Les Cahiers du GERAD ISSN: 0711–2440

Gaussian Variable Neighborhood Search for Continuous Optimization

E. Carrizosa, M. Dražić,
Z. Dražić, N., Mladenović

G–2011–26

June 2011

Gaussian Variable Neighborhood Search for Continuous Optimization

Emilio Carrizosa

Faculdad de Matemáticas
Universidad de Sevilla
Spain
ecarrizosa@us.es

Milan Dražić
Zorica Dražić

Faculty of Mathematics
University of Belgrade
Serbia
mdrazic@sezampro.rs; lolaz@sezampro.rs

Nenad Mladenović

GERAD & School of Mathematics
Brunel University-West London
United Kingdom
nenad.mladenovic@brunel.ac.uk

June 2011

Les Cahiers du GERAD
G–2011–26

Abstract

Variable Neighborhood Search (VNS) has shown to be a powerful tool for solving both discrete and box-constrained continuous optimization problems. In this note we extend the methodology by allowing also to address unconstrained continuous optimization problems.

Instead of perturbing the incumbent solution by randomly generating a trial point in a ball of a given metric, we propose to perturb the incumbent solution by adding some noise, following a Gaussian distribution. This way of generating new trial points allows one to give, in a simple and intuitive way, preference to some directions in the search space, or, contrarily, to treat uniformly all directions. Computational results show some advantages of this new approach.

Key Words: Global optimization, Nonlinear programming, Metaheuristics, Variable neighborhood search, Gaussian distribution.

Résumé

La recherche à voisinage variable est un outil puissant pour résoudre les problèmes d'optimisation discrète ou continue avec contraintes d'intervalles. Dans cette note, nous étendons la méthode à des problèmes d'optimisation continue sans contrainte.

Au lieu de perturber la solution dans une boule d'une métrique donnée, nous proposons de perturber la solution selon une distribution gaussienne. Avec cette procédure, il est facile de donner une préférence à certaines directions de recherche, ou, au contraire, de chercher uniformément dans toutes les directions. Les résultats montrent certains avantages de cette nouvelle approche d'un point de vue numérique.

Mots clés : optimisation globale, programmation non linéaire, métaheuristiques, recherche à voisinage variable, distribution gaussienne.

Acknowledgments: The research of E. Carrizosa is partially supported by grants MTM2009-14039 (Ministerio de Educación y Ciencia, Spain) and FQM329 (Junta de Andalucía, Spain). Part of this research was done while N. Mladenović was visiting the Instituto de Matemáticas de la Universidad de Sevilla (Grant SAB2009-0144, Ministerio de Educación y Ciencia, Spain). The last three authors are also partially supported by Serbian Ministry of Science, project number 144007.

1 Introduction

In this paper we consider an unconstrained Nonlinear Program (NLP) of the form

$$\underset{x\in\mathbb{R}^n}{\text{global min}}\, f(x) \qquad (NLP)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuous function. No further assumptions are made on f. In particular f does not need to be convex or smooth, and it may be obtained as the output of a numerical subroutine.

Unconstrained NLPs naturally arise in many applications, e.g. in advanced engineering design, data analysis, financial planning, risk management, scientific modelling, chemistry, etc. In many cases of practical interest such problems are very difficult because of the presence of many local minima, the number of which may grow exponentially with the dimension of the problem. For problems of even moderate size, methods that offer a guarantee of finding the true global minimum are too time-consuming. Hence, different (meta)heuristic approaches, which rely heavily on computer power, have been developed, see [1] for a recent presentation of the state-of-the-art.

A benchmark metaheuristic is Variable Neighborhood Search, [2, 3, 4, 5, 6, 7], which in its most popular version takes the following form:

	`Algorithm VNS`
	/* *Initialization* */
01	Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{\max}$
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 at random a point $y \in \mathcal{N}_k(x^*)$
08	Apply some *local search* method from y to obtain a local minimum y'
09	**if** $f(y') < f^*$ **then**
10	Set $x^* \leftarrow y'$, $f^* \leftarrow f(y')$ 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 of using several geometric neighborhood structures and random distributions in shaking step led to the `Glob-VNS` variant of VNS [2, 4], which turned out to be noticeably more efficient compared to variants with fixed geometry and distribution. In most cases `Glob-VNS` uses $m = 4$ (*geometry, distribution*) pairs, but the user can arbitrarily set their number and combinations.

```
Algorithm Glob-VNS
/* Initialization */
Select the pairs (G_l, P_l), l = 1,...,m of geometry structures and
distribution types and a set of radii ρ_i, i = 1,...,k_max
Choose an arbitrary initial point x ∈ S
Set x* ← x, f* ← f(x)
/* Main loop */
repeat the following steps until the stopping condition is met
    Set l ← 1
    repeat the following steps until l > m
        Form the neighborhoods N_k, k = 1,...,k_max using
        geometry structure G_l and radii ρ_k
        Set k ← 1
        repeat the following steps until k > k_max
            Shake: Generate at random a point y ∈ N_k(x*) using
            random distribution P_l
            Apply some local search method from y to obtain a local
            minimum y'
            if f(y') < f* then
                Set x* ← y', f* ← f(y') and goto line 05
            endif
            Set k ← k + 1
        end
        Set l ← l + 1
    end
end
Stop. Point x* is an approximate solution of the problem.
```

In order to make the algorithm `Glob-VNS` applicable, some choices must be done. First, a *geometry* $\mathcal{G}$ for the neighborhood structure $\mathcal{N}_k(x)$, $k = 1, \ldots, k_{\max}$, $x \in \mathbb{R}^n$, is needed. The most popular choices are

$$\mathcal{N}_k(x) = \{y |\ \rho(x, y) \leq \rho_k\}, \tag{1}$$

or

$$\mathcal{N}_k(x) = \{y |\ \rho_{k-1} < \rho(x, y) \leq \rho_k\}. \tag{2}$$

Metric $\rho(\cdot)$ is usually an ℓ_p distance, $1 \leq p \leq \infty$, [2, 3, 4, 5], typically $p = 1, 2, \infty$. The geometry of neighborhood structures $\mathcal{G}$ is thus determined by the choice of metric $\rho(\cdot)$, and $\mathcal{N}_k(x)$ is determined by $\mathcal{G}$ and ρ_k. Both [2] and [3] use neighborhoods as defined in (2). In [3] the ℓ_∞ norm is used, while in [2] the choice of metric is either left to the analyst, or changed automatically in some predefined order. The radii ρ_k are monotonically increasing in k, and they are either defined by the user or calculated automatically in the optimization process.

One also needs to specify the *distribution* $\mathcal{P}$ used for obtaining the random point y from $\mathcal{N}_k(x)$ in the *shaking* step. Drawing y uniformly in $\mathcal{N}_k(x)$ is the most popular choice. The computational burden for generating points uniformly distributed in $\mathcal{N}_k(x)$ will obviously depend on the geometry of the set. Whereas this issue is trivial for the ℓ_∞ norm, things are more complicated if other norms are used. Fore instance, to generate random points uniformly distributed in the unit sphere B of the the Euclidian norm, different algorithms can be used. For instance, in the acceptance-rejection method, one generates a random point uniformly distributed in the cube $Q = [-1, 1]^n$, the point is discarded if it lies outside B, and the process is repeated until a point falling into B is found. This approach is simple to implement but it is suitable only for small space dimensions. Indeed, since the ratio of the volumes of B and Q tends to zero, this approach becomes inefficient when the dimension increases. Alternatively, one can use spherical coordinates to overcome this problem, but the algorithm uses computationally costly trigonometric functions. A more efficient proposal has two steps: (i) using Ahrens-Dieter algorithm ([8, 9]) for fast generation of gaussian

univariate variables, one generates $n-$dimensional random vectors which, after normalization, gives us a point uniformly distributed on a unit sphere (ii) a random radius r is generated taking into account that the density function for r is proportional to the surface of the sphere of radius r, that is, to Cr^{n-1}. The cumulative distribution function $F(x)$ and its inverse can be easily computed, yielding $r = F^{-1}(u)$ where $u \in [0, 1]$ is uniformly distributed. Observe that by an appropriate modification of step (ii) we can also efficiently generate uniformly distributed point from a neighborhood of the type (2).

In the `Glob-VNS` implementation, the user can specify the geometry structures $\mathcal{G}_l$ induced by the ℓ_1, ℓ_2 or ℓ_∞ metric in (1) and (2), and $\mathcal{P}_l$ as uniform or a hypergeometric distribution [2, 4]. The user can arbitrarily define the number and order of combinations $(\mathcal{G}_l, \mathcal{P}_l),\ l = 1, \ldots, m$.

Although, as discussed above, drawing y uniformly in $\mathcal{N}_k(x)$ is the most popular choice, it is not the only possible one. For instance, the method proposed in [7] can be seen as taking ℓ_2 in (2), and then shaking by sampling following a mixture of one-dimensional uniform distributions along the directions given by the different eigenvectors of the hessian at x. Assigning different probabilities to the different eigenvectors (and thus to the different directions) allows one to give higher priority to those directions considered to be more promising. Observe that second-order information is used, thus limiting the applicability of the procedure to smooth functions; moreover, we believe that much efforts may be unnecessarily taken to get neighbors that adapt to the curvature around incumbent, since this step is followed by a local search which also takes into account the local behavior of the function.

With respect to the *local search* procedures, different proposals have been made. The commercial solver SNOPT [10] is proposed in [3], a trust-region type method is suggested in [7], while in [2] the analyst has a menu of six different local search optimizers. The *local search* and the *shaking* stages can be merged and done simultaneously, and the metric ρ may vary from one neighborhood to the next. This is done, for instance, in [6]. Two different neighborhoods, $\mathcal{N}_1(x)$ and $\mathcal{N}_2(x)$, are used. With $\mathcal{N}_1(x)$, random directions from the current point x are generated, and a one-dimensional search along the direction is performed. This is of course equivalent to take $\mathcal{N}_1(x)$ as in (1), with ρ as the Euclidean distance, and combine it with a local search through the line passing through x and the point y generated in $\mathcal{N}_1(x)$. It is proposed to repeat this process r times, r being a parameter. This can be seen as imposing a multistart method, with r trials, on top of the local search strategy. The second neighborhood $\mathcal{N}_2(x)$ proposed in [6] has the form of (1), now ρ taken as the ℓ_∞ norm.

It is interesting to note that computational results reported by all VNS-based heuristics were very promising, usually outperforming other recent approaches from the literature. However, one should observe that, since the number $k_{\max}$ of different neighbors is assumed to be finite, one cannot reach any point in $\mathbb{R}^n$ from an incumbent solution, and thus one may not reach the region of attraction of the true global optimum. In other words, the strategy is most promising when the problem under consideration is not unconstrained but box-constrained,

$$\operatorname*{global\,min}_{x \in S} f(x), \tag{3}$$

with $S = \{x \in \mathbb{R}^n \mid a_i \leq x_i \leq b_i,\ i = 1, 2, \ldots, n\}$. For unconstrained NLPs, lower and upper bounds on variables are set to arbitrarily large negative and positive values, respectively. Then the radii ρ_k are chosen so that the full sequence of neighborhoods $\mathcal{N}_k(x)$, $1 \leq k \leq k_{\max}$ allows one to reach any point in the (huge) box S, assumed to contain an optimal solution of (NLP). This is a drawback of the existing versions of VNS, since, if we want the largest neighborhoods to reach any point in S, a very large number $k_{\max}$ should be chosen. The so-obtained radii may then be less efficient for the problem.

In this note we suggest a new variant of VNS which avoids this limitation. Instead of defining a sequence of neighborhoods $\mathcal{N}_1(x), \ldots, \mathcal{N}_{k_{\max}}(x)$, and shaking by sampling (eventually from a uniform distribution) on $\mathcal{N}_k(x)$, we define a sequence of *shaking distributions* $\mathcal{P}_1(x), \ldots, \mathcal{P}_{k_{\max}}(x)$, and the trial points are drawn from such shaking distributions. For simplicity we assume that each $\mathcal{P}_k(x)$ is a n-variate Gaussian distribution centered at x, and call this version *Gaussian VNS*, `Gauss-VNS`. With this approach, one can jump from an incumbent x to any trial point in the space, and thus the region of attraction of the true global optimum is

reachable from any starting point. Moreover, by an adequate choice of the covariance matrices of the shaking distributions, higher priority to more promising search directions can be given, as in [7], or we can treat equally all directions by taking diagonal covariance matrices for the gaussian distributions. Computational results on standard test functions from the literature show that the average number of function evaluations needed to find the global minimum is smaller than in existing VNS-based methods.

The paper is organized as follows. In Section 2, details of our `Gauss-VNS` method are given. Section 3 reports our computational experience on test instances from the literature. The paper ends with Section 4, where some concluding remarks are given and future lines of research are outlined.

2 Gaussian VNS

We generalize the paradigm of neighborhoods so that problems with unbounded domains can be addressed. The idea is to replace the class $\{\mathcal{N}_k(x)\}_{1\leq k\leq k_{\max}}$ of neighborhoods of point x by a class of *probability distributions* $\{\mathcal{P}_k(x)\}_{1\leq k\leq k_{\max}}$. The next random point in the shaking step is generated using the probability distribution $\mathcal{P}_k(x)$.

If we take as $\mathcal{P}_k(x)$ the uniform (or some other previously mentioned) distribution with support $\mathcal{N}_k(x)$, then we recover the classical approach. For a distribution with unbounded support, a natural choice is the multivariate Gaussian distribution. We assume in what follows that each $\mathcal{P}_k(x)$ is a multivariate Gaussian distribution with mean x and covariance matrix Σ_k. In other words, the trial point in the shaking process is generated from an n-dimensional random vector y with density function of the form

$$\varphi(y) = \frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(y-x)^\top \Sigma_k^{-1}(y-x)}$$

	`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.

From the implementation point of view it is important to have an efficient generator for Gaussian random points. Random values following $\mathcal{P}_k(x)$ are easily obtained from n independent values $z_1, \ldots, z_n$ of a univariate Gaussian distribution with mean 0 and variance 1. Indeed, if $\Sigma_k = L_k L_k^\top$ is the Cholesky decomposition of the symmetric positive definite matrix Σ_k, then it turns out that the random vector $x + Lz$, with $z = (z_1, \ldots, z_n)$, is distributed as $\mathcal{P}_k(x)$. Hence generating a random vector from $\mathcal{P}_k(x)$ is reduced to first calculating the Cholesky decomposition of Σ_k and then generating n univariate independent Gaussian with 0 mean and

variance 1. The process is even simpler if one assumes the covariance matrices Σ_k to be multiple of the identity matrix I,

$$\Sigma_k = \sigma_k^2 I, \qquad k = 1, 2, \ldots, k_{\max}, \tag{4}$$

since in such a case the Cholesky decomposition is simply $\Sigma_k = (\sigma_k I)\,(\sigma_k I)^\top$. In this particular case coordinates z_i of the random vector z are univariate independent Gaussian variables with 0 mean and variance σ_k.

Comparing `Gauss-VNS` with `Glob-VNS`, we see that `Gauss-VNS` has less parameters to be specified. For `Glob-VNS` the user must specify the number and combination of (*geometry, distribution*) pairs $(\mathcal{G}_l, \mathcal{P}_l)$, $l = 1, \ldots, m$, and radii ρ_k, $k = 1, \ldots, k_{\max}$. However, in `Gauss-VNS` all neighborhoods are equal (to S or $\mathbb{R}^n$) and only one family of distribution (Gaussian) is used. With the obvious choice of $\Sigma_k = \sigma_k^2 I$ only the variances σ_k, $k = 1, \ldots, k_{\max}$ should be specified.

3 Numerical experiments

Recently many metaheuristic based methods for solving continuous global optimization have been proposed. Heuristics are usually compared by the number of function evaluations until the optimal solution is reached. However, such a comparison is not easy. Indeed, different methods use different stopping conditions, see e.g. [11] for a list of commonly used stopping criteria, different precision, they are implemented in different programming languages and they run in different computers. Despite of those difficulties for direct comparison, we decided to compare our `Gauss-VNS` heuristic only with the four recent VNS-based approaches that were already briefly described in the Introduction of this note. Note that these VNS-based methods compared favorable with other metaheuristics used for the comparison purposes in papers where they were proposed, and thus comparison against other types of metaheuristics is not performed here. The computational effort, measured by means of the number of function evaluations, is given for the following methods:

- `Gauss-VNS` - this paper;
- `Glob-VNS` - Dražić et al. (2006), [2];
- `VNS-1` - Bielaire et al. (2010), [7];
- `VNS-2` - Toksari, Güner (2007) [6] and
- `VNS-3` - Audet et al. (2008), [12].

Software platform. The two methods `Glob-VNS` and `Gauss-VNS` were integrated into the package `GLOBC`, a test platform for numerical experiments with VNS. It is recently expanded with algorithm `Gauss-VNS`. `GLOBC` is coded in C+ computer language. As mentioned earlier, the quality of any method for solving (NLP) is usually measured by the number of function evaluations until the optimal (or best known) solution is reached. However, the computational effort strongly depends on a number of input parameters such as a tolerance value for the stopping rule or the choice of the local optimizer. Here we use the same package `GLOBC`, which contains different heuristic algorithms, but they use mostly the same set of parameters. This approach gives us a more realistic picture of effectiveness of the new algorithm `Gauss-VNS` compared to the existing ones. The total execution time is set to be sufficiently large so that the global minimum is found in every test run. Then, we measure the average computer effort until such optimum was found.

VNS parameters. The best-found parameterizations for `Glob-VNS`, as given in [2], are also used for `Gauss-VNS`. In other words, no efforts have been made to estimate the parameters values in favor or `Gauss-VNS`, which competes against the best-found parameterizations of `Glob-VNS`. The number of neighborhoods for both `Gauss-VNS` and `Glob-VNS` are fixed to $k_{\max} = 5$, and all tolerances for the local search are set to $1e-4$. The remaining parameters of `Glob-VNS` (`Gauss-VNS`), namely the radii ρ_k (deviations σ_k) were tuned for each instance, but always following a geometric progression. Typical choices for ρ_k, (σ_k) $k = 1, \ldots, 5$ were (0.1, 0.2, 0.5, 1.0, 2.0) or (0.1, 0.5, 1.0, 3.0, 5.0). The local-search procedure was chosen from a list of well-known methods: Nelder-Mead (NM), Hooke-Jeeves (HJ), Rosenbrock (RO), Steepest Descent (SD), Fletcher-Powell (FP) and Fletcher-Reeves (FR). One-dimensional search is done with the Quadratic Approximation method

(QA). The local-search procedure was not optimized for `Gauss-VNS` as well: we simply took the best options obtained for `Glob-VNS`, which gives a clear advantage for `Glob-VNS` against `Gauss-VNS`. Despite of that, it appears that our `Gauss-VNS` is comparable, and sometimes even better than the optimized `Glob-VNS`.

For `Gauss-VNS`, we assumed the covariance matrices Σ_k to have the form (4). The Ahrens-Dieter algorithm ([8, 9]) was implemented to generate univariate Gaussian variables in the *shaking* phase.

Comparison on standard test instances. Experiments were performed on a set of standard test functions from literature ([7, 13]). We measured the computer effort (the number of function calls plus n times the number of gradient calls) made until the algorithm finds the global minimum. Each experiment was repeated 10 times, and the average values are taken as results.

The dimensionality of most of these standard test problems is rather low, and should not be considered at all as the size limit of problems VNS can successfully handle. However, the results give a clear picture about how computational effort varies between the different versions analyzed.

The numerical results are summarized in Table 1. The results in columns 4 (`VNS-1`) and 5 (`VNS-2`) are the same as reported in [7] and [6] respectively. The next four columns contain information about the local minimizer used in `Glob-VNS` and `Gauss-VNS` and computational effort for `Glob-VNS` and `Gauss-VNS`. Finally, the last column marked *% Improvement* contains the ratio of previous two calculated values, namely

$$\frac{f_{\texttt{Glob-VNS}} - f_{\texttt{Gauss-VNS}}}{f_{\texttt{Gauss-VNS}}} \cdot 100\%.$$

The smallest computational effort among 4 methods is boldfaced.

Table 1: Standard test functions

function		n	Computer effort		local	Computer effort		% Improvement
			`VNS-1`	`VNS-2`	minim.	`Glob-VNS`	`Gauss-VNS`	
Branin	RC	2	153	308	FR	131	**112**	16.96%
asom	ES	2	167	–	HJ	163	**148**	10.14%
Goldstein and Price	GP	2	–	206	NM	260	**116**	124.14%
Rastrigin	RA	2	246	–	RO	206	**199**	3.52%
Hump	HM	2	335	–	NM	160	**80**	100.00%
Shubert	SH	2	**366**	–	FR	382	591	-35.36%
De Joung	DJ	3	104	–	FP	38	**26**	46.45%
Hartmann	H3,4	3	249	521	NM	246	**223**	10.31%
Hartmann	H6,4	6	735	1244	HJ	**397**	448	-11.38%
Colville	CV	4	854	–	NM	669	**497**	34.61%
Shekel	S4,10	4	590	988	SD	599	**399**	50.13%
Griewank	GR	6	807	–	SD	135	**126**	7.14%
Dixon	DX	10	2148	–	FP	1640	**1576**	4.06%
Rosenbrock	R2	2	556	–	NM	158	**125**	26.40%
Rosenbrock	R5	5	**1120**	–	NM	1286	1308	-1.68%
Rosenbrock	R10	10	2653	–	FP	**2357**	2561	-7.97%
Rosenbrock	R50	50	**11934**	–	FR	38621	37901	1.90%
Rosenbrock	R100	100	**30165**	–	FR	147274	122446	20.28%
Zakharov	Z2	2	251	–	FR	179	**133**	34.59%
Zakharov	Z5	5	837	–	FR	728	**461**	57.92%
Zakharov	Z10	10	1705	–	FR	1142	**1010**	13.07%
Zakharov	Z50	50	17932	–	FR	**4304**	5302	-17.28%
							Average	22.17%

The results from Table 1 in columns `VNS-1` [7] and `VNS-2` from [6] are not directly comparable with those reported for `Glob-VNS` and `Gauss-VNS` due to different stopping criteria and tolerance parameters used in corresponding programs, but they are nevertheless illustrative. The remarkable performance of `VNS-1` for R50 and R100 functions are consequence of a better local minimizer used, i.e., a truncated conjugate gradient algorithm for the trust-region problem. However, the Rosenbrock test functions have only one local minimum, and thus they are not suitable for comparing the global optimization methods.

As Table 1 shows, `Gauss-VNS` heuristic outperformed `Glob-VNS` in most of the standard test instances. Since the main goal of our numerical test was to compare VNS-based heuristics, we fixed in advance most of parameters for all test functions. This implies that these results should not be considered as best-possible for each test function. Further, `Gauss-VNS` used the same parameters as `Glob-VNS`, which makes it possible to perform even better with other configuration of parameters.

Comparison on large size test problems. The behavior of our new algorithm was also tested in problems of higher dimensionality. Three challenging test problems from the literature were chosen, RAn, MPEn, ACn.

Rastrigin function (RAn) (see Figure 1 for $n = 2$):

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

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

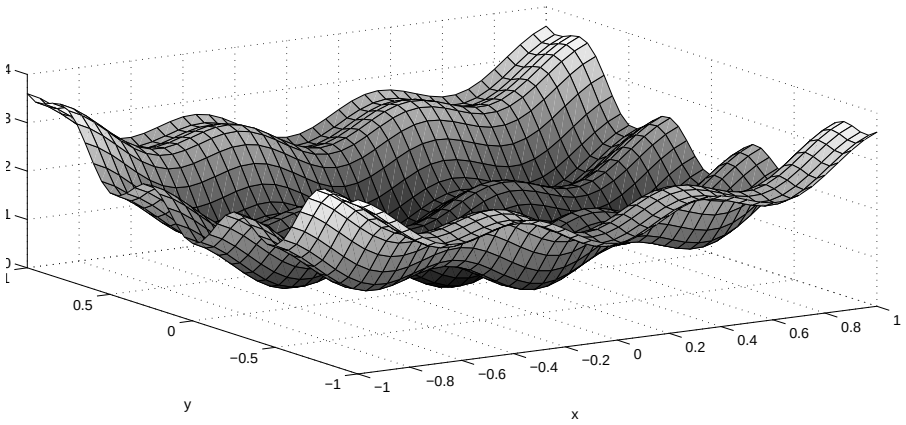

Figure 1: Rastrigin test function RA2

Molecular potential energy function (MPEn) function [14, 15]:

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} \left(1 + \cos 3x_i + \frac{(-1)^i}{\sqrt{10.60099896 - 4.141720682\cos x_i}}\right), \\ & 0 \leq x_i \leq 5, \quad i = 1, \ldots, n, \quad f_{\min} = -0.0411183034 \cdot n \end{aligned}$$

and Ackley (ACn) function [16, 17]:

$$\begin{aligned} f(x) &= 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum\nolimits_{i=1}^{n}\cos(2\pi x_i)\right) \\ & -15 \leq x_i \leq 30, \quad i = 1, \ldots, n, \quad f_{\min} = 0. \end{aligned}$$

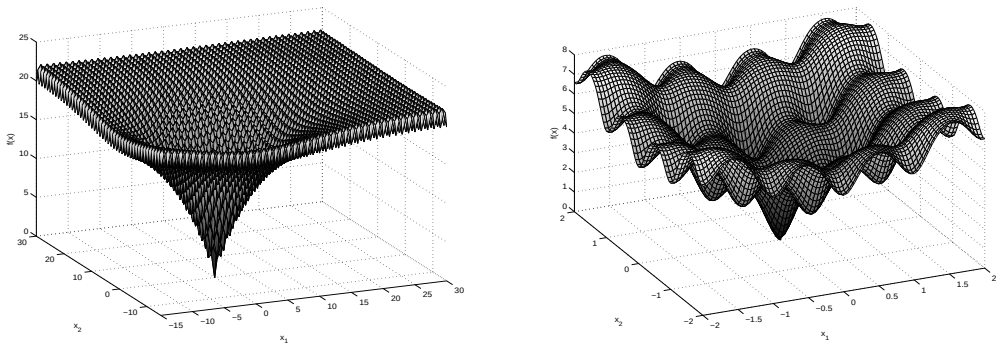

Figure 2: Ackley function with bounds [-15,30]×[-15,30], and $[-2, 2] \times [-2, 2]$

All three functions have an exponential number of local minima (11^n for RAn, 3^n for MPEn and 45^n for ACn). The tolerances were set to 1e-5. The comparison between `Glob-VNS` and `Gauss-VNS` on Rastrigin, Molecular potential energy and Ackley functions are presented in Tables 2, 3 and 4, where k_{max} is increased to 15 and 10 respectively.

It appears that no systematic advantage of one heuristic over the others exists. `Glob-VNS` performed better for RAn, while `Gauss-VNS` was better for MPEn and superior for ACn. We believe that `Glob-VNS` performs better for instances with local minima distributed rectangularly (RAn) while `Gauss-VNS` gives better results in case of spherical-like function shapes. For Ackley function `Gauss-VNS` even improves its superiority for higher dimensions.

Comparison on a two-dimensional instance. Finally we compare `Glob-VNS` and `Gauss-VNS` with Mesh Adaptive Direct Search method (MADS) coupled with VNS (`VNS-3`) [12]. To do this, one test instance from [12] is used. This test instance, that was firstly suggested in [18] (problem 4), has 2 variables $a, b \in [-5, 5]$:

$$f(a,b) = e^{\sin(50a)} + \sin(60e^b) + \sin(70\sin a) + \sin(\sin(80b)) - \\ - \sin(10(a+b)) + (a^2+b^2)/4. \tag{5}$$

The global optimum (a^*, b^*) of $f(a, b)$ is known, namely,

$$f(a^*, b^*) = f(-0.024, 0.211) = -3.307.$$

The graph of the objective function (5) is shown in Figure 3. Beside the plot on the entire domain $[-5, 5] \times [-5, 5]$ we zoom in $f(a, b)$ on the rectangle $[-0.2, 0.2] \times [0.0, 0.4]$. As initial solution we use $(-3, 3)$, the same point used in [12], where several algorithmic variants of MADS had been tested. Those that contain VNS are denoted as C, E, F, E+F. The results for `Glob-VNS`, `Gauss-VNS` and `VNS-3` (C,D,E,F,E+F) are summarized in Table 5. Again, the performances of these methods are not easy to compare. The stopping criterion of the `VNS-3` stops in most cases before the global minimum is found. This explains why the average errors in the best function values are rather big for this test function. On the other hand, `Glob-VNS` and `Gauss-VNS` are designed to search for the solution for a longer time (as much as we can afford) with better chances to find the global minimum for harder problems. In Table 5 the overall computational effort until the program stops is also presented. In order to compare with `VNS-3`, we also limited the number of VNS meta-iterations to examine how well the algorithms behave in given time. In all cases tolerances were set to 1e-4, Hook-Jeeves was used as local search algorithm and average values for 10 test runs are presented (always with the same initial solution (3,3)). It appears that `Glob-VNS` is more efficient than `Gauss-VNS`. Moreover, both VNS versions found approximate solutions with less function evaluations than MADS (`VNS-3`). For instance, while `VNS-3` found the solution with an error of 7.62% in 6146 function evaluations, `Glob-VNS` found it with 5.16% with 4374 evaluations. For an error of 9.01% `VNS-3` took 5182 evaluations while `Gauss-VNS` needed 3930 for an error of 8.92%.

Table 2: Rastrigin function

function	n	local minim.	$k_{\max}$	Computer effort `Glob-VNS`	Computer effort `Gauss-VNS`	% Improvement
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: Molecular potential energy function

function	n	local minim.	$k_{\max}$	Computer effort `Glob-VNS`	Computer effort `Gauss-VNS`	% Improvement
MPE10	10	SD	10	8102	**5015**	61.56%
MPE20	20	SD	10	26647	**21172**	25.86%
MPE30	30	SD	10	66441	**49162**	35.15%
MPE40	40	SD	10	118006	**109468**	7.80%
MPE50	50	SD	10	202280	**143309**	41.15%
MPE100	100	SD	10	**830343**	1183873	-29.86%
MPE150	150	SD	10	**2353315**	2802372	-16.02%
MPE200	200	SD	10	7683209	**5859705**	31.12%
					Average	19.59%

Table 4: Ackley function

function	n	local minim.	$k_{\max}$	Computer effort `Glob-VNS`	Computer effort `Gauss-VNS`	% Improvement
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%

4 Conclusions

In this note a new VNS-based heuristic for solving continuous unconstrained optimization problems has been presented. It is called `Gauss-VNS` since the Gauss distribution is used for generating random points x' from the k^{th} neighborhood of the incumbent solution x ($x' \in \mathcal{N}_k(x)$). This simplifies previous VNS-based methods, as proposed in [4], where four different distribution types were considered: (i) uniform in the ball of the ℓ_1 norm; (ii) uniform in the ball of the ℓ_∞ norm; (iii) hypergeometric in the ball of the ℓ_1 norm and (iv) special distribution in the ball of the ℓ_∞ norm. Instead, the normal distribution with different values of the parameter σ_k^2 is used. This way, the concept of neighborhood $\mathcal{N}_k(x)$ is replaced by the concept of (Gaussian) distribution $\mathcal{P}_k(x)$. Beside simplifying the previous algorithm, our Gauss-VNS has two additional desirable properties: (i) it can be used in cases where there are no bounds on variables (box constraints); (ii) it allows us to generate uniformly distributed points in the ball of the ℓ_2 norm (previously only ℓ_1 and ℓ_∞ norms were used).

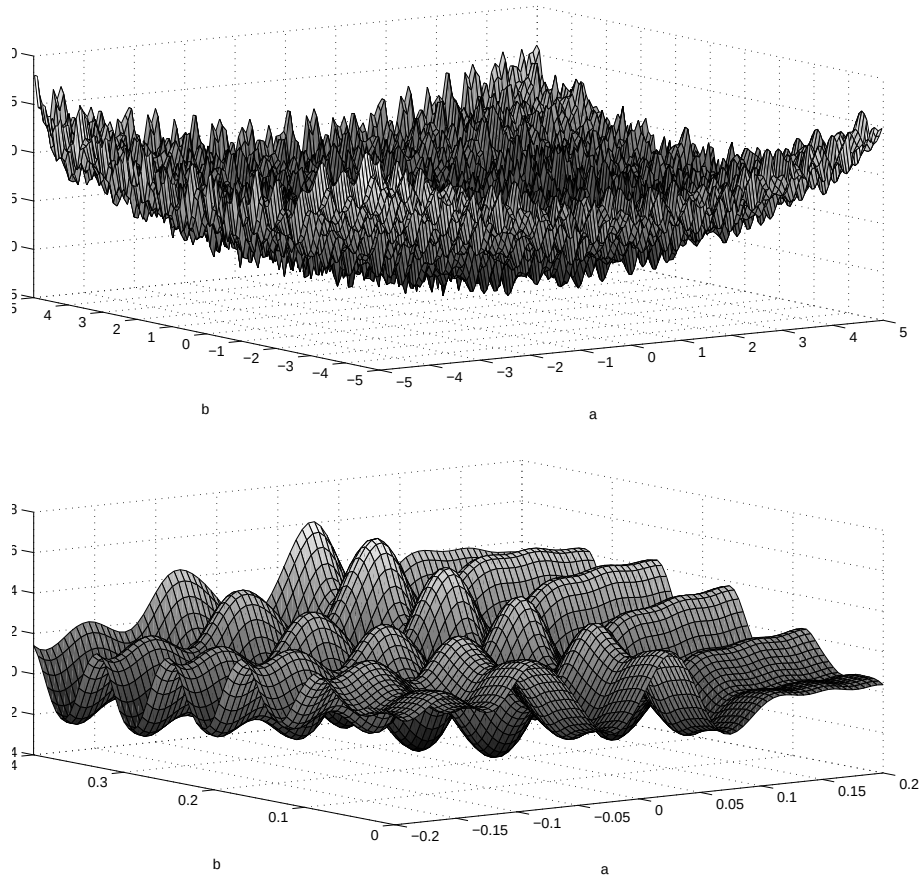

Figure 3: Function (5) with bounds [-5,5]×[-5,5], and $[-0.2, 0.2] \times [0.0, 0.4]$

Table 5: Comparison of 3 methods on 2-dimensional function (5) from [18]

	# Iterations	f_{best}	% error	Comp. efforts	Total Comp. efforts
Glob-VNS	300	-3.3069	0%	7303	26133
	200	-3.2970	0.30%	5907	17466
	100	-3.2286	2.37%	3994	8750
	50	-3.1364	5.16%	3158	4374
	30	-2.9789	9.92%	2356	2630
Gauss-VNS	400	-3.3069	0%	11585	37918
	300	-3.2970	0.30%	9149	27391
	200	-3.2871	0.60%	8259	17199
	100	-3.1539	4.63%	4265	7953
	50	-3.0121	8.92%	2801	3930
	30	-2.9059	12.13%	1648	2364
MADS-VNS C		-3.009	9.01%		5182
(VNS-3) E		-3.055	7.62%		6146
F		-2.837	14.21%		2809
E+F		-2.778	15.99%		3171

It is shown that the results obtained by our `Gauss-VNS` are comparable with four recent VNS-based heuristics from the literature. In comparing these methods it should be taken into account that `GLOB-VNS` is a state-of-the-art method for solving unconstrained NLPs, and also the fact that the comparison is performed in favor of `GLOB-VNS`, since no efforts were made to tune the parameters for `Gauss-VNS`, and instead the tuned parameters of `GLOB-VNS` were used.

Future research may contain extension to the constrained case, as well as the design of procedures for automatic estimation of the range of the parameter σ^2 during the execution of the code.

References

[1] M. Gendreau and J. Potvin. *Handbook of Metaheuristics*. Springer, 2010.

[2] M. Dražić, V. Kovačević-Vujčić, M. Čangalović, and N. Mladenović. *Global optimization: from theory to implementation*, chapter GLOB-A new VNS-based Software for Global Optimization, pages 135–154. Springer, 2006.

[3] L. Liberti and M. Dražić. Variable neighbourhood search for the global optimization of constrained nlps. In *Proceedings of GO Workshop, Almeria, Spain, 2005*, pages 1–5, 2005.

[4] N. Mladenović, M. Dražić, V. Kovačević-Vujčić, and M. Čangalović. General variable neighborhood search for the continuous optimization. *European Journal of Operational Research*, 191(3):753–770, 2008.

[5] N. Mladenović, J. Petrović, V. Kovačević-Vujčić, and M. Čangalović. Solving spread spectrum radar polyphase code design problem by tabu search and variable neighborhood search. *European Journal of Operational Research*, 151:389–399, 2003.

[6] M.D. Toksari and E. Güner. Solving the unconstrained optimization problem by a variable neighborhood search. *Journal of Mathematical Analysis and Applications*, 328(2):1178–1187, 2007.

[7] M. Bierlaire, M. Thémans, and N. Zufferey. A heuristic for nonlinear global optimization. *INFORMS Journal on Computing*, 22(1):59–70, 2010.

[8] J.H. Ahrens and U. Dieter. Efficient table-free sampling methods for the exponential, cauchy, and normal distributions. *Communications of the ACM*, 31(11):1330–1337, 1988.

[9] G.S. Fishman. *Monte Carlo: concepts, algorithms, and applications*. Springer, 1996.

[10] P.E. Gill, W. Murray, and M.A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

[11] Q.H. Zhao, D. Urošević, N. Mladenović, and P. Hansen. A restarted and modified simplex search for unconstrained optimization. *Computers & Operations Research*, 36(12):3263–3271, 2009.

[12] C. Audet, V. Béchard, and S.L. Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41(2):299–318, 2008.

[13] A.R. Hedar and M. Fukushima. Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. *Optimization Methods and Software*, 17(5):891–912, 2002.

[14] C. Lavor and N. Maculan. A function to test methods applied to global minimization of potential energy of molecules. *Numerical algorithms*, 35(2):287–300, 2004.

[15] M. Dražić, C. Lavor, N. Maculan, and N. Mladenović. A continuous variable neighborhood search heuristic for finding the three-dimensional structure of a molecule. *European Journal of Operational Research*, 185(3):1265–1273, 2008.

[16] `http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page295.htm`.

[17] `http://tracer.lcc.uma.es/problems/ackley/ackley.html`.

[18] L.N. Trefethen. A hundred-dollar, hundred-digit challenge. *SIAM news*, 35(1):01–02, 2002.