Les Cahiers du GERAD ISSN: 0711–2440

Continuous variable neighborhood search (C-VNS) for solving systems of nonlinear equations

J. Pei, Z. Dražić, M. Dražić,
N. Mladenović, P. M. Pardalos

G–2018–52

July 2018

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Continuous variable neighborhood search (C-VNS) for solving systems of nonlinear equations

Jun Pei [a]
Zorica Dražić [b]
Milan Dražić [b]
Nenad Mladenović [c]
Panos M. Pardalos [d]

[a] *School of Management, Hefei University of Technology; Key Laboratory of Process Optimization and Intelligent Decision-making of Ministry of Education, Hefei, China*

[b] *Faculty of Mathematics, University of Belgrade, Serbia*

[c] *GERAD & Mathematical Institute, Serbian Academy of Sciences and Arts, Serbia*

[d] *Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA*

feiyijun.ufl@gmail.com
lolaz@sezampro.rs
mdrazic@sezampro.rs
nenad@mi.sanu.ac.rs
pardalos@ufl.edu

July 2018
Les Cahiers du GERAD
G–2018–52

Abstract: In this paper we propose C-VNS (Continuous variable neighborhood search) method for finding all solutions to a nonlinear system of equations (NSE). We transform the NSE problem into an equivalent optimization problem, and we use a new objective function that allows us to find all zeros. Instead of the usual sum-of-squares objective function, our objective function is presented as the sum of absolute values. Theoretical investigation confirms that our objective function provides more accurate solutions, irrespective of what optimization method is used. We sacrifice the smoothness property to increase precision. Computational analysis on standard test instances shows that our C-VNS based method is more precise and much faster than the two recent methods from the literature we compared with. Moreover, similar conclusions are derived after comparing our C-VNS based heuristic with many other methods from the literature.

Keywords: System of nonlinear equations, continuous Optimization, variable neighborhood search, direct search methods

1 Introduction

Problem formulation. Consider the problem of finding solutions to the nonlinear system of equations (NSE)

$$\begin{aligned} f_1(x) &= 0 \\ f_2(x) &= 0 \\ &\cdots \\ f_n(x) &= 0 \end{aligned} \tag{1}$$

where $x \in S = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n] \subset R^n$ and $f_1, f_2, \ldots f_n$ are nonlinear real valued continuous functions on S. The number of solutions to a NSE problem can be finite, infinite, or there can be no solution.

To solve the NSE problem, a number of transformation techniques have been presented in the literature. They can be classified into three categories: (i) single-objective-based optimization methods; (ii) constrained-optimization-based techniques; and (iii) multi-objective-based optimization techniques (Gong et al. 2017).

In the first category, which we use in this paper, NSE is usually transformed into a single-objective optimization problem by introducing the function:

$$F_0(x) = \sum_{i=1}^{n} |f_i(x)|^p \,, \qquad p > 0 \,. \tag{2}$$

By definition, it holds that $F_0(x) \geq 0$. Thus, $x^* \in N(x)$ is the solution to the system (1) if and only if $F_0(x^*) = 0$, i.e., x^* is a global minimizer of $F_0(x)$ in S with $F_0(x^*) = 0$. Most often, the used values of p are 1 and 2, i.e.,

$$\min_x F_0(x) = \sum_{i=1}^{n} |f_i(x)| \tag{3}$$

or

$$\min_x F_0(x) = \sum_{i=1}^{n} f_i^2(x) \tag{4}$$

The motivation for finding all solutions is problem specific, sometimes it is important and sometimes irrelevant. In many cases, decision makers need to have a set of alternative solutions. For example, in some data analysis problems, we need solutions where most decision variables are zero. Moreover, from the mathematical point of view, finding the solution to the NSE, means finding all vectors that satisfy all equalities. In this paper we are trying to find all solutions. To make it easier, it is possible to add a penalty function to $F_0(x)$ (see for example (Hirsch et al. 2009)), and the aim is not to allow finding the solutions already found in previous iterations, i.e., $F_0(x)$ is modified in each iteration.

Complexity. From the complexity point of view, the problem of solving NSE is NP-hard, even when the equations are multi-variable polynomials. For example, the n-dimensional 0-1 knapsack problem, which is known to be NP-hard, can be easily formulated as a system of $n+1$ equations: 0-1 condition for each variable x_j can be presented as $x_j^2 - x_j = 0$. Therefore, many existing approaches for solving nonlinear systems of equations rely on powerful heuristics. These heuristics may find multiple solutions but may fail to recognize when the system is infeasible.

Some applications. NSE is a type of challenging and non-trivial problem in many real-world applications, such as engineered materials, chemical processes, electronic circuits, petroleum geological prospecting, and computational mechanics (Mo et al. 2009, Pourjafari and Mojallali 2012). Applications in engineering include describing the space of possible configurations of robot structure and solving the forward kinematic problem of robot kinematics application (Cox et al. 2007, Hirsch et al. 2007, Lafmejani et al. 2015), settling the double retrograde vaporization problem in chemical engineering (Henderson et al. 2010, Sacco and Henderson 2011), and identifying geolocation area in Global Position System (Paláncz et al. 2008, 2010, Romero and Mason 2015).

Solution techniques. For solving NSE as an optimization problem, we know the best objective function value in advance. Indeed, the solution x^* that satisfies $F(x^*) = 0$ is optimum, and there is no need to prove this in a complete enumeration scheme. This fact is the main reason to have more than twenty heuristic approaches proposed in the literature, which additionally clearly indicates that exact solution methods cannot be the right choice for solving the NSE problem.

There are many techniques for finding just one solution to the given system, such as Newton and quasi-Newton methods (Luo et al. 2008, Buhmiler et al. 2010, Al-Baali et al. 2014), Newton-like method (González-Lima and de Oca 2009, Argyros 2009, Sharma and Guha 2016, Krejić et al. 2007), Newton-type methods (Darvishi and Barati 2007, Fang et al. 2009, Fischer et al. 2016), and iterative methods (Awawdeh 2010, Cordero et al. 2012, Wang and Meng 2013, Shoja et al. 2017). To get the insight into a presentation of global optimization approaches for enclosing all solutions to constrained systems of nonlinear equations, the reader is directed to Soares (2013) and Henderson et al. (2017).

In Hirsch et al. (2009), Continuous Greedy Randomized Adaptive Search Procedures (C-GRASP) were utilized to obtain all roots of NSE. This algorithm was a multi-start procedure where the initial solution for local improvement was constructed in a greedy randomized fashion. Based on a multiplicative type penalty merit function, Ramadas et al. (2015) addressed the problem of computing multiple roots of NSE by designing a repulsion algorithm that invokes the Nelder-Mead local search algorithm and a penalty-type merit function. Also, they improved this algorithm by combining the repulsion algorithm with harmony search (HS) (Ramadas et al. 2014). In addition, particle swarm optimization (PSO) algorithms have shown good performance when solving such problems. Mo et al. (2009) transformed the NSE problem into a high-dimensional optimization problem, and retrofitted PSO with chaos search, Newton-type methods, and Conjugate Direction method (CD). Jaberipour et al. (2011) proposed PSO to cope with NSE, where each particle was updated to avoid getting trapped into local minima, while Turgut et al. (2014) applied different chaotic maps to enhance the effectiveness and robustness of PSO.

In recent years, much attention has been drawn to the application of multi-objective evolutionary algorithms (MOEA). Song et al. (2015) transformed NSE into a bi-objective optimization problem and handled it with MOEA. Qin et al. (2015) switched NSE into a multi-objective optimization problem (MOP) and used MOEA to solve it. Gong et al. (2017) converted NSE into a weighted bi-objective optimization problem and suggested an adaptive multi-objective differential evolution (DE). In addition, Genetic algorithms (GA) (Mhetre 2012, Pourrajabian et al. 2013, Silva et al. 2014), artificial bee colony algorithms (ABC) (Jia and He 2012), cuckoo optimization algorithms (COA) (Abdollahi et al. 2016), Invasive weed optimization (IWO) (Pourjafari and Mojallali 2012, Zhou et al. 2013), Imperialist competitive algorithms (ICA) (Abdollahi et al. 2013), and fuzzy adaptive simulated annealing algorithm (fuzzy-ASA) (Oliveira and Petraglia 2013) have also been utilized to deal with NSE. Tremendous effort has been made to solve NSE. However, most of the existing methods cannot make a good balance between convergence speed and solution quality. Hence, more efficient algorithms should be designed to solve the problem.

Contribution. The following new results are presented in this paper: (i) For the first time the C-VNS based heuristic is applied to solve NSE; (ii) It is theoretically and empirically shown how the choice of reformulation to the equivalent optimization problem, i.e., the choice of an objective function, influences the precision of the final solutions; (iii) Comparative analysis shows that our approach provides more accurate solutions in much less time than the recent heuristics based on Continuous Greedy randomized adaptive search procedure (C-GRASP) (Hirsch et al. 2009) and harmony search (HS) (Ramadas and Fernandes 2013). Moreover, similar conclusions regarding effectiveness and efficiency are derived after comparison with more than ten methods from the literature.

Outline of the paper. In the next section, we discuss the use of different objective functions and penalties to get all solutions of the system (1) if such solutions are finite. In Section 3, we give the rules of our C-VNS based heuristics for solving NSE. In Section 4, computational analysis is presented, while in Section 5, we conclude the paper and propose possible future research directions.

2 Accuracy of the approximate solution

In this section, we first discuss accuracy in the vicinity of one solution of the objective function (2), and then the accuracy of the objective function is investigated for finding all solutions of NSE. Several auxiliary obvious estimates, important for the error analysis, are summarized in the next lemmas. The results from this section could be used for any NSE solution method.

2.1 Accuracy in finding one solution of NSE

In the following, a few results on accuracy of the approximate solution of NSE are presented. Let x^* denote an exact solution of NSE (1), let $x = x^* + d$ be an approximate solution, and $N(x^*)$ be a neighborhood of x^*. The Euclidean norm $\|d\|$ represents the error of an approximate solution. The solution x^* is called *single* if $\partial f_i(x^*)/\partial x_j \neq 0$ for some $1 \leq i, j \leq n$. Otherwise, if $\partial f_i(x^*)/\partial x_j = 0$ for all $1 \leq i, j \leq n$, then the solution is called *double* or *multiple* when all derivatives of higher order vanish in x^*, too.

The objective function to be minimized is given by

$$F_0(x) = \sum_{i=1}^{n} |f_i(x)|^p \,, \qquad p > 0 \,.$$

Lemma 1 *Let functions $f_i(x)$, $1 \leq i \leq n$, be Lipschitz continuous in the neighborhood $N(x^*)$ with Lipschitz constant L. Then*

$$||d|| \geq \gamma \cdot (F_0(x))^{1/p} \,, \tag{5}$$

where constant γ depends on n, p, and L.

Proof. As a consequence of the Lipschitz continuity of $f_i(x)$

$$|f_i(x)| = |f_i(x^* + d) - f_i(x^*)| \leq L||d|| \,,$$

and

$$F_0(x) = \sum_{i=1}^{n} |f_i(x)|^p \leq nL^p||d||^p \,,$$

which leads to (5) with $\gamma = (n^{1/p}L)^{-1}$. □

Lemma 2 *Let functions $f_i(x) \in C^1(N(x^*))$, $1 \leq i \leq n$, be smooth functions in $N(x^*)$. Then, the inequality (5) holds.*

Proof. Functions $f_i(x) \in C^1(N(x^*))$ have bounded first derivatives, and consequently, they are Lipschitz continuous in $N(x^*)$ and Lemma 1 can be applied. □

Lemma 3 *Let $p = 1$, functions $f_i(x) \in C^2(N(x^*))$, $1 \leq i \leq n$, and $\partial f_i(x^*)/\partial x_j = 0$ for all $1 \leq i, j \leq n$. Then,*

$$\|d\| \geq \gamma \cdot (F_0(x))^{1/2} \,.$$

Proof. From the Taylor expansion

$$f_i(x) = f_i(x^* + d) = f(x^*) + \nabla f_i \cdot d + \frac{1}{2}\, d^T \cdot \nabla^2 f_i(\bar{x}) \cdot d \,,$$

taking into account that $f_i(x^*) = 0$ and $\partial f_i(x^*)/\partial x_j = 0$, and since the second order derivatives are bounded in $N(x^*)$,

$$F_0(x) = \sum_{i=1}^{n} |f_i(x)| \leq \; n \cdot \gamma \cdot \|d\|^2 \,,$$

which proves the claim. □

Lemma 4 *Let $p = 2$, functions $f_i(x) \in C^2(N(x^*))$, $1 \le i \le n$. Then*

$$\|d\| \ge \gamma \cdot (F_0(x))^{1/2} .$$

Proof. The original system is equivalent to the system $g_i(x) = 0$, $1 \le i \le n$, where $g_i(x) = (f_i(x))^2$. Since $\partial g_i(x^*)/\partial x_j = 2 f_i(x^*) \partial f_i(x^*)/\partial x_j = 0$, the result follows from Lemma 3, applied to the second system. □

Lemma 5 *Let $p = 2$, functions $f_i(x) \in C^4(N(x^*))$, $1 \le i \le n$, and $\partial f_i(x^*)/\partial x_j = 0$ for all $1 \le i, j \le n$. Then*

$$\|d\| \ge \gamma \cdot (F_0(x))^{1/4} .$$

Proof. For the function $g_i(x) = (f_i(x))^2$, partial derivatives up to the third order are

$$\begin{aligned}
\frac{\partial g_i}{\partial x_j} &= 2\, f_i(x) \frac{\partial f_i}{\partial x_j} , \\
\frac{\partial^2 g_i}{\partial x_j \partial x_k} &= 2\, \frac{\partial f_i}{\partial x_k} \frac{\partial f_i}{\partial x_j} + 2\, f_i(x) \frac{\partial^2 f_i}{\partial x_j \partial x_k} , \\
\frac{\partial^3 g_i}{\partial x_j \partial x_k \partial x_m} &= 2\, \frac{\partial^2 f_i}{\partial x_k \partial x_m} \frac{\partial f_i}{\partial x_j} + 2\, \frac{\partial f_i}{\partial x_k} \frac{\partial^2 f_i}{\partial x_j \partial x_m} + 2\, \frac{\partial f_i}{\partial x_m} \frac{\partial^2 f_i}{\partial x_j \partial x_k} + 2\, f_i(x) \frac{\partial^3 f_i}{\partial x_j \partial x_k \partial x_m} .
\end{aligned}$$

Since $f_i(x^*) = 0$ and $\partial f_i(x^*)/\partial x_j = 0$ for all $1 \le i, j \le n$, all derivatives of $g_i(x)$ up to the third order vanish in x^*. As the fourth derivatives of $g_i(x)$ are bounded in $N(x^*)$, it follows from the Taylor expansion that

$$|g_i(x)| \le \gamma_1 \|d\|^4.$$

Finally,

$$F_0(x) = \sum_{i=1}^{n} (f_i(x))^2 = \sum_{i=1}^{n} |g_i(x)| \le n \cdot \gamma_1 \|d\|^4.$$

The result of this lemma follows by setting $\gamma = (n \cdot \gamma_1)^{-\frac{1}{4}}$. □

Lemmas 1–5 show how the value of the objective function is related to the accuracy of the approximate solution. The most obvious stopping criterion in optimization algorithms is $F_0(x) < \delta$, for some sufficiently small $\delta > 0$. It turns out that the error of the solution $\|d\|$ is dependent not only on δ, but also on the value of p, and the multiplicity of the solution. For the same value of δ, the error $\|d\|$ is significantly greater for $p = 2$ than that for $p = 1$. Also, for the same δ and p, the error is significantly greater for double and multiple roots than for the single root. Based on these relationships, one can choose the appropriate value of δ for the desired approximate solution error.

2.2 Accuracy in finding all roots

Application of any heuristic method for solving the continuous global optimization problem (2) can produce an approximate solution to the system (1) if the objective function is sufficiently small. If there are many solutions, this approach is not adequate since there is no guarantee for finding all solutions. To overcome this problem, we modify the objective function in the neighborhood of the found solutions. Suppose that we have found solutions $x^1, x^2, \ldots, x^k$, $k \ge 0$. For finding the next solution, if there is one, we minimize the function:

$$F(x) = F_0(x) + F_k(x) = \sum_{i=1}^{n} |f_i(x)|^p + \sum_{j=1}^{k} a\, \varphi\left(\frac{\|x - x^j\|}{\rho} \right) , \tag{6}$$

where a and ρ are some positive values, and $\varphi(x)$ is a continuous real function, positive for $|x| < 1$ and zero elsewhere. Throughout the paper, it is assumed that $\varphi(x) = 0$ for $|x| \ge 1$.

If the value of the global minimum is sufficiently small, i.e., $F(x^*) < \delta$, then another approximate solution $x_{k+1} = x^*$ is found; the value of k is increased by 1, and the global optimization search for more solutions

is repeated. Otherwise, if $F(x^*) \geq \delta$, then the algorithm stops because there are no more approximate solutions.

The function $\varphi(x)$ should be calculated fast and has to be smooth enough for the local optimizer used in C-VNS, or for any other local search based method. Some of the examples are given below:

(i) $\varphi(x) = 1 - |x| \quad$ for $|x| < 1, \quad 0$ elsewhere;
(ii) $\varphi(x) = (1 - x^2)^2 \quad$ for $|x| < 1, \quad 0$ elsewhere;
(iii) $\varphi(x) = (1 + \cos(\pi x))/2 \quad$ for $|x| < 1, \quad 0$ elsewhere;
(iv) $\varphi(x) = \exp(-\frac{x^2}{1-x^2}) \quad$ for $|x| < 1, \quad 0$ elsewhere.

In case (i), the function $\varphi(x)$ is continuous, but not smooth; in cases (ii) and (iii) it has continuous first derivative but discontinuous second derivative; and in case (iv), it is infinitely differentiable. The first two functions can be calculated faster than the last two. In the experimental results given in the next section, we use those two functions.

Note that the function (6) is not convex, since any penalty function $\varphi(x)$ is obviously non-convex. It could be concave, but as it is not important for our global optimization method, we will illustrate it later. Moreover, the non-convexity follows directly from the fact that there could be many solutions.

Values of a and ρ influence the behavior of the local optimizer near the found solutions. The term $a\varphi(||x - x^j||/\rho)$ acts like a repealer in ρ neighborhood of x^j. For larger values of a and ρ, this repealer is stronger and the local minimization is directed away from the basin of $F(x)$ near x^j, so it is more likely to find another solution. The drawback of large values of ρ is that we cannot find all of the solutions close to each other. In this case, decreasing ρ and a can solve the problem, but a local minimizer is more likely to be stuck near the found solutions. For larger values of p, or in the case of double or multiple solutions, or for smaller values of ρ, values of $F(x)$ can be smaller than a given tolerance for points outside the ρ-neighborhood of the solution, so false solutions may be detected. The following property from (6) holds.

Lemma 6 *If the system (1) has exactly K solutions and the value of ρ from (6) is smaller than the minimal distance between the solutions, applying global optimization repeatedly on the function (6) guarantees finding all the solutions.*

Proof. Proof. It is clear that $F(x) \geq 0, \quad F(x^j) > 0, \ j = 1, \ldots, k$ and $F(x^*) = 0$ for all solutions of (1) are at a distance at least ρ from all known solutions $x^1, x^2, \ldots, x^k$. The result holds from the fact that, by definition, $\varphi(x) = 0$ for $|x| \geq 1$. Indeed, $\varphi(\|x - x^j\|/\rho) = 0$, since $\|x - x^j\|/\rho > 1$, for all $j = 1, .., k$. □

Note that the values of a and ρ can vary in this iterative process of solving K times the minimization problem (6).

The three parameters, p, a, and ρ, and the choice of the function $\varphi(x)$ can influence the performance of the method for finding all solutions. Parameter p is usually set to 1 or 2.

Lemma 7 *The following properties are obvious: (i) In the case of smooth functions $f_i(x), \ i = 1, \ldots, n$, $F(x)$ will be smooth only for $p > 1$ and for smooth $\varphi(x)$; (ii) In the case of analytic $f_i(x), \ i = 1, \ldots, n$ and $p = 2$, function $F(x)$ is not analytic for $k \geq 1$ even for the infinitely smooth $\varphi(x)$.*

Hence, higher order methods may not perform as expected in the vicinity of found solutions. In the small neighborhood of the solution, for $||x - x^*|| \sim \varepsilon$, in the case of a single solution, we expect that $|f_i(x)| \sim \varepsilon$. However, it is not always the case.

Theorem 1 *In the case of a single solution, if all functions $f_i(x), \ i = 1, .., n$ are smooth and if $F_0(x) < \delta$ for sufficiently small real $\delta > 0$, then the error of the approximate solution of function (2) is proportional to $\varepsilon = \delta^{\frac{1}{p}}$.*

Proof. Since δ is sufficiently small, all terms in the second sum in (6) are equal to 0, so $F(x) = F_0(x)$ in some small neighborhood of the approximate solution. The result then holds directly from Lemma 2 and Lemma 3. □

For the case of a double solution, the following result holds.

Theorem 2 *In the case of a double solution, if all functions $f_i(x)$, $i = 1, .., n$ are smooth and if $F(x) < \delta$, the error of the approximate solution is proportional to $\varepsilon = \delta^{\frac{1}{2p}}$.*

Proof. The proof is analog to the proof of the previous theorem, except using Lemma 3 and Lemma 5 at the end. □

For the larger values of p, more stringent tolerances for global optimization are required for the same acceptable solution error. Usual choices are $p = 1$, where a non-differentiable optimizer should be used, and $p = 2$ where a gradient or higher order methods can be used if functions $f_i(x)$ are smooth enough. Note that the convergence speed of higher order methods is reduced in the case of double or multiple solutions.

3 Continuous variable neighborhood search for solving NSE

In this section we first give a general pseudo-code for C-VNS, and then its implementing way is illustrated for solving NSE. Finally, we discuss details of the new algorithm for finding all solutions, which is embedded into C-VNS. It uses theoretical results from the previous section.

3.1 Continuous VNS

The VNS meta-heuristic is well-established in the literature. It was primarily designed for solving hard discrete problems (Mladenović and Hansen 1997, Hansen and Mladenović 2001, Hansen et al. 2017), but it is also very effective in continuous global optimization problems (Kovačević-Vujčić et al. 2004, Mladenović et al. 2008, Dražić et al. 2008, Carrizosa et al. 2012). The basic idea is to define a set of neighborhood structures $\mathcal{N}_k,\ k = 1, \ldots, k_{\max}$ with the corresponding random number distributions that can be used in a systematic way to search the solution space. The C-VNS algorithm can be summarized as

```
          Algorithm C-VNS
          /* Initialization */
01        Select the set of neighborhood structures N_k, k = 1,...,k_max
          with the corresponding random 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 05
11                endif
12                Set k ← k + 1
13            end
14        end
15        Stop. Point x* is an approximate solution of the problem.
```

The usual neighborhoods for continuous optimization are defined by balls in ℓ_1, ℓ_2, or ℓ_∞ metrics with increasing set of predefined radii $r_1, \ldots, r_{k_{\max}}$. More precisely, the following parameter values should be defined in C-VNS: (i) a set of neighborhood structures, where each neighborhood is defined by the ordered pair (metric, radius); (ii) a distribution, or a way to generate a random point in the neighborhood defined by metric and radius; (iii) the total number of neighborhoods, k_{max}; (iv) a local search routine.

3.2 Less is more approach for solving NSE by C-VNS

In our C-VNS based heuristic for solving NSE, we transform the system into a single objective function (2). For finding all solutions, we use the objective function (6). C-VNS for solving NSE is integrated into the package GLOBC, which is a test platform for numerical experiments with C-VNS (Dražić et al. 2006). GLOBC implements various neighborhood structures and random distributions, and can use several local minimizers for smooth and non-smooth optimization (Mladenović et al. 2008).

In implementation of C-VNS, we follow the principles of the recent Less is more approach (LIMA) (Mladenović et al. 2016, Costa et al. 2017, Gonçalves et al. 2018). The basic idea of LIMA is to use the minimum number of ingredients in building a search method, such that it provides better results than results obtained by the currently best from the literature. Thus, in developing C-VNS for the NSE, we apply the minimalism principle in choosing some, among various options provided by C-VNS. As discussed earlier, the optimal value for solving NSE is equal to zero, and therefore, it is used as a stopping condition $F^* \leq tol$ for C-VNS, besides the time limit. Let

$$B_i = \{x \mid ||x - x^*||_{l_\infty} \leq r_i\}$$

denote an l_∞ ball centered at the current best point x^*, with radii r_i. Radii are automatically generated to be equally spaced so that $B_{k_{\max}}$ covers the entire solution space S. Note that S represents a hypercube. The neighborhoods $\mathcal{N}_k$ are taken as:

$$\mathcal{N}_1 = B_1 \cap S,\ \mathcal{N}_k = (B_k \setminus B_{k-1}) \cap S,\ k = 2, \ldots, k_{\max}.$$

C-VNS was set to use $k_{\max} = 5$. In all tests, uniform random distribution in $\mathcal{N}_k$ and, unless otherwise stated, the Nelder-Mead local optimizer were used. Therefore, in order to make our method robust and user-friendly, we keep the number of C-VNS parameters at minimum, i.e., the same values of parameters are used in all tests, without fine tuning: (i) ℓ_∞ norm with automatic generation of radii, (ii) uniform distribution; (iii) $k_{max} = 5$, and (iv) Nelder-Mead local search. Let us note again that we follow the LIMA to get the minimum number of ingredients in the search.

3.3 Finding all solutions

A new module, based on presented algorithm for finding multiple solutions, is developed in order to handle multiple solutions and dynamical change of penalty parameters in function (6). We now present its pseudo-code.

	Algorithm for finding multiple solutions of (1) by C-VNS
	/* *Initialization* */
01	Select p, ρ_0, $\rho_{\min}$, a_0, $q_\rho < 1$, $q_a < 1$, tol, K and function $\varphi(x)$
02	Set $k = 0$, $\rho \leftarrow \rho_0$, $a \leftarrow a_0$
	/* *Main loop* */
03	**while** $k < K$ and $\rho \geq \rho_{\min}$
04	Perform global optimization for function (6) by C-VNS resulting with x^* as the best approximation of optimal point
05	**if** $F(x_*) \leq tol$ **then**
06	Set $k \leftarrow k + 1$
07	Set $sol(k) = x^*$
08	**else**
09	Set $\rho \leftarrow q_\rho \cdot \rho$
10	Set $a \leftarrow q_a \cdot a$
11	**endif**
12	**endwhile**
13	**Stop**. Points $sol(i)$, $i = 1, \ldots, k$ are approximate solutions of (1).

C-VNS is used for finding the minimum of function (2). One of the stopping conditions for C-VNS is $F(x) \leq tol$. If this condition is true, a new approximate solution of NSE is found, the function $F(x)$ is modified, and a new C-VNS optimization starts in order to find the next solution. In case when $F(x) > tol$ is satisfied, no additional solution is found, and the parameter ρ is reduced to enable finding the solutions closer to the previously found solutions. When ρ becomes smaller than $\rho_{\min}$, or maximum number K of solutions is found, the algorithm stops.

Additional parameters are chosen as follows. The maximum number of solutions to search for was set to K =20, unless otherwise stated. For a given tolerance tol, C-VNS terminates with a new approximate system solution x, if $F(x) \leq tol$. Unless otherwise stated, $tol = 10^{-7}$ is used in all examples.

Obviously, fitting different neighborhood structures, different construction of radii, and different distributions for finding all solutions, for each particular problem, could improve the efficiency of our C-VNS for NSE. However, our main goal was to present the effective, efficient, robust, and user-friendly algorithm for finding all solutions.

4 Computational results

The effectiveness and efficiency of C-VNS in finding the multiple solutions of the system (1) are illustrated in this section. We first describe test problems in subsection 4.1, and the results are analyzed in subsection 4.2. Next, we compare our C-VNS with C-GRASP (Hirsch et al. 2009) and Harmony search (Ramadas and Fernandes 2013) in subsections 4.3 and 4.4. The reasons why we choose these two benchmark methods are: (i) they are both recent and based on some meta-heuristic principles, as our C-VNS; (ii) they both use many test instances available on the web; (iii) computational results provided included detailed analysis. Subsequently, the problems from Floudas et al. (2013) are particularly tested by our C-VNS in subsection 4.5. Then, the gradient type local minimizers for $1 < p \leq 2$ are tested in subsection 4.6, and the problem with arbitrarily large variables is tested in subsection 4.7. Moreover, in subsection 4.8, we compare the performance of our C-VNS with several other methods from the literature.

In order to make use of our method on test instances from this paper public, we made the executable version of GLOBC available at http://www.mi.sanu.ac.rs/~nenad/GLOBC/. All test instances from this paper are already coded in executed version, and can be recognized by their names. Configuration files with parameters for each test problem are also included, with short instruction how to run examples. More information on parameters in configuration files user can find in Dražić et al. (2006).

4.1 Test problems

The following test problems are taken from Hirsch et al. (2009) and Ramadas and Fernandes (2013).

4.1.1 Reactor problem

This problem deals with a model of two continuous non-adiabatic stirred tank reactors with a recycle ratio parameter R.

$$f_1(x_1, x_2) = (1 - R)\left(\frac{D}{10(1+\beta_1)} - x_1\right)\exp\left(\frac{10x_1}{1+\frac{10x_1}{\gamma}}\right) - x_1 \tag{7}$$

$$f_2(x_1, x_2) = x_1 - (1+\beta_2)x_2 + (1-R)\left(\frac{D}{10} - \beta_1 x_1 - (1+\beta_2)x_2\right)\exp\left(\frac{10x_2}{1+\frac{10x_2}{\gamma}}\right)$$

The parameters γ, D, β_1, and β_2 are set to 1000, 22, 2, and 2 respectively, and $x_1, x_2 \in [0, 1]$. Depending on the value of the parameter R, the system has variable number of solutions, presented in Table 1, as well as the minimal Euclidean solution distances.

Table 1: Number of solutions of system (7)

R	# of solutions	solution distance	R	# of solutions	solution distance
0.935	1		0.965	5	0.2528
0.940	1		0.970	5	0.2290
0.9409858	1		0.975	5	0.3098
0.9409859	3	0.0005	0.980	5	0.2530
0.945	3	0.1329	0.985	5	0.1521
0.9472741	3	0.1506	0.9879306	5	0.0009
0.9472742	5	0.0003	0.9879307	3	0.0972
0.950	5	0.0918	0.9884905	3	0.0002
0.9559233	5	0.1669	0.9884906	1	
0.9559234	7	0.0006	0.990	1	
0.960	7	0.1339	0.995	1	
0.9615703	7	0.0008			
0.9615704	5	0.2216			

4.1.2 Steering problem

This is a three-dimensional problem for an automotive steering given by the system

$$\begin{aligned} f_i(x_1,x_2,x_3) = &[E_i(x_2\sin(\psi_i)-x_3)-F_i(x_2\sin(\psi_i)-x_3)]^2 \\ &+[F_i(1+x_2\cos(\psi_i))-E_i(x_2\cos(\psi_i)-1)]^2 \\ &-[(1+x_2\cos(\psi_i))(x+2\sin(\psi_i)-x_3)x_1 \\ &-(x_2\sin(\psi_i)-x_3)(x_2\cos(\psi_i)-x_3)x_1]^2, \quad i=1,2,3 \end{aligned} \tag{8}$$

where

$$\begin{aligned} E_i &= x_2(\cos(\phi_i)-\cos(\phi_0))-x_2x_3(\sin(\phi_i)-\sin(\phi_0))-(x_2\sin(\phi_i)-x_3)x_1 \\ F_i &= -x_2\cos(\psi_i)-x_2x_3\sin(\psi_i)+x_2\cos(\psi_0)+x_1x_3+(x_3-x_1)x_2\sin(\psi_0) \end{aligned}$$

and the values of angles ϕ_i and ψ_i are given in Table 2. In the domain $[0.06,1]^3$, this problem has two solutions.

Table 2: Angular data (in radians) for Steering problem

i	ψ_i	ϕ_i
0	1.3954170041747090114	1.7461756494150842271
1	1.7444828545735749268	2.0364691127919609051
2	2.0656234369405315689	2.2390977868265978920
3	2.4600678478912500533	2.4600678409809344550

4.1.3 A non-smooth problem

This illustrative non-smooth problem is reported in Ramadas and Fernandes (2013) to be not an easy task for some standard minimizers as MATLAB "fsolve" function.

$$\begin{aligned} f_1(x_1,x_2) &= x_1^2-x_2^2 \\ f_2(x_1,x_2) &= 1-|x_1-x_2| \end{aligned} \tag{9}$$

The system has two solutions: $(0.5,-0.5)$ and $(-0.5,0.5)$. The domain for the variables is set to $[-10,10]^2$.

4.1.4 Merlet problem

Merlet problem is defined by the system

$$\begin{aligned} f_1(x_1,x_2) &= -\sin(x_1)\cos(x_2)-2\cos(x_1)\sin(x_2) \\ f_2(x_1,x_2) &= -\cos(x_1)\sin(x_2)-2\sin(x_1)\cos(x_2) \end{aligned} \tag{10}$$

where $0 \le x_i \le 2\pi$. In this domain, the system has 13 solutions.

4.1.5 Floudas problem

Floudas problem is defined by the system

$$f_1(x_1, x_2) = 0.5\sin(x_1 x_2) - 0.25\frac{x_2}{\pi} - 0.5x_1 \tag{11}$$
$$f_2(x_1, x_2) = \left(1 - \frac{0.25}{\pi}\right)\left(e^{2x_1} - e\right) + e\frac{x_2}{\pi} - 2ex_1$$

where $0.25 \leq x_1 \leq 1$ and $1.5 \leq x_2 \leq 2\pi$. In this domain, the system has two solutions.

4.1.6 Effati-Grosan-1 problem

This test problem is defined by the system

$$f_1(x_1, x_2) = \cos(2x_1) - \cos(2x_2) - 0.4 \tag{12}$$
$$f_2(x_1, x_2) = 2(x_2 - x_1) + \sin(2x_2) - \sin(2x_1) - 1.2$$

where $-a \leq x_i \leq a$ with a variable parameter a. In Turgut et al. (2014), one solution is reported for $a = 2$, 13 solutions for $a = 10$, and 127 solutions for $a = 100$.

4.1.7 Effati-Grosan-2 problem

This test problem is defined by the system

$$f_1(x_1, x_2) = e^{x_1} + x_1 x_2 - 1 \tag{13}$$
$$f_2(x_1, x_2) = \sin(x_1 x_2) + x_1 + x_2 - 1$$

where $-a \leq x_i \leq a$ with a variable parameter a. In Turgut et al. (2014), one solution is reported for $a = 2$, $a = 10$, and $a = 100$. The solution is point $(0, 1)$.

4.2 Test results

All experimental results were obtained on Intel Core i7-6700 3.4 GHz processor with 3.35 GB RAM running Windows 7 32bit.

The new method presented in this paper was integrated into the package GLOBC, a test platform for numerical experiments with C-VNS (Mladenović et al. (2008)). It is coded in ANSI C computer language. Used parameter values were explained earlier in section 3. In addition, the time when the last approximate solution was found is recorded, as well as the number of function evaluations (computer effort). The experiment is repeated 100 times with different random seeds, and the average values for 100 runs are presented.

4.2.1 Reactor problem

For the reactor problem, the parameters were set to:

$$tol = 10^{-7},\ \rho_0 = 0.01,\ \rho_{\min} = 0.001,\ q_\rho = 0.1,\ a_0 = 1,\ q_a = 0.5.$$

The results are presented in Table 3.

For different values of parameter R, the number of solutions of the system is presented in the second column. For each value of R, three possible objective functions $F(x)$ in (6) are considered:

$$(1)\ p = 1,\ \varphi(x) = 1 - |x|;\ (2)\ p = 2,\ \varphi(x) = (1 - x^2)^2;\ (3)\ p = 2,\ \varphi(x) = \exp(-x).$$

For each case, 100 runs were executed, and the average number of found solutions, the average time when the last solution was found, and the number of function evaluations were reported. Next, we analyzed the results in more details, mostly to illustrate, and verify the theory from Section 3.

Table 3: Experimental results for Reactor problem

	#	$p=1,\ \varphi(x)=1-x$			$p=2,\ \varphi(x)=(1-x^2)^2$			$p=2,\ \varphi(x)=\exp(-x)$		
R	sol	sol	time	comp.eff.	sol	time	comp.eff.	sol	time	comp.eff.
0.935	1	1	$< 1ms$	1364	1	$< 1ms$	649	1	$< 1ms$	649
0.940	1	1	$< 1ms$	1443	1	$< 1ms$	609	1	$< 1ms$	609
0.9409858	1	2	0.001	1818	13.20	0.132	523200	11.80	0.198	518228
0.9409859	3	2	$< 1ms$	1387	9.74	0.082	342172	11.36	0.120	495416
0.941	3	3	0.025	125154	13.52	0.122	492403	11.62	0.120	493829
0.942	3	3	0.002	8107	6.50	0.058	265471	7.01	0.070	314632
0.943	3	3	0.002	11049	5.84	0.052	240496	5.84	0.067	307757
0.944	3	3	0.006	31124	5.03	0.056	259121	4.96	0.062	289681
0.945	3	3	0.006	30402	4.94	0.060	278547	4.86	0.068	315032
0.946	3	3	0.012	61056	3.38	0.022	106321	3.86	0.052	247542
0.947	3	3	0.002	7664	4.05	0.043	197776	4.01	0.047	216739
0.9472741	3	3	0.002	7766	14.34	0.050	201212	14.5	0.055	221165
0.9472742	5	4	0.011	53098	14.41	0.050	204057	14.56	0.056	223051
0.948	5	5	0.018	81210	7.66	0.046	193972	7.42	0.050	212451
0.949	5	5	0.003	15023	7.01	0.046	194072	7.03	0.053	225564
0.950	5	5	0.005	22140	5.67	0.025	107341	5.51	0.023	98000
0.951	5	5	0.005	24464	5.33	0.016	66210	5.48	0.024	103602
0.952	5	5	0.009	41876	5.31	0.019	80632	6.24	0.018	78641
0.953	5	5	0.006	25364	5.21	0.016	67856	5.18	0.016	71457
0.954	5	5	0.002	10745	5.10	0.006	24749	5.53	0.031	135072
0.955	5	5	0.009	43527	5.01	0.002	7767	5.05	0.004	16817
0.9559233	5	6	0.010	44519	13.80	0.046	177422	13.76	0.053	205072
0.9559234	7	6	0.008	36452	13.89	0.047	180104	14.00	0.055	211253
0.956	7	7	0.013	51528	12.69	0.049	190598	12.35	0.057	219064
0.957	7	7	0.011	47800	7.03	0.007	28413	7	0.006	25822
0.958	7	7	0.029	124790	7.08	0.014	57930	7.01	0.012	49132
0.959	7	7	0.032	138374	7	0.018	76115	7	0.022	91034
0.960	7	7	0.014	60275	7	0.008	31183	7	0.007	31080
0.961	7	7	0.007	30628	7.79	0.042	171440	7.73	0.049	196013
0.9615703	7	6	0.007	31143	14.30	0.08	303903	14.97	0.099	367160
0.9615704	5	5	0.004	19021	14.34	0.080	305326	14.94	0.106	386703
0.962	5	5	0.008	38184	5	0.002	9887	5	0.003	11288
0.965	5	5	0.009	41242	5	0.004	16137	5	0.004	18646
0.970	5	5	0.006	27189	5	0.002	10262	5	0.002	10895
0.975	5	5	0.004	17493	5	0.002	8636	5	0.002	10827
0.980	5	5	0.004	16767	5	0.002	9613	5	0.003	12222
0.985	5	5	0.003	14465	5	0.002	9613	5	0.001	4350
0.986	5	5	0.002	6886	5	0.002	8299	5	0.002	9426
0.987	5	5	0.003	8889	5.09	0.007	28594	5.23	0.018	75273
0.9879306	5	4	0.001	6512	12.89	0.090	351595	12.57	0.104	399692
0.9879307	3	3	0.001	6887	12.89	0.090	350093	12.59	0.106	410065
0.988	3	3	0.002	7267	3	0.001	3488	3	0.001	4621
0.9884905	3	2	0.001	3698	6.96	0.034	155847	7.05	0.038	168540
0.9884906	1	1	$< 1ms$	600	6.96	0.034	155476	6.88	0.037	168164
0.989	1	1	$< 1ms$	621	1	$< 1ms$	393	1	$< 1ms$	393
0.990	1	1	$< 1ms$	608	1	$< 1ms$	427	1	$< 1ms$	428
0.995	1	1	$< 1ms$	201	1	$< 1ms$	69	1	$< 1ms$	69

Analysis of the results. As it can be seen from Table 1, for $R = 0.9409858$, $R = 0.9472741$, $R = 0.9559233$, $R = 0.9615703$, $R = 0.9879306$, and $R = 0.9884905$ the system has double roots. Thus, when R is near these values, for x in δ neighborhood of the solution, according to Lemmas 1–5, we expect that $F(x) \sim \delta^2$ for $p = 1$ and $F(x) \sim \delta^4$ for $p = 2$. For values of R not near these values, we expect $F(x) \sim \delta$ for $p = 1$ and $F(x) \sim \delta^2$ for $p = 2$. This is the reason why, in case $p = 2$, C-VNS finds a large number of approximate solutions for R near the value where multiple system roots are present. To avoid this problem, one can increase the value of $\rho_{\min}$, but then two solutions closer than $\rho_{\min}$ cannot be distinguished. In the case $p = 1$, the C-VNS obtained the true number of solutions, except for the critical values of R, where it also behaved well. For $R = 0.94009858$ the C-VNS found two solutions: one true solution, and another false approximate solution with very small objective function value in the place where two new solutions emerged.

For $R = 0.94009858$, only one of those two new solutions was found, since they are at a distance 0.0005, which is less than $\rho_{\min} = 0.001$. The same argument holds for other values of R with multiple solutions.

In Hirsch et al. (2009), the objective function with $p = 2$ and $\varphi(x) = \exp(-x)$ is proposed with fixed $\rho = 0.01$. The only difference from the third case in Table 3 is that $\rho_{\min} = 0.001$ is used in present tests. Taking $\rho_{\min} = 0.01$, the correct number of solutions for cases reported in Hirsch et al. (2009) is verified.

As a conclusion one can say that more accurate results can be given using $p = 1$. For non-smooth problems, the usability of the method is also extended using a robust optimizer. In case $p = 2$ and a smooth $\varphi(x)$, the objective function does not reduce the smoothness of the initial problem, but even if $f_i(x)$ are analytic, the function $F(x)$ is not, although it is smooth, and higher order minimizers may not perform as expected near already found solutions. There is no significant difference in test results for $\varphi(x) = (1 - x^2)^2$ and $\varphi(x) = \exp(-x)$, so it is better to use the first kernel function as it is smooth, and it can be evaluated more quickly than the second, which is neither smooth nor continuous.

4.2.2 Additional test problems

For all test problems in this section, unless otherwise stated, the parameters were set to: $tol = 10^{-7}$, $\rho_0 = 0.01$, $\rho_{\min} = 0.001$, $q_\rho = 0.1$, $a_0 = 1$, $q_a = 0.5$, and maximum number of approximate results was limited to 20. The results are presented in Table 4.

Table 4: Experimental results for other test problems

Problem	#		$p = 1,\ \varphi(x) = 1 - \lvert x\rvert$			$p = 2,\ \varphi(x) = (1 - x^2)^2$		
name	sol	Precision	sol	time	comp.eff.	sol	time	comp.eff.
Steering	2	$\rho_{\min} = 0.001$	2	0.005	7451	20	0.003	3423
	2	$\rho_{\min} = 0.01$	2	0.002	3381	20	0.060	78583
	2	$\rho_{\min} = 0.1$	2	0.061	86609	10.13	0.086	118337
Non-smooth	2		2	$< 1ms$	726	2	$< 1ms$	359
Merlet	13		13	0.018	55596	13	0.011	33199
Floudas	2	$\rho_{\min} = 0.001$	2	0.002	7808	4.05	0.009	37341
	2	$\rho_{\min} = 0.01$	2	$< 1ms$	608	2	0.001	5193
Effati-Grosan-1	1	$a = 2$	1	$< 1ms$	739	1	$< 1ms$	81
	13	$a = 10$	13	0.038	120451	13	0.014	40798
	25	$a = 20$	25	0.211	528947	25	0.064	150501
	63	$a = 50$	63	2.652	3822159	63	0.973	1351649
	127	$a = 100$	127	12.306	11310533	127	3.876	3242900
Effati-Grosan-2	1	$a = 2$	1	$< 1ms$	530	1	$< 1ms$	92
	1	$a = 10$	1	$< 1ms$	973	1	$< 1ms$	114
	1	$a = 100$	1	$< 1ms$	2093	1	$< 1ms$	310

For the Steering problem, the parameters were set to: $\rho_0 = 10\rho_{\min}$, $\rho_{\min} = 0.1, 0.01, 0.001$. As it can be seen, $p = 1$ gives accurate results, and $p = 2$ is not suitable for this problem.

For the Non-smooth problem, in both cases $p = 1$ and $p = 2$, two solutions were found within less than $1ms$, with 726 and 359 function evaluations, respectively. It is worth mentioning that for $p = 1$, the errors in two obtained solutions were of order 10^{-8}, but for $p = 2$, the errors were of order 10^{-4}. For obtaining the errors of order 10^{-8} in case of $p = 2$, one must set $tol = 10^{-14}$ or smaller, which is near the limit of machine precision. This also applies for all other test problems.

For the Merlet problem,both cases $p = 1$ and $p = 2$ were successful, and all 13 solutions were found with comparable time and computer effort.

In the Floudas problem, case $p = 1$ was successful, but $p = 2$ produced some false solutions for $\rho_{\min} = 0.001$. Only for larger $\rho_{\min} = 0.01$, the accurate number of solutions is obtained. The drawback of taking larger values for $\rho_{\min}$ is that different solutions could not be distinguished if they are closer than $\rho_{\min}$.

For the Effati-Grosan-1 and Effati-Grosan-2 problems, the maximum number of approximate solutions was limited to 200. Both cases $p = 1$ and $p = 2$ were successful and all solutions were found. For $p = 2$, the solutions were obtained with considerably less computer effort, however, as stated before, for $p = 1$ the errors in the obtained solutions were of order 10^{-8}, but for $p = 2$ the errors were of order 10^{-4}. For obtaining the errors of order 10^{-8} in case of $p = 2$, one must set $tol = 10^{-14}$ or smaller.

4.3 Comparison with C-GRASP

For comparison with the results from Zhou et al. (2013), the parameters were set to: $tol = 10^{-7}$, $\rho_0 = 0.1$, $\rho_{\min} = 0.01$, $q_\rho = 0.1$, $a_0 = 1$, $q_a = 0.5$, and maximum number of approximate results was limited to 20. The results are presented in Table 5.

Table 5: Comparison of two C-VNS variants with C-GRASP for the Reactor problem

	#	$p=1,\ \varphi(x)=1-\lvert x\rvert$			$p=2,\ \varphi(x)=(1-x^2)^2$			$p=2$, GRASP	
R	sol	sol	time	comp.eff.	sol	time	comp.eff.	sol	time
0.935	1	1	$< 1ms$	1364	1	$< 1ms$	649	1	0.600
0.940	1	1	$< 1ms$	1443	1	$< 1ms$	609	1	0.770
0.945	3	3	0.001	5674	3	0.001	2810	3	0.190
0.950	5	5	0.018	81332	5	0.011	48143	4.99	1.110
0.955	5	5	0.003	13038	5	0.003	11141	5	1.690
0.960	7	7	0.005	19566	7	0.005	21512	6.96	2.410
0.965	5	5	0.003	15540	5	0.001	5793	4.95	1.810
0.970	5	5	0.001	4309	5	0.001	4087	4.99	1.340
0.975	5	5	0.003	15561	5	0.002	6773	4.98	1.830
0.980	5	5	0.002	9134	5	0.001	3900	4.98	1.900
0.985	5	5	0.002	8980	5	0.002	6161	4.99	2.230
0.990	1	1	$< 1ms$	608	1	$< 1ms$	428	1	0.010
0.995	1	1	$< 1ms$	201	1	$< 1ms$	69	1	0.010

One can observe the following:

(i) When we compare C-VNS and C-GRASP using the same transformation function (for $p = 2$), we see that our C-VNS is much faster and more precise than GRASP; the average number of solutions for the C-GRASP is less than the correct number in 7 out of 13 instances;

(ii) Both C-VNS variants recognized all solutions; as expected, the smooth problem for p=2 is solved faster, but its stopping condition is smaller and therefore, the solution is less precise.

Table 6: Comparison of two C-VNS variants with C-GRASP on Steering, Merlet and Floudas problems

	#		p=1, $\varphi(x)=1-\lvert x\rvert$		p=2, $\varphi(x)=(1-x^2)^2$		p=2, GRASP
Problem name	sol		time	comp.eff.	time	comp.eff.	time
Steering problem	2	1st solution	0.001	1883	$< 1ms$	110	0.840
	2	all solutions	0.002	3381	$< 1ms$	221	5.060
Merlet problem	13	1st solution	$< 1ms$	532	$< 1ms$	70	0.004
	13	all solutions	0.020	62394	0.009	27249	3.00
Floudas problem	2	1st solution	$< 1ms$	638	$< 1ms$	61	0.071
	2	all solutions	0.001	5193	$< 1ms$	608	0.390

For the Steering problem, in Hirsch et al. (2009) only the average times for obtaining the first and the second solutions are reported: $0.84s$ and $5.06s$. For $p = 1$, $\varphi(x) = 1 - |x|$, C-VNS found the first solution within 0.001 second (1883 function calls), and both solutions within 0.001 second (3381 function calls). For $p = 2$, $\varphi(x) = (1 - x^2)^2$, the times were both under $1ms$, with 110 and 221 function calls for the first and for both solutions. For this problem, when $p = 2$, we observe that a lot of false solutions are found, both for $\varphi(x) = 1 - |x|$ and $\varphi(x) = \exp(-x)$, which is illustrated in Table 4.

For the Merlet problem, from the figure presented in Hirsch et al. (2009), average times for obtaining the first and all 13 solutions are approximately $0.004s$ and $3s$. For $p = 1$, $\varphi(x) = 1 - |x|$, C-VNS found the first

solution for less than $1ms$ (532 function calls), and all 13 solutions for $0.020s$ (62394 function calls). For $p = 2$, $\varphi(x) = (1-x^2)^2$, the first solution was found for less than $1ms$ (70 function calls), and all 13 solutions for $0.009s$ (27249 function calls).

Finally, for the Floudas problem, average times for obtaining the first and both solutions are $0.071s$ and $0.390s$ as reported in Hirsch et al. (2009). For $p = 1$, $\varphi(x) = 1 - |x|$, C-VNS found the first solution within less than $1ms$ (638 function calls), and both solutions within 0.001 seconds (5193 function calls). For $p = 2$, $\varphi(x) = (1 - x^2)^2$, the first solution was found for less than 1 ms (61 function calls), and both solutions for less than 1 ms (608 function calls).

4.4 Comparison with Harmony search

In Ramadas and Fernandes (2013), Harmony search meta-heuristic (HS) was applied to solve the nonlinear system of equations. Five variants of HS algorithms were proposed, but all of them stopped after finding the first solution. The merit function for minimization is $M(x) = \sum_{i=1}^{n} f_i(x)^2$. The solution is considered to be found if $\sqrt{M(x)} \leq 10^{-6}$. Each test problem was repeated 30 times, and only the best result (number of function evaluations) is reported if the solution was found.

In order to compare C-VNS for NSE with HS, we set $tol = 10^{-6}$ for $p = 1$ and $tol = 10^{-12}$ for $p = 2$. Other parameters were set to $\rho_0 = 0.01$, $\rho_{\min} = 0.001$, $q_\rho = 0.1$, $a_0 = 1$, and $q_a = 0.5$. We repeated each test problem 30 times, and C-VNS for NSE was successful in every run. In Table 7 for each test problem, average and the minimal number of function calls in 30 runs are reported. The last column contains the best result from all 5 variants of HS meta-heuristic.

Table 7: Comparison of C-VNS and Harmony search

		$p=1,\ \varphi(x)=1-\lvert x\rvert$		$p=2,\ \varphi(x)=(1-x^2)^2$		HS best
Problem name	Parameter	average	minimum	average	minimum	minimum
Non-smooth		163	88	108	86	278
Reactor	R =0.95	102	88	96	84	550
Steering		1725	129	783	171	not found
Merlet		129	84	101	77	292
Floudas		136	81	110	71	550
Effati-Grosan-1	$a = 2$	146	105	121	96	517
	a=10	214	99	154	98	501
	a=100	402	123	155	121	445
Effati-Grosan-2	$a = 2$	138	116	121	106	484
	$a = 10$	179	78	182	71	463
	$a = 100$	397	145	438	142	482

From Table 7, it is evident that C-VNS for NSE outperforms HS considerably in every test problem. C-VNS obtained the correct solutions for all test problems in every test run. For HS, one or more variants were unsuccessful for all 30 runs, but how many times they were unsuccessful, were not reported.

4.5 Test problems from Floudas et al. (1999)

In addition to the problems from this section, which were also solved with other algorithms, we tested the effectiveness of our method on five problems from Floudas et al. (2013), Chapter 14, Section 1. Note that Problem 4 has been discussed in more details earlier in this section (Floudas et al. (1999) problem from 4.1.5). For all test problems in this subsection, the parameters were set to: $tol = 10^{-7}$, $\rho_0 = 0.01$, $\rho_{\min} = 0.001$, $q_\rho = 0.1$, $a_0 = 1$, $q_a = 0.5$, and maximum number of approximate results were limited to $K = 20$. The tests were repeated 100 times and the average results are presented in Table 8.

As in the previous test problems, the option $p = 1$, $\varphi(x) = 1 - |x|$ was more successful than $p = 2$, $\varphi(x) = (1 - x^2)^2$. For the first option, we obtained the correct number of solutions in all test runs. The second option resulted in correct number of solutions for problems 1, 4, 5, and 6, but for problems 2 and 3, it

Table 8: Comparison of two transformation functions for problems from (Floudas et al. 1999)

Problem	dim-ension	# of solutions	p=1, $\varphi(x) = 1 - \|x\|$			p=2, $\varphi(x) = (1 - x^2)^2$		
			sol	time	comp. effort	sol	time	comp. effort
14.1.1 Himmelblau function	2	9	9	0.005	23260	9	0.003	12793
14.1.2 Equilibrium Combustion	5	1	1	0.051	634531	20	0.071	176948
14.1.3 Test Problem 3	2	1	1	0.001	3013	20	0.094	274005
14.1.4 Test Problem 4	2	2	2	0.002	7808	4	0.009	37341
14.1.5 Test Problem 5	5	2	2	0.047	179782	2	0.001	4987
14.1.6 Test Problem 6	8	16	16	2.700	5926615	16	0.963	1866031

gave large number of false solutions ($\leq K = 20$). That number would be even larger, if the maximum number of approximate solutions K was set to be larger. The difference in execution times is the consequence of significantly better precision of solutions in case $p = 1$ than $p = 2$ for the same value of $tol = 10^{-7}$. According to Lemma 2, the expected accuracy of the solutions is $\varepsilon \sim tol$ for $p = 1$, and $\varepsilon \sim \sqrt{tol}$ for $p = 2$, and more iterations are necessary for more accurate solutions.

Test problems from Table 8 also appear in Maranas and Floudas (1995) and Stuber et al. (2010). In both papers branch-and-bound global optimization algorithm with convex relaxation was applied. Since one iteration in these methods includes a convex minimization with unknown number of function evaluations, CPU times remain the only way to compare the efficiency of methods. In Maranas and Floudas (1995), all 6 problems from Table 8 were solved on a 66MHz HP-730 workstation with tolerance 1e-4. In Stuber et al. (2010), 5 problems from Table 8 were tested on 2.66 GHz Intel Core2 Quad processor with tolerance 1e-8. One of test problems (14.1.2) did not converge. Other problems from this paper are either single variable equations or functions to be minimized, not equations.

We used the same parameters as for $p = 1$, $\varphi(x) = 1 - |x|$ option in Table 8 with matching tolerances $tol = 10^{-4}$ and $tol = 10^{-8}$. Each test was repeated 100 times, and average execution time was recorded. Test results are presented in Table 9. CPU times of examples from Maranas and Floudas (1995) and Stuber et al. (2010) are presented in the fifth column as well as equivalent CPU times scaled as it were executed on a 3.4 GHz processor to compare with our results in the last column. Different CPU times for two test problems from Maranas and Floudas (1995) are obtained for different convex lower bounding parameter α. As seen from the results, our algorithm is at least 10 times more efficient for 14.1.1–4 in Maranas and Floudas (1995) and for 14.1.1–5 in Stuber et al. (2010), and for remaining examples gives comparable results.

Table 9: Comparison with problems from (Maranas and Floudas 1995) and (Stuber et al. 2010)

Problem	dim-ension	# of solutions				p=1, $\varphi(x) = 1 - \|x\|$	
			sol	time	equiv. time	sol	time
			Maranas and Floudas (1995), tol = 1e-4			tol = 1e-4	
14.1.1 Himmelblau function	2	9	9	10.89	0.211	9	0.003
14.1.2 Equilibrium Combustion	5	1	1	31.7	0.615	1	0.054
14.1.3 Test Problem 3	2	1	1	1.5	0.029	1	0.001
14.1.4 Test Problem 4	2	2	2	2	0.039	2	0.001
14.1.5 Test Problem 5	5	2	2	0.35 - 22.16	0.007 - 0.430	2	0.014
14.1.6 Test Problem 6	8	16	16	141.41 - 987.91	2.127 - 19.177	16	0.968
			Stuber et al. (2010), tol = 1e-8			tol = 1e-8	
14.1.1 Himmelblau function	2	9	9	0.082	0.063	9	0.005
14.1.2 Equilibrium Combustion	5	1	0	not found		1	0.174
14.1.3 Test Problem 3	2	1	1	0.02	0.015	1	0.001
14.1.5 Test Problem 5	5	2	2	0.755	0.577	2	0.058
14.1.6 Test Problem 6	8	16	16	3.627	2.774	16	2.661

4.6 Comparison with BARON

BARON (Sahinidis 1996) is a highly ranked computational system for solving non-convex optimization problems to global optimality. Purely continuous, purely integer, and mixed-integer nonlinear problems can be solved with the software. It uses advanced branch-and-bound optimization concepts. As described in BARON manual, it can be used to find all solutions of NSE by declaring the equations from the system as constraints, and finding all feasible points for such a problem.

We used the demo version of BARON with suggested MATLAB interface. Demo version can handle problems with up to 10 variables, 10 constraints and 50 nonlinear operations. Further, both demo and commercial versions can not handle trigonometric functions. For these reasons, we were limited in our comparison to problems which satisfy these limitations. There are several tolerance parameters in BARON, and we used their default values: Absolute termination tolerance 1e-6, Relative termination tolerance 1e-9, Absolute constraint feasibility tolerance 1e-5, Separation distance between solutions 1e-4. Developers of BARON told us that BARON uses an absolute tolerance of 1e-11, so we tested our algorithm with the same tolerance 1e-11 in all test instances, with $p = 1$, $\varphi(x) = 1 - |x|$, and other parameters the same as in previous tests. We repeated our tests 100 times, and recorded average results. We repeated BARON tests for 50 times. In each run BARON reported the same number of solutions, and the execution times were almost identical, which is expectable from the exact solver.

In order to check the accuracy of results obtained by BARON, we calculated the average value of function values $F_0(x)$ from (3) taken for all solutions reported by BARON. These values are marked as "F0 aver." in following tables. By algorithm design, $F_0(x) \leq tol =$ 1e-11 for all solutions obtained with our algorithm.

We first tested BARON on problems from (Floudas et al. 1999). The problem 14.1.4 and 14.1.6 exceed the limitation of a demo version, but we presented the result for 14.1.6 (Robot kinematics problem) from the BARON manual (number of solutions, timing not present). The comparison results are presented in Table 10. Our algorithm found the correct number of solutions in every run for all problems. BARON found the correct number of solutions just for problems 14.1.1 and 14.1.5. For problems 14.1.2 and 14.1.3, BARON reports false larger number of solutions with significantly larger residual values $F_0(x)$. Those values are, on the other hand, in consent with BARON's Absolute constraint feasibility tolerance of 1e-5. Our algorithm was considerably faster on all tested problems. In problem 14.1.6 BARON reported only 14 out of 16 solutions.

Table 10: Comparison with BARON solver for problems from (Floudas et al. 1999)

			p=1, $\varphi(x) = 1 - \|x\|$ $tol = 1e-11$		BARON		
Problem	dim-ension	# of solutions	solutions	time	solutions	time	F0 aver.
14.1.1 Himmelblau function	2	9	9	0.006	9	0.153	1.34e-07
14.1.2 Equilibrium Combustion	5	1	1	0.160	5	0.640	1.24e-05
14.1.3 Test Problem 3	2	1	1	0.001	13	3.012	6.74e-06
14.1.5 Test Problem 5	5	2	2	0.007	2	0.050	2.42e-11
14.1.6 Test Problem 6	8	16	16	7.227	14		

For a Non-smooth problem (9), our algorithm found both solutions for less than 1ms, and BARON also found both solutions for 0.074s with F0 aver. value of 1.78e-15.

Last problem to consider in comparison was the Reactor problem (7), with bifurcation phenomena while varying the parameter R. The comparison results are presented in Table 11. For various values of parameter R, the second column contains the exact number of solutions. Next three columns contain test results for our algorithm, with $tol =$ 1e-11: number of solutions found, time in seconds and number of function evaluations. Last three columns contain test results for BARON: number of solutions, time in seconds, and average value of function values $F_0(x)$ from (3) taken for all solutions reported by BARON. By algorithm design, $F_0(x) \leq tol =$ 1e-11 for all our solutions. Values from the last column show that our solutions are better than BARON solutions, and are found significantly faster. It is evident that near bifurcation points, where double solution exists, BARON reports false larger number of solutions with 5 orders of magnitude larger residual values $F_0(x)$, but in consent with BARON's tolerance parameters.

Table 11: Comparison with BARON for Reactor problem

	#	$p=1,\ \varphi(x)=1-\|x\|$			BARON		
R	sol	sol	time	comp.eff.	sol	time	F0 aver.
0.935	1	1	$< 1ms$	1649	1	0.197	6.76e-12
0.940	1	1	$< 1ms$	1608	1	0.210	5.57e-12
0.9409858	1	1	$< 1ms$	1573	4	0.759	2.26e-06
0.9409859	3	2	$< 1ms$	1762	7	0.534	2.27e-06
0.941	3	3	0.031	148920	5	0.359	2.18e-06
0.942	3	3	0.003	13006	3	0.242	4.43e-11
0.943	3	3	0.002	9621	3	0.210	3.37e-11
0.944	3	3	0.003	14900	3	0.164	1.37e-10
0.945	3	3	0.007	24633	3	0.168	2.51e-11
0.946	3	3	0.016	79518	3	0.131	4.92e-10
0.947	3	3	0.002	9653	3	0.231	4.07e-11
0.9472741	3	3	0.002	9289	17	4.693	3.30e-06
0.9472742	5	4	0.016	76211	12	0.644	1.62e-06
0.948	5	5	0.023	104685	5	0.276	1.80e-06
0.949	5	5	0.004	19532	5	0.205	3.87e-09
0.950	5	5	0.006	25774	5	0.167	7.58e-10
0.951	5	5	0.008	34055	5	0.170	3.32e-07
0.952	5	5	0.011	51219	5	0.138	1.68e-08
0.953	5	5	0.007	31393	5	0.153	4.05e-11
0.954	5	5	0.003	14176	5	0.135	9.46e-11
0.955	5	5	0.006	27116	5	0.161	6.10e-11
0.9559233	5	5	0.009	42093	18	2.461	1.98e-06
0.9559234	7	6	0.010	43402	16	0.398	1.82e-06
0.956	7	7	0.015	65183	7	0.266	7.63e-07
0.957	7	7	0.015	64510	7	0.185	4.93e-09
0.958	7	7	0.035	147791	7	0.180	7.20e-07
0.959	7	7	0.037	155884	7	0.161	9.10e-07
0.960	7	7	0.017	71426	7	0.194	4.88e-11
0.961	7	7	0.010	43961	7	0.167	1.07e-07
0.9615703	7	6	0.009	39251	20	0.600	3.18e-06
0.9615704	5	5	0.007	32628	15	4.044	3.65e-06
0.962	5	5	0.009	43260	5	0.148	4.73e-09
0.965	5	5	0.011	50991	5	0.120	3.71e-09
0.970	5	5	0.009	40160	5	0.111	1.70e-09
0.975	5	5	0.005	22505	5	0.114	5.27e-11
0.980	5	5	0.004	20178	5	0.119	2.27e-09
0.985	5	5	0.004	16702	5	0.121	4.19e-09
0.986	5	5	0.002	11157	5	0.146	3.40e-09
0.987	5	5	0.003	12617	5	0.177	6.66e-11
0.9879306	5	4	0.002	8103	10	0.329	3.60e-06
0.9879307	3	3	0.002	9037	8	1.517	2.78e-06
0.988	3	3	0.002	9730	3	0.210	6.81e-11
0.9884905	3	2	0.001	5100	9	0.392	4.09e-06
0.9884906	1	1	$< 1ms$	764	6	0.380	5.81e-06
0.989	1	1	$< 1ms$	809	1	0.160	8.38e-11
0.990	1	1	$< 1ms$	831	1	0.097	7.18e-11
0.995	1	1	$< 1ms$	272	1	0.078	3.65e-11

4.7 Testing gradient type local minimizers for $1 < p \leq 2$

In all previous computational results, we have used the direct search Nelder-Mead (NM) method for finding a local minimum of a continuous function within C-VNS. The use of just one descent method for all test problems and all values of parameter p is twofold: (i) NM is applicable also in both smooth and non-smooth optimization; (ii) the desired property of any heuristic is its simplicity. However, if the optimization problem is smooth, then a gradient type method can be used instead. If the functions $f_i(x),\ i = 1, \ldots, n$ and $\varphi(x)$ are smooth, and $p > 1$, the function (6) will be smooth as well, so gradient type methods can be used for local minimization. For $p = 1$, function (6) is not differentiable in the solution points. For the same value of δ in stopping condition $F(x) < \delta$, according to results in Section 3, smaller values of p result in more accurate solutions. Moreover, computational results from this section show that the choice of $p = 1$ is much less prone to false solution detection than $p = 2$. On the other hand, for $p = 1 + \varepsilon$ and $p < 2$, although the gradient

of $F(x)$ is continuous, the second derivatives are singular at the solution points. This fact clearly shows the inferiority of gradient methods.

In order to confirm experimentally this theoretical observation, i.e., to show that gradient type methods cannot be efficient for $p = 1 + \varepsilon, p < 2$, we conducted the following experiment. In addition to NM local optimizer, we selected classical gradient (Steepest descend), and two well-known conjugate-gradient local optimizers, i.e., Fletcher-Powell and Fletcher-Reeves methods. Effati-Grosan-1 test problem for $a = 2$ is solved for various values of $1 < p \leq 2$ with fixed parameters: $tol = 10^{-6}$, $\rho_0 = 0.1$, $\rho_{\min} = 0.01$, $q_\rho = 0.1$, $a_0 = 1$, $q_a = 0.5$, $\varphi(x) = (1 - x^2)^2$. Gradients were computed numerically with step $h = 10^{-8}$. Quadratic approximation method was used for line optimization. Tests were repeated 30 times for each method and for each value of p. The average number of function evaluations until the solution was found is presented in Table 12.

Table 12: Number of function evaluation (or computational efforts) of different local optimizers for $1 \leq p \leq 2$

p	Nelder-Mead	Steepest descend	Fletcher-Powell	Fletcher-Reeves
2.0	75	270	78	78
1.9	79	277	103	116
1.8	78	284	133	154
1.7	84	289	156	201
1.6	88	391	209	248
1.5	96	524	278	335
1.4	101	641	299	494
1.3	110	947	382	633
1.2	115	12975	1057	2104
1.1	128	10547297	25174	307915
1.0	146	-	-	-

The solution was found in every run in all cases and for each method. As it can be seen from the Table 12, NM performed well for all values of p. For $p = 2$, Fletcher-Powell and Fletcher-Reeves methods were equally efficient as Nelder-Mead, while Steepest descend performed worse. Reducing the value of p, all gradient type methods were gradually less efficient, and for p close to 1, they were extremely inefficient. More iterations for smaller values of p are expected because more accurate solutions were obtained for the same fixed tolerance in stopping criterion. Again, a theoretical explanation of gradient type methods poor behavior for smaller values of $1 < p < 2$ (that the second derivatives are singular at solution points for $p < 2$), is confirmed experimentally.

4.8 Problem with arbitrarily large dimension

The purpose of this subsection is to check how good is our new method in solving large dimensional problems. The discrete integral equation problem, taken from Martinez (1986) is an example of a dense nonlinear system with arbitrary number of variables n,

$$f_i(x) = x_i + \frac{h}{2}\left[(1-t_i)\sum_{j=1}^{i} t_j(x_j + t_j + 1)^3 + t_i\sum_{j=1}^{i}(1-t_j)(x_j + t_j + 1)^3\right], \tag{14}$$

where $i = 1, .., n$, $h = 1/(n+1)$, $t_i = ih$, and $x_0 = x_{n+1} = 0$. There are no reported details on intervals and number of solutions. We tested the effectiveness and the efficiency of our method for various values of $n \leq 1000$. In all the tests, we used $tol = 10^{-6}$, $\rho_0 = 0.01$, $\rho_{\min} = 0.001$, $q_\rho = 0.1$, $a_0 = 1$, $q_a = 0.5$, $p = 1$, $\varphi(x) = 1 - |x|$. Solutions are searched in the hypercube $[-2, 2]^n$. Hook-Jeeves pattern search local minimizer was used since it is better adapted than NM to large-scale problems. In every instance, exactly one solution was detected. Instances of 10 different dimensions $n = 20, 50, 100, 200, ..., 1000$ were run by our C-VNS based heuristic. For each n, the execution time and the number of function evaluations until the solution was found are reported in Table 13.

It appears that our C-VNS based heuristic is able to correctly solve problem with 100 variables in less than 0.5 seconds, and NSE with 1000 variables in less than 6 minutes.

Table 13: Experimental results for large dimensional problem (14)

n	time	comp.eff.	n	time	comp.eff.
20	0.006	3362	500	37.449	93145
50	0.046	8745	600	64.032	110558
100	0.339	19062	700	108.324	135999
200	2.544	38072	800	186.653	179410
300	9.786	66703	900	229.394	176608
400	19.877	75931	1000	325.430	204244

In Martinez (1986), this problem was solved using gradient type accelerated successive orthogonal projections method for $n = 500$ and $n = 1000$. Tolerance used in stopping condition is not reported. Since single precision arithmetic on PDP10 computer was used, we assumed that $tol > 10^{-6}$. For $n = 500$, convergence took place after 5 iterations, 826 projection steps and 104 seconds in best case, and for $n = 1000$, after 5 iterations, 1647 projection steps and 426 seconds. As in every iteration a Jacobian matrix has to be calculated, followed by the decomposition of matrix using orthogonal projections method, and solving a system of equations of dimension n, it is very hard to compare computational effort with our method. Nevertheless, we can say that our general NSE solver is comparable with problem specific method proposed in Martinez (1986).

4.9 Comparison with more methods

In this subsection, we compare our C-VNS based heuristic for solving NSE with some more methods. First of all, it is clear that such a task in case of NSE is almost impossible:

1. There are more than 30 methods in the literature;
2. Many of such methods use a single instance that is not available;
3. Very often CPU times or number of function evaluations are not provided;
4. Most methods do not look for all solutions of the problem in a systematic way. Some methods just look for a single solution;
5. Different methods use different objective functions, so it is difficult to compare the accuracy of solutions.
6. When the same instances are considered in different methods, then the following problems in comparison appear:
   (a) not the same space domain is used, resulting in different number of solutions in that domain and different efficiency;
   (b) not the same stopping condition is used, and often it is not clear what is the precision of the solutions;
   (c) not the same way of presenting results is used; in some papers results are given in tables and some graphically. In many papers only the number of iterations is reported with unclear computer efforts for one iteration.

Despite of all difficulties, we try here to do comparative analysis in the following way. First, we select about 20 papers that propose methods for solving NSE. Then, we divide them into 2 groups: (i) papers that use the same instances as we do; (ii) papers with instances that we did not use, but having at least two common instances.

4.9.1 Comparison with methods that use the same instances as us.

Comparison of our method with C-GRASP (Hirsch et al. 2009) and Harmony search algorithm (Ramadas et al. 2014) was already presented in the previous sections. Here, we compare our results with the results reported by other methods. Since the objective functions are different in the following methods, we calculate the value *EqTol* as the value of our objective function (3) ($p = 1$) on the solutions provided by those algorithms.

Reactor problem, defined in 5.1.1. and solved in 5.2.1.

- Biased random-key genetic algorithm in Silva et al. (2014) found the correct number of solutions for various values of parameter R with $EqTol > 10^{-3}$. Average results in the form (R, time) are: (0.945, 29.820s), (0.950, 0.415s), (0.955, 0.379s), (0.960, 2.611s), (0.965, 6.329s), (0.970, 6.844s), (0.975, 8.270s), (0.980, 10.980s), (0.985, 11.970s). Our method found all solutions with $tol = 10^{-7}$ but, at least, 40 times faster (see Table 3, case $p = 1$).
- Invasive weed optimization algorithm in Pourjafari and Mojallali (2012) found the correct number of solutions for $0.935 \leq R \leq 0.960$. The solutions are presented with 7 decimal places, but there were no data regarding tolerance, time, and number of function evaluations. Therefore, direct comparison was not possible.

Steering problem, defined in 5.1.2. and solved by C-VNS in 5.2.2.

- Biased random-key genetic algorithm in Silva et al. (2014) finds two solutions with $EqTol > 10^{-3}$ for 0.215s. Our method finds both solutions with $tol = 10^{-7}$ for $0.005s$ or less, as can be seen in Table 4. Thus, our C-VNS based method is faster and more effective.

Merlet problem. defined in 5.1.4. and solved in 5.2.2.

- Biased random-key genetic algorithm in Silva et al. (2014) finds all 13 solutions with $EqTol > 10^{-3}$ for 45s. Our method finds all 13 solutions with $tol = 10^{-7}$ for 0.018 s, as can be seen in Table 4. Again, our C-VNS based heuristic is more precise and efficient.

Floudas problem, defined in 5.1.5. and solved in 5.2.2.

- Biased random-key genetic algorithm in Silva et al. (2014) finds two solutions with $EqTol > 10^{-3}$ for 0.608s. Our method is able to find both solutions with $tol = 10^{-7}$ for less than $0.001s$, as can be seen in Table 4.
- Cuckoo optimization algorithm in Abdollahi et al. (2016) finds two solutions with $EqTol = 2 \cdot 10^{-17}$ in 150 iterations, using 10,000 function evaluations. According to the convergence graph provided in the paper, it would take 30 iterations (around 2000 function evaluations) if $EqTol = 10^{-7}$. For the same $tol = 10^{-7}$, our method is able to find both solutions for 608 function evaluations, which is more than three times less than the number of evaluations in Abdollahi et al. (2016).
- Imperialist competitive algorithm in Abdollahi et al. (2013) finds also 2 solutions, but with $EqTol = 10^{-12}$, using 250 iterations and 200 countries (50000 function evaluations). From the presented convergence graph, for $EqTol = 10^{-7}$, it would be required 20 iterations (around 4000 function evaluations). As it was said earlier, for the same $tol = 10^{-7}$, our method is able to find both solutions for 608 function evaluations, which is more than six times less than the method proposed in Abdollahi et al. (2013).

Effati-Grosan-1 problem, defined in 5.1.6. and solved by C-VNS in 5.2.2.

- Differential evolution invasive weed optimization algorithm in Zhou et al. (2013) finds a solution with not specified domain, and with $EqTol < 10^{-5}$. In addition, 500 iterations are spent, maximum population size is set to 15, hence, 7500 function evaluations are used. Our method finds one solution (case $a = 2$) with $tol = 10^{-7}$ after 739 function evaluations (see Table 4). This means that we have made more than 10 times less computational efforts.
- Fuzzy adaptive simulated annealing algorithm in Oliveira and Petraglia (2013) obtained one solution in the domain $[0, 10]^2$ with $EqTol \in [9 \cdot 10^{-15}, 7 \cdot 10^{-8}]$ with 5000000 function evaluations. Our method, with $EqTol = 10^{-10}$, found 7 solutions in the domain $[0, 10]^2$.

Effati-Grosan-2 problem. defined in 5.1.7. and solved by C-VNS in 5.2.2.

- Differential evolution invasive weed optimization algorithm in Zhou et al. (2013) found a solution, in the domain not specified, with $EqTol < 10^{-5}$ after 500 iterations, population max. 15 (max. 7500 function evaluations). Our method found a solution with $tol = 10^{-7}$ after 530-2093 function evaluations, depending on a region, as can be seen from Table 4.
- Fuzzy adaptive simulated annealing algorithm in Oliveira and Petraglia (2013) obtained one solution with $EqTol < 10^{-15}$ in the domain $[-10, 10]^2$ with 450000 function evaluations. Our method, in the same domain, found the solution with less stringent $tol = 10^{-7}$ after only 973 function evaluations, as can be seen from Table 4.

Equilibrium Combustion from Table 8. In Floudas et al. (1999), the solution space is defined as $[0.001, 10]^5$, since the variables must be positive by their nature. In this domain, there is only one solution. In the following two papers, the solution space was set to $[-10, 10]^5$, resulting in multiple solutions. This is a good example to demonstrate how large tolerance may result in many false solutions. In the domain $[0.001, 10]^5$, our algorithm (with $p = 1$) detected over 100 solutions for $tol = 10^{-3}$, over 70 solutions for $tol = 10^{-4}$, 8 solutions for $tol = 10^{-4}$, and just one solution for $tol \leq 10^{-5}$.

- Differential evolution invasive weed optimization algorithm in Zhou et al. (2013) found 14 solutions with $EqTol > 10^{-2}$ after 800 iterations, population max. 15. Two of those solutions were with positive coordinates.
- Fuzzy adaptive simulated annealing algorithm in Oliveira and Petraglia (2013) found 7 solutions with $EqTol > 6 \cdot 10^{-3}$ after 1100000 function evaluations; 4 of those solutions were with positive coordinates.

Test Problem 6 from Table 8.

- Imperialist competitive algorithm in Abdollahi et al. (2013) presented a single solution (out of 16) with $EqTol \sim 10^{-15}$ after 1000 iterations with 300 countries (300000 function evaluations). Our algorithm found all 16 solutions with $tol = 10^{-7}$ after 5926615 function evaluations, as can be seen from Table 8.
- Biased random-key genetic algorithm in Silva et al. (2014) with $EqTol > 10^{-3}$ found all 16 solutions for more than 110s. Our algorithm found all 16 solutions with $tol = 10^{-7}$ after $2.7s$, as can be seen from Table 8.
- Particle swarm optimization algorithm in Turgut et al. (2014) presented a single solution (out of 16), $EqTol \sim 10^{-18}$, 219 generations, unknown population size. Our algorithm found all 16 solutions with $tol = 10^{-7}$.

4.9.2 Comparison on other commonly used instances.

In this subsection, we present test results of our method for three commonly used test problems (see Zhou et al. (2013) or Oliveira and Petraglia (2013)).

1) Neurophysiology application. This system of $n = 6$ nonlinear equations has 4 parameters c_i, $i = 1, ..., 4$. In all tests found in literature, only the special case $c_i = 0$, $i = 1, ..., 4$ is considered. In that case, the system has infinite number of solutions in the domain $[-10, 10]^6$ (and in $[-a, a]^6$ for $a \geq 1$).

2) Economics modeling application. This system of n nonlinear equations has $n - 1$ parameters c_i, $i = 1, ..., n - 1$. In all tests found in literature, only the special case $c_i = 0$, $i = 1, ..., n - 1$ is considered. In that case, the system has infinite number of solutions in the domain $[-10, 10]^n$ (and in $[-a, a]^n$ for $a \geq 1$). In all tests, the dimension of the problem was set to $n = 20$.

3) Interval arithmetic benchmark. This is a system of $n = 10$ nonlinear equations. In the domain $[-2, 2]^{10}$, the system has exactly one solution, as shown in Oliveira and Petraglia (2013).

In all tests we used $\rho_0 = 0.01$, $\rho_{\min} = 0.001$, $q_\rho = 0.1$, $a_0 = 1$, $q_a = 0.5$, $p = 1$, $\varphi(x) = 1 - |x|$, and Nelder-Mead local minimizer. Three different tolerances were used: $tol = 10^{-3}$, $tol = 10^{-7}$, $tol = 10^{-15}$. For

the first two problems, we set the maximum of different found solutions to 1, 10, 20, 50, and 100, respectively. For the third problem, this maximum was set to 20, but exactly one solution was found in every run. Each experiment was repeated 30 times, and average times, and number of function evaluations until all listed number of solutions were found, are presented in Table 14.

Table 14: C-VNS results for commonly used test problems

	number of	tol=1e-3		tol=1e-7		tol=1e-15	
Problem name	solutions	time	comp. eff.	time	comp. eff.	time	comp. eff.
1) Neurophysiology	1	$< 1ms$	736	0.001	1680	0.001	4428
application	10	0.005	11434	0.006	18984	0.041	132750
	20	0.022	52904	0.053	123867	0.113	304939
	50	0.197	311185	0.423	682345	1.009	1574310
	100	1.125	1132651	2.559	2538378	3.946	4320766
2) Economics	1	0.001	967	0.002	1893	0.003	3176
modeling application	10	0.011	11472	0.020	21977	0.035	40933
	20	0.027	26323	0.052	51665	0.088	92042
	50	0.137	99320	0.243	178726	0.383	293335
	100	0.558	278508	0.907	461721	1.194	652238
3) Interval arithmetic benchmark	1	0.001	2021	0.003	7754	0.011	27047

As we discussed it earlier, comparison with other methods is difficult since different objective functions are used, and often, there is no data on running time of algorithms, and no report on the number of function evaluations until the solutions are found, etc. Also,it is not clear how many function evaluations are performed in one iteration of population based methods. When the solutions were presented, we calculated the value $EqTol$ as the value of our objective function (3) ($p = 1$) on those solutions. We present test data for these three problems for other methods from literature.

Differential evolution invasive weed optimization (Zhou et al. 2013)

1. 12 solutions, $EqTol > 2 \cdot 10^{-3}$, time=13.11s, 800 iterations, population max. 15.
2. 4 solutions, $EqTol > 10^{-3}$, time=18.14s, 800 iterations, population max. 15.
3. 8 solutions, $EqTol > 10^{-3}$, time=15.05s, 2000 iterations, population max. 15.

Fuzzy adaptive simulated annealing (Oliveira and Petraglia 2013)

1. 8 solutions, $EqTol \in [10^{-17}, 3 \cdot 10^{-7}]$, < 60000 function evaluations.
2. 4 solutions, $EqTol < 10^{-80}$, < 21000 function evaluations.
3. 1 solution, $EqTol < 10^{-15}$, < 50000 function evaluations.

Genetic algorithm (Pourrajabian et al. 2013)

1. 4 solutions presented, but 20 solutions on the graph with $EqTol > 10^{-7}$, Time execution is $> 2s$ for population of 10, $> 10s$ for population of 40.
2. 1 solution, $EqTol > 10^{-8}$, execution time 2-5s depending on the population size from 10 to 100.

Multi-objective optimization evolutionary algorithm (Song et al. 2015)

1. average of 73.1 solutions found with $EqTol > 2 \cdot 10^{-2}$, after 500 generations of population of 100 (50000 function evaluations).
2. average of 20.2 solutions found with $EqTol > 10^{-1}$, after 500 generations of population of 100 (50000 function evaluations).

Many-objective Hype algorithm (Qin et al. 2015)

1. average of 38.2 solutions found, after 500 generations of population of 100 (50000 function evaluations).
2. average of 13.0 solutions found, after 500 generations of population of 100 (50000 function evaluations).

Particle swarm optimization (Turgut et al. 2014)

1. 1 solution, $EqTol < 10^{-16}$, 747 generations, unknown population size.
2. 1 solution, $EqTol < 10^{-16}$, 487 iterations, unknown population size.

Hybrid artificial bee colony algorithm (Jia and He 2012)

1. 1 solution, $EqTol \sim 10^{-8}$, 1000 iteration, population size 50 for ABC and 50 for PSO (100000 function evaluations).

Imperialist competitive algorithm (Abdollahi et al. 2013)

1. 1 solution, $EqTol < 10^{-10}$ for the best solution, 200 iteration with 300 countries (60000 function evaluations), with average $EqTol > 10^{-2}$ for all countries. After 20 iterations (6000 function evaluations) $EqTol > 10^{-5}$ for the best solution.

When comparing the results of presented methods with our results, given in Table 14, it is clear that our C-VNS based heuristic outperforms all presented methods in terms of the number of solutions found, and in terms of execution time spent. Fuzzy adaptive simulated annealing heuristic (Oliveira and Petraglia 2013) finds solutions with small error, but fails to find more solutions for problems [1] and [2]. We found that for $tol = 10^{-15}$, our method requires similar number of function evaluations, taking into account the number of solutions they found. Only in Song et al. (2015), larger number of solutions are found, but with very large tolerance. In Zhou et al. (2013), the large number of false solutions for problem 3) is reported due to the large tolerance.

5 Conclusions

In this paper we propose Continuous variable neighborhood search (C-VNS) based heuristic for finding all solutions of a non-linear system of equations (NSE). As it is usually done, we first transform the system into an optimization problem and propose a new objective function for finding all roots. Theoretical results regarding the accuracy of such a new function are provided. As a consequence, it is shown why some commonly used transformation functions wrongly report more solutions than the system actually has. We then select the transformation function that theoretically produces the most precise solutions. Extensive computational results show that our C-VNS based heuristic has solved successfully and efficiently all test problems where optimal solutions are known; it finds all solutions very fast without producing wrong roots. Comparison of absolute value function and a smooth quadratic transformation function, using our C-VNS heuristic in both cases, confirms theoretical observation that the usual quadratic transformation function may produce a large number of wrong solutions. Moreover, our heuristic outperforms two recently proposed heuristics, based on two other meta-heuristic paradigms, i.e., on C-GRASP and Harmony search. The extensive computational comparative analysis is performed as well, showing that our new C-VNS based heuristic may be considered among the very best methods for solving NSE.

Future work may include application of the new heuristic in solving numerous engineering and manufacturing problems that can be modeled as NSE.

References

Abdollahi M, Bouyer A, Abdollahi D (2016) Improved cuckoo optimization algorithm for solving systems of nonlinear equations. The Journal of Supercomputing 72(3):1246–1269.

Abdollahi M, Isazadeh A, Abdollahi D (2013) Imperialist competitive algorithm for solving systems of nonlinear equations. Computers & Mathematics with Applications 65(12):1894–1908.

Al-Baali M, Spedicato E, Maggioni F (2014) Broyden's quasi-Newton methods for a nonlinear system of equations and unconstrained optimization: a review and open problems. Optimization Methods and Software 29(5):937–954.

Argyros IK (2009) On a class of newton-like methods for solving nonlinear equations. Journal of Computational and Applied Mathematics 228(1):115–122.

Awawdeh F (2010) On new iterative method for solving systems of nonlinear equations. Numerical Algorithms 54(3):395–409.

Buhmiler S, Krejić N, Lužanin Z (2010) Practical quasi-Newton algorithms for singular nonlinear systems. Numerical Algorithms 55(4):481–502.

Carrizosa E, Dražić M, Dražić Z, Mladenović N (2012) Gaussian variable neighborhood search for continuous optimization. Computers & Operations Research 39(9):2206–2213.

Cordero A, Hueso JL, Martínez E, Torregrosa JR (2012) Increasing the convergence order of an iterative method for nonlinear systems. Applied Mathematics Letters 25(12):2369–2374.

Costa LR, Aloise D, Mladenović N (2017) Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. Information Sciences 415:247–253.

Cox D, Little J, O'Shea D (2007) Robotics and automatic geometric theorem proving. Ideals, Varieties, and Algorithms, 265–316 (Springer).

Darvishi M, Barati A (2007) A third-order Newton-type method to solve systems of nonlinear equations. Applied Mathematics and Computation 187(2):630–635.

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

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

Fang L, He G, Hu Y (2009) A new smoothing Newton-type method for second-order cone programming problems. Applied Mathematics and Computation 215(3):1020–1029.

Fischer A, Herrich M, Izmailov AF, Solodov MV (2016) Convergence conditions for Newton-type methods applied to complementarity systems with nonisolated solutions. Computational Optimization and Applications 63(2):425–459.

Floudas CA, Klepeis J, Pardalos PM (1999) Global optimization approaches in protein folding and peptide docking. DIMACS series in discrete mathematics and theoretical computer science 47:141–171.

Floudas CA, Pardalos PM, Adjiman C, Esposito WR, Gümüs ZH, Harding ST, Klepeis JL, Meyer CA, Schweiger CA (2013) Handbook of test problems in local and global optimization (Springer Science & Business Media).

Gonçalves K, Aloise D, Souza X, Mladenović N (2018) Less is more: Simplified Nelder-Mead method for large unconstrained optimization. Yugoslav Journal of Operations Research (to appear).

Gong W, Wang Y, Cai Z, Yang S (2017) A weighted biobjective transformation technique for locating multiple optimal solutions of nonlinear equation systems. IEEE Transactions on Evolutionary Computation 21(5):697–713.

González-Lima MD, de Oca FM (2009) A Newton-like method for nonlinear system of equations. Numerical Algorithms 52(3):479.

Hansen P, Mladenović N (2001) Variable neighborhood search: Principles and applications. European journal of operational research 130(3):449–467.

Hansen P, Mladenović N, Todosijević R, Hanafi S (2017) Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization 5(3):423–454.

Henderson N, de Sá Rêgo M, Imbiriba J (2017) Topographical global initialization for finding all solutions of nonlinear systems with constraints. Applied Numerical Mathematics 112:155–166.

Henderson N, Sacco WF, Platt GM (2010) Finding more than one root of nonlinear equations via a polarization technique: an application to double retrograde vaporization. Chemical Engineering Research and Design 88(5):551–561.

Hirsch MJ, Meneses C, Pardalos PM, Resende MG (2007) Global optimization by continuous GRASP. Optimization Letters 1(2):201–212.

Hirsch MJ, Pardalos PM, Resende MG (2009) Solving systems of nonlinear equations with continuous GRASP. Nonlinear Analysis: Real World Applications 10(4):2000–2006.

Jaberipour M, Khorram E, Karimi B (2011) Particle swarm algorithm for solving systems of nonlinear equations. Computers & Mathematics with Applications 62(2):566–576.

Jia R, He D (2012) Hybrid artificial bee colony algorithm for solving nonlinear system of equations. Proceedings of Eighth International Conference on Computational Intelligence and Security (CIS), 2012, 56–60 (IEEE).

Kovačević-Vujčić V, Čangalović M, Dražić M, Mladenović N (2004) VNS-based heuristics for continuous global optimization. Modelling, Computation and Optimization in Information Systems and Management Sciences. Hermes Science Publishing 215–222.

Krejić N, Lužanin Z, Radeka I (2007) Newton-like method for nonlinear banded block diagonal system. Applied mathematics and computation 189(2):1705–1711.

Lafmejani AS, Kalhor A, Masouleh MT (2015) A new development of homotopy continuation method, applied in solving nonlinear kinematic system of equations of parallel mechanisms. Proceedings of 3rd RSI International Conference on Robotics and Mechatronics (ICROM), 2015, 737–742 (IEEE).

Luo YZ, Tang GJ, Zhou LN (2008) Hybrid approach for solving systems of nonlinear equations using chaos optimization and quasi-newton method. Applied Soft Computing 8(2):1068–1073.

Maranas CD, Floudas CA (1995) Finding all solutions of nonlinearly constrained systems of equations. Journal of Global Optimization 7(2):143–182.

Martinez J (1986) Solving systems of nonlinear equations by means of an accelerated successive orthogonal projections method. Journal of computational and applied mathematics 16(2):169–179.

Mhetre PS (2012) Genetic algorithm for linear and nonlinear equation. International Journal of Advanced Engineering Technology 3(2):114–118.

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

Mladenović N, Hansen P (1997) Variable neighborhood search. Computers & operations research 24(11):1097–1100.

Mladenović N, Todosijević R, Urošević D (2016) Less is more: basic variable neighborhood search for minimum differential dispersion problem. Information Sciences 326:160–171.

Mo Y, Liu H, Wang Q (2009) Conjugate direction particle swarm optimization solving systems of nonlinear equations. Computers & Mathematics with Applications 57(11):1877–1882.

Oliveira HA, Petraglia A (2013) Solving nonlinear systems of functional equations with fuzzy adaptive simulated annealing. Applied Soft Computing 13(11):4349–4357.

Paláncz B, Awange JL, Zaletnyik P, Lewis RH (2010) Linear homotopy solution of nonlinear systems of equations in geodesy. Journal of Geodesy 84(1):79–95.

Paláncz B, Zaletnyik P, Awange JL, Grafarend EW (2008) Dixon resultant's solution of systems of geodetic polynomial equations. Journal of Geodesy 82(8):505–511.

Pourjafari E, Mojallali H (2012) Solving nonlinear equations systems with a new approach based on invasive weed optimization algorithm and clustering. Swarm and Evolutionary Computation 4:33–43.

Pourrajabian A, Ebrahimi R, Mirzaei M, Shams M (2013) Applying genetic algorithms for solving nonlinear algebraic equations. Applied Mathematics and Computation 219(24):11483–11494.

Qin S, Zeng S, Dong W, Li X (2015) Nonlinear equation systems solved by many-objective hype. Evolutionary Computation (CEC), 2015 IEEE Congress on, 2691–2696 (IEEE).

Ramadas GC, Fernandes EM, Rocha AMA (2014) Multiple roots of systems of equations by repulsion merit functions. International Conference on Computational Science and Its Applications, 126–139 (Springer).

Ramadas GC, Fernandes EMdG (2013) Solving systems of nonlinear equations by harmony search. Proceedings of the 13th International Conference Computational and Mathematical Methods in Science and Engineering, 1176–1186.

Ramadas GC, Rocha AMA, Fernandes EM (2015) Testing Nelder-Mead based repulsion algorithms for multiple roots of nonlinear systems via a two-level factorial design of experiments. PloS one 10(4):e0121844.

Romero LA, Mason J (2015) Geolocation using TOA, FOA, and altitude information at singular geometries. IEEE Transactions on Aerospace and Electronic Systems 51(2):1069–1078.

Sacco W, Henderson N (2011) Finding all solutions of nonlinear systems using a hybrid metaheuristic with fuzzy clustering means. Applied Soft Computing 11(8):5424–5432.

Sahinidis NV (1996) BARON: A general purpose global optimization software package. Journal of Global Optimization 8(2):201–205.

Sharma JR, Guha RK (2016) Simple yet efficient newton-like method for systems of nonlinear equations. Calcolo 53(3):451–473.

Shoja A, Vahidi A, Babolian E (2017) A spectral iterative method for solving nonlinear singular Volterra integral equations of Abel type. Applied Numerical Mathematics 112:79–90.

Silva RM, Resende MG, Pardalos PM (2014) Finding multiple roots of a box-constrained system of nonlinear equations with a biased random-key genetic algorithm. Journal of Global Optimization 60(2):289–306.

Soares RdP (2013) Finding all real solutions of nonlinear systems of equations with discontinuities by a modified affine arithmetic. Computers & Chemical Engineering 48:48–57.

Song W, Wang Y, Li HX, Cai Z (2015) Locating multiple optimal solutions of nonlinear equation systems based on multiobjective optimization. IEEE Transactions on Evolutionary Computation 19(3):414–431.

Stuber M, Kumar V, Barton P (2010) Nonsmooth exclusion test for finding all solutions of nonlinear equations. BIT Numerical Mathematics 50(4):885–917.

Turgut OE, Turgut MS, Coban MT (2014) Chaotic quantum behaved particle swarm optimization algorithm for solving nonlinear system of equations. Computers & Mathematics with Applications 68(4):508–530.

Wang CL, Meng GY (2013) Parallel multisplitting two-stage iterative methods with general weighting matrices for non-symmetric positive definite systems. Applied Mathematics Letters 26(11):1065–1069.

Zhou Y, Luo Q, Chen H (2013) A novel differential evolution invasive weed optimization algorithm for solving nonlinear equations systems. Journal of Applied Mathematics 2013, Article ID 757391:1–18.