1 Neighborhood Search

If you want a guarantee, buy a toaster.
— Clint Eastwood

An approximate answer to the right question is worth a great deal more
than a precise answer to the wrong question.
— John Tukey

This chapter givesa brief overview of local search. Its main purpose is to present local search algorithms generically in terms of a small set of concepts and operations. Chapter 2 shows how to instantiate the generic local search algorithm to obtain a variety of well-known heuristics and meta-heuristics. Thus this presentation stresses the commonalities underlying local search algorithms, not their differences. Readers interested in more detailed presentations of local search should consult, for instance, [1, 92].

1.1 Local Search

A local search algorithm typically starts from a solution (that is, an assignment of values to its decision variables) and moves from solutions to neighboring solutions in hope of improving a function f. The function f measures the quality of solutions to the problem at hand. In satisfiability problems, it typically provides a distance from the current solution to a feasible solution. In a pure optimization problem, it expresses the objective of the problem or a function of finer granularity that differentiates between solutions with the same objective value. In applications composed of both constraints and an objective, the function f may combine feasibility and optimality measures appropriate for the problem at hand. This chapter simply assumes the availability of a function f to be minimized.

The main operation of a local search algorithm amounts to moving from a solution s to one of its neighbors. The set of neighboring solutions of s, denoted by $N(s)$, is called the *neighborhood* of s. At a specific computation step, some of these neighbors may be *legal*, in which case they may be selected, or they may be *forbidden*. Once the legal neighbors are identified (by operation L), the local search selects one of them and decides whether to move to this neighbor or to stay at s (operation S). These concepts, which define the *moves* in local search, are illustrated in figure 1.1. It shows the solution s, its neighborhood $N(s)$, its set $L(N(s), s)$ of legal moves, and the selected solution (the bold thick circle).

Figure 1.2 depicts a simple generic local search template. The search starts from an initial state s (line 2) and performs a number of iterations (line 4). The move takes place in line 7 and consists of selecting a new solution by composing operations N, L, and S:

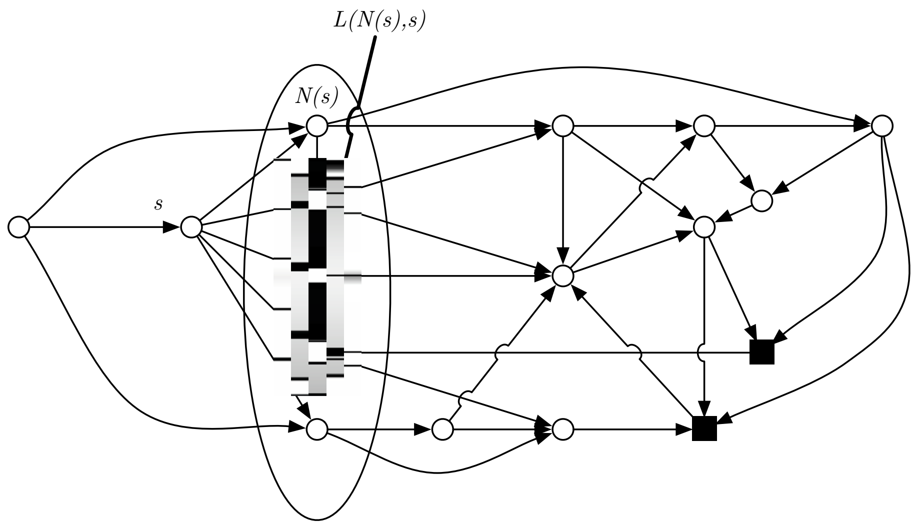


Figure 1.1: The Local Search Move.

```
function LOCALSEARCH {
    $s$ := GENERATEINITIALSOLUTION();
    $s^*$ := $s$;
    for $k := 1$ to $MaxTrials$ do
        if $satisfiable(s) \wedge f(s) < f(s^*)$ then
            $s^* := s$;
        $s := S(L(N(s), s), s)$;
    return $s^*$;
}
```

Figure 1.2: The Basic Local Search Template.

$s := S(L(N(s), s), s)$;

Lines 4 and 5 simply keep the best solution s^* encountered so far.

Some local search algorithms feature very simple implementations of some of these operations. For instance, in some local search algorithms, all moves may be legal. In other algorithms, these operations may be rather complex and may rely on sophisticated data structures and algorithms as well as on randomization.

1.2 Illustration

To illustrate the local search schema, this section presents a greedy local improvement algorithm in graph partitioning.

The Problem The graph-partitioning problem consists of finding a balanced partition of the vertices of a graph that minimizes the number of edges with one endpoint in each partition. More formally, a balanced partition of a graph $G = (V, E)$ is a pair $\langle P_1, P_2 \rangle$ such that $P_1 \cup P_2 = V$ and $|P_1| = |P_2|$. The cost of a partition $P = \langle P_1, P_2 \rangle$, denoted by $f(P)$, is the number of edges with one endpoint in each set:

$$f(\langle P_1, P_2 \rangle) = \#\{(v, w) \in E \ \mid \ v \in P_1 \ \& \ w \in P_2\}.$$

The set of feasible solutions, denoted by S, is the set of balanced partitions, and the set of optimal solutions, denoted by S^*, is specified as

$$\{s \in S \ \mid \ f(s) = \min_{p \in S} f(p)\}.$$

Figure 1.3 depicts a balanced partition of cost 9.

The Neighborhood The most natural move for graph partitioning consists of swapping two vertices, i.e, selecting a vertex a from P_1 and a vertex b from P_2 and assigning a to P_2 and b to P_1. Such a move is depicted in figure 1.4 and produces a partition of cost 5. More formally, the neighborhood function N is defined as

$$N(\langle P_1, P_2 \rangle) = \{\langle P_1 \setminus \{a\} \cup \{b\}, P_2 \setminus \{b\} \cup \{a\} \rangle \ \mid \ a \in P_1 \land b \in P_2\}.$$

The neighborhood N has a fundamental property: if the solution s is a balanced partition, all the solutions in $N(s)$ are balanced partitions as well. As a consequence, any local search starting from a balanced partition explores only balanced partitions and need not represent the balancing constraint explicitly. In other words, the local search algorithm considers only feasible solutions.

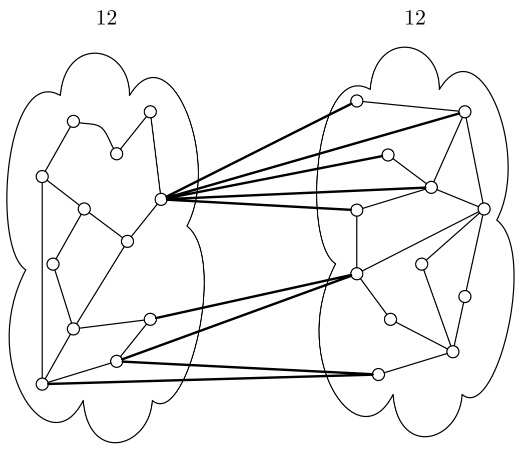


Figure 1.3: A Graph Partition of Cost 9.

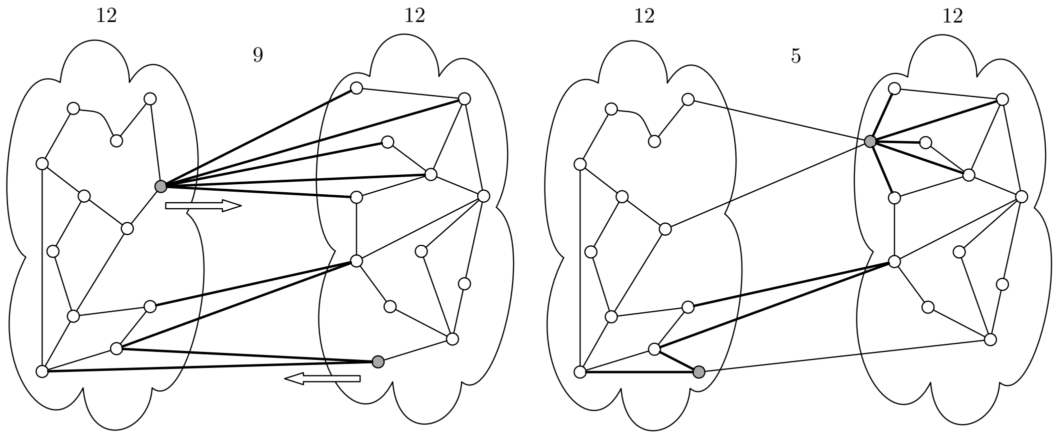


Figure 1.4: A Move from Neighborhood N.

Legal Moves Local improvement algorithms require the neighbors to improve the objective value. As a consequence, the legal moves in the local improvement algorithm are specified by

$$L(N, s) = \{n \in N \mid f(n) < f(s)\}.$$

Selection It remains to specify how to choose the neighbor. Since the algorithm is greedy, the algorithm selects the best legal neighbor, that is, an element from

$$S(\mathcal{M}, s) = \{n \in \mathcal{M} \mid f(n) = \min_{s \in \mathcal{M}} f(s)\}.$$

1.3 Formalization

This section formalizes the concepts introduced so far and defines a variety of concepts mentioned throughout the book. It assumes an underlying combinatorial optimization problem $\mathcal{P}$ of the form

$$\begin{array}{l} \min f(\vec{x}) \quad \text{subject to} \\ C_1(\vec{x}) \\ \vdots \\ C_n(\vec{x}) \end{array}$$

where $\vec{x}$ is a vector of n (discrete) decision variables, f is an objective function $\mathcal{N}^n \rightarrow \mathcal{N}$ that associates a performance measure with a variable assignment, and $C_1, \ldots, C_n$ are constraints defining the solution space. The concepts of solutions, feasible solutions, and optimal solutions have the following (natural) meanings.

Definition 1.1 A *solution* to $\mathcal{P}$ is an assignment of values to the variables in $\vec{x}$. The set of solutions to $\mathcal{P}$ is denoted by $\mathcal{L}_{\mathcal{P}}$.

Definition 1.2 A *feasible solution* of $\mathcal{P}$ is a solution $\hat{x}$ that satisfies $C_1(\hat{x}) \wedge \ldots \wedge C_n(\hat{x})$. The set of all feasible solutions of $\mathcal{P}$ is denoted by $\tilde{\mathcal{L}_{\mathcal{P}}}$.

Definition 1.3 The set of *optimal solutions* to $\mathcal{P}$, denoted by ${\mathcal{L}_{\mathcal{P}}}^*$ is defined as

$${\mathcal{L}_{\mathcal{P}}}^* = \{s \in \tilde{\mathcal{L}_{\mathcal{P}}} \mid f(s) = \min_{k \in \tilde{\mathcal{L}_{\mathcal{P}}}} f(k)\}$$

Many local search algorithms consider only solutions that satisfy some of the constraints. This set of solutions over which the algorithm is defined is called the search space.

Definition 1.4 A *search space* for a combinatorial optimization problem $\mathcal{P}$ is a set $\hat{\mathcal{L}_{\mathcal{P}}}$ such that $\mathcal{L}_{\mathcal{P}} \subseteq \hat{\mathcal{L}_{\mathcal{P}}} \subseteq \mathcal{N}^n$. Elements of the set $\hat{\mathcal{L}_{\mathcal{P}}}$ often satisfy a subset of $\{C_1, \ldots, C_n\}$.

The concepts of neighborhood, transition graph, and local optimality are central in local search, since they define how to move from solution to solution.

Definition 1.5 A *neighborhood* is a pair $\langle \hat{\mathcal{L}_\mathcal{P}}, N \rangle$, where $\hat{\mathcal{L}_\mathcal{P}}$ is a search space and N is a mapping $N : \hat{\mathcal{L}_\mathcal{P}} \rightarrow 2^{\hat{\mathcal{L}_\mathcal{P}}}$ that defines, for each solution s, the set of adjacent solutions $N(s) \subseteq \hat{\mathcal{L}_\mathcal{P}}$. Whenever the relation $s \in N(j) \Leftrightarrow j \in N(s)$ holds, the neighborhood is said to be symmetric.

Definition 1.6 The *transition graph* $G(\hat{\mathcal{L}_\mathcal{P}}, N)$ associated to a neighborhood $\langle \hat{\mathcal{L}_\mathcal{P}}, N \rangle$ is the graph whose nodes are solutions in $\hat{\mathcal{L}_\mathcal{P}}$ and where an arc $a \rightarrow b$ exists if $b \in N(a)$. The reflexive and transitive closure of $\rightarrow$ is denoted by $\rightarrow^*$.

A solution is locally optimal if none of its neighbors have a smaller cost. Note that local optimality is always defined with respect to a specific neighborhood function.

Definition 1.7 A solution s in $\mathcal{L}_\mathcal{P}$ is *locally optimal* with respect to N if

$$f(s) \leq \min_{i \in N(s)} f(i).$$

The set of locally optimal solutions with respect to N is denoted ${\mathcal{L}_\mathcal{P}}^+$.

One of the critical issues in local search is to escape local minima, which is why local search algorithms typically feature interesting legality and selection criteria.

Definition 1.8 A *legality condition* L is a function $(2^{\hat{\mathcal{L}_\mathcal{P}}} \times \hat{\mathcal{L}_\mathcal{P}}) \rightarrow 2^{\hat{\mathcal{L}_\mathcal{P}}}$ that filters sets of solutions from the search space. A *selection rule* $S(\mathcal{M}, s)$ is a function $S : (2^{\hat{\mathcal{L}_\mathcal{P}}} \times \hat{\mathcal{L}_\mathcal{P}}) \rightarrow \hat{\mathcal{L}_\mathcal{P}}$ that picks an element s from $\mathcal{M}$ according to some strategy and decides to accept it or to select the current solution s instead.

Definition 1.9 A *local search algorithm* for $\mathcal{P}$ is a path

$$s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_k$$

in the transition graph $G(\hat{\mathcal{L}_\mathcal{P}}, N)$ for P such

$$s_{i+1} = S(L(N(s_i), s_i), s_i) \quad (1 \leq i \leq k).$$

Typically, such a local search produces a final computation state s_k that belongs to ${\mathcal{L}_\mathcal{P}}^+$ (for a given neighborhood function N). The role of heuristics and metaheuristics is to drive the search toward high-quality local optima and, ideally, those in ${\mathcal{L}_\mathcal{P}}^*$.

1.4 Properties of Neighborhood

The effectiveness of a local search algorithm critically depends upon its neighborhood. This section briefly reviews some fundamental properties of neighborhoods.

Neighborhood Size The size of a neighborhood $N : \hat{\mathcal{L}_{\mathcal{P}}} \rightarrow 2^{\hat{\mathcal{L}_{\mathcal{P}}}}$ for a solution s is the set of solutions in $N(s)$. Typically, large neighborhoods induce shorter paths to high-quality solutions but require also more time to explore. This trade-off between the length of the paths and the exploration is a key design decision. Sometimes it is preferable to select a linear neighborhood (in the size of the problem $\mathcal{P}$) over a quadratic neighborhood, even if the resulting path is longer. Sometimes, however, the resulting neighborhood is not large enough to produce high-quality solutions in reasonable time.

Neighborhood Connectivity Another fundamental property of a neighborhood is its connectivity. Informally speaking, a neighborhood is connected if there exists a path from any solution s to an optimal solution s^*. This ensures that the neighborhood is strong enough to reach optimal solutions, although the heuristic and metaheuristic may prevent the algorithm from reaching them in practice.

Definition 1.10 A neighborhood $N : \hat{\mathcal{L}_{\mathcal{P}}} \rightarrow 2^{\hat{\mathcal{L}_{\mathcal{P}}}}$ *is weakly connected* if and only if, for each solution s, there exists a path $s \rightarrow^* s^*$ to an optimal solution s^*.

Definition 1.11 A neighborhood $N : \hat{\mathcal{L}_{\mathcal{P}}} \rightarrow 2^{\hat{\mathcal{L}_{\mathcal{P}}}}$ *is optimally connected* if and only if, for each pair of solutions s_1, s_2, there exists a path $s_1 \rightarrow^* s_2$.

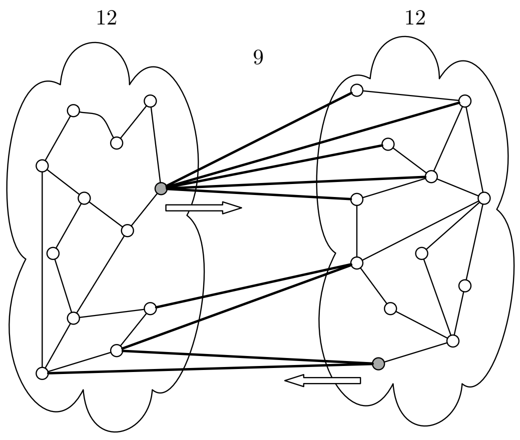
Using (weakly) connected neighborhoods brings several advantages:

- Local search algorithms typically do not need a restarting strategy to reach optimal solutions, since there exist paths leading from each solution to an optimal solution.[1]
- Randomized heuristics, where there is a nonzero probability of accepting a neighbor $k \in N(s)$ for each solution s, may be guaranteed (under certain conditions) to reach a global optimum (in the limit). In other words, connectivity is a requirement for the convergence proofs of metaheuristics such as simulated annealing.

Observe, however, that there are very effective local searches based on neighborhoods that are not connected. One of them is presented in chapter 21 of this book.

Neighborhood Constraints One of the critical issues in local search is to determine how to combine feasibility and optimality requirements. The simplest option is to maintain feasibility at all times and explore only feasible solutions in the neighborhood search. This is the approach taken

[1]A restarting strategy may still be useful for performance results.



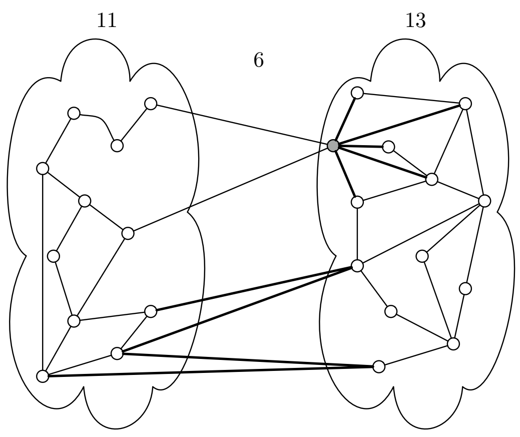


Figure 1.5: A Move from Neighborhood N'.

by neighborhood N in the graph-partitioning problem presented in Figure 1.4. The neighborhood implicitly maintains the balancing constraint, and it suffices to start with a balanced solution to maintain feasibility in the local search. In some applications, however, it is preferable to relax a subset of the constraints and explore a larger search space. Consider, for instance, the neighborhood N' for graph partitioning, in which a single vertex is relocated:

$$N'(\langle P_1, P_2\rangle) = \{\langle P_1 \setminus \{a\}, P_2 \cup \{a\}\rangle \ \mid \ a \in P_1\} \ \cup \ \{\langle P_1 \cup \{b\}, P_2 \setminus \{b\}\rangle \ \mid \ b \in P_2\}.$$

Figure 1.5 illustrates such a move that decreases the cost of the partition from 9 to 6.

The neighbor N' induces the local search to explore infeasible solutions or, in other words, unbalanced partitions. It is thus important to drive the search not only toward high-quality solutions, but also toward feasibility. Once again, there are many different ways to approach this issue; several of them are illustrated in this book. One of them entails designing an objective function that combines feasibility and optimality components. For instance, for a neighborhood N' in graph partitioning, the algorithm may use the objective

$$\alpha \cdot \#\{(x, y) \in E \ \mid \ x \in P_1 \ \& \ y \ \in P_2\} + \beta \cdot (\#\{x \in E \ \mid x \in P_1\} - \frac{n}{2})^2$$

for some well-chosen values of α and β and $\#V = n$. The left member of the objective captures the cost of the partition, while the right member penalizes its imbalance.