



STK4051/STK9051

Computational Statistics

Riccardo De Bin

`debin@math.uio.no`

Outline of the lecture

- Combinatorial Optimization
 - Genetic Algorithms
 - Definitions and the Canonical Algorithm
 - Variations
 - Initialization and Parameter Values
 - Tabu Algorithms
 - Basic Definitions
 - The Tabu List
 - Aspiration Criteria
 - Diversification
 - Intensification
 - Comprehensive Tabu Algorithm

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:

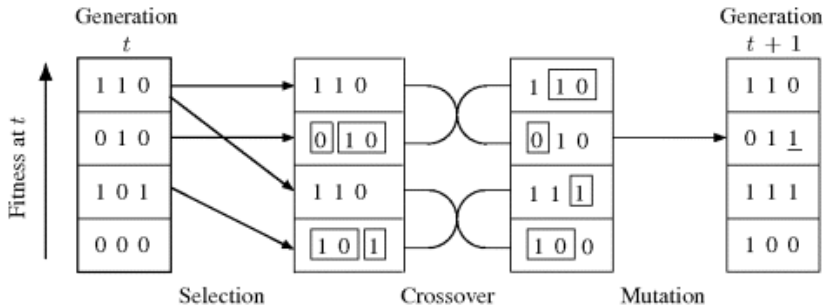
- assume 9 predictors (**genes**);
- the **alleles** are 0 (exclusion) and 1 (inclusion);
- the **genotype** $\mathbf{g}_i^{(t)} = '100110001'$ corresponds to the **phenotype** of a model containing only predictors 1, 4, 5, and 9;
- the **fitness** of this individual equals the negative AIC.

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

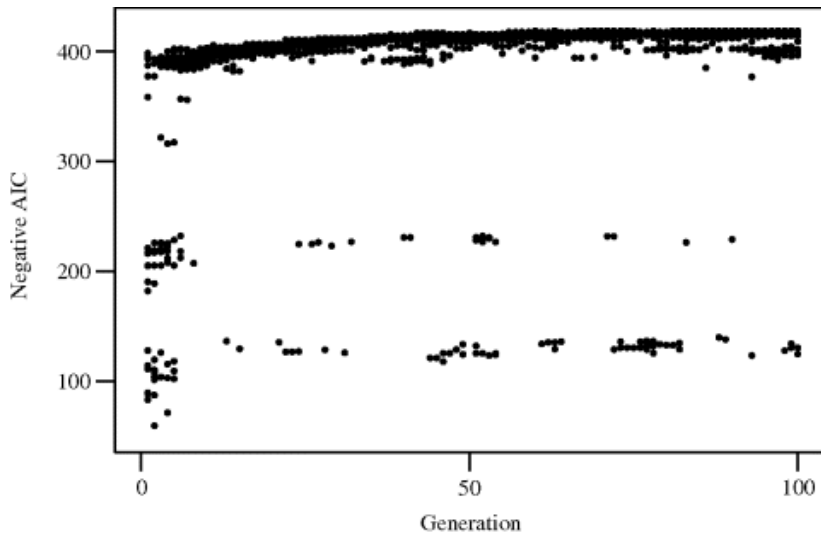
Apply a **genetic algorithm** for regression model selection:

- 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.

Tabu Algorithms: Introduction

Basic idea: local search algorithm with a set of additional rules,

- that guide the selection of moves to promote the discovery of the global maximum;
- the neighbourhoods change at each iteration;
- to contrast the tendency of standard ascent algorithms to end up in globally uncompetitive local maxima.

There are four general types of rules:

1. temporarily forbid certain potential moves (make them **tabu**);
2. aspiration to better solutions;
3. intensification of search in promising areas of solution space;
4. diversification to broaden the exploration of the solution space.

Tabu Algorithms: Introduction

Tabu search:

- allows downhill moves when no uphill move can be found;
- in this way it is possible to escape from a local minimum;
- need to ensure that the next step (or a future one) does not simply reverse the downhill move;
 - ▶ otherwise no real long-term benefit of the downhill move;
- certain moves are temporarily forbidden,
 - ▶ to prevent such cycling;
 - ▶ based on the recent history of the algorithm;
- steepest ascent/mildest descent:
 - ▶ early form of tabu search;
 - ▶ move to the least unfavourable neighbour when no uphill move.

Tabu Algorithms: Basic definitions

Tabu search is an **iterative algorithm**:

- **initiated** at time $t = 0$ with candidate solution $\theta^{(0)}$;
- at the t -th iteration, a **new candidate** solution $\theta^{(t+1)}$ is selected from a **neighbourhood** of $\theta^{(t)}$;
- the neighbourhood $\mathcal{N}(\theta^{(t)}, H^{(t)})$,
 - ▶ depends on the **search history**, $H^{(t)}$, of the algorithm;
 - ▶ it suffices for $H^{(t)}$ to be a **selective** history;
 - ▶ remembering only matters **necessary** for the future operations.
- the identification of the preferred $\theta^{(t+1)}$ in $\mathcal{N}(\theta^{(t)}, H^{(t)})$ may depend not only on f but also on the search history,
 - ▶ use an **augmented objective function**, $f_{H^{(t)}}$.

Tabu Algorithms: Move attributes

A step from $\theta^{(t)}$ to $\theta^{(t+1)}$ is characterized by **attributes**:

- used to describe **moves** or **types of moves** that will be forbidden, encouraged, or discouraged in future iterations;
- denote the a -th attribute as A_a ;
- the **negation** of an attribute is also an attribute,
 - ▶ e.g., if A_a swaps the values of $\theta_i^{(t)}$ and $\theta_j^{(t+1)}$, then $\bar{A}_a$ corresponds to not making that swap.
- as the algorithm progresses, the attributes **vary**;
- the **recency** of an attribute is the number of **steps passed since** a move had that attribute,
 - ▶ $R(A_a, H^{(t)}) = 0$ if A_a is expressed in the move yielding $\theta^{(t)}$;
 - ▶ $R(A_a, H^{(t)}) = 1$ if most recently in the move yielding $\theta^{(t-1)}$;
 - ▶ and so forth ...

Tabu Algorithms: Examples of attributes

Attribute	Model Selection Example
A change in the value of $\theta_i^{(t)}$. The attribute may be the value from which the move began, or the value to which it arrived.	A_1 : Whether the i th predictor is added (or deleted) from the model.
A swap in the values of $\theta_i^{(t)}$ and $\theta_j^{(t)}$ when $\theta_i^{(t)} \neq \theta_j^{(t)}$.	A_2 : Whether the absent variable is exchanged for the variable present in the model.
A change in the value of f resulting from the step, $f(\boldsymbol{\theta}^{(t+1)}) - f(\boldsymbol{\theta}^{(t)})$.	A_3 : The reduction in AIC achieved by the move.

Tabu Algorithms: Examples of attributes

Attribute	Model Selection Example
The value of some other strategically chosen function, g , i.e., $g(\boldsymbol{\theta}^{(t+1)})$.	A_4 : The number of predictors in the new model.
A change in the value of g resulting from the step, $g(\boldsymbol{\theta}^{(t+1)}) - g(\boldsymbol{\theta}^{(t)})$.	A_5 : The change in a different variable selection criterion such as Mallows' C_p or adjusted R^2 .

Tabu Algorithms: The tabu list

The **tabu list** temporarily forbids some types of moves:

- avoiding cycling (coming back to the previous candidate);
- the tabu list is a list of attributes, not moves,
 - ▶ a single tabu attribute may forbid entire classes of moves;
- moves with attributed on the tabu list are excluded from the neighbourhood, $\mathcal{N}(\theta^{(t)}, H^{(t)}) = \left\{ \theta : \theta \in \mathcal{N}(\theta^{(t)}) \text{ and no attribute of } \theta \text{ is currently tabu} \right\};$
- each time a move with attribute A_a is taken, $\overline{A}_a$ is put there;
- it lasts for τ iterations;

Tabu Algorithms: The tabu list

The tabu **expires** when $R(A_a, H^{(t)})$ first equals τ ,

- $\bar{A}_a$ is **removed** from the tabu list;
- by that time, the candidate solution should have **changed enough** that reversing the move is not longer problematic;

The parameter τ is called **tabu tenure**:

- can be fixed or vary, randomly or systematically,
 - ▶ if systematically, usually **depending on** the attribute;
 - ▶ attributes affection **more moves, shorter τ** ;
- it should be **long enough** to prevent cycling;
- but **short enough** to not deteriorate the search;
- **recommended** values: $7 \leq \tau \leq 20$, or $0.5\sqrt{p} \leq \tau \leq 2\sqrt{p}$.

Tabu Algorithms: Aspiration

One wants to **override** the tabu when it is wise:

- permit a tabu move if it provides the **highest** $f_{H^{(t)}}$ so far,
 - ▶ e.g., swapping two components of θ ;
 - ▶ further swaps are in the tabu list;
 - ▶ the candidate solutions have drifted away from the region of solution space being explored when the tabu began;
 - ▶ another swap may lead to a drastic increase in $f_{H^{(t)}}$;
- **“aspiration by influence”**: overrides the tabu of reversing a **low-influence move** if a **high-influence move** is made before,
 - ▶ the recent high-influence step may have moved search to a new region where further local exploration is useful.
- **aspiration criteria** can be used to **encourage non-tabu moves**,
 - ▶ e.g., downweight low-influence to favour high-influence moves;
 - ▶ incorporate penalties in $f_{H^{(t)}}$.

Tabu Algorithms: Diversification

Diversification:

- rules based on how often an attribute is observed to **increase the diversity** of the candidate solutions;
- **frequencies** over the entire history or the last ψ moves;
- **residence frequency**: A_a corresponds to a **feature** of $\theta^{(t)}$;
 - ▶ e.g., the inclusion of the predictor x_i in the regression model;
- **transition frequency**: A_a corresponds to a **change induced** by moving from one candidate solution to another;
 - ▶ e.g., attribute that signal when a move reduced the AIC;
- **high residence** frequencies can be associated to:
 - ▶ high-quality solutions, if happening in nearly optimal regions;
 - ▶ bad solutions, when getting stuck in a low-quality region;
- **high transition** frequencies may be “crack fillers”:
 - ▶ can fine tune good solutions but no major improvement;

Tabu Algorithms: Diversification

An approach employing frequency to increase search diversification is to **incorporate a penalty or incentive** function in $f_{H^{(t)}}$,

$$f_{H^{(t)}}(\boldsymbol{\theta}^{(t+1)}) = \begin{cases} f(\boldsymbol{\theta}^{(t+1)}) & \text{if } f(\boldsymbol{\theta}^{(t+1)}) \geq f(\boldsymbol{\theta}^{(t)}) \\ f(\boldsymbol{\theta}^{(t+1)}) - cF(A_a, H^{(t)}) & \text{if } f(\boldsymbol{\theta}^{(t+1)}) < f(\boldsymbol{\theta}^{(t)}) \end{cases}$$

with $c > 0$ and $F(A_a, H^{(t)})$ a frequency function, for example the average count of occurrences of the a -th attribute thus far.

Note:

- if all non-tabu moves are downhill, then this approach **discourages** moves that have the high-frequency attribute A_a ;
- an analogous strategy can be crafted for **uphill moves**.

Tabu Algorithms: Intensification

Intensification:

- use frequencies to **guide the search effort** in particular areas;
- construct $f_{H(t)}$ to:
 - ▶ **reward retention** of attributes that correspond to high-quality high-frequency solutions;
 - ▶ **penalize removal** of such attributes.

Tabu Algorithms: Comprehensive algorithm

After **initialization** and identification of a list of problem-specific attributes, the algorithm proceeds as follows:

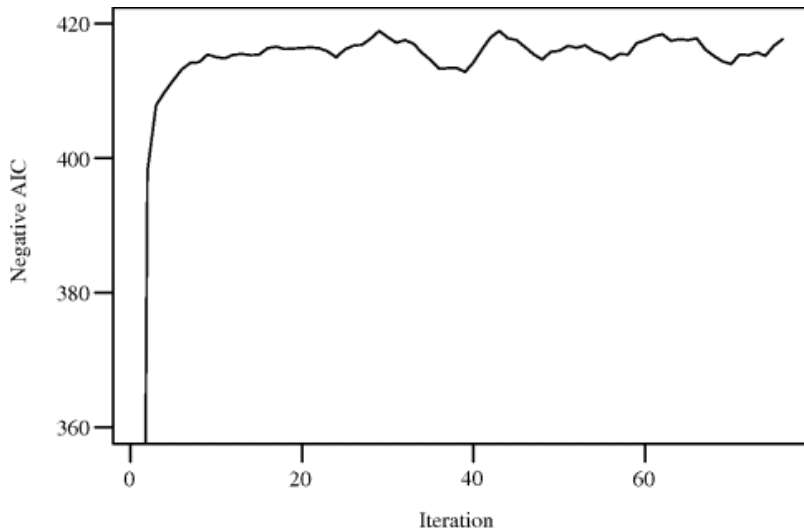
1. construct $f_{H^{(t)}}(\boldsymbol{\theta})$ based on $f(\boldsymbol{\theta})$, and perhaps diversification/intensification penalties or incentives;
2. identify neighbours of $\boldsymbol{\theta}^{(t)}$, namely the members of $\mathcal{N}(\boldsymbol{\theta}^{(t)})$;
3. rank neighbours in decreasing order of improvement, by $f_{H^{(t)}}$;
4. select the highest ranking neighbour;
5. is this neighbour on the tabu list? If not, go to step 8;
6. does it pass an aspiration criterion? If so, go to step 8;
7. if no neighbour of $\boldsymbol{\theta}^{(t)}$ has been adopted as $\boldsymbol{\theta}^{(t+1)}$, then stop. Otherwise, select the next best neighbor and go to step 5;
8. Adopt this solution as $\boldsymbol{\theta}^{(t+1)}$
9. update the tabu list;
10. has a stopping criterion been met? If so, stop. Otherwise, increment t and go to step 1.

Tabu Algorithms: Example

Apply a **simple tabu search** to the variable selection problem:

- regression modelling of the baseball data;
- only **attributes** signalling presence/absence of each predictor;
- set $\tau = 5$ moves;
- the algorithm was run for 75 moves from a **random start**;
- **aspiration criterion** permitted an otherwise tabu move if it yielded an objective function value above the best seen.

Tabu Algorithms: Example



(Givens & Hoeting, 2013, figure 3.7)

References |

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