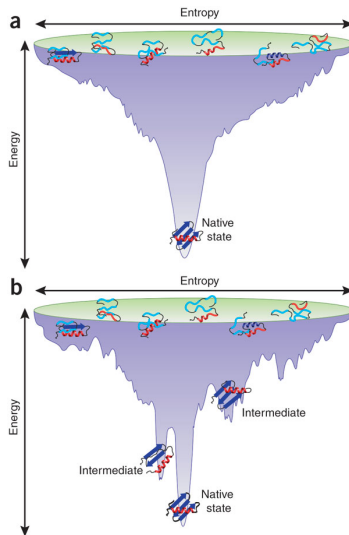
- ▶ Heat up the sample where it equilibrates fast
- ▶ Quench it below T_c
- ▶ Wait t_w
- ▶ Measure a parameter $q(t_w, t_w + t)$
- ▶ Often q is a covariance (X observable):

$$q(s, t) = E(X_t X_s) - E(X_t)E(X_s)$$



Spin glass: Trap model (Bouchaud)

- ▶ The evolution of the particle system is represented by a Markov process in a random energy landscape
- ▶ The process will spend most time into deep valleys of lowest energy where it will be trapped
- ▶ The time spent in these valleys is random and aging will appear when the mean time spent in these valleys diverges
- ▶ Order parameter: the magnetization and the two point spin correlation between spins at the same site in two different replicas



Rugged energy landscape

- ▶ Typical example NP-complete problems:
 - ▶ Traveling salesman
 - ▶ Graph partitioning
 - ▶ Spin glass
- ▶ No full optimization is possible (do we need it?)
- ▶ Very good minimas can be obtained by stochastic optimization
 - ▶ Simulated annealing
 - ▶ Genetic algorithm

Optimization

- ▶ General optimization
- ▶ Parameters of the system x (input)
 - ▶ for networks: adjacency matrix, degree distribution
 - ▶ for pattern recognition: data, or processed data (e.g. Fourier spectrum, etc.)
- ▶ Optimized property: $y = f(x)$, we search for $f(\cdot)$ which gives the desired y
 - ▶ any measurable quantity
 - ▶ classification of data (e.g. $y = 1$ for cat, $y = 2$ for dog, etc.)
- ▶ Loss function, $L(f)$, the quantity to be minimized (Energy/Hamiltonian)
 - ▶ Least square: $L(f) = [y - f(x)]^2$
 - ▶ Hamming distance: $L(f) = \begin{cases} 1 & \text{if } f(x) \neq y \\ 0 & \text{otherwise} \end{cases}$

Linear regression

- ▶ Assume linear form for the loss function:

$$f_w(x) = w^T x$$

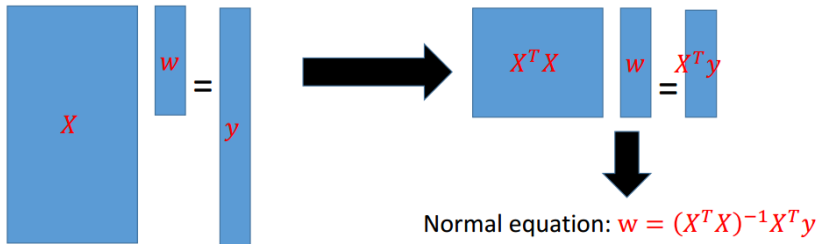
- ▶ n data points
- ▶ m variables
- ▶ The problem must not be linear, e.g. polynomial fit w contains the coefficients of the polynomial (i indexes data points):

$$y_i = w_0 + w_1 x_i^{(1)} + w_2 x_i^{(2)} + \dots + w_m x_i^{(m)} + \varepsilon_i$$
$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ 1 & x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(m)} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

- ▶ If X is a square matrix than $w = X^{-1}y$
- ▶ Otherwise $w = (X^T X)^{-1} X^T y$

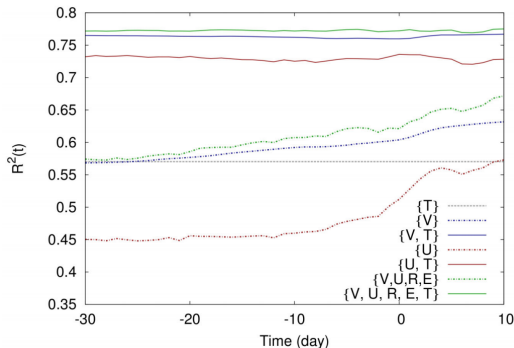
Linear regression

- ▶ If X is a square matrix than $w = X^{-1}y$
- ▶ Otherwise $w = (X^T X)^{-1} X^T y$

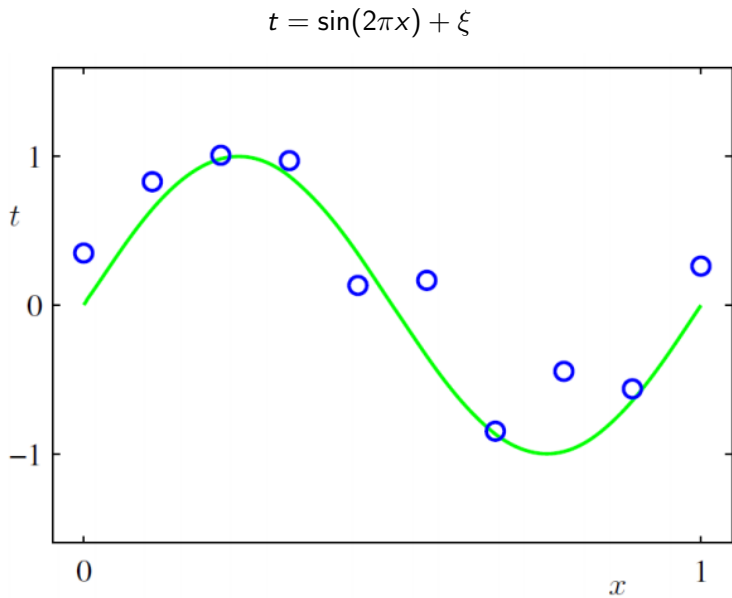


Linear regression: Example

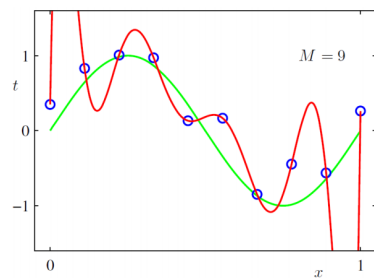
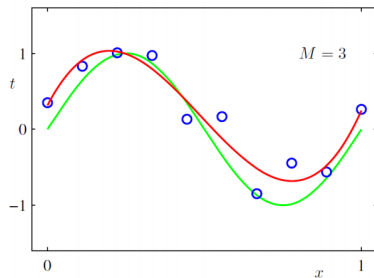
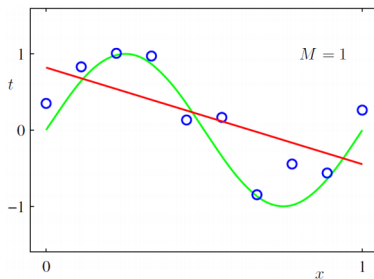
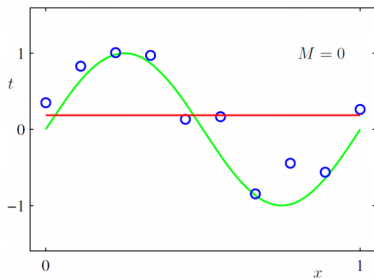
- ▶ Movie success prediction
- ▶ Variables:
 - V Number of views of the Wikipedia page
 - U Number of editors of the Wikipedia page
 - E Number of edits made on the Wikipedia page
 - R Collaborative rigor of Wikipedia editing
 - T Number of theaters that screen the movie
- ▶ Time=0 day of release



Linear regression: Over fitting



Linear regression: Over fitting



Simulated annealing

- ▶ Loss function: e.g. energy E
- ▶ Minimize energy like in a physical system
- ▶ Vary parameter set w in an ergodic way (all possible values must be reachable)
- ▶ Observe detailed balance:

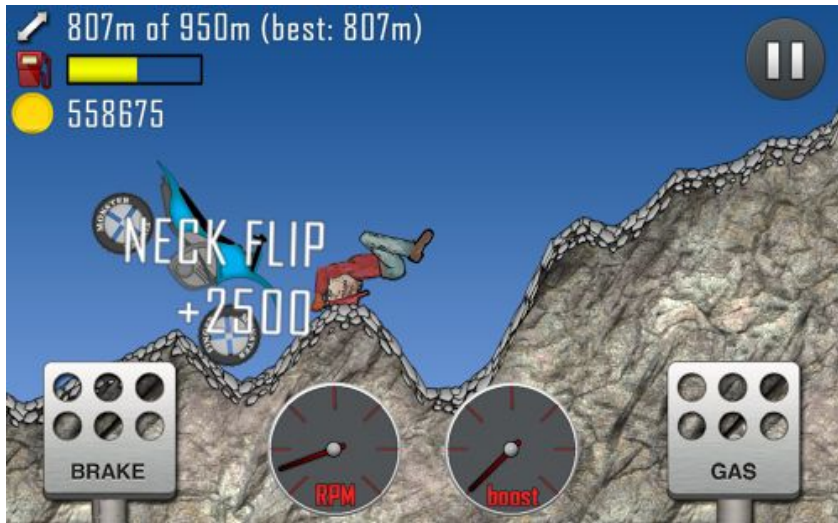
$$p(i \rightarrow j) = \begin{cases} 1 & \text{if } E_j < E_i \\ \exp[\beta(E_i - E_j)] & \text{otherwise} \end{cases}$$

- ▶ where $\beta \simeq 1/T$
- ▶ Slowly decrease T

Simulated annealing

- ▶ Cool down the system slowly
- ▶ Speed is crucial and many experiments are needed
- ▶ No guarantee that we find something meaningful
- ▶ Warm up and down if needed, if the system quenched into a local minimum
- ▶ One needs a Hamiltonian (or a fitness function) and an elementary move
 - ▶ Spin glass: Metropolis

Hill climb



Travelling salesman

- ▶ N cities on the 2d space
- ▶ Distance between the cities is the Euclidean distance (birds flight)
- ▶ The traveller must visit all cities once
- ▶ The trajectory is circular so the traveller must return to the starting city
- ▶ The optimized quantity is the travelled distance

Travelling salesman

- ▶ Minimal travelling path for visiting a number of cities
- ▶ Elementary move: swap two cities ($T \sim \text{alcohol}$)



Genetic algorithm

- ▶ Learn from nature



Genetic algorithm

- ▶ Learn from nature
- ▶ Let the fittest to survive
 - ▶ Fitness function, e.g. energy, length, etc.
- ▶ Combine different strategies
- ▶ State is represented by a vector (genetic code or genotype)
 - ▶ Phasespace, city order, neural network parameters, etc.
- ▶ Offsprings have two parents with shared genetic code
- ▶ Mutations
- ▶ Those who are not fit enough die out
 - ▶ Keep the number of agents fixed



Genetic algorithm terminology

- ▶ Chromosome: Carrier of the genetic representation
- ▶ Gene: Smallest units in the chromosome with individual meaning
- ▶ Fitness: The measure of the success of an individual with a given chromosome
- ▶ Population: Set of chromosomes from which the parents are selected. Its size should be larger than the length of the chromosome
- ▶ Parents: Pair of chromosomes, which produce offsprings
- ▶ Selection principle: The way parents are selected (random, elitistic)
- ▶ Crossover: Recombination of the genes of the parents by mixing
- ▶ Crossover rate: The rate by which crossover takes place ($\sim 90\%$)
- ▶ Mutation: Random change of genes
- ▶ Mutation rate: The rate by which mutation takes place ($\sim 1\%$)
- ▶ Generation: The pool after one sweep.

Genetic algorithm schema

1. Start with a randomly generated population
2. Calculate the fitnesses
3. Selection
 - ▶ Random
 - ▶ Best fitness (keep top 50% and generate new 50%)
 - ▶ Roulette (Monte-Carlo) selection
4. Crossover: offsprings must be viable (Sometimes difficult)

Parents

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

9	8	7	6	5	4	3	2	1
---	---	---	---	---	---	---	---	---

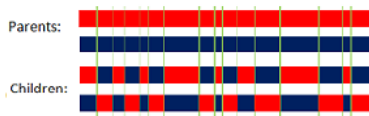
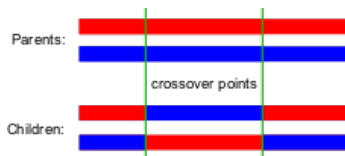
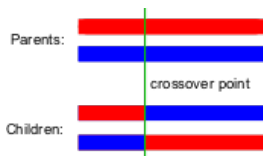
Offspring

					6	7	8	
--	--	--	--	--	---	---	---	--

9	5	4	3	2	6	7	8	1
---	---	---	---	---	---	---	---	---

Genetic algorithm: Reproduction

► Two parents and two children

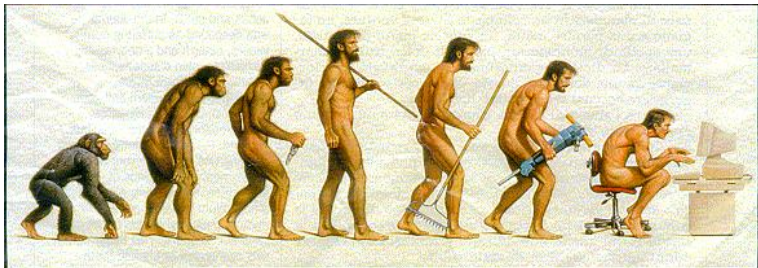


With a probability of 0.5, children have 50% genes from first parent and 50% of genes from second parent even with randomly chosen crossover points.

Genetic algorithm schema

1. Start with a randomly generated population
2. Calculate the fitnesses
3. Selection
 - ▶ Random
 - ▶ Best fitness (keep top 50% and generate new 50%)
 - ▶ Roulette (Monte-Carlo) selection
4. Crossover: offsprings must be viable (Sometimes difficult)
 - ▶ One-point
 - ▶ Two-point
 - ▶ Uniform
5. Mutation: small rate

Genetic algorithm example



Practice

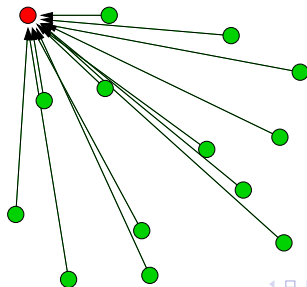
Genetic algorithm

Network optimization example

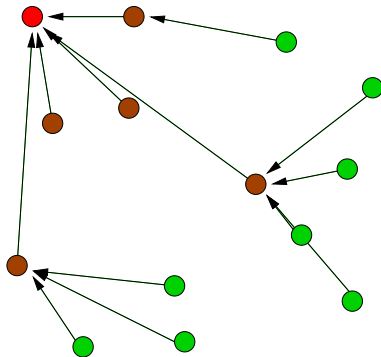
- ▶ Optimize transmission energy consumption (Jin et al. 2003)
- ▶ Sensors scattered in space
- ▶ One data collector
- ▶ Intermediate collectors can be installed
- ▶ Energy consumption

$$E(k, d) = E_{elec} + d^2 E_{amp},$$

where E_{elec} is the base electric need of a radio station, E_{amp} is the energy need of an amplifier and d is the distance to transmit to.

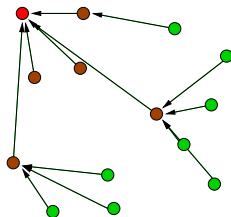


Network optimization example 2.



- Coding: bit
 - 1: direct connection to data collector (it is a cluster head)
 - 0: connection to nearest cluster head

Network optimization example 3.



- Coding: bit

- 1: direct connection to data collector (it is a cluster head)
- 0: connection to nearest cluster head

Indv1: 1 1 1 0 0 1 0 1

Indv2: 1 0 1 1 1 1 1 0

Crossover point

After crossover, two offspring are created as below:

Child1: 1 1 1 0 1 1 1 0

Child2: 1 0 1 1 0 1 0 1

- Crossover:

- Mutation: change a bit

Network optimization example results

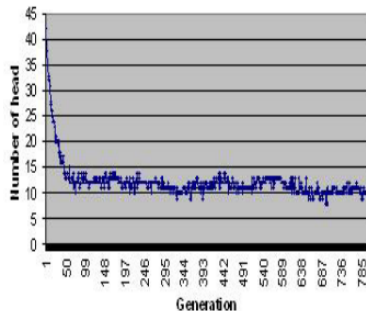
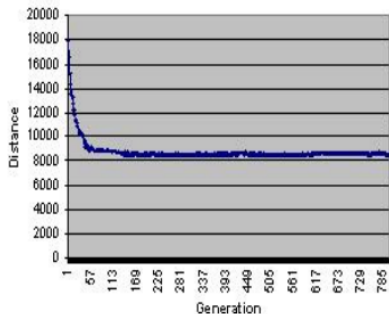


Table 1: Test Results of Three Problem Size

Node Size	Population	Converge after generation	Head (%)	Distance decreased
100	80	105	10.0%	76.85%
200	160	120	10.0%	81.20%
400	300	145	11.2%	82.20%