Meta-heuristic optimization

Nenad Mladenović,

Mathematical Institute, Serbian Academy of Sciences and Arts, Belgrade, Serbia

Department of Mathematics, Brunel University London UK.

- 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,...)
- Metaheurestics (Simulated annealing, Tabu search, GRASP, Variable neighborhood search, Genetic search, Evolutionary methods, Particle swarm optimization, ...)

Optimization problems

A deterministic optimization problem may be formulated as

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

- where $\mathcal{S}, X, x$ and f denote the solution space, the feasible set, a feasible solution and a real-valued objective function, respectively.
- If $\mathcal{S}$ is a finite but large set, a combinatorial optimization problem is defined.
- If $\mathcal{S} = R^n$, 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.

Metaheuristics

- Local search type
  - Simulated Annealing (Kirpatrick et al (1983);
  - Tabu search (Glover 1990)
  - GRASP (Greedy randomized adaptive search procedure) (Feo, Resende 1992)
  - Variable neighborhood search (Mladenovic 1995)
  - Other (Guided search, Noisy search, Large neighborhood search, Very large neighborhood search, Path relinking, Scatter search, Iterated local search)
- Inspired by nature
  - Genetic Algorithm (Memetic)
  - Ant colony optimization
  - Particle swarm optimization
  - Bee colony optimization, etc.
- Matheuristics
- Hybrids

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 `VarMetric(x)`
let $x \in R^n$ 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 `BestImprovement(x)`
repeat

$x' \leftarrow x$
$x \leftarrow arg\min_{y \in N(x)} f(y)$
until $(f(x) \geq f(x'))$;

Function `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` $(f(x) < f(x_i)$ `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)

- VN Branching
- VN Pump
- Continuous VNS
- Mixed Nonlinear VNS (RECIPE), etc.

Neighborhood change

Function `NeighbourhoodChange` (x, x', k)
if $f(x') < f(x)$ **then**

$x \leftarrow x'$; $k \leftarrow 1$ /* Make a move */
else

$k \leftarrow k + 1$ /* Next neighborhood */
end

Reduced VNS

Function `RVNS` (x, k_{max}, t_{max})
repeat

$k \leftarrow 1$
repeat

$x' \leftarrow$ `Shake`(x, k)
`NeighborhoodChange` (x, x', k)
until $k = k_{max}$;

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

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

- 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.

VND

Function `VND` (x, k'_{max})
repeat

$k \leftarrow 1$
repeat

$x' \leftarrow arg\min_{y \in \mathcal{N}'_k(x)} f(x)$ /* Find the best neighbor in $\mathcal{N}_k(x)$ */
`NeighbourhoodChange` (x, x', k) /* Change neighbourhood */
until $k = k'_{max}$;
until no improvement is obtained;

Sequential VND

Function Seq-VND(x, ℓ_{max})
$\ell \leftarrow 1$ // Neighborhood counter
repeat
  $i \leftarrow 0$ // Neighbor counter
  repeat
    $i \leftarrow i + 1$
    $x' \leftarrow arg\min\{f(x), f(x_i)\}, x_i \in N_\ell(x)$ // Compare
  until $(f(x') < f(x)$ or $i = |N_\ell(x)|)$
  $\ell, x \leftarrow$ NeighborhoodChange (x, x', ℓ); // Neighborhood change
until $\ell = \ell_{max}$

- The final solution of Seq-VND should be a local minimum with respect to all ℓ_m neighborhoods.
- The chances to reach a global minimum are larger than with a single neighborhood structu
- 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

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

Nested VND

- Assume that we define two neighborhood structures ($\ell_{max} = 2$). In the nested VND we 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 neighborhoods included, i.e.,

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

- 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 lo minima w.r.t. `Seq-VND`.

Nested VND

Function `Nest-VND` (x, x', k)
Make an order of all $\ell_{max} \geq 2$ neighborhoods that will be used in the search
Find an initial solution x; let $x_{opt} = x$, $f_{opt} = f(x)$
Set $\ell = \ell_{max}$
repeat

> **if** all solutions from ℓ neighborhood are visited **then** $\ell = \ell + 1$
> **if** there is any non visited solution $x_\ell \in N_\ell(x)$ and $\ell \geq 2$ **then** $x_{cur} = x_\ell$, $\ell = \ell - 1$
> **if** $\ell = 1$ **then**
>
> > Find objective function value $f = f(x_{cur})$
> > **if** $f < f_{opt}$ **then** $x_{opt} = x_{cur}$, $f_{opt} = f_{cur}$

until $\ell = \ell_{max} + 1$ (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 sequent strategy. We can interrupt nesting at some level b ($1 \leq b \leq \ell_{max}$) and continue with t 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, boost parameter b may seen as a balance between intensification and diversification in deterministic local search wi several neighborhoods.
- Its cardinality is clearly

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

Basic VNS

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

Function `VNS` (x, k_{max}, t_{max})
repeat

$k \leftarrow 1$
repeat

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

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

General VNS

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

repeat

  $k \leftarrow 1$

  repeat

    $x' \leftarrow \texttt{Shake}(x, k)$

    $x'' \leftarrow \texttt{VND}(x', k'_{max})$

    $\texttt{NeighborhoodChange}(x, x'', k)$

  until $k = k_{max}$;

  $t \leftarrow \texttt{CpuTime}()$

until $t > t_{max}$;

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 $(x, k_{max}, t_{max}, \alpha)$

repeat

$k \leftarrow 1$; $x_{best} \leftarrow x$

repeat

$x' \leftarrow \texttt{Shake}(x, k)$

$x'' \leftarrow \texttt{FirstImprovement}(x')$

$\texttt{KeepBest}\ (x_{best}, x)$

$\texttt{NeighbourhoodChangeS}(x, x'', k, \alpha)$

until $k = k_{max}$;

$x \leftarrow x_{best}$

$t \leftarrow \texttt{CpuTime}()$

until $t > t_{max}$;

Extensions

Function `BI-VNS` (x, k_{max}, t_{max})
repeat

$k \leftarrow 1$ $x_{best} \leftarrow x$
repeat

$x' \leftarrow$ `Shake`(x, k)
$x'' \leftarrow$ `FirstImprovement`(x')
`KeepBest`(x_{best}, x'')
$k \leftarrow k + 1$
until $k = k_{max}$;

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

Extensions

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

$k \leftarrow 1$
repeat

for $\ell = 1$ to k **do**

$x' \leftarrow$ Shake(x, k)
$x'' \leftarrow$ FirstImprovement(x')
KeepBest(x, x'')
end

NeighbourhoodChange(x, x'', k)
until $k = k_{max}$;

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

Test Instance	$\|V\|$	$\|T\|$	$\|W\|$	Objective function value B&C	TS	GVNS	(%) dev GVNS vs. TS	B&C	Size B&C	GVNS	Time (sec) B&C	GVNS
kroA100	25	1	75	2356.96	2321.32	2356.92	-1.53	0.00	9	9	1622.75	83.09
kroA100	25	6	75	2588.61	2576.96	2588.57	-0.45	0.00	10	10	137.82	82.80
kroA100	25	12	75	2725.51	2723.86	2725.48	-0.06	0.00	16	16	42.80	8.72
kroA100	25	18	75	2879.15	2877.97	2879.10	-0.04	0.00	20	20	135.96	1.73
kroA150	37	1	113	3516.82	3490.45	3516.94	-0.76	0.00	8	8	3867.79	241.33
kroA150	37	9	113	3882.45	3853.74	3882.40	-0.74	0.00	14	14	75.73	34.36
kroA150	37	18	113	4166.33	4166.33	4166.28	0.00	0.00	20	20	8.05	6.60
kroA150	37	27	113	4268.36	4268.37	4268.31	0.00	0.00	30	30	16.58	3.67
kroA200	50	1	150	3775.93	3636.83	3781.07	-3.97	-0.14	9	10	7200.73	68.17
kroA200	50	12	150	3938.36	3910.39	3938.29	-0.71	0.00	17	17	868.07	20.44
kroA200	50	25	150	4545.32	4545.33	4545.28	0.00	0.00	28	28	321.16	5.71
kroA200	50	37	150	4914.69	4849.82	4914.63	-1.34	0.00	38	38	1.67	0.90
kroB100	25	1	75	2444.09	2442.18	2444.05	-0.08	0.00	8	8	48.29	46.97
kroB100	25	6	75	2392.91	2392.91	2392.88	0.00	0.00	11	11	8.70	42.56
kroB100	25	12	75	2507.47	2507.46	2507.43	0.00	0.00	15	15	2.32	37.22
kroB100	25	18	75	2599.72	2599.71	2599.68	0.00	0.00	20	20	18.85	34.42
kroB150	37	1	113	3005.74	2948.05	3013.83	-2.23	-0.27	8	9	7235.95	25.87
kroB150	37	9	113	3282.67	3272.50	3282.61	-0.31	0.00	15	15	41.91	19.46
kroB150	37	18	113	3525.58	3525.57	3525.53	0.00	0.00	22	22	18.24	10.97
kroB150	37	27	113	3700.56	3700.55	3700.50	0.00	0.00	29	29	3.45	2.60
kroB200	50	1	150	3561.77	3515.09	3561.76	-1.33	0.00	10	10	3877.27	40.86
kroB200	50	12	150	3537.64	3528.10	3537.58	-0.27	0.00	20	20	252.72	68.24
kroB200	50	25	150	4209.81	4209.40	4209.76	-0.01	0.00	30	30	163.37	17.72
kroB200	50	37	150	4597.73	4597.72	4597.69	0.00	0.00	38	38	10.33	3.40
kroC100	25	1	75	2114.74	2039.45	2114.70	-3.69	0.00	8	8	166.30	45.98
kroC100	25	6	75	2287.62	2234.55	2287.60	-2.37	0.00	10	10	53.78	25.73
kroC100	25	12	75	2303.61	2303.60	2303.57	0.00	0.00	15	15	7.81	6.78
kroC100	25	18	75	2524.54	2491.42	2524.49	-1.33	0.00	18	18	0.26	2.60
kroD100	25	1	75	2335.90	2306.69	2335.86	-1.26	0.00	8	8	158.82	76.47
kroD100	25	6	75	2435.53	2422.13	2435.51	-0.55	0.00	12	12	40.72	123.76
kroD100	25	12	75	2522.18	2494.43	2522.16	-1.11	0.00	15	15	0.47	8.88
kroD100	25	18	75	2642.83	2642.82	2642.78	0.00	0.00	19	19	1.13	3.86
kroE100	25	1	75	2618.70	2526.25	2618.66	-3.66	0.00	7	7	10.73	26.55
kroE100	25	6	75	2561.25	2492.11	2561.22	-2.77	0.00	10	10	1.74	11.49
kroE100	25	12	75	2659.51	2651.28	2659.50	-0.31	0.00	16	16	1.08	9.84
kroE100	25	18	75	2766.33	2766.34	2766.30	0.00	0.00	22	22	10.00	10.22
Average				3130.47	3106.44	3130.80	-0.86	-0.01	16.81	16.86	734.26	35.00

Table 1: Computational results on extended AtTSP instances

Thank you for your attention!

nenad.mladenovic@brunel.ac.uk