



STK4051/STK9051

Computational Statistics

Riccardo De Bin

`debin@math.uio.no`

Outline of the lecture

- Combinatorial Optimization
 - Hard Problems and NP-Completeness
 - Local Search
 - Simulated Annealing
 - Neighbourhoods and Proposals
 - Cooling Schedule and Convergence
 - Genetic Algorithms
 - Definitions and the Canonical Algorithm
 - Variations
 - Initialization and Parameter Values

Combinatorial Optimization: Introduction

Setting:

- maximization of $f(\boldsymbol{\theta})$ wrt $\boldsymbol{\theta} = (\theta_1, \dots, \theta_p)$, $\boldsymbol{\theta} \in \Theta$;
- Θ consists of N elements, with N a positive integer.
- e.g., model fitting:
 - ▶ many discrete choices to identify the form of the model;
 - ▶ other parameters are easily optimized if the best configuration were known.

We need search methods that:

- do not require f to be smooth, continuous, or differentiable;
- can effectively search over vast discrete spaces;
- hard optimization problems, generally combinatorial.

Hard Problems and NP-Completeness: travelling salesman problem

The travelling 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.

I.e.,:

- we seek to minimize the total travel distance;
- a tour corresponds to a permutation of the integers $1, \dots, p$,
 - specifies the sequence in which the cities are visited;
- there are $(p-1)!/2$ possible routes,
 - the point of origin and direction of travel are arbitrary;
 - $123 \dots p = p \dots 321$.

Hard Problems and NP-Completeness: travelling salesman problem

The travelling salesman problem is **NP-complete**,

- there is **no known solution** in $\mathcal{O}(p^k)$ steps for any k .

P (**polynomial**) versus NP (**non-deterministic polynomial**) problems. Example:

- P-problem can be solved in $\mathcal{O}(p^2)$ operations;
- NP-problem can be solved in $\mathcal{O}(p!)$ operations.

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 **age of the universe** is about 1.4×10^{10} years;
- plenty of time to get a coffee!

Hard Problems and NP-Completeness: variable selection in regression

Typical model fitting procedure:

- find the best model $Y = \beta_0 + \sum_{j=1}^p \beta_{i_j} x_{i_j} + \epsilon$;
- **variable selection**: find set $\mathcal{S} \subset \{1, \dots, p\}$;
- model estimation: $\hat{\beta}_j$ for $j \in \mathcal{S}$;
- the goal is to **minimize a loss**, e.g. AIC

$$AIC = n \log(n^{-1} \sum_{i=1}^n (y_i - f(x; \hat{\beta}))^2) + 2|\mathcal{S}|$$

- the variable selection problem requires an optimization over a space of 2^p possible models!
 - ▶ 2^{p+1} including (for some unknown reasons) the intercept.

Local Search: Introduction

New perspective on optimization:

- **forget** algorithms that are **guaranteed** to find the global maximum, but will **never** succeed within a practical time limit;
- **turn to** algorithms that can find a **good local maximum** within **tolerable** time.

Trade of global optimality for speed!

Heuristic algorithms are based on two principles:

- **iterative improvement** of a current candidate solution,
 - ▶ $\theta^{(t)} \longrightarrow \theta^{(t+1)}$ is a **move** or **step**;
- **limitation** of the search to a **local neighbourhood**, $\mathcal{N}(\theta^{(t)})$,
 - ▶ $\mathcal{N}(\theta^{(t)})$ contains candidate solutions that are **near** $\theta^{(t)}$.

Note: **no** heuristic works well in all problems.

Local Search: simple algorithms

Definitions:

- A $\mathcal{N}(\theta^{(t)})$ that allows k changes to the current candidate solution is called **k -neighbourhood**;
- any move is called a **k -change**;
- Example (regression model):
 - ▶ $\theta^{(t)}$ binary vector coding inclusion/exclusion of X_i s;
 - ▶ 1-neighbourhood: set of models that add or remove one X_i .

Methods:

- **Steepest Ascent:** $\theta^{(t+1)} = \text{best } \theta \text{ in } \mathcal{N}(\theta^{(t)})$;
- **Simple Ascent:** $\theta^{(t+1)} = \text{any better } \theta \text{ in } \mathcal{N}(\theta^{(t)})$.
- Example (regression model):
 - ▶ forward/backward selection in a steepest ascent with $k = 1$.

Local Search: k -optimization

Note:

- steepest ascent is **not always** advantageous,
 - ▶ favours **immediate** reward;
 - ▶ prone to stuck in **local minima**;
- from time to time, better choose a **suboptimal solution**,
 - ▶ **better exploration** of the space;
 - ▶ in a direction that provides better **rewards later**.
- when k is large,
 - ▶ break the k -change into smaller parts;
 - ▶ allow one or more to be suboptimal.

Local Search: Random starts local search

High **risk** to find a local minimum → **random starts local search**:

- select **many starting points at random**;
- **run** simple/steepest ascent from each point;
- choose **best solution** as final answer.

Works **surprisingly well** compared to more sophisticated local search techniques!

- use it **even** when implementing strategies that provides high chances to find a global minimum;
- additional confidence.

Local Search: Example

Example (Baseball Salaries, data from Watnik, 1998):

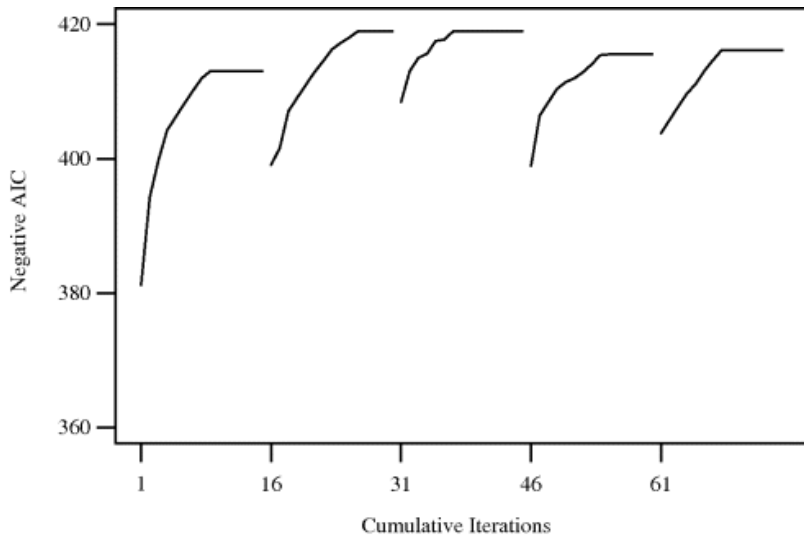
- predict (log of) salaries of baseball players;
- 27 predictors, performance statistics from the previous year;
- $2^{27} = 134\,217\,728$ possible models;
- (intercept correctly always included);
- 1-change steepest ascent.

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
Efroyim.			•		•	•		•		•	•			•					•		•	-400.16

(Givens & Hoeting, 2013, table 3.2)

Local Search: Example



(Givens & Hoeting, 2013, figure 3.3)

Simulated Annealing: Introduction

Simulated Annealing:

- popular technique for **combinatorial optimization**;
- **generic** and **simple** to implement;
- **well studied** limiting behaviour;
- limiting behaviour rather **theoretical**;
- convergence can be **annoyingly slow**.

Motivated by **analogy to the physics** of **slowly cooling** solids:

- thermal energy decreases slowly, **but with random increases**;
 - ▶ at temperature τ , **increase** of ΔE with $p \propto \exp\{-\Delta E/k\tau\}$;
- if the cooling is slow enough and deep enough, the molecules have **minimal** potential energy.

Simulated Annealing: Introduction

Based on the physics analogy:

- we talk about minimization;
- θ corresponds to the **state of the material**;
- $f(\theta)$ is the corresponding **energy level**;
- random changes to the current state $\theta^{(t)} \rightarrow \theta^{(t+1)}$:
 - ▶ based on the **Boltzmann distribution** $p \propto \exp\{-\Delta E/k\tau\}$
 - ▶ k the Boltzmann constant, τ a parameter called **temperature**.

Simulated Annealing relies on the **temperature sequence**

$\tau_0, \tau_1, \dots$

- temperature parameterizes how **willing** the algorithm should make **non-improving steps**;
 - ▶ **higher** temperature, **higher** chances to accept uphill moves;
 - ▶ i.e., **discourages** convergence to the first local minimum;
 - ▶ later in the process (lower temperatures), more focus on finding the **minimum** around the current solution.

Simulated Annealing: algorithm

Simulated annealing initialization:

- **iterative procedure**, start at iteration $t = 0$;
 - ▶ with **initial point** $\theta^{(t)}$ and **temperature** τ_0 ;
- based on different **stages** $j = 0, 1, 2, \dots$ of m_j iterations.

Simulated annealing iteration:

1. select a **candidate** solution θ^*
 - ▶ **within** $\mathcal{N}(\theta^{(t)})$, the neighbourhood of $\theta^{(t)}$;
 - ▶ according to a **proposal density** $g^{(t)}(\cdot | \theta^{(t)})$;
2. **randomly** decide whether to **adopt** θ^* or $\theta^{(t)}$;
 - ▶ $\theta^{(t+1)} = \theta^*$ with $p = \min \left(1, \exp \left\{ (f(\theta^{(t)}) - f(\theta^*)) / \tau_j \right\} \right)$;
 - ▶ $\theta^{(t+1)} = \theta^{(t)}$ otherwise;
3. Repeat steps 1 and 2 a total of m_j **times**.
4. Update $j = j + 1$, $\tau_j = \alpha(\tau_{j-1})$, $m_j = \beta(m_{j-1})$. Go to step 1

Simulated Annealing: algorithm

Note:

- $\alpha()$ should slowly decrease temperatures to zero;
- $\beta()$ should increase stage lengths as temperatures drop;
- θ^* always adopted when superior to $\theta^{(t)}$, sometimes anyway;
- *stochastic descent*, enable to escape from local minima.

Possible stopping criteria:

- reach the total number of iterations $\sum_j m_j$;
- reach the minimum temperature;
- complete the schedule of τ_j and m_j ;
- (relative or absolute) distance within the tolerance.

Simulated Annealing: Neighbourhoods and Proposals

Choosing the **neighbourhood**:

- ideally **small** and **easy to compute**;
 - ▶ e.g., in variable selection, add/remove a predictor;
- its structure must allow **all elements** of Θ to **communicate**,
 - ▶ i.e., any two candidate solutions must be **connectable via a sequence of neighbourhoods**.

Proposal distribution:

- **discrete uniform** over $\mathcal{N}(\theta^{(t)})$,
 - ▶ fast and simple solution.

Updating the **objective function**:

- to speed up the process, it is useful to find a **fast update**.

Simulated Annealing: Neighbourhoods and Proposals

Example (travelling salesman problem):

- each tour θ is a permutation of the cities $1, 2, \dots, p$;
- $\mathcal{N}(\theta^{(t)})$ is generated by removing two non-adjacent links and reconnecting the tour (by reversing the segments);
 - ▶ e.g., the tour 143256 is a neighbour of the tour 123456;
- two links are altered, the process of generating such neighbours is a 2-change, and it yields a 2-neighbourhood;
- $\mathcal{N}(\theta^{(t)})$ contains $p(p-3)/2$ unique 2-change neighbours
 - ▶ note that $p(p-3)/2 \ll (p-1)!/2$;
- updating the objective process can be speeded up:
 - ▶ new tour length = old tour length - distance of the broken segments + distance new segments.

Simulated Annealing: Cooling schedule and convergence

Cooling schedule: sequence of stage lengths m_j and τ_j ;

- Cooling should be **slow**!

Limiting behaviour:

- simulated annealing produces a **Markov** chain because $\theta^{(t+1)}$ depends only on $\theta^{(t)}$.
- 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**!

Simulated Annealing: theoretical considerations

Simulated annealing can be viewed as producing:

- a **sequence of homogeneous** Markov chains (one at each temperature);
- a **single inhomogeneous** Markov chain (with temperature decreasing between transitions).

These two views:

- lead to **different approaches** to defining limiting behaviour;
- **same conclusion**: the limiting distribution has support **only** on the global minima.

Simulated Annealing: theoretical considerations

To understand that cooling leads to convergence to global minima:

- first fix temperature at τ ;
- $\forall \theta_i, \theta_j \in \Theta$, suppose that proposing θ_i from $\mathcal{N}(\theta_j)$ has the same probability as proposing θ_j from $\mathcal{N}(\theta_i)$;

Then:

- the sequence of $\theta^{(t)}$ generated by simulated annealing is a Markov chain with stationary distribution

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

This approach is called **Metropolis algorithm**.

Simulated Annealing: convergence

Suppose:

- there are M global minima;
- the set of these solutions is $\mathcal{M}$;
- denote the minimal value of f on Θ as $f_{\min}$.

Then:

- the **stationary distribution** of the chain for a fixed τ is

$$\pi_{\tau}(\theta_i) = \frac{\exp\{-(f(\theta_i) - f_{\min})/\tau\}}{M + \sum_{j \notin \mathcal{M}} \exp\{-(f(\theta_j) - f_{\min})/\tau\}} \quad (1)$$

for each $\theta_i \in \Theta$;

- $\lim_{\tau \downarrow 0} \exp\{-(f(\theta_i) - f_{\min})/\tau\} = \begin{cases} 1 & \text{if } i \in \mathcal{M} \\ 0 & \text{otherwise.} \end{cases}$
- $\lim_{\tau \downarrow 0} \pi_{\tau}(\theta_i) = \begin{cases} 1/M & \text{if } i \in \mathcal{M} \\ 0 & \text{otherwise.} \end{cases}$

Simulated Annealing: Practical advice on cooling schedules:

Recall:

- $\tau_j = \alpha(\tau_{j-1})$ is the **temperature** at stage j ;
- $m_j = \beta(\tau_{j-1})$ is the **number of iterations** at stage j .

Possible approaches for setting a **cooling schedule**:

1. **set** $m_j = 1$ for all j and **reduce τ very slowly** according to $\alpha(\tau_{j-1}) = \frac{\tau_{j-1}}{1+a\tau_{j-1}}$ for a small value of a ;
 2. **set** $\alpha(\tau_{j-1}) = a\tau_{j-1}$ for $a < 1$ (usually $a \geq 0.9$). **Increase m_j as τ decreases**, e.g., $\beta(m_{j-1}) = bm_{j-1}$ for $b > 1$.
 3. use $\alpha(\tau_{j-1}) = \frac{\tau_{j-1}}{1+\tau_{j-1} \log\{1+r\}/(3s_{\tau_{j-1}})}$ where $s_{\tau_{j-1}}$ depends (in a complicated way) on f and r is a small real number.
- ... many other, see references in the book.

Simulated Annealing: Practical advice on cooling schedules:

Selection of the **initial** parameter values and schedule:

- **problem dependent**;
- try and fail.

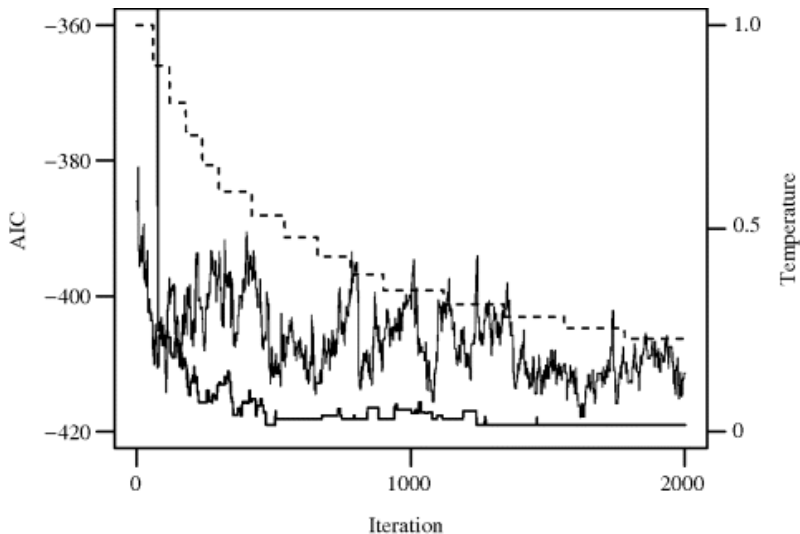
A **guideline** for the initial temperature:

- use $\tau_0 > 0$ so that $\exp \{(f(\boldsymbol{\theta}_i) - f(\boldsymbol{\theta}_j))/\tau_0\} \approx 1 \ \forall \boldsymbol{\theta}_i, \boldsymbol{\theta}_j \in \Theta$;
- reason: all points in the parameter space have a **reasonable chance of being visited** in early iterations;
- larger temperature drops **require** longer runs.

Empirical result:

- Long runs at high temperatures are **unneeded**;
- advise: decrease temperature **rapidly at first, slowly later**.

Simulated Annealing: Example



(Givens & Hoeting, 2013, figure 3.4)

Simulated Annealing: Example

In this example (variable selection on baseball salaries data):

- minimize AIC;
- **1-change** $\mathcal{N}(\theta^{(t)})$, add or remove one predictor;
- **discrete uniform** distribution for $\theta^{(t)} \in \mathcal{N}(\theta^{(t)})$;
- **15-stage cooling schedule**, with **stage lengths** of 5×60 , 5×120 , and 5×220 ;
- **cooling** by $\alpha(\tau_{j-1}) = 0.9\tau_{j-1}$ after each stage;

Results:

- bottom curve, $\tau_0 = 1$: sticking because **little tolerance** for uphill moves;
- top solid line, $\tau_0 = 10$: **considerable mixing**, with many uphill proposals accepted as moves.

Simulated Annealing: Enhancements

Possible improvements:

- employ multiple random starts;
- avoid constraints in Θ , use penalties;
- adaptively restrict neighbourhoods at each iteration;
- approximate f to ensure quick evaluation;
- introduce penalties to discourage revisit states recently visited;
- use other acceptance distributions than the Boltzmann one;
- use reheating strategies at low temperatures;
- ...

Genetic Algorithms: Introduction

Genetic algorithms:

- mimic the process of Darwinian **natural selection**;
- candidate solutions are **organisms** with their **genetic code**;
- **fitness** of an organism as quality of a candidate solution;
- **breeding** happens:
 - ▶ among **highly fit** organisms to provide the **best** opportunity to pass desirable attributes to future generations;
 - ▶ 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.

Genetic Algorithms: Definitions

Based on the **biological analogy**:

- every **candidate solution** is an individual or **organism**;
 - every organism is completely described by its **chromosome**;
 - a chromosome is a sequence of C **genes**;
 - the genes' **possible values** are called **alleles**;
 - all possible alleles form an **alphabet**;
 - the **position** of a gene in the chromosome is its **locus**.
-
- Example of chromosome: '100110001' (regression example).

Genetic Algorithms: Definitions

Based on the **biological analogy**:

- The information encoded in an individual's chromosome is its **genotype**, ϑ ;
- The expression of the genotype in the organism itself is its **phenotype**, θ
- GA's consider **many** candidate solutions **simultaneously**;
- the t -th iteration (**generation**) consists of
 - ▶ a collection of P organisms $\theta_1^{(t)}, \dots, \theta_P^{(t)}$;
 - ▶ with corresponding **genotypes** (encodings) $\vartheta_1^{(t)}, \dots, \vartheta_P^{(t)}$.

Genetic Algorithms: Definitions

Based on the **biological analogy**:

- $f(\theta_i^{(t)})$ is the **fitness** of an organism (genotype);
- **natural selection** favours organisms with **high** fitness;
 - ▶ as generations progress, organisms **inherit from their parents** bits of genetic code that are associated with high fitness;
- 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 those of two **parent** chromosomes belonging to the t -th generation.

Genetic Algorithms: Example

Consider a regression model selection problem:

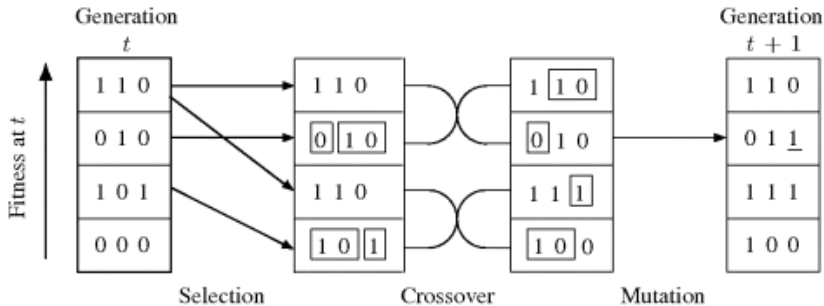
- assume 9 predictors (**genes**);
- the **alleles** are 0 (exclusion) and 1 (inclusion);
- the **genotype** $\mathbf{g}_i^{(t)} = '100110001'$ corresponds to the **phenotype** of a model containing only predictors 1, 4, 5, and 9;
- the **fitness** of this individual equals the negative AIC.

Genetic Algorithms: Selection mechanisms

Breeding drives most genetic change. It consists of:

- **selection mechanism:** two parents are chosen to breed,
 - ▶ e.g., select one parent (or both) with probability proportional to fitness and the other completely at random.
- **breeding:** combine their chromosomes to produce offspring,
 - ▶ **Crossover**,
 - ▶ select a random position between two adjacent loci;
 - ▶ split both parent chromosomes at this position;
 - ▶ glue the left chromosome segment from one parent to the right segment from the other parent;
 - ▶ the remaining segments form a second offspring or discarded;
 - ▶ **Mutation**,
 - ▶ randomly change one or more alleles in the chromosome;
 - ▶ usually applied after crossover;
 - ▶ **Permutation**,
 - ▶ require more sophisticated crossover and mutation operators.
- the new offspring become members of generation $t + 1$.

Genetic Algorithms: example



(Givens & Hoeting, 2013, figure 3.5)

Genetic Algorithms: Initialization, Termination, and Parameter Values

Initial population: purely **random** chromosomes.

Generation size (denoted by **P**):

- larger **P** provides **greater search diversity**;
- larger **P** provides **greater computing burden**;
- for binary encoding, the suggestion is $C \leq P \leq 2C$,
 - where C is the chromosome length;
- for permutation chromosomes, $2C \leq P \leq 20C$.
- real applications often have $10 \leq P \leq 200$.

Mutation rates:

- typically **very low**, around 1%.
- suggestions include $1/C$ or $\propto 1/(P\sqrt{C})$.

Genetic Algorithms: Example

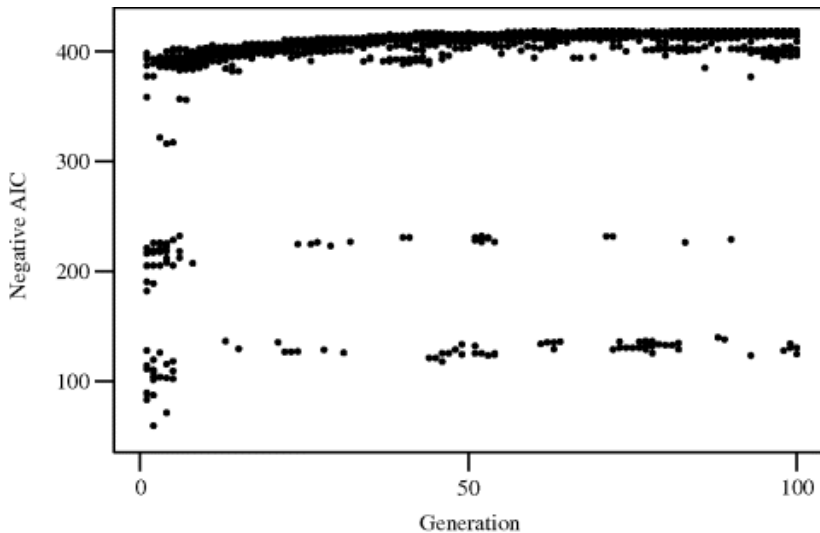
Apply a **genetic algorithm** for regression model selection:

- baseball data;
- 100 generations of size $P = 20$;
- using rank-based fitness, simple crossover, and 1% mutation;

Note:

- Darwinian '**survival of the fittest**' is clearly seen;
- the twenty random starting individuals quickly coalesce into three effective subspecies;
- the best of these slowly overwhelming the rest;
- the best model was first found in generation 87.

Genetic Algorithms: Example



(Givens & Hoeting, 2013, figure 3.6)

Genetic Algorithms: Fitness

Consider the **fitness of an organism**.

Naive option: equate the **objective function value** $f(\theta)$

- may lead to **fast** convergence **to a poor local optimum**;
- **a few** of the very best individuals **dominate** the breeding and their offspring saturate subsequent generations;
- **lack the genetic diversity** needed to produce offspring that might typify other, more profitable regions of solution space;
- this problem is especially troublesome **in early stages**;
- other choices yield superior optimization performance.

Genetic Algorithms: Fitness

Common approach: ignore the values and use only their **ranks**,

- select with probability **proportional to**

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

where r_i is the rank of $f(\theta_i^{(t)})$ among generation t .

- gives selection probabilities:
 - ▶ $1/P$ for the **median** quality candidate;
 - ▶ $2/(P+1)$ (roughly double) for the **best** chromosome;
- important to maintain **firm selective pressure** without allowing a few individuals to cause premature convergence;
- selective pressure must be **balanced carefully**, as genetic algorithms can be slow to find a good optimum.

Genetic Algorithms: Selection Mechanisms

Selecting parents for breeding (**selection mechanism**):

- base the selection on **fitness ranks** is far more common than using selection probabilities proportional to fitness;
- another common approach is **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;
 - ▶ **randomly pair** parents for breeding;
- in this way:
 - ▶ the **best** individual will breed P times;
 - ▶ the **median** individual will breed once on average;
 - ▶ the **worst** individual will not breed at all.

Genetic Algorithms: Updating Generations

Populations can be **partially** updated:

- define the **generation gap**, G , the **proportion** of the generation to be replaced by generated offspring,
 - ▶ $G = 1$ implies **distinct, non-overlapping** generations;
 - ▶ $G = 1/P$ corresponds to update **one offspring at a time**;
 - ▶ replaces the **least fit** (or some random unfit) individual;
 - ▶ **more variance and higher selective pressure** than standard GA;
 - ▶ ensuring the survival of the **best** current solution (any $G < 1$);
- one should **not allow duplicate** individuals in the population:
 - ▶ duplicate individuals wastes **computing resources**;
 - ▶ **distorts** the parent **selection** criterion (duplicated chromosomes may have more chances to produce offspring).

Genetic Algorithms: Genetic Operators

Other possibilities to **increase genetic mixing**:

- choose **more than one** crossover point,
 - ▶ the gene sequence between them are swapped between parents;
- at **each loci** randomly choose an allele from those expressed in that position in the parents,
 - ▶ the parental origins of adjacent genes could be **independent**;
 - ▶ or **correlated**, with strength of correlation controlling the degree to which offspring **resemble** a single parent;
- use allele alphabets with **more than two** elements,
 - ▶ **messy genetic algorithms** employ variable-length encoding with genetic operators that adapt to changing length.

Genetic Algorithms: Permutation chromosomes

We mentioned permutation to make genetic changes:

- **mutation is easy**, swap two random alleles;
- **crossover is hard**:
 - ▶ let $p = 9$ and consider crossover for the two parent chromosomes '752631948' and '912386754';
 - ▶ let the crossover point fall between the 3rd and 4th loci, splitting the chromosomes as '752|631948' and '912|386754'.
 - ▶ ordinary crossover produces '752386754' and '912631948';
 - ▶ both are **invalid permutation**: contain duplicate alleles.

Genetic Algorithms: Order crossover

Instead, use **order crossover**:

- a **random collection of loci** is chosen,
 - ▶ e.g., for the two parent chromosomes '752631948' and '912386754', the 4th, 6th, and 7th loci;
- consider the **alleles** in these loci in the first parent,
 - ▶ in the example, 6, 1, and 9;
- **rearrange** the alleles in the second parent to impose this order
 - ▶ the remaining alleles in the second parent are '**238*754';
 - ▶ inserting 6, 1, and 9 in this order yields '612389754';
- the parents can be switched to produce a second offspring.

Genetic Algorithms: Edge recombination crossover

Or **edge recombination crossover** (here using the travelling salesman analogy):

1. construct a table that stores **all the links** that lead into and out of each city in either parent;
2. choose the **initial city** from one parent,
 - ▶ favouring the one with fewer links;
3. note the **onward links** from this city;
4. **delete** the chosen city everywhere in the table;
5. **select** from among the choices in step 3,
 - ▶ favouring the choice with fewest onward links;
6. **update the table** again and continue in this fashion.

Genetic Algorithms: Example

Example for parents '752631948' and '912386754'.

Step 1		Step 2		Step 3	
City	Links	City	Links	City	Links
1	3, 9, 2	1	3, 2	1	3, 2
2	5, 6, 1, 3	2	5, 6, 1, 3	2	5, 6, 1, 3
3	6, 1, 2, 8	3	6, 1, 2, 8	3	6, 1, 2, 8
4	9, 8, 5	4	8, 5	4	Used
5	7, 2, 4	5	7, 2, 4	5	7, 2
6	2, 3, 8, 7	6	2, 3, 8, 7	6	2, 3, 8, 7
7	8, 5, 6	7	8, 5, 6	7	8, 5, 6
8	4, 7, 3, 6	8	4, 7, 3, 6	8	7, 3, 6
9	1, 4	9	Used	9	Used
'9*****'		'94*****'		'945*****'	

Genetic Algorithms: Initialization and Parameter Values

Possible improvements:

- heuristic initialization strategies to improve random starts;
- P may be decreased progressively as iterations continue,
 - ▶ equal sizes for subsequent generations are not required;
 - ▶ use a somewhat large generation size P for early generations,
 - ▶ discourage early convergence and promote search diversity;
 - ▶ if P is too large, the algorithm becomes too slow;
 - ▶ rank-based selection mechanisms are more common;
- allow probabilities of mutation, crossover and other parameters of the genetic algorithm to vary adaptively over time,
 - ▶ promote search diversity as generations become less diverse;
 - ▶ e.g., mutation rate inversely proportional to the diversity.

References |

GIVENS, G. H. & HOETING, J. A. (2013). *Computational Statistics*. Wiley.

WATNIK, M. R. (1998). Pay for play: Are baseball salaries based on performance? *Journal of Statistics Education* **6**.