Heuristic Algorithms

Master's Degree in Computer Science/Mathematics

Roberto Cordone

DI - Università degli Studi di Milano

Schedule:	Thursday 14.30 - 16.30 in classroom 303
	Friday 14.30 - 16.30 in classroom 303
Office hours:	on appointment
E-mail:	roberto.cordone@unimi.it
Web page:	https://homes.di.unimi.it/cordone/courses/2022-ae/2022-ae.html
Ariel site:	https://rcordoneha.ariel.ctu.unimi.it

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

General scheme

The basic idea is that

- good solutions share components with the global optimum
- different solutions can share different components
- combining different solutions, it is possible to merge optimal components more easily than building them step by step

The typical scheme of recombination heuristics is

- build a starting population of solutions
- as long as a suitable termination condition does not hold
- at each iteration (generation) update the population
  - extract single individuals and apply exchange operations to them
  - extract subsets of individuals (usually, pairs) and apply recombination operations to them
  - collect the individuals thus generated and choose whether to accept or not each of them and how many copies into the new population

Scatter Search

Scatter Search (*SS*), proposed by Glover (1977), proceeds as follows

1. generate a starting population of solutions
2. improve all of them with an exchange procedure
3. build a *reference set* $R = B \cup D$ where
   - subset B includes the best known solutions
   - subset D includes the "farthest" solutions (from B and each other) (this requires a distance definition, e.g. the Hamming distance)
4. for each pair of solutions $(x, y) \in B \times (B \cup D)$
   - "recombine" x and y, generating z
   - improve z obtaining z' with an exchange procedure
   - if $z' \notin B$ and B contains a worse solution, replace it with z' (we want no duplicates in the reference set)
   - if $z' \notin D$ and D includes a closer solution, replace it with z' (we want no duplicates in the reference set)
5. terminate when R is unchanged

The rationale is that

- the recombinations in $B \times B$ intensify the search
- the recombinations in $B \times D$ diversify the search

General scheme of the *Scatter Search* approach

Algorithm ScatterSearch(I, P, n_B, n_D)

$B := \emptyset$; $D := \emptyset$;

Repeat

Stop $=$ *true*;

For each $x \in P$ *do*

$z :=$ SteepestDescent(I, x); *If* $f(z) < f(x^*)$ *then* $x^* := z$;

$y_B := \arg\max_{y \in B} f(y)$; $y_D := \arg\min_{y \in D} d(y, B \cup D \setminus \{y\})$;

If $z \notin B$ *and* $(|B| < n_B$ *or* $f(z) < f(y_B))$ *then*

{ B keeps the n_B best unique solutions }

$B := B \cup \{z\}$; Stop $:=$ *false*; *If* $|B| > n_B$ *then* $B := B \setminus \{y_B\}$;

ElseIf $z \notin D$ *and* $(|D| < n_D$ *or* $d(z, B \cup D \setminus \{y_D\}) > d(y_D, B \cup D \setminus \{y_D\}))$ *then*

{ D keeps the n_D most diverse unique solutions }

$D := D \cup \{z\}$; Stop $:=$ *false*; *If* $|D| > n_D$ *then* $D := D \setminus \{y_D\}$;

EndIf

EndFor

$P := \emptyset$;

For each $(x, y) \in B \times (B \cup D)$ *do* { Recombine to build the new population }

$P := P \cup$ Recombine(x, y, I);

EndFor

until Stop $=$ *true*;

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

Recombination procedure

The recombination procedure depends on the problem

Usually, solutions x and y are manipulated as subsets

1. include in z all the elements shared by x and y:

$$z := x \cap y$$

(both solutions concur in suggesting those elements)

2. augment solution z adding elements from $x \setminus z$ or $y \setminus z$
    - chosen at random or with a greedy selection criterium
    - alternatively from each source or freely from the two sources

(this is similar to a restricted constructive heuristic)

3. if necessary, add external elements from $B \setminus (x \cup y)$
4. if subset z is unfeasible, apply an auxiliary exchange heuristic to make it feasible (repair procedure)

Examples

MDP

- start with $z := x \cap y$
- augment z with $k - |z|$ random or greedy points from $x \setminus z$ or $y \setminus z$
- no repair procedure is required

Max-SAT

- start with $z := x \cap y$
- augment z with $n - |z|$ random or greedy truth assignments from $x \setminus z$ or $y \setminus z$
- no repair procedure is required

Examples

KP

- start with $z := x \cap y$
- augment z with random or greedy elements from $x \setminus z$ or $y \setminus z$ respecting the capacity
- no repair procedure is required
- the solution could be augmented with elements from $B \setminus (x \cup y)$

SCP

- start with $z := x \cap y$
- augment z with random or greedy columns from $x \setminus z$ or $y \setminus z$ (avoiding the redundant ones)
- remove the redundant columns with a destructive phase

Path Relinking

Path Relinking (*PR*), proposed by Glover (1989), is generally used as a final intensification procedure more than as a stand-alone method

Given a neighbourhood N and an exchange heuristic based on it

- collect in a reference set R the best solutions generated by the auxiliary heuristic (elite solutions)
- for each pair of solutions x and y in R
  - build a path γ_{xy} from x to y in the search space of neighbourhood N applying to $z^{(0)} = x$ the auxiliary exchange heuristic,
    but choosing at each step the solution closest to the destination y

    $$z^{(k+1)} := \arg \min_{z \in N(z^{(k)})} d(z, y)$$

    where d is a suitable metric function on the solutions
    In case of equal distance, optimise the objective function f
  - find the best solution z^*_{xy} along the path (and improve it)

    $$z^*_{xy} := \arg \min_{k \in \{1, \ldots, |\gamma_{xy}| - 1\}} f(z^{(k)})$$

  - if $z^*_{xy} \notin R$ and is better than the worst in R, add it to R

General scheme of the *Path Relinking* approach

Algorithm PathRelinking(I, P, n_R)

Repeat

  $R := \emptyset$;

  For each $x \in P$ *do*

    $z :=$ SteepestDescent(I, x); *If* $f(z) < f(x^*)$ *then* $x^* := z$;

    $y_R := \arg\max_{y \in R} f(y)$;

    If $z \notin R$ *and* $(|R| < n_R$ *or* $f(z) < f(y_R))$ *then*

      { R keeps the n_R best unique solutions }

      $R := R \cup \{z\}$; Stop $:=$ *false*; *If* $|R| > n_R$ *then* $R := R \setminus \{y_R\}$;

    EndIf

  EndFor

  $P := \emptyset$;

  For each $x \in R$ *and* $y \in R \setminus \{x\}$ *do* { Recombine to build the new population }

    $z := x$; $z^* := x$;

    While $z \neq y$ *do* { Build a path from x to y }

      $Z := \arg\min_{z' \in N(z)} d(z', y)$; $z := \arg\min_{z' \in Z} f(z')$;

      If $f(z) < f(z^*)$ *then* $z^* := z$

    EndWhile;

    If $z^* \notin P$ *then* $P := P \cup \{z^*\}$;

  EndFor

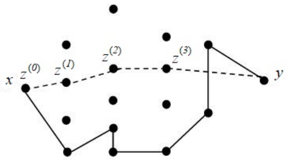
until Stop $=$ *true*;

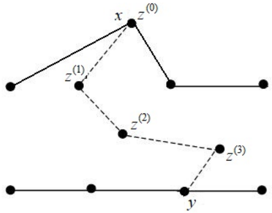
Return $(x^*, f(x^*))$;

Relinking paths

The paths explored in this way

- intensify the search, because they connect good solutions
- diversify the search, because they follow different paths with respect to the exchange heuristic (especially if the extremes are far away)





- since the distance of $z^{(k)}$ from y is decreasing, one can explore
  - worsening solutions without the risk of cyclic behaviours
  - unfeasible subsets without the risk of not getting back to feasibility

  (*they do not improve directly, but open the way to improvements*)

Variants

Given two solutions x and y, Path Relinking has several variants:

- *forward path relinking*: build a path from the worse to the better one
- *backward path relinking*: build a path from the better to the worse one
- *back-and-forward path relinking*: build both paths
- *mixed path relinking*: build a path with alternative steps from each extreme (updating the destination)
- *truncated path relinking*: build only the first steps of the path (if the good solutions are experimentally close to each other)
- *external path relinking*: build a path from one moving away from the other (if the good solutions are experimentally far from each other)