Monte Carlo Sampling

Chapter 4

2009 Course on Probabilistic Graphical Models
(Artificial Neural Networks,
Studies in Artificial Intelligence and Cognitive Process)

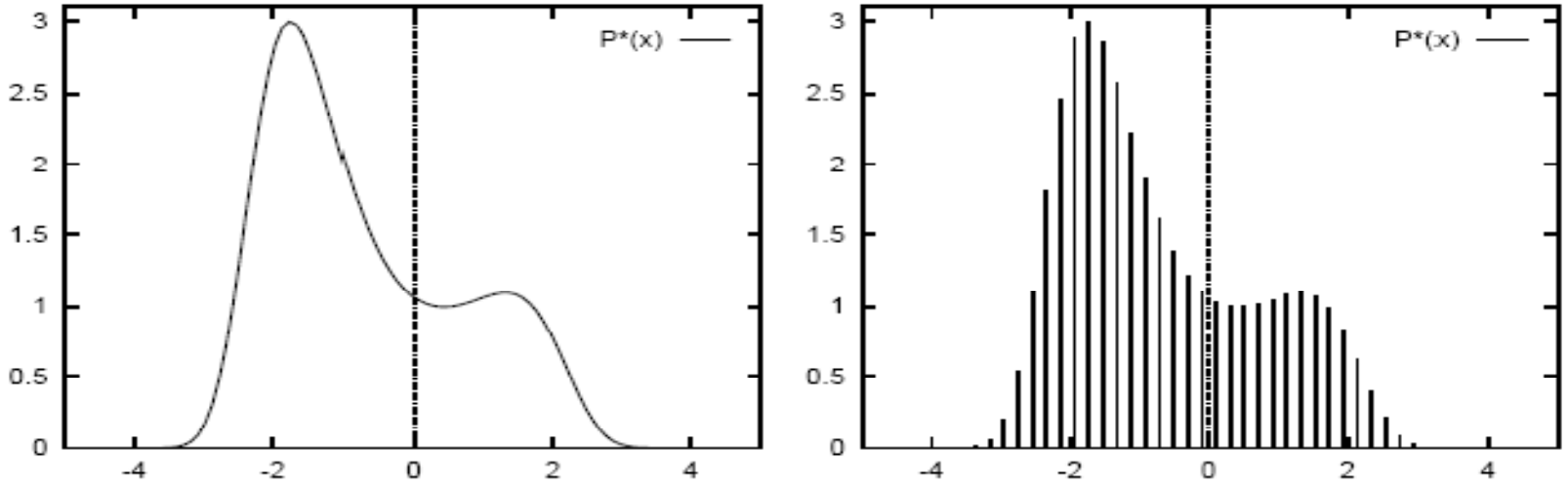
Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University

Overview

- Introduction
- Transformation Method
- Rejection Sampling
- Importance Sampling
  - Sampling-Importance-Resampling (SIR)
- Markov Chain Monte Carlo Methods
  - Metropolis Algorithm
  - Metropolis-Hastings Algorithm
  - Gibbs Sampling
- Evolutionary MCMC

Introduction: The Problem (1/4)

1. Generate samples from a given probability distribution $P(x)$.
2. Finding the expectation of some function $f(x)$ with respect to a probability distribution $P(x)$.
   - Can be approximated by sampling independent points from the distribution P and summation.
   - The accuracy does not depend on the dimensionality of the space to be sampled.



Intro: Monte Carlo Sampling (2/4)

- General tools for approximating probability distributions
- Computational techniques using random numbers
- Two main applications of Monte Carlo methods
  1. Generate samples $\{\mathbf{x}^{(k)} \mid k = 1, \ ..., \ K\}$ from a given p.d.f. $p(\mathbf{x})$
  2. Estimate expectation of a function $f(\mathbf{x})$ under $p(\mathbf{x})$ using random samples $\{\mathbf{x}^{(k)} \mid k = 1, \ ..., \ K\}$

$$\mathrm{E}_p[f(\mathbf{x})] = \langle f(\mathbf{x}) \rangle = \int p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x}$$

$$\cong \frac{1}{K} \sum_{k=1}^{K} f(\mathbf{x}^{(k)})$$

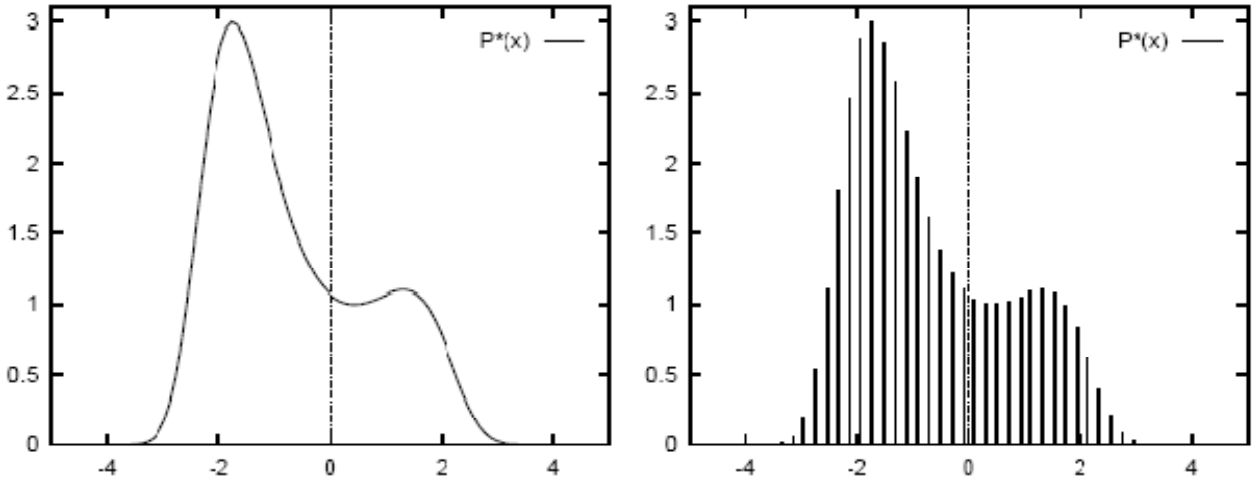
Introduction (3/4)

- Why is sampling from $P(x)$ hard?
  - We typically do not know the normalizing constant.

$$P(\mathbf{x}) = P^*(\mathbf{x})/Z \qquad Z = \int \mathrm{d}^N\mathbf{x}\ P^*(\mathbf{x})$$

  - There is no obvious way to sample from P without enumerating most or all of the possible states.

$$P^*(x) = \exp\left[0.4(x-0.4)^2 - 0.08x^4\right],\ x \in (-\infty, \infty)$$



$$Z = \sum_i p_i^*$$

$$p_i = p_i^*/Z$$

Introduction (4/4)

- Uniform sampling
  - Drawing random samples uniformly from the state space and evaluating $P^*(\mathbf{x})$ at those points
  - This will only stand a chance of giving a good estimate if we make the number of samples R sufficiently large that we are likely to hit the typical set at least once or twice.
  - Uniform sampling is unlikely to be useful in most high-dimensional distribution problems.

Transformation Method

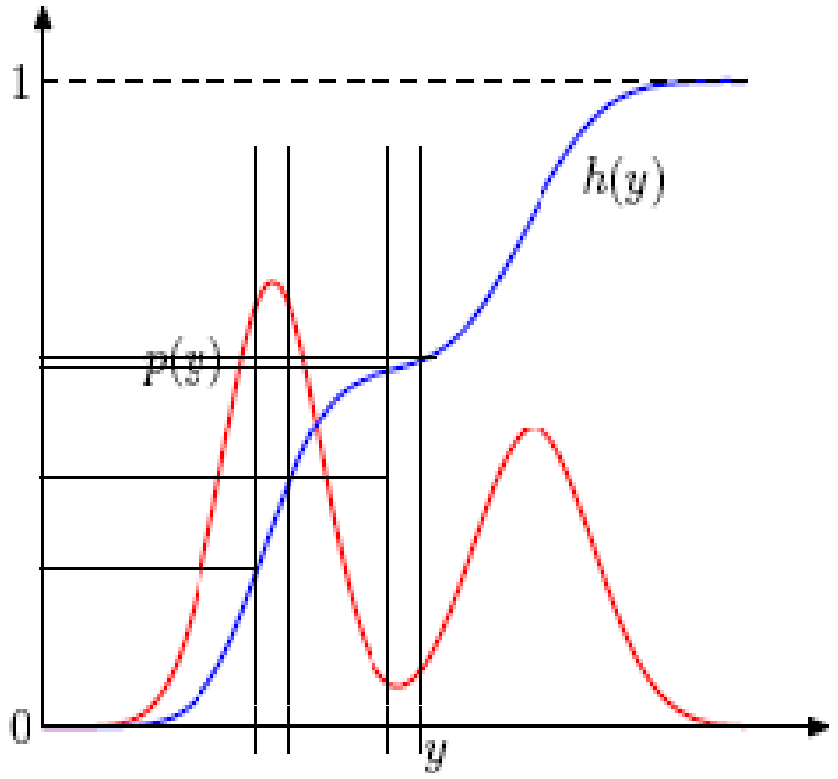
- Transform samples from uniform distribution.

$$P(z) \sim U(0,1)$$

$$z = h(y) \equiv \int_{-\infty}^{y} P(\hat{y})d\hat{y}$$

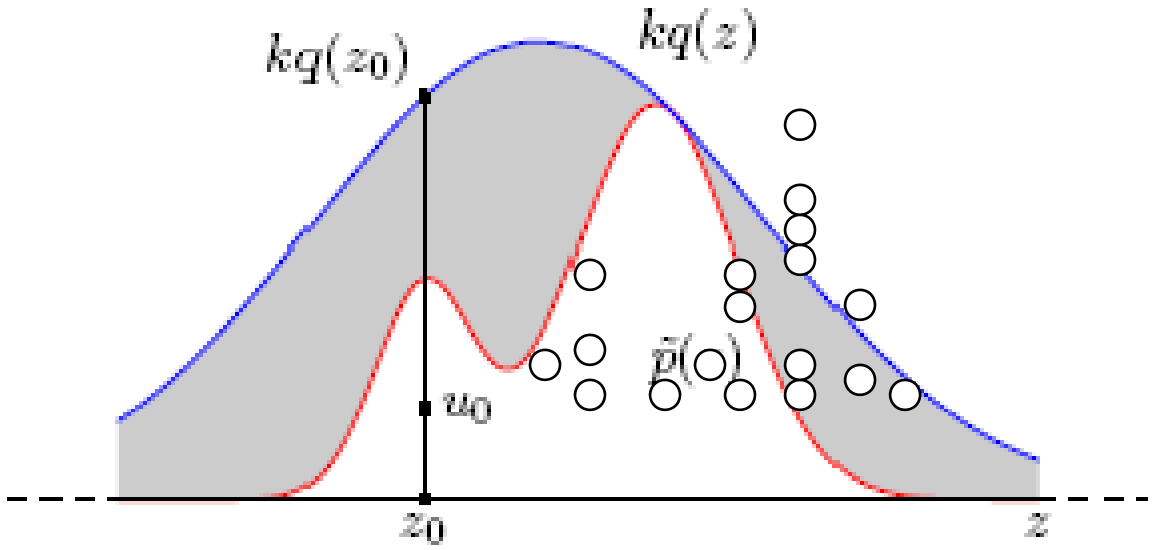
$$y = h^{-1}(z) = f(z)$$

Can we get h^{-1} always?



Rejection Sampling (1/2)

- Assumptions
  - ♦ Sampling directly from target distribution $P(x)$ is difficult.
  - ♦ Estimating $Q(x)$ is easy for any value of x.



1. Generate z_0 from $q(x)$.
2. Generate u_0 from uniform distribution $[0, kq(z_0)]$.
3. If $u_0 > p(x)$, reject (z_0, u_0).
4. Otherwise, accept (z_0, u_0).

How to choose *q*?

Rejection Sampling (2/2)

- In high dimensions,
  - ♦ If *k* is large, the acceptances will be very rare.
    - ■ In general *k* grows exponentially with the dimensionality *N*, so the acceptance rate is expected to be exponentially small in *N*.

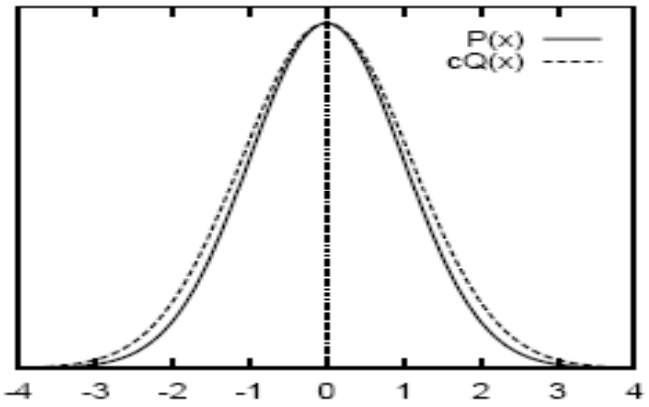


Figure 29.9. A Gaussian $P(x)$ and a slightly broader Gaussian $Q(x)$ scaled up by a factor c such that $cQ(x) \geq P(x)$.

$$c = \frac{(2\pi\sigma_q^2)^{N/2}}{(2\pi\sigma_p^2)^{N/2}} = \exp\left(N \ln \frac{\sigma_q}{\sigma_p}\right)$$

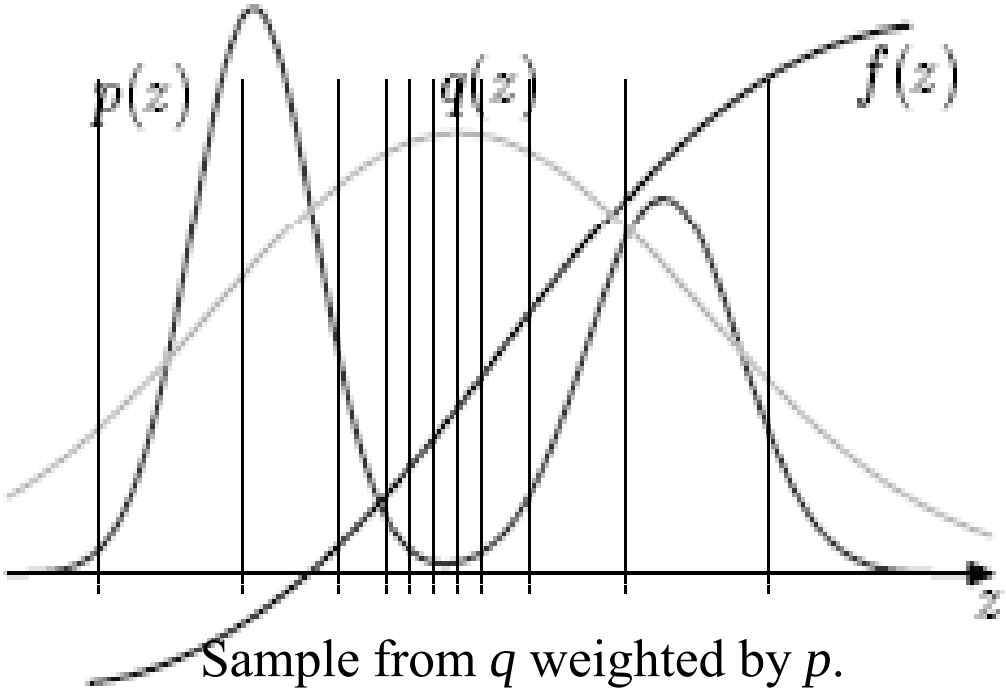
When $N = 1000$, $\sigma_q / \sigma_p = 1.01$, $c \approx 20{,}000$

- High rejection ratio
- Acceptance ratio is expected to be exponentially small with the dimensionality *N*.

Importance Sampling (1/3)

- A method for estimating the expectation of a function
- A generalization of the uniform sampling
- We have a simpler density $Q(x)$ from which we can generate samples and which we can evaluate to within a multiplicative constant.

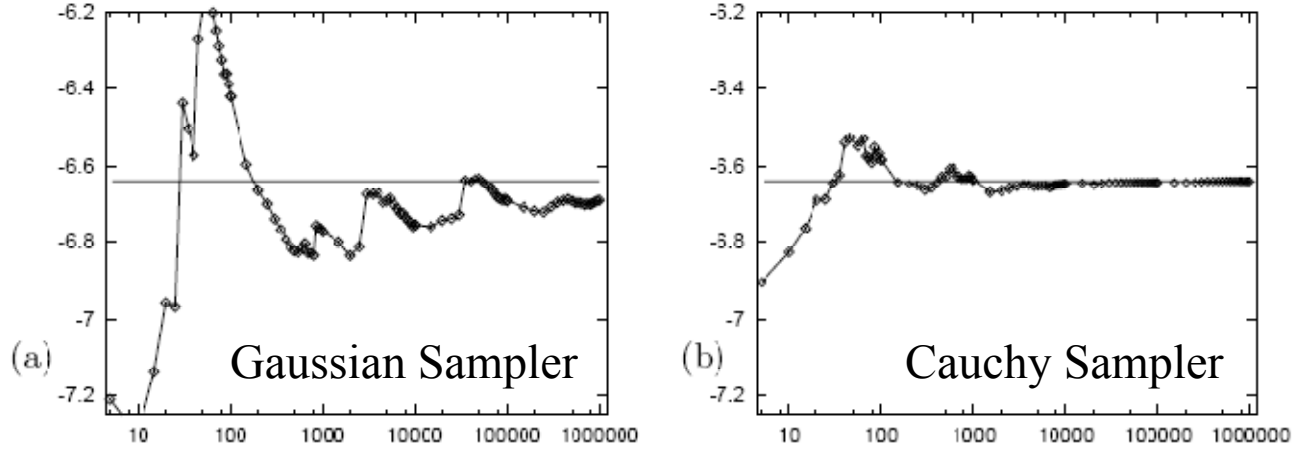
$$
\begin{aligned}
E[f] &= \int f(z)\,p(z)\,dz \\
&= \int f(z)\,\frac{p(z)}{q(z)}\,q(z)\,dz \\
&\cong \frac{1}{L}\sum_{i=1}^{L}\frac{p(z^{(l)})}{q(z^{(l)})}\,f(z^{(l)})
\end{aligned}
$$



Sample from q weighted by p.

Importance Sampling (2/3)

- Depends on the choice of Q.



- In high dimensions,
  - We need to obtain samples that lies in the typical set of P, and this may be take a long time unless Q is a good approximation to P.
  - The variation of weights q/p is large (order of $\exp(N^{1/2})$, unless Q is a near-perfect approximation to P.

Importance Sampling (3/3)

- Sampling-Importance-Resampling (SIR)
  - It is difficult to set *k* in rejection sampling

1. *Sampling* from *q*.
2. Set weight on each sample as in *importance sampling*.
3. *Resample* from the samples.

  - Final samples approximate *p* as the sample size increases.

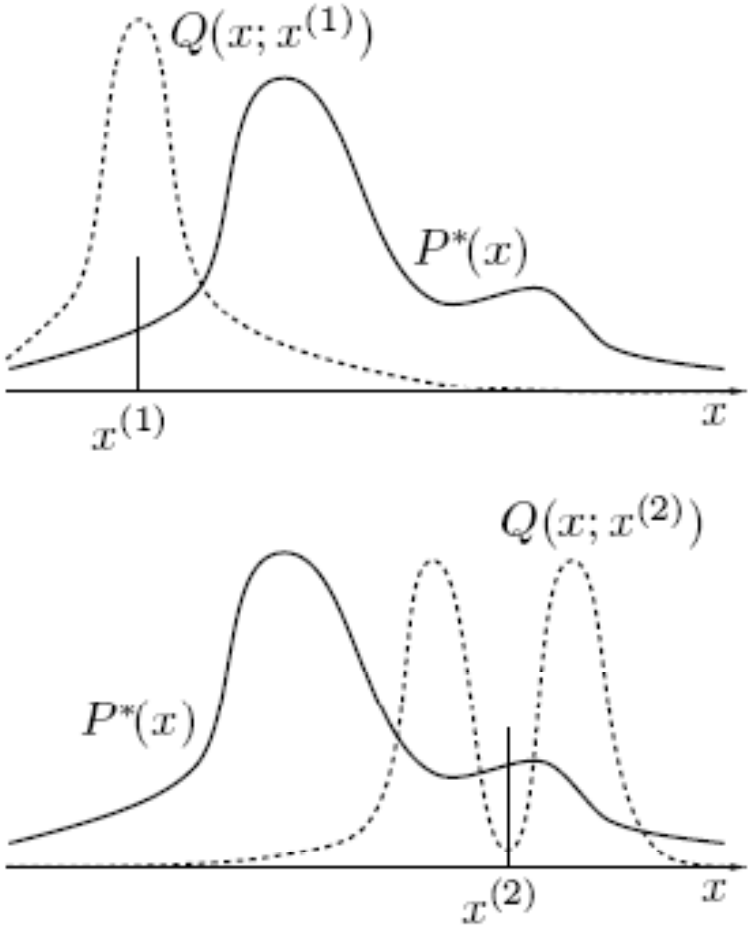
Markov Chain Monte Carlo Method (1/5)

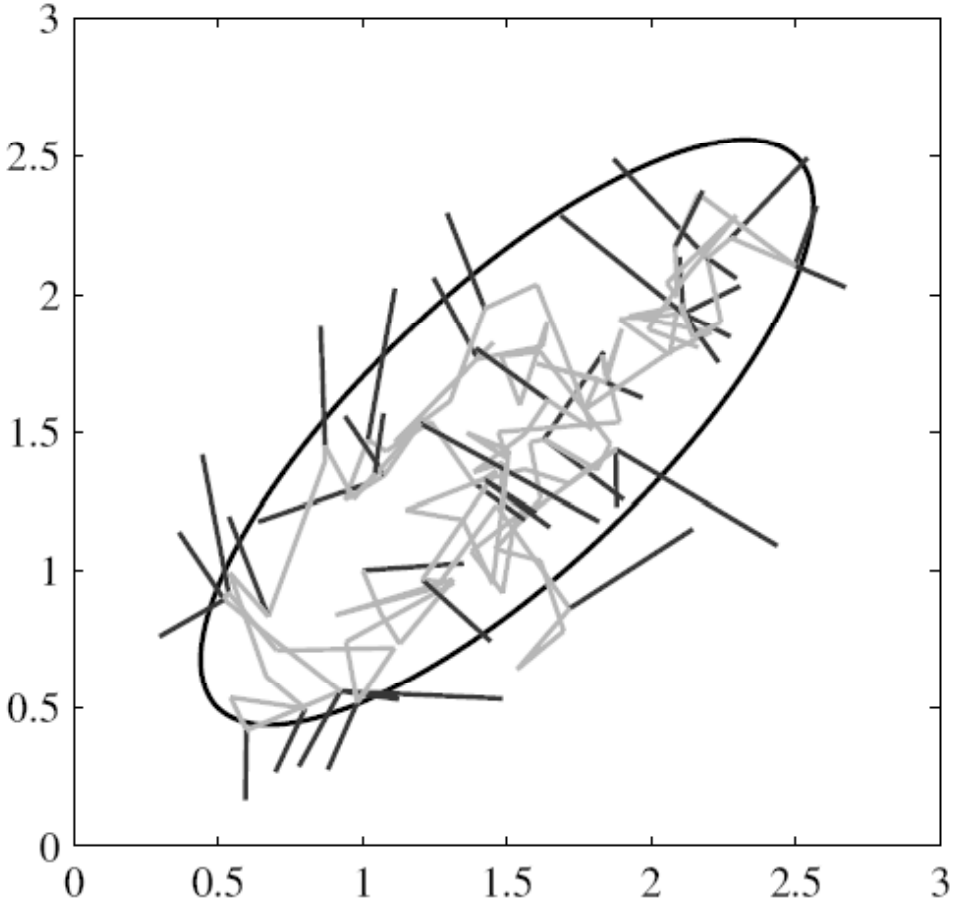
- Allows sampling from a large class of distributions.
- Scales well with the dimensionality of the sample space.
- Basic Metropolis Algorithm (a simple MCMC)
  - Maintain a record of state $z^{(t)}$: forms a Markov chain.
  - Next state is sampled from $q(z|z^{(t)})$ (q must be symmetric).
  - Candidate state from q is accepted with probability of

$$A(z, z^{(t)}) = \min(1, \frac{\tilde{p}(z)}{\tilde{p}(z^{(t)})})$$

  - If rejected, current state is added to the record and becomes the next state.
  - Distribution of z tends to p in the infinity.
  - The original sequence is autocorrelated and get every M^{th} sample to get independent samples. For large M, the retained samples will be independent.

Markov Chain Monte Carlo Method (2/5)





Markov Chain Monte Carlo Method (3/5)

- Under what conditions will a Markov chain converge to the desired distribution?
  - If we set up a Markov chain such that the desired distribution is invariant with respect to the chain.

    Transition probabilities do not change over time.

    - For a homogeneous Markov chain with transition probabilities $T(z', z)$, a distribution $p^*(z)$ is invariant with respect to the chain if,

      $$\tilde{p}(\mathbf{z}) = \sum_{\mathbf{z}'} T(\mathbf{z}', \mathbf{z}) \tilde{p}(\mathbf{z}')$$

      The marginal distribution for each state does not change after transitions.
    - A sufficient (not necessary) condition for ensuring the required distribution p is invariant is to choose the transition probabilities to satisfy the property of detailed balance: reversible Markov chain.

      $$\tilde{p}(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') = \tilde{p}(\mathbf{z}') T(\mathbf{z}', \mathbf{z})$$

      $$\sum_{\mathbf{z}'} \tilde{p}(\mathbf{z}') T(\mathbf{z}', \mathbf{z}) = \sum_{\mathbf{z}'} \tilde{p}(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') = \tilde{p}(\mathbf{z}) \sum_{\mathbf{z}'} p(\mathbf{z}' \mid \mathbf{z}) = \tilde{p}(\mathbf{z})$$

  - For ergodic Markov chain, the distribution $p(z^{(t)})$ converges to the required invariant distribution irrespective to initial distributions $p(z^{(0)})$.

Markov Chain Monte Carlo Method

- Metropolis-Hastings algorithm
  - Generalization of Metropolis algorithm
  - Q can be non-symmetric
  - Modified accept probability:

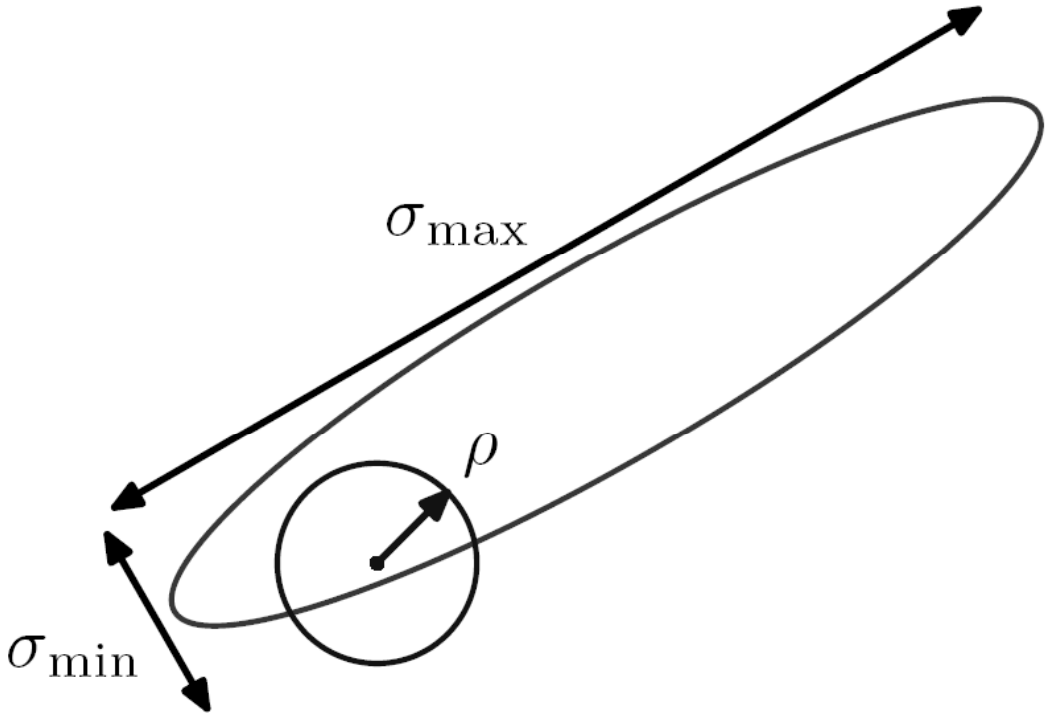
$$A_k(\mathbf{z}^*, \mathbf{z}^{(t)}) = \min\left(1, \frac{\tilde{p}(\mathbf{z}^*) q_k(\mathbf{z}^{(t)} \mid \mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(t)}) q_k(\mathbf{z}^* \mid \mathbf{z}^{(t)})}\right)$$

  - Proof of detailed balance

$$\begin{aligned}
\tilde{p}(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') &= \tilde{p}(\mathbf{z}) q_k(\mathbf{z}' \mid \mathbf{z}) A_k(\mathbf{z}', \mathbf{z}) \\
&= \min\left(\tilde{p}(\mathbf{z}) q_k(\mathbf{z}' \mid \mathbf{z}), \tilde{p}(\mathbf{z}') q_k(\mathbf{z} \mid \mathbf{z}')\right) \\
&= \min\left(\tilde{p}(\mathbf{z}') q_k(\mathbf{z} \mid \mathbf{z}'), \tilde{p}(\mathbf{z}) q_k(\mathbf{z}' \mid \mathbf{z})\right) \\
&= \tilde{p}(\mathbf{z}') q_k(\mathbf{z} \mid \mathbf{z}') \min\left(1, \frac{\tilde{p}(\mathbf{z}) q_k(\mathbf{z}' \mid \mathbf{z})}{\tilde{p}(\mathbf{z}') q_k(\mathbf{z} \mid \mathbf{z}')}\right) \\
&= \tilde{p}(\mathbf{z}') q_k(\mathbf{z} \mid \mathbf{z}') A_k(\mathbf{z}, \mathbf{z}') = \tilde{p}(\mathbf{z}') T(\mathbf{z}', \mathbf{z})
\end{aligned}$$

Markov Chain Monte Carlo Method (5/5)

- The common choice for q is Gaussian
  - Trade-off between step size and convergence time
  - Small variance – high acceptance ratio – slow random walk.



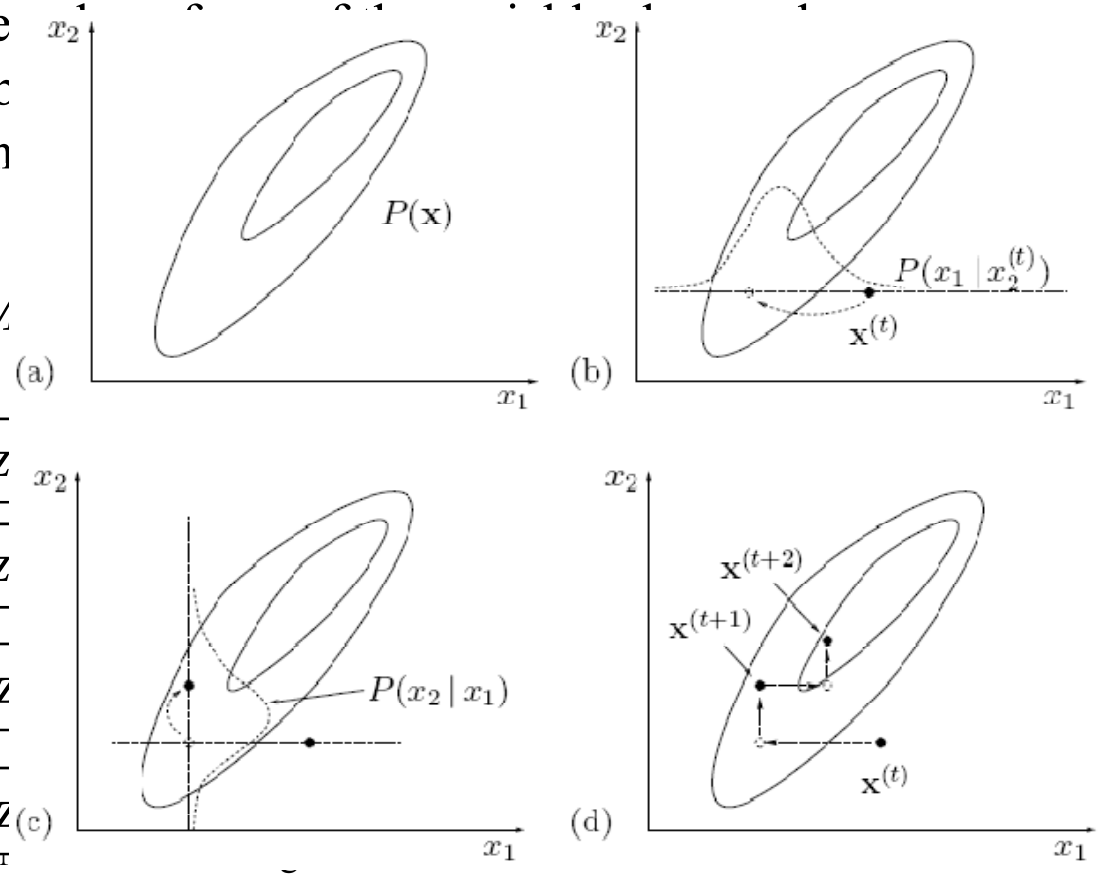
Gibbs Sampling (1/3)

- Simple and widely applicable MCMC algorithm.
- Special case of Metropolis-Hastings algorithm.
  - Each step replaces the
    drawn from the distrib
    values of the remainin
- The procedure
  1. Initialize z_i, i=1, …, M
  2. For t=1,…,T
     - Sample $z_1^{t+1} \sim p(z$

$z_2^{t+1} \sim p(z$

$z_i^{t+1} \sim p(z$

$z_M^{t+1} \sim p(z$



Gibbs Sampling (2/3)

1. $p(z)=p(z_{\setminus i})p(z_i|z_{\setminus i})$ is an invariant of each of Gibbs sampling steps and whole Markov chain.
   - ♦ At each step, the marginal distribution $p(z_{\setminus i})$ is invariant.
   - ♦ Each step correctly samples from the conditional distribution $p(z_i|z_{\setminus i})$.
2. The Markov chain defined is ergodic.
   - ♦ The conditional distribution must be non-zero.

☞ The Gibbs sampling correctly samples from p.

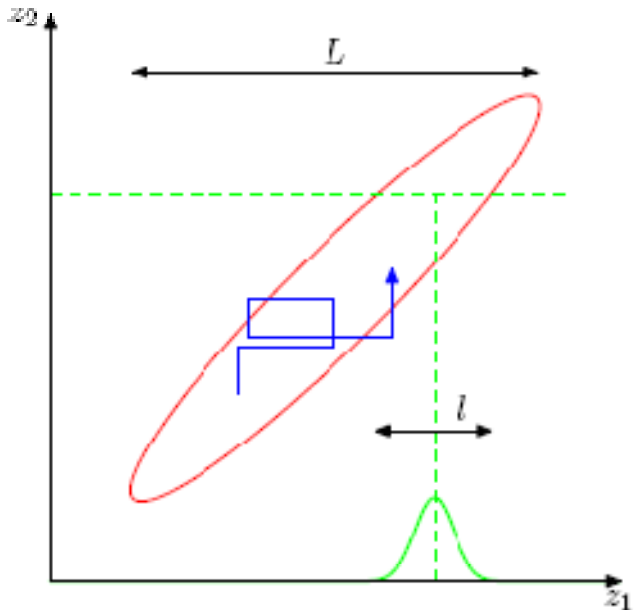
● Gibbs sampling as an instance of Metropolis-Hastings algorithm.

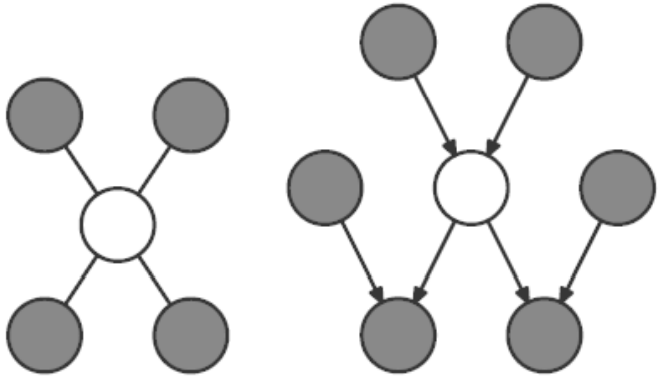
- ♦ A step involving z_k in which $z_{\setminus k}$ remain fixed.
- ♦ Transition probability $q_k(\mathbf{z}^*|\mathbf{z}) = p(z^*_k|z_{\setminus k})$, $\boxed{\mathbf{z}^*_{\setminus k} = \mathbf{z}_{\setminus k}}$

$$\boxed{A(\mathbf{z}^*, \mathbf{z}) = \frac{p(\mathbf{z}^*)q_k(\mathbf{z}|\mathbf{z}^*)}{p(\mathbf{z})q_k(\mathbf{z}^*|\mathbf{z})} = \frac{p(z_k^*|\mathbf{z}_{\setminus k}^*)p(\mathbf{z}_{\setminus k}^*)p(z_k|\mathbf{z}_{\setminus k}^*)}{p(z_k|\mathbf{z}_{\setminus k})p(\mathbf{z}_{\setminus k})p(z_k^*|\mathbf{z}_{\setminus k})} = 1}$$

Gibbs Sampling (3/3)

- Random walk behavior
  - The number of steps needed to obtain independent samples is of order $(L/l)^2$.
  - Over-relaxation
- The practical applicability depends on the ease of sampling from the conditional distributions





References

- Pattern Recognition and Machine Learning
  - Ch. 11 Sampling Methods
  - C. Bishop, Springer, 2006.
  - E-Book available from library.

- Information Theory, Inference, and Learning Algorithms
  - Ch. 29 Monte Carlo Methods
  - D. MacKay, Cambridge University Press, 2003.
  - Pdf file available from http://www.inference.phy.cam.ac.uk/mackay/itila/.

Evolutionary MCMC & Estimation of Distribution Algorithms (EDA)

Course on Probabilistic Graphical Models

(Artificial Neural Networks,
Studies in Artificial Intelligence and Cognitive Process)

Evolutionary Computation

- Basic concept
  - ♦ Use of Darwinian-like evolutionary processes to solve difficult computational problems.

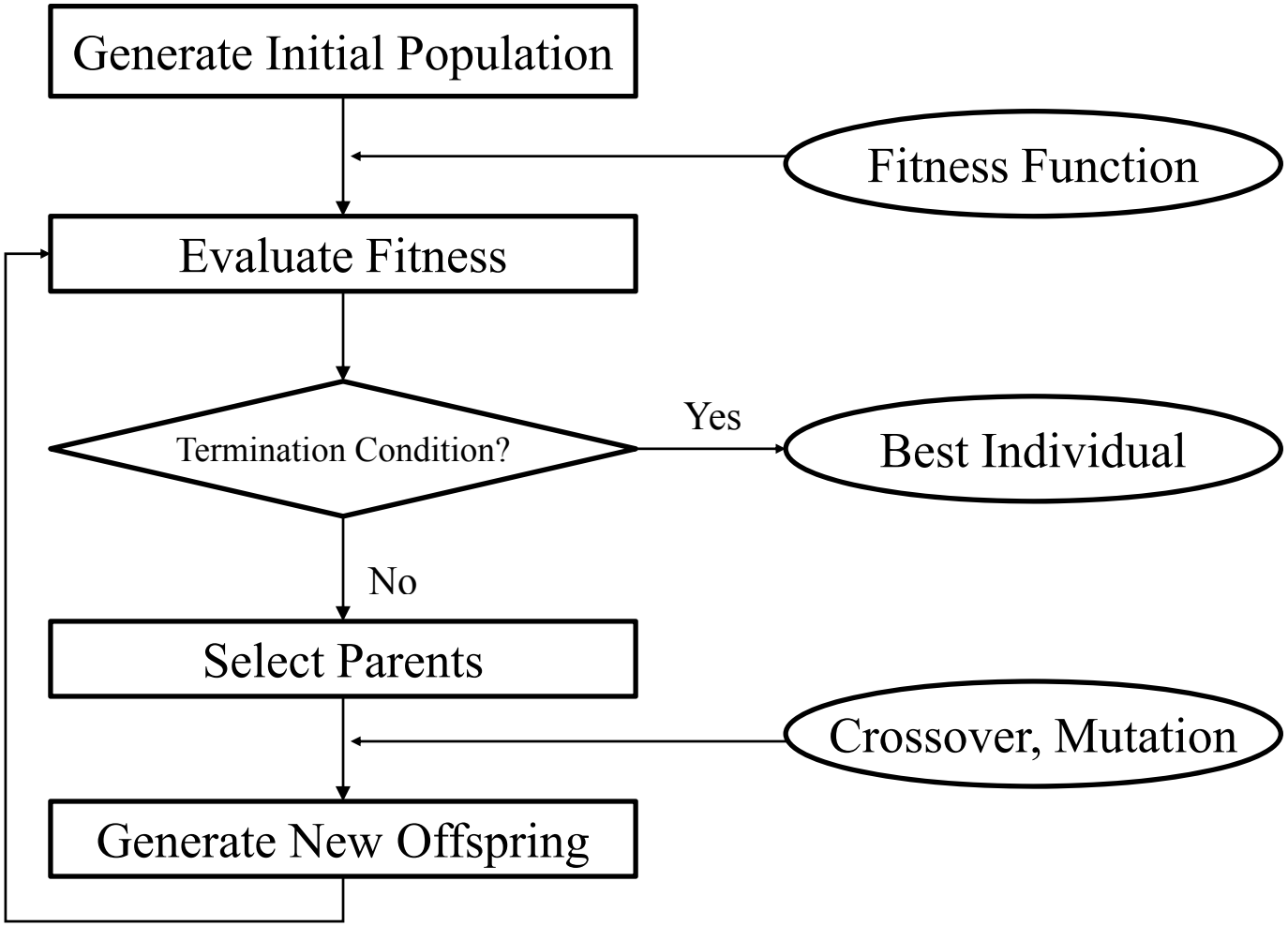
  ⇒Hence the name, "Evolutionary Computation"

  - ♦ Biological evolutionary process
    - ■ Population of parents produce descendants
    - ■ Descendants are changed from their parents
    - ■ Selective survival of descendants for next generation

EVOLUTION		PROBLEM SOLVING
Individual	⟷	Candidate Solution
Fitness	⟷	Quality
Environment	⟷	Problem

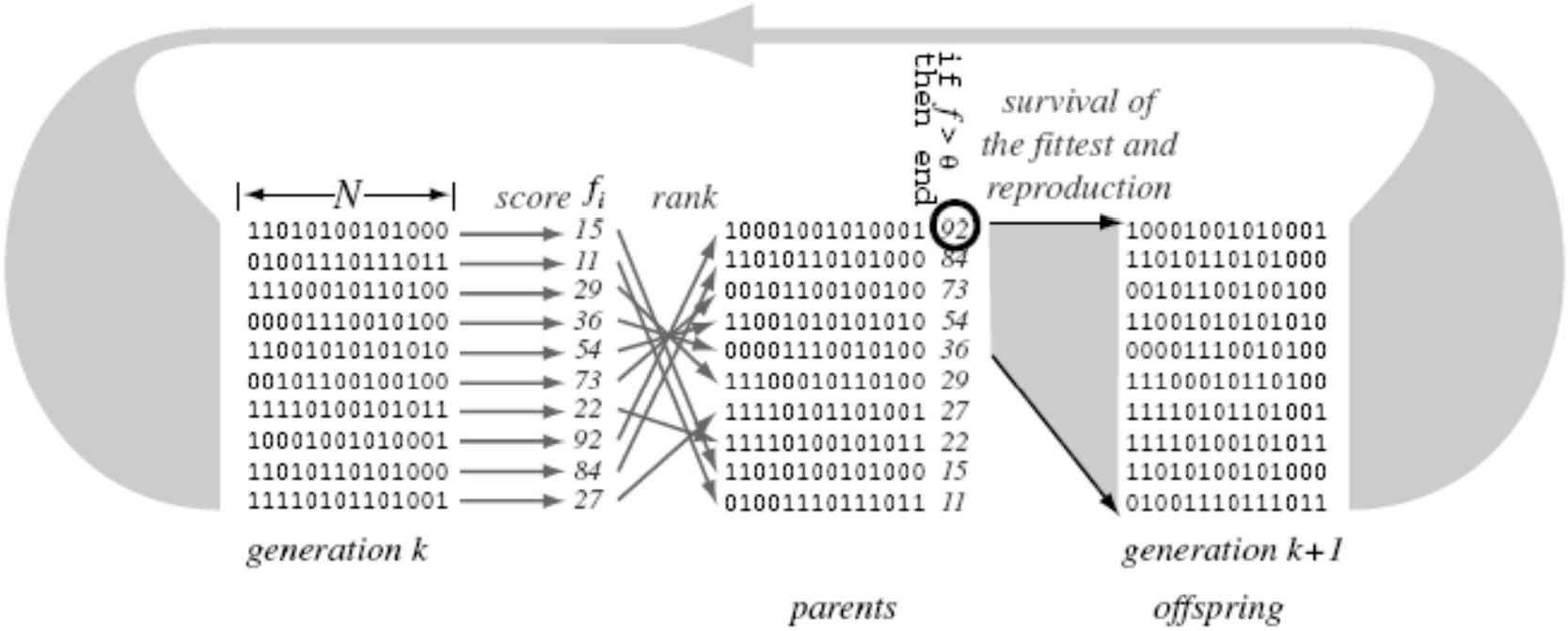
Evolutionary Computation

- General Framework



Evolutionary Computation

- Typical run



Paradigms in Evolutionary Computation

- Evolutionary Programming (EP)
  - [L. Fogel et al., 1966]
  - FSMs, mutation only, tournament selection
- Evolution Strategy (ES)
  - [I. Rechenberg, 1973]
  - Real values, mainly mutation, ranking selection
- Genetic Algorithm (GA)
  - [J. Holland, 1975]
  - Bitstrings, mainly crossover, proportionate selection
- Genetic Programming (GP)
  - [J. Koza, 1992]
  - Trees, mainly crossover, proportionate selection

Evolutionary Computation

- Representations
  - Candidate solutions (individuals) exist in phenotype space.
  - They are encoded in chromosomes, which exist in genotype space.
  - Encoding : phenotype → genotype (not necessarily one to one)
  - Decoding : genotype → phenotype (must be one to one)
  - **In order to find the global optimum, every feasible solution must be represented in genotype space.**
- Population
  - Usually has a fixed size and is a *multiset* of genotypes
  - Some sophisticated EAs also assert a spatial structure on the population e.g., a grid.
  - Diversity of a population refers to the number of different fitnesses / phenotypes / genotypes present (note not the same thing).

Evolutionary Computation

- Fitness function
  - Represents the requirements that the population should adapt to
  - a.k.a. *quality* function or *objective* function
  - Assigns a single real-valued fitness to each phenotype which forms the basis for selection
    - So the more discrimination (different values) the better
  - Typically we talk about fitness being maximised
    - Some problems may be best posed as minimisation problems, but conversion is trivial.

Evolutionary Computation

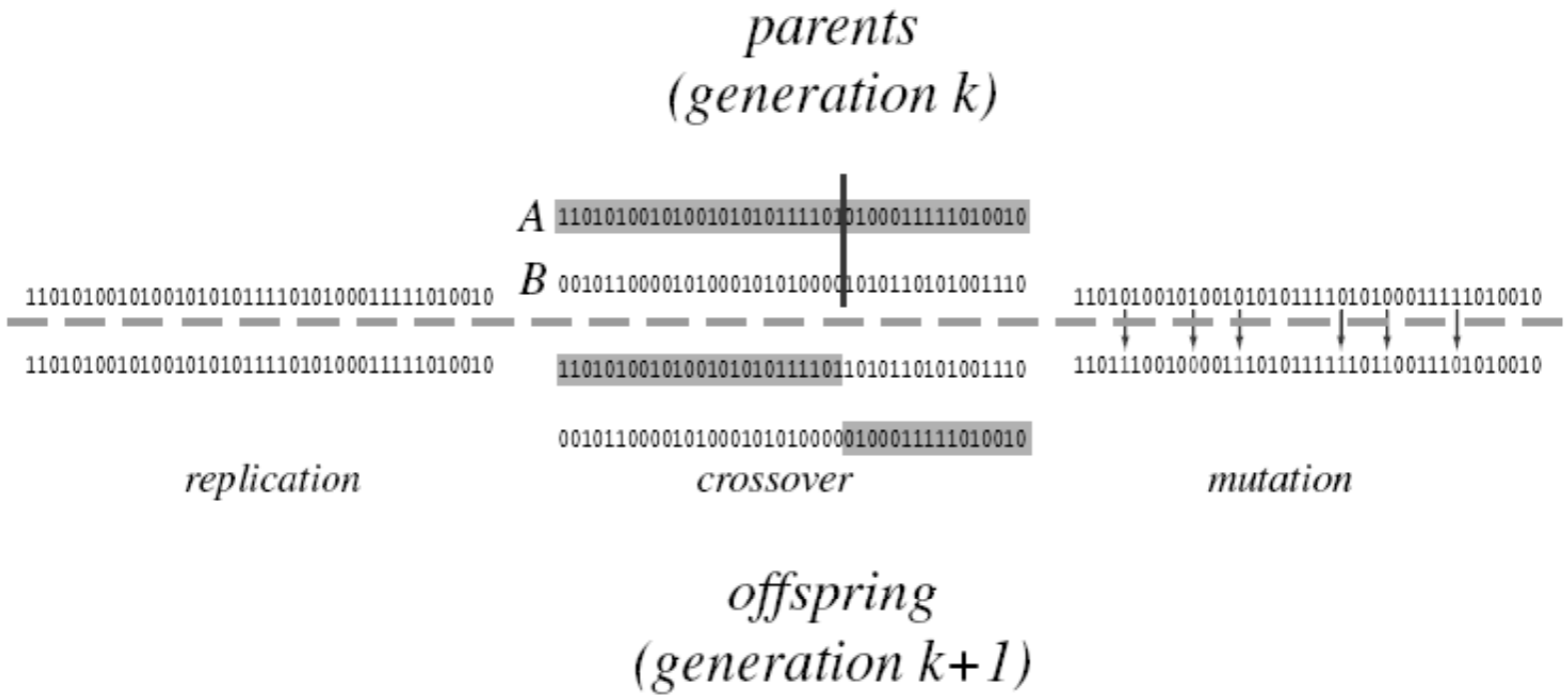
- Parent selection mechanism
  - Assigns probabilities for an individual to be selected as parents.
  - Selection probabilities are relative to current population.
    - Different probabilities can be assigned to the same individuals.
  - Usually depends on the individual's fitness and probabilistic.
    - High quality solutions more likely to become parents than low quality ones
  - Selection pressure – the degree of correlation between the individual's fitness and its selection probability.
    - High selection pressure results in reducing search scope.
    - Even worst in current population usually has non-zero probability of becoming a parent
  - This *stochastic* nature can aid escape from local optima.

Evolutionary Computation

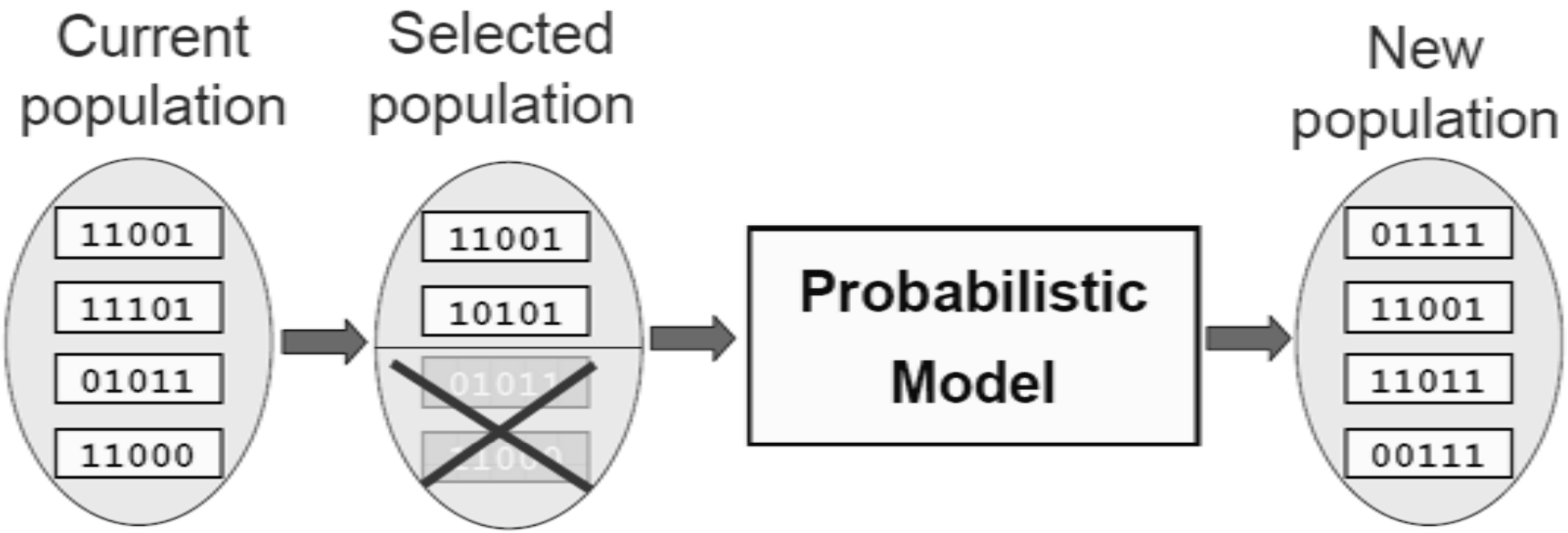
- Crossover (Recombination)
  - Mix information from parents into offspring in stochastic way.
  - Hope is that some are better by combining elements of genotypes that lead to good traits.
- Mutation
  - It is applied to one genotype and delivers a (slightly) modified mutant, the child or offspring of it.
  - New elements can be introduced to the population.
- The role of crossover/mutation is different in various subtypes.

Evolutionary Computation

- Examples of crossover/mutation



Estimation of Distribution Algorithms (EDAs)



- EDAs replace crossover and mutation with building and sampling probabilistic model.
- Junction between evolutionary computation and machine learning (graphical models).

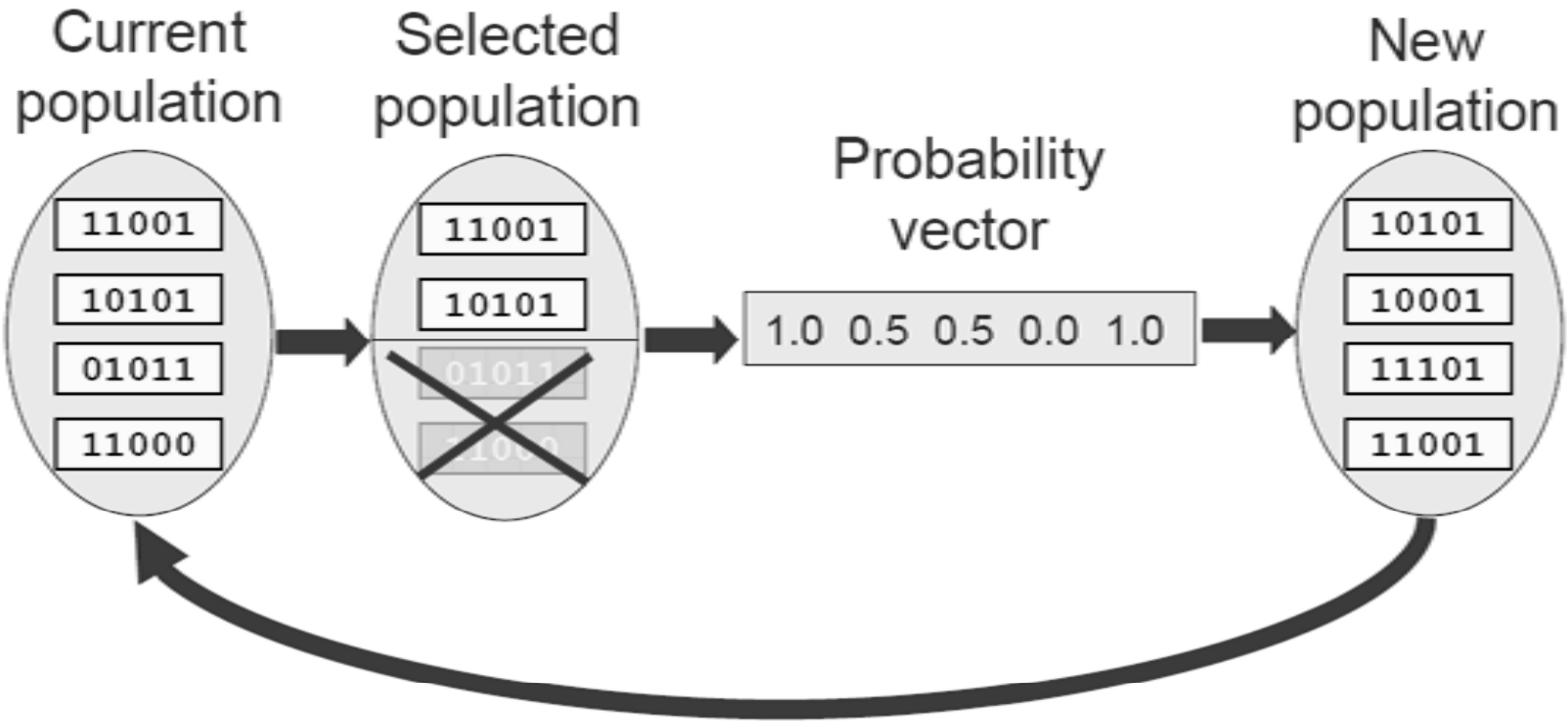
Estimation of Distribution Algorithms (EDAs)

- Comparison with conventional genetic algorithms
  - Both perform similar task
    - "Generate new solutions using probability distribution based on selected solutions."
  - Genetic algorithms
    - Probability distribution of selected solutions is built and sampled implicitly by crossover/mutation.
  - EDAs
    - Explicit probabilistic model of selected candidate solutions is built and sampled.

Estimation of Distribution Algorithms (EDAs)

- Objective: finding appropriate model that represent the distribution of good candidate solutions.
- EDAs are classified by the probabilistic models they use.
  - Univariate models
    - Assumes no dependency between variables.
    - PBIL (Baluja, 94), UMDA (Mühlenbein & Paass, 96), cGA (Harik et al., 97)
  - Bivariate models
    - Assumes binary dependency between pair of variables.
    - MIMIC (deBonet et al, 97), COMIT (Baluja and Davis, 97), BMDA (Pelikan and Muhlenbein, 99)
  - Multivariate models
    - Assumes most general dependency between groups of variables.
    - ECGA (Harik, 99), FDA (Mühlenbein &Mahnig, 99), BOA (Pelikan et al, 99)

EDAs with Univariate Models

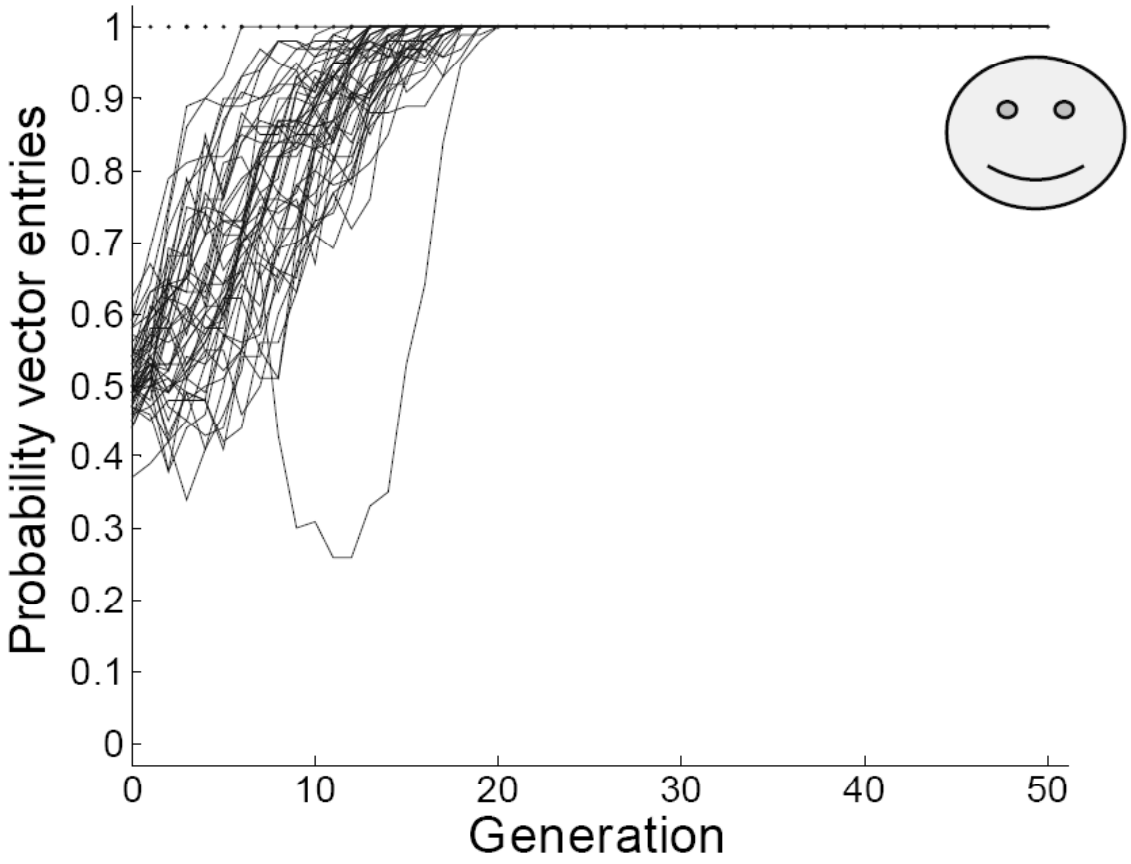


- Bits that perform get more copies.
- The context of each bit is ignored.

EDAs with Univariate Models

- Example 1: OneMax function.

$$f(\mathbf{X}) = \sum_{i=1} X_i$$

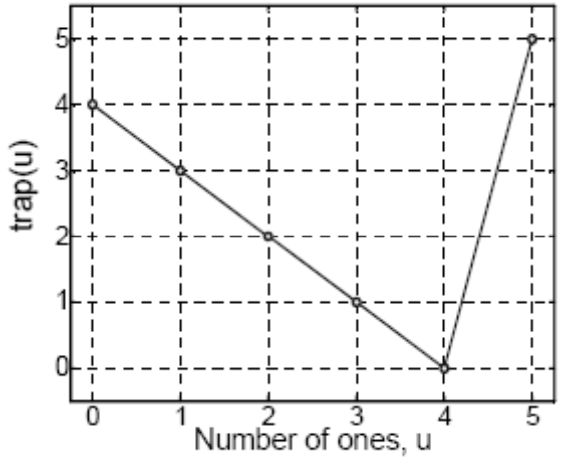


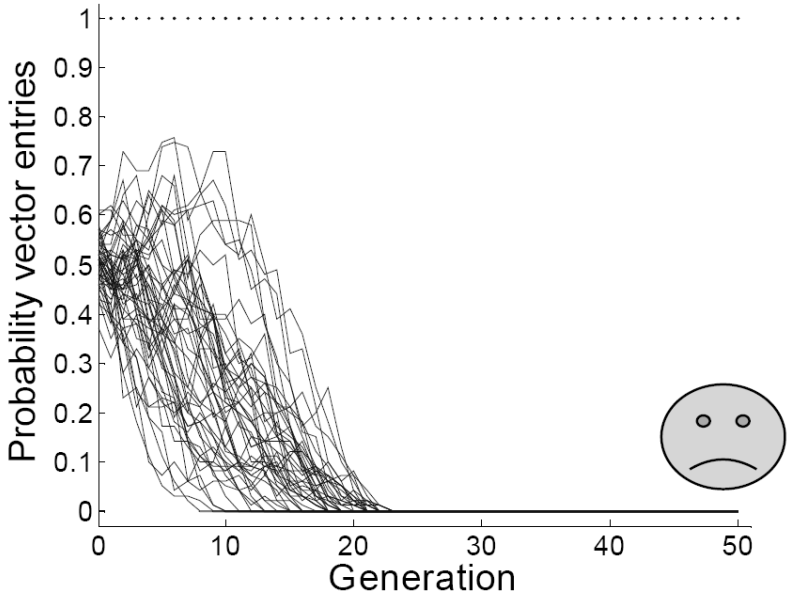
EDAs with Univariate Models

- Example 2: Trap5 function.
  - Input string is partitioned into disjoint groups of 5 bits.

$$Trap5(ones) = \begin{cases} 5 & \text{if } ones = 5 \\ 4 - ones & \text{otherwise} \end{cases}$$

1 1 0 1 0 0 1 1 0 0 ⇒ 1 + 2 = 3





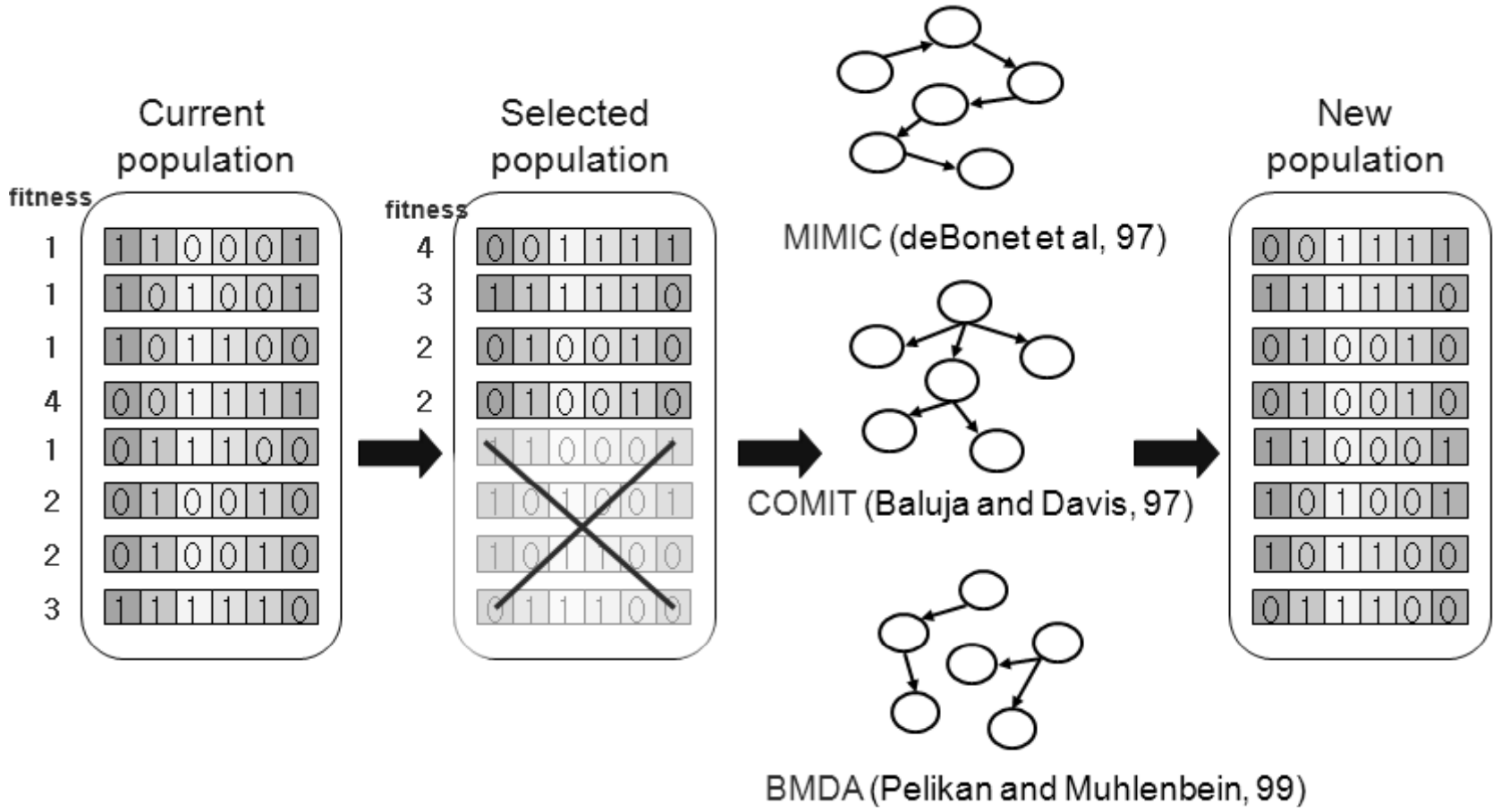
EDAs with Univariate Models

- Why failure in Trap5 function?
  - Trap5(0****)=2
  - Trap5(1****)=1.375
- Single bits are misleading.

$\Rightarrow$ Consider a group of related bits as one.

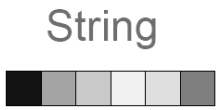
- Bivariate models
- Multivariate models
- Requires measure for model comparison.

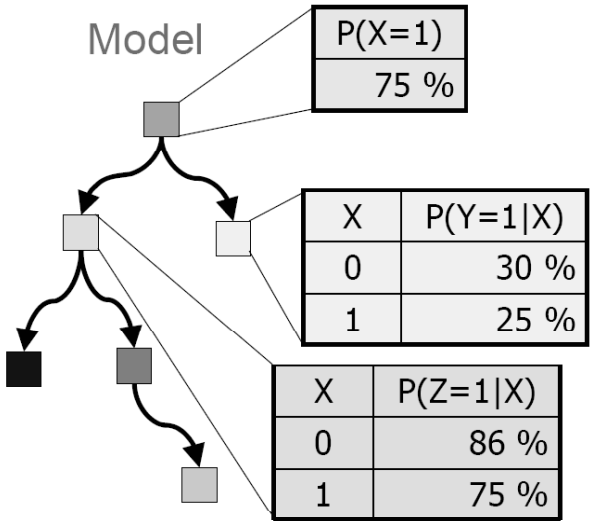
EDAs with Bivariate Models



EDAs with Bivariate Models

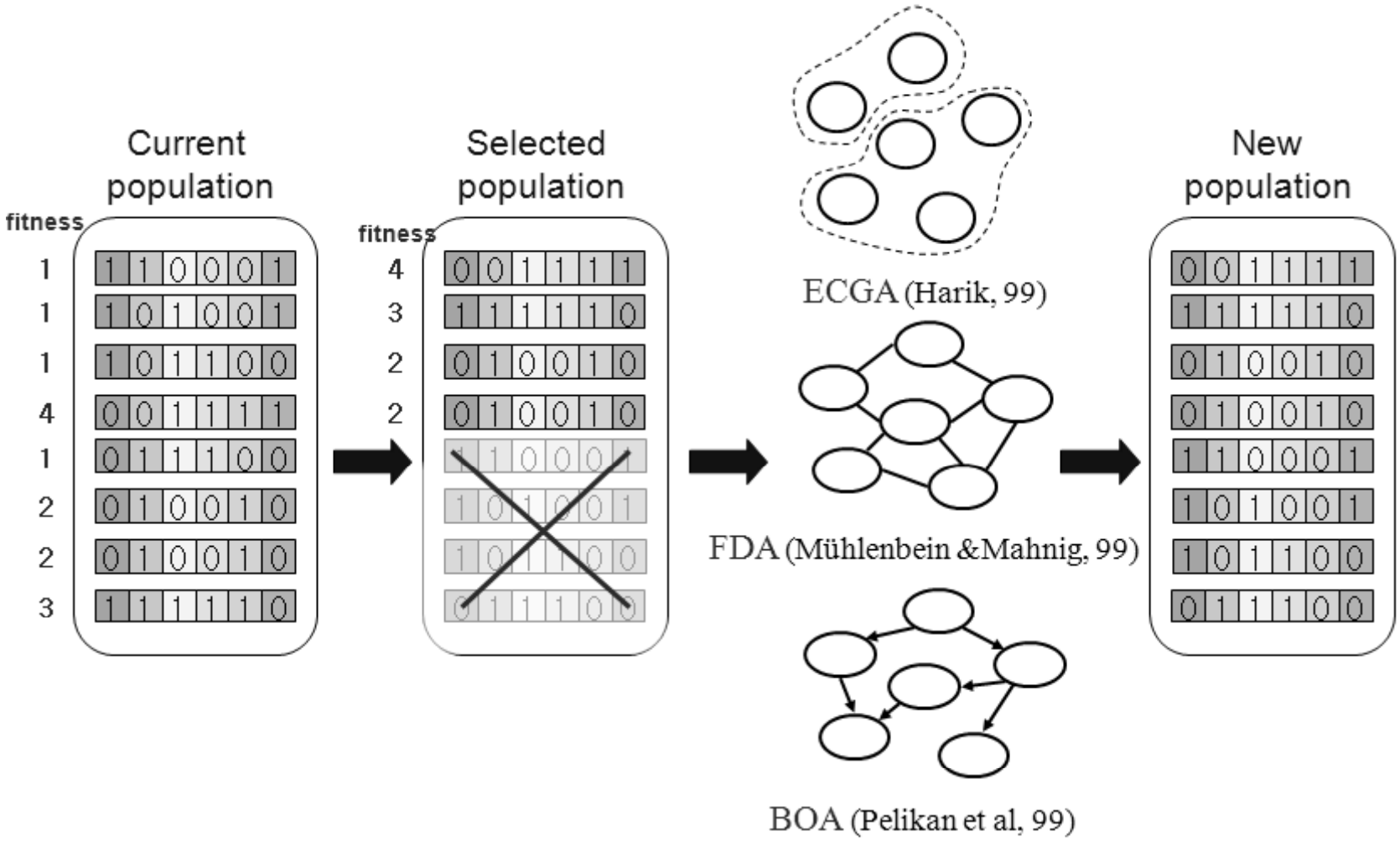
- Tree model in COMIT.





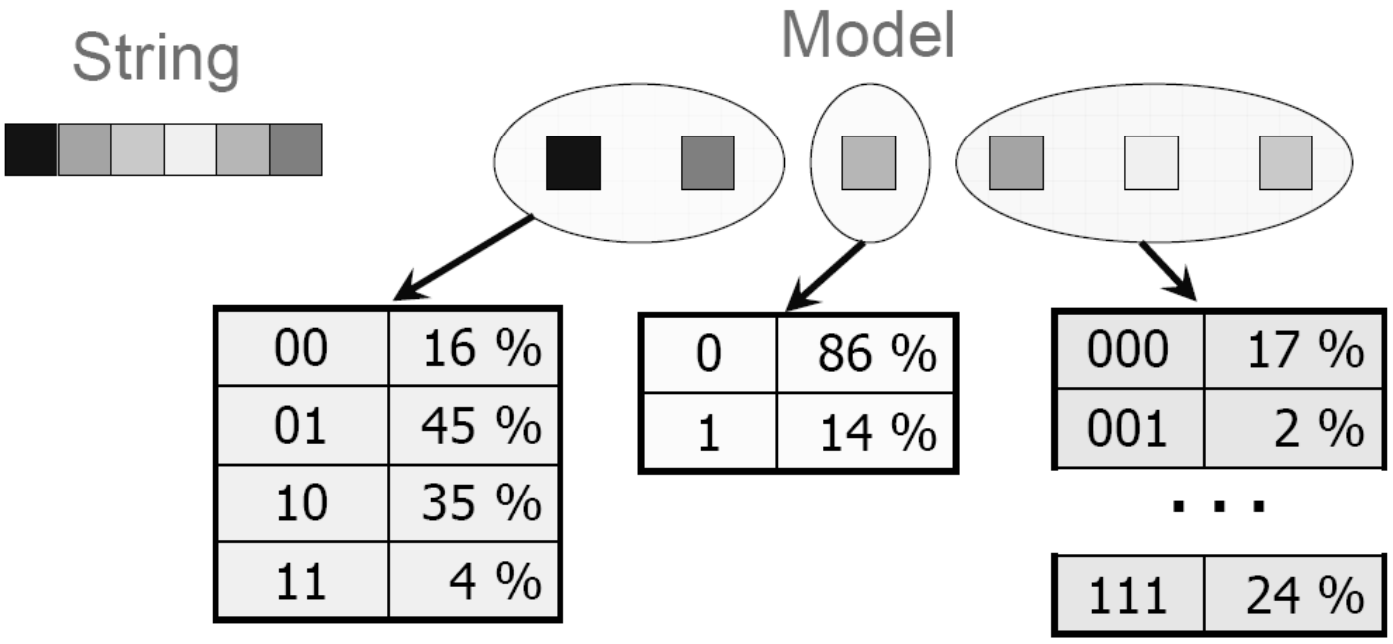
- How to learn a tree model?
  - Find tree that maximizes mutual information between connected nodes. $I(X_i, X_j) = \sum_{a,b} P(X_i = a, X_j = b) \log \frac{P(X_i = a, X_j = b)}{P(X_i = a) P(X_j = b)}$
  - Start with tree with no edges $\Rightarrow$ Add edge with large mutual information (greedy search).

EDAs with Multivariate Models



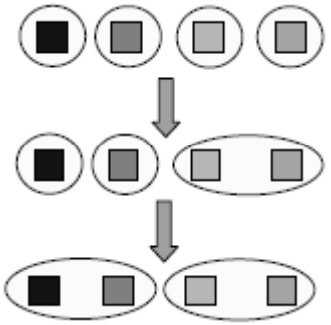
EDAs with Multivariate Models

- Probability Model in ECGA (Extended Compact GA).
  - ♦ Disjoint groups of variables



EDAs with Multivariate Models

- Learning Model in ECGA.
  - Start with each bit in a separate group.
  - Each iteration merges two groups for best improvement in minimum description length.



$$MDL(M,D) = D_{Model} + D_{Data}$$

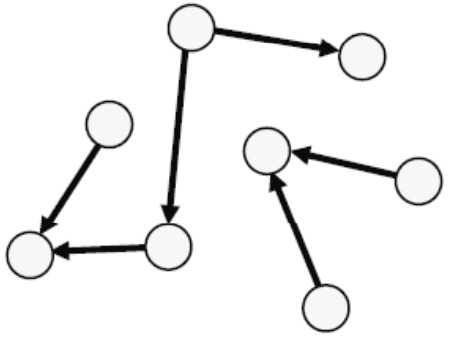
$$D_{Model} = \sum_{g \in G} 2^{|g|-1} \log N$$

$$D_{Data} = -N \sum_{x} p(X) \log p(X)$$

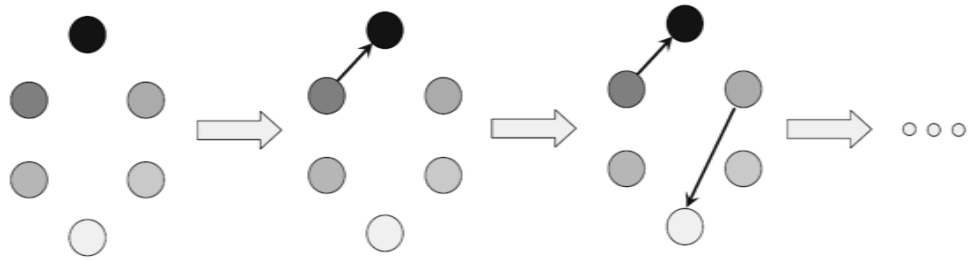
- Sampling in ECGA
  - Sample group of bits at a time.
  - Based on the observed frequencies.

EDAs with Multivariate Models

- Bayesian network model in BOA (Bayesian Optimization Algorithm).
  - ♦ Directed acyclic graph.



- Learning BN model.
  - ♦ Start with empty network.
  - ♦ Add edge / Delete edge / Reverse edge that improves the scoring metric the most until no more improvement is possible.



EDAs with Multivariate Models

- Scoring metric for Bayesian network.
  - Bayesian metric: Bayesian-Dirichlet with likelihood equivalence

$$BD(B) = p(B)\prod_{i=1}^{n}\prod_{\pi_i}\frac{\Gamma(m'(\pi_i))}{\Gamma(m'(\pi_i)+m(\pi_i))}\prod_{x_i}\frac{\Gamma(m'(x_i,\pi_i)+m(x_i,\pi_i))}{\Gamma(m'(x_i,\pi_i))}$$

  - Minimum description length metric: Bayesian information criterion (BIC)

$$BIC(B) = \sum_{i=1}^{n}\left(-H(X_i \mid \Pi_i)N - 2^{|\Pi_i|}\frac{\log_2 N}{2}\right)$$

References

- Introduction to Evolutionary Computing
  - A. E. Eiben and J. E Smith, Springer, 2003.
- Estimation of Distribution Algorithms
  - P. Larranaga and J. A. Lozano (Eds.), Kluwer, 2002.
- Further information
  - Conferences
    - IEEE Congress on Evolutionary Computation (CEC)
    - Genetic and Evolutionary Computation Conference (GECCO)
    - Parallel Problem Solving from Nature (PPSN)
    - Foundation of Genetic Algorithms (FOGA)
  - Journals
    - IEEE Transactions on Evolutionary Computation
    - Evolutionary Computation
    - Genetic Programming and Evolvable Machines