DM841
DISCRETE OPTIMIZATION

Part 2 – Heuristics
Local Search Theory

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark
Outline

1. Local Search Revisited
   - Search Space Properties
   - Neighborhoods Formalized
   - Distances
   - Landscape Characteristics

2. Metaheuristics
Outline

1. Local Search Revisited
   - Search Space Properties
   - Neighborhoods Formalized
   - Distances
   - Landscape Characteristics

2. Metaheuristics
Outline

1. Local Search Revisited
   Search Space Properties
   Neighborhoods Formalized
   Distances
   Landscape Characteristics

2. Metaheuristics
Neighborhood function $\mathcal{N}_\pi : S_\pi \to 2^{S_\pi}$

Also defined as: $\mathcal{N} : S \times S \to \{T, F\}$ or $\mathcal{N} \subseteq S \times S$

- neighborhood (set) of candidate solution $s$: $N(s) := \{s' \in S \mid \mathcal{N}(s, s')\}$
- neighborhood size is $|N(s)|$
- neighborhood is symmetric if: $s' \in N(s) \Rightarrow s \in N(s')$
- neighborhood graph of $(S, N, \pi)$ is a directed graph: $G_{\mathcal{N}_\pi} := (V, A)$ with $V = S_\pi$ and $(uv) \in A \Leftrightarrow v \in N(u)$
  (if symmetric neighborhood $\rightsquigarrow$ undirected graph)

Notation: $N$ when set, $\mathcal{N}$ when collection of sets or function
A neighborhood function is also defined by means of an operator (aka move). An operator $\Delta$ is a collection of operator functions $\delta : S \rightarrow S$ such that

$$s' \in N(s) \implies \exists \delta \in \Delta, \delta(s) = s'$$

**Definition**

$k$-exchange neighborhood: candidate solutions $s, s'$ are neighbors iff $s$ differs from $s'$ in at most $k$ solution components

**Examples:**

- 1-exchange (flip) neighborhood for SAT
  (solution components = single variable assignments)
- 2-exchange neighborhood for TSP
  (solution components = edges in given graph)
Definition:

- **Local minimum**: search position without improving neighbors wrt given evaluation function $f$ and neighborhood $\mathcal{N}$, i.e., position $s \in S$ such that $f(s) \leq f(s')$ for all $s' \in \mathcal{N}(s)$.

- **Strict local minimum**: search position $s \in S$ such that $f(s) < f(s')$ for all $s' \in \mathcal{N}(s)$.

- **Local maxima** and **strict local maxima**: defined analogously.
Note:

- Local search implements a walk through the neighborhood graph.

- Procedural versions of init, step and terminate implement sampling from respective probability distributions.

- Local search algorithms can be described as Markov processes: behavior in any search state \( \{s, m\} \) depends only on current position \( s \) and higher order MP if (limited) memory \( m \).
Search step (or move):
pair of search positions \( s, s' \) for which
\( s' \) can be reached from \( s \) in one step, i.e., \( N(s, s') \) and
\[ \text{step}(\{s, m\}, \{s', m'\}) > 0 \]
for some memory states \( m, m' \in M \).

- **Search trajectory**: finite sequence of search positions \( \langle s_0, s_1, \ldots, s_k \rangle \) such that \( (s_{i-1}, s_i) \) is a search step for any \( i \in \{1, \ldots, k\} \)
and the probability of initializing the search at \( s_0 \)
is greater than zero, i.e., \( \text{init}(\{s_0, m\}) > 0 \)
for some memory state \( m \in M \).

- **Search strategy**: specified by \( \text{init} \) and \( \text{step} \) function; to some extent independent of problem instance and other components of LS algorithm.
  - random
  - based on evaluation function
  - based on memory
Outline

1. Local Search Revisited
   - Search Space Properties
   - Neighborhoods Formalized
   - Distances
   - Landscape Characteristics

2. Metaheuristics
Neighborhood Operator

Goal: providing a formal description of neighborhood functions for the three main solution representations:
- **Permutation**
  - linear permutation: Single Machine Total Weighted Tardiness Problem
  - circular permutation: Traveling Salesman Problem
- **Assignment**: SAT, CSP
- **Set, Partition**: Max Independent Set

A neighborhood function \( N : S \rightarrow 2^S \) is also defined through an operator. An operator \( \Delta \) is a collection of operator functions \( \delta : S \rightarrow S \) such that

\[
\forall s' \in N(s) \iff \exists \delta \in \Delta \mid \delta(s) = s'
\]
**Permutations**

\( S_n \) indicates the set all permutations of the numbers \( \{1, 2, \ldots, n\} \)

\((1, 2 \ldots, n)\) is the identity permutation \( \iota \).

If \( \pi \in \Pi(n) \) and \( 1 \leq i \leq n \) then:

- \( \pi_i \) is the element at position \( i \)
- \( pos_\pi(i) \) is the position of element \( i \)

Alternatively, a permutation is a bijective function \( \pi(i) = \pi_i \)

The permutation product \( \pi \cdot \pi' \) is the composition \( (\pi \cdot \pi')_i = \pi'(\pi(i)) \)

For each \( \pi \) there exists a permutation such that \( \pi^{-1} \cdot \pi = \iota \)

\( \pi^{-1}(i) = pos_\pi(i) \)

\[ \Delta_N \subset S_n \]
**Linear Permutations**

**Swap operator**

\[ \Delta_S = \{ \delta_S^i \mid 1 \leq i \leq n \} \]

\[ \delta_S^i(\pi_1 \ldots \pi_i \pi_{i+1} \ldots \pi_n) = (\pi_1 \ldots \pi_{i+1} \pi_i \ldots \pi_n) \]

**Interchange operator**

\[ \Delta_X = \{ \delta_X^{ij} \mid 1 \leq i < j \leq n \} \]

\[ \delta_X^{ij}(\pi) = (\pi_1 \ldots \pi_i \pi_{i+1} \ldots \pi_j \pi_{i+1} \ldots \pi_{j-1} \pi_i \pi_{j+1} \ldots \pi_n) \]

\((\equiv \text{set of all transpositions})\)

**Insert operator**

\[ \Delta_I = \{ \delta_I^{ij} \mid 1 \leq i \leq n, 1 \leq j \leq n, j \neq i \} \]

\[ \delta_I^{ij}(\pi) = \begin{cases} (\pi_1 \ldots \pi_{i-1} \pi_i \pi_{i+1} \ldots \pi_j \pi_j \pi_{j+1} \ldots \pi_n) & \text{if } i < j \\ (\pi_1 \ldots \pi_j \pi_i \pi_{j+1} \ldots \pi_{i-1} \pi_i \pi_{i+1} \ldots \pi_n) & \text{if } i > j \end{cases} \]
Circular Permutations

Reversal (2-edge-exchange)

\[ \Delta_R = \{ \delta_{ij}^R | 1 \leq i < j \leq n \} \]

\[ \delta_{ij}^R(\pi) = (\pi_1 \ldots \pi_{i-1} \pi_j \ldots \pi_i \pi_{j+1} \ldots \pi_n) \]

Block moves (3-edge-exchange)

\[ \Delta_B = \{ \delta_{ijk}^B | 1 \leq i < j < k \leq n \} \]

\[ \delta_{ijk}^B(\pi) = (\pi_1 \ldots \pi_{i-1} \pi_j \ldots \pi_k \pi_i \ldots \pi_{j-1} \pi_{k+1} \ldots \pi_n) \]

Short block move (Or-edge-exchange)

\[ \Delta_{SB} = \{ \delta_{ij}^{SB} | 1 \leq i < j \leq n \} \]

\[ \delta_{ij}^{SB}(\pi) = (\pi_1 \ldots \pi_{i-1} \pi_j \pi_{j+1} \pi_{j+2} \pi_i \ldots \pi_{j-1} \pi_{j+3} \ldots \pi_n) \]
Assignments

An assignment can be represented as a mapping
\( \sigma : \{X_1 \ldots X_n\} \rightarrow \{v : v \in D, |D| = k\} : \)

\( \sigma = \{X_i = v_i, X_j = v_j, \ldots\} \)

One-exchange operator

\( \Delta_{1E} = \{\delta_{1E}^{il} | 1 \leq i \leq n, 1 \leq l \leq k\} \)

\( \delta_{1E}^{il}(\sigma) = \{\sigma' : \sigma'(X_i) = v_l \text{ and } \sigma'(X_j) = \sigma(X_j) \forall j \neq i\} \)

Two-exchange operator

\( \Delta_{2E} = \{\delta_{2E}^{ij} | 1 \leq i < j \leq n\} \)

\( \delta_{2E}^{ij}(\sigma) = \{\sigma' : \sigma'(X_i) = \sigma(X_j), \sigma'(X_j) = \sigma(X_i) \text{ and } \sigma'(X_l) = \sigma(X_l) \forall l \neq i, j\} \)
Partitioning

An assignment can be represented as a partition of objects selected and not selected $s : \{X\} \rightarrow \{C, \bar{C}\}$
(it can also be represented by a bit string)

**One-addition** operator

$$\Delta_{1E} = \{\delta_{1E}^v | v \in \bar{C}\}$$

$$\delta_{1E}^v(s) = \{s : C' = C \cup v \text{ and } \bar{C}' = \bar{C} \setminus v\}$$

**One-deletion** operator

$$\Delta_{1E} = \{\delta_{1E}^v | v \in C\}$$

$$\delta_{1E}^v(s) = \{s : C' = C \setminus v \text{ and } \bar{C}' = \bar{C} \cup v\}$$

**Swap** operator

$$\Delta_{1E} = \{\delta_{1E}^v | v \in C, u \in \bar{C}\}$$

$$\delta_{1E}^v(s) = \{s : C' = C \cup u \setminus v \text{ and } \bar{C}' = \bar{C} \cup v \setminus u\}$$
Outline

1. Local Search Revisited
   - Search Space Properties
   - Neighborhoods Formalized
   - Distances
   - Landscape Characteristics

2. Metaheuristics
Distances

Set of paths in $\mathcal{N}$ with $s, s' \in S$:

$$\Phi(s, s') = \{(s_1, \ldots, s_h) | s_1 = s, s_h = s' \forall i : 1 \leq i \leq h - 1, \langle s_i, s_{i+1} \rangle \in E_N\}$$

If $\phi = (s_1, \ldots, s_h) \in \Phi(s, s')$ let $|\phi| = h$ be the length of the path; then the distance between any two solutions $s, s'$ is the length of shortest path between $s$ and $s'$ in $\mathcal{N}$:

$$d_{\mathcal{N}}(s, s') = \min_{\phi \in \Phi(s,s')} |\phi|$$

$diam(\mathcal{N}) = \max\{d_{\mathcal{N}}(s, s') | s, s' \in S\} \ (= \text{maximal distance between any two candidate solutions})$

$(= \text{worst-case lower bound for number of search steps required for reaching (optimal) solutions})$

**Note:** with permutations it is easy to see that:

$$d_{\mathcal{N}}(\pi, \pi') = d_{\mathcal{N}}(\pi^{-1} \cdot \pi', \nu)$$
Distances for Linear Permutation Representations

- Swap neighborhood operator
  Computable in $O(n^2)$ by the precedence based distance metric:
  \[ d_S(\pi, \pi') = \#\{\langle i, j \rangle | 1 \leq i < j \leq n, pos_{\pi'}(\pi_j) < pos_{\pi'}(\pi_i) \} \]
  \[ \text{diam}(G_N) = n(n - 1)/2 \]

- Interchange neighborhood operator
  Computable in $O(n) + O(n)$ since
  \[ d_X(\pi, \pi') = d_X(\pi^{-1} \cdot \pi', \iota) = n - c(\pi^{-1} \cdot \pi') \]
  \[ c(\pi) \] is the number of disjoint cycles that decompose a permutation.
  \[ \text{diam}(G_NX) = n - 1 \]

- Insert neighborhood operator
  Computable in $O(n) + O(n \log(n))$ since
  \[ d_I(\pi, \pi') = d_I(\pi^{-1} \cdot \pi', \iota) = n - |lis(\pi^{-1} \cdot \pi')| \] where \( lis(\pi) \) denotes the length of the longest increasing subsequence.
  \[ \text{diam}(G_NI) = n - 1 \]
Distances for Circular Permutation Representations

- Reversal neighborhood operator
  sorting by reversal is known to be NP-hard
  surrogate in TSP: bond distance

- Block moves neighborhood operator
  unknown whether it is NP-hard but there does not exist a proved polynomial-time algorithm
Distances for Assignment Representations

- Hamming Distance

- An assignment can be seen as a partition of $n$ in $k$ mutually exclusive non-empty subsets

One-exchange neighborhood operator

The partition-distance $d_{1E}(\mathcal{P}, \mathcal{P}')$ between two partitions $\mathcal{P}$ and $\mathcal{P}'$ is the minimum number of elements that must be moved between subsets in $\mathcal{P}$ so that the resulting partition equals $\mathcal{P}'$.

The partition-distance can be computed in polynomial time by solving an assignment problem. Given the assignment matrix $M$ where in each cell $(i,j)$ it is $|S_i \cap S'_j|$ with $S_i \in \mathcal{P}$ and $S'_j \in \mathcal{P}'$ and defined $A(\mathcal{P}, \mathcal{P}')$ the assignment of maximal sum then it is $d_{1E}(\mathcal{P}, \mathcal{P}') = n - A(\mathcal{P}, \mathcal{P}')$
Example: Search space size and diameter for SAT

SAT instance with $n$ variables, 1-flip neighborhood:
$G_N = n$-dimensional hypercube; diameter of $G_N = n$. 
Example: Search space size and diameter for the TSP

- Search space size $= (n - 1)!/2$

- Insert neighborhood
  - size $= (n - 3)n$
  - diameter $= n - 2$

- 2-exchange neighborhood
  - size $= \binom{n}{2} = n \cdot (n - 1)/2$
  - diameter in $[n/2, n - 2]$

- 3-exchange neighborhood
  - size $= \binom{n}{3} = n \cdot (n - 1) \cdot (n - 2)/6$
  - diameter in $[n/3, n - 1]$
Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two different neighborhood functions for the same instance $(S, f, \pi)$ of a combinatorial optimization problem. If for all solutions $s \in S$ we have $N_1(s) \subseteq N_2(s)$ then we say that $\mathcal{N}_2$ dominates $\mathcal{N}_1$.

Example:

In TSP, 1-insert is dominated by 3-exchange. (1-insert corresponds to 3-exchange and there are 3-exchanges that are not 1-insert)
Search Landscape

Given:

- Problem instance $\pi$
- Search space $S_\pi$
- Neighborhood function $N : S \subseteq 2^S$
- Evaluation function $f_\pi : S \rightarrow \mathbb{R}$

Definition:

The **search landscape** $L$ is the vertex-labeled neighborhood graph given by the triplet $\mathcal{L} = \langle S_\pi, N_\pi, f_\pi \rangle$. 
Transition Graph of Iterative Improvement

Given $\mathcal{L} = \langle S_\pi, N_\pi, f_\pi \rangle$, the transition graph of iterative improvement is a directed acyclic subgraph obtained from $\mathcal{L}$ by deleting all arcs $(i, j)$ for which it holds that the cost of solution $j$ is worse than or equal to the cost of solution $i$.

It can be defined for other algorithms as well and it plays a central role in the theoretical analysis of proofs of convergence.
Ideal visualization of landscapes principles

- Simplified landscape representation
- Tabu Search
- Guided Local Search
- Iterated Local Search
- Evolutionary Alg.
The behavior and performance of an LS algorithm on a given problem instance crucially depends on properties of the respective search landscape.

Simple properties:

- search space size $|S|$
- reachability: solution $j$ is reachable from solution $i$ if neighborhood graph has a path from $i$ to $j$. 
  - strongly connected neighborhood graph
  - weakly optimally connected neighborhood graph
- distance between solutions
- neighborhood size (ie, degree of vertices in neigh. graph)
- cost of fully examining the neighborhood
- relation between different neighborhood functions (if $N_1(s) \subseteq N_2(s)$ for all $s \in S$ then $N_2$ dominates $N_1$)
1. Local Search Revisited
   Search Space Properties
   Neighborhoods Formalized
   Distances
   Landscape Characteristics

2. Metaheuristics
Other Search Space Properties

- number of (optimal) solutions $|S'|$, solution density $|S'|/|S|$
- distribution of solutions within the neighborhood graph
Phase Transition for 3-SAT

Random instances $\sim m$ clauses of $n$ uniformly chosen variables

![Graphs showing phase transition for 3-SAT](image)
Classification of search positions

<table>
<thead>
<tr>
<th>position type</th>
<th>&gt;</th>
<th>=</th>
<th>&lt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLMIN (strict local min)</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LMIN (local min)</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>IPLAT (interior plateau)</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>SLOPE</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>LEDGE</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LMAX (local max)</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SLMAX (strict local max)</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

“+” = present, “−” absent; table entries refer to neighbors with larger (“>”), equal (“=”), and smaller (“<”) evaluation function values
Other Search Space Properties

- plateux
- barrier and basins
Outline

1. Local Search Revisited
   - Search Space Properties
   - Neighborhoods Formalized
   - Distances
   - Landscape Characteristics

2. Metaheuristics
Escaping Local Optima

Possibilities:

- **Restart**: re-initialize search whenever a local optimum is encountered. (Often rather ineffective due to cost of initialization.)

- **Non-improving steps**: in local optima, allow selection of candidate solutions with equal or worse evaluation function value, *e.g.*, using minimally worsening steps. (Can lead to long walks in *plateaus*, *i.e.*, regions of search positions with identical evaluation function.)

- **Diversify the neighborhood**: multiple, variable-size, rich (while still preserving incremental algorithmics insights)

*Note*: None of these mechanisms is guaranteed to always escape effectively from local optima.
Diversification vs Intensification

- **Intensification**: aims at greedily increasing solution quality, e.g., by exploiting the evaluation function.

- **Diversification**: aims at preventing search stagnation, that is, the search process getting trapped in confined regions.

- Goal-directed and randomized components of LS strategy need to be balanced carefully.

Examples:

- Iterative Improvement (II): *intensification* strategy.
- Uninformed Random Walk/Picking (URW/P): *diversification* strategy.

Balanced combination of intensification and diversification mechanisms forms the basis for advanced LS methods.
‘Simple’ Metaheuristics

Goal:
Effectively escape from local minima of given evaluation function.

General approach:
For fixed neighborhood, use step function that permits worsening search steps.

Specific methods:
- Stochastic Local Search
- Simulated Annealing
- (Guided Local Search)
- Tabu Search
- Iterated Local Search
- Variable Neighborhood Search
- Evolutionary Algorithms