Yugoslav Journal of Operations Research
30 (2020), Number 1, 3–17
DOI: https://doi.org/10.2298/YJOR190115004D

INFLUENCE OF A NEIGHBORHOOD SHAPE ON THE EFFICIENCY OF CONTINUOUS VARIABLE NEIGHBORHOOD SEARCH

Milan DRAŽIĆ
Faculty of Mathematics, University of Belgrade,
Studentski trg 16, 11000 Belgrade, Serbia
mdrazic@matf.bg.ac.rs

Received: January 2019 / Accepted: February 2019

Abstract: The efficiency of a Variable neighborhood search metaheuristic for continuous global optimization problems greatly depends on geometric shape of neighborhood structures used by the algorithm. Among the neighborhoods defined by balls in ℓ_p, $1 \leq p \leq \infty$ metric, we tested the ℓ_1, ℓ_2, and ℓ_∞ ball shape neighborhoods, for which there exist efficient algorithms for obtaining uniformly distributed points. On many challenging high-dimensional problems, our exhaustive testings showed that, popular and the easiest for implementation, ℓ_∞ ball shape of neighborhoods performed the worst, and much better efficiency was obtained with ℓ_1 and ℓ_2.

Keywords: Global Optimization, Continuous Optimization, Metaheuristic Algorithms, Variable Neighbourhood Search.

MSC: 90C59, 90C06, 90C30.

1. VNS FOR CONTINUOUS OPTIMIZATION

The goal of this paper is to explore the influence of neighborhood structure shapes on the efficiency of a Variable Neighborhood Search metaheuristic (VNS). The continuous unconstrained global optimization problem, with technically introduced box constraints, has the form

$$\operatorname*{global\,min}_{x \in S} f(x), \quad S = \{x \in R^n \mid a_i \leq x_i \leq b_i,\ i = 1, 2, \ldots, n\}$$

where $f : R^n \to R$ is a continuous function. In many cases when local minima are present, such problem can be very difficult. For large dimensions of the problem, the number of local minima grows exponentially with the dimension, making

the problem NP-hard. Exact methods can be applied only to the lower problem dimensions, so heuristic approaches are the only way to obtain good approximate solutions.

The VNS metaheuristic was first introduced by Mladenović and Hansen in [1, 2] for combinatorial optimization problems, and the applications to continuous optimization problems were given in [2, 3, 4]. The VNS approach has been implemented within the software package GLOB for unconstrained and constrained continuous optimization problems [5, 6, 7, 8] with several neighborhood structures and random point distributions.

The basic steps of the VNS metaheuristic are given as follows:

```
          Algorithm VNS
          /* Initialization */
01        Select the set of neighborhood structures N_k, k = 1,...,k_max,
          with corresponding random point distributions
02        Choose an arbitrary initial point x ∈ S
03        Set x* ← x, f* ← f(x)
          /* Main loop */
04        repeat the following steps until the stopping condition is met
05             Set k ← 1
06             repeat the following steps until k > k_max
07                  Shake: Generate at random a point y ∈ N_k(x*)
08                  Apply some local search method from y to obtain a local
                    minimum y'
09                  if f(y') < f* then
10                       Set x* ← y', f* ← f(y') and goto line 05
11                  endif
12                  Set k ← k + 1
13             end
14        end
15        Stop. Point x* is an approximate solution of the problem.
```

A good local optimizer is desirable to locate a local minimum efficiently, but for avoiding the local optima trap, the shaking step is crucial. Efficiency in finding better local minima depends on the geometry of neighborhoods and the random point distribution used in shaking step.

2. NEIGHBORHOODS INDUCED BY ℓ_p METRIC

In order to induce a set of neighborhood structures $\mathcal{N}_k$ on the solution space S, the usual approach is to use some distance function $\rho(x, y)$ that specifies the distance between points $x, y \in S$. For the continuous optimization problems, where $S \subseteq R^n$, $\rho(x, y)$ is most often defined by Euclidean, rectangular, or other ℓ_p metric:

$$\rho(x,y) = ||x-y||_p = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}, \ (1 \leq p < \infty),$$

or

$$\rho(x,y) = ||x-y||_\infty = \max_{1\le i\le n} |x_i - y_i|, \quad (p=\infty).$$

These metrics lead to different geometric shapes of neighborhoods that are to be explored. The neighborhood $\mathcal{N}_k(x)$ denotes a set of points in the k–th neighborhood of x and, using the metric ρ, it is defined as a ball

$$\mathcal{N}_k(x) = \{y \in S \mid \rho(x,y) \le \rho_k\},$$

or spherical shell

$$\mathcal{N}_k(x) = \{y \in S \mid \rho_{k-1} \le \rho(x,y) \le \rho_k\},$$

where ρ_k is the radius (size) of $\mathcal{N}_k(x)$ monotonically increasing with k.

The simplest for implementation is to generate a uniformly distributed random point in ℓ_∞ ball. If z_i are uniformly distributed points from the interval $[0,1]$, then

$$y_i = x_i + 2\rho_k(z_i - 0.5), \quad i = 1,\ldots,n\,.$$

This simplicity is the main reason why ℓ_∞ neighborhoods are so popular in implementations.

n	ℓ_1	ℓ_2	ℓ_∞	n	ℓ_1	ℓ_2	ℓ_∞
2	2	3.1416	4	20	4.3100e-13	2.5807e-02	1.0486e+06
3	1.3333	4.1888	8	30	4.0480e-24	2.1915e-05	1.0737e+09
4	0.66667	4.9348	16	40	1.3476e-36	3.6047e-09	1.0995e+12
5	0.26667	5.2638	32	50	3.7019e-50	1.7302e-13	1.1259e+15
6	0.088889	5.1677	64	60	1.3856e-64	3.0963e-18	1.1529e+18
7	0.025397	4.7248	128	70	9.8559e-80	2.4323e-23	1.1806e+21
8	0.0063492	4.0587	256	80	1.6892e-95	9.4265e-29	1.2089e+24
9	0.0014109	3.2985	512	90	8.3323e-112	1.9676e-34	1.2379e+27
10	0.00028219	2.5502	1024	100	1.3583e-128	2.3682e-40	1.2677e+30

Table 1: Unit ball volumes in three different metrics

Uniformly distributed point in ℓ_p ball for $1 \le p < \infty$ can be obtained with the acceptance-rejection method by repeatedly generating uniformly generated point in ℓ_∞ ball until it is also within the ℓ_p ball. For a large space dimension n this method is very inefficient due to huge volume differences between ℓ_∞ and ℓ_p balls, see Table 1 and Figure 1. Fortunately, there are efficient algorithms for generating uniformly distributed points inside ℓ_2 and ℓ_1 balls. Also, it is not hard to obtain uniformly distributed points in spherical shells for ℓ_∞, ℓ_2 and ℓ_1 metrics.

Along with the uniformly distributed points in ℓ_1 ball (or spherical shell), a special random point distribution in ℓ_1 was proven to be very efficient for some problems. The random point y is obtained in two steps: first a random point $z = (z_1, z_2, \ldots, z_n)$ on the ℓ_1 unit sphere is generated using the special distribution: (i) z_1 is taken uniformly on $[-1,1]$, z_k is taken uniformly from $[-A_k, A_k]$, where

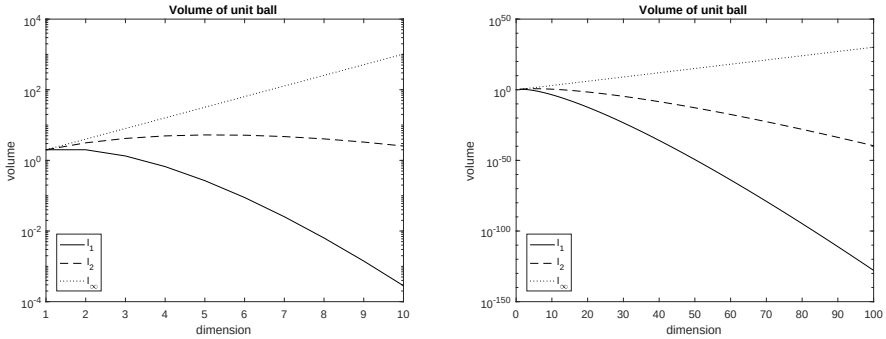

Figure 1: Volumes of ℓ_1, ℓ_2 and ℓ_∞ unit balls

$A_k = 1 - |z_1| - \ldots - |z_{k-1}|$, $k = 2, \ldots, n-1$, and the last z_n takes A_n with random sign; (ii) coordinates of z are permuted randomly. After that, a random radius uniformly distributed in $[0, \rho_k]$ (or $[\rho_{k-1}, \rho_k]$) is determined in order to get a point y in $\mathcal{N}_k(x)$.

In the further analysis, we denote the shapes of neighborhoods as follows: S1, S2, and S3 for ℓ_1, ℓ_2, and ℓ_∞ balls and spherical shells with uniformly distributed random points, and S1s for ℓ_1 ball and the spherical shell with the special distribution, described above.

The main goal of this paper is to examine the effectiveness of neighborhood shapes in the shaking step of the VNS algorithm. This step represents the diversification part of the VNS metaheuristic, which is crucial for escaping from a local minimum to better solutions in the solution space. Local search represent the intensification part of the algorithm, responsible for efficiency of finding a local minimum reachable from the generated random point. In order to discover the influence of neighborhood shapes on the VNS algorithm efficiency, we test S1, S1s, S2 and S3 neighborhood shapes both in ball and spherical shell variants, while all other algorithm parameters are fixed. The results should be valuable to software developers implementing the VNS metaheuristic for continuous optimization problems.

3. COMPUTATIONAL EXPERIMENTS

For all the tests, we used the software package GLOB, a test platform for numerical experiments with various variants of the VNS ([5]). GLOB is coded in ANSI C programming language as a single core console application. It supports neighborhood structures S1, S1s, S2, and S3, described in the previous section, and has a choice of several local minimizers. All the experimental results were obtained using a PC equipped with an Intel Core i7-6700 3.4 GHz processor with 16 GB RAM running 64bit Windows 7.

In all the tests, we used $k_{\max} = 10$ neighborhoods with radii r_i calculated automatically as a geometrical sequence of numbers. The tolerance for detecting that

the optimal solution was found was set to 1e-6 for smooth problems, and 1e-4 for non-smooth problems. Steepest descend local minimizer was used for smooth problems, and Nelder–Mead with restarts for non-smooth problems. Although some other local minimizers could be more effective, we did not use them because their role is not significant for testing effectiveness of different neighborhood shapes. In each test, the maximum run time limit was set to 30 sec. Execution time and the number of function and gradient evaluations were recorded until the global minimum was reached within the given tolerance, or the time limit was reached. A test run was considered successful if the global minimum was reached before the time limit. Each test was repeated 20 times with a different initial random point, and the average results are reported. The computer effort was calculated as $N_f + nN_g$ for the steepest descend method, and N_f for Nelder–Mead method, where N_f and N_g are the number of function and gradient calls, and n is the dimension of the problem.

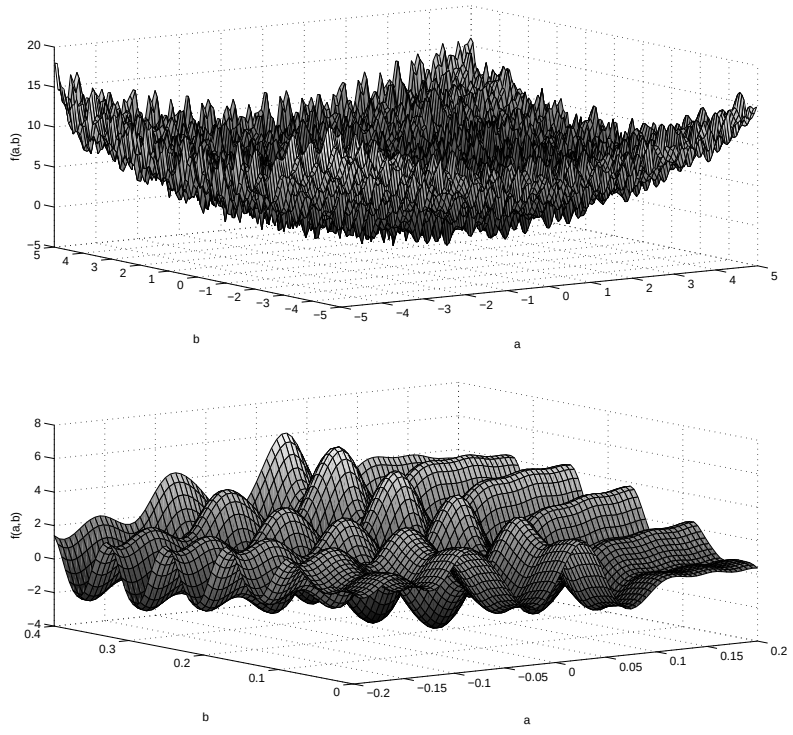

Figure 2: Trefethen 4 function with bounds [-5,5]×[-5,5], and $[-0.2, 0.2] \times [0.0, 0.4]$.

A set of test functions was chosen to represent challenging problems in high dimension solution spaces, or with many local minima, which makes them hard or impossible to solve by direct methods.

Trefethen 4 function. We start with a two dimensional problem with many local minima proposed in [9] (problem 4), see also [10]. This test instance has 2 variables $a, b \in [-5, 5]$:

$$f(a,b) = e^{\sin(50a)} + \sin(60e^b) + \sin(70\sin a) + \sin(\sin(80b)) - \\ - \sin(10(a+b)) + (a^2+b^2)/4.$$

The global optimum of $f(a,b)$ is $f_{\min} = -3.306868647$. The graph of this function on two different scales is presented in Figure 2.

		ball		shell	
n	shape	comp.eff.	time	comp.eff.	time
2	S1	101,549	0.021	134,690	0.028
	S1s	84,980	0.018	169,424	0.035
	S2	111,385	0.023	410,290	0.083
	S3	109,617	0.022	285,342	0.058

Table 2: Results for Trefethen 4 function

The computational results are presented in Table 2. The first column contains the dimension of the problem, and used neighborhood shape in the second, as denoted in the previous section. The average computer efforts and execution times in seconds from 20 runs are presented in the third and fourth columns for ball shape neighborhoods, and in the fifth and sixth column for spherical shell shapes. In every run, the global minimum was reached within the 30 sec time limit. It is evident from the results that ball neighborhood shapes are more efficient than shell shapes for the same metric. Comparing different metrics, shape S1s showed to be the most successful, although the other shapes did not perform significantly worse.

Rosenbrock function.

$$f(x) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right)$$

for $-10 \leq x_i \leq 10,\ i = 1, \ldots, n$. The global minimum is $f_{\min} = 0$. This function is a standard problem for optimization algorithms performance testing. Local minimizers as steepest descend (used here within VNS) and Nelder-Mead exhibit poor performance if used alone, without VNS algorithm.

The results for Rosenbrock function for problem dimension up to 200 are presented in Table 3. In every run the global minimum was reached within the 30 sec time limit. The first observation is that there is no significant difference in performance between ball and spherical shell variant of neighborhoods. Neighborhood shape S3 performed the worst for both small and large space dimensions. Shape S1s was slightly better than S1 in all cases. Shape S2 was competitive with S1s and S1 for $n \leq 150$, but for $n = 200$, it was significantly less efficient.

		ball		shell	
n	shape	comp.eff.	time	comp.eff.	time
10	S1	175,251	0.007	188,667	0.007
	S1s	125,028	0.005	106,047	0.004
	S2	134,256	0.006	135,278	0.006
	S3	379,827	0.014	336,029	0.013
20	S1	272,472	0.012	249,717	0.011
	S1s	240,418	0.010	241,514	0.010
	S2	213,283	0.009	224,682	0.009
	S3	479,981	0.019	523,073	0.021
50	S1	738,750	0.029	744,078	0.030
	S1s	670,541	0.026	697,932	0.027
	S2	1,001,924	0.044	827,911	0.036
	S3	10,132,164	0.440	8,279,130	0.361
100	S1	2,342,324	0.090	2,352,889	0.091
	S1s	2,189,264	0.081	2,095,220	0.077
	S2	2,044,212	0.079	2,165,929	0.087
	S3	23,679,282	1.015	27,870,293	1.206
150	S1	3,752,823	0.139	3,747,885	0.139
	S1s	3,375,741	0.120	3,315,324	0.119
	S2	3,957,138	0.152	3,679,985	0.140
	S3	40,560,739	1.695	30,906,761	1.290
200	S1	6,678,584	0.249	6,318,854	0.236
	S1s	5,564,922	0.198	5,439,822	0.194
	S2	29,566,111	1.549	11,141,342	0.529
	S3	74,860,051	3.130	105,666,113	4.462

Table 3: Results for Rosenbrock function

Ackley function. Proposed in [11, 12]:

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

for $-15 \leq x_i \leq 30,\ i = 1, \ldots, n$, has a global minimum $f_{\min} = 0$, and 45^n local minima. The graph of this function for $n = 2$ on two different scales is presented in Figure 3.

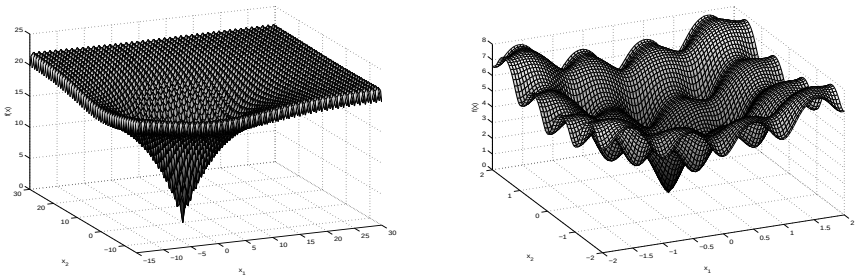

Figure 3: Ackley function for $n = 2$ with bounds [-15,30]×[-15,30], and $[-2, 2] \times [-2, 2]$.

The results for Ackley function for problem dimension up to 50 are presented in Table 4. In every run, the global minimum was reached within the 30 sec time limit. As in the previous test problems, there is no significant difference in performance between ball and spherical shell variant of the neighborhoods. Contrary

		ball		shell	
n	shape	comp.eff.	time	comp.eff.	time
10	S1	62,151	0.013	53,513	0.012
	S1s	63,889	0.014	65,680	0.014
	S2	49,823	0.011	50,588	0.011
	S3	43,811	0.009	42,456	0.009
20	S1	194,772	0.054	200,953	0.055
	S1s	215,500	0.058	228,459	0.061
	S2	142,364	0.039	155,474	0.043
	S3	138,594	0.038	135,496	0.037
30	S1	427,888	0.132	400,766	0.124
	S1s	486,548	0.145	528,480	0.158
	S2	263,078	0.082	281,063	0.087
	S3	255,680	0.078	261,403	0.081
40	S1	824,502	0.273	817,231	0.269
	S1s	1,031,772	0.324	1,205,204	0.382
	S2	457,885	0.151	431,186	0.143
	S3	410,697	0.135	436,004	0.144
50	S1	2,277,783	0.774	1,623,275	0.554
	S1s	2,247,472	0.732	2,195,793	0.720
	S2	632,166	0.218	616,375	0.215
	S3	756,055	0.259	762,336	0.262

Table 4: Experimental results for Ackley function

to Rosenbrock function, neighborhood shapes S2 and S3 performed significantly better than S1 and S1s, S1s beeing the worst overall. While S3 was better for smaller dimensions, S2 was the best for $n = 50$. Good performance of S2 shape can be explained by spherical symmetry of the function on larger scale.

Rastrigin function.

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

for $-5.12 \leq x_i \leq 5.12,\ i = 1, \ldots, n$, has a global minimum $f_{\min} = 0$ and 11^n local minima. The graph of this function for $n = 2$ is presented in Figure 4.

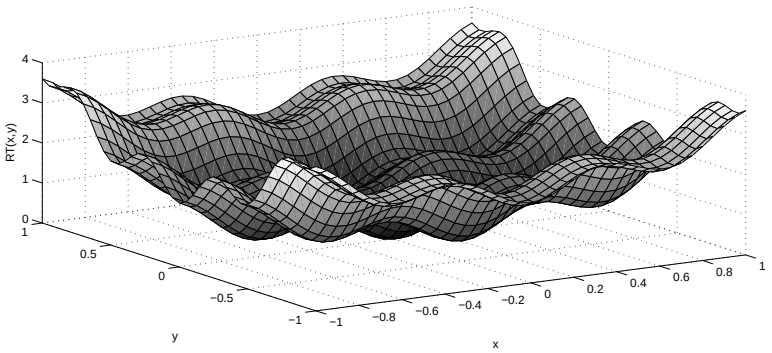

Figure 4: Rastrigin function for $n = 2$ for with bounds [-1,1]×[-1,1].

		ball				shell			
n	shape	comp.eff.	time	succ.	error	comp.eff.	time	succ.	error
10	S1	49,021	0.009	20		60,399	0.012	20	
	S1s	36,362	0.007	20		38,373	0.007	20	
	S2	78,146	0.014	20		70,869	0.013	20	
	S3	93,857	0.018	20		95,451	0.018	20	
20	S1	259,948	0.063	20		239,873	0.058	20	
	S1s	126,846	0.030	20		121,491	0.029	20	
	S2	407,110	0.095	20		462,573	0.108	20	
	S3	585,289	0.136	20		544,657	0.129	20	
30	S1	540,162	0.147	20		599,701	0.165	20	
	S1s	250,345	0.068	20		222,695	0.061	20	
	S2	1,404,324	0.372	20		1,917,442	0.508	20	
	S3	2,213,492	0.587	20		2,281,569	0.615	20	
40	S1	1,138,379	0.334	20		1,376,997	0.407	20	
	S1s	435,630	0.125	20		451,957	0.132	20	
	S2	2,651,781	0.763	20		2,980,924	0.862	20	
	S3	8,193,466	2.351	20		8,323,397	2.413	20	
50	S1	1,950,614	0.598	20		1,953,808	0.608	20	
	S1s	701,919	0.210	20		738,755	0.224	20	
	S2	3,651,184	1.116	20		3,474,303	1.063	20	
	S3	26,463,963	7.952	20		29,761,343	9.034	20	
60	S1	2,936,964	0.918	20		2,902,176	0.919	20	
	S1s	1,011,087	0.311	20		972,489	0.303	20	
	S2	3,605,867	1.140	20		3,100,591	0.984	20	
	S3	62,799,699	19.333	10	0.60	54,233,167	16.777	12	0.50
70	S1	4,199,080	1.336	20		4,166,472	1.339	20	
	S1s	1,526,418	0.475	20		1,358,786	0.433	20	
	S2	4,205,767	1.356	20		4,315,736	1.404	20	
	S3			0	2.64			0	2.54
80	S1	6,021,715	1.952	20		5,700,173	1.871	20	
	S1s	1,680,877	0.535	20		1,846,936	0.595	20	
	S2	5,729,992	1.899	20		5,669,393	1.879	20	
	S3			0	5.32			0	5.42
90	S1	7,157,711	2.367	20		7,237,914	2.429	20	
	S1s	2,397,990	0.772	20		2,261,338	0.743	20	
	S2	8,717,552	2.924	20		8,537,969	2.886	20	
	S3			0	9.90			0	9.55
100	S1	9,368,881	3.162	20		9,209,980	3.144	20	
	S1s	2,971,556	0.971	20		2,845,786	0.945	20	
	S2	14,066,449	4.844	20		12,093,366	4.207	20	
	S3			0	14.32			0	14.78

Table 5: Results for Rastrigin function

The results for Rastrigin function for problem dimension up to 100 are presented in Table 5. Since not all runs were successful in finding the global minimum within the 30 sec time limit, two new columns are introduced. Columns marked "succ." contain the number of runs in which the global minimum was reached with proposed tolerance within the time limit. In cases when not all test runs were successful, columns marked "error" contain the average optimal function value error. Columns "comp.eff." and "time" contain average computer effort and execution time in seconds for the successful runs.

Results from Table 5 show that there is no significant difference between ball and spherical shell neighborhood variants, shell variant being slightly more efficient on average. Neighborhood shape S3 performed very inefficient, and for high problem dimensions it failed to provide the global minimum in every run. Shape S1s performed the best, while S1 and S2 were comparable, S1 being consistently

better than S2.

Molecular potential energy (MPE) function. Introduced in [13], see also [6], this function was proposed for testing methods for global minimization of potential energy of molecules:

$$f(x) = \sum_{i=1}^{n} \left(1 + \cos 3x_i + \frac{(-1)^i}{\sqrt{10.60099896 - 4.141720682 \cos x_i}} \right)$$

for $0 \leq x_i \leq 5,\ i = 1, \ldots, n$, has a global minimum $f_{\min} = -0.0411183034 \cdot n$ and 3^n local minima.

		ball				shell			
n	shape	comp.eff.	time	succ.	% error	comp.eff.	time	succ.	% error
10	S1	30,015	0.010	20		34,439	0.011	20	
	S1s	13,809	0.005	20		13,359	0.004	20	
	S2	34,772	0.011	20		34,596	0.011	20	
	S3	118,409	0.038	20		80,510	0.026	20	
20	S1	194,198	0.083	20		244,582	0.106	20	
	S1s	46,391	0.021	20		51,333	0.023	20	
	S2	166,932	0.072	20		178,088	0.076	20	
	S3	10,794,746	4.489	20		7,722,668	3.228	20	
30	S1	552,584	0.267	20		739,538	0.362	20	
	S1s	110,277	0.057	20		122,178	0.063	20	
	S2	358,984	0.178	20		376,430	0.188	20	
	S3	41,512,825	19.299	1	11.60	54,488,832	25.555	1	8.96
40	S1	1,530,434	0.786	20		1,486,620	0.771	20	
	S1s	185,029	0.104	20		198,909	0.113	20	
	S2	583,497	0.317	20		594,570	0.324	20	
	S3			0	27.13			0	30.86
50	S1	3,315,594	1.763	20		2,696,179	1.442	20	
	S1s	289,291	0.173	20		317,072	0.192	20	
	S2	899,990	0.515	20		1,009,371	0.575	20	
	S3			0	39.24			0	41.44
60	S1	5,665,036	3.063	20		6,034,739	3.268	20	
	S1s	455,756	0.282	20		514,346	0.323	20	
	S2	1,941,462	1.143	20		1,555,161	0.914	20	
	S3			0	50.98			0	55.29
70	S1	7,844,683	4.273	20		8,460,620	4.628	20	
	S1s	607,348	0.391	20		669,236	0.436	20	
	S2	3,654,745	2.250	20		3,421,294	2.101	20	
	S3			0	66.50			0	64.63
80	S1	13,784,376	7.798	20		14,689,218	8.055	20	
	S1s	788,130	0.520	20		836,788	0.555	20	
	S2	6,237,038	4.039	20		6,767,309	4.363	20	
	S3			0	76.38			0	73.89
90	S1	18,575,697	10.219	20		18,658,937	10.264	20	
	S1s	1,020,315	0.688	20		1,158,930	0.787	20	
	S2	12,529,044	8.608	20		10,116,006	6.880	20	
	S3			0	78.86			0	82.95
100	S1	26,394,502	14.359	20		25,194,108	13.807	19	0.10
	S1s	1,357,828	0.925	20		1,530,488	1.061	20	
	S2	17,011,040	11.940	20		17,261,623	12.228	20	
	S3			0	86.93			0	83.93

Table 6: Results for MPE function

The results for Molecular potential energy (MPE) function for problem dimensions up to 100 are presented in Table 6. Like in the case of Rastrigin function,

neighborhood shape S3 performed very inefficient, failing to provide global minima even for problem dimension $n = 30$. Shape S1s performed the best, exceptionally better than others. Comparing shapes S1 and S2, shape S2 performed better than S1. Results also show that there is no significant difference between ball and spherical shell neighborhood variants, precedence varying from case to case. Nevertheless, for the most efficient S1s shape, ball variant performed slightly better than the spherical shell variant.

We also tested the efficiency of neighborhood shapes on four non-differentiable functions in high-dimensional space. From the set of 10 test problems proposed in [14], we chose 4 most challenging problems (see [15]). They are as follows:

Generalization of MXHILB function.

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

for $-10 \le x_i \le 10,\ i = 1, \ldots, n$, has a global minimum $f_{\min} = 0$.

		ball				shell			
n	shape	comp.eff.	time	succ.	error	comp.eff.	time	succ.	error
30	S1	134,520	0.305	20		135,012	0.317	20	
	S1s	123,354	0.281	20		123,354	0.283	20	
	S2	138,366	0.314	20		138,366	0.321	20	
	S3	142,601	0.324	20		142,601	0.328	20	
40	S1	1,831,959	6.834	20		1,880,604	6.999	20	
	S1s	572,084	2.116	20		480,358	1.767	20	
	S2	294,221	1.094	20		273,001	1.004	20	
	S3	6,631,926	24.539	3	1.14E-04	5,080,383	19.171	6	1.08E-04
50	S1	1,727,061	10.142	7	1.25E-04	3,386,658	19.828	6	1.21E-04
	S1s	1,056,715	6.152	19	9.96E-05	1,420,834	8.223	20	
	S2	609,996	3.486	20		657,980	3.842	20	
	S3			0	2.66E-04			0	2.63E-03

Table 7: Results for Generalization of MXHILB function

The results for Generalization of MXHILB function for problem dimensions $n = 30, 40, 50$ are presented in Table 7. Ball and spherical shell neighborhood variants performed the same on average, with no significant difference. For $n = 30$ all four neighborhood shapes were effective, S1s being slightly better. Neighborhood shape S3 led to optimal solutions only in several runs for $n = 40$, and none for $n = 50$. Shape S1 also couldn't reach optimal solutions in most runs for $n = 50$, and for $n = 40$ it was very inefficient. Although the shape S1s was the best for $n = 30$, $n = 40$ and $n = 50$, the shape S2 performed the best with a significant margin.

Number of active faces function.

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

for $-10 \leq x_i \leq 10,\ i = 1, \ldots, n$, has a global minimum $f_{\min} = 0$.

		ball				shell			
n	shape	comp.eff.	time	succ.	error	comp.eff.	time	succ.	error
30	S1	262,705	0.475	20		261,016	0.468	20	
	S1s	187,764	0.336	20		187,764	0.336	20	
	S2	188,018	0.338	20		188,018	0.337	20	
	S3	302,300	0.542	20		259,477	0.466	20	
40	S1	4,420,157	11.805	10	1.22E-04	5,731,825	15.208	12	1.12E-04
	S1s	1,445,912	3.770	20		1,050,530	2.727	20	
	S2	1,475,094	3.870	20		1,095,937	2.841	20	
	S3			0	2.29E-04	5,393,734	14.292	1	2.63E-04
50	S1			0	2.03E-04	5,601,173	22.458	1	2.31E-04
	S1s	2,503,346	9.856	13	1.07E-04	3,843,610	15.238	12	1.04E-04
	S2	3,965,567	15.544	6	1.10E-04	3,414,306	13.521	8	1.07E-04
	S3			0	6.91E-03			0	6.98E-04

Table 8: Results for Number of active faces function

The results for Number of active faces function for problem dimension up to $n = 50$ are presented in Table 8. There was no significant difference between ball and spherical shell neighborhood variants, advantage varying from case to case. Neighborhood shape S3 performed the worst, particularly bad for $n = 40$ and 50. Shape S1 follows, struggling to obtain optimal solutions for $n = 40$. Neighborhood shapes S1s and S2 performed much better. Shape S1s was slightly more effective than S2 for $n = 30$ and 40, and it reached optimal solutions in more cases than S2 for $n = 50$.

Chained Mifflin 2 function.

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

for $-10 \leq x_i \leq 10,\ i = 1, \ldots, n$, has a global minimum $f_{\min} = -20.6535$ for $n = 30$, $f_{\min} = -27.7243$ for $n = 40$ and $f_{\min} = -34.7950$ for $n = 50$.

		ball				shell			
n	shape	comp.eff.	time	succ.	% error	comp.eff.	time	succ.	% error
30	S1	1,676,231	2.251	20		1,524,497	2.043	20	
	S1s	948,998	1.241	20		832,751	1.091	20	
	S2	1,202,220	1.587	20		1,228,610	1.618	20	
	S3	3,156,177	4.299	20		4,078,934	5.561	20	
40	S1	3,723,598	7.522	19	0.02	4,160,843	8.467	20	
	S1s	2,575,037	5.141	20		2,792,982	5.55	20	
	S2	3,713,276	7.325	20		3,189,617	6.316	20	
	S3	6,881,290	14.064	5	0.02	8,568,826	17.647	10	0.02
50	S1	5,103,430	16.252	14	0.02	5,174,332	16.653	13	0.02
	S1s	4,133,162	12.829	18	0.02	4,214,633	13.095	19	0.02
	S2	4,464,450	13.864	19	0.02	3,748,401	11.663	20	
	S3			0	0.05			0	0.04

Table 9: Results for Chained Mifflin 2 function

The results for Chained Mifflin 2 function for problem dimension up to $n = 50$ are presented in Table 9. Ball and spherical shell neighborhood variants performed

similarly, with no clear advantage between them. Neighborhood shape S3 again performed the worst, particularly for $n = 50$. Among the other three shapes, S1 was the least competitive. Neighborhood shapes S1s and S2 performed the best, S1s being the most efficient in most cases, and S2 reaching the optimal solution in more cases than S1s for $n = 50$.

Chained crescent II function.

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

for $-10 \leq x_i \leq 10,\ i = 1, \ldots, n$, has a global minimum $f_{\min} = 0$.

		ball				shell			
n	shape	comp.eff.	time	succ.	error	comp.eff.	time	succ.	error
30	S1	1,113,540	1.615	20		1,015,723	1.470	20	
	S1s	1,107,510	1.581	20		1,425,974	2.038	20	
	S2	1,103,026	1.581	20		1,556,183	2.241	20	
	S3	1,240,956	1.819	20		1,358,937	1.995	20	
40	S1	4,993,715	10.769	8	1.18E-04	5,277,355	11.286	10	1.23E-04
	S1s	4,854,389	10.093	17	1.08E-04	4,812,370	10.065	13	1.10E-04
	S2	4,903,779	10.184	17	1.01E-04	6,450,901	13.516	19	9.99E-05
	S3	8,035,215	17.272	7	1.22E-04	10,847,742	23.424	4	1.36E-04
50	S1	3,063,428	10.147	20		2,810,161	9.259	16	1.04E-04
	S1s	2,628,553	8.488	20		1,359,642	7.521	18	1.02E-04
	S2	2,397,515	7.710	20		2,636,275	8.411	20	
	S3	4,934,812	16.752	15	1.18E-04	3,904,913	13.144	13	1.11E-04

Table 10: Results for Chained Crescent II function

The results for Chained Crescent II function for problem dimension up to $n = 50$ are presented in Table 10. Ball and spherical shell neighborhood variants performed similarly, without clear advantage between them. The VNS with restarted Nelder-Mead as a local minimizer performed noticeably more effective for $n = 50$ than for $n = 40$. The same effect was also noticed in [15] for some other modifications of Nelder-Mead minimizers. Neighborhood shape S3 was the least successful, followed by S1. Neighborhood shapes S1s and S2 were competitive in speed, S2 being better overall since it reached optimal solutions in more cases than S1s.

4. CONCLUSIONS

In this study we examined the influence of a neighborhood shape to the efficiency of the VNS metaheuristic for continuous unconstrained global optimization problems. We considered neighborhoods defined as balls in ℓ_p metrics for $p = 1, 2, \infty$, along with spherical shells, the spaces between two concentric spheres. The Most simple for software implementation, and thus most common, are ℓ_∞ balls with uniform random point distribution (marked S3 in this study). Uniformly distributed points in ℓ_2 Euclidean ball (marked S2) and ℓ_1 ball (marked S1) can be

also used in the VNS by using efficient uniform random number generators for these balls. Finally, we define a special random number distribution in ℓ_1 ball (marked S1s), which proved to be very efficient for some problems.

Our exhaustive testing on a set of challenging smooth and non-differentiable functions led to the following conclusions:

First, there was no significant difference in efficiency of VNS between ball and spherical shell neighborhood shapes. They performed very similar, except for two-dimensional Trefethen 4 function, which is considered as an easier problem due to its low dimensionality. From the implementation point of view, introducing more complicated spherical shell neighborhoods is not justified regarding the algorithm efficiency.

Second, neighborhood shapes S1, S1s, S2, and S3 exhibited remarkably different performances, especially for high dimensional problems. S3 box-shaped neighborhood, the easiest to implement, was inefficient, and often unsuccessful in higher dimensions, for all test problems except for Ackley function, for which S2 performed slightly better than S3. So, using S3 alone, or together with the other neighborhood shapes, can degrade performance of the VNS. S1s shape performed the best overall, being the best in six test problems, and close to best S2 shape in two cases. Only in one case, for Ackley function, S1s performed badly. Neighborhood shape S2 performed badly only for low dimensional Trefethen 4 problem. It performed the best in three test problems and close to best in the fourth. Comparing shapes S1 and S1s with the same geometry but different random point distribution, S1 performed worse than S1s in all problems except the Ackley function, for which they both had poor performance. So, S1s and S2 are proved to be the most successful.

Neighborhood shape S1s performed the best overall. In a few problems, S2 performed the best, S1s being close. Only in one problem, S2 was the best, and S1s performed poorly. From implementation point of view, neighborhood shape S1s (ℓ_1 ball with special random distribution) is the best choice if used alone. If the implementation can combine more neighborhood shapes, neighborhood shape S1s in combination with S2 (ℓ_2 ball with uniform random distribution) would be overall more efficient than the other shape combinations for test problems from this study.

Future work will focus on experiments with more neighborhood shapes and more random distributions. Also, combinations of neighborhood shapes within the VNS will be tested.

Acknowledgement: This research was supported by the Ministry of Education, Science and Technology of Serbia, project number 174010.

REFERENCES

[1] Mladenović, N., Hansen, P., "Variable neighborhood search", *Computers & Operations Research*, 24 (1997) 1097–1100.

[2] Hansen, P., Mladenović, N., "An introduction to variable neighborhood search", *Metaheuristics*, Springer, Boston, MA, 1999 (433–458).
[3] Brimberg, J., Hansen, P., Mladenović, N., Taillard, E., "Improvements and comparison of heuristics for solving the uncapacitated multisource weber problem", *Operations Research*, 48 (3) (2000) 444–460.
[4] Mladenović, N., Petrović, J., Kovačević-Vujčić, V., Čangalović, M., "Solving spread spectrum radar polyphase code design problem by tabu search and variable neighborhood search", *European Journal of Operational Research*, 151 (2003) 389–399.
[5] Dražić, M., Kovačević-Vujčić, V., Čangalović, M., Mladenović, N., "GLOB – a new VNS-based software for global optimization", *Global optimization*, Springer, Boston, MA, 2006 (135–154),.
[6] Dražić, M., Lavor, C., Maculan, N., Mladenović, N., "A continuous variable neighborhood search heuristic for finding the three-dimensional structure of a molecule", *European Journal of Operational Research*, 185 (3) (2008) 1265–1273.
[7] Mladenović, N., Dražić, M., Kovačević-Vujčić, V., Čangalović, M., "General variable neighborhood search for the continuous optimization", *European Journal of Operational Research*, 191 (3) (2008) 753–770.
[8] Carrizosa, E., Dražić, M., Dražić, Z., Mladenović, N., "Gaussian variable neighborhood search for continuous optimization", *Computers & Operations Research*, 39 (9) (2012) 2206–2213.
[9] Trefethen, L., "A hundred-dollar, hundred-digit challenge", *SIAM news*, 35 (1) (2002) 01–02.
[10] Audet, C., Béchard, V., Digabel, S., "Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search", *Journal of Global Optimization*, 41 (2) (2008) 299–318.
[11] `http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/-TestGO_files/Page295.htm`.
[12] `http://tracer.lcc.uma.es/problems/ackley/ackley.html`.
[13] Lavor, C., Maculan, N., "A function to test methods applied to global minimization of potential energy of molecules", Numerical algorithms, 35(2) (2004) 287–300.
[14] Haarala, M., Miettinen, K., Mäkelä, M., "New limited memory bundle method for large-scale nonsmooth optimization", *Optimization Methods and Software*, 19 (6) (2004) 673–692.
[15] Dražić, M., Dražić, Z., Mladenović, N., Urošević, D., Zhao, Q., "Continuous variable neighbourhood search with modified Nelder–Mead for non-differentiable optimization", *IMA Journal of Management Mathematics*, 27 (1) (2014) 75–88.