Computational statistics

Chapter 2: Combinatorial optimization

Thierry Denœux
Université de technologie de Compiègne

2022-2023

Combinatorial optimization

- Assume we seek the maximum of $f(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ where $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and $\boldsymbol{\Theta}$ consists of N elements for a finite positive integer N.
- $\boldsymbol{\Theta}$ is generally generated by combining or sequencing p items, which may be done in a very large number of ways.
- In statistical applications, it is not uncommon for a likelihood function to depend on configuration parameters which describe the form of a statistical model and for which there are many discrete choices, as well as a small number of other parameters that are easily optimized if the best configuration were known.
- In such cases, we may view $f(\boldsymbol{\theta})$ as the log profile likelihood of a configuration $\boldsymbol{\theta}$, i.e., the highest log-likelihood attainable using that configuration.

Example: traveling salesman problem

- The salesman must visit each of p cities exactly once and return to his point of origin, using the shortest total travel distance. We seek to minimize the total travel distance over all possible routes. There are $(p-1)!/2$ possible routes (since the point of origin and direction of travel are arbitrary).
- Any tour corresponds to a permutation of the integers $1, \ldots, p$, which specifies the sequence in which the cities are visited.
- Statisticians frequently face combinatorial problems where $\boldsymbol{\Theta}$ has $p!$ or 2^p elements.

Complexity

- The traveling salesman problem is NP-complete. There is no known solution in $\mathcal{O}(p^k)$ steps for any k.
- Consider two problems. The first can be solved in $\mathcal{O}(p^2)$ operations, and the second $\mathcal{O}(p!)$ operations.

	Time to solve problem of order...	
p	$\mathcal{O}(p^2)$	$\mathcal{O}(p!)$
20	1 minute	1 minute
21	1.10 minutes	21 minutes
25	1.57 minutes	12.1 years
30	2.25 minutes	207 million years
50	6.25 minutes	2.4×10^{40} years

The big number exceeds a billion billion billion times longer than the age of the universe!

- There are optimization problems that are inherently too difficult to solve exactly by traditional means.

Example 1: Variable selection in regression

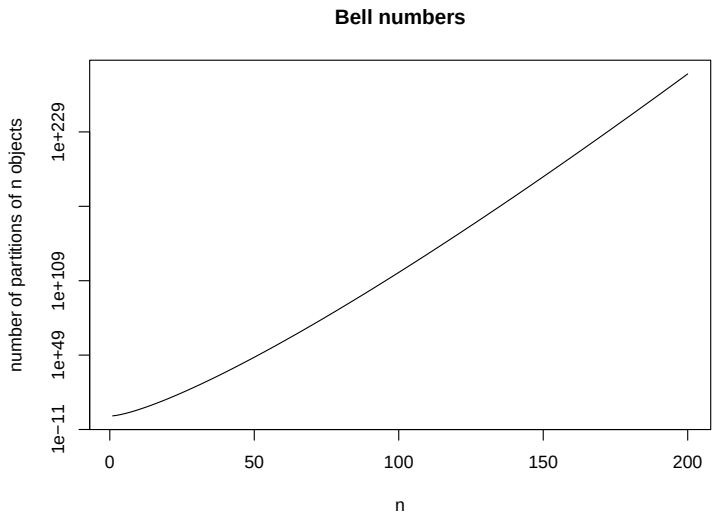
- Given a dependent variable Y and a set of candidate predictors $x_1, x_2, \ldots, x_p$, we must find the best model of the form $Y = \beta_0 + \sum_{j=1}^{s} \beta_{i_j} x_{i_j} + \epsilon$ where $\{i_1, \ldots, i_s\}$ is a subset of $\{1, \ldots, p\}$.
- The goal is to select the best model according to the Akaike information criterion (AIC), i.e., minimize AIC. (Reminder: $\text{AIC} = -2\ell(\widehat{\theta}) + 2p$, where p is the number of parameters).
- The variable selection problem requires an optimization over a space of 2^{p+1} possible models, since each variable and the intercept may be included or omitted.

Example 2: Clustering

- Problem: Find a partition of a set of n points that optimizes some criterion.
- Number of partitions on n points: Bell numbers

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

- $B_1 = 1, B_2 = 2, \ldots, B_5 = 52, B_6 = 203, B_{10} = 115975, B_{20} \approx 5 \cdot 10^{13}$



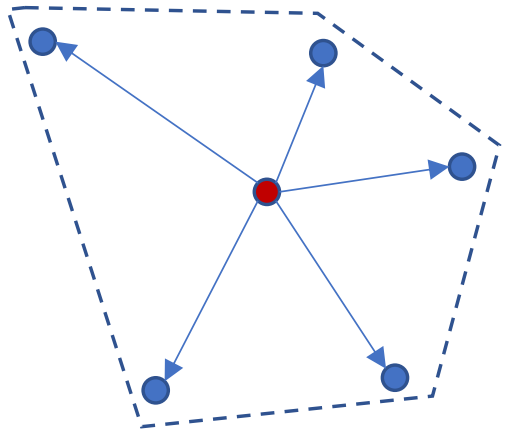
Heuristic search

- Abandon algorithms that are guaranteed to find the global maximum (under suitable conditions) but will never succeed within a practical time limit.
- Turn instead to algorithms that can find a good local maximum within tolerable time.
- We need to trade global optimality for speed.

Overview

Local search

- Relies on
  - iterative improvement of a current candidate solution, and
  - limitation of the search to a local neighborhood at any particular iteration.
- $\boldsymbol{\theta}^{(t)} \longrightarrow \boldsymbol{\theta}^{(t+1)}$ is a move or step.



Definition of neighborhood $\mathcal{N}(\boldsymbol{\theta}^{(t)})$

- Keep it simple. E.g.: define $\mathcal{N}(\boldsymbol{\theta}^{(t)})$ by allowing k changes to the current candidate solution. This is a k-neighborhood, and any move is called a k-change.
- In the regression model selection problem, suppose $\boldsymbol{\theta}^{(t)}$ is a binary vector coding the inclusion/exclusion of each potential predictor. A simple 1-neighborhood is the set of models that either add or omit one predictor from $\boldsymbol{\theta}^{(t)}$.
- Search strategies
  - Simple ascent: $\boldsymbol{\theta}^{(t+1)}$ = any better $\boldsymbol{\theta}$ in $\mathcal{N}(\boldsymbol{\theta}^{(t)})$.
  - Steepest ascent: $\boldsymbol{\theta}^{(t+1)}$ = best $\boldsymbol{\theta}$ in $\mathcal{N}(\boldsymbol{\theta}^{(t)})$.

Random starts local search

- Approach:
  - Select many starting points by stratified or completely at random sampling
  - Run simple/steepest ascent from each point
  - Choose best solution as final answer.
- Works surprisingly well compared to more sophisticated local search techniques.
- The random starts strategy can be overlaid on any optimization method to improve the odds of finding a globally competitive final answer, or even the global optimum.

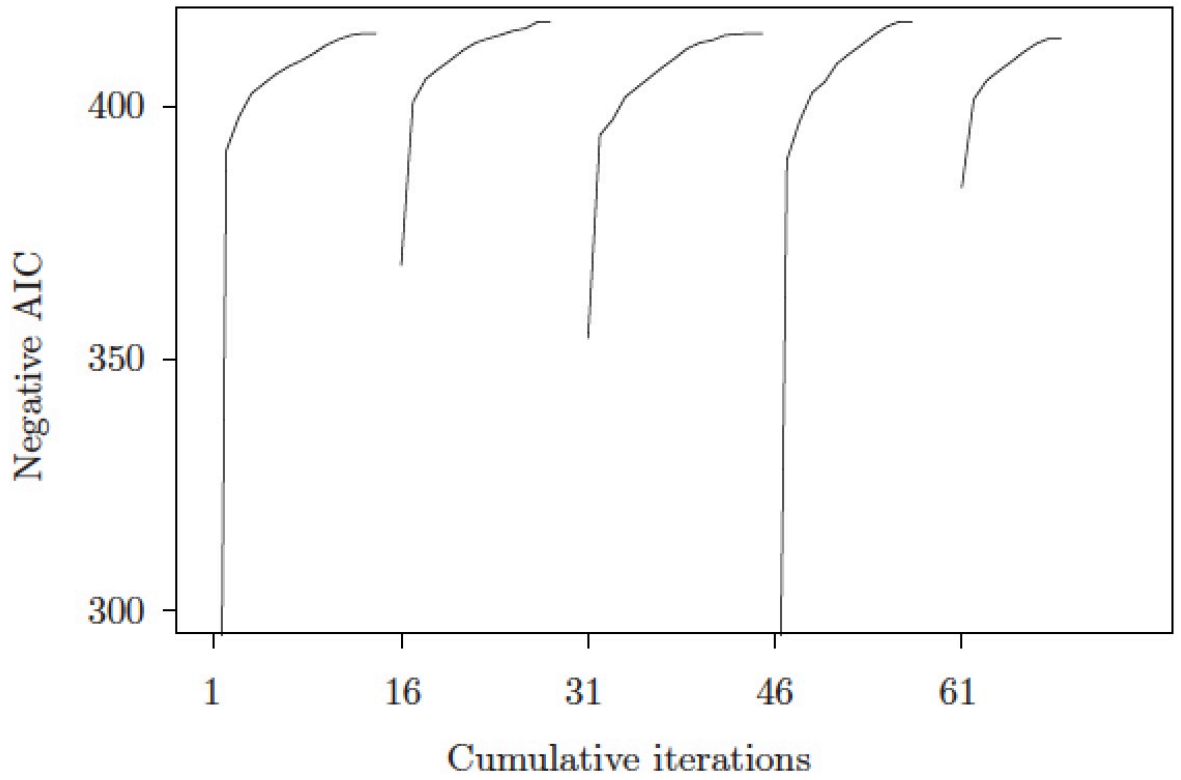
Example: Regression with 27 predictors and 2^{27} possible models

Table: Local search (LS) model selection results using 1-change steepest ascent, for 5 random starts.

	Predictors selected																					
Method	1	2	3	6	7	8	9	10	12	13	14	15	16	18	19	20	21	22	24	25	26	AIC
LS (2,4)		•	•	•		•		•		•	•	•	•						•	•	•	−416.95
S-Plus	•		•	•		•		•		•	•	•	•						•	•	•	−416.94
LS (1)			•	•		•		•	•	•	•					•	•	•				−414.60
LS (3)			•	•		•		•		•	•					•	•	•	•	•		−414.16
LS (5)			•			•	•	•		•	•				•	•	•	•				−413.52
Efroymson			•		•	•		•		•	•			•					•		•	−400.16

The S-Plus (function `step()`) and Efroymson's methods are greedy stepwise procedures.

Example: Regression (continued)



Results of random starts local search by 1-change steepest ascent for the regression example, for up to 15 iterations from each of five random starts. Only AIC values between -300 and -420 are shown.

Overview

Basic principles

- Motivated by analogy to the physics of slowly cooling solids. The method models the physical process of heating a material and then slowly lowering the temperature to decrease defects, thus minimizing the system energy.
- Traditionally problems are posed as minimizations. We adopt this convention for this algorithm only.
- Relies on a sequence of numbers, τ_0, τ_1, $\tau_2 \ldots$, called temperatures. Temperature parameterizes how willing the algorithm is to make non-improving steps.

Basic principles (continued)

- Starts at time $t = 0$ with an initial point $\boldsymbol{\theta}^{(0)}$ and a temperature τ_0. The algorithm is run in stages indexed by $j = 0, 1, 2, \ldots$, and each stage consists of several iterations. The length of the jth stage is m_j.
- The new candidate solution is always adopted when it is better to the current solution, and sometimes adopted even when it is worse.
- This is stochastic descent designed to enable escape from uncompetitive local minima.

Algorithm

1. Select a candidate solution $\boldsymbol{\theta}^*$ within the neighborhood of $\boldsymbol{\theta}^{(t)}$, say $\mathcal{N}(\boldsymbol{\theta}^{(t)})$, according to a proposal density $g^{(t)}(\cdot \mid \boldsymbol{\theta}^{(t)})$.
2. Randomly decide whether to adopt $\boldsymbol{\theta}^*$ as the next candidate solution or to keep another copy of the current solution. Specifically, we have $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^*$ with probability equal to

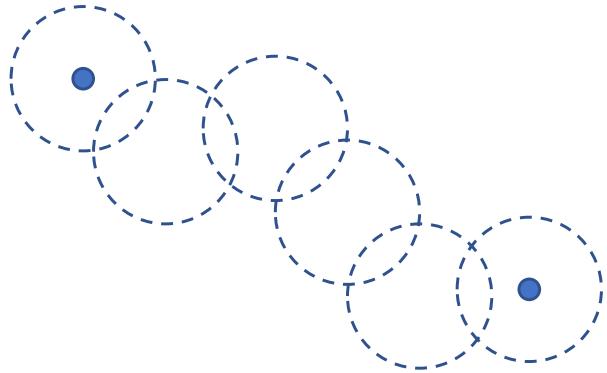
$$\min\left(1, \exp\left\{(f(\boldsymbol{\theta}^{(t)}) - f(\boldsymbol{\theta}^*))/\tau_j\right\}\right).$$

   Otherwise, let $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)}$.
3. Repeat steps 1 and 2 a total of m_j times.
4. Increment j. Update $\tau_{j+1} = \alpha(\tau_j)$ and $m_{j+1} = \beta(m_j)$. Go to step 1.

$\alpha(\cdot)$ should slowly decrease temperatures to zero.
$\beta(\cdot)$ should increase stage lengths as temperatures drop.

Neighborhoods for simulated annealing

- Small and easily computed. E.g., in variable selection, add or remove one variable.
- Neighborhood structure must allow all elements of $\boldsymbol{\Theta}$ to communicate (any two candidate solutions must be connectable via a sequence of neighborhoods).



- Common proposal distribution: discrete uniform over $\mathcal{N}(\boldsymbol{\theta}^{(t)})$.

Cooling schedule and convergence

- Cooling should be slow.
- Simulated annealing produces a Markov chain $\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots$ because $\boldsymbol{\theta}^{(t+1)}$ depends only on $\boldsymbol{\theta}^{(t)}$.
- The distribution of $\boldsymbol{\theta}^{(t)}$ converges to some stationary distribution as $t \to \infty$.
- In principle, we would like to run the chain at each fixed temperature long enough that the Markov chain is approximately in its stationary distribution before the temperature is reduced.
- Theoretical analysis of cooling schedules leads to "recommendations" requiring impracticably long run lengths, sometimes longer than exhaustive search.

Theoretical results

Theorem

- *First fix temperature at τ. Suppose that proposing $\boldsymbol{\theta}_i$ from $\mathcal{N}(\boldsymbol{\theta}_j)$ has the same probability as proposing $\boldsymbol{\theta}_j$ from $\mathcal{N}(\boldsymbol{\theta}_i)$ for any pair of solutions $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ in $\boldsymbol{\Theta}$.*
- *Then the sequence of $\boldsymbol{\theta}^{(t)}$ generated by simulated annealing is a Markov chain with stationary distribution*

$$\pi_\tau(\boldsymbol{\theta}) \propto \exp\{-f(\boldsymbol{\theta})/\tau\}.$$

- *Consequently,*

$$\lim_{t\to\infty} P(\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}) = \pi_\tau(\boldsymbol{\theta}).$$

Theoretical results (continued)

Theorem

Suppose there are M global minima and the set of these solutions is $\mathcal{M}$. Then,

$$\lim_{\tau \downarrow 0} \pi_\tau(\boldsymbol{\theta}_i) = \begin{cases} 1/M & \text{if } i \in \mathcal{M} \\ 0 & \text{otherwise.} \end{cases}$$

If, for each τ, we wait long enough to reach the stationary distribution, and if we let τ tend to 0, then we are sure to reach a global minimum (after a possibly infinite amount of time!).

Practical advice on cooling schedules

- Popular approach: set $m_j = 1$ for all j and reduce temperature very slowly according to

$$\alpha(\tau_j) = \frac{\tau_j}{1 + a\tau_j}$$

for a small value of a.

- A second option is to set

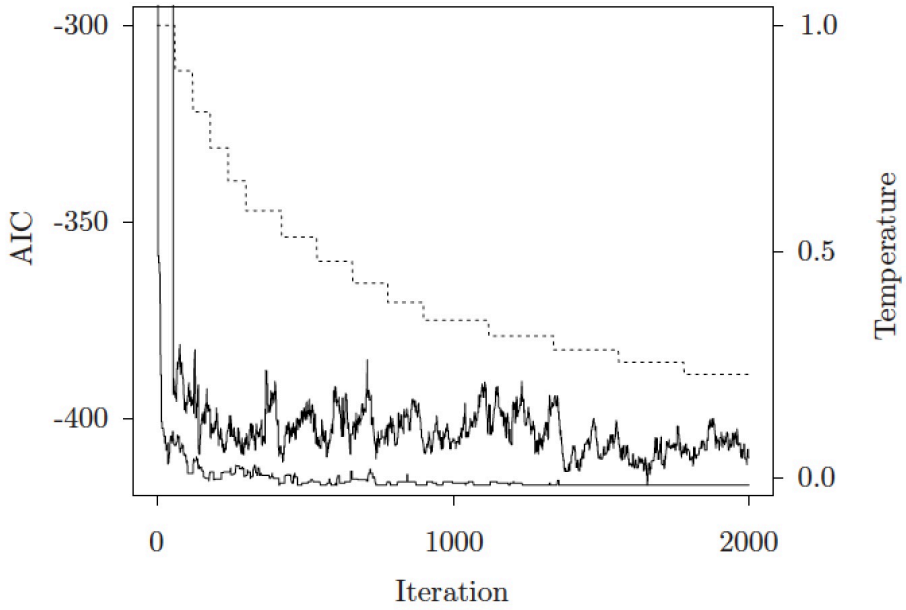
$$\alpha(\tau_j) = a\tau_j$$

for $a < 1$ (usually $a \geq 0.9$). Increase stage lengths as temperatures decrease, using, e.g., $\beta(m_j) = bm_j$ for $b > 1$.

Practical advice on initial temperature

- Selection of the initial temperature, τ_0, is usually problem dependent.
- Try using a positive τ_0 value so that $\exp\{(f(\boldsymbol{\theta}_i) - f(\boldsymbol{\theta}_j))/\tau_0\}$ is close to one for any pair of solutions $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ in $\boldsymbol{\Theta}$. This provides any point in the parameter space with a reasonable chance of being visited in early iterations of the algorithm.
- Final advice: Long runs at high temperatures are unneeded. Decrease temperature rapidly at first but slowly thereafter.

Example: Regression



Temperature for the bottom curve is shown with the dotted line and the right axis. Neighborhoods given by adding or deleting one predictor. Discrete uniform proposal. 15-stage cooling schedule with stage lengths of 5×60, 5×120, and 5×220. Cooling given by $\alpha(\tau_{j-1}) = 0.9\tau_{j-1}$ after each stage. The bottom curve corresponds to $\tau_0 = 1$, where we observe sticking because there is little tolerance for uphill moves. A second run with $\tau_0 = 10$ (top solid line) yielded considerable mixing, with many uphill proposals accepted as moves.

Simulated annealing in R

- Function `optim`, select `method="SANN"`.
- Parameters:
  - `control$temp`: τ_0 (default $= 10$)
  - `control$tmax`: number m_j of function evaluations at each temperature (default $= 10$)
  - `gr`: function to generate a new candidate point.
- Cooling schedule:

$$\tau_{j+1} = \frac{\tau_0}{\log[j\,\text{tmax} + e]}$$

Overview

Basic Principles

- Mimic the process of Darwinian natural selection.
- Candidate solutions are envisioned as biological organisms represented by their genetic code.
- The fitness of an organism is analogous to the quality of a candidate solution.
- Breeding among highly fit organisms provides the best opportunity to pass along desirable attributes to future generations, while breeding among less fit organisms (and rare genetic mutations) ensures population diversity.
- Over time, the organisms in the population should evolve to become increasingly fit, thereby providing a set of increasingly good candidate solutions to the optimization problem.

Basic definitions

- Every candidate solution corresponds to an individual or organism, and every organism is completely described by its chromosome.
- A chromosome is a sequence of C symbols, each of which consists of a single choice from a pre-determined alphabet. E.g., '100110001'.
- The C elements of the chromosome are the genes. The values that might be stored in a gene (i.e., the alphabet) are alleles. The position of a gene in the chromosome is its locus.

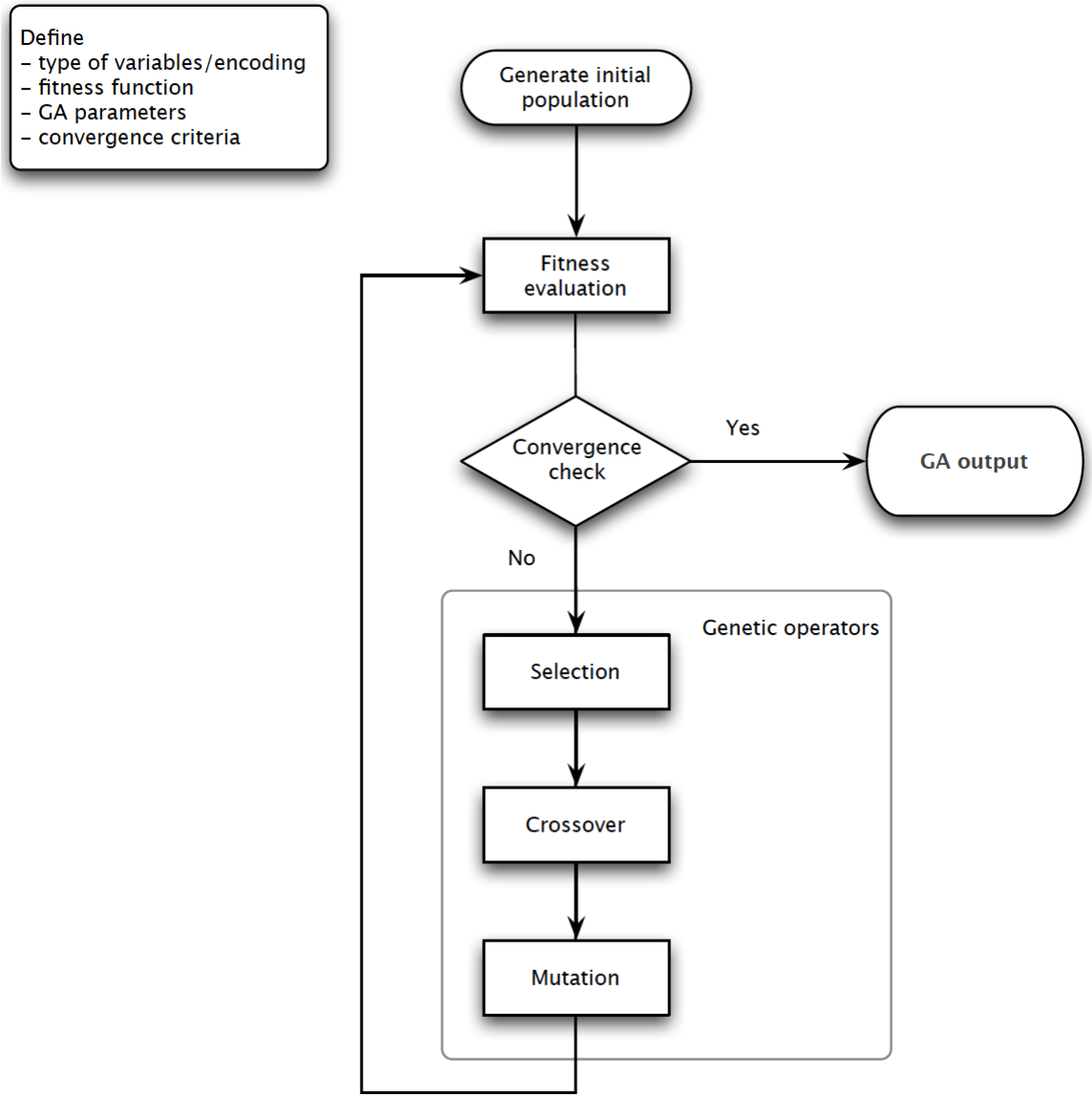
Basic definitions (continued)

- The information encoded in an individual's chromosome is its genotype, $\boldsymbol{\vartheta}$. The expression of the genotype in the organism itself is its phenotype, $\boldsymbol{\theta}$.
- The fitness of an organism (or its genotype) depends on the corresponding $f(\boldsymbol{\theta}_i^{(t)})$.
- Example: regression model selection. Assume 9 predictors.
  - Genotype $\boldsymbol{\vartheta}_i^{(t)}$ ='100110001' corresponds to the phenotype of a model containing only the fitted parameters for predictors 1, 4, 5, and 9.
  - The fitness of this individual equals the negative AIC of this model.

Basic definitions (continued)

- GA's consider many candidate solutions simultaneously. Let the t-th generation consist of a collection of P organisms $\boldsymbol{\theta}_1^{(t)}, \ldots, \boldsymbol{\theta}_P^{(t)}$ having corresponding genotypes (encodings) $\boldsymbol{\vartheta}_1^{(t)}, \ldots, \boldsymbol{\vartheta}_P^{(t)}$.
- As generations progress, organisms inherit from their parents bits of genetic code that are associated with high fitness if fit parents are predominantly selected for breeding.
- An offspring is a new organism inserted in the $(t+1)$-th generation to replace a member of the t-th generation. The offspring's chromosome is determined from two parent chromosomes belonging to the t-th generation.

Flow-chart of a GA



Selection mechanisms

- Breeding drives most genetic change. The process by which parents are chosen to produce offspring is called the selection mechanism.
- Simplest: Select one parent with probability proportional to fitness and select the other parent completely at random, or select both parents with probability proportional to fitness.
- Better: Select a parent with probability equal to

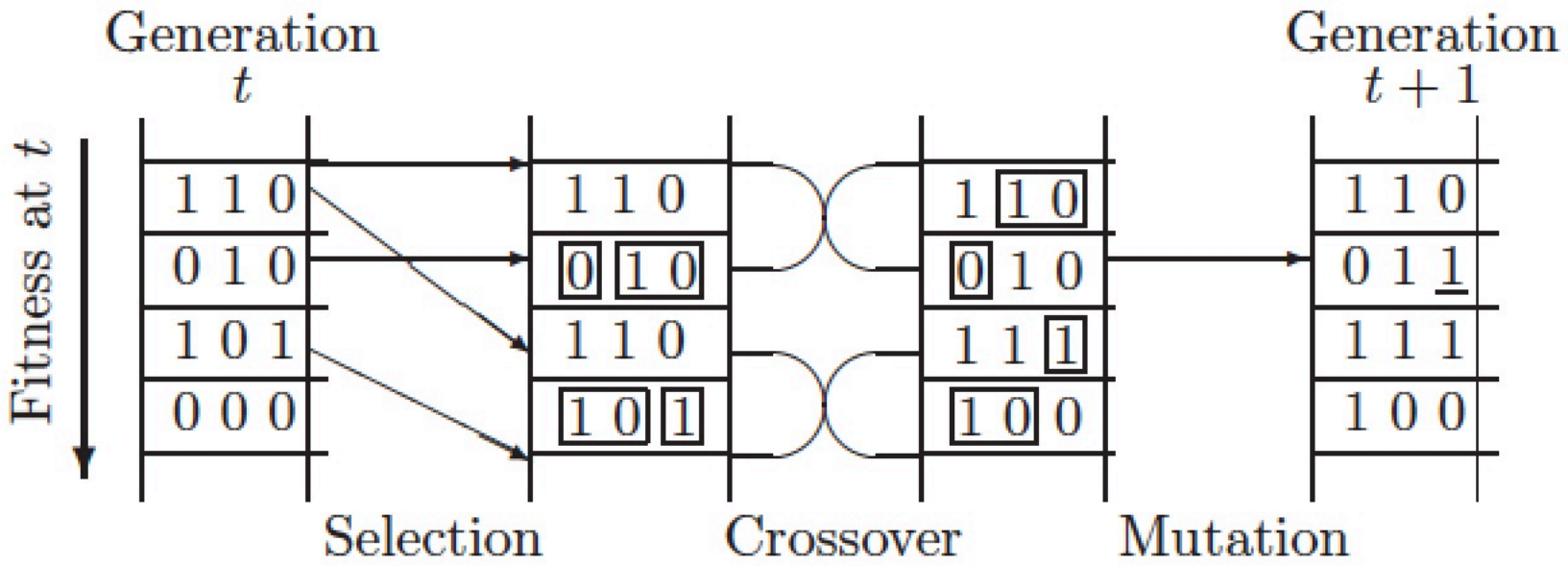
$$\frac{r_i}{P(P+1)/2}$$

  where r_i is the rank of $f(\boldsymbol{\theta}_i^{(t)})$ among generation t.
- Popular: Tournament selection. Randomly partition the set of chromosomes in generation t into k disjoint subsets of equal size. Select the best individual in each group as a parent. Repeat to obtain sufficient parents. Finally, randomly pair parents for breeding.

Genetic operators

- After two parents from the t-th generation have been selected for breeding, their chromosomes are combined in some way that allows attributes from each parent to be inherited by their offspring, who become members of generation $t + 1$.
- Crossover: Select a random position between two adjacent loci and split both parent chromosomes at this position. Glue the left chromosome segment from one parent to the right segment from the other parent to form an offspring chromosome. (See next slide)
- Mutation: Randomly change one or more alleles in the chromosome. Usually applied after breeding (crossover).

Example



An example of generation production in a genetic algorithm for a population of size $P = 4$ with chromosomes of length $C = 3$. Crossovers are illustrated by boxing portions of some chromosomes. Mutation is indicated by an underlined gene in the final column.

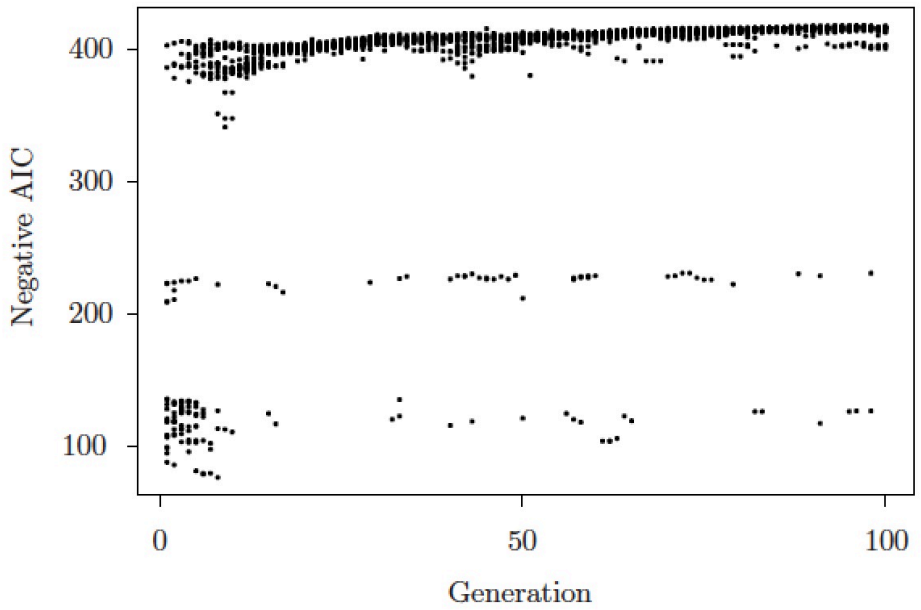
Practical advice

- Initial population: Purely random chromosomes.
- Generation size: Larger provides greater search diversity, but also greater computing burden. For binary encoding of chromosomes, try $C \leq P \leq 2C$, where C is the chromosome length. Real applications often have $10 \leq P \leq 200$, but a review of empirical studies suggests that P can often be as small as 30.
- Mutation rates: Typically very low, in the neighborhood of 1%. Studies have suggested a rate of $1/C$ or a rate roughly proportional to $1/(P\sqrt{C})$.

Practical advice (continued)

- Define the generation gap, G, to be the proportion of the generation to be replaced by generated offspring.
- $G = 1$ corresponds to a canonical genetic algorithm with distinct, non-overlapping generations.
- At the other extreme, $G = 1/P$ corresponds to incremental updating of the population one offspring at a time. Such a steady state GA genetic algorithm produces one offspring at a time to replace the least fit (or some random relatively unfit) individual.
- When $G < 1$, we can ensure that the k best chromosomes are kept in the next generation (elitist strategy).

Example



Results of a genetic algorithm for regression model selection, with 100 generations of size $P = 20$, using rank-based fitness, simple crossover, and 1% mutation. Darwinian 'survival of the fittest' is clearly seen. The twenty random starting individuals quickly coalesce into three effective subspecies, with the best of these slowly overwhelming the rest. The best model was first found in generation 87.

Genetic algorithms in R

- Package `GA`.
- Main function: `ga`
- Some arguments:
  - `type`: set to `"binary"` (other values: `"real-valued"`, `"permutation"`)
  - `fitness`: the fitness function
  - `popSize`: the population size
  - `pcrossover`: the probability of crossover between pairs of chromosomes. Default = 0.8.
  - `pmutation`: the probability of mutation in a parent chromosome. Default = 0.1.
  - `elitism`: the number of best fitness individuals to survive at each generation. By default the top 5% individuals will survive at each iteration.
  - `run`: the number of consecutive generations without any improvement in the best fitness value before the GA is stopped.