UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Local search optimization

Introduction to metaheuristics

Jose M Sallan

Outline

Local search optimization

Local search optimization algorithms move from solution to solution in the **search space** of candidate solutions by applying **local changes**, until a solution deemed optimal is found or a time bound is elapsed.

Elements of a local search algorithm:

- An **initial solution**, random or obtained through an heuristic
- A way to perform **local changes** on a solution
- A specific way to explore the search space

Local change definition

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Local changes are defined by going from the current solution to another solution of its **neighbourhood**.

Neighbourhood definition is problem specific, and depends on how the solution is **encoded**:

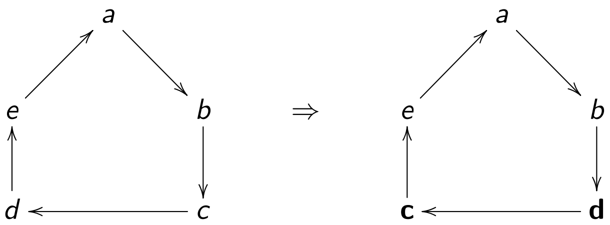
- Knapsack problem: vector of logicals (TRUE if solution is in the knapsack)
- Flowshop problem: sequence is defined as a permutation
- Travelling salesperson problem: permutation, list of edges

Local change definition

Neighborhood definition for the TSP

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Swap of two nodes (TSP)



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Local change definition

Neighborhood definition for the TSP

Swap of two edges (TSP)

$$
\begin{tikzcd}
i-1 \arrow[r] & i \arrow[d, bend left] \\
k+1 \arrow[u, bend left] & k \arrow[l]
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
i-1 \arrow[dr] & i \arrow[dl] \\
k+1 \arrow[u, bend left] & k \arrow[u, bend right]
\end{tikzcd}
$$

Local change definition

Neighborhood definition for the flowshop problem

Swapping two adjacent elements (for solution 12345):

1	**21**345
2	1**32**45
3	12**43**5
4	123**54**

For a problem with n tasks, this neighbourhood has $n-1$ solutions

Local change definition

Neighborhood definition for the flowshop problem

Swapping any pair of elements:

$$\begin{array}{cc} (1,2) & \mathbf{21}345 \\ (1,3) & \mathbf{3}2\mathbf{1}45 \\ (1,4) & \mathbf{4}23\mathbf{1}5 \\ & \cdots \\ (3,4) & 12\mathbf{43}5 \\ (3,5) & 12\mathbf{5}4\mathbf{3} \\ (4,5) & 123\mathbf{54} \end{array}$$

For a problem with n tasks, this neighbourhood has $n\left(n-1\right)/2$ solutions

Local change definition

Neighborhood definition for the flowshop problem

Insertion of element i in position j:

$$
\begin{array}{ll}
(1,2) & 2\mathbf{1}345 \\
(1,3) & 23\mathbf{1}45 \\
(1,4) & 234\mathbf{1}5 \\
 & \dots \\
(3,4) & 124\mathbf{3}5 \\
(3,5) & 1245\mathbf{3} \\
(4,5) & 1235\mathbf{4}
\end{array}
$$

For a problem with n tasks, this neighbourhood has $n\,(n-1)$ solutions

Search space exploration

Every (generic) way of exploring the search space leads to a local search metaheuristic:

- Hill climbing
- Simulated annealing
- Tabu search
- GRASP
- Iterated local search

Local search metaheuristics combine **exploitation** (improve current solution) with **exploration** (explore new solutions to avoid being trapped in local optima)

Hill climbing

Hill climbing (HC) algorithms attempt to find a better solution incrementally changing the current solution. This is repeated until no better solution is found.

Variants of hill climbing:

- **Simple HC:** selects the first or closer solution at each step
- **Steepest descent / ascent HC:** selects the best solution of all neighbourhood
- **Stochastic HC:** check one or several random elements of neighbourhood and choose best

Steepest descent HC returns a local optimum (better value of the objective function than any of its neighbourhood)

Hill climbing pseudocode

Input: A starting solution s_0, a fitness function f and neighborhood definition N
Output: A satisfactory solution s

$s^* \leftarrow s_0$
$k \leftarrow 1$
while k=*1* **do**
    $s' \leftarrow \text{MIN}\left\{f\left(N\left(s^*\right)\right)\right\}$
    if $f\left(s'\right) \leq f\left(s^*\right)$ **then**
        $s^* \leftarrow s'$
    else
        $k \leftarrow 0$
    end
end
return s^*

Algorithm 1: Steepest descent hillclimbing

Hill climbing pseudocode

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Solutions considered in the algorithm:

- Starting solution s_0
- Current best solution s^*
- Candidate solution s'

Simulated annealing

A variant of hill climbing, where a worse solution coming from local search can be accepted with probability $p = c(t)$, where t is the number of iterations.

The name and inspiration come from **annealing** in metallurgy, a technique involving heating and **controlled cooling** of a material to increase the size of its crystals and reduce their defects.

Simulated annealing

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

A possible function of probability of accepting a solution s' coming from local search from s in a minimizing problem can be:

$$c(t) = \begin{cases} 1 & \text{if } f(s') \leq f(s) \\ e^{-\mu(f(s')-f(s))/T} & \text{if } f(s') > f(s) \end{cases}$$

Where T is a temperature value which decreases with iterations t.

Solutions considered:

- Starting solution s_0
- Current solution s
- Candidate solution s'
- Best (current) solution s^*

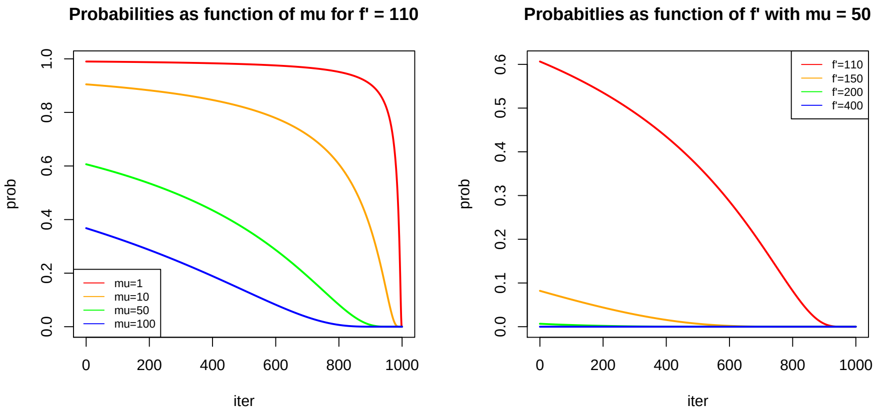
Simulated annealing

Accepting for **exploration** a candidate solution s' worse than current solution s depends on:

- **Temperature:** as temperature decreases (number of iterations increases), decreases probability of exploration
- **Quality of solution:** as $f(s')$ increases, decreases probability of exploration

The evolution of the probability can be adjusted through parameter μ.

Simulated annealing



Probabilities for $f = 100$

Simulated annealing pseudocode

Input: A starting solution s_0, a fitness function f, number of iterations T_{max} and neighborhood definition N
Output: A satisfactory solution s^*

$s \leftarrow s_0$
$s^* \leftarrow s_0$
$T \leftarrow T_{max}$
while $T > 0$ **do**
  select randomly $s' \in N(s)$
  if $f(s') \leq f(s)$ **then**
    $s \leftarrow s'$
    $s^* \leftarrow s'$
  else
    if $e^{-\mu(f(s')-f(s))/T} > U(0,1)$ **then**
      $s \leftarrow s'$
    end
  end
  $T \leftarrow T - 1$
end
return s^*

Algorithm 2: Simulated annealing

Tabu search

Tabu search (TS) is a metaheuristic similar to HC, but tries to avoid be stuck in local optima:

- accepting to explore the best solution found in each iteration, even if it is not better than the current best solution.
- discourages selecting solutions previously visited prohibiting (making **tabu**) some moves.

TS heuristics use memory structures (tabu lists, or rather tabu queues) to store prohibited moves.

A example of tabu move

Exploring a neighbourhood of the 12345 solution of a TSP of $n = 5$:

21345
1**32**45
12**43**5 Best move
123**54**
5234**1**

The move **43** is included in the tabu list after this iteration, as brings back solution 12345.

Tabu search pseudocode

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Input: A starting solution s_0, a fitness function f, iterations T_{max} and neighborhood definition N

Output: A satisfactory solution s^*

$s \leftarrow s_0$
$s^* \leftarrow s_0$
$T \leftarrow 0$
empty tabu list
while $T < T_{max}$ **do**
    $s' \leftarrow \text{MIN}\left\{f\left(\overline{N}\left(s\right)\right)\right\}$
    if $f\left(s'\right) < f\left(s^*\right)$ **then**
        $s^* \leftarrow s'$
    end
    $s \leftarrow s'$
    update tabu list
    obtain $\overline{N}$ deleting tabu moves from N
    $T \leftarrow T + 1$
end
return s^*

Algorithm 3: Tabu search

GRASP

GRASP stands for Greedy Adaptive Search Procedure.

A GRASP solution is obtained in two steps:

- **construction** of an initial solution
- **refinement** by local search.

To increase exploration, randomness is introduced in the construction phase:

- A constructive heuristic is used as a template.
- Instead of selecting the best element in each step, a random element from a **restricted candidate list** (RCL) is chosen.
- The RCL must be updated in each step.

A GRASP for the TSP

A possible implementation of GRASP for the TSP can be:

- **Construction:** through a savings-based heuristic, choosing the next edge from a RCL of k compatible edges of maximum saving value.
- **Refinement:** simulated annealing using node swap to define the neighbourhood.

Iterated local search

Iterated Local Search (ILS) heuristics include:

- a **local search** heuristic.
- a way to produce a **perturbation** of a solution.
- an **acceptance criterion** for a candidate solution.

Then, for each step:

- obtain a perturbation s' of current solution s^*.
- obtain $s^{*\prime}$ making local search from s'.
- replace s^* by $s^{*\prime}$ if an acceptance criterion is met.

Iterated local search

Elements of ILS:

- The local search can be more or less sophisticated (for instance, HC, SA or TS)
- The perturbation must be not so strong as look like random restart, and not so weak as to be undone by local search
- Perturbation should be complementary (i. e., different) from local search

Conclusions

Several procedures are available to refine the results of a given initial solution:

- Hill climbing
- Simulated annealing
- Tabu search

Others, such as iterated local search or GRASP, apply local search techniques to a set of more or less close solutions.

Many of these are based in combining exploration of search space with exploitation thorugh solution improvement, to avoid local optima.