Heuristic Algorithms

Master's Degree in Computer Science/Mathematics

Roberto Cordone
DI - Università degli Studi di Milano

Schedule:	Wednesday 13.30 - 16.30 in classroom Alfa
	Thursday 09.30 - 12.30 in classroom Alfa
Office hours:	on appointment
E-mail:	roberto.cordone@unimi.it
Web page:	https://homes.di.unimi.it/cordone/courses/2026-ae/2026-ae.html
Ariel site:	https://myariel.unimi.it/course/view.php?id=7439

Recombination heuristics

Constructive and exchange heuristics manage one solution at a time
(except for the *Ant System*)

Recombination heuristics manage several solutions in parallel

- start from a set (population) of solutions (individuals) obtained somehow
- recombine the individuals generating a new population

Their original aspect is the use of operations working on several solutions,
but they often include features of other approaches (sometimes renamed)

Some are nearly or fully deterministic

- Scatter Search
- Path Relinking

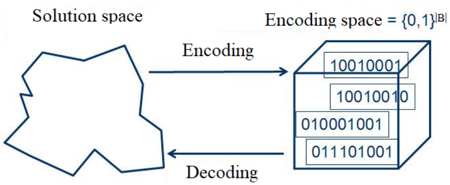
others are strongly randomised (often based on biological metaphors)

- genetic algorithms
- memetic algorithms
- evolution strategies

Of course the effectiveness of a method does not depend on the metaphor

Encoding-based algorithms

Many recombination heuristics define and manipulate encodings of the solutions (i.e., compact representations), rather than the solutions



The aims of this approach are

- abstraction: conceptually distinguishing the method from the problem to which it is applied
- generality: build operators effective on every problem represented with a given family of encodings

In a strict sense, every representation of a solution in memory is an encoding: the term "encoding" tends to be used for the more involved and compact ones

The difference is blurred

Genetic algorithm

The genetic algorithm, proposed by Holland (1975), is the most famous

It builds and encodes a population $X^{(0)}$, and repeatedly applies:

1. selection: generate a new population starting from the current one
2. crossover: recombine subsets of two or more individuals
3. mutation: modify the individuals

Algorithm GeneticAlgorithm$(I, X^{(0)})$

$\Xi := Encode(X^{(0)})$; $x^* := \arg\min_{x \in X^{(0)}} f(x)$; { Best solution found so far }

For $g = 1$ *to* n_g *do*

$\Xi := Selection(\Xi)$;

$\Xi := Crossover(\Xi)$;

$x_c := \arg\min_{\xi \in \Xi} f(x(\xi))$;

If $f(x_c) < f(x^*)$ *then* $x^* := x_c$;

$\Xi := Mutation(\Xi)$;

$x_m := \arg\min_{\xi \in \Xi} f(x(\xi))$;

If $f(x_m) < f(x^*)$ *then* $x^* := x_m$;

EndFor;

Return $(x^*, f(x^*))$;

Features of a good encoding

The performance of a genetic algorithm depends on the encoding

The following properties should be satisfied (with decreasing importance)

1. each solution should have an encoding, except for dominated ones; otherwise, there would be unreachable solutions
2. different solutions should have different encodings (or the best solution with a given encoding should be easy to find); otherwise, there would be unreachable solutions
3. each encoding should correspond to a feasible solution; otherwise, the population would include useless individuals
4. each solution should correspond to the same number of encodings; otherwise, some solutions would be unduly favoured
5. the encoding and decoding operations should be efficient, otherwise, the algorithm would be inefficient
6. locality: small changes to the encoding should induce small changes to the solution, otherwise intensification and diversification would be impossible

These conditions depend very much on the constraints of the problem

(*so much for abstraction...*)

Feasible and unfeasible encodings

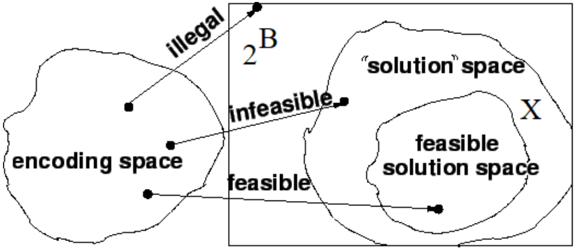
Mutation and crossover operators easily produce unfeasible subsets
if property 3 is not satisfied; this may imply the violation of

1. quantitative constraints (e.g., a capacity is exceeded)
2. structural constraints (e.g., the solution is not made of circuits)

The distinction is conventional

The second kind of infeasibility is harder to repair, because
it concerns constraints that interact more strongly with each other

Some encodings guarantee structural (though not quantitative) feasibility



Encodings: the incidence vector

The most direct encoding for Combinatorial Optimisation problems is the binary incidence vector $\xi \in \mathbb{B}^{|B|}$

$$\begin{cases} \xi_i = 1 \text{ indicates that } i \in x \\ \xi_i = 0 \text{ indicates that } i \notin x \end{cases}$$

A generic binary vector corresponds

- in the *KP* to a set of objects: its weight could be excessive
- in the *SCP* to a set of columns: it could leave uncovered rows
- in the *PMSP* and in the *BPP* to a set of assignments of tasks (objects) to machines (containers): it could make zero or more assignments for an element; in the *BPP*, it could violate the capacity of some container;
- in the *TSP* to a set of arcs: it could not form a Hamiltonian circuit
- in the *CMSTP* (*VRP*) to a set of edges (arcs): it could not form a tree (set of cycles), or exceed the capacity of the subtrees (circuits)

Encodings: symbolic strings

If the ground set is partitioned into components

(*objects, tasks, Boolean variables, vertices, nodes...*)

$$B = \bigcup_{c \in C} B_c \quad \text{with } B_c \cap B_{c'} = \emptyset \text{ for each } c \neq c'$$

and the feasible solutions contain one element of each component

$$|x \cap B_c| = 1 \text{ for each } c$$

one can

- define for each $c \in C$ an alphabet of symbols describing component B_c
- encode the solution into a string of symbols $\xi \in B_1 \times \ldots B_{|C|}$

$$\xi_c = \alpha \text{ indicates that } x \cap B_c = \{(c, \alpha)\}$$

Examples of encodings:

- *Max-SAT*: a string of n Boolean values, one for each logical variable
- *PMSP*: a string of machine labels, one for each task
- *BPP/CMSTP*: a string of container/subtree labels, one for each object/vertex:
  - the structural constraint on object assignment is enforced
  - the quantitative constraint on capacity is neglected
- for the *VRP*, a string of vehicle labels, one for each node (but capacity is neglected and decoding the circuit for each vehicle is an $\mathcal{NP}$-complete problem)
- the solutions of the *TSP*, the *KP*, the *SCP* are not partitions

Encodings: permutations of a set

A common encoding is given by the permutations of a set

- if the solutions are permutations, this is the natural encoding
  (*TSP* solutions are subsets of arcs, but also permutations of nodes)
- if the solutions are partitions and the objective is additive on the subsets, the *order-first split-second* method transforms permutations into partitions
  (*but solutions and encodings do not correspond one-to-one*!)
- if the problem admits a constructive algorithm that at each step
  1. chooses an element
  2. chooses how to add it to the solution (if many ways exist)

  we can feed elements to the algorithm following the permutation
  (*depending on step* 2*, some solutions could have no encoding*)

Selection

At each generation g a new population $\Xi^{(g)}$ is built extracting $n_p = |\Xi^{(g)}|$ individuals from the current population $\Xi^{(g-1)}$

$$\Xi^{(g)} := \text{Selection}\left(\Xi^{(g-1)}\right);$$

The extraction follows two fundamental criteria

1. an individual can be extracted more than once
2. better individuals are extracted with higher probability

$$\varphi(\xi) > \varphi(\xi') \Rightarrow \pi_\xi \geq \pi_{\xi'}$$

where the fitness $\varphi(\xi)$ is a measure of the quality of individual ξ

- for a maximisation problem, commonly

$$\varphi(\xi) = f(x(\xi))$$

- for a minimisation problem, commonly

$$\varphi(\xi) = UB - f(x(\xi))$$

where $UB \geq f^*$ is a suitable upper bound on the optimum

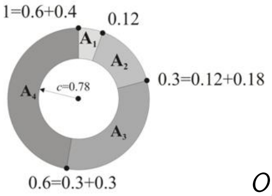
Proportional selection

The original scheme proposed by Holland (1975) assumed a probability proportional to fitness

$$\pi_\xi = \frac{\varphi(\xi)}{\sum_{\xi \in \Xi} \varphi(\xi)}$$

This is named roulette-wheel selection or spinning wheel selection:

- given the fitness for $\xi_i \in \Xi$ $(i = 1, \ldots, n_p)$
- build the intervals $\Gamma_i = \left(\sum_{k=1}^{i-1} \pi_{\xi_i} ; \sum_{k=1}^{i} \pi_{\xi_i} \right]$ in $O(n_p)$ time
- extract a random number $r \in U(0; 1]$
- choose individual i^* such that $r \in \Gamma_{i^*}$ in $O(\log n_p)$ time each



Overall $O(n_p \log n_p)$ *time*

Rank selection

The proportional selection suffers from

- stagnation: in the long term, all individuals tend to have a good fitness, and therefore similar selection probabilities
- premature convergence: if the best individuals are bad and the other ones very bad, the selection quickly generates a bad population

To overcome these limitations, one should at the same time

- assign different probabilities to the individuals
- limit the difference of probability among the individuals

The rank selection method

- sorts the individuals by nondecreasing fitness

$$\Xi^{(g)} = \{\xi_1, \ldots, \xi_{n_p}\} \text{ with } \varphi_{\xi_1} \leq \ldots \leq \varphi_{\xi_{n_p}}$$

- assigns to the k-th individual a probability equal to

$$\pi_{\xi_k} = \frac{k}{\sum_{k=1}^{n_p} k} = \frac{2k}{n_p (n_p - 1)}$$

It can be done in $O(n_p \log n_p)$ time as in the previous case

Tournament selection

An efficient compromise consists in

- extracting n_p random subsets $\Xi_1, \ldots, \Xi_{n_p}$ of size α
- selecting the best individual from each subset

$$\xi_k := \arg\max_{\xi \in \Xi_k} \varphi(\xi) \qquad k = 1, \ldots, n_p$$

in time $O(n_p \alpha)$

Parameter α tunes the strength of the selection:

- $\alpha \approx n_p$ favours the best individuals
- $\alpha \approx 2$ leaves chances to the bad individuals

All selection procedures admit an elitist variant, which
includes in the new population the best individual of the current one

(*always keep the best individual found so far*)