CIVIL-557
Decision-Aid Methodologies in Transportation
Variable Neighborhood Search (VNS)

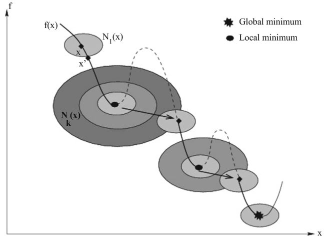
Fabian Torres

Transport and Mobility Laboratory TRANSP-OR
École Polytechnique Fédérale de Lausanne EPFL

EPFL

Variable Neighborhood Search (VNS)

- VNS is a **metaheuristic** for solving combinatorial and **global** optimization problems.
- Its basic idea is a systematic **change of neighborhood** both within:
  - **a descent phase**: to find a local optimum.
  - **a perturbation phase**: to get out of the corresponding valley.



Variable Neighborhood Search

Consider an optimization problem formulated as:

$$\min\{f(x)|x \in X, \quad X \subseteq \Omega\}$$

where Ω is the solution space and X is the feasible set.

- A solution $x^* \in X$ is a **global minimum** if $f(x^*) \leq f(x), \quad \forall x \in X$.
- $N_k(k = 1, ..., k_{max})$ is a finite set of pre-selected **neighborhood structure**.
- $N_k(x)$ is the set of solutions in the k^{th} neighborhood of x.
- A solution $x' \in X$ is **local minimum** with respect to N_k if there is no solution $x \in N_k(x') \subseteq X$ such that $f(x) < f(x')$.

EPFL

VNS Principles

VNS is based on three simple facts:

- **Fact 1**: A local minimum with respect to **one** neighborhood structure is not necessarily a local minimum for **another** neighborhood structure.
- **Fact 2**: A **global** minimum is a local minimum with respect to **all** possible neighborhood structures.
- **Fact 3**: For many problems, local minima with respect to one or several N_k are relatively close to each other.

Neighborhood change

- Consider x the current iterate and consider x' a solution obtained when performing a local search with respect to the k^{th} neighborhood.
- Compare $f(x)$ and $f(x')$.
- If an improvement is obtained, the current iterate is updated and k is returned to its initial value.
- Otherwise, the next neighborhood is considered.

Algorithm 1 Neighborhood change

```
Function NeighborhoodChange (x, x', k)
  if f(x') < f(x) then
      x ← x' // Make a move
      k ← 1 // Initial neighborhood
  else
      k ← k + 1 // Next neighborhood
return x, k
```

EPFL

Variable neighborhood descent (VND)

The variable neighborhood descent method is obtained if a change of neighborhoods is performed in a **deterministic** way.

Algorithm 2 Variable neighborhood descent

```
Function VND (x, k_max)
  k ← 1
  repeat
      x' ← arg min_{y∈N_k(x)} f(y) // Find the best neighbor in N_k(x)
      x, k ← NeighborhoodChange (x, x', k) // Change neighborhood
  until k = k_max
return x
```

Basic Variable Neighborhood Search (BVNS)

The basic VNS method combines **deterministic and stochastic** changes of neighborhood:

- The stochastic part is represented by a **shake function** that generates a point x' a **random** from the k^{th} neighborhood of x ($x' \in N_k(x)$).
- The deterministic part is a **best improvement** function which is a local search algorithm.

We also assume that a stopping condition has been chosen like the maximum CPU time allowed t_{max}.

Therefore, BVNS uses two parameters: t_{max} and k_{max}.

Algorithm 7 Basic VNS

Function BVNS(x, k_{max}, t_{max})

```
t ← 0
while t < t_max do
    k ← 1
    repeat
        x' ← Shake(x,k)                      // Shaking
        x'' ← BestImprovement(x')            // Local search
        x,k ← NeighborhoodChange(x,x'',k)    // Change neighborhood
    until k = k_max
    t ← CpuTime()
return x
```

General VNS

General VNS is a BVNS where the local search step is replaced by VND algorithm.

Algorithm 8 General VNS

<u>**Function**</u> GVNS $(x, \ell_{max}, k_{max}, t_{max})$

repeat
  $k \leftarrow 1$
  repeat
    $x' \leftarrow \texttt{Shake}(x, k)$
    $x'' \leftarrow \texttt{VND}(x', \ell_{max})$
    $x, k \leftarrow \texttt{NeighborhoodChange}(x, x'', k)$
  until $k = k_{max}$
  $t \leftarrow \texttt{CpuTime}()$
until $t > t_{max}$
return x

This general VNS (VNS/VND) approach has led to some of the most successful applications reported in the literature.

Case Study: The Multi-Depot Vehicle Routing Problem

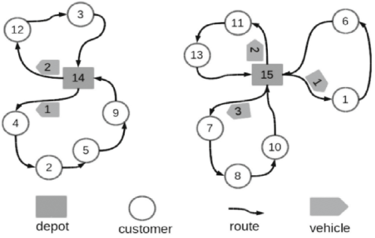
MDVRP is a CVRP in which there is more than one depot.

- Let $G = (V, A)$ be a complete graph, where V is a set of nodes and A is a set of arcs.
- The set of nodes are partitioned into two subsets:
  - the set of customers to be served, given by $V_{CST} = \{1, 2, ..., N\}$,
  - the set of depots $V_{DEP} = \{N + 1, N + 2, ..., N + M\}$.
- There is a non-negative **cost** c_{ij} associated with each $arc(i, j) \in A$.
- For each customer, there is a non-negative **demand** d_i and there is no demand at the depot nodes.
- In each depot, there are K identical vehicles, each with **capacity** Q.
- The **service time** at each customer i is t_i.
- The maximum route duration time is set to T.
- A conversion factor w_{ij} to transform the cost c_{ij} into time units.

Example of MDVRP solution

Let's consider an instance of MDVRP where $V_{CST} = \{1, 2, ..., 13\}$ and $V_{DEP} = \{N+1, N+2, ..., N+M\}$.

- A solution s of the MDRVP is represented by a list vector.
- Each position of this vector indicates a depot and each list indicates the visit routes to be performed by vehicles from that depot.
- The solution below is $s = \{r_1, r_2\}$ where $r_1 = [\mathbf{14}\ 4\ 2\ 5\ 9\ \mathbf{14}\ 12\ 3\ \mathbf{14}]$ and $r_2 = [\mathbf{15}\ 1\ 6\ \mathbf{15}\ 11\ 13\ \mathbf{15}\ 7\ 8\ 10\ \mathbf{15}]$.



Mathematical formulation MDVRP

Decision variables

- x_{ijk}: a binary decision variable which is equal to 1 when vehicle k visits node j immediately after node i, and 0 otherwise.
- y_i: auxiliary continuous variables are used in the subtour elimination constraints.

Objective function

$$\min \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} \sum_{k=1}^{K} c_{ij} x_{ijk}$$

EPFL

$$\sum_{i=1}^{N+M} \sum_{k=1}^{K} x_{ijk} = 1 \quad (j = 1, \ldots, N) \tag{2}$$

$$\sum_{j=1}^{N+M} \sum_{k=1}^{K} x_{ijk} = 1 \quad (i = 1, \ldots, N) \tag{3}$$

$$\sum_{i=1}^{N+M} x_{ihk} - \sum_{j=1}^{N+M} x_{hjk} = 0 \quad (k = 1, \ldots, K;\ h = 1, \ldots, N+M) \tag{4}$$

$$\sum_{i=1}^{N+M} \sum_{j=1}^{N+M} d_i x_{ijk} \leq Q \quad (k = 1, \ldots, K) \tag{5}$$

$$\sum_{i=1}^{N+M} \sum_{j=1}^{N+M} (c_{ij} w_{ij} + t_i) x_{ijk} \leq T \quad (k = 1, \ldots, K) \tag{6}$$

$$\sum_{i=N+1}^{N+M} \sum_{j=1}^{N} x_{ijk} \leq 1 \quad (k = 1, \ldots, K) \tag{7}$$

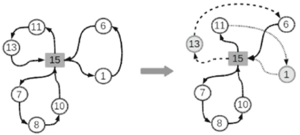
$$\sum_{j=N+1}^{N+M} \sum_{i=1}^{N} x_{ijk} \leq 1 \quad (k = 1, \ldots, K) \tag{8}$$

$$\begin{gathered} y_i - y_j + (N+M) x_{ijk} \leq N + M - 1 \\ \text{for } 1 \leq i \neq j \leq N \text{ and } 1 \leq k \leq K \end{gathered} \tag{9}$$

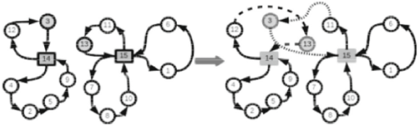
$$x_{ijk} \in \{0, 1\} \ \forall\, i, j, k \tag{10}$$

Neighborhoods for MDVRP

- *Swap*$(1,1)$: permutation between a customer v_j from a route r_k and a customer v_t from a route r_l.

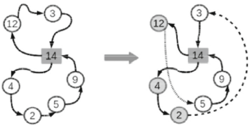


(a) Swap(1,1) in one depot.

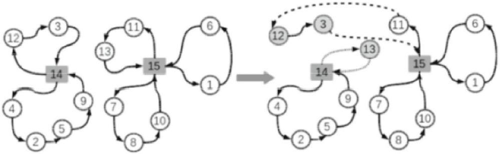


(b) Swap(1,1) in two different depots.

- *Swap*$(2,1)$: permutation of two adjacent customers v_j and v_{j+1} from a route r_k by a customer v_t from a route r_l.



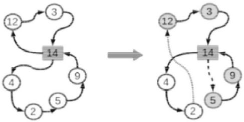
(a) Swap(2,1) in one depot.



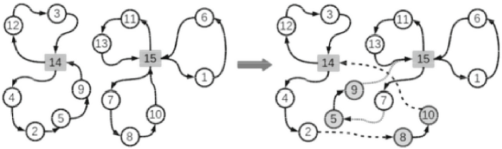
(b) Swap(2,1) in two different depots.

Neighborhoods for MDVRP

- *Swap*$(2,2)$: permutation between two adjacent customers v_j and v_{j+1} from a route r_k by another two adjacent customers v_t and v_{t+1}, $\forall v_t, v_{t+1} \in V_{CST}$, belonging to a route r_l.



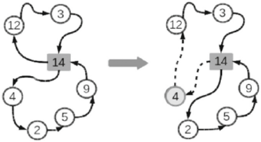
(a) Swap(2,2) in one depot.



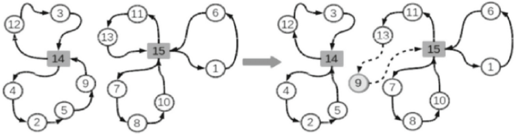
(b) Swap(2,2) in two different depots.

Neighborhoods for MDVRP

- *Shift*$(1,0)$: transference of a customer v_j from a route r_k to a route r_l.

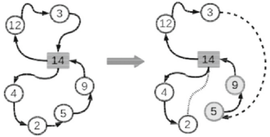


(a) Shift(1,0) in one depot.

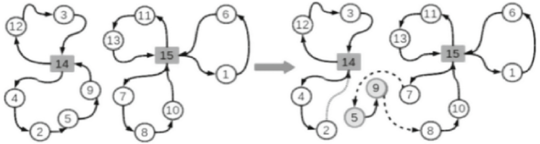


(b) Shift(1,0) in two different depots.

- *Shift*$(2,0)$: transference of two adjacent customersv_j and v_{j+1} from a route r_k to a route r_l.



(a) Shift(2,0) in one depot.



(b) Shift(2,0) in two different depots.

General VNS applied to MDVRP

- GVNS is a VNS algorithm in which the local search is made by the Variable Neighborhood Descent algorithm.
- However, for this problem the Randomized Variable Neighborhood Descent (RVND) is used as local search of GVNS.

Algorithm 1 : GVNS

Let s an initial solution;
$k \leftarrow 1$; ▷ Initial $N_k(s)$.
$s' \leftarrow s$;
while $iter < IterMax$ or $t < maxTime$ **do** ▷ Stopping criterion
    $s' \leftarrow Perturbation(s, k, level)$; ▷ Generate s' with neighborhood N_k
    $s'' \leftarrow RVND(s')$; ▷ Best Improvement Strategy
    if $f(s'') < f(s)$ **then**
        $s \leftarrow s''$;
        $k \leftarrow 1$;
        $iter \leftarrow 0$;
    else
        $iter \leftarrow iter + 1$;
        $k \leftarrow k + 1$; ▷ Change of neighborhood
    end if
end while
return s;

Perturbation algorithm

- In order to generate a **shake move** with these neighborhoods, each move is applied p times.
- The value of p is a random integer between 1 and level.

Algorithm 3 : Perturbation(s, k, $level$)

```
p ← rand(level);                         ▷ Generate a number between 1 and level
for (i = 1; i ≤ p; i++) do
    switch k do
        case 1: Shift(1,0)(s);
        case 2: Swap(2,2)(s);
        case 3: Swap(2,1)(s);
end for
return s;
```

RVND algorithm

- Unlike the VND method, which uses a **deterministic** neighborhood ordering, RVND applies a **random** neighborhood ordering scheme.
- RVND avoids **parameter tuning**, but may also avoid looking for the **best order**, which may be highly dependent on the instance.

Algorithm 2 : RVND(s)

```
Let L_r the list of r neighborhoods for local searches;
L_r ← randomize (L_r);                ▷ Put the list of neighborhoods in a random order
k ← 1;                                ▷ Initial N_k(s).
while k ≤ r do
    p ← L_r(k);
    Find the best neighbor s' ∈ N^(p)(s);
    if f(s') < f(s) then
        s ← s';
        k ← 1;
    else
        k ← k + 1;                    ▷ Neighborhood exchange
    end if
end while
return s;
```

Conclusion

- The computational experiments showed that the GVNS algorithm is a good alternative to solve the MDVRP, since its results are competitive and, unlike other algorithms, it has few parameters.
- Originally designed for approximate solution of combinatorial optimization problems, it was extended to address **mixed integer programs**, **nonlinear programs**, and recently **mixed integer nonlinear programs**.
- Applications are rapidly increasing in number and pertain to many fields:
  - location theory,
  - cluster analysis,
  - scheduling,
  - network design...

Main references

- Bezerra, S.N., de Souza, S.R., Souza, M.J.F.: *A VNS-Based Algorithm with Adaptive Local Search for Solving the Multi-Depot Vehicle Routing Problem*, Lecture Notes in Computer Science, vol 11328. Springer,2019.
- Bezerra, S.N., de Souza, S.R., Souza, M.J.F.: *A GVNS algorithm for solving the multi-depot vehicle routing problem*. Electron. Notes Discrete Math, 2018.