2.3 LOCAL SEARCH

Local search[12] is likely the oldest and simplest metaheuristic method [3,598]. It starts at a given initial solution. At each iteration, the heuristic replaces the current solution by a neighbor that improves the objective function (Fig. 2.21). The search

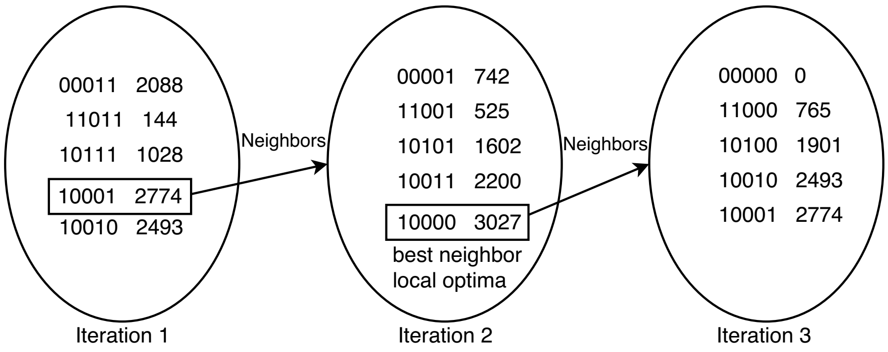

FIGURE 2.21 Local search process using a binary representation of solutions, a flip move operator, and the best neighbor selection strategy. The objective function to maximize is $x^3 - 60x^2 + 900x$. The global optimal solution is $f(01010) = f(10) = 4000$, while the final local optima found is $s = (10000)$, starting from the solution $s_0 = (10001)$.

[12]Also referred as hill climbing, descent, iterative improvement, and so on. In some literature, local search also refers to general S-metaheuristics.

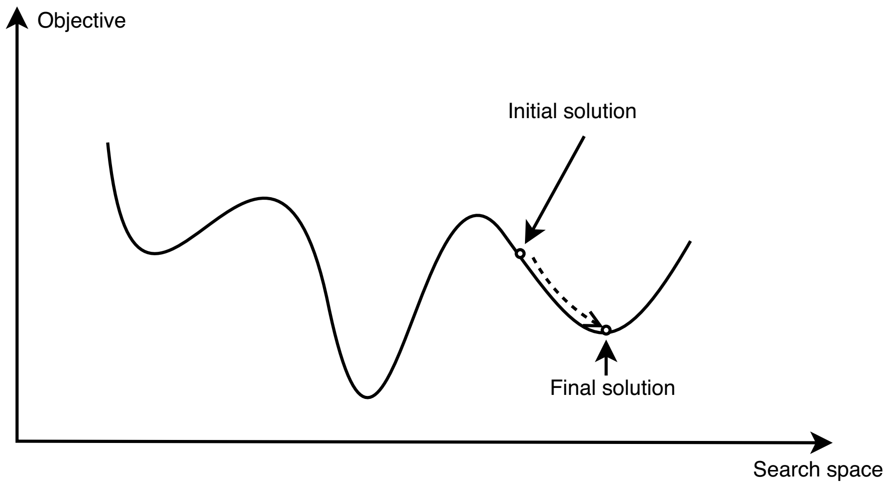

FIGURE 2.22 Local search (steepest descent) behavior in a given landscape.

stops when all candidate neighbors are worse than the current solution, meaning a local optimum is reached. For large neighborhoods, the candidate solutions may be a subset of the neighborhood. The main objective of this restricted neighborhood strategy is to speed up the search. Variants of LS may be distinguished according to the order in which the neighboring solutions are generated (deterministic/stochastic) and the selection strategy (selection of the neighboring solution) (Fig. 2.22).

LS may be seen as a descent walk in the graph representing the search space. This graph may be defined by $G = (S, V)$, where S represents the set of all feasible solutions of the search space and V represents the neighborhood relation. In the graph G, an edge (i, j) will connect to any neighboring solutions s_i and s_j. For a given solution s, the number of associated edges will be $|N(s)|$ (number of neighbors). Algorithm 2.2 illustrates the template of a local search algorithm.

Algorithm 2.2 Template of a local search algorithm.

```
$s = s_0$ ; /* Generate an initial solution $s_0$ */
**While** not Termination_Criterion **Do**
   Generate $(N(s))$ ; /* Generation of candidate neighbors */
   **If** there is no better neighbor **Then** Stop ;
   $s = s'$ ; /* Select a better neighbor $s' \in N(s)$ */
**Endwhile**
**Output** Final solution found (local optima).
```

From an initial solution s_0, the algorithm will generate a sequence $s_1, s_2, \ldots, s_k$ of solutions with the following characteristics (Fig. 1.32):

- The size of the sequence k is unknown a *priori*.
- $s_{i+1} \in N(s_i), \forall i \in [0, k - 1]$.

- $f(s_{i+1}) < f(s_i), \forall i \in [0, k-1]$.[13]
- s_k is a local optimum: $f(s_k) \leq f(s), \forall s \in N(s_k)$.

In addition to the definition of the initial solution and the neighborhood, designing a local search algorithm has to address the selection strategy of the neighbor that will determine the next current solution.

2.3.1 Selection of the Neighbor

Many strategies can be applied in the selection of a better neighbor (Fig. 2.23):

- **Best improvement (steepest descent):** In this strategy, the best neighbor (i.e., neighbor that improves the most the cost function) is selected. The neighborhood is evaluated in a fully deterministic manner. Hence, the exploration of the neighborhood is *exhaustive*, and all possible moves are tried for a solution to select the best neighboring solution. This type of exploration may be time-consuming for large neighborhoods.
- **First improvement:** This strategy consists in choosing the first improving neighbor that is better than the current solution. Then, an improving neighbor is immediately selected to replace the current solution. This strategy involves a partial evaluation of the neighborhood. In a *cyclic* exploration, the neighborhood is evaluated in a deterministic way following a given order of generating the neighbors. In the worst case (i.e., when no improvement is found), a complete evaluation of the neighborhood is performed.
- **Random selection:** In this strategy, a random selection is applied to those neighbors improving the current solution.

A compromise in terms of quality of solutions and search time may consist in using the first improvement strategy when the initial solution is randomly generated

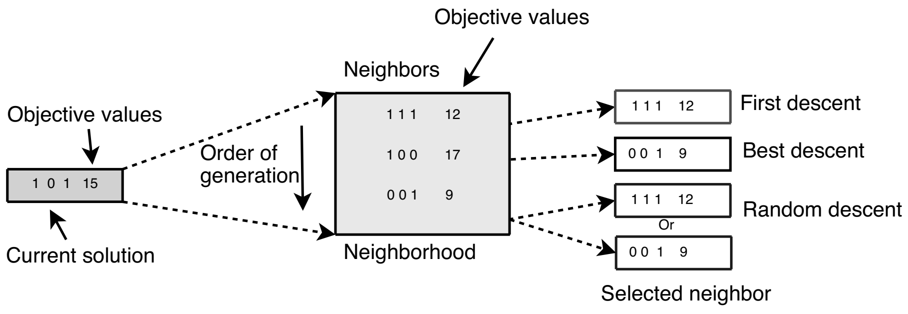

FIGURE 2.23 Selection strategies of improving neighbors.

[13]The problem is supposed to be a minimization problem.

and the best improvement strategy when the initial solution is generated using a greedy procedure. In practice, on many applications, it has been observed that the first improving strategy leads to the same quality of solutions as the best improving strategy while using a smaller computational time. Moreover, the probability of premature convergence to a local optima is less important in the first improvement strategy.

Example 2.21 Exhaustive versus cyclic exploration. Let us consider the neighborhood for the TSP that is based on the 2-opt move operator. A tour is represented by a permutation $(\pi_1, \pi_2, \ldots, \pi_n)$. In the exhaustive exploration, all 2-opt moves are applied. In a cyclic exploration, the order may be given as follows: apply successfully the 2-opt operator using the exchange of vertices (π_1, π_2) with (π_3, π_4), (π_4, π_5), and so on until an improving neighbor is found. In the worst case, the whole neighborhood is explored.

The order of application can be changed during the search. For instance, the initial parameter that determines the order may be chosen randomly. Instead of using always the edge (π_1, π_2), any edge (π_i, π_{i+1}) may represent the initial move.

Example 2.22 Cyclic exploration for a clique partitioning problem (CPP). Given a complete undirected graph $G = (V, E, w)$, where w represents positive or negative weights. To each edge (i, j) is associated a weight $w(i, j) = w(j, i)$. The objective of the CPP problem is to find a partition of the set of vertices V into k cliques $C_1, C_2, .., C_k$, minimizing the sum of the weights of the edges that have both end points in the same clique:

$$f(C_1, C_2, \ldots, C_k) = \frac{1}{2} \sum_{l=1}^{k} \sum_{(i,j) \in C_l \times C_l, i \neq j} w(i, j)$$

The number of cliques k is not fixed [125].

A solution s for the CPP problem may be represented by a discrete vector, where each element of the vector specifies the class of the vertex. The size of the vector is equal to the number of vertices $(n = |V|)$. The neighborhood may be defined by the transformation operator that consists in moving a single node to another class C. A class C of a solution s may be empty. The size of this neighborhood varies during the search and is equal to $(k + 1)n - \alpha$, where k is the number of classes and α is the number of single-node classes.

The cyclic exploration of a neighborhood may be based on finding the best class for a given node i. Obviously, one can use the given order of the vertices $(v_1, v_2, \ldots, v_n)$ or a random order generated uniformly. The vertices are explored in the specified order. A move is carried out as soon as an improving solution is found. All the nodes of the graph are tried once before the next time. This complete cyclic exploration is iterated until no improving neighbor is found. At the end of the process, all the nodes are in their best classes (i.e., local optimum related to the given move operator).

2.3.2 Escaping from Local Optima

In general, local search is a very easy method to design and implement and gives fairly good solutions very quickly. This is why it is a widely used optimization method in practice. One of the main disadvantages of LS is that it converges toward local optima. Moreover, the algorithm can be very sensitive to the initial solution; that is, a large variability of the quality of solutions may be obtained for some problems. Moreover, there is no means to estimate the relative error from the global optimum and the number of iterations performed may not be known in advance. Even if the complexity, in practice, is acceptable, the worst case complexity of LS is exponential! Local search works well if there are not too many local optima in the search space or the quality of the different local optima is more or less similar. If the objective function is highly multimodal, which is the case for the majority of optimization problems, local search is usually not an effective method to use.

As the main disadvantage of local search algorithms is the convergence toward local optima, many alternatives algorithms have been proposed to avoid becoming stuck at local optima. These algorithms are become popular since the 1980s. Four different families of approaches can be used to avoid local optima (Fig. 2.24) :

- **Iterating from different initial solutions:** This strategy is applied in multistart local search, iterated local search, GRASP, and so forth.

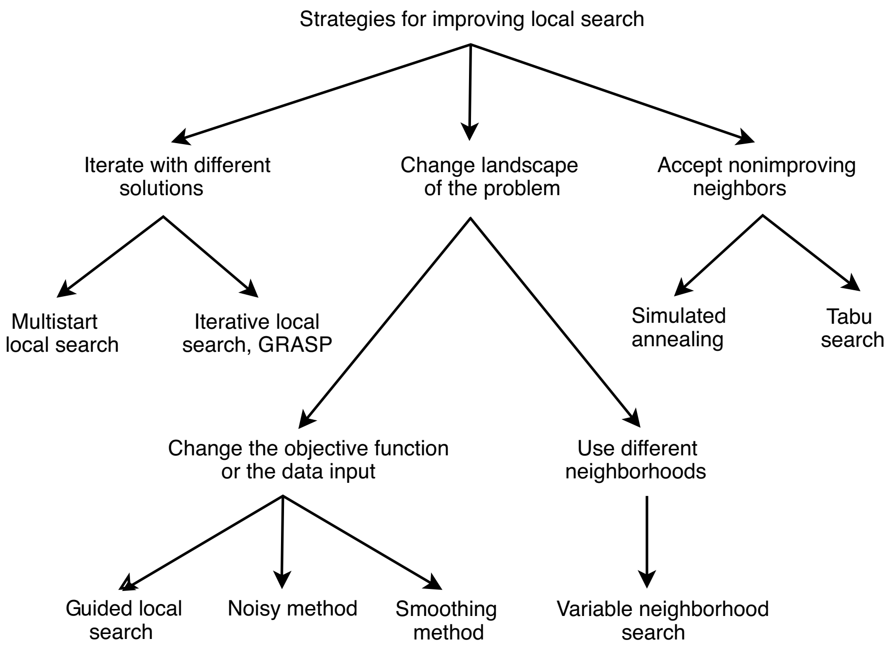
Strategies for improving local search
Iterate with different solutions
Change landscape of the problem
Accept nonimproving neighbors
Multistart local search
Iterative local search, GRASP
Simulated annealing
Tabu search
Change the objective function or the data input
Use different neighborhoods
Guided local search
Noisy method
Smoothing method
Variable neighborhood search

FIGURE 2.24 S-metaheuristic family of algorithms for improving local search and escaping from local optima.

- **Accepting nonimproving neighbors:** These approaches enable moves that degrade the current solution. It becomes possible to move out the basin of attraction of a given local optimum. Simulated annealing and tabu search are popular representative of this class of algorithms. Simulated annealing was the first algorithm addressing explicitly the question “why should we consider only downhill moves?”
- **Changing the neighborhood:** This class of approaches consists in changing the neighborhood structure during the search. For instance, this approach is used in variable neighborhood search strategies.
- **Changing the objective function or the input data of the problem:** In this class, the problem is transformed by perturbing the input data of the problem, the objective function or the constraints, in the hope to solve more efficiently the original problem. This approach has been implemented in the guided local search, the smoothing strategies, and the noising methods. The two last approaches may be viewed as approaches changing the landscape of the problem to solve.