Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Gaussian variable neighborhood search for continuous optimization

Emilio Carrizosa [a], Milan Dražić [b], Zorica Dražić [b], Nenad Mladenović [c,*]

[a] Faculdad de Matemáticas, Universidad de Sevilla, Spain
[b] Faculty of Mathematics, University of Belgrade, Serbia
[c] School of Mathematics, Brunel University-West London, UK

ARTICLE INFO

Available online 20 November 2011

Keywords:
Global optimization
Nonlinear programming
Metaheuristics
Variable neighborhood search
Gaussian distribution

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.

1. Introduction

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

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

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 modeling, 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 [2–7], which in its most popular version takes the following form:

Algorithm 1. `Algorithm VNS`.

```
    /* Initialization */
01  Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1,\dots,k_{\max}$
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. $x^*$ is an approximate solution of the problem.
```

The idea of using several geometric neighborhood structures and random distributions in the shaking step led to the `Glob-VNS` variant of VNS [2,4], which turned out to be noticeably more

* Corresponding author. Tel.: +44 018 952 66151.
E-mail addresses: ecarrizosa@us.es (E. Carrizosa), mdrazic@sezampro.rs (M. Dražić), lolaz@sezampro.rs (Z. Dražić), Nenad.Mladenovic@brunel.ac.uk (N. Mladenović).

0305-0548/$ - see front matter Crown Copyright © 2011 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.11.003

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 2. Algorithm Glob-VNS.

```
   /* Initialization */
01 Select the pairs $(\mathcal{G}_l,\mathcal{P}_l)$, $l=1,\ldots,m$ of geometry
   structures and distribution types of the 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 $\rho_k$
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. $x^*$ 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

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

or

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

Metric $\rho(\cdot)$ is usually an ℓ_p distance, $1 \le p \le \infty$ [2–5], 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 [2,3] 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. For instance, to generate random points uniformly distributed in the unit sphere B of the Euclidian norm, different algorithms can be used. For example, 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 the 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 the 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

$$\text{global} \min_{x \in S} f(x), \tag{3}$$

with $S = \{x \in \mathbb{R}^n | a_i \le x_i \le b_i,\ i = 1,2,\ldots,n\}$. 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 $\mathcal{N}_k(x), 1 \le k \le k_{\max}$ 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 an 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 $\{\mathcal{N}_k(x)\}_{1 \le k \le k_{\max}}$ of neighborhoods of point x by a class 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 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

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

Algorithm 3. Algorithm Gauss-VNS.

```
   /* Initialization */
01 Select the set of covariance matrices $\Sigma_k$, $k = 1,\ldots,k_{\max}$
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 $\Sigma_k$
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. $x^*$ 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

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

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

The main purpose of this section is to compare our new Gaussian VNS with the previous VNS based heuristics for solving continuous global optimization. Therefore, we first compare it with those successful VNS heuristics that have recently appeared in the literature. Then we perform comparison with recent metaheuristics based global minimizers.

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 the 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 is 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 this fact, 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.

3.1. Comparison of VNS based methods

Recently many metaheuristic methods for solving continuous global optimization problems 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, they use 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 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. 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. [2];
- VNS-1—Bielaire et al. [7];
- VNS-2—Toksari and Güner [6] and
- VNS-3—Audet et al. [11].

Comparison on standard test instances: Experiments were performed on a set of standard test functions from the literature [7,12]. 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 the computational effort in 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 [7,6]

Table 1
Standard test functions.

Function		n	Computer effort		Local minim.	Computer effort		% Deviation
			VNS-1	VNS-2		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	**11,934**	–	FR	38621	37901	1.90
Rosenbrock	R100	100	**30,165**	–	FR	147,274	122,446	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	17,932	–	FR	**4304**	5302	−17.28
Average								22.17

respectively. The next columns contain information about the local minimizer used in `Glob-VNS` and `Gauss-VNS` and the average computational efforts for `Glob-VNS` and `Gauss-VNS` obtained in their 10 restarts. Finally, the last column, marked **% deviation**, contains the ratio of the two previous calculated values, namely

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

The smallest computational effort among four methods is boldfaced.

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 the 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 standard test instances. Since the main goal of our numerical test was to compare VNS-based heuristics, we fixed in advance most 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, and ACn.

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

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

$$-5.12\le x_i\le 5.12,\quad i=1,\dots,n,\ f_{\min}=0.$$

Molecular potential energy function (MPEn) function [13,14]:

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

$$0\le x_i\le 5,\quad i=1,\dots,n,\ f_{\min}=-0.0411183034\cdot n$$

and Ackley (ACn) function [15,16] (Fig. 2):

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

$$-15\le x_i\le 30,\quad i=1,\dots,n,\ f_{\min}=0.$$

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 2–4, 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

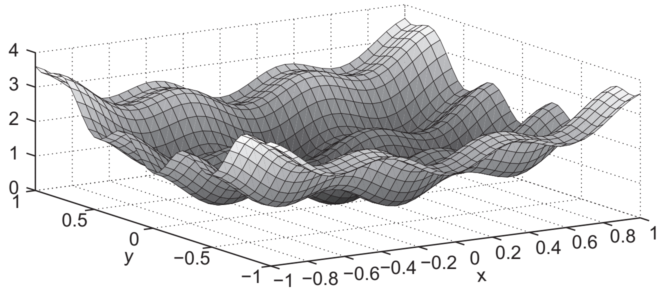

Fig. 1. Rastrigin test function RA2.

Table 2
Rastrigin function.

Function	n	Local minim.	$k_{\max}$	Computer effort		% Deviation
				`Glob-VNS`	`Gauss-VNS`	
RA10	10	SD	15	**52,471**	85,589	−38.69
RA20	20	SD	15	**213,597**	287,075	−25.60
RA30	30	SD	15	**366,950**	599,635	−38.80
RA40	40	SD	15	**697,160**	1,115,923	−37.53
RA50	50	SD	15	**1,334,842**	1,504,701	−11.29
RA100	100	SD	15	**5,388,075**	6,248,753	−13.77
RA150	150	SD	15	**11,007,093**	13,678,014	−19.53
RA200	200	SD	15	**24,026,456**	31,639,001	−24.06
Average						−26.16

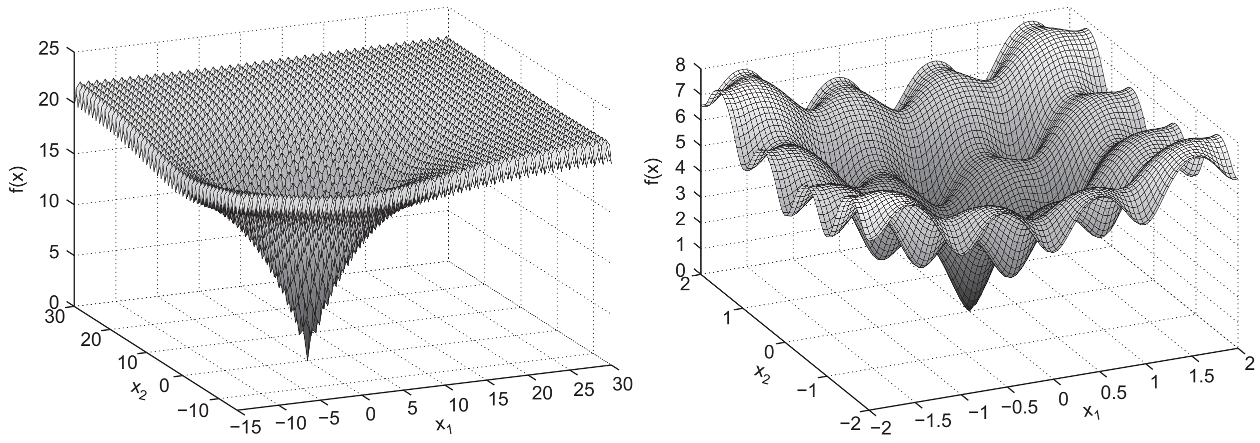

Fig. 2. Ackley function with bounds $[-15,30]\times[-15,30]$, and $[-2,2]\times[-2,2]$.

results in case of spherical-like function shapes. For Ackley function Gauss-VNS even improves its superiority for higher dimensions.

Table 3
Molecular potential energy function.

function	n	local minim.	k_{max}	Computer effort		% deviation(%)
				Glob-VNS	Gauss-VNS	
MPE10	10	SD	10	8102	**5015**	61.56
MPE20	20	SD	10	26,647	**21,172**	25.86
MPE30	30	SD	10	66,441	**49,162**	35.15
MPE40	40	SD	10	118,006	**109,468**	7.80
MPE50	50	SD	10	202,280	**143,309**	41.15
MPE100	100	SD	10	**830,343**	1,183,873	−29.86
MPE150	150	SD	10	**2,353,315**	2,802,372	−16.02
MPE200	200	SD	10	7,683,209	**5,859,705**	31.12
Average						19.59

Table 4
Ackley function.

Function	n	Local minim.	k_{max}	Computer effort		% Deviation
				Glob-VNS	Gauss-VNS	
AC10	10	SD	10	188,670	**50,149**	276.22
AC20	20	SD	10	433,194	**158,412**	173.46
AC30	30	SD	10	909,918	**304,825**	198.51
AC40	40	SD	10	1,577,138	**528,718**	198.30
AC50	50	SD	10	4,791,075	**1,143,721**	318.90
AC60	60	SD	10	7,820,247	**2,315,178**	237.78
AC70	70	SD	10	36,641,634	**4,255,533**	761.04
AC80	80	SD	10	212,944,367	**17,180,658**	1139.44
Average						412.96%

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) [11]. To do this, one test instance from [11] is used. This test instance, that was firstly suggested in [17] (problem 4), has two 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 Fig. 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 [11], 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 VNS-3 makes the algorithm stop in most cases before the global minimum is found. This explains why 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.

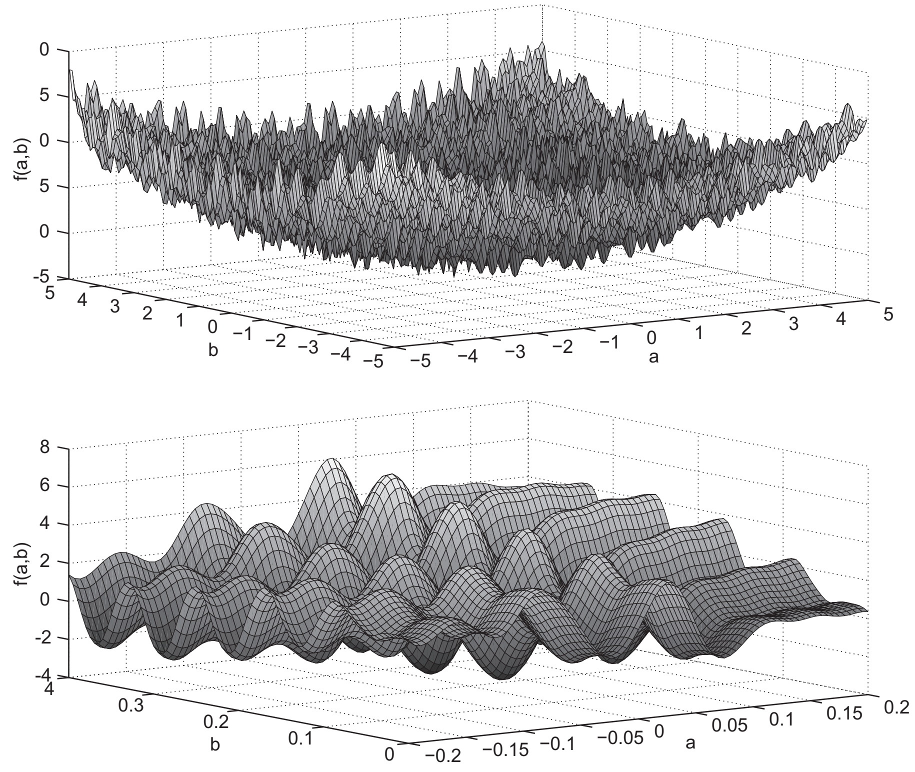

Fig. 3. Function (5) with bounds $[-5,5]\times[-5,5]$, and $[-0.2,0.2]\times[0.0,0.4]$.

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%.

3.2. Comparison with recent metaheuristics

In this subsection we compare our new Gauss-VNS with several metaheuristic based algorithms from the literature, such as Tabu search, Genetic algorithm, Ant Colony Optimization, Scatter Search, Swarm intelligence, and their hybrids. We made efforts to find the best representatives, among dozens, for each of these metaheuristics. The chosen methods used for comparison are listed in Table 6. So, in the list there are two tabu search based heuristics (CRTS and DTS), one genetic algorithm (GNM), one Scatter search (CSS), one Ant colony (HCIAC), one Swarm intelligence (SNM), one Restarted simplex search (RMNM) and our Gauss VNS.

It is not easy to perform correct comparison of these different approaches due to many reasons. Besides those previously mentioned, different methods could use different formulas to measure success or termination of the search. Last column of Table 6 indicates which stopping criterion is used among the following three:

$$|\tilde{f} - f_{min}| < 10^{-4}|f_{min}| + 10^{-6}, \tag{6}$$

$$|\tilde{f} - f_{min}| < 10^{-4}|f_{init}| + 10^{-6}, \tag{7}$$

$$|\tilde{f} - f_{min}| < 10^{-4}|f_{min}| + 10^{-4}. \tag{8}$$

The meaning of variables in the above formulas are as follows: $\tilde{f}$ is the objective function value obtained by the method; f_{min} denotes known global minimum value and f_{init} is empirical average of the objective function values calculated over 100 randomly selected feasible points. It is obvious that the criterion (6) is more strict than the two others, i.e., it is necessary to perform more function evaluations to satisfy it. Our Gauss VNS use it.

Finally, in Table 7, eight methods are compared based on average computational efforts they spent (out of 100 random initializations), before the known optimal solution f_{min} was reached. Some methods could not get f_{min} in each 100 runs. In such cases we denote in brackets the number of successful runs. For example, average number of function evaluations for the GNM method in solving SH_5 instance is 698, but measured only in 85 restarts, since 15 times the optimal solution was not reached.

It appears that our Gaus VNS heuristic outperforms other seven approaches from the literature on average. It needed minimum efforts in four out of nine standard test instances, and was the second in three cases.

4. Conclusions

In this note we present a new VNS-based heuristic for solving continuous unconstrained optimization problems. We call it Gauss-VNS since it uses Gauss distribution for generating random point x' from the kth neighborhood of the incumbent solution x ($x' \in \mathcal{N}_k(x)$). It in fact simplifies the previous VNS version proposed in [4], where four different distribution types

Table 5
Comparison of three methods on 2-dimensional function (5) from [17].

Heuristic methods	# Iterations	f_{best}	% Error	Comp. efforts	Total comp. efforts
Glob-VNS	300	−3.3069	0	7303	26,133
	200	−3.2970	0.30	5907	17,466
	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	11,585	37,918
	300	−3.2970	0.30	9149	27,391
	200	−3.2871	0.60	8259	17,199
	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
E (VNS-3)		−3.055	7.62		6146
F		−2.837	14.21		2809
E+F		−2.778	15.99		3171

Table 6
The global metaheuristics based minimizers.

Method	Abbrev.	Reference	Stopping
Genetic and Nelder–Mead	GNM	Chelouah and Siarry [18]	(7)
Continuous Reactive Tabu Search	CRTS[a]	Battiti and Tecchiolli [19]	(6)
Swarm with Nelder–Mead	SNM	Fan et al. [20]	(7)
Continuous scatter search	CSS	Herrera et al. [21]	(6)
Restarted modified Nelder–Mead	RMNM	Zhao et al. [22]	(6)
Ant Colony Optimization	ACO	Toksari [23]	(6)
Tabu search	TS	Al-Sultan, Al-Fawzan [24]	(8)
Gauss Variable neighborhood search	Gauss-VNS	This paper	(6)

[a] The best results of variants CRTSmin and CRTSave are chosen.

Table 7
Computational efforts (CE) of eight metaheuristic based minimizers. In bold and italic are marked the smallest and the second smallest CE for each instance, respectively (only methods that solved 100% instances are considered).

Test function	GNM[a]	CRTS[a]	SNM[a]	CSS[a]	RMNM[a]	ACO[b]	TS[b]	Gauss VNS[a]
BR	295	**38**	230	65	*60*	324	398	112
GP	259	171	304	**108**	69 (80)	264	281	*116*
HT_3	492	513	436	–	**67**	528	578	*223*
HT_6	930	*750*	–	–	398 (50)	1344	2125	**448**
SB	**345**	–	753	762	275 (40)	–	–	*591*
RO_2	459	–	440	292	*224*	924	1632	**125**
RO_{10}	14,563 (83)	–	*3303*	5847 (75)	5946 (95)	8726	11,448	**2561**
SH_5	698 (85)	*664*	850	1197	912 (90)	**648**	753	1042
SH_{10}	635 (85)	*693*	–	–	318 (75)	1044	1203	**399**

[a] The average CE of 100 runs.
[b] The average CE of four runs.

were considered: (i) uniform in ℓ_1 norm; (ii) uniform in ℓ_∞ norm; (iii) hypergeometric in ℓ_1 and (iv) special constructed distribution in ℓ_∞ norm. Instead of using all of them, we try just Gauss normal distribution, with different values of parameter σ_k^2. In that way, $\mathcal{N}_k(x)$ becomes $\mathcal{P}_k(x)$. Beside improving user-friendliness by simplifying 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 ℓ_2 norm (previously only ℓ_1 and ℓ_∞ norms were used).

It is shown that the results obtained by our `Gauss-VNS` are comparable with recent four VNS-based heuristics from the literature. Future research may contain extension to the constrained case, as well as automatic estimation of range of parameter σ^2 during the execution of the code.

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) and also while E. Carrizosa visited the Centre de Recerca Matematica at Barcelona. Last three authors are also partially supported by Serbian Ministry of Science, project number 174010.

References

[1] Gendreau M, Potvin J. Handbook of metaheuristics. Springer; 2010.
[2] Dražić M, Kovačević-Vujčić V, Čangalović M, Mladenović N. Global optimization: from theory to implementation. Chapter GLOB—a new VNS-based software for global optimization, Springer; 2006, p. 135–154.
[3] Liberti L, Dražić M. Variable neighbourhood search for the global optimization of constrained nlps. In: Proceedings of GO workshop, Almeria, Spain; 2005, p. 1–5.
[4] Mladenović N, Dražić M, Kovačević-Vujčić V, Čangalović M. General variable neighborhood search for the continuous optimization. European Journal of Operational Research 2008;191(3):753–70.
[5] 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 2003;151: 389–99.
[6] Toksari MD, Güner E. Solving the unconstrained optimization problem by a variable neighborhood search. Journal of Mathematical Analysis and Applications 2007;328(2):1178–87.
[7] Bierlaire M, Thémans M, Zufferey N. A heuristic for nonlinear global optimization. INFORMS Journal on Computing 2010;22(1):59–70.
[8] Ahrens JH, Dieter U. Efficient table-free sampling methods for the exponential, cauchy, and normal distributions. Communications of the ACM 1988;31(11):1330–7.
[9] Fishman GS. Monte Carlo: concepts, algorithms, and applications. Springer; 1996.
[10] Gill PE, Murray W, Saunders MA. Snopt: an sqp algorithm for large-scale constrained optimization. SIAM Journal on Optimization 2002;12(4): 979–1006.
[11] Audet C, Béchard V, Digabel SL. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. Journal of Global Optimization 2008;41(2):299–318.
[12] Hedar AR, Fukushima M. Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. Optimization Methods and Software 2002;17(5):891–912.
[13] Lavor C, Maculan N. A function to test methods applied to global minimization of potential energy of molecules. Numerical algorithms 2004;35(2): 287–300.
[14] 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 2008;185(3): 1265–73.
[15] 〈http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page295.htm〉.
[16] 〈http://tracer.lcc.uma.es/problems/ackley/ackley.html〉.
[17] Trefethen LN. A hundred-dollar, hundred-digit challenge. SIAM News 2002;35(1):01–2.
[18] Chelouah R, Siarry P. Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. European Journal of Operational Research 2003;148(2):335–48.
[19] Battiti R, Tecchiolli G. The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization. Annals of Operations Research 1996;63(2):151–88.
[20] Fan SKS, Liang YC, Zahara E. A genetic algorithm and a particle swarm optimizer hybridized with Nelder–Mead simplex search. Computers & Industrial Engineering 2006;50(4):401–25.
[21] Herrera F, Lozano M, Molina D. Continuous scatter search: an analysis of the integration of some combination methods and improvement strategies. European Journal of Operational Research 2006;169(2):450–76.
[22] Zhao QH, Urošević D, Mladenović N, Hansen P. A restarted and modified simplex search for unconstrained optimization. Computers & Operations Research 2009;36(12):3263–71.
[23] Toksari MD. Minimizing the multimodal functions with ant colony optimization approach. Expert Systems with Applications 2009;36(3):6030–5.
[24] Al-Sultan KS, Al-Fawzan MA. A tabu search Hooke and Jeeves algorithm for unconstrained optimization. European Journal of Operational Research 1997;103(1):198–208.