Variable neighborhood search for the travelling salesman problem and its variants

Nenad Mladenović,

Brunel University-West London UK

FON, MI-SANU, University of Belgrade

- Introduction

- Variable neighborhood search algorithms
  - Variable metric algorithm;
  - VND;
  - Reduced VNS;
  - Basic VNS;
  - Skewed VNS;
  - General VNS;
  - More extensions
  - VN Decomposition search;
  - Primal-Dual VNS

Balcor 2011, Thessaloniki, Greece
Introduction

- Optimization problems (continuous-discrete, static-dynamic, deterministic-stochastic)
- Exact methods, Heuristics, Simulation (Monte-Carlo)
- Classical heuristics (constructive (greedy add, greedy drop), relaxation based, space reduction, local search, Lagrangian heuristics,...)
- Metaheuristics (Simulated annealing, Tabu search, GRASP, Variable neighborhood search, Genetic search, Evolutionary methods, Particle swarm optimization, ....)
- Variable neighborhood search
Optimization problems

A deterministic optimization problem may be formulated as

$$\min \{ f(x) | x \in X, X \subseteq S \},$$  \hspace{1cm} (1)

- where $S$, $X$, $x$ and $f$ denote the solution space, the feasible set, a feasible solution and a real-valued objective function, respectively.

- If $S$ is a finite but large set, a combinatorial optimization problem is defined.

- If $S = R^n$, we refer to continuous optimization.

- A solution $x^* \in X$ is optimal if

$$f(x^*) \leq f(x), \ \forall x \in X.$$

- An exact algorithm for problem (1), if one exists, finds an optimal solution $x^*$, together with the proof of its optimality, or shows that there is no feasible solution, i.e., $X = \emptyset$, or the solution is unbounded.

- For continuous optimization, it is reasonable to allow for some degree of tolerance, i.e., to stop when sufficient convergence is detected.
Variable metric algorithm

Assume that the function $f(x)$ is approximated by its Taylor series

$$f(x) = \frac{1}{2}x^T Ax - b^T x$$

$$x_{i+1} - x_i = -H_{i+1}(\nabla f(x_{i+1}) - \nabla f(x_i)).$$

Function $\text{VarMetric}(x)$

let $x \in \mathbb{R}^n$ be an initial solution

$H \leftarrow I; g \leftarrow -\nabla f(x)$

for $i = 1$ to $n$ do

$$\alpha^* \leftarrow \arg \min_\alpha f(x + \alpha \cdot Hg)$$

$$x \leftarrow x + \alpha^* \cdot Hg$$

$$g \leftarrow -\nabla f(x)$$

$H \leftarrow H + U$

end
Local search

Function $\text{BestImprovement}(x)$

repeat

\[ x' \leftarrow x \]
\[ x \leftarrow \arg\min_{y \in N(x)} f(y) \]

until \((f(x) \geq f(x'))\);

Function $\text{FirstImprovement}(x)$

repeat

\[ x' \leftarrow x; \; i \leftarrow 0 \]

repeat

\[ i \leftarrow i + 1 \]
\[ x \leftarrow \arg\min\{f(x), f(x_i)\}, \; x_i \in N(x) \]

until \((f(x) < f(x_i) \text{ or } i = |N(x)|)\);

until \((f(x) \geq f(x'))\);
Variable neighborhood search

- Let $\mathcal{N}_k$, $(k = 1, \ldots, k_{max})$, a finite set of pre-selected neighborhood structures,

- $\mathcal{N}_k(x)$ the set of solutions in the $k^{th}$ neighborhood of $x$.

- Most local search heuristics use only one neighborhood structure, i.e., $k_{max} = 1$.

- An optimal solution $x_{opt}$ (or global minimum) is a feasible solution where a minimum is reached.

- We call $x' \in X$ a local minimum with respect to $\mathcal{N}_k$ (w.r.t. $\mathcal{N}_k$ for short), if there is no solution $x \in \mathcal{N}_k(x') \subseteq X$ such that $f(x) < f(x')$.

- Metaheuristics (based on local search procedures) try to continue the search by other means after finding the first local minimum. VNS is based on three simple facts:
  - A local minimum w.r.t. one neighborhood structure is not necessarily so for another;
  - A global minimum is a local minimum w.r.t. all possible neighborhood structures;
  - For many problems, local minima w.r.t. one or several $\mathcal{N}_k$ are relatively close to each other.
Variable neighborhood search

- In order to solve optimization problem by using several neighborhoods, facts 1 to 3 can be used in three different ways:
  - (i) deterministic;
  - (ii) stochastic;
  - (iii) both deterministic and stochastic.

- Some VNS variants
  - Variable neighborhood descent (VND) (sequential, nested)
  - Reduced VNS (RVNS)
  - Basic VNS (BVNS)
  - Skewed VNS (SVNS)
  - General VNS (GVNS)
  - VN Decomposition Search (VNDS)
  - Parallel VNS (PVNS)
  - Primal Dual VNS (P-D VNS)
  - Reactive VNS
  - Backward-Forward VNS
  - Best improvement VNS
  - Exterior point VNS
  - VN Simplex Search (VNSS)

Balcor 2011, Thessaloniki, Greece
 VN Branching
 VN Pump
 Continuous VNS
 Mixed Nonlinear VNS (RECIPE), etc.
Neighborhood change

Function \texttt{NeighbourhoodChange} \((x, x', k)\)

\textbf{if} \(f(x') < f(x)\) \textbf{then}

\[ x \leftarrow x'; \; k \leftarrow 1 \; /\!\!/ \text{ Make a move } \]

\textbf{else}

\[ k \leftarrow k + 1 \; /\!\!/ \text{ Next neighborhood } \]

\textbf{end}
Reduced VNS

Function \textbf{RVNS} \((x, k_{\text{max}}, t_{\text{max}})\)

\textbf{repeat}

\hspace{1cm} k \leftarrow 1

\textbf{repeat}

\hspace{2cm} x' \leftarrow \text{Shake}(x, k)

\hspace{2cm} \text{NeighborhoodChange}(x, x', k)

\hspace{2cm} \textbf{until} \; k = k_{\text{max}};

\hspace{1cm} t \leftarrow \text{CpuTime}()

\textbf{until} \; t > t_{\text{max}};

\begin{itemize}
  \item RVNS is useful in very large instances, for which local search is costly.
  \item It has been observed that the best value for the parameter \(k_{\text{max}}\) is often 2.
  \item The maximum number of iterations between two improvements is usually used as a stopping condition.
  \item RVNS is akin to a Monte-Carlo method, but is more systematic
\end{itemize}

\begin{itemize}
  \item Balcor 2011, Thessaloniki, Greece
\end{itemize}
When applied to the $p$-Median problem, RVNS gave solutions as good as the Fast Interchange heuristic of Whitaker while being 20 to 40 times faster.
Function VND \((x, k'_{\text{max}})\)

\[
\text{repeat}
\]

\[
k \leftarrow 1
\]

\[
\text{repeat}
\]

\[
x' \leftarrow \arg \min_{y \in \mathcal{N}_{k'}(x)} f(x) \quad /* \text{Find the best neighbor in } \mathcal{N}_k(x) */
\]

\[
\text{NeighbourhoodChange} \ (x, x', k) \quad /* \text{Change neighbourhood} */
\]

\[
\text{until} \ k = k'_{\text{max}};
\]

\[
\text{until} \ \text{no improvement is obtained};
\]
Sequential VND

**Function** Seq-VND($x, \ell_{max}$)

\[
\ell \leftarrow 1 \quad \text{// Neighborhood counter}
\]

repeat

\[
i \leftarrow 0 \quad \text{// Neighbor counter}
\]

repeat

\[
i \leftarrow i + 1
\]

\[
x' \leftarrow \arg \min \{f(x), f(x_i)\}, \quad x_i \in N_\ell(x) \quad \text{// Compare}
\]

until (\(f(x') < f(x)\) or \(i = |N_\ell(x)|\))

\[
\ell, x \leftarrow \text{NeighborhoodChange} \left(x, x', \ell\right); \quad \text{// Neighborhood change}
\]

until \(\ell = \ell_{max}\)

- The final solution of Seq-VND should be a local minimum with respect to all \(\ell_{max}\) neighborhoods.
- The chances to reach a global minimum are larger than with a single neighborhood structure.
- The total size of Seq-VND is equal to the union of all neighborhoods used.
If neighborhoods are disjoint (no common element in any two) then the following holds:

$$|\mathcal{N}_{\text{Seq-VND}}(x)| = \sum_{\ell=1}^{\ell_{\text{max}}} |\mathcal{N}_\ell(x)|, \ x \in X.$$
Nested VND

- Assume that we define two neighborhood structures ($\ell_{max} = 2$). In the nested VND we in fact perform local search with respect to the first neighborhood in any point of the second.
- The cardinality of neighborhood obtained with the nested VND is product of cardinalities of neighborhoods included, i.e.,

$$|\mathcal{N}_{\text{Nest-VND}}(x)| = \prod_{\ell=1}^{\ell_{max}} |\mathcal{N}_\ell(x)|, \ x \in X.$$

- The pure Nest-VND neighborhood is much larger than the sequential one.
- The number of local minima w.r.t. Nest-VND will be much smaller than the number of local minima w.r.t. Seq-VND.
Function Nest-VND \((x, x', k)\)

Make an order of all \(l_{\max} \geq 2\) neighborhoods that will be used in the search

Find an initial solution \(x\); let \(x_{opt} = x\), \(f_{opt} = f(x)\)

Set \(l = l_{max}\)

repeat

\[
\begin{align*}
\text{if} \quad & \text{all solutions from } l \text{ neighborhood are visited then } l = l + 1 \\
\text{if} \quad & \text{there is any non visited solution } x_l \in N_l(x) \text{ and } l \geq 2 \text{ then } x_{\text{cur}} = x_l, \quad l = l - 1 \\
\text{if} \quad & l = 1 \text{ then} \\
\quad & \text{Find objective function value } f = f(x_{\text{cur}}) \\
\quad & \text{if } f < f_{opt} \text{ then } x_{opt} = x_{\text{cur}}, \ f_{opt} = f_{cur} \\
\end{align*}
\]

until \(l = l_{max} + 1\) (i.e., until there is no more points in the last neighborhood)
Mixed nested VND

- After exploring $b$ (a parameter) neighborhoods, we switch from a nested to a sequential strategy. We can interrupt nesting at some level $b$ ($1 \leq b \leq \ell_{max}$) and continue with the list of the remaining neighborhoods in sequential manner.
- If $b = 1$, we get Seq-VND. If $b = \ell_{max}$ we get Nest-VND.
- Since nested VND intensifies the search in a deterministic way, the boost parameter $b$ may be seen as a balance between intensification and diversification in deterministic local search with several neighborhoods.
- Its cardinality is clearly

$$|\mathcal{N}_{Mix-VND}(x)| = \sum_{\ell=b}^{\ell_{max}} |\mathcal{N}_\ell(x)| + \prod_{\ell=1}^{b-1} |\mathcal{N}_\ell(x)|, \quad x \in X.$$
Basic VNS

The Basic VNS (BVNS) method \[?] combines deterministic and stochastic changes of neighbourhood. Its steps are given in Algorithm 8.

Function \( \text{VNS} (x, k_{\text{max}}, t_{\text{max}}) \)

repeat

\( k \leftarrow 1 \)

repeat

\( x' \leftarrow \text{Shake}(x, k) \)  /* Shaking */
\( x'' \leftarrow \text{FirstImprovement}(x') \)  /* Local search */
NeighbourhoodChange\( (x, x'', k) \)  /* Change neighbourhood */
\( \text{until} \ k = k_{\text{max}}; \)

\( t \leftarrow \text{CpuTime()} \)
\( \text{until} \ t > t_{\text{max}}; \)

• Balcor 2011, Thessaloniki, Greece
General VNS

Function $\text{GVNS} \ (x, k'_{\text{max}}, k_{\text{max}}, t_{\text{max}})$

repeat

\[ k \leftarrow 1 \]

repeat

\[ x' \leftarrow \text{Shake}(x, k) \]
\[ x'' \leftarrow \text{VND}(x', k'_{\text{max}}) \]
\[ \text{NeighborhoodChange}(x, x'', k) \]

until $k = k_{\text{max}}$;

\[ t \leftarrow \text{CpuTime()} \]
until $t > t_{\text{max}}$;

• Balcor 2011, Thessaloniki, Greece
Skewed VNS

Function NeighbourhoodChangeS(x, x'', k, α)

if \( f(x'') - \alpha \rho(x, x'') < f(x) \) then

\[ x \leftarrow x''; \ k \leftarrow 1 \]

else

\[ k \leftarrow k + 1 \]

end
Function SVNS (\(x, k_{\text{max}}, t_{\text{max}}, \alpha\))

```
repeat

\(k \leftarrow 1; \ x_{\text{best}} \leftarrow x\)

repeat

\(x' \leftarrow \text{Shake}(x, k)\)
\(x'' \leftarrow \text{FirstImprovement}(x')\)
\(\text{KeepBest}\ (x_{\text{best}}, x)\)
\(\text{NeighbourhoodChangeS}(x, x'', k, \alpha)\)

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

\(x \leftarrow x_{\text{best}}\)
\(t \leftarrow \text{CpuTime}\()\)
until \(t > t_{\text{max}};\)
```

Balcor 2011, Thessaloniki, Greece
Extensions

Function $\text{BI-VNS}(x, k_{\max}, t_{\max})$

repeat

$k \leftarrow 1 \quad x_{\text{best}} \leftarrow x$

repeat

$x' \leftarrow \text{Shake}(x, k)$
$x'' \leftarrow \text{FirstImprovement}(x')$
$\text{KeepBest}(x_{\text{best}}, x'')$

$k \leftarrow k + 1$

until $k = k_{\max}$;

$x \leftarrow x_{\text{best}}$
$t \leftarrow \text{CpuTime}()$
until $t > t_{\max}$;
Extensions

Function \text{FH-VNS} (x, k_{max}, t_{max})

\text{repeat}

\hspace{1em} k \leftarrow 1

\text{repeat}

\hspace{2em} \text{for } \ell = 1 \text{ to } k \text{ do}

\hspace{3em} x' \leftarrow \text{Shake}(x, k)

\hspace{3em} x'' \leftarrow \text{FirstImprovement}(x')

\hspace{3em} \text{KeepBest}(x, x'')

\hspace{2em} \text{end}

\hspace{1em} \text{NeighbourhoodChange}(x, x'', k)

\hspace{1em} \text{until } k = k_{max};

\hspace{1em} t \leftarrow \text{CpuTime()}

\hspace{1em} \text{until } t > t_{max};