Heuristic Algorithms

Master's Degree in Computer Science/Mathematics

Roberto Cordone
DI - Università degli Studi di Milano

Schedule:	Thursday 14.30 - 16.30 on MS-Teams
	Friday 14.30 - 16.30 on MS-Teams
Office hours:	on appointment
E-mail:	roberto.cordone@unimi.it
Web page:	https://homes.di.unimi.it/cordone/courses/2020-ae/2020-ae.html
Ariel site:	https://rcordoneha.ariel.ctu.unimi.it

Overcoming local optima

The *steepest descent* exchange heuristics only provide local optima

In order to improve, one can

- repeat the search (*How to avoid following the same path?*)
- extend the search (*How to avoid falling in the same optimum?*)

In the constructive algorithms only repetition was possible

The constructive metaheuristics exploit

- randomization
- memory

to operate on $\Delta_A^+(x)$ and $\varphi_A(i,x)$

The exchange metaheuristics exploit them to operate on

1. the starting solution $x^{(0)}$ (multi-start, *ILS*, *VNS*)
2. the neighbourhood $N(x)$ (*VND*)
3. the selection criterium $\varphi(x, A, D)$ (*DLS* or *GLS*, *SA*, *TS*)

Termination condition

A search that repeats or proceeds beyond local optimum can ideally be infinite

In pratice, one uses "absolute" termination conditions

1. a given total number of explorations of the neighbourhood or
   a given total number of repetitions of the local search
2. a given total execution time
3. a given value of the objective
4. a given improvement of the objective with respect to the starting solution

or "relative" termination conditions

1. a given number of explorations of the neighbourhood or repetitions after the last improvement of f^*
2. a given execution time after the last improvement
3. a given minimum value of the ratio between improvement of the objective and number of explorations or execution time
   (*e.g.: f^* improves less than* 1% *in the last* 1 000 *explorations*)

Fair comparisons require absolute conditions (time or number of explorations)

Modify the starting solution: random generation

It is possible to create different starting solutions

- generating them at random
- applying different constructive heuristics
- modifying solutions generated by the exchange algorithm

The advantages of random generation are

- conceptual simplicity
- quickness for the problems in which it is easy to guarantee feasibility
- control on the probability distribution in X based on
  - element cost (e.g., favour the cheapest elements)
  - element frequency during the past search, to favour the most frequent elements (intensification) or the less frequent ones (diversification)

This combines randomization and memory

- asymptotic convergence to the optimum (in infinite time)

The disadvantages of random generation are

- scarse quality of the starting solutions (*not the final ones!*)
- long times before reaching the local optimum

This depends on the complexity of the exchange algorithm

- inefficiency when deciding feasibility is $\mathcal{NP}$-complete

Modify the starting solution: constructive procedures

Multi-start methods are the classical approach

- design several constructive heuristics
- each constructive heuristic generates a starting solution
- each starting solution is improved by the exchange heuristic

The disadvantages are

1. scarce control: the generated solutions tend to be similar
2. impossibility to proceed indefinitely: the number of repetitions is fixed
3. high design effort: several different algorithms must be designed
4. no guarantee of convergence, not even in infinite time

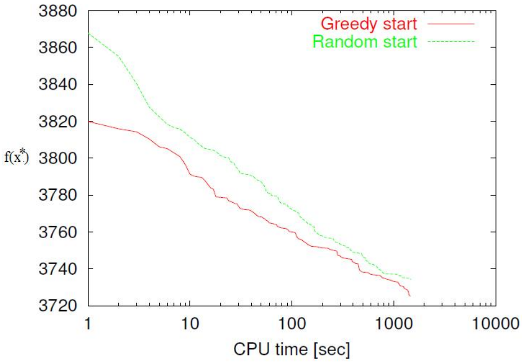
Consequently, constructive metaheuristics are preferred nowadays

GRASP and Ant System include by definition an exchange procedure

Influence of the starting solution

If the exchange heuristic is

- good, the starting solution has a short-lived influence:
  a random or heuristic generation of $x^{(0)}$ are very similar
- bad, the starting solution has a long-lived influence:
  a good heuristic to generate $x^{(0)}$ is useful



This exchange heuristic is not very good

Modify the starting solution exploiting the previous ones

The idea is to exploit the information on previously visited solutions

- save reference solutions, such as the best local optimum found so far and possibly other local optima
- generate the new starting solution modifying the reference ones

The advantages of this approach are

- control: the modification can be reduced or increased *ad libitum*
- good quality: the starting solution is very good
- conceptual simplicity
- implementation simplicity: the modification can be performed with the operations definining the neighbourhood
- asymptotic convergence to the optimum under suitable conditions

Iterated Local Search (*ILS*)

The Iterated Local Search (*ILS*) requires

- an *steepest descent* exchange heuristic to produce local optima
- a perturbation procedure to generate the starting solutions
- an acceptance condition to decide whether to change the reference solution x
- a termination condition

Algorithm IteratedLocalSearch$\left(I, x^{(0)}\right)$

$x := \text{SteepestDescent}\left(x^{(0)}\right)$; $x^* := x$;

For $l := 1$ *to* ℓ *do*

  $x' := \text{Perturbate}(x)$;

  $x' := \text{SteepestDescent}(x')$;

  If $\text{Accept}(x', x^*)$ *then* $x := x'$;

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

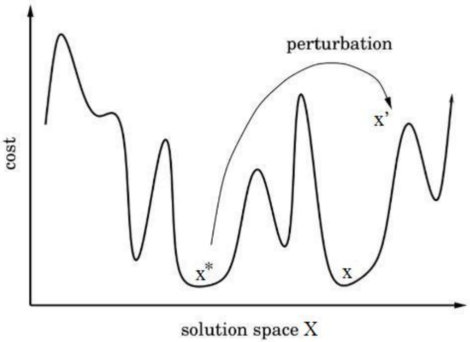
EndWhile;

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

Iterated Local Search (*ILS*)

The idea is that

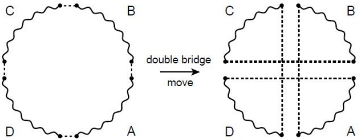
- the exchange heuristic quickly explores an attraction basin, terminating into a local optimum
- the perturbation procedure moves to another attraction basin
- the acceptance condition evaluates if the new local optimum is a promising starting point for the following perturbation



Example: *ILS* for the *TSP*

A classical application of *ILS* to the *TSP* uses

- exchange heuristic: *steepest descent* with neighbourhood $N_{\mathcal{R}_2}$ or $N_{\mathcal{R}_3}$
- perturbation procedure: a *double-bridge* move that is particular kind of 4-exchange



- acceptance condition: the best known solution improves

$$f(x') < f(x^*)$$

Perturbation procedure

Let $\mathcal{O}$ be the operation set that defines neighbourhood $N_{\mathcal{O}}$

The perturbation procedure performs a random operation o

- with $o \in \mathcal{O}' \nsubseteq \mathcal{O}$, to avoid that the exchange heuristic drive solution x' back to the starting local optimum x

Two typical definitions of $\mathcal{O}'$ are

- sequences of $k > 1$ operations of $\mathcal{O}$
  (generating a random sequence is cheap)
- conceptually different operations
  (e.g., vertex exchanges instead of arc exchanges)

The main difficulty of *ILS* is in tuning the perturbation: if it is

- too strong, it turns the search into a random restart
- too weak, it guides the search back to the starting optimum
  - wasting time
  - possibly losing the asymptotic convergence

Ideally one would like to enter any basin and get out of any basin

Acceptance condition

Algorithm IteratedLocalSearch$\left(I, x^{(0)}\right)$

$x :=$ SteepestDescent$\left(x^{(0)}\right)$; $x^* := x$;

For $l := 1$ *to* ℓ *do*

  $x' :=$ Perturbate(x);

  $x' :=$ SteepestDescent(x');

  If Accept(x', x^*) *then* $x := x'$;

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

EndWhile;

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

The acceptance condition balances intensification and diversification

- accepting only improving solutions favours intensification

$$\text{Accept}(x', x^*) := (f(x') < f(x^*))$$

  The reference solution is always the best found: $x = x^*$

- accepting any solution favours diversification

$$\text{Accept}(x', x^*) := \textit{true}$$

  The reference solution is always the last optimum found: $x = x'$

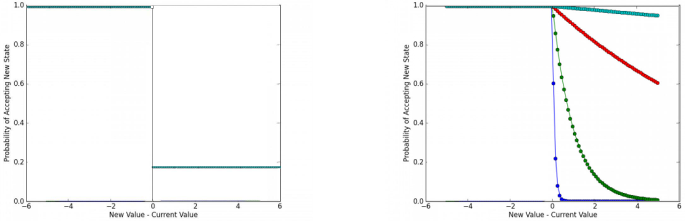
Acceptance condition

Intermediate strategies can be defined based on $\delta f = f(x') - f(x^*)$

- if $\delta f < 0$, always accept x'
- if $\delta f \geq 0$, accept x' with probability $\pi(\delta f)$,
  where $\pi(\cdot)$ is a nonincreasing function

The most typical cases are:

- constant probability: $\pi(\delta f) = \bar{\pi} \in (0;1)$ for each $\delta f \geq 0$
- monotonically decreasing probability with $\pi(0) = 1$ and
  $\lim_{\delta f \to +\infty} \pi(\delta) = 0$



Memory can also be used, accepting x' more easily
if many iterations have elapsed since the last improvement of x^*

Variable Neighbourhood Search (*VNS*)

A method very similar to *ILS* is the *Variable Neighbourhood Search* proposed by Hansen and Mladenović (1997)

The main differences betwween *ILS* and *VNS* are the use of

- the strict acceptance condition: $f(x') < f(x^*)$
- an adaptive perturbation mechanism instead of the fixed one

VNS often introduces also neighbourhood modifications

The perturbation mechanism is based on a hierarchy of neighbourhoods, that is a family of neighbourhoods with an increasing parametric size k

$$N_1 \subset N_2 \subset \ldots \subset N_k \subset \ldots N_{k_{\max}}$$

Typically one uses the parameterised neighbourhoods

- N_{H_k}, based on the Hamming distance between subsets
- $N_{\mathcal{O}_k}$, based on the sequences of operations from a basic set $\mathcal{O}$

and extracts $x^{(0)}$ randomly from a neighbourhood of the hierarchy

Adaptive perturbation mechanism

It is called *variable neighbourhood* because the neighbourhood used to extract $x^{(0)}$ varies based on the results of the exchange heuristic

- if a better solution is found, use the smallest neighbourhood, to generate a starting solution very close to x^* (intensification)
- if a worse solution is found, use a slightly larger neighbourhood, to generate a starting solution slightly farther from x^* (diversification)

The method has three parameters

1. $k_{\min}$ identifies the smallest neighbourhood to generate new solutions
2. $k_{\max}$ identifies the largest neighbourhood to generate new solutions
3. δk is the increase of k between two subsequent attempts

The exchange heuristic adopts the smallest neighbourhood to be efficient

(N_1, *or anyway* N_k *with* $k \leq k_{\min}$)

General scheme of the *VNS*

Algorithm VariableNeighbourhoodSearch$(I, x^{(0)})$

$x :=$ SteepestDescent$(x^{(0)})$; $x^* := x$;

$k := k_{\min}$;

For $l := 1$ *to* ℓ *do*

  $x' :=$ Shaking(x^*, k);

  $x' :=$ SteepestDescent(x');

  If $f(x') < f(x^*)$

    then $x^* := x'$; $k := k_{\min}$;

    else $k := k + \delta k$;

  If $k > k_{\max}$ *then* $k := k_{min}$;

EndWhile;

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

- the reference solution x' is always the best known solution x^*
- the starting solution is obtained extracting it at random from the current neighbourhood of the reference solution $N_k(x^*)$
- the exchange heuristic produces a local optimum with respect to the basic neighbourhood N
- if the best known solution improves, the current neighbourhood becomes $N_{k_{\min}}$
- otherwise, move to a larger neighbourhood $N_{k+\delta k}$, never exceeding $N_{k_{\max}}$

Parameter tuning

The value of $k_{\min}$ must be

- large enough to get out of the current attraction basin
- small enough to avoid jumping over the adjacent attraction basins

In general, one sets $k_{\min} = 1$, and increases it if experimentally profitable

The value of $k_{\max}$ must be

- large enough to reach any useful attraction basin
- small enough to avoid reaching useless regions of the solution space

Example: the diameter of the search space for the basic neighbourhood: $\min(m, n-m)$ for the *MDP*; n for the *TSP* and *MAX-SAT*, etc. . .

The value of δk must be

- large enough to reach $k_{\max}$ in a reasonable time
- small enough to allow each reasonable value of k

In general, one sets $\delta k = 1$

In order to favour diversification, it is possible to accept x' when

$$f(x') < f(x^*) + \alpha \, d_H(x', x^*)$$

where

- $d_H(x', x^*)$ is the Hamming distance fra x' and x^*
- $\alpha > 0$ is a suitable parameter

This allows to accept worsening solutions as long as they are far away

- $\alpha \approx 0$ tends to accept only improving solutions
- $\alpha \gg 0$ tends to accept any solution

Of course, the random strategies seen for the ILS can also be adopted