New Variable neighborhood search methods and applications

Nenad Mladenović,

Mathematical institute, Serbian Academy of Sciences and Arts Belgrade Serbia

School of Mathematics, Brunel University London UK

- Introduction
- Variable neighborhood search algorithms
- Recent advances in VNS
  - Global Continuous VNS
  - New neighborhoods for discrete problems
  - 3-level VNS
  - Matheuristics and VNS
  - VNS Hybrids
- Conclusions

Introduction

- Mladenovic (1995) - Variable neighborhood algorithm - a new metaheuristic for combinatorial optimization.
- Mladenovic, Hansen (1997) Variable neighborhood search.
- This paper cited around 1,620 times (Google scholar).
- This paper cited > 600 times (Web of Knowledge)
- Euro Mini Conferences devoted to VNS 2005 and 2012;
- Next VNS conference will take part in October 2014 in Tunisia;
- 7 VNS special issues (JOH, EJOR, COR, JOGO, YUJOR and IMA-MAN 2 times).
- Several book Chapters.

Preliminaries - 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

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 and Reduced VNS

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
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;

Basic VNS

The Basic VNS (BVNS) method [**?**] combines deterministic and stochastic changes of neighbourhood. Its steps are given in Algorithm 6.

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

    $x' \leftarrow$ Shake(x, k)

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

    NeighborhoodChange(x, x'', k)

  until $k = k_{max}$;

  $t \leftarrow$ CpuTime()

until $t > t_{max}$;

Attraction probabilities

- Given an incumbent solution x with objective function value f_{r+1},
- let σ_{rk} = the probability that a better solution will be obtained from random starting poi $y \in \mathcal{N}_k(x)$; i.e.,
- σ_{rk} is the attraction probability associated with $\mathcal{N}_k(x)$.
- Then the probability of obtaining a better solution in the next iteration of VNS is given by

$$\sigma_r = \sigma_{r1} + \sum_{k=2}^{k_{max}} \sigma_{rk} \prod_{\ell=1}^{k-1} (1 - \sigma_{r\ell}).$$

- The main question (which no one appears to have asked) relates to the behavior of t attraction probabilities in the different neighborhoods $\mathcal{N}_k(x)$; i.e., how do these attracti probabilities change with distance $\rho(x, y)$.
- Based on empirical evidence it would appear in general that $\sigma_r \gg \theta_r$, the correspondi probability of improvement in the next iteration of MLS. This implies that attracti probabilities tend to be much larger in the closer (smaller) neighborhoods of x.

Variable neighborhood search-Example of attraction probabilities

- Suppose that S contains 1000 points, of which a total of 10 points are in attraction basi that lead to an improved solution.
- Suppose further that there are 3 neighborhoods used, $\mathcal{N}_1(x), \mathcal{N}_2(x), \mathcal{N}_3(x)$, containing 90 and 900 points, respectively; of the 10 attraction points, one is found in $\mathcal{N}_1(x)$, two $\mathcal{N}_2(x)$, and the remaining seven in $\mathcal{N}_3(x)$.
- Then,
$$\sigma_r = \frac{1}{9} + \frac{2}{90}(1 - \frac{1}{9}) + \frac{7}{900}(1 - \frac{1}{9})(1 - \frac{2}{90}) = 0.1377.$$
- The expected number of equivalent MLS iterations (or local searches) for one iteration VNS is obtained by the formula,
$$E[\#LS] = 1 \times \sigma_{r1} + \sum_{k=2}^{k_{max}-1} k\sigma_{rk} \prod_{\ell=1}^{k-1}(1 - \sigma_{r\ell}) + k_{max} \prod_{\ell=1}^{k_{max}-1}(1 - \sigma_{r\ell}),$$
for $k_{max} \geq 3$.
- For our example, this gives
$$E[\#LS] = 1 \times \frac{1}{9} + 2 \times \frac{2}{90}(1 - \frac{1}{9}) + 3 \times (1 - \frac{1}{9})(1 - \frac{2}{90}) = 2.758.$$
- Comparing $\sigma_r = 0.1377$ with $\theta_r \times E[\#LS] = 0.01 \times 2.758 = 0.02758$, we obser a five-fold increase in the probability of obtaining an improved solution resulting from t distribution of attraction points within the "closer" neighborhoods of the incumbent solutio

j^{th} best solut.	$p = 5$		$p = 10$		$p = 15$	
	MLS (θ_{j-1})	VNS (σ_{j-1})	MLS (θ_{j-1})	VNS (σ_{j-1})	MLS (θ_{j-1})	VNS (σ_{j-1})
2	0.0690	0.3364	0.0034	0.2636	0.0001	0.3192
3	0.0875	0.6664	0.0074	0.6112	0.0002	0.3851
4	0.1002	0.7248	0.0092	0.6972	0.0003	0.2291
5	0.1019	0.4632	0.0104	0.0578	0.0005	0.2513
6	0.1504	0.8981	0.0139	0.6776	0.0006	0.2995
7	0.1805	0.5267	0.0201	0.4582	0.0007	0.3310
8	0.1879	0.6060	0.0210	0.5508	0.0008	0.3992
9	0.1903	0.7928	0.0239	0.8784	0.0009	0.4893
10	0.1916	0.9403	0.0241	0.9674	0.0010	0.5226
11	0.1991	0.6584	0.0242	0.3177	0.0012	0.4891
12	0.2068	0.2362	0.0269	0.6912	0.0013	0.5752
13	0.2106	0.7669	0.0273	0.6547	0.0014	0.1420
14	0.2129	0.5777	0.0299	0.5754	0.0016	0.2788
15	0.2189	0.9915	0.0314	0.6441	0.0017	0.6283
16	0.2713	0.9228	0.0340	0.3922	0.0019	0.2604
17	0.2861	0.1875	0.0357	0.4832	0.0021	0.5176
18	0.4168	0.8619	0.0388	0.6268	0.0022	0.4238
19	0.4264	0.8870	0.0398	0.8595	0.0023	0.3297
20	0.4280	0.8899	0.0405	0.8056	0.0025	0.4131

Table 1: Attraction probabilities of MLS and VNS.

New neighborhood types within 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_{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

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

Nested VND

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

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

- The pure `Nest-VND` neighborhood is much larger than the sequential.
- 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`.

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 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, boost parameter b may be seen as a balance between intensification and diversification in deterministic local search with several neighborhoods.
- Its cardinality is clearly

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

n	Q	Local Search	Min. % dev	Max. % dev	Avg. % dev	Avg. time
200	10	forward-insertion	95.650	255.517	181.195	0.088
		backward-insertion	94.753	252.827	186.318	0.081
		2–opt	13.882	242.910	32.433	0.138
		Seq–VND	12.275	242.910	27.808	0.163
		Seq–VND–3	8.991	242.910	24.309	0.478
		Mix–VND	1.269	242.910	12.958	2.989
400	10	forward-insertion	78.320	218.770	165.603	0.385
		backward-insertion	81.881	218.770	169.416	0.317
		2–opt	12.078	217.104	21.852	0.831
		Seq–VND	10.684	217.104	18.954	0.769
		Seq–VND–3	8.951	203.738	16.035	4.062
		Mix–VND	1.492	217.104	6.573	26.569

Table 2: Comparison of different local search algorithms on two instances

Continuous VNS

Algorithm Glob-VNS

/* Initialization */

01 Select the pairs $(\mathcal{G}_l, \mathcal{P}_l),\ l = 1, \ldots, m$ of geometry structures and distribution types and a set of radii $\rho_i,\ i = 1, \ldots, k_{\max}$

02 Choose an arbitrary initial point $x \in S$

03 Set $x^* \leftarrow x$, $f^* \leftarrow f(x)$

/* Main loop */

04 repeat the following steps until the stopping condition is met

05 Set $l \leftarrow 1$

06 repeat the following steps until $l > m$

07 Form the neighborhoods $\mathcal{N}_k,\ k = 1, \ldots, k_{\max}$ using geometry structure $\mathcal{G}_l$ and radii ρ_k

08 Set $k \leftarrow 1$

09 repeat the following steps until $k > k_{\max}$

10 Shake: Generate at random a point $y \in \mathcal{N}_k(x^*)$ using random distribution $\mathcal{P}_l$

11 Apply some local search method from y to obtain a local minimum y'

12 if $f(y') < f^*$ then

13 Set $x^* \leftarrow y'$, $f^* \leftarrow f(y')$ and goto line 05

14 endif

15 Set $k \leftarrow k + 1$

16 end

17 Set $l \leftarrow l + 1$

18 end

19 end

20 Stop. Point x^* is an approximate solution of the problem

	Algorithm Gauss-VNS
	/* Initialization */
01	Select the set of covariance matrices Σ_k, $k = 1, \ldots, k_{\max}$
02	Choose an arbitrary initial point $x \in S$
03	Set $x^* \leftarrow x$, $f^* \leftarrow f(x)$
	/* Main loop */
04	repeat the following steps until the stopping condition is met
05	Set $k \leftarrow 1$
06	repeat the following steps until $k > k_{\max}$
07	Shake: Generate y from a Gaussian distribution with mean x^* and covariance matrix Σ_k
08	Apply some local search method from y to obtain a local minimum y'
09	if $f(y') < f^*$ then
10	Set $x^* \leftarrow y'$, $f^* \leftarrow f(y')$ and goto line 05
11	endif
12	Set $k \leftarrow k + 1$
13	end
14	end
15	Stop. Point x^* is an approximate solution of the problem.

- The idea is to replace the class $\{\mathcal{N}_k(x)\}_{1 \le k \le k_{\max}}$ of neighborhoods of point x by a cla of probability distributions $\{\mathcal{P}_k(x)\}_{1 \le k \le k_{\max}}$.

- The next random point in the shaking step is generated using the probability distributi $\mathcal{P}_k(x)$.
- Comparison between `GLOB` and `Gauss VNS` are given in Table 1 on Ackley and Rastrigin te functions with different value of n.

$$f(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$$

$$-15 \leq x_i \leq 30, \quad i = 1, \ldots, n, \quad f_{\min} = 0,$$

$$f(x) = 10n + \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right)$$

$$-5.12 \leq x_i \leq 5.12, \quad i = 1, \ldots, n, \quad f_{\min} = 0$$

- For both heuristics, the steepest descent (SD) local search is used.
- All test instances are solved, but with different computational efforts (the number of functi evaluations before known global optimum is reached).

function	n	local minim.	$k_{\max}$	Computer effort (CE)		% deviation
				Glob-VNS	Gauss-VNS	
AC10	10	SD	10	188670	50149	276.22%
AC20	20	SD	10	433194	158412	173.46%
AC30	30	SD	10	909918	304825	198.51%
AC40	40	SD	10	1577138	528718	198.30%
AC50	50	SD	10	4791075	1143721	318.90%
AC60	60	SD	10	7820247	2315178	237.78%
AC70	70	SD	10	36641634	4255533	761.04%
AC80	80	SD	10	212944367	17180658	1139.44%
					Average	412.96%
RA10	10	SD	15	52471	85589	-38.69%
RA20	20	SD	15	213597	287075	-25.60%
RA30	30	SD	15	366950	599635	-38.80%
RA40	40	SD	15	697160	1115923	-37.53%
RA50	50	SD	15	1334842	1504701	-11.29%
RA100	100	SD	15	5388075	6248753	-13.77%
RA150	150	SD	15	11007093	13678014	-19.53%
RA200	200	SD	15	24026456	31639001	-24.06%
					Average	-26.16%

Table 3: Comparison of Glob-VNS and Gauss-VNS on Ackley and Rastrigin functions

Restarted Modified Nelder Mead Algorithm

function RMNM($f, n, x^*, size$)
$x_1 \leftarrow$ `InitialPoint`; $j \leftarrow 0$ **repeat**

    $X \leftarrow$ `InitialSimplex`$(x_1, size)$; $X^* \leftarrow X$; $x^* \leftarrow x_1$
    while Stopping condition is not met **do**

        if not `MNMIteration`(X) **then**

            $X \leftarrow X^*$; $j \leftarrow j + 1$
            $X \leftarrow$ `Shrink`(X, j)
            if $j = n$ **then**

                $X^* \leftarrow X$; $j \leftarrow 0$

        else

            $X^* \leftarrow X$; $j \leftarrow 0$

until $f(x_1) < f(x^*)$

return x^*

Non-differentiable test instances

LND1. Generalization of MAXQ

$$f(x) = \max_{1 \le i \le n} x_i^2 \ , \quad f_{\min} = 0 \ .$$

LND2. Generalization of MXHILB

$$f(x) = \max_{1 \le i \le n} \left| \sum_{j=1}^{n} \frac{x_j}{i+j-1} \right| \ , \quad f_{\min} = 0 \ .$$

LND3. Chained LQ

$$f(x) = \sum_{i=1}^{n-1} \max\{-x_i - x_{i+1},\ -x_i - x_{i+1} + (x_i^2 + x_{i+1}^2 - 1)\} \ , \quad f_{\min} = -(n-1)\sqrt{}$$

LND4. Chained CB3 I

$$f(x) = \sum_{i=1}^{n-1} \max\{x_i^4 + x_{i+1}^2,\ (2-x_i)^2 + (2-x_{i+1})^2,\ 2e^{-x_i + x_{i+1}}\} \ , \quad f_{\min} = 2(n-1$$

LND5. Chained CB3 II

$$f(x) = \max\{\sum_{i=1}^{n-1} (x_i^4 + x_{i+1}^2),\ \sum_{i=1}^{n-1} ((2-x_i)^2 + (2-x_{i+1})^2),\ \sum_{i=1}^{n-1} (2e^{-x_i + x_{i+1}})\}, \quad f_{\min}$$

LND6. Number of active faces

$$f(x) = \max_{1 \le i \le n} \left\{ g(-\sum_{j=1}^{n} x_j),\ g(x_i) \right\}, \quad g(y) = \ln(|y| + 1), \quad f_{\min} = 0 .$$

LND7. Nonsmooth generalization of Brown function 2

$$f(x) = \sum_{i=1}^{n-1} (|x_i|^{x_{i+1}^2+1} + |x_{i+1}|^{x_i^2+1}), \quad f_{\min} = 0 .$$

LND8. Chained Mifflin 2

$$f(x) = \sum_{i=1}^{n-1} (-x_i + 2(x_i^2 + x_{i+1}^2 - 1) + 1.75|x_i^2 + x_{i+1}^2 - 1|)$$

$f_{\min}$ varies , $f_{\min} = -34.795$ for $n = 50$, $f_{\min} = -140.86$ for $n = 200$.

LND9. Chained crescent I

$$f(x) = \max\{\sum_{i=1}^{n-1} (x_i^2 + (x_{i+1}-1)^2 + x_{i+1} - 1),\ \sum_{i=1}^{n-1} (-x_i^2 - (x_{i+1}-1)^2 + x_{i+1} + 1)\} ,$$

LND10. Chained crescent II

$$f(x) = \sum_{i=1}^{n-1} \max\{x_i^2+(x_{i+1}-1)^2+x_{i+1}-1,\ -x_i^2-(x_{i+1}-1)^2+x_{i+1}+1\}\,, \quad f_{\min} =$$

function		VNS+NM			VNS+RMNM		
n	f_{opt}	CE	SUCC.	f_{min}	CE	SUCC.	f_{min}
LND1. Generalization of MAXQ							
10	0	5,657	10		5,601	10	
20	0	28,351	10		26,637	10	
30	0	67,566	10		68,105	10	
40	0	120,672	10		125,107	10	
50	0	253,598	10		243,742	10	
LND2. Generalization of MXHILB							
10	0	62,219	10		10,873	10	
20	0	5,611,105	0	0.000185	63,118	10	
30	0	2,627,389	0	0.000488	167,127	10	
40	0	1,563,528	0	0.000709	747,442	8	0.000101
50	0	995,644	0	0.000989	1,005,813	1	0.000151
LND3. Chained LQ							
10	-12.7279	9,296	10		5,277	10	
20	-26.8701	122,393	10		29,266	10	
30	-41.0122	550,353	10		62,813	10	
40	-55.1543	4,861,354	5	-55.1426	384,261	10	
50	-69.2965	2,819,445	8	-69.2823	154,489	10	
LND4. Chained CB3 I							
10	18	8,601	10		8,292	10	
20	38	36,317	10		31,394	10	
30	58	51,788	10		51,699	10	
40	78	97,106	10		93,124	10	
50	98	201,942	10		143,845	10	
LND5. Chained CB3 II							

function		VNS+NM			VNS+RMNM		
n	F_{opt}	CE	SUCC.	F_{min}	CE	SUCC.	F_{min}
LND6. Number of active faces							
10	0	199,318	10		7,776	10	
20	0	12,991,251	0	0.000218	145,763	10	
30	0	8,069,203	0	0.001117	8,189,895	1	0.000139
40	0	5,486,154	0	0.00220	5,799,417	0	0.000254
50	0	3,878,521	0	0.002894	4,088,098	0	0.000403
LND7. Nonsmooth generalization of Brown function 2							
10	0	34,591	10		8,498	10	
20	0	6,869,657	1	0.000182	43,204	10	
30	0	4,413,961	0	0.000691	79,623	10	
40	0	3,089,010	0	0.00146	226,122	10	
50	0	2,344,423	0	0.00299	999,753	9	0.0001
LND8. Chained Mifflin 2							
10	-6.5146	30,960	10		20,413	10	
20	-13.5831	787,015	10		195,133	10	
30	-20.6535	2,595,076	9	-20.6493	1,071,285	10	
40	-27.7243	4,619,462	7	-27.718	2,442,460	9	-27.7186
50	-34.795	4,166,819	2	-34.7809	3,179,337	6	-34.7871
LND9. Chained crescent I							
10	0	14,831	10		7,626	10	
20	0	81,819	10		35,149	10	
30	0	130,844	10		68,264	10	
40	0	306,036	10		73,833	10	
50	0	290,264	10		90,509	10	
LND10. Chained crescent II							

3-level VNS

- Our basic method is Skewed Variable neighborhood search (SVNS)
- SVNS is the VNS variant that allows moves to slightly worse solutions, but only if they are far from the incumbent.

Function `3L-VNS` $(x, k_{max}, t_{max}, \alpha)$
repeat

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

$x' \leftarrow$ `Shake`(x, k) /* Shaking */
$x'' \leftarrow$ `VNDS`(x') /* VNDS Local search */
if $f(x'') < f(x_{best})$ then $x_{best} \leftarrow x''$
`NeighborhoodChangeS`$(x, x'', x_{best}, k, \alpha)$ /* Skewed move */
until $k = k_{max}$;

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

- 3L-VNS is a Skewed VNS, with VNDS used instead of a local search
- The distance function $\rho(x, y)$ measures the number of different assignment in x and y.

Thank you for your attention!

nenad.mladenovic@brunel.ac.uk