Metaheuristics

2.9, 2.11

MEYSAM AGHIGHI

2.9 Other single-solution based metaheuristics

Some other strategies used by S-metaheuristics to escape from local optima:

- Smoothing
- Noisy
  - Based on transformations of the landscape by changing the input data
- GRASP
  - An iterative greedy heuristic to solve combinatorial optimization problems

Smoothing Methods

- Reduces:
  - Number of local optima
  - Basins of attraction
- Does not change:
  - Region of the global optima
- After smoothing, any S-metaheuristic (or even P-metaheuristic) can be used.

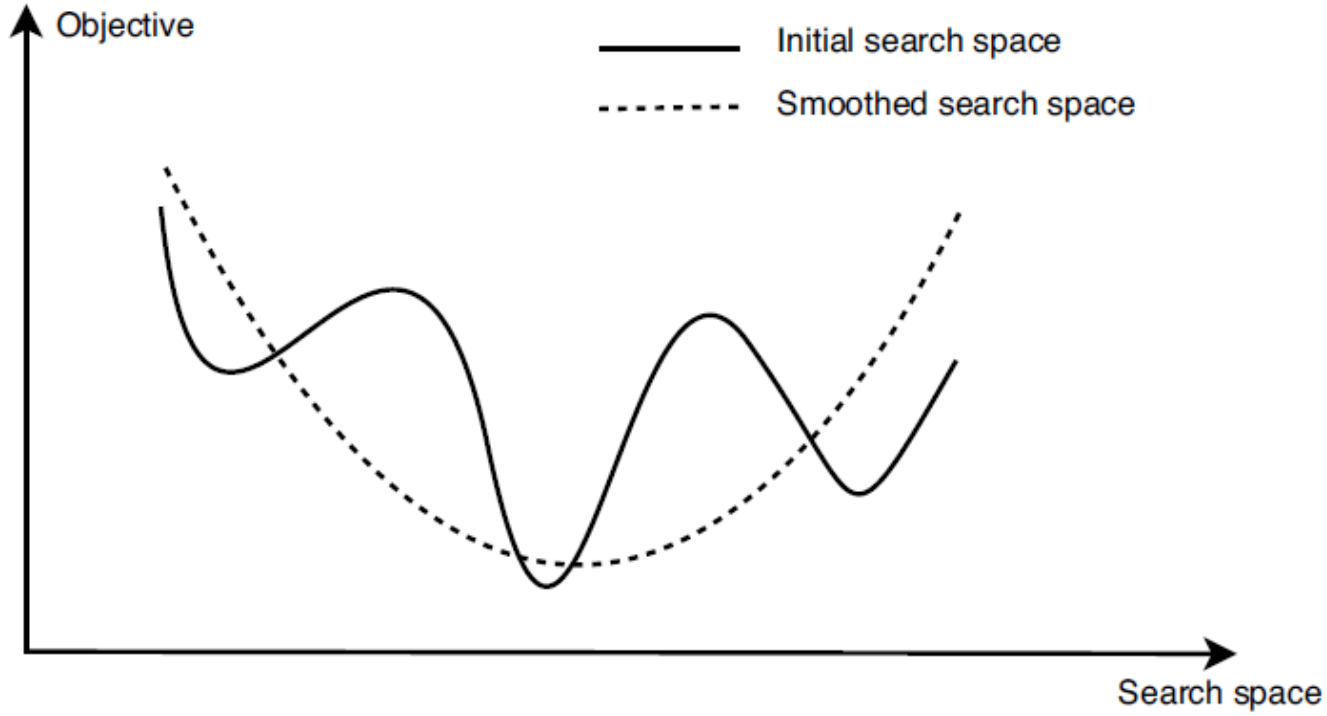

FIGURE 2.33 Search space smoothing that results in an easiest problem to solve. The smoothed landscape has less local optima than the original one.

Smoothing Methods

- Start by a large α – smoothing factor
- Use solution of the smoothed landscape as the initial solution for the next iteration
- Decrease α every iteration
- $\alpha = 1$ represents the original space

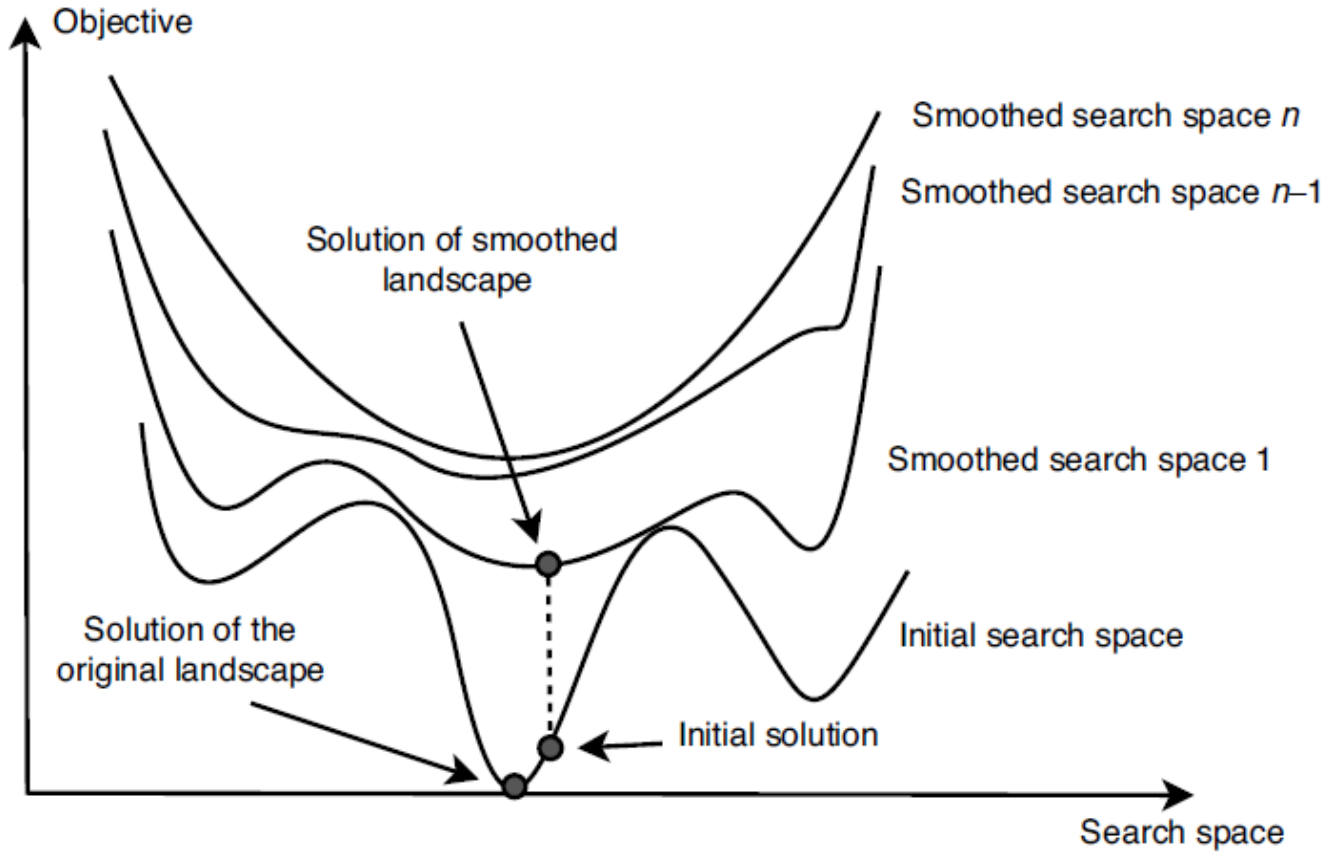

FIGURE 2.34 Successive smoothing of the landscape. The solution found at the step i will guide the search at the iteration $i + 1$ in a more rugged landscape.

Smoothing Methods

Algorithm 2.15 Template of the smoothing algorithm.

Input: S-metaheuristic LS, α_0, Instance I.
$s = s_0$; /* Generation of the initial solution */
$\alpha = \alpha_0$; /* Initialization of the smoothing factor */
Repeat
  $I = I(\alpha)$; /* Smoothing operation of the instance I */
  $s = LS(s, I)$; /* Search using the instance I and the initial solution s */
  $\alpha = g(\alpha)$; /* Reduce the smoothing factor, e.g. $\alpha = \alpha - 1$ */
Until $\alpha < 1$ /* Original problem */
Output: Best solution found.

Smoothing Methods

- Example 2.38 – Smoothing operation for the TSP
  - A trivial case for TSP is where all the distances are equal: $d_{ij} = \bar{d}$,

$$\bar{d} = \frac{1}{n(n-1)} \sum_{i \neq j} d_{ij}$$

  - Smoothing of the distances:

$$d_{ij}(\alpha) = \begin{cases} \bar{d} + \left(d_{ij} - \bar{d}\right)^{\alpha} \; if \; d_{ij} \geq \bar{d} \\ \bar{d} - \left(\bar{d} - d_{ij}\right)^{\alpha} \; if \; d_{ij} < \bar{d} \end{cases}$$

  - $d_{ij}(\alpha) \rightarrow \bar{d}$ when $\alpha \gg 1$. Flat landscape represents the first instance.
  - $d_{ij}(\alpha) = d_{ij}$ when $\alpha = 1$. Last landscape represents the original problem.
  - Normalize distances first: $d_{ij} = D_{ij}/D_{max}$

Smoothing – main parameters

- Appropriate choice of
  - The initial value of the smoothing factor α
  - The controlling strategy of the smoothing factor α

- The larger α_0, the more time consuming the algorithm.

Noisy Method

- The data parameters are replaced by "noised" data.
- At each iteration of the search, the noise is reduced.
- For instance, consider a graph $G = (V, E)$ with weights w_{ij} of the edges (i, j). The weights are replaced by

$$w_{ij}^{\text{noised}} = w_{ij} + \rho_{ij}$$

- The search terminates when the noise is low enough. The extreme stopping criterion is when the mean and the standard deviation of the noise are both 0.
- Contrary to traditional S-metaheuristics, an improving move may be rejected in the noising algorithm.

Noisy Method – a more general way

- The noise is considered in computing the neighbor of a solution

$$\Delta_{\text{noised}}(s, s') = \Delta(s, s') + \rho_k$$

  - ρ_k is a noise changing at each iteration k
- Randomness occurs in a different way
  - In the first method after fixing the noise, the local search can be deterministic.
  - In the second method, randomness takes place at each iteration of the local search
- The choice of noising method may have a great impact on performance.

Noisy Method - algorithm

Algorithm 2.16 Template of the noising method.

Input: r.
$s = s_0$; /* Generation of the initial solution */
$r = r_{max}$; /* Initialization of the noising factor */
Repeat
  $\Delta_{noised}(s, s') = \Delta(s, s') + r$; /* Noising operation of f */
  $s = LS(s, r)$; /* Search using the noising factor r */
  $r = g(r)$; /* Reduce the noising factor */
Until $r = r_{min}$ /* Original problem */
Output: Best solution found.

Noisy Method – parameter tuning

- For other S-metaheuristics, some parameters must be tuned:
  - **Noise rate:** The external values depending on the data $r_{\min}, r_{\max}$.
  - **Decreasing noise rate:** Usually geometrical decreasing function ($r = r\alpha, \alpha \in (0,1)$). Below is another example.
  - **Probability distribution:** Depends on the problem, but usually uniform distribution.

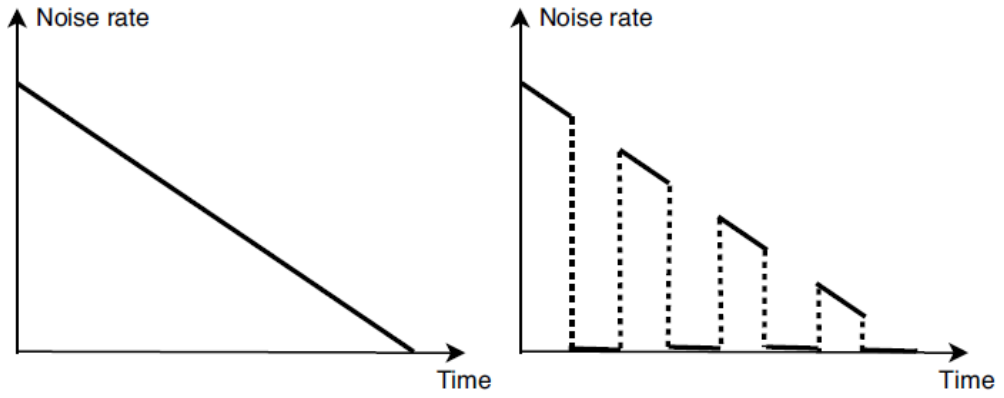

FIGURE 2.35 Different noising methods. (a) The noise rate is decreased linearly down to 0. (b) The noise decreases linearly, but "unperturbed" local searches are applied from time to time.

Noisy Method - example

- Example 2.39 – Noising algorithm for graph partitioning
  - Input: A complete weighted (+/-) graph $G = (V, E)$
  - Obj: Find a partition of V into subsets in order to minimize the sum of the weights of edges with both ends in the same subset.
- A noise with a uniform distribution $[-r, r]$ and an arithmetic decrease of the noise rate r, may be applied.
- Neighborhood: transferring a vertex from a subset to another one.
- Tuning parameters
  - $0.8w_{max} \leq r_{max} \leq 0.95w_{max}$, $0.15w_{max} \leq r_{min} \leq 0.5w_{max}$, weights range is $[-w_{max}, w_{max}]$.
  - The total number of iterations depends on the time the user wants to spend.
  - $\beta \leq \alpha \leq 2\beta$
    α denotes the number of pairs (noised LS, unnoised LS) applied between two restarts and β the number of restarts.
  - Where do we decrease the noise rate?

GRASP

- GRASP = Greedy Randomized Adaptive Search Procedure
- GRASP metaheuristic is an iterative greedy heuristic to solve combinatorial optimization problems.
- Each iteration has two steps: construction and local search.
- Construction: A feasible solution is built using a randomized greedy algorithm.
- The schema is repeated until a given number of iterations and the best found solution

GRASP - algorithm

Algorithm 2.17 Template of the greedy randomized adaptive search procedure.

Input: Number of iterations.
Repeat
  s = Random-Greedy(seed) ; /* apply a randomized greedy heuristic */
  $s' = Local - Search(s)$; /* apply a local search algorithm to the solution */
Until Stopping criteria /* e.g. a given number of iterations */
Output: Best solution found.

GRASP - design

- Main design questions: Greedy construction, Local search procedure
- **Greedy construction:** At each iteration the elements that can be included in the partial solution are ordered (decreasing value) in the list using local heuristic. From this list, a subset is generated called *restricted candidate list* (RCL).
  - Cardinality-based criteria
  - Value-based criteria: selecting the solutions that are better than a given threshold (more common).
- **Local search:** Since the solutions found by the construction procedure are not guaranteed to be local optima, it is beneficial to carry out a local search. Traditionally, a simple local search but any other S-metaheuristic can be used.

GRASP – greedy randomized algorithm

Algorithm 2.18 Template of the greedy randomized algorithm.

$s = \{\}$; /* Initial solution (null) */
Evaluate the incremental costs of all candidate elements ;
Repeat
    Build the restricted candidate list RCL ;
    /* select a random element from the list RCL */
    e_i = Random-Selection(RCL) ;
    If $s \cup e_i \in F$ **Then** /* Test the feasibility of the solution */
    $s = s \cup e_i$;
    Reevaluate the incremental costs of candidate elements ;
Until Complete solution found.

2.11 Conclusions

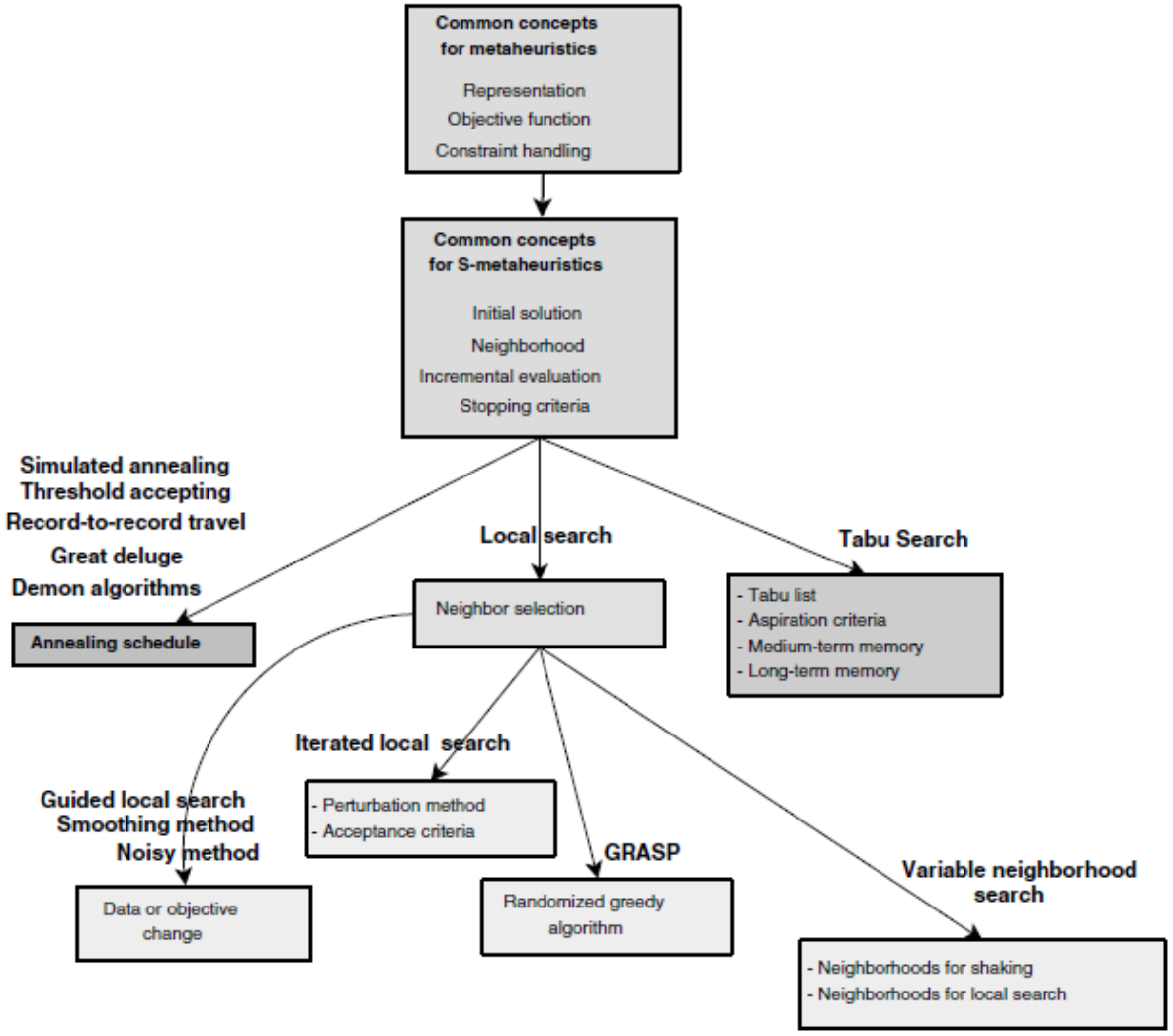

FIGURE 2.43 Common concepts and relationships in S-metaheuristics.

Conclusions

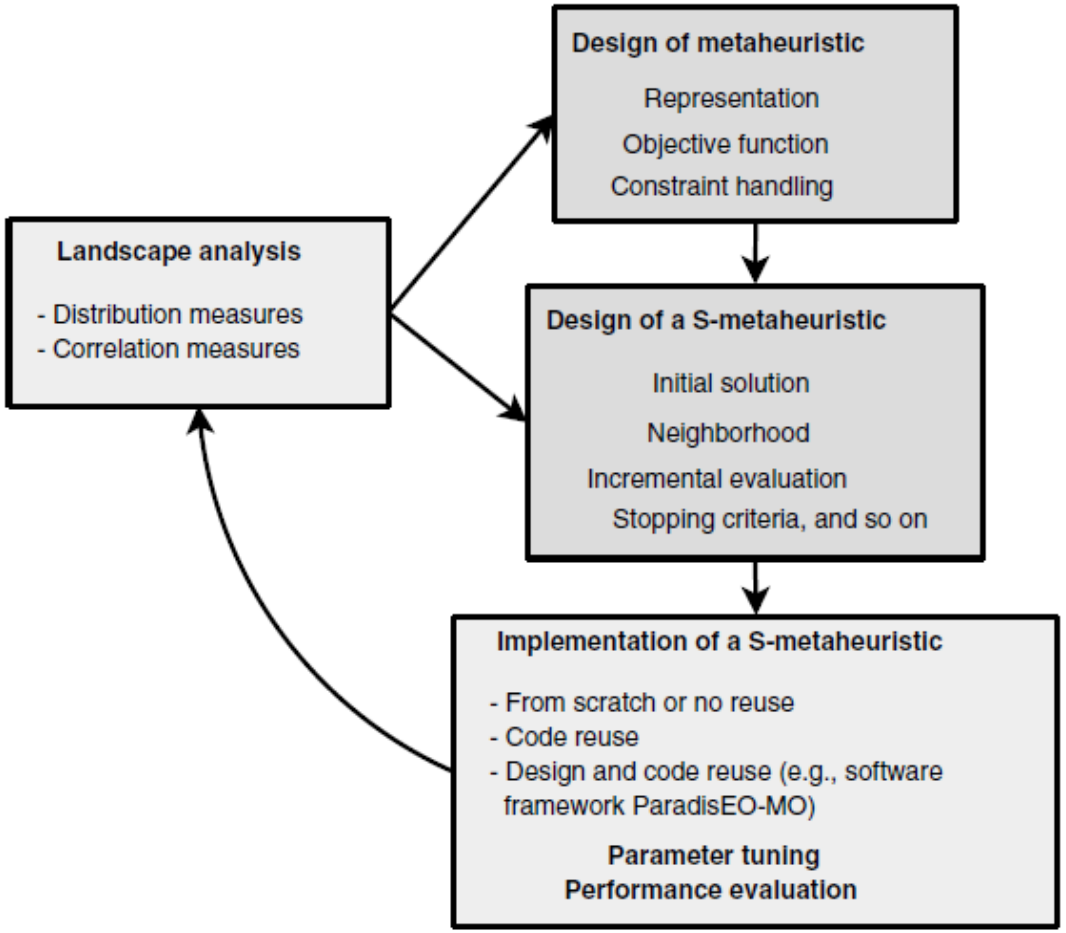

FIGURE 2.44 Development process of a S-metaheuristic.

The End