Lecture 7

Very Large Scale Neighborhoods Techniques.
Variable Neighborhood Search.

Marco Chiarandini

Outline

1. Very Large Scale Neighborhoods
   - Variable Depth Search
   - Ejection Chains
   - Dynasearch
   - Assignment Neighborhoods
   - Cyclic Exchange Neighborhoods

2. Variable Neighborhood Search
   - The p-median Problem

3. Simple Metaheuristics

4. Randomized Iterative Improvement

Very Large Scale Neighborhoods

Key idea: use very large neighborhoods that can be searched very efficiently (preferably in polynomial time) or are searched heuristically

Small neighborhoods:
- might be short-sighted
- need many steps to traverse the search space

Large neighborhoods:
- introduce large modifications to reach higher quality solutions
- allows to traverse the search space in few steps

Very large scale neighborhood search:
- define an exponentially large neighborhood (though, $O(n^3)$ might already be large)
- define a polynomial time search algorithm to search the neighborhood (= solve the neighborhood search problem, NSP). This might be:
  - exact (leads to a best improvement strategy)
  - heuristic (some improving moves might be missed)
Examples of VLSN Search (see paper, Ahuja, Ergun, Orlin, Punnen, 2002):

- based on concatenation of simple moves
  - Variable Depth Search (TSP)
  - Ejection Chains
- based on Dynamic Programming or Network Flows
  - Dynasearch (ex. SMTWTP)
  - Assignment based neighborhoods (ex. TSP)
  - Cyclic exchange neighborhood (ex. VRP)
- based on polynomially solvable special cases of hard combinatorial optimization problems
  - Pyramidal tours
  - Halin Graphs

⇒ Idea: turn a special case into a neighborhood
VLSN allows to use the literature on polynomial time algorithms

---

**Variable Depth Search**

- **Key idea:** Complex steps in large neighborhoods = variable-length sequences of simple steps in small neighborhood.
- Use various feasibility restrictions on selection of simple search steps to limit time complexity of constructing complex steps.
- Perform Iterative Improvement w.r.t. complex steps.

**Variable Depth Search (VDS):**

determine initial candidate solution \( s \)

\[ \hat{s} := s \]

While \( s \) is not locally optimal:

- Repeat:
  - select best feasible neighbor \( t \)
  - If \( g(t) < g(\hat{s}) \): \( \hat{s} := t \)
  - Until construction of complex step has been completed

\[ s := \hat{s} \]

---

**Example: The Lin-Kernighan (LK) Algorithm for the TSP (1)**

- Complex search steps correspond to sequences of 2-exchange steps and are constructed from sequences of Hamiltonian paths
- \( \delta \)-path: Hamiltonian path \( p + 1 \) edge connecting one end of \( p \) to interior node of \( p \)

![Image of LK exchange step](image)

- Start with Hamiltonian path \( (u, \ldots, v) \):

  ![Image of initial Hamiltonian path](image)

- Obtain \( \delta \)-path by adding an edge \( (v, w) \):

  ![Image of \( \delta \)-path](image)

- Break cycle by removing edge \( (w, v') \):

  ![Image of cycle break](image)

- Note: Hamiltonian path can be completed into Hamiltonian cycle by adding edge \( (v', u) \):

  ![Image of cycle completion](image)
### Construction of complex LK steps:

1. start with current candidate solution (Hamiltonian cycle) $s$; set $t^* := s$;
   set $p := s$
2. obtain $\delta$-path $p'$ by replacing one edge in $p$
3. consider Hamiltonian cycle $t$ obtained from $p$ by (uniquely) defined edge exchange
4. if $w(t) < w(t^*)$ then set $t^* := t$; $p := p'$; go to step 2
5. else accept $t^*$ as new current candidate solution $s$

**Note:** This can be interpreted as sequence of 1-exchange steps that alternate between $\delta$-paths and Hamiltonian cycles.

### Additional mechanisms used by LK algorithm:

- **Pruning exact rule:** If a sequence of numbers has a positive sum, there is a cyclic permutation of these numbers such that every partial sum is positive.
  $\Rightarrow$ need to consider only gains whose partial sum remains positive
- **Tabu restriction:** Any edge that has been added cannot be removed and any edge that has been removed cannot be added in the same LK step.
  **Note:** This limits the number of simple steps in a complex LK step.
- **Limited form of backtracking** ensures that local minimum found by the algorithm is optimal w.r.t. standard 3-exchange neighborhood
  - (For further details, see original article)

### Graph Partitioning Problem

- **Given:** an undirected graph $G = (V,E)$ with costs on its edges
- **Task:** partition the vertices of $G$ into subsets no larger than a given maximum size so as to minimize the total cost of the edges in the cut

### Note:

Variable depth search algorithms have been very successful for other problems, including:

- the Graph Partitioning Problem [Kernighan and Lin, 1970];
- the Unconstrained Binary Quadratic Programming Problem [Merz and Freisleben, 2002];
- the Generalized Assignment Problem [Yagiura et al., 1999].
### Ejection Chains

- Attempt to use large neighborhoods without examining them exhaustively.
- Sequences of successive steps each influenced by the precedent and determined by myopic choices.
- Limited in length.
- Local optimality in the large neighborhood is not guaranteed.

**Example (on TSP):** successive 2-exchanges where each exchange involves one edge of the previous.

**Example (on GCP):** successive 1-exchanges: a vertex $v_1$ changes color from $\varphi(v_1) = c_1$ to $c_2$, in turn forcing some vertex $v_2$ with color $\varphi(v_2) = c_2$ to change to another color $c_3$ (which may be different or equal to $c_1$) and again forcing a vertex $v_3$ with color $\varphi(v_3) = c_3$ to change to color $c_4$.

### Dynasearch

- Iterative improvement method based on building complex search steps from combinations of simple search steps.
- Simple search steps constituting any given complex step are required to be mutually independent, i.e., do not interfere with each other w.r.t. effect on evaluation function and feasibility of candidate solutions.

**Example:** Independent 2-exchange steps for the TSP:

\[
\begin{align*}
&u_1, u_2, u_3, u_4, \ldots, u_{n-1}, u_n, u_{n+1} \\
\Rightarrow &
\end{align*}
\]

*Therefore:* Overall effect of complex search step = sum of effects of constituting simple steps; complex search steps maintain feasibility of candidate solutions.

**Key idea:** Efficiently find optimal combination of mutually independent simple search steps using *Dynamic Programming*.

### Assignment Neighborhoods

- Neighborhood defined by finding a minimum cost matching on a (non-)bipartite improvement graph.

**Example:** TSP

Neighborhood: Eject $k$ nodes and reinsert them optimally.

### Cyclic Exchange Neighborhoods

- Possible for problems where solution can be represented as form of partitioning.
- Definition of a partitioning problem:
  Given a set $W$ of $n$ elements, a set of subsets $T$ of $W$, $T = \{T_1, T_2, \ldots, T_k\}$ such that $W = T_1 \cup \ldots \cup T_k$ and $T_k \cap T'_k = \emptyset$ and a cost function $c : T \rightarrow \mathbb{R}$:

\[
\begin{align*}
\min c(T) &= \sum_{k=1}^{k} c(T_k) \\
\text{s.t. } &T \text{ is a partition of } W
\end{align*}
\]

**Cyclic exchange:**
Neighborhood search

▶ Define an improvement graph
▶ Solve the relative
  Subset Disjoint Negative Cost Cycle Problem
  Subset Disjoint Minimum Cost Cycle Problem

Outline

1. Very Large Scale Neighborhoods
   Variable Depth Search
   Ejection Chains
   Dynasearch
   Assignment Neighborhoods
   Cyclic Exchange Neighborhoods

2. Variable Neighborhood Search
   The p-median Problem

3. Simple Metaheuristics

4. Randomized Iterative Improvement

Variable Neighborhood Search (VNS)

Variable Neighborhood Search is an SLS method that is based on the systematic change of the neighborhood during the search.

Central observations

▶ a local minimum w.r.t. one neighborhood structure is not necessarily locally minimal w.r.t. another neighborhood structure
▶ a global optimum is locally optimal w.r.t. all neighborhood structures

Principle: change the neighborhood during the search

Several adaptations of this central principle

▶ (Basic) Variable Neighborhood Descent (VND)
▶ Variable Neighborhood Search (VNS)
▶ Reduced Variable Neighborhood Search (RVNS)
▶ Variable Neighborhood Decomposition Search (VNDS)
▶ Skewed Variable Neighborhood Search (SVNS)

Notation

▶ $N_k, k = 1, 2, \ldots, k_{\text{max}}$ is a set of neighborhood structures
▶ $N_k(s)$ is the set of solutions in the $k$-th neighborhood of $s$
How to generate the various neighborhood structures?

- for many problems different neighborhood structures (local searches) exist / are in use
- change parameters of existing local search algorithms
- use k-exchange neighborhoods; these can be naturally extended
- many neighborhood structures are associated with distance measures; in this case increase the distance

**Basic Variable Neighborhood Descent (BVND)**

**Procedure VND**

**input**: \( N_k, k = 1, 2, \ldots, k_{\text{max}} \), and an initial solution \( s \)

**output**: a local optimum \( s \) for \( N_k, k = 1, 2, \ldots, k_{\text{max}} \)

\( k \leftarrow 1 \)

repeat

\( s' \leftarrow \text{IterativeImprovement}(s, N_k) \)

if \( g(s') < g(s) \) then

\( s \leftarrow s' \)

\( (k \leftarrow 1) \)

else

\( k \leftarrow k + 1 \)

until \( k = k_{\text{max}} \);

**Variable Neighborhood Descent (VND)**

**Procedure VND**

**input**: \( N_k, k = 1, 2, \ldots, k_{\text{max}} \), and an initial solution \( s \)

**output**: a local optimum \( s \) for \( N_k, k = 1, 2, \ldots, k_{\text{max}} \)

\( k \leftarrow 1 \)

repeat

\( s' \leftarrow \text{IterativeImprovement}(s, N_k) \)

if \( g(s') < g(s) \) then

\( s \leftarrow s' \)

\( (k \leftarrow 1) \)

else

\( k \leftarrow k + 1 \)

until \( k = k_{\text{max}} \);

- Final solution is locally optimal w.r.t. all neighborhoods
- First improvement may be applied instead of best improvement
- Typically, order neighborhoods from smallest to largest
- If iterative improvement algorithms II\( k \), \( k = 1, \ldots, k_{\text{max}} \) are available as black-box procedures
  - order black-boxes
  - apply them in the given order
  - possibly iterate starting from the first one
  - order chosen by: solution quality and speed
Example

VND for single-machine total weighted tardiness problem

- Candidate solutions are permutations of job indexes
- Two neighborhoods: swap and insert
- Influence of different starting heuristics also considered

<table>
<thead>
<tr>
<th>Initial</th>
<th>Swap</th>
<th>Insert</th>
<th>Swap+Insert</th>
<th>Insert+Swap</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDD</td>
<td>0.62</td>
<td>0.140</td>
<td>1.19</td>
<td>0.24</td>
</tr>
<tr>
<td>MDD</td>
<td>0.65</td>
<td>0.078</td>
<td>1.31</td>
<td>0.40</td>
</tr>
</tbody>
</table>

$\Delta_{\text{avg}}$ deviation from best-known solutions, averaged over 100 instances

Basic Variable Neighborhood Search (VNS)

Procedure BVNS

**input**: $\mathcal{N}_k, k = 1, 2, \ldots, k_{\text{max}}$, and an initial solution $s$

**output**: a local optimum $s$ for $\mathcal{N}_k, k = 1, 2, \ldots, k_{\text{max}}$

```
repeat
    $k \leftarrow 1$
    repeat
        $s' \leftarrow \text{RandomPicking}(s, \mathcal{N}_k)$
        $s'' \leftarrow \text{IterativeImprovement}(s', \mathcal{N}_1)$
        if $g(s'') < g(s)$ then
            $s \leftarrow s''$
            $k \leftarrow 1$
        else
            $k \leftarrow k + 1$
    until $k = k_{\text{max}}$
until Termination Condition
```

To decide:

- which neighborhoods
- how many
- which order
- which change strategy

- Extended version: parameters $k_{\text{min}}$ and $k_{\text{step}}$; set $k \leftarrow k_{\text{min}}$ and increase by $k_{\text{step}}$ if no better solution is found (achieves diversification)

Extensions (1)

Reduced Variable Neighborhood Search (RVNS)

- same as VNS except that no IterativeImprovement procedure is applied
- only explores randomly different neighborhoods
- can be faster than standard local search algorithms for reaching good quality solutions
Variable Neighborhood Decomposition Search (VNDS)

- same as in VNS but in Iterative Improvement all solution components are kept fixed except $k$ randomly chosen
- Iterative Improvement is applied on the $k$ unfixed components

Iterative Improvement can be substituted by exhaustive search up to a maximum size $b$ (parameter) of the problem.

Extension (3)

Skewed Variable Neighborhood Search (SVNS)

- Derived from VNS
- Accept $s \leftarrow s''$ when $s''$ is worse
  - according to some probability
  - skewed VNS: accept if
    \[
    g(s'') - \alpha \cdot d(s, s'') < g(s)
    \]
    $d(s, s'')$ measure the distance between solutions
    (underlying idea: avoiding degeneration to multi-start)

The p-median Problem

- Given:
  - a set $U$ of locations for $n$ users
  - a set $F$ of locations of $m$ facilities
  - a distance matrix $D = [d_{ij}] \in \mathbb{R}^{n \times m}$
- Task: Select $p$ locations of $F$ where to install facilities such that the sum of the distances of each user to its closest installed facility is minimized, i.e.,

\[
\min \sum_{i \in U} \min_{j \in F} d_{ij} \quad J \subseteq F \text{ and } |J| = p
\]
‘Simple’ SLS Methods

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:
- Randomized Iterative Improvement
- Probabilistic Iterative Improvement
- Simulated Annealing
- Tabu Search
- Dynamic Local Search

Outline
1. Very Large Scale Neighborhoods
   - Variable Depth Search
   - Ejection Chains
   - Dynasearch
   - Assignment Neighborhoods
   - Cyclic Exchange Neighborhoods
2. Variable Neighborhood Search
   - The p-median Problem
3. Simple Metaheuristics
4. Randomized Iterative Improvement

Walk SAT & Min-Conflict Heuristics

```plaintext
procedure WalkSAT (F, maxTries, maxSteps, sימי)
    input: CNF formula F, positive integers maxTries and maxSteps,
           heuristic function sימי
    output: model of F or 'no solution found'
    for try := 1 to maxTries do
        a := randomly chosen assignment of the variables in formula F;
        for step := 1 to maxSteps do
            if a satisfies F then return a end
            c := randomly selected clause unsatisfied under a;
            x := variable selected from c according to heuristic function sימי;
            a := a with x flipped;
        end
    end
    return 'no solution found'
end WalkSAT
```

Figure 6.3 The WalkSAT algorithm family. All random selections are according to a uniform probability distribution over the underlying sets; WalkSAT algorithms differ in the variable selection heuristic sימי.

Attribute Based Hill Climber

- attributes are solution elements that change in a move
- each attribute has associated a value:
  - the value of the best solution visited that contains it
  - infinity, otherwise
- at each step, a solution in N is acceptable iff it contains an attribute that has never been seen in a solution of such high quality before

$$N^i(s) = \{s' \in N(s) : \exists \alpha \in s' \text{ s.t. } f(s') < \phi(\alpha)\}$$

where

$$\phi(\alpha) = \begin{cases} 
\infty & \text{if } V \cap S^\alpha = \emptyset \\
\min\{f(s) : s \in V \cap S^\alpha\} & \text{otherwise}
\end{cases}$$

with V set of visited solutions and S^α set of solutions that contains α.
Randomized Iterative Improvement

Key idea: In each search step, with a fixed probability perform an uninformed random walk step instead of an iterative improvement step.

Randomized Iterative Improvement (RII):

determine initial candidate solution $s$

While termination condition is not satisfied:

With probability $wp$

choose a neighbor $s'$ of $s$ uniformly at random

Otherwise:

choose a neighbor $s'$ of $s$ such that $g(s') < g(s)$ or,

if no such $s'$ exists, choose $s'$ such that $g(s')$ is minimal

$s := s'$

Example: Randomized Iterative Improvement for GCP

procedure GUWGCP($F, wp, maxSteps$)

input: a graph $G$ and $k$, probability $wp$, integer $maxSteps$

output: a proper coloring $\varphi$ for $G$ or $\emptyset$

choose coloring $\varphi$ of $G$ uniformly at random;

steps := 0;

while not ($\varphi$ is not proper) and (steps < $maxSteps$) do

with probability $wp$ do

select $v$ in $V$ and $c$ in $\Gamma$ uniformly at random;

otherwise

select $v$ and $c$ in $V^c$ and $\Gamma$ uniformly at random from those that decrease of number of unsat. edge constr. is max.;

change color of $v$ in $\varphi$;

steps := steps + 1;

end

if $\varphi$ is proper for $G$ then return $\varphi$

else return $\emptyset$

end

GUWGCP

Note:

▶ No need to terminate search when local minimum is encountered

Instead: Bound number of search steps or CPU time from beginning of search or after last improvement.

▶ Probabilistic mechanism permits arbitrary long sequences of random walk steps

Therefore: When run sufficiently long, RII is guaranteed to find (optimal) solution to any problem instance with arbitrarily high probability.

▶ A variant of RII has successfully been applied to SAT (GWSAT algorithm)

▶ A variant of GUWSAT, GWSAT [Selman et al., 1994], was at some point state-of-the-art for SAT.

▶ Generally, RII is often outperformed by more complex LS methods.
Novelty

Key idea: combine Randomized Iterative Improvement with Min-Conflicts

Example on GCP