Probabilistic Algorithms

Paul Cotofrei

information management institute

PA 2016

Outline

Physical background

- **Annealing** - family of techniques for creating metals with desirable mechanical properties.
- Based on heat treatment
  - heating the tool to a temperature below the melting temperature, but high enough so that the crystalline grains change shape
  - then slowly cooled - softer, ductile material
  - or rapidly cooled - harder surface material
- Know-how: the scheduling temperatures and times
- Simulated annealing - introduced by physicist exposed to computer simulation, but ignored by mathematicians
- Hard optimization problems: design of computer circuits, optimally pack machine-based instructions for high-performed compilers, shortest circuit for Travel Salesman Problem

SA for physical systems

- Simulates the cooling process of a physical system
  - start at a very high temperature
  - temperature decreases slowly by steps
  - transition from a high-energy unordered regime to a low energy (partially) ordered regime
  - optimization process stops when the system is frozen in a (quasi) optimum state at a low temperature
- Sequence of moves at each temperature step
- How to choose the transition probability
- Canonical equilibrium distribution of classical systems
  - If Γ is the set of states, $\sigma \in \Gamma$ and $H(\sigma)$ is the system energy in state σ, then

$$P_{eq}(\sigma) = \frac{1}{Z} \exp\left(-\frac{H(\sigma)}{k_B T}\right)$$

where Z is a normalizing constant, T the temperature and $k_B > 0$ a physical constant

Transition probability

- At equilibrium,

$$\frac{P(\sigma \to \tau)}{P(\tau \to \sigma)} = \exp\left(-\frac{\Delta H}{k_B T}\right)$$

- Different choices
  - Metropolis criterion

$$P(\sigma \to \tau) = \begin{cases} \exp\left(-\frac{\Delta H}{k_B T}\right) & \text{if } \Delta H > 0 \\ 1 & \text{if not} \end{cases}$$

    - If the new state is better or as good as the actual state, it is always accepted
    - If is worse, there is a positive probability to be accepted
  - Heat bath condition

$$P(\sigma \to \tau) = \frac{1}{1 + \exp\left(\frac{\Delta H}{k_B T}\right)}$$

    - Any new state is accepted with a positive probability
- Metropolis criterion - faster dynamics ($p_{Metropolis} > p_{heat\ bath}$)

SA algorithm

Step 0 [Initialization] Set initial temperature $T = T_0$ and initial configuration $\hat{\mathbf{d}}_0 = \mathbf{d}_{curr}$; calculate $L(\mathbf{d}_{curr})$.

Step 1 [Candidate value] Relative to $\mathbf{d}_{curr}$, randomly generate new configuration $\mathbf{d}_{new}$ and calculate $L(\mathbf{d}_{new})$.

Step 2 [Accept/reject transition] Let $\Delta L = L(\mathbf{d}_{new}) - L(\mathbf{d}_{curr})$. If $\Delta L < 0$ accept θ_{new}. If not, generate δ uniform on $(0, 1)$ and accept θ_{new} if $\delta < \exp(-\frac{\Delta L}{T})$; otherwise keep θ_{curr}.

Step 3 [Iterate at fixed temperature] Repeat steps 1 and 2 using the same T until an equilibrium is reached.

Step 4 [Decrease temperature] Decrease T according to the annealing schedule and return to Step 1. Continue till stop conditions are filled.

Remark: By default, the constant k_B is set to 1 in all SA related algorithms

Generate new configuration

- Perturbation method: $\mathbf{d}_{new} = \mathbf{d}_{curr} + \delta_{pert}$
- Perturbations δ_{pert} generated by
  1. a p-dimensional normal variable
  2. a spherically uniform distributed variable
  3. a multivariate Cauchy distributed variable
- Small perturbations: change only one component at time
  - moves derived from pattern search approach
- $\mathbf{d}_{new}$ close to $\mathbf{d}_{curr}$ - use small moves to generate the new configuration (problem dependent)
  1. Swap
  2. Translation
  3. Inversion

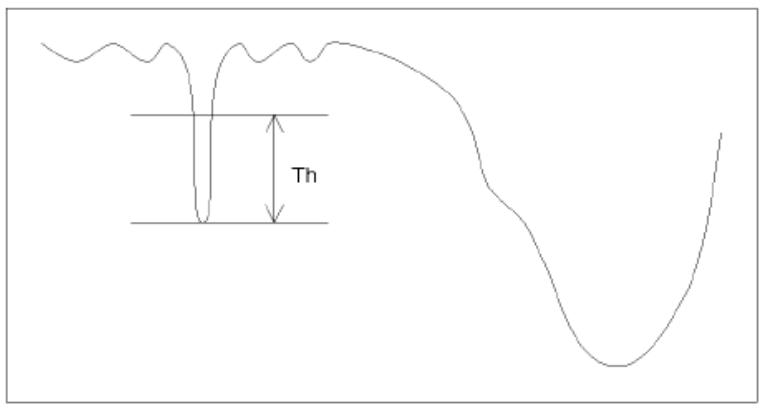
Outline

Threshold accepting (TA)

- Transition probability

$$P(\mathbf{d}_{curr} \rightarrow \mathbf{d}_{new}) = \begin{cases} 1 & \text{if } \Delta L < Th \\ 0 & \text{otherwise} \end{cases}$$

  - The move is accepted if the new solution is either better or equally good ($\Delta L \leq 0$) or is not worse than a given threshold ($0 < \Delta L < Th$).
- *Th* - threshold lowered gradually to zero (the same role as the temperature for SA)
- TA may be trapped in a "golf hole" (states with a much lower energy than all of their neighbors)
- TA is much faster than SA (no need to generate δ uniformly on (0,1)); considered as a "first approximation" of SA



The great deluge algorithm (GDA)

- Random walk through a subset $\Gamma_T \subseteq \Gamma$, where each state in Γ_T has an energy smaller than a certain level $\mathcal{T}$
  - "great deluge" - can walk only on dry configurations
- Transition probability

$$P(\mathbf{d}_{curr} \rightarrow \mathbf{d}_{new}) = \begin{cases} 1 & \text{if } L(\mathbf{d}_{new}) \leq \mathcal{T} \\ 0 & \text{otherwise} \end{cases}$$

  - Transition probability depends only on the energy of the new configuration
- To speed the convergence

$$P(\mathbf{d}_{curr} \rightarrow \mathbf{d}_{new}) = \begin{cases} 1 & \text{if } L(\mathbf{d}_{new}) \leq \mathcal{T} \\ 1 & \text{if } \Delta L \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Ergodicity** - starting at a random chosen configuration and using the random walk acceptance criterion, every other configuration must be reachable.

Outline

Standard cooling schedules

- Temperature-dependent parameters, with role in the optimization process, derived from physical systems
  - **Specific heat** $C(T) = (k_B T^2)^{-1} Var(\mathcal{H})$
  - **Freezing temperature** T_f - the value where the specific heat exhibits a wide peak
  - ΔT_f - the half of width of the specific heat peak
- General cooling schedule - necessary and sufficient for having a probability of one to stop in a global optimum

$$T = \frac{a}{b + \log(t)}$$

  - a, b - positive constants depending on problem
  - t - times $(0, 1, 2, ...)$, as the number of Monte Carlo steps
  - time to get the optimum for SA algorithm - infinite !
- Need faster ways of cooling
  - may be applied generally
  - don't guarantee a convergence to the global minimum

Standard cooling schedules (cont.)

- **Linear/arithmetic cooling** $T = a - b \times t$
  - a - initial temperature
  - b - decrement (usually in $[0.01, 0.2]$)
- **Exponential cooling** $T = a \times b^t$
  - a - initial temperature
  - b - cooling factor (usually in $[0.8, 0.99]$)
- The best cooling schedule - depends on the problem and of resources
  - $C(T)$ (almost) symmetric when plotted against a linear temperature scale $\Rightarrow$ linear cooling schedule
  - $C(T)$ (almost) symmetric when plotted against logarithmic temperature scale $\Rightarrow$ exponential cooling schedule

Initial temperature

- If initial temperature too low - system restricted to local minima of initial configuration
- If initial temperature too high - too much calculation time wasted
- Ad-hoc procedure
  - At the beginning of optimization run, perform a random walk
  - Consider $|\Delta L|_{max}$ - maximum difference of L occurring between successive configurations
  - For SA with an acceptance rate of moves equal 0.9

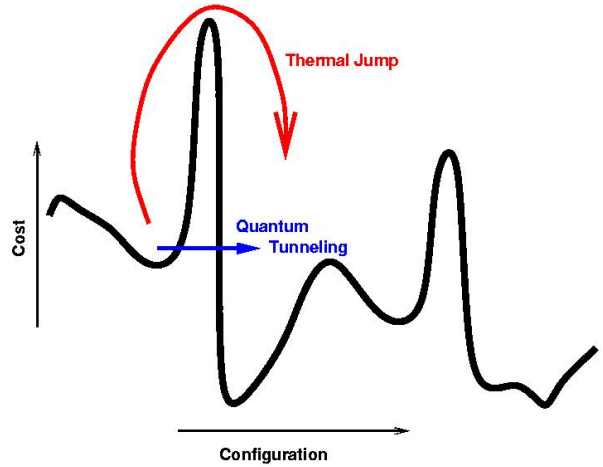
$$T_{init} = -\frac{|\Delta L|_{max}}{ln(0.9)} \approx 10 \times |\Delta L|_{max}$$

  - For TA - $T_{init} = |\Delta L|_{max}$
  - For GDA - $T_{init} = L_{max}$, where L_{max} is the maximum value of loss function during the random walk
- *Remark:* Final temperature - 0 for linear cooling, 1 for exponential cooling.

Outline

Search space smoothing

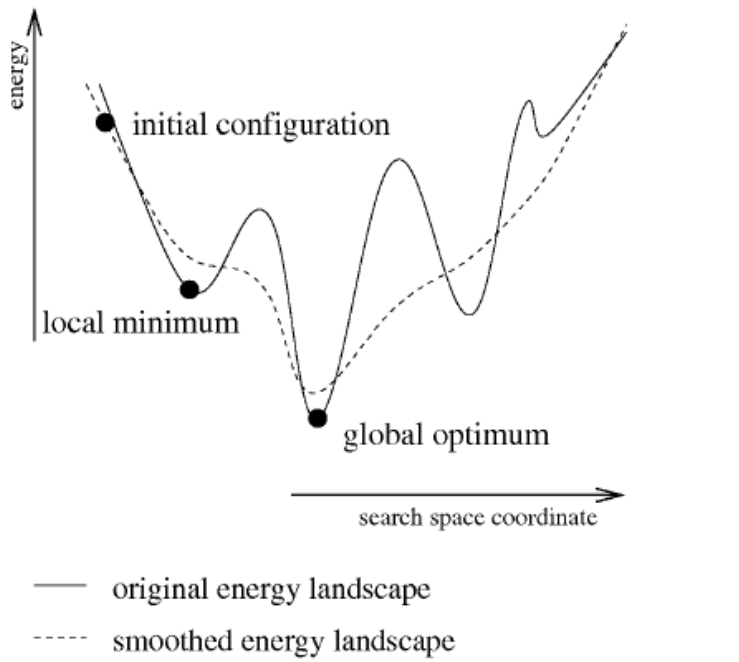
- Approaches to avoid to be trap in a bad local minima
  - First approach (SA related algorithms) - possibility to "climb" over barriers by accepting, with a given probability, less good configurations
  - Second approach - introduce additional moves (so a larger neighborhood) allowing to "tunnel" through barriers (quantum annealing); increase the possibility to "miss" the global optimum



  - Third approach - smooth the energy landscape so the random walk can easily "jump" over barriers (or remove barriers completely)
- Ideal way of smoothing: the number of minima is reduced to one global optimum

Smooth methods

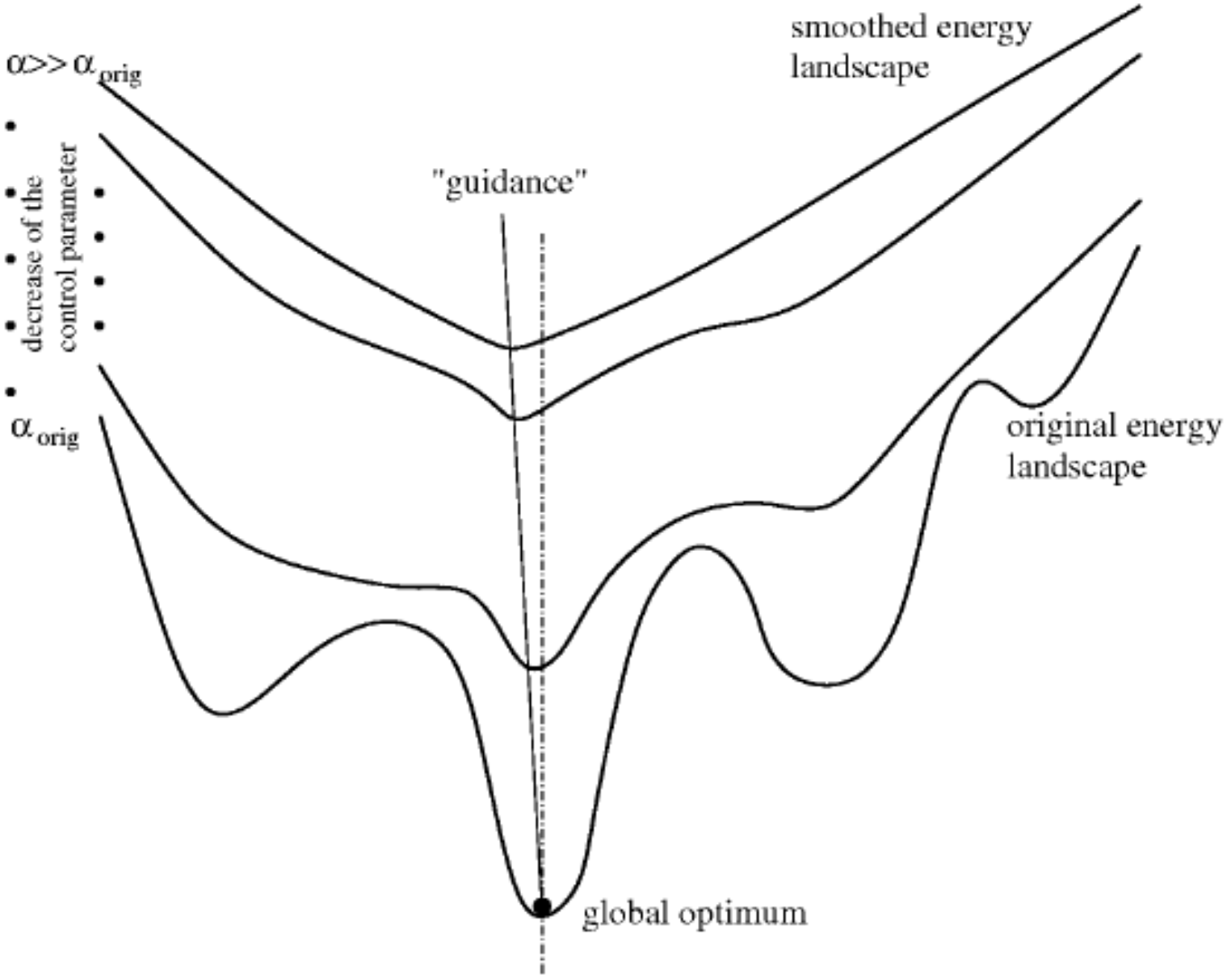
- How to smooth the energy landscape?
  - Change the loss function by smoothing the value in each configuration: $L(\mathbf{d}) \rightarrow f(L(\mathbf{d}))$
  - Apply the smoothing function f on the loss function difference $\Delta L = L(\mathbf{d}_{new}) - L(\mathbf{d}_{curr})$ and get $f(\Delta L)$
  - If $L(\mathbf{d}) = \sum_i L_i(\mathbf{d})$, then different smoothing functions f_i may be applied: $\sum_i f_i(L_i(\mathbf{d}))$
  - Similarly, ΔL may be smoothed as $\sum_i f_i(\Delta L_i(\mathbf{d}))$



Smooth methods (cont.)

- Dilemma - efficient smooth means knowing the shape of energy landscape which means knowing the optimum which means no need to smooth !
- Use of a non-linear function, as $f(x) = \ln(x)$
- Difficulty - local minima or global minima may be displaced
  - an optimization in the smoothed landscape will end up in a configuration which is not local/global optimum for original system
- **Smoothness control parameter** - govern the smoothing process
  - at the beginning - large value for control parameter, so one global optimum
  - at each successive iteration - reduce slowly control parameter (desmoothing process), to keep the guidance effect

Smoothness control parameter



Search Space Smoothing for TSP

Step 1. Normalise all distances:

- $d_n(i,j) = \frac{d(i,j)}{d_{max}}$, where $d_{max} = \max(d(i,j)), \forall i,j = 1..n$
  % linear transformation - the energy landscape is not changed

Step 2. Calculate the normalized mean:

$$\overline{d} = \frac{1}{n(n-1)} \sum_{\substack{i,j=1..n \\ i \neq j}} d_n(i,j), \text{ so } \overline{d} \in (0,1]$$

Step 3. Calculate the deviations: $\Delta(i,j) = d_n(i,j) - \overline{d}$
(we have $0 \leq |\Delta(i,j)| < 1$)

Step 4. Introduce a power law for the smoothed distances

$$d_\alpha(i,j) = \begin{cases} \overline{d} + \Delta(i,j)^\alpha & \text{if } \Delta(i,j) \geq 0 \\ \overline{d} - (-\Delta(i,j))^\alpha & \text{if } \Delta(i,j) < 0 \end{cases}$$

Gu & Huang approach

- For $\alpha \gg 1$, $d_\alpha(i,j) \approx \overline{d}$ - extreme smooth landscape
- For $\alpha = 1$, $d_\alpha(i,j) = d_n(i,j)$ - retrieve original energy landscape
- Start with a large smoothness control parameter α.
- At each value of α, apply several greedy steps using the nearest exchange move (two successive nodes in the path are exchanged) until a minimum is reached.
- Decrease slowly α (using a linear approach) and repeat the greedy algorithm

Variants of Smoothing formula

- Exponential smoothing ($\alpha = 0$ - original landscape)

$$d_\alpha(i,j) = \begin{cases} \overline{d} + \frac{\alpha}{exp(\frac{\alpha}{\Delta(i,j)})-1} & \text{if } \Delta(i,j) \geq 0 \\ \overline{d} - \frac{\alpha}{exp(\frac{\alpha}{-\Delta(i,j)})-1} & \text{if } \Delta(i,j) < 0 \end{cases}$$

- Sigmoidal smoothing ($\alpha \to 0$ - original landscape)

$$d_\alpha(i,j) = \overline{d} + \frac{tanh(\alpha\Delta(i,j))}{\alpha}$$

- Logarithmic smoothing

$$d_\alpha(i,j) = \begin{cases} \overline{d} + \frac{\log(1+\alpha\Delta(i,j))}{\alpha} & \text{if } \Delta(i,j) \geq 0 \\ \overline{d} - \frac{\log(1-\alpha\Delta(i,j))}{\alpha} & \text{if } \Delta(i,j) < 0 \end{cases}$$

Outline

Darwin's natural selection

- Darwin's observations and deductions
  - if no external influences occur, the food resources are limited but stable over time
  - the individuals of a species produce more offspring than can grow into adulthood
  - individuals compete for these limited resources, so a struggle for survival ensues
  - some of the variations between the individuals will affect their fitness and hence, their ability to survive
  - a good fraction of these variations are inheritable
- Natural selection in brief: those individuals that are better adapted survive longer and have a larger probability to mate, thus passing on their variations to the next generation

Offsprings

- Generating offsprings in nature
  - by cloning - an individual splits into two identical halves; some small errors may occur
  - by mating - two individuals not identical mate with each other and have offsprings together, exhibiting a mix of their parents traits
- Two ways to generate new (combination of) traits
  - **mutation** - small random changes occurring with a small probability
  - **crossover** - mix of traits as common offspring of two individuals
- **Chromosome** - store the information about traits

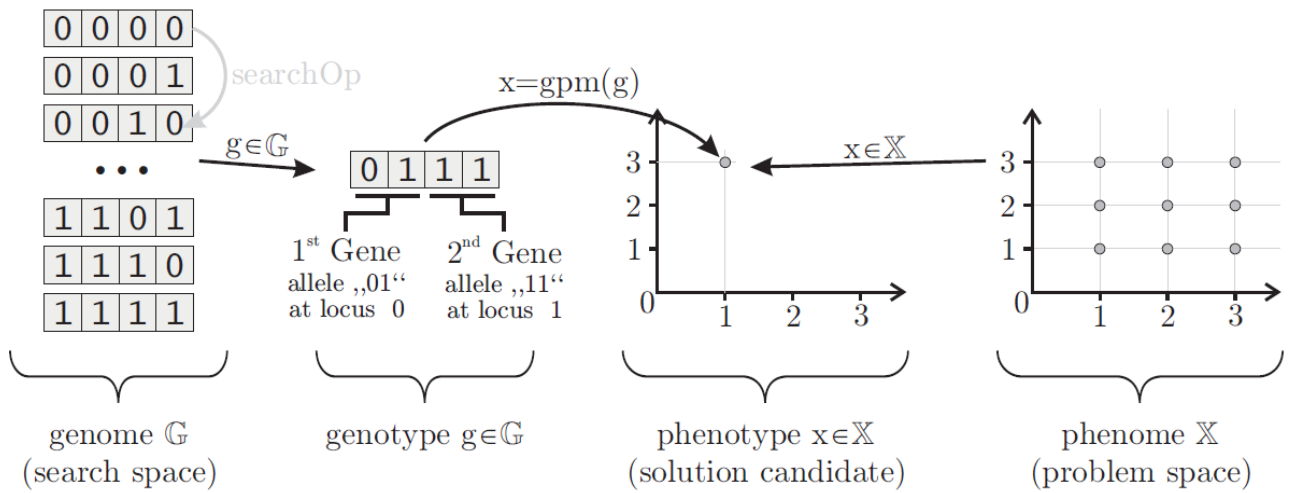
Genotype/Phenotype

- Genome - the whole hereditary information of a species
- Genotype - the hereditary information of a specific individual
- Gene - the distinguishable units of information in a genotype
- Allele - the value of specific gene
- Locus - the position where a specific gene can be found in a genotype
- Phenotype - the individual's observable characteristics and traits
  - **genotype(G) + environment(E)** $\rightarrow$ **phenotype(P)**
- Phenome - the whole observable characteristics of a species

Search and Problem Space

- Let be $\min_{\mathbf{d} \in \Theta} L(\mathbf{d})$ an optimisation problem (O)
- The **search space** G (genome) of an optimisation problem is the set of all elements g (genotype) which can be processed by search operations;
- The **problem space** Θ (phenome) of an optimization problem is the set containing all elements **d** (phenotype) which could be its solution.
- The **genotype-phenotype mapping** (or ontogenetic mapping) $gpm : G \rightarrow \Theta$ is a relation which maps the elements of the search space G to elements in the problem space Θ.

$$\forall g \in G, \exists \mathbf{d} \in \Theta : gpm(g) = \mathbf{d}$$



Search and Problem Space (II)

- For most optimisation algorithms, $G = \Theta$, so $gpm(x) = x$
  - different problem spaces $\Rightarrow$ different sets of search operations
- A huge advantage of using the same search space G for different problems
  - only the mapping *gpm* must be defined
- An *individual* p is a tuple $(p.g, p.\mathbf{d})$ of an element $p.g \in G$ (search space) and the corresponding element $p.\mathbf{d} = gpm(p.g) \in \Theta$ (problem space)
- A *population* Pop is a list of individuals used during an optimization process
- The *fitness* value of an individual p corresponds to its utility in the subsequent steps of optimisation algorithm
  - the fitness may depend on the genotype $p.g$, the phenotype $p.\mathbf{d}$ and the population Pop

Genome design

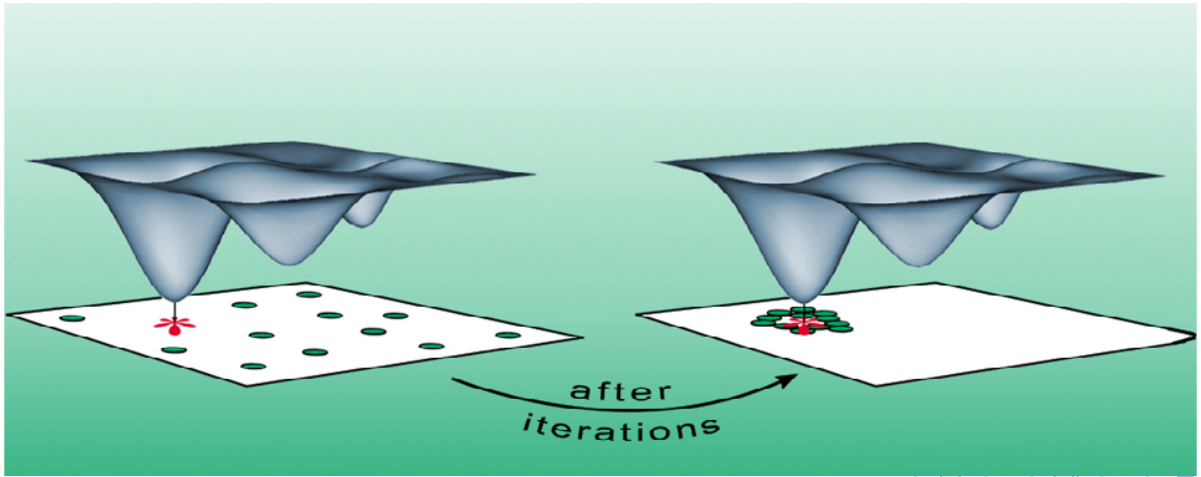
1. The representations in the search space should be as short as possible.
2. The alphabet of the encoding and the lengths of the different genes should be as small as possible.
3. A good *gpm* must be surjective: $\forall \mathbf{d} \in \Theta \, \exists g \in G : gmp(g) = \mathbf{d}$
4. The search space G should be unbiased in the sense that all phenotypes are represented by the same number of genotypes
5. *gmp* should be bijective; the inverse gmp^{-1} is the denoted the coding function
6. The representations in the search space should possess strong causality (locality), i. e., small changes in the genotype lead to small changes in the phenotype

Evolutionary Optimization Heuristics

- Use three biological-based key ingredients on an *artificial population* of individuals: mutation, crossover, selection pressure
- Class of set-based improvement heuristics
- Two types of optimization heuristics using Darwin's principles
  - **Evolution Strategies** - German school (Rechenberg)
  - **Genetic Algorithms** - American school (Holland)
- The difference is on the codding of individuals (the form of search space G)
  - GA - usually bit-strings (wide codding); adapted for combinatorial optimization problem
  - ES - usually real numbers (short codding); adapted for continuous problem

Prototype of evolutionary heuristic

- The fundamental steps
  0. Initialization - Select an initial population size and values for the elements of the population; encode the elements of **d** in a manner convenient for the problem
  1. Evolution - Mix the current population individuals according to algorithmic analogues of principles of evolution and produce a new set of population individuals (a new *generation*)
  2. Evaluation - Measure the performance of the new population individuals and determine if the algorithm should be terminated. If not, return to step 1.



Elements coding

- **Standard bit coding**
  - Gene = bit $b_i \in \{0, 1\}$;
  - Genotype = sequence of bits $b = (b_1, b_2, \ldots, b_n)$, n fix
  - **Encoding procedure**: $\mathbf{d} \rightarrow b$, $\mathbf{d}$ scalar
    0. Let $\mathbf{d}_{min}$ and $\mathbf{d}_{max}$ such that $\mathbf{d} \in [\mathbf{d}_{min}, \mathbf{d}_{max}]$. Let m be the relevant precision ($m > 0$ - number of digits after the decimal point; $m < 0$ - number of digits before the decimal point).
    1. Let $n = \min\{a | a \geq 1, 10^m(\mathbf{d}_{max} - \mathbf{d}_{min}) \leq 2^a - 1\}$ be the number of bits. Let $d = (\mathbf{d}_{max} - \mathbf{d}_{min})/(2^n - 1)$
    2. Given $\mathbf{d}$, calculate $B = round((\mathbf{d} - \mathbf{d}_{min})/d)$. Represent $\mathbf{d}$ as the standard binary representation of B.
  - Remark: $\mathbf{d}_{min} \rightarrow (0, 0, ..., 0)$ and $\mathbf{d}_{max} \rightarrow (1, 1, ..., 1)$
  - **Decoding procedure**

$$\mathbf{d} = \mathbf{d}_{min} + \frac{\mathbf{d}_{max} - \mathbf{d}_{min}}{2^n - 1} \sum_{i=1}^{n} b_i 2^{n-i}$$

- If $\mathbf{d} = (t_1, .., t_p)$ and $b_i = code(t_i)$ then $code(\mathbf{d}) = b_1 b_2 ... b_p$
  - we may have $length(b_i) \neq length(b_j)$

Elements coding

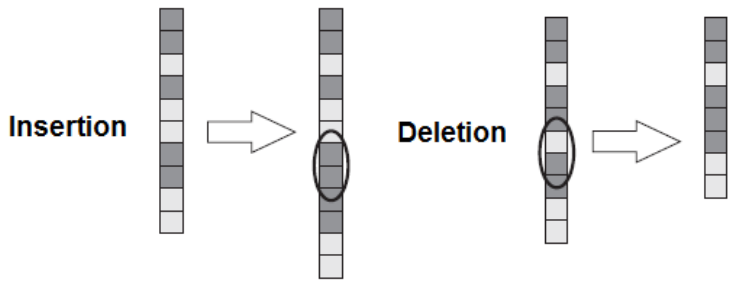
- **Grey coding** - adjacent floating-point values differ by only one bit in the chromosome representation.
  - there is $O(b!2^b)$ gray codes, where b is the number of bits in the representation
  - Michalewicz procedure for translating standard binary coding to gray coding
- Multiple character encodings (more than 2 elements in the string alphabet)
- 10-character coding - referred to as *real-number coding*
  - $code(\mathbf{d}) = \mathbf{d}$ (vector of integer or real numbers)
  - $G = \Theta$
- Messy coding - the ordering of the genes is not fixed
  - each gene is a tuple (ϕ, χ), where ϕ is the position (locus) and χ is the value (allele)
  - 000111 can be represented as
    $g_1 = ((0,0),(1,0),(2,0),(3,1),(4,1),(5,1))$ or as
    $g_2 = ((5,1),(1,0),(3,1),(2,0),(0,0),(4,1))$

Standard Evolutionary Operations

- Mutation - makes "slight" random modifications to some or all chromosomes
- Duplication - creates a "clone" for a chromosome
- Crossover/Recombination - takes pair of parents from the mating pool and creates offspring
- Selection - the mechanism for choosing a set of chromosomes from the present generation to create a mating pool
  - Selection tends to pick best population elements as parents
- Elitism - the strategy consisting to pass best chromosome(s) to the next generation
  - Elite chromosomes also eligible for selection as parents
  - Inclusion of elitism critical to practical performance of GA

Mutation

- Fixed length string chromosome $g = (g_1, g_2, \ldots, g_n)$
  - **switching** - randomly select one (or more) g_i; if $g_i \in \{0, 1\}$, $g_i = 1 - g_i$; if not, generate a new allele from the coding alphabet and replace g_i
  - **shifting**
    - turn around a partial sequence of genes, i.e. $(g_1, .., g_{i-1}, g_i, g_{i+1}, .., g_{j-1}, g_j, g_{j+1}, .., g_n)$ becomes $(g_1, .., g_{i-1}, g_j, g_{j-1}, .., g_{i+1}, g_j, g_{j+1}, .., g_n)$
    - move a randomly selected gene in another position, i.e. $(g_1, .., g_{i-1}, g_{i+1}, .., g_j, g_i, g_{j+1}, .., g_n)$
- Variable-length string chromosome - two new mutation operations
  - **insertion** - insert a couple of genes with randomly chosen alleles at any given position
  - **deletion** - delete elements from the string



Mutation (II)

- Messy codes
  - **inversion** operator - reverses the order of genes between two randomly chosen positions
- Real number code p_i
  - adding a random number r_i: $p_i^{new} = p_i^{curr} + r_i$
  - usually, $r_i \sim N(0, \sigma_i^2)$
  - σ_i is decreasing during the optimization run (more emphasis to the small mutations at the end)

Crossover

- Breaking and recombining chromosomes
- Let $p = (p_1, \ldots, p_n)$ and $q = (q_1, \ldots, q_n)$
  - **one-point crossover** - select randomly a position $i \in \{1, .., n\}$; combine p and q to get the children
    $c = (p_1, .., p_i, q_{i+1}, .., q_n)$ and
    $d = (q_1, .., q_i, p_{i+1}, .., p_n)$
  - **two-points crossover** - select randomly two positions, $i < j$; combine p and q to get the children
    $c = (p_1, .., p_i, q_{i+1}, .., q_j, p_{j+1}, .., p_n)$ and
    $d = (q_1, .., q_i, p_{i+1}, .., p_j, q_{j+1}, .., q_n)$
  - **N points crossover** - generate a random bitstring r; if $r_i = 0$ then $c_i = p_i$ and $d_i = q_i$; else $c_i = q_i$ and $d_i = p_i$

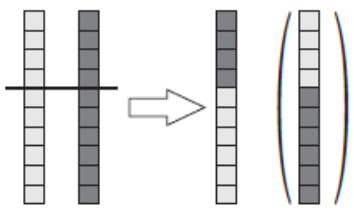


Fig. 3.5.a: Single-point Crossover (SPX).

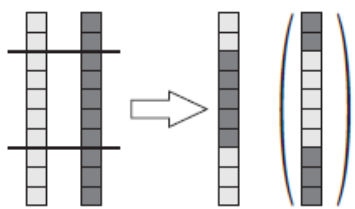


Fig. 3.5.b: Two-point Crossover (TPX).

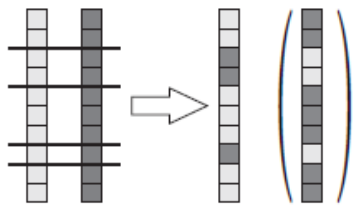


Fig. 3.5.c: Multi-point Crossover (MPX).

Crossover (II)

- For variable-length string chromosomes: the strings are no longer necessarily split at the same position
- Particular case: homologous crossover (only genes at the same loci are exchanged)

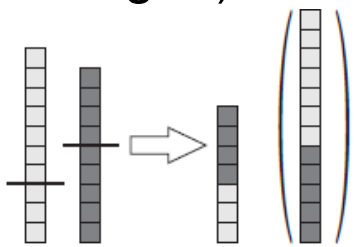


Fig. 3.7.a: Single-Point Crossover

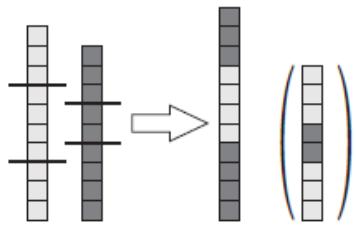


Fig. 3.7.b: Two-Point Crossover

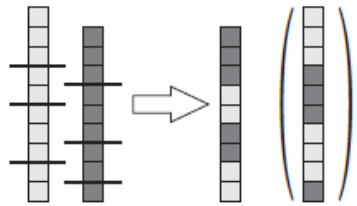


Fig. 3.7.c: Multi-Point Crossover

- Messy codes:
  - **cut** operator - splits a genotype g into two with the probability $p_c = (len(g) - 1)p_k$ where p_k is a bitwise probability and $len(g)$ the length of the genotype
  - **splice** operator joins two chromosomes with a predefined probability p_s by appending one to the other
  - cut and splice may generate overspecification (more genes for the same position) or underspecification (no gene for a specific position)

Fitness function

- The utility of an individual $(p.g, p.\mathbf{d})$ is given by a fitness function
- Higher the fitness value, better the candidate solution
- Common choice is $-L(p.\mathbf{d})$
- Other choices desirable in some applications
  - Add positive constant to ensure fitness always ≥ 0
  - Rescaling to avoid extreme values that can cause instabilities due to selection step

Selection

- *Selection methods* - deterministic or random choice based on fitness
  - *select*(N, M, F): select M individuals from a population of size N using the fitness function F
  - selection may be *without* replacement (an individual may be selected max. once) or *with* replacement (an individual may be selected multiple times)
- **Truncation** - returns the $k < M$ best elements from the population
  - deterministic selection
  - the k elements are duplicated as often as need to reach M individuals
  - usually, $N/3 \leq k \leq N/2$

Selection: Roulette-wheel

- **Roulette-wheel selection** - standard method
- Probability for an individual $x \in Pop$ to be selected - equal to its fitness divided by the total fitness in the population

$$P(x) = \frac{F(x)}{\sum_{y \in Pop} F(y)}$$

- Drawback: probability of selection highly dependents on **units** and **scaling** for fitness function
- Solution: the fitness function must be normalized in the range $[0, 1]$

$$min_F = \min\{F(y), y \in Pop\} \; max_F = \max\{F(y), y \in Pop\}$$

$$norm(F(x)) = \frac{F(x) - min_F}{max_F - min_F}, \quad P(x) = \frac{norm(F(x))}{\sum_{y \in Pop} norm(F(y))}$$

- Exemple: $F(x_1) = 10$, $F(x_2) = 20$, $F(x_3) = 30$, $F(x_4) = 40$. We have $min_F = 10$ and $max_F = 40$, so $max_F - min_F = 30$.

$$norm(F(x_1)) = 0, \; norm(F(x_2)) = \frac{1}{3}, \; norm(F(x_3)) = \frac{2}{3}, \; norm(F(x_4)) = 1$$

$$P(x_1) = 0, \quad P(x_2) = \frac{1}{6}, \quad P(x_3) = \frac{1}{3}, \quad P(x_4) = \frac{1}{2}$$

- May cause premature convergence to local optima

Selection: Tournament

- **Tournament** selection and **rank** selection methods reduce sensitivity to choice of fitness function
- Tournament selection - a pair of individuals is selected uniformly, and the one with the highest fitness is selected
  - The probability for an individual to participate to an tournament is $2/N$.
  - If $M \approx N$, the best individual will have, *in average*, two copies in the mating pool, the average individuals - one copie, and the worst individual - no one.
- Variant: k individuals are selected uniformly in a tournament and a probability p is defined
  - The best individual is included in the mating pool with a probability p
  - The second individual is included in the mating pool with a probability $p(1-p)$
  - The i^{th} individual is included in the mating pool with a probability $p(1-p)^{(i-1)}$

Selection: Rank

- Rank selection - the probability to select an individual is proportional to (a power q of) the rank (1 to N) of the fitness value.
- If k is the expected number of copies of the best individual in the mating pool, then the power $q = \frac{1}{1-\frac{\log(k)}{\log(M)}}$
- The algorithm

Step 1 Sort the list of fitness values $\{F(x)|x \in Pop\}$
Step 2 $q = (1 - \frac{\log(k)}{\log(M)})^{-1}$
Step 3 For $i = 1$ to M
  - Generate uniformly $u \in (0, 1)$
  - Select from Pop the element with index $\lfloor u^q N \rfloor$

Tournament selection creates copies of the better fraction of the population and almost none of the others; Rank selection assigns very high probabilities to very few individuals but preservers also the less fitter ones.

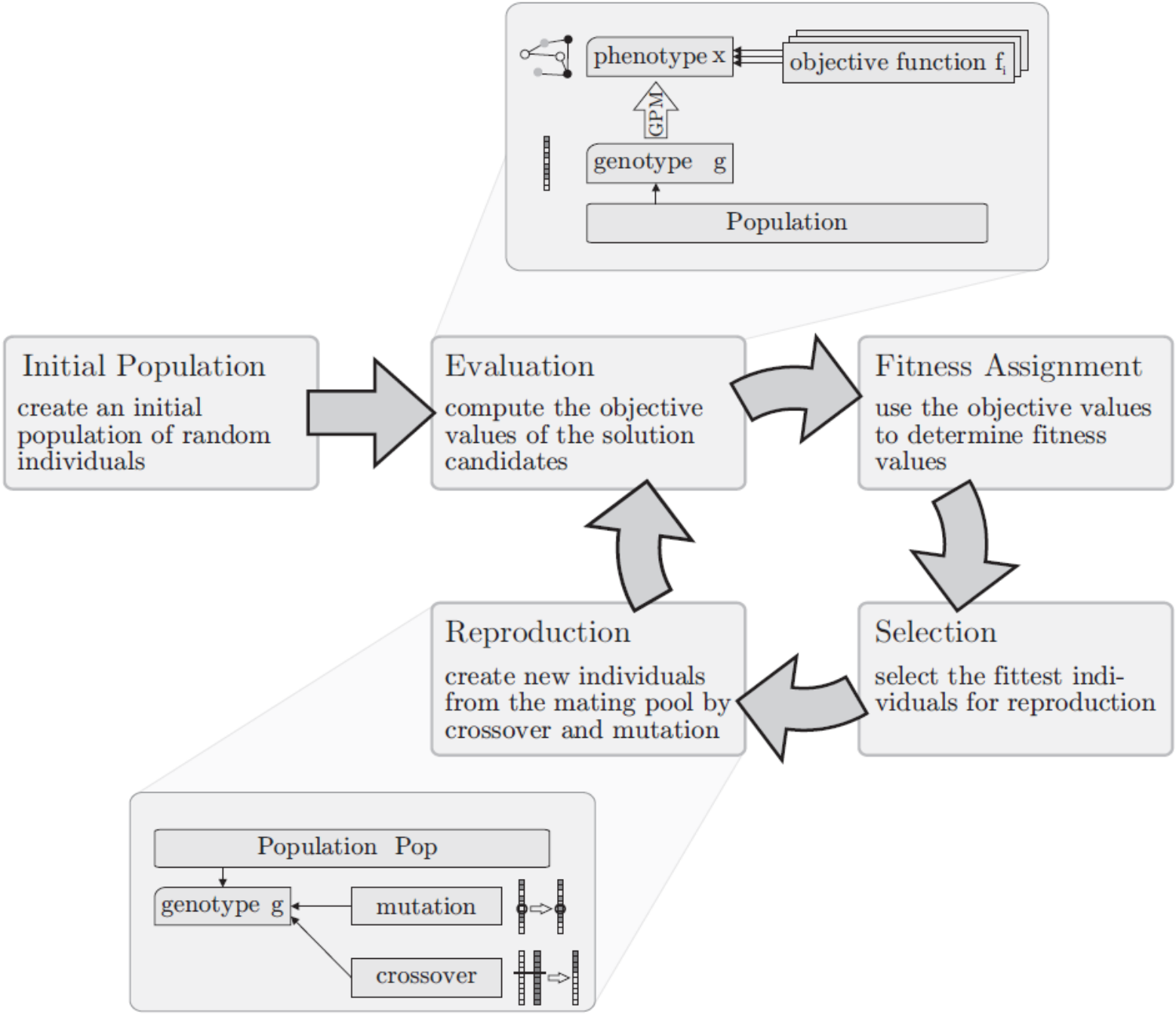
The family of Evolutionary Algorithms

1. **Genetic algorithms** (GA) - the search space G is the space of (bit) strings.
2. **Evolution Strategies** (ES) - the search space G is the space of real vectors $\Theta \subseteq R^p$
3. **Genetic Programming** (GP) - the search space G is the space of programs/algorithms (usually, under a tree representation)
4. **Learning Classifier Systems** (LCS) - online learning approaches assigning output values to given input values using a genetic algorithm to find new rules for this mapping.
5. **Evolutionary programming** (EP) - treats the instances of the genome as different species rather than as individuals.

Distinct features of EAs:

a) The population size or the number of populations used
b) The method of selecting the individuals for reproduction.
c) The way the offspring is included into the population(s).

The basic cycle of GA



Essential Steps of Basic GA

Step 0. **(initialization)**: Randomly generate initial population of N chromosomes and evaluate fitness function.

Step 1. **(parent selection)**: Set $N_e = 0$ if elitism strategy is not used; $0 < N_e < N$ otherwise. Generate the mating pool by selecting with replacement $N - N_e$ parents from full population.

Step 2. **(crossover)**: Create $(N - N_e)/2$ pairs of chromosomes by uniform selection from the mating pool. For each pair of parents perform, with a probability P_c, the crossover operation at a randomly chosen splice point (or points) . If no crossover ($p = 1 - P_c$) then clone the two parents.

Step 3. **(replacement and mutation)**: Replace the non-elite $N - N_e$ chromosomes with the current population of offspring from step 2. Perform mutation on the genes with a small probability P_m.

Step 4. **(end test)** Compute the fitness values for the new population of N chromosomes. Terminate the algorithm if stopping criterion is met; else return to step 1.

Essential Steps of Basic ES Algorithm

Step 0. **(initialization)** Randomly or deterministically generate the initial population of N values of $\theta \in \Theta$ and evaluate L in each of the values.

Step 1. **(offspring)** Generate λ offspring (by cloning and mutation) from current population of N values of θ such that all λ values satisfy direct or indirect constraints on θ.

Step 2. **(selection)** For $(N + \lambda)ES$, select N best values from combined population of N original values plus λ offspring; for $(N, \lambda)ES$, select N best values from population of $\lambda > N$ offspring only.

Step 3. **(repeat or terminate)** Repeat Steps 1 and 2 or terminate.

- Variant of Step 1. - an offspring is created using ρ individuals
  - $\rho = 2$: *dominant-recessive recombination* - each gene of the two parents are randomly chosen to be dominant or recessive; the offspring receives only dominant genes
  - $\rho > 2$: the offspring's gene at locus *i* is the arithmetic mean of parents' genes at locus *i*

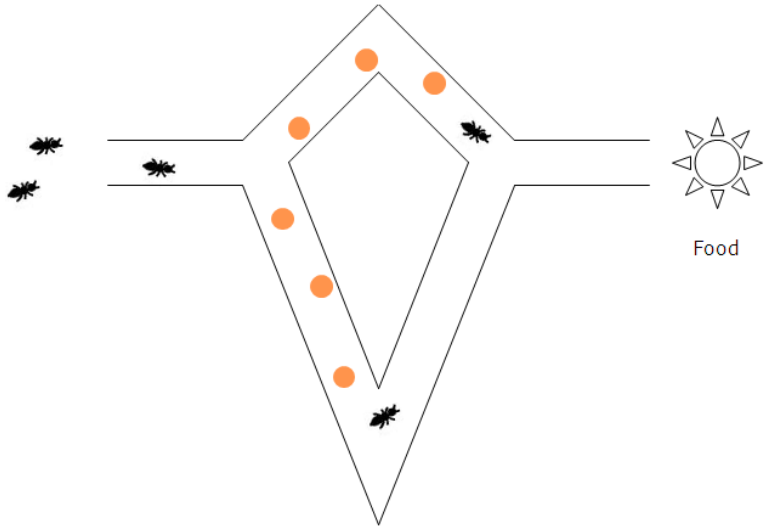
Choice of algorithm-specific coefficients

- Population size N
  - $20 \leq N \leq 100$
  - $N = O(\frac{len*2^s}{s})$, where *len* is chromosome length and s is the average length defined as $\frac{len}{p}$, where p is the problem dimension
- The crossover probability P_c
  - $0.60 \leq P_c \leq 0.95$
- The mutation probability P_m
  - $0.001 \leq P_m \leq 0.01$
  - rule: no mutation for about $30\% - 40\%$ of chromosomes, small mutations for about $30\% - 40\%$ of chromosomes and large mutation for the rest
- adaptive methods for changing some of the coefficient settings over the course of a run

Outline

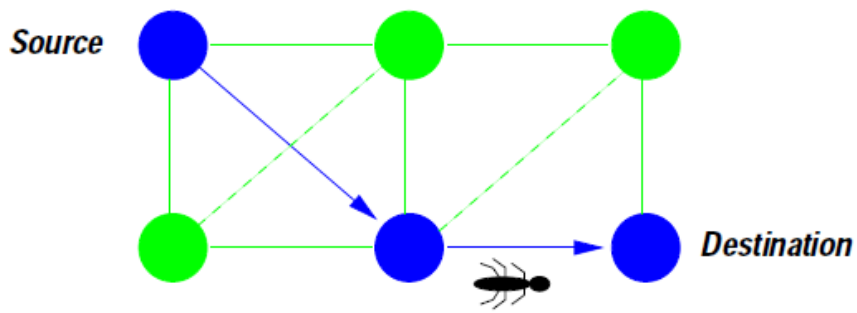
Ant Colony Optimization

- Performance of social insects
  - Can explore vast areas without global view of the ground
  - Can find the food and bring it back to the nest
  - Will converge to the shortest path
- How?
  - By leaving pheromone behind them
  - Wherever they go, they let pheromone behind here, marking the area as explored and communicating to the other ants that the way is known
  - Autocatalytic behavior: the more ants follow the trail, the more attractive the trail becomes



The simple ACO algorithm

- Objective: find the shortest path between a pair of nodes on a graph
- Source node S and destination node D
- For each arc (i, j) associate the variable τ_{ij}, called **artificial pheromone trail**
  - any ant may read/write τ_{ij}
  - τ_{ij} value - proportional to the utility (as estimated by ants) to use the arc (i, j)



Ant behavior

- A step-by-step constructive decision to build the solution
- At each node, local information is used in a stochastic way
- Ant k in node i having $\mathcal{N}_i$ one-step neighbors
  - Select the node $j \in \mathcal{N}_i$ as the next node according to the probability $p_{ij}^k = \tau_{ij}$
  - Update the pheromone information: $\tau_{ij} = \tau_{ij} + \Delta\tau$
  - Evaporation: $\tau = (1 - \rho)\tau$, for any pheromone trail
- After arriving in D, the ant dies.

Ants general characteristics

- Search for minimum cost feasible solutions
- Has a *memory* $\mathcal{M}$ used to
  - (i) build feasible solutions
  - (ii) evaluate the solution found
  - (iii) retrace the path backward
- Can move from node *i* to any node *j* in its *feasible neighborhood* $\mathcal{N}_i$
- Has assigned a *start node* and one/more *termination conditions*
- *Probabilistic decision rule* to move from *i* to *j* depends on
  - (i) pheromone trails τ_{ij} - represents the group intelligence
  - (ii) connection cost for arc (i, j) - represents the intelligence of individual ants
  - (iii) private memory $\mathcal{M}$
  - (iv) problem constraints

Pheromone models

- **Pheromone update**
  - *online update* - after moving, an ant can update the pheromone trail τ_{ij} with
    - (i) an equal amount for each arc (ant-density model) or
    - (ii) arc cost proportionally amount (for ant-quantity model)
  - *off-line update* - once built a solution, an ant can retrace the same path backward and update the pheromone trails on traversed arcs (ant-cycle model); the higher the quality of solution, the more pheromone the ant is allowed to put
  - *evaporation* - the pheromone trail intensity of connections decreases over time to avoid premature convergence

ACO for TSP

Step 0. (**Initialisation**) Put a small amount of pheromone on each edge ($\tau_{ij}(0) = \delta_0, \forall i, j = 1..n$); Place M ants ($M > n$) randomly on the nodes such that at each node at least one ant starts its roundtrip. Set the tabu list (the set of visited nodes) of each ant as empty.

Step 1. Repeat the following steps n times

- Move (i) For each ant, select a new node according to the transition probability
  (ii) Add the new node to the tabu list of the ant
- Update Update the amounts of pheromones on each edge and the probabilities for choosing an edge

Step 3. M roundtrips are created. If some final criterion is not meat, empty the tabu list of each ant and return to Step 1.

The transition probability

- The amount of pheromone on the edge (i, j) at step t: $\tau_{ij}(t)$
- An ant in the node i at step t will choose the next node j not belonging to the ant's tabu list the with the probability

$$p_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha} d(i,j)^{-\beta}}{\sum_{\substack{k=1..n \\ k \notin \text{ant's tabu list}}} \tau_{ik}(t)^{\alpha} d(i,k)^{-\beta}}$$

- If j belongs to ant's tabu list, $p_{ij}(t) = 0$
- The parameters α and β control the relative weight of pheromone amount and distance value
  - $\alpha = 0$: the closest cities are more likely to be selected (like the greedy algorithm)
  - $\beta = 0$: only the pheromone amplification works
  - $\alpha, \beta \neq 0$: compromise between local search and global collective approach

Pheromone update

- $\Delta_{ij}^{k}(t)$ - the amount of pheromone added by the ant k which moves from the node i to the node j (or vice versa) during the step t
  - ant-density model: $\Delta_{ij}^{k}(t) = Q_d$, Q_d constant
  - ant-quantity model: $\Delta_{ij}^{k}(t) = Q_q/d(i,j)$, Q_q constant
- The update may be also applied **after** the ant k finishes its roundtrip
  - $\Delta_{ij}^{k} = Q_c/L_k$, where Q_c constant, L_k is the length of the tour performed by the ant k and (i,j) is an edge that are part of the tour
- Pheromone update:

$$\tau_{ij}(t+1) = \mu \times \tau_{ij}(t) + \sum_{k=1}^{M} \Delta_{ij}^{k}(t)$$

where $\mu < 1$ is the evaporation coefficient