Les Cahiers du GERAD ISSN: 0711–2440

DE-VNS: Self-Adaptive Differential Evolution with Crossover Neighbourhood Search for Continuous Global Optimization

D. Kovačević, N. Mladenović,
B. Petrović, P. Milošević

G–2013–40

June 2013

DE-VNS: Self-Adaptive Differential Evolution with Crossover Neighbourhood Search for Continuous Global Optimization

Darko Kovačević

Faculty of Organizational Sciences
University of Belgrade
Belgrade, Serbia

darko.kovacevic@nbs.rs

Nenad Mladenović

GERAD & School of Mathematics
Brunel University-West London
London, UK

nenad.mladenovic@brunel.ac.uk

Bratislav Petrović
Pavle Milošević

Faculty of Organizational Sciences
University of Belgrade
Belgrade, Serbia

bratislav.petrovic@fon.bg.ac.rs
pavle.milosevic@fon.bg.ac.rs

June 2013

Les Cahiers du GERAD

G–2013–40

Abstract: In this paper, we suggest DE-VNS as a new heuristic that combines two well known metaheuristic approaches: Differential Evolution (DE) and Variable Neighbourhood Search (VNS), which have, in the last few years, attracted a considerable attention both by academics and practitioners. In our hybrid heuristic, the idea of neighbourhood change is used to estimate the crossover parameter of DE. We propose a family of distributions to be used in order to control the distances among solutions in the search space. Our hybrid heuristic has excellent characteristics, and it comes out that it is more favorable than the recent DE approaches when tested on standard and large instances from the literature.

Key Words: Global Optimization, Hybrid heuristics, Differential Evolution, Variable Neighbourhood Search, Parametric Distributions.

1 Introduction

This paper is dealing with multimodal problems occurring when finding global optima in difficult unconstrained nonlinear problems over continuous spaces. Their general form is given below:

$$(\min) f(x), x \in X \subseteq R^n \tag{1}$$

where $f : R^n \to R$ is generally nonlinear, non-convex function defined on R^n, and X is a feasible set.

One of the popular methods used for solving these problems is Differential Evolution (DE) , proposed by Storn and Price [1]. Its simple and straightforward ideas contain three main parts: strategy, crossover and selection. Strategy diversifies the population, which alleviates such problems as the premature convergence; although there are many strategies, only a few may be suitable for solving a particular problem. Crossover determines whether the target or the trial vector v will survive to the next generation, while selection is based on solutions quality. Vesterstroem and Thomsen [2] compared the DE algorithm with Particle Swarm Optimization (PSO) and Evolutionary Algorithms (EAs) on numerical benchmark problems. For the majority of the problems, DE outperformed both PSO and EAs in terms of the solution's quality. Sun et al. [3] proposed a combination of DE and the Estimation of Distribution Algorithm (EDA), where the search toward a promising area uses sampling of new solutions following the distribution function. Liu and Lampinen [4] reported that the effectiveness, efficiency, and robustness of the DE algorithm are sensitive to the settings of the control parameters, namely mutation parameter F and crossover parameter CR. The best settings for the control parameters can be different for different functions and for the same function with different requirements. A "fast EP" (FEP) is proposed by Yao et al. [5], using a Cauchy mutation as the primary search operator (instead of conventionally used Gaussian). Lee and Yao [6] described a further generalization of FEP by using mutation based on the Levy probability distribution; note that, Cauchy probability distribution is a special case of the Levy probability distribution. This way, variations of a single mutation enables discovering a wide region of the search space, wider than the one got by Gaussian distributions; large variations of the mutated offspring can help to escape from local optima. Furthermore, three crucial control parameters involved in DE - population size N, mutation parameter F, and crossover rate CR, may significantly influence the optimization performance of the DE. Different problems usually require different setting for the control parameters. Self-adaptation allows an DE strategy to adapt itself to any general class of problems by reconfiguring [7, 8, 9]. DE algorithm has gained much attention with successful applications in different domains, e.g. [10, 11, 12].

Mladenovic and Hansen [13] presented Variable Neighbourhood Search (VNS) approach, a metaheuristic which does not follow vector trajectories. It explores increasingly distant neighbourhoods of the current best solution. If a better solution is found, VNS jumps from the current solution to the new one; there is a variety of hybrids, where VNS is combined with other optimizers. Mladenovic et al. [14] have presented the idea of using several geometric neighbourhood structures and random distributions in the shaking step. That idea led to the Glob-VNS, which turned out to be noticeably more efficient than the variants with fixed geometry and fixed distribution. Instead of using a sequence of neighbourhoods and shaking by sampling, Carrizosa et al. [15] define a sequence of shaking distribution derived from Gaussian n-variate distribution centered at x. This optimizer is called Gauss-VNS. Yang et al. [16] sugest hybridization of DE with neighbourhood search. The resulting algorithm, known as NSDE, performs mutation by adding a normally distributed random value to each target vector.

In this paper, we propose a self-adaptive algorithm based on the DE, which incorporates the features of VNS approach. Particular attention is devoted to the mechanism of self-adaptation of crossover parameter CR, based on systematic change of neighbourhoods.

The paper is organized as follows. In the next section, we give a short review of conventional DE. Section 3 describes the proposed DE-VNS. In Section 4, we present computational results demonstrating the performance of our DE-VNS in comparison with the state-of-the-art DE variants. In the final section, we give conclusions and guidelines for the future work.

2 Differential Evolution

In this section, we briefly summarize the basic procedure of Differential Evolution (DE), and introduce the notation and the terminology.

DE is a parallel direct search method which utilizes D-dimensional parameter vectors as a population for each generation. The initial vector population is chosen randomly and should cover the entire parameter space. After initialization, DE enters a loop of evolutionary operations: mutation, crossover, and selection.

2.1 Strategy

Strategies in DE enable the algorithm to explore the search space and maintain the diversity. There are two basic strategies used in differential evolution heuristics:

- The *"DE/rand/1/bin"* strategy usually demonstrates slow convergence speed and bears stronger exploration capability. Therefore, this strategy is suitable for solving multimodal problems and performs better than the strategies relying on the best solution found so far. Target vector for this strategy is generated in this way:

$$v_i = x_{r_1^i} + F \cdot \left(x_{r_2^i} - x_{r_3^i} \right) \tag{2}$$

- The *"DE/best/1/bin"* strategy is a degenerated case of the *"DE/rand-to-best/1/bin"* strategy with λ equal to 0. It usually has the fast convergence speed and performs well when solving unimodal problems. However, it is more likely to get stuck at a local optimum and thereby, leads to a premature convergence when solving multimodal problems. This mutation strategy has the following form:

$$v_i = x_{best} + F \cdot \left(x_{r_1^i} - x_{r_2^i} \right) \tag{3}$$

2.2 Crossover

After applying a chosen strategy, the next phase is crossover. The undetermined offspring y_i is generated by the crossover operation on v_i as:

$$y_i\left(g+1\right) = \begin{cases} v_i, & \text{if } \left(rand\left(0,1\right) \leq CR\right) \text{ or } \left(j + j_{rand}\right) \\ y_i\left(g\right), & \text{otherwise} \end{cases} \tag{4}$$

where j_{rand} is randomly chosen index to ensure that the trail vector y_i does not duplicate v_i, $rand(0,1)$ represents a uniform random value between 0 and 1, $CR \in (0,1)$ is a crossover rate, and $g = 1,\ldots,n$ is a generation. Presented crossover phase is the version used in basic DE algorithms.

2.3 Selection

The task of selection phase is to determine better offspring and parent vector. The better vector survives into the next generation, and the worse vector is discarded. The selection operation can be expressed as follows:

$$x_{r_1^i}\left(g+1\right) = \begin{cases} y_i\left(g+1\right), & \text{if } f\left(y_i\left(g+1\right)\right) < f\left(x_{r_1^i}\right) \\ x_{r_1^i}, & \text{otherwise} \end{cases} \tag{5}$$

where $x_{r_1^i}\left(g+1\right)$ is the offspring of $x_{r_1^i}$ in the next generation $g = 1,\ldots,n$.

These 3 steps are repeated until some specific termination criteria are satisfied. Termination criteria can be: reaching the value of success threshold, exceeding the number of generations, maximum CPU time used in the search, etc.

3 DE-VNS for Continuous Global Optimization

Variable Neighbourhood Search is a metaheuristic based on a principle of systematic change of neighbourhoods within the search; VNS and its extensions are characterized by the simplicity of the basic scheme. The search in the different neighbourhoods is performed in a deterministic, stochastic or combined way. VNS with deterministic rules is called Variable Neighbourhood Descent (VND), Reduced VNS is VNS that uses stochastic rules. Basic VNS is the third variant that combines a random selection of a point in shaking or perturbation step followed by a deterministic local search from that point. Inspired by VNS main idea, we present a hybrid system based on the effective DE algorithms that we extend with the concept of variable neighbourhood perturbation parameter vector. The main ideas of the proposed DE-VNS algorithm are elucidated as follows.

3.1 Strategy

Numerous approaches have been proposed in order to find an adequate measure of the selection between *"DE/rand/1/bin"* and *"DE/best/1/bin"* strategies. It should enable global search of the solution space on one hand, and rapid convergence of the strategy, on the other. In this paper, we will introduce the modification of *"DE/rand/1/bin"* strategy that will comprise the characteristics of *"DE/rand/1/bin"*, named *"DE/order-random/1/bin"* (see details in [17] for different strategies within DE). The main idea is that within the *"DE/rand/1/bin"*, we will try to find a local *"DE/best/1/bin"* strategy as shown:

$$v_i = x_{r_1^i} + F \cdot \left(x_{r_2^i} - x_{r_3^i} \right) \tag{6}$$

where $x_{r_1^i}, x_{r_2^i}, x_{r_3^i}$ are randomly selected at each step:

$$f\left(x_{r_1^i}\right) < f\left(x_{r_2^i}\right) \text{and} f\left(x_{r_1^i}\right) < f\left(x_{r_3^i}\right) \tag{7}$$

3.2 Estimation of the Mutation Parameter F

In the conventional realizations of DE, the choice of three control parameters: the population size N, the mutation parameter F, and the crossover parameter CR largely determines the outcome of the optimization. There is a great number of empirical results e.g. [18] that help us to choose the parameter values. However, there is not a strict set of parameters related to the broader group of practical problems. Higher values of the population size N allow greater diversification in the solution space and therefore, a better exploration of the solution space. On the other hand, large population causes slower convergence, which is reflected in the speed of the algorithm when solving large scale optimization problems. Note that if the values of the mutation parameter F are larger, the larger portion of the solution space is explored. The problem of selecting the mutation parameter N is solved by introducing roulette methods [19]. Roulette methods gradually give a higher probability of drawing the values of mutation parameter proven to be successful in previous iterations.

The parameter F is estimated by introducing the competition. Assume that we have H different values for the parameter F; choose one among them at random with the probability $p_h, h = 1, \ldots, H$. The probabilities can be adjusted according to the success rate of the preceding steps of searching the solution space. The h^{th} value of F is successful if it generates a trial point y such that $f(y) < f(x_i)$. When n_h is the current number of the $f(y)$ setting successes, the probability p_h can be calculated as the relative frequency as:

$$p_h = \frac{n_h + n_0}{\sum_{i=1}^{H} (n_i + n_0)} = \frac{n_h + n_0}{n_0 \cdot H + \sum_{i=1}^{H} n_i} \tag{8}$$

where $n_0 > 0$ is a constant. Setting $n_0 > 1$ prevents a dramatic change in p_h by the use of theh^{th} parameter settings. If any of the probabilities drops below a given threshold $\delta > 0$ and $\delta < 1$, current values of p_h are reset to their starting values $p_h = 1/H$. Thus the premature convergence of probabilities p_h is avoided. The competition structure prefers the parameters related to the success in such a way that an algorithm is self-adapted each time when the better solution is found over probabilities p_h for that settings.

The degree of similarity between offspring and trail vector changes as it depend on the values of crossover parameter CR. We define the degree of similarity of these two vectors as follows. The nearest dimensional surroundings of the offspring are defined with small values of the parameter CR, so the changes in vectors are of small number dimensions, as it is illustrated on the additive problem at Fig. 1.

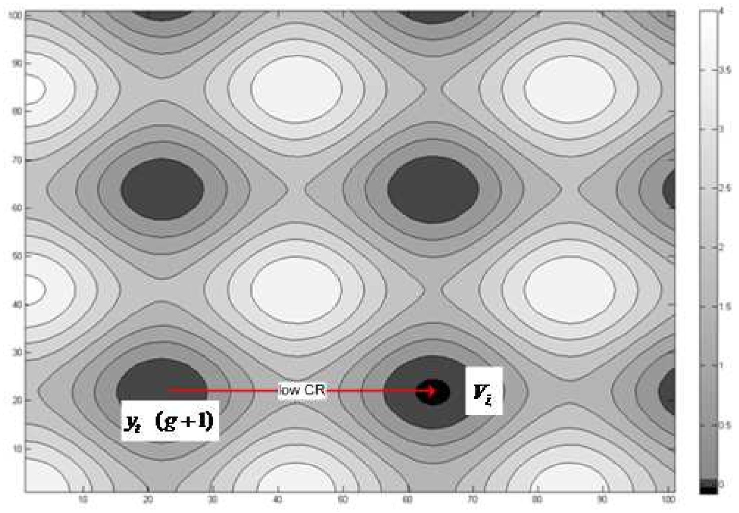


Figure 1: Additive multimodal problem

For the higher values of crossover parameter CR, we get further dimensional surroundings, and the offspring vector can change by more dimensions at once. Each approach has advantages and disadvantages. If crossover parameter CR is small, the algorithm may not be able to perform such a jump which could find a better solution by changing the small number of dimensions, as it is shown in Fig. 2.

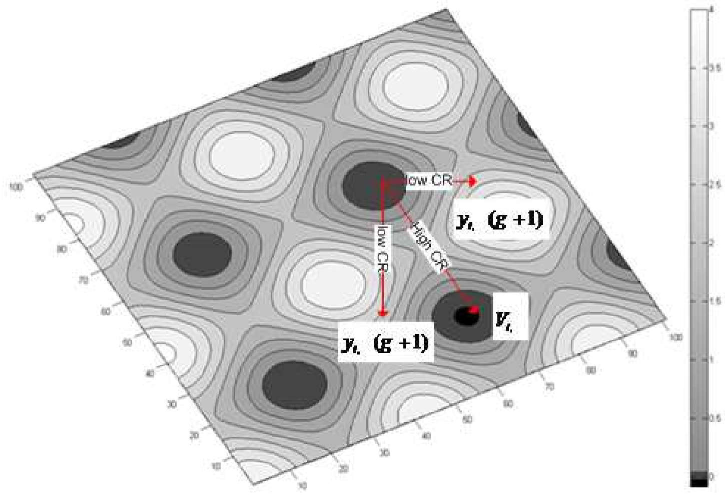


Figure 2: Rotated multimodal problem

In the example of the rotated ellipse, Salomon [20] explains why CR factor should not have low values during the whole course of the optimization. For large values of CR, the entire population could converge too fast and remain trapped in a local optimum. This is the reason to start our parameter from the closer neighbourhoods to the offspring vector, so the population would not converge too fast. On the other hand, if our algorithm is found in local optima or is on the correct path to the global optima, we want higher values of CR for two reasons:

- To provide that the algorithm can escape from local optima by several dimensions, allowing an effective jump to any point over the solution space;
- To speed up the convergence if the algorithm is located in the vicinity of global optimum.

We will rely on the choice of probability distributions from which we want to choose the parameter CR. We are not defining a clear boundary, in the sense of VND, among dimensional distances of an offspring vector, just the expected value of dimensions. Therefore, we will form a family of parametric distributions that will allow us to effectively select expected number of dimensions from the offspring neighbourhoods.

3.3 Estimation of the Crossover Parameter CR with Variable Neighbourhood Search

This section deals with the method of selecting a crossover parameter value. Numerous studies show that it is difficult to choose this parameter, because different values of CR correspond to different sets of problems. We introduce a family of adaptive distributions that depends on crossover neighbourhood parameter - neighbourhood factor par. The CR parameter is than stochastically chosen with respect to these distributions. The Variable Neighbourhood Search is included in solving this problem in the following manner. When the algorithm finds a child vector better than the parent, in the next iterations, it is required that a neighbourhood factor CR reduces its value. That ensures that the crossover parameter will be at the closest neighbourhoods; as the entire population would not converge too quickly so, a better search through the solution space is achieved. Therefore, the value of the neighbourhood factor CR will gradually increase, in the iterations in which the algorithm cannot find a more satisfactory solution, by using the *stepfactor* which will allow further dimensional environments around the parent vector to be investigated. In the case of finding a favourable child vector, algorithm resets the CR value based on improvement of fitness function value. This property of parameters allows the adaptation of the algorithm to the given problem, so there is no need for parameterization of the algorithm by the user.

Analysing the distributions from which CR derives values

The analysis is started by observing the distributions from which crossover parameter CR derives its values, and it considers problems divided into three main classes:

1. Additive multimodal problems : Schwefel, Ackley, Griewank, Rastrigin, Molecular potential energy (MPE), Michalewicz's function and Six-hump;
2. Unimodal problems : Sphere, Rosenbrock and De Jong's variations;
3. Noncontinuous problems: step function, molecule or sphere packing [21].

Each group of the problems was repeated 100 times using the algorithm *"DE/rand/1/bin"* with CR and F parameters, as a random process with a uniform distribution. The values of CR and F were recorded in the case when fitness function value was better and the algorithm achieved the predefined tolerance. Moving windows are defined so that the bins contain the recorded values of the parameters CR in the first iterations, midd iterations of the algorithm, as well as the final part of the algorithm, respectively, in case when the algorithm succeed to achieve a global optimum. Empirical probability distribution functions of defined classes are presented in Fig. 3.

The goal of our analysis is to make a family of parametric distributions that mimic the distributions presented at Figs. 3 and 4. To find out which parametric distribution best reflect the empirical distribution, we fitted a number of distributions such as: beta, Weilbull, exponential, gamma, normal, inverse gaussian, Pareto, nakagami, etc. Moving windows are defined as 1000 successful iterations i.e. iterations in which the better child vector is found. Distribution from such defined moving window with best estimated most likelihood function is recorded. The array of recorded distributions is then divided by quintiles: 0%–25% 25%–50%, 50%–75%, 75%–100%, which effectively describes the various distributions of the path to the global optimum. Based on maximum likelihood estimation, a set of 6 distributions that best describes the distribution of crossover parameter CR is obtained as it can be seen on Figs. 5–8.

Allocation of fitted distributions along intervals and classes of problems is shown at Figs. 5, 6 and 7 respectively.

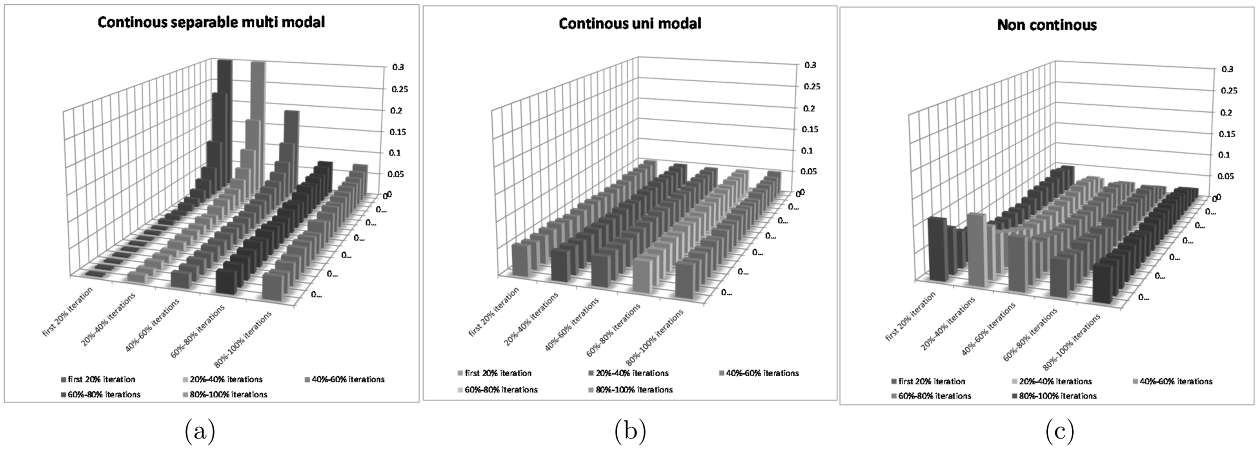


Figure 3: Empirical probability distribution functions for (a) Continuous separable multimodal (b) Continuous unimodal (c) Noncontinuous

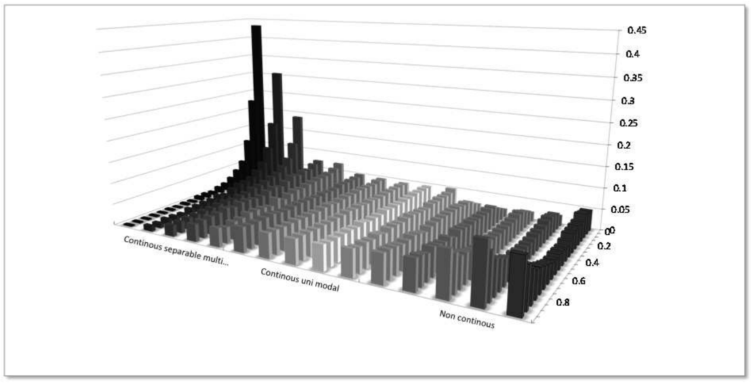


Figure 4: Empirical probability distribution functions for all defined classes

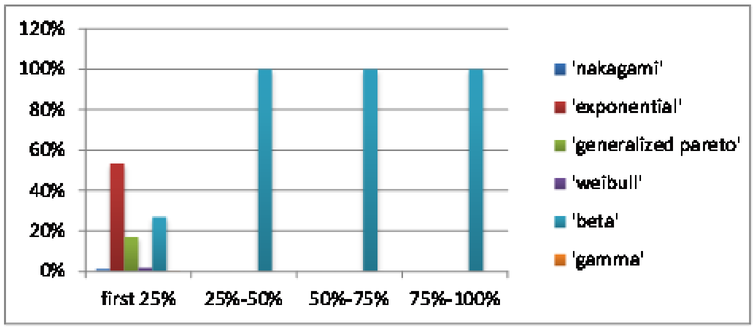


Figure 5: The portion of fitted parametric probability distribution functions for continuous separable multimodal problems

Beta Distribution

According to these figures, in the final iteration of the algorithm, CR parameter values are taken from the beta distribution in nearly 100% of cases, while in the first quintile, in addition to the beta distribution, the exponential distribution occurs in a large percentage. Bearing in mind that the exponential distribution is a special case of the beta distribution, the beta distributions are considered as the major distribution of parameters CR in the further analysis. It should be noted that the structure of the distribution changes depends on the applied strategy. In our work, the usage of the beta distribution is related to *"DE/rand/1/bin"*

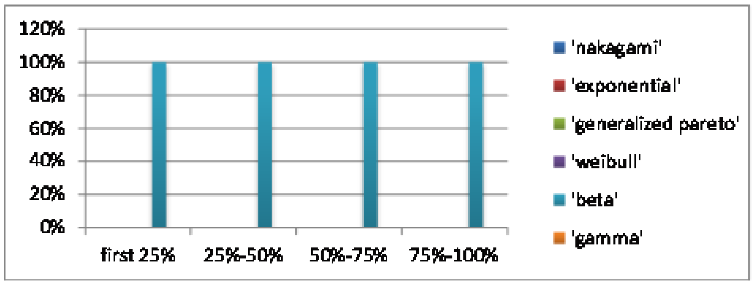


Figure 6: The portion of fitted parametric probability distribution functions for continuous unimodal problems

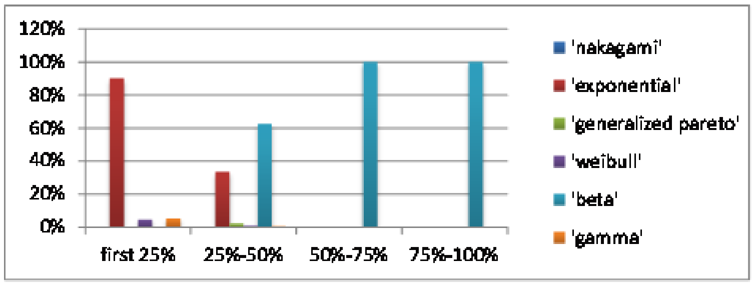


Figure 7: Portion of fitted parametric probability distribution functions for non-continuous problems

strategy. Beta distribution is defined as follows:

$$f\left(x,\alpha,\beta\right)=\frac{x^{\alpha-1}\left(1-x\right)^{\beta-1}}{\int_0^1 u^{\alpha-1}\left(1-u\right)^{\beta-1}du}=\frac{1}{B\left(\alpha,\beta\right)}x^{\alpha-1}\left(1-x\right)^{\beta-1} \tag{9}$$

As it is shown at Fig. 3. (a), algorithm achieved the best results starting from a beta distribution which has the feature favouring small values of CR for the continuous multimodal problem. In this way, detailed search over the solution space is achieved. That capacity, consistent with the multimodal definition of a problem, is archived for well-defined values of α,β:

$$\alpha=1,\beta>1 \tag{10}$$

The final steps in all defined distributions become uniform which is a special case of beta distribution for values:

$$\alpha=1,\beta=1 \tag{11}$$

For the unimodal class of problems, during the search, the distribution takes the successful values that are uniform all the way. As it can be seen, there is no favouring of certain values of crossover parameter because there is no problem multimodality, so there is no problem of trapping in local optima.

The best results for the beta distribution of the non-continuous class of problems is obtained by distributions that sample the entire domain of CR. In the final phase, distribution is slightly shifting to the value of one i.e. distribution favours high values of CR. The non-continuous algorithm seeks the best leap towards better solutions by increasing number of dimensions in order to skip a discontinuity in the solution space. In this case, the parameters of the distribution are:

$$\alpha>1,\beta=1 \tag{12}$$

It can be seen that the distributions are not completely symmetrical for the small and large values of CR. For the probability of distributions that favours large values of CR, the distribution itself does not converge to zero for small values of CR i.e. the distribution tries to preserve lower level of CR at some level.

Two-sided power distribution

The beta distribution has difficulties concerned with its maximum likelihood parameter estimation of two parameters, whose parameters do not have a clear-cut meaning. Johnson [22] and Johnson and Kotz [23] dealt with neglected applications of triangular distributions as an alternative to the beta distribution. Johnson used triangular distributions as a proxy for the beta distribution, specifically in problems of assessment of risk and uncertainty, such as the project evaluation and review technique (PERT). The parameters of a triangular distribution have a one-to-one correspondence with an optimistic estimate a, most likely estimate m and pessimistic estimate b of a quantity under consideration, providing to the triangular distribution its intuitive appeal (see, for example Williams [24]). Similarly to the beta distribution, the triangular distribution can be positively or negatively skewed (or symmetrical) but must remain unimodal. It is pointed out that there is no triangular distribution which would reasonably approximate uniform, J-shaped or U-shaped distributions.

In this paper we investigate an extension of the three-parameter triangular distribution, so called the two-sided power (TSP) distribution [25], as a proven alternative to the beta distribution. The four-parameter distribution proposed herein does allow J-shaped and U-shaped forms.

The cumulative distribution function of $TSP(a, m, b, par)$ is:

$$F\left(x|a,m,b,par\right) = \begin{cases} \frac{m-a}{b-a}\left(\frac{x-a}{m-a}\right)^{\frac{1}{par}}, a < x < m \\ 1 - \frac{b-m}{b-a}\left(\frac{b-x}{b-m}\right)^{\frac{1}{par}}, m < x < b \end{cases} \quad (13)$$

TSP family of distributions depending on the parameter par with$a = 0$, $b = 1$ and $m = 0$ is shown at Fig. 8.

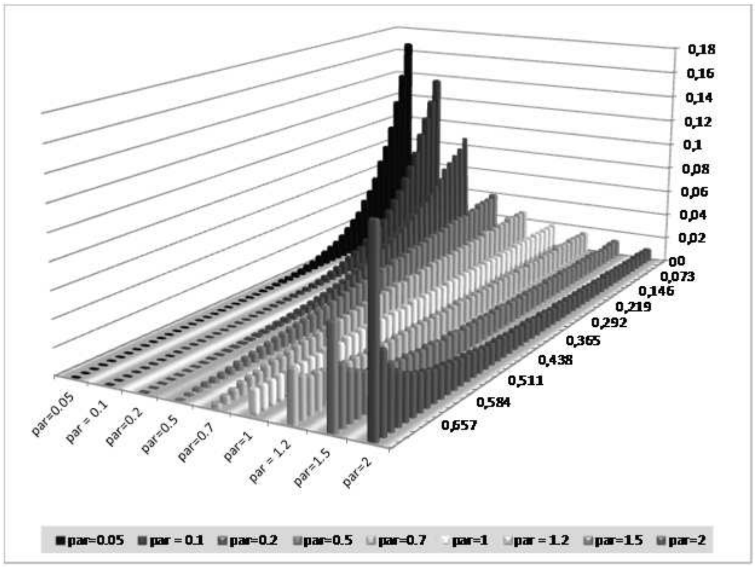


Figure 8: Family of TSP distribution over parameter par

3.4 Evaluation of Neighbourhood Search Parameter

Based on estimation of parameters m and using the procedure discussed in Appendix 1, it may be assumed that $\hat{r}\left(a,b\right) = 1; \hat{m}\left(a,b\right) = X_{(1)} = a$ and

$$p\hat{a}r\left(a,b\right) = -\frac{\log\left[M\left\{a,b,\hat{r}(a,b)\right\}\right]}{s} \quad (14)$$

We applied the estimation Neighbourhood search parameters par on a set of problems – global continuous problems. Using this analysis, we will try to determine the extent of movement parameters par. The maximum likelihood estimates are obtained by using the same recorded values of CR, when the algorithm achieved the predefined tolerance, as mentioned before. In Fig. 9, the estimated value of the neighbourhood factor par and the limits of movement of this parameter are shown.

For some problems (Schwefel, Rastrigin, MPE) parameter neighbourhood factor par, in the initial iterations, starts from low values. This indicates that crossover has a smaller number of dimensions or, in closer dimensional environments, greater exploration of the solution space in the initial phase. In the final iterations of the algorithm, larger values of the neighbourhood factor par are preferred. This indicates that the algorithm is in the vicinity of the global optimum, which allows crossover between the offsprings by several dimensions at once. In some problems (Griewank, Ackley, Rosenbrock), it is evident that the algorithm moves from the higher values of the parameters par, indicating that the used strategy *"DE/rand/1/bin"* has no problem of local optima trapping. In all cases, in the final iterations of the algorithm, values of the neighbourhood factor par tends to 0.7.

Parameters a and b are defined by the limits of parameter CR i.e. $a = CRmin = 0$, $b = CRmax = 1$. The analysis of shape for continuous multimodal distributions suggests zero as the value of parameter $m - m = 0$. For the continues multimodal problem, the algorithm achieves the best results starting from the TSP distribution, with preferable small values of crossover parameter CR or a detailed search over the solution space, which is consistent with the multimodal definition of the problem, i.e. for $par \to 0$. On the other hand, in the final iterations, all CR values are equally favoured, leading to values of the parameter $par \to 0.7$.

3.5 DE-VNS Pseudo-Code

Finally, we present our DE-VNS pseudo-code of Algorithm 1.

Algorithm 1 DE-VNS pseudo-code

```
 1: Randomly initialize a population of $N$ individuals $P_g = \{X_1, \ldots, X_N\}, i = 1, \ldots, N$
 2: Evaluate the population
 3: Setting initial roulette probability for $F$
 4: WHILE stopping criteria
 5:     Set $k \leftarrow 1$
 6:     WHILE until $k > N$
 7:         Calculate roulette probability for $F$ parameter $p_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(n_j + n_0)}$
 8:         Sampling $CR$ from adaptive beta distribution as:
            $$CR = F\left(x|a,m,b,par\right) = \begin{cases} \frac{m-a}{b-a}\left(\frac{x-a}{m-a}\right)^{\frac{1}{par}}, a < x < m \\ 1 - \frac{b-m}{b-a}\left(\frac{b-x}{b-m}\right)^{\frac{1}{par}}, m < x < b \end{cases}$$
 9:         Applying strategy "DE/Rand-Local-Best/1/bin" with obtained $F$ and $CR$ parameters
10:         Evaluate child vector $f(y^k_{child})$
11:         IF $f(y^k_{child}) \leq f(y^k_{parent})$ then
12:             $par^k_{new} = \max(par_{\min}, par^k_{old} - (f(y^k_{child}) - f(y^k_{parent})))$
13:             $parent = child$
14:         ELSE
15:             $par^k_{new} = par^k_{old} + stepfactor$
16:             $par^k_{new} = \min(par^k_{new}, par_{\max})$
17:         END IF
18:         Set $k \leftarrow k + 1$
19:     END WHILE
20: END WHILE
21: Stopping criteria: $x^*$ is an approximate solution of the problem
```

The loop starts after the first three steps used for initialization and evaluation of the population, and for setting initial roulette probability of mutation parameter; roulette probability of mutation parameter is calculated, and crossover parameter value is sampled from TSP distribution in each iteration. The next steps are: applying strategy, and evaluating child vector. If the child vector is better than the parent vector, neighbourhood factor par value is reset, based on improvement of fitness function value. If child vector is not better, par value is increased by using $stepfactor$.

4 Computational results

Experimental settings for the benchmarks and DE heuristics used for comparison are explained in this section so, and the numerical results are given.

4.1 Test functions

Seven common benchmark functions are used for the numerical experiments. The problems are selected from [20]. We shall focus on multimodal problems, but unimodal problems will be also included in the analysis, in order to demonstrate the robustness of the proposed approach on a wide class of problems.

Schwefel function

Schwefel's function is a deceptive one in the sense that the global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong direction.

$$F_1 = \sum_{i=1}^{n} \left[-x_i sin\left(\sqrt{|x_i|}\right) \right] \tag{15}$$

with $-500 \leq x_i \leq 500$ and $\min F_1(420.9687, 420.9687, \ldots, 420.9687) = -418.9829 \cdot n$.

Ackley function

Ackley's is a widely used multimodal test function. This function has an exponential term that covers its surface with numerous local minima. The complexity of Ackley function is moderate. In order to obtain good results for this function, the search strategy must combine the exploratory and exploitative components efficiently.

$$F_2 = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=0}^{n} x_i^2}\right) - exp\left(\sqrt{\frac{1}{n}}\sum_{i=0}^{n} cos\left(2\pi x_i\right)\right) + 20 \tag{16}$$

with $-32 \leq x_i \leq 32$ and $\min F_2(0, 0, \ldots, 0) = 0$.

Griewank function

The Griewank's function is similar to the function of Rastrigin's. It has many widespread local minima regularly distributed over the solution space.

$$F_3 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{17}$$

with $-600 \leq x_i \leq 600$ and $\min F_3(0, 0, \ldots, 0) = 0$.

Rastrigin function

The Rastrigin's function is based on the function of De Jong with the addition of cosine modulation in order to produce frequent local minima. Thus, the test function is highly multimodal. However, the locations of the minima are regularly distributed.

$$F_4 = 10\,n + \sum_{i=1}^{n} \left(x_i^2 - 10\,cos\,(2\pi x_i)\right) \tag{18}$$

with $-5.12 \leq x_i \leq 5.12$ and $\min F_4(0, 0, \ldots, 0) = 0$.

Molecular potential energy (MPE) function

MPE function [26] is a functional form similar to general potential energy functions, whose global minimum is known. The number of local minima of this function increases exponentially with the size of the problem.

$$F_5 = \sum_{i=1}^{n} \left(1 + cos\,(3x_i) + \frac{(-1)^i}{\sqrt{10.60099896 - 4.141720682 \cdot cos\,(x_i)}}\right) \tag{19}$$

with $0 \leq x_i \leq 5$ and $\min F_5(0, 0, \ldots, 0) = -0.0411183034 \cdot n$.

Rosenbrock function

The Rosenbrock's valley is a classic optimization problem, also known as banana function or the second function of De Jong. The global optimum lies inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult and hence, this problem has been frequently used to test the performance of optimization algorithms. The function has the following definition:

$$F_6 = \sum_{i=1}^{n-1} \left[100 \cdot \left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right] \tag{20}$$

with $-5 \leq x_i \leq 5$ and $\min F_6(0, 0, \ldots, 0) = 0$.

Sphere function

The sum of different powers is a commonly used unimodal test function. Sphere function is a simple and strongly convex function used in the theory of evolutionary strategies. It has the following definition:

$$F_7 = \sum_{i=1}^{n} x_i^2 \tag{21}$$

with $-1 \leq x_i \leq 1$ and $\min F_7(0, 0, \ldots, 0) = 0$.

4.2 DE based heuristics used in comparison

As it is previously mentioned, the performance of the conventional DE algorithm highly depends on the chosen trial vector generation strategy and associated parameter values used. In literature [27], the parameter adaptation techniques are divided into three categories: deterministic, adaptive, and self-adaptive control rules. Deterministic rules modify the parameters according to certain predetermined rationales without utilizing any feedback from the search process. Adaptive rules incorporate some form of the feedback from the search procedure to guide the parameter adaptation. Self-adaptive rules directly encode parameters into the individuals and evolve them together with the encoded solutions. In addition to the DE-VNS approach, which is proposed in this paper, we analyse the problem of global optimization, and compare our algorithm with different variations of DE methods. Two common versions of DE without parameter adaptation are taken into consideration: *"DE/best/1/bin"*(Debest) and *"DE/rand/1/bin"*. Three self-adapted algorithms with distinguished characteristics are also chosen: SaDE, JADE, and CoDE.

DE based on "DE/rand/1/bin" strategy

Since it is well known that *"DE/rand/1/bin"* version of the strategy has good searching possibility of the solution space and therefore, better performance in solving multimodal global optimization problem, we use one variant of this strategy for the comparison. Used values of control parameters are $F = 0.5$ and $CR = 0.3$, which proved to be promising strategy according to [28].

Debest

Although the main focus of this paper is global multimodal problems, *"DE/best/1/bin"* with defined control parameters $F = 0.5$ and $CR = 0.3$ is taken in consideration because this algorithm is well-known and is commonly used for comparison. The values of F and CR are chosen based on 'good behaviour' of *"DE/rand/1/bin"* at these values of control parameters.

SaDE

Self-adaptive DE (SaDE) algorithm [28] is one of the state of the art algorithms that avoids the expensive computational costs on searching the most appropriate trial vector generation strategy, as well as its associated parameter values by a trial-and-error procedure. Instead of employing the computationally expensive trial-and-error search for the most suitable strategy and its parameter values, we maintain a strategy candidate pool including several effective trial vector generation strategies with effective, yet diverse characteristics. During evolution, one strategy will be chosen from the candidate pool and applied to perform the mutation operation. The choice is made according to probability learned from the previous experience in generating promising solutions. Four trial vector generation strategies are included into the strategy candidate pool: *"DE/rand/1/bin"*, *"DE/rand-to-best/2/bin"*, *"DE/rand/1/bin"*, and *"DE/rand-to-ran/1/bin"*. In the SaDE algorithm, N is left as a user-specified parameter because it highly replies on the complexity of a given problem. The parameter F is approximated by a normal distribution with mean value 0.5 and standard deviation 0.3. This setting enables us to maintain both exploitation and exploration power throughout the evolution process. Parameter CR has normal distribution $N\left(CR_m, 0.1\right)$, where CR_m is initialized as 0.5. To adapt CR to the proper values, the authors update CR_m every 25 generations, based on the recorded successful CR values since the last CR_m update.

JADE

Zhang and Sanderson [29] have presented their DE algorithm named JADE. The main contribution of JADE is the implementation of a new mutation strategy *"DE/current-to-pbest"*. This strategy has an optional external archive and updating control parameters in an adaptive manner. *"DE/current-to-pbest"* strategy is a generalization of the classic strategy *"DE/current-to-best"*. The archive is initiated to be empty. After each generation, the parent solutions that fail in the selection process are added to the archive. If the archive size exceeds a certain threshold, then some solutions are randomly removed from it. The role of the archive is to provide information about the progress direction and to improve the diversity of the population. Crossover probability CR_i is randomly taken from a normal distribution of mean μ_{CR} and standard deviation 0.1, and then truncated to [0, 1]. μ_{CR} is initialized to be 0.5 and then updated at the end of each generation as:

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot mean_A\left(CR_{succ}\right) \tag{22}$$

where c is a constant between 0 and 1, $mean_A()$ is the arithmetic mean and CR_{succ} is a set of all successful crossover probabilities CR_i. The mutation factor F_i is randomly taken from a Cauchy distribution with location parameter μ_F and scale parameter 0.1, and then truncated to be 1 if $F_i \geq 1$ or regenerated if $F_i \leq 0$. The location parameter μ_F is initialized to be 0.5 and then updated at the end of each generation as:

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot mean_L\left(F_{succ}\right) \tag{23}$$

where $mean_L()$ is the Lehmer mean and F_{succ} is the set of all successful mutation factors.

CoDE

Wang et al. [30] proposed novel method based on Differential Evolution, called composite DE (CoDE). This method uses three trial vector generation strategies and three control parameter settings and randomly combines them to generate trial vectors. The strategy candidate pool is consisted of the following strategies: *"DE/rand/1/bin"*, *"DE/rand/2/bin"*, and *"DE/current-to-rand/2/bin"*. The three control parameter settings are: $F = 1.0$, $CR = 0.1$, $F = 1.0$, $CR = 0.9$, and $F = 0.8$, $CR = 0.2$. In each generation, each of these trial vector generation strategies is used to create a new trial vector with a control parameter setting randomly chosen from the parameter candidate pool. Thus, three trial vectors are generated for each target vector and are compared in the next step. The best trial vector enters the next generation if it is better than its target vector.

For all DE variants used in this study, the maximum number of parameter evaluation *eval max* and population *pop* were set as it is shown in Table 1.

Table 1: Values of *evalmax* and *pop* parameters for all DE variants

Dimension	*evalmax*	*pop*
10	2.00E+05	34
20	2e+51e+6	44
30	1.50E+06	50
50	5e+52.5e+6	80
100	1.00E+06	100

Because of the DE-VNS parameters characteristic mentioned before, the following values for the maximum and minimum value of *par* were used – $parmin = 0$ and $parmax = 0.7$. We have used an arbitrary step defined as:

$$stepfactor = \frac{1}{10 \cdot D \cdot \log_2(D)} \tag{24}$$

Parameters a,b, and m were defined as follows – $a = 0$, $b = 1$, $m = 0$. For competitive settings we use $F = 0.4$, 0.6, 0.8 and 1 with defined $n_0 = 2$; $\delta = 0.05$.

Experiments are performed on Pentium dual core computer with 2GB of memory, using MATLAB.

4.3 Comparison

In analysing the behaviour of the proposed algorithm, we focus on the unconstrained continuous multimodal global optimization problems. To show the robustness of the proposed algorithm, we use the number of dimensions (D) from 10 to 100, so to cover that problem with the lower dimensions and large scale problem, as well. For the predefined tolerance around the global optimum, a value of 1e-6 is used. The other stopping condition is the maximum number of function evaluations, that is 50000*D for Rosenbrock's function and 10000*D for other functions. Each problem is repeated 25 times in order to obtain credible data.

Number of function evaluations (FEs) is one of the metrics that we are interested in. In addition to FEs metrics, we are also interested in success rate (SR), or the percentage of successfully achieved optima in the predefined tolerance.

In Tables 2 to 8, the results of the test are shown. Gray areas indicate that for a given problem, the tolerance around the global optimum is not reached in all 100% of cases. Tables also display the basic statistics for function evaluations: minimum FEs number (*eval min*), average FEs number (*eval avg*) and maximum FEs number (*eval max*). Another statistic that we are interested in is named $fmin_D$ and it presents tolerance, in cases where tolerance around the global optima is achieved, or the difference between the average cost function and known global optima in other cases. The best results are marked with bold fields, provided that the metaheuristic achieved SR percentage of 100%. The last column in all of these tables is an overall score (OS) for each of selected algorithms. This variable is calculated as:

$$OS = \log\left(\sum_{D} \frac{evalavg}{D} fmin_D\right), \begin{cases} D = 10, 20, 30, 50, & \text{for Rosenbrock} \\ D = 10, 20, 50, 100, & \text{in other cases} \end{cases} \tag{25}$$

The overall score is the logarithm of the sum of elements that characterize optimization for the selected dimensional problem. The average number of function evaluations *eval avg* is divided with the number of dimensions, because it is important that both small and large dimensional problems have the same importance. Not reaching the global minimum is penalized by $fmin_D$. Success rate is not included in the formula, so there is no double counting. Optimization is considered better if OS value is less. The logarithm is used for scaling the results. Algorithms are ranked using this criterion function in each of these tables.

As it is previously mentioned, DE-VNS is developed for multimodal problems whereon it achieves the best results. On hard functions, like MPE, Schwefel and Rastrigin, our algorithm has the lowest number of FEs for all dimensions. That is a notable result, because DE-VNS is compared with state of the art algorithms for lower dimension problems (DE self-adapted algorithms). Also, DE-VNS performs excellent on large scale problems. For maximum dimensional problems, on 6 of 7 test functions (Rosenbrock's and all of multimodal functions) it has shown to be the fastest algorithm. That result can be indication for future work - to test DE-VNS at huge scale problems, up to 10000 D, and to compare it with algorithms that are adjusted for large scale optimization. On unimodal functions JADE, it outperforms all the selected algorithms, except on 50 dimensional Rosenbrock's function. According to OS, DE-VNS demonstrates best results on 5 of 7 selected problems.

It should be emphasized that our DE-VNS is the only algorithm that has SR 100% on all test problems (on all test functions, for all dimensions). CoDE algorithm is the second best in terms of robustness, and it does not reach 100% success rate in 5 of 28 problems.

In Table 9, all algorithms taken into consideration are ranked for each test function. The last column, *all*, is the arithmetic mean of rankings. That value for DE-VNS is 1.43, JADE performed well at all test functions, and its arithmetic mean of rankings is 2.86. Overall performances of SaDE, *"DE/rand/1/bin"* and CoDE are similar. Debest algorithm has a characteristic that converges prematurely at multimodal problems, and it is the last, as can be seen in Fig. 10.

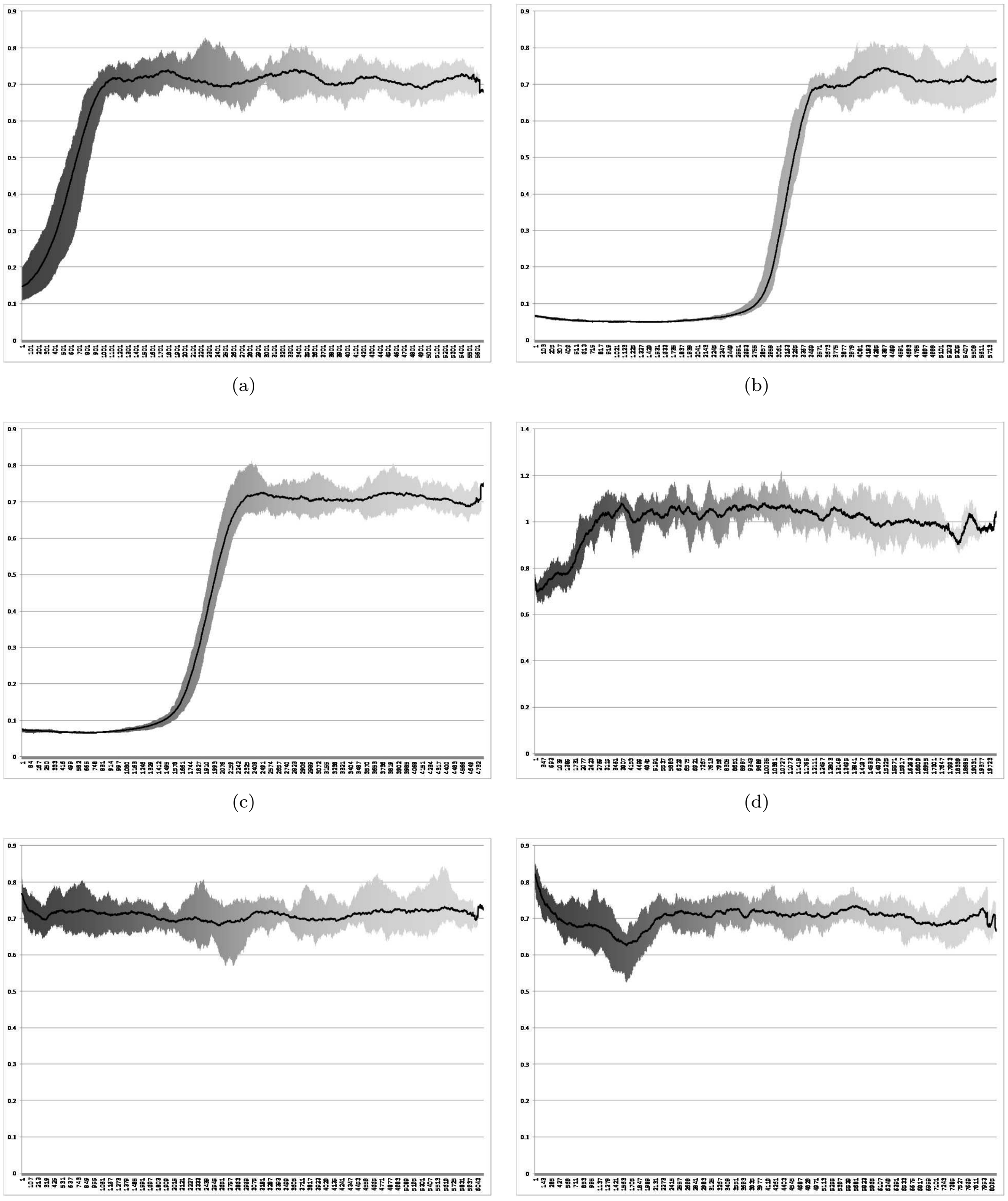
For each function, we compare the speed of convergence for the four methods that have proved to be the best- our DE-VNS, JADE, SaDE, and CoDE. We compare the results for 50 and 100 dimensions. For the comparison, we used the convergence path that is closest to the mean number of function evaluations. Data are presented in Fig. 11.

Figures for Schwefel's function indicate that DE-VNS and SaDE converge fast toward the solution. For Rastrigin, DE-VNS and JADE converge rapidly. On these problems, CoDE algorithm has slow convergence rate. On the other hand, CoDE algorithm almost always converged toward the solution but its convergence speed differs from very slow at Schwefel and Rastrigin, to much faster, especially at 100 D Greiwenk and 50 D Rosenbrock.

DE-VNS have the greatest convergence rate for all multimodal problems, except Greiwenk. JADE algorithm is faster for 50 D Griewank, but it does not converge for 100 D. At 100 dimensional problem, CoDE and DE-VNS have similar speed. Only our algorithm converges toward the solution for every chosen test function.

5 Conclusion

In this paper, we presented a new approach for solving the continuous global optimization problem. Our method, called DE-VNS, is based on application of Variable neighbourhood search (VNS) heuristic within Differential evolution (DE) heuristic. The change of neighbourhoods is used for automatic estimation of the crossover parameter CR of DE. Numerical results show very good performance of the proposed algorithm. For example, the success rate on all instances tested was 100%; the computational efforts (the number of function evaluations before the global optimum is reached) are consistently smaller than those obtained by the other methods. Our comparative study was concentrated on comparison with other DE based heuristics. However, future work may consist in comparison of our hybrid DE-VNS heuristic with other state-of-the-art heuristics.



(a) (b) (c) (d) (e) (f)

Figure 9: Estimated neighbourhood parameter *par* for a) Schwefel's; b) Rastrigin's; c) MPE; d) Rosenbrock's; e) Griewank's; f) Ackley's

Table 2: Results for Schwefel's function

		10					20					50					100					Overall
		eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	score
1	DE-VNS	**7,901**	**8,901**	**10,101**	**100%**	**1.00E-06**	**21,201**	**23,431**	**27,301**	**100%**	**1.00E-06**	**70,401**	**79,081**	**89,101**	**100%**	**1.00E-06**	**172,201**	**189,791**	**210,001**	**100%**	**1.00E-06**	-2.2564
2	SaDE	12,001	12,101	13,001	100%	1.00E-06	25,001	27,601	29,001	100%	1.00E-06	84,001	87,501	93,001	100%	1.00E-06	220,001	233,901	249,001	100%	1.00E-06	-2.17528
3	DE/Rand/1; F=0.5; CR=0.3	9,001	9,441	9,901	100%	1.00E-06	23,601	24,601	25,601	100%	1.00E-06	94,301	101,161	105,301	100%	1.00E-06	350,501	385,181	408,701	100%	1.00E-06	-2.09425
4	JaDE	9,001	56,401	100,000	50%	71.1	31,001	150,401	201,001	30%	94.8	102,001	263,901	500,000	60%	59.2	223,001	615,601	1,000,000	50%	107.2	6.319376
5	CoDE	25,001	25,801	27,001	100%	1.00E-06	76,001	80,101	85,001	100%	1.00E-06	338,001	356,601	376,001	100%	1.00E-06	1,000,000	1,000,000	1,000,000	0%	3.24E+03	7.51098
6	Debest	6,601	53,771	100,000	50%	82.9	38,601	77,391	200,000	80%	23.68	500,000	500,000	500,000	0%	1,193,073	1,000,000	1,000,000	1,000,000	0%	21044.87	10.08428

Table 3: Results for Ackley's function

		10					20					50					100					Overall
		eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	score
1	DE-VNS	9,401	10,441	13,501	100%	1.00E-06	22,701	26,241	30,001	100%	1.00E-06	**78,101**	**86,231**	**98,901**	**100%**	**1.00E-06**	**182,801**	**204,521**	**237,601**	**100%**	**1.00E-06**	-2.212824
2	DE/Rand/1; F=0.5; CR=0.3	11,401	11,771	12,101	100%	1.00E-06	27,501	28,211	29,001	100%	1.00E-06	94,301	96,011	97,501	100%	1.00E-06	293,201	297,981	304,001	100%	1.00E-06	-2.125653
3	CoDE	30,001	30,501	31,001	100%	1.00E-06	80,001	83,101	86,001	100%	1.00E-06	228,001	239,601	248,001	100%	1.00E-06	444,001	457,901	470,001	100%	1.00E-06	-1.780516
4	JaDE	**9,001**	**9,801**	**10,001**	**100%**	**1.00E-06**	**15,001**	**16,201**	**18,001**	**100%**	**1.00E-06**	30,001	78,001	500,000	90%	8.70E-02	1,000,000	1,000,000	1,000,000	0%	1.53E+00	4.188527
5	SaDE	9,001	9,901	10,001	100%	1.00E-06	19,001	20,601	21,001	100%	1.00E-06	500,000	500,000	500,000	0%	1.25E+00	1,000,000	1,000,000	1,000,000	0%	2.55E+00	4.579784
6	Debest	8,701	9,411	10,001	100%	1.00E-06	53,801	55,211	58,001	100%	1.00E-06	500,000	500,000	500,000	0%	6.34E-02	1,000,000	1,000,000	1,000,000	0%	1.20E+01	5.082484

Table 4: Results for Griewank's function

		10					20					50					100					Overall
		eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	score
1	DE/Rand/1; F=0.5; CR=0.3	**16,101**	**17,501**	**20,001**	**100%**	**1.00E-06**	**21,301**	**23,971**	**26,901**	**100%**	**1.00E-06**	**68,401**	**71,001**	**74,001**	**100%**	**1.00E-06**	212,601	216,321	218,501	100%	1.00E-06	-2.184962
2	DE-VNS	11,801	22,101	47,401	100%	1.00E-06	22,701	24,961	28,101	100%	1.00E-06	67,401	73,821	80,201	100%	1.00E-06	**158,801**	**168,961**	**180,701**	**100%**	**1.00E-06**	-2.178868
3	CoDE	57,001	63,201	69,001	100%	1.00E-06	63,001	79,801	104,001	100%	1.00E-06	165,001	173,501	186,001	100%	1.00E-06	320,001	330,301	339,001	100%	1.00E-06	-1.767431
4	JaDE	19,001	32,801	101,001	100%	1.00E-06	**12,001**	**17,501**	**30,001**	**100%**	**1.00E-06**	**22,001**	**22,801**	**27,001**	**100%**	**1.00E-06**	43,001	332,001	1,000,000	70%	9.30E-03	1.489687
5	SaDE	16,001	38,201	100,000	80%	2.46E-03	13,001	70,901	200,000	70%	3.60E-03	40,001	226,301	500,000	60%	1.20E-02	100,000	195,401	1,000,000	90%	1.72E-03	1.902181
6	Debest	29,601	51,121	73,301	100%	1.00E-06	41,801	69,971	90,301	100%	1.00E-06	500,000	500,000	500,000	0%	3.89E-01	1,000,000	1,000,000	1,000,000	0%	4.11E+02	6.614082

Table 5: Results for Rastrigin's function

		10					20					50					100					Overall
		eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	score
1	DE-VNS	**10,001**	**10,691**	**11,401**	**100%**	**1.00E-06**	**25,001**	**27,811**	**31,101**	**100%**	**1.00E-06**	**85,001**	**89,741**	**95,601**	**100%**	**1.00E-06**	**200,101**	**220,541**	**243,701**	**100%**	**1.00E-06**	-2.18978
2	JaDE	13,001	13,901	15,001	100%	1.00E-06	33,001	34,301	35,001	100%	1.00E-06	108,001	111,201	114,001	100%	1.00E-06	231,001	247,801	257,001	100%	1.00E-06	-2.10751
3	SaDE	13,001	14,901	16,001	100%	1.00E-06	34,001	37,201	42,001	100%	1.00E-06	125,001	206,401	500,000	80%	1.98E-01	1,000,000	1,000,000	1,000,000	0%	3.28E+00	4.526611
4	CoDE	31,001	34,001	36,001	100%	1.00E-06	117,001	124,901	132,001	100%	1.00E-06	500,000	500,000	500,000	0%	3.04E+01	1,000,000	1,000,000	1,000,000	0%	2.12E+02	6.385088
5	DE/Rand/1; F=0.5; CR=0.3	11,601	13,151	14,201	100%	1.00E-06	73,601	78,551	86,601	100%	1.00E-06	500,000	500,000	500,000	0%	1.43E+02	1,000,000	1,000,000	1,000,000	0%	5.58E+02	6.84565
6	Debest	21,701	25,001	28,301	100%	1.00E-06	200,000	200,000	200,000	0%	4.41E+01	500,000	500,000	500,000	0%	3.59E+02	1,000,000	1,000,000	1,000,000	0%	1.07E+03	7.167273

Table 6: Results for MPE function

		10					20					50					100					Overall
		eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	score
1	DE-VNS	**6,101**	**6,831**	**7,601**	**100%**	**1.00E-06**	**16,801**	**19,121**	**20,901**	**100%**	**1.00E-06**	**52,701**	**59,221**	**66,701**	**100%**	**1.00E-06**	**122,901**	**143,831**	**160,701**	**100%**	**1.00E-06**	-2.3703988
2	JaDE	10,001	11,301	13,001	100%	1.00E-06	27,001	97,601	200,000	60%	8.22E-02	89,001	131,901	500,000	80%	8.23E-02	190,001	520,901	1,000,000	60%	3.22E-02	2.8954117
3	SaDE	11,001	11,701	12,001	100%	1.00E-06	25,001	43,501	200,000	90%	8.17E-03	88,001	460,601	500,000	10%	8.20E-02	1,000,000	1,000,000	1,000,000	0%	3.04E-01	3.5812847
4	CoDE	21,001	22,901	24,001	100%	1.00E-06	69,001	73,101	77,001	100%	1.00E-06	372,001	390,201	402,001	100%	1.00E-06	1,000,000	1,000,000	1,000,000	0%	4.23E+00	4.6263405
5	DE/Rand/1; F=0.5; CR=0.3	45,001	49,761	51,601	100%	1.00E-06	200,000	200,000	200,000	0%	6.74E-02	500,000	500,000	500,000	0%	6.36E+00	1,000,000	1,000,000	1,000,000	0%	3.37E+01	5.6038823
6	Debest	8,501	16,261	23,401	100%	1.00E-06	200,000	200,000	200,000	0%	1.56E+00	500,000	500,000	500,000	0%	1.75E+01	1,000,000	1,000,000	1,000,000	0%	5.27E+01	5.8556373

Table 7: Results for Rosenbrock's function

		10					20					30					50					Overall
		eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	score
1	DE-VNS	45,001	53,901	69,001	100%	1.00E-06	183,001	234,301	415,001	100%	1.00E-06	281,001	424,301	516,001	100%	1.00E-06	**729,001**	**1,142,001**	**1,442,001**	**100%**	**1.00E-06**	-1.26689
2	CoDE	52,001	55,501	59,001	100%	1.00E-06	193,001	202,301	218,001	100%	1.00E-06	410,001	426,401	446,001	100%	1.00E-06	853,001	1,167,601	2,500,000	85%	3.99E-01	3.969299
3	JaDE	**15,001**	**33,801**	**101,001**	**100%**	**1.00E-06**	**38,001**	**43,301**	**52,001**	**100%**	**1.00E-06**	**70,001**	**79,401**	**96,001**	**100%**	**1.00E-06**	157,001	645,901	2,500,000	80%	7.91E-01	4.009373
4	SaDE	263,001	434,401	501,001	100%	1.00E-06	1,000,000	1,000,000	1,000,000	0%	8.00E-01	1,500,000	1,500,000	1,500,000	0%	2.18E+00	2,500,000	2,500,000	2,500,000	0%	9.48E+00	5.794648
5	DE/Rand/1; F=0.5; CR=0.3	455,001	495,201	500,000	20%	5.67E-02	1,000,000	1,000,000	1,000,000	0%	1.98E+00	1,500,000	1,500,000	1,500,000	0%	6.25E+00	2,500,000	2,500,000	2,500,000	0%	1.81E+01	6.120213
6	Debest	104,001	112,401	125,001	100%	1.00E-06	551,001	572,601	598,001	100%	1.00E-06	1,500,000	1,500,000	1,500,000	0%	1.16E-04	2,500,000	2,500,000	2,500,000	0%	2.71E+01	6.132454

Table 8: Results for Sphere function

		10					20					50					100					Overall
		eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	eval-min	eval- avg	eval max	SR	fmin	score
1	JaDE	**6,001**	**6,301**	**7,001**	**100%**	**1.00E-06**	**11,001**	**11,101**	**12,001**	**100%**	**1.00E-06**	**21,001**	**22,001**	**23,001**	**100%**	**1.00E-06**	**44,001**	**45,401**	**48,001**	**100%**	**1.00E-06**	-2.682108
2	SaDE	7,001	7,001	7,001	100%	1.00E-06	13,001	14,201	16,001	100%	1.00E-06	40,001	43,301	46,001	100%	1.00E-06	104,001	110,701	117,001	100%	1.00E-06	-2.470675
3	DE-VNS	8,001	8,201	9,001	100%	1.00E-06	19,001	21,001	22,001	100%	1.00E-06	64,001	72,101	83,001	100%	1.00E-06	153,001	165,001	189,001	100%	1.00E-06	-2.304327
4	DE/Rand/1; F=0.5; CR=0.3	8,001	8,501	9,001	100%	1.00E-06	20,001	20,401	21,001	100%	1.00E-06	70,001	70,801	71,001	100%	1.00E-06	224,001	226,901	230,001	100%	1.00E-06	-2.255302
5	CoDE	19,001	20,701	21,001	100%	1.00E-06	57,001	58,301	60,001	100%	1.00E-06	167,001	172,101	178,001	100%	1.00E-06	331,001	338,101	344,001	100%	1.00E-06	-1.927817
6	Debest	24,001	25,501	27,001	100%	1.00E-06	109,001	114,601	119,001	100%	1.00E-06	500,000	500,000	500,000	0%	5.30E-02	1,000,000	1,000,000	1,000,000	0%	4.22E+02	4.49302

Table 9: Rankings of compared metaheuristics based on Eq. 25

		f_1	f_2	f_3	f_4	f_5	f_6	f_7	*all*
1	DE-VNS	1	1	2	1	1	1	3	1.4286
2	SaDE	2	5	5	3	3	4	2	3.4286
3	DE/Rand/1; F=0.5; CR=0.3	3	2	1	5	5	5	4	3.5714
4	JaDE	4	4	4	2	2	3	1	2.8571
5	CoDE	5	3	3	4	4	2	5	3.7143
6	Debest	6	6	6	6	6	6	6	6

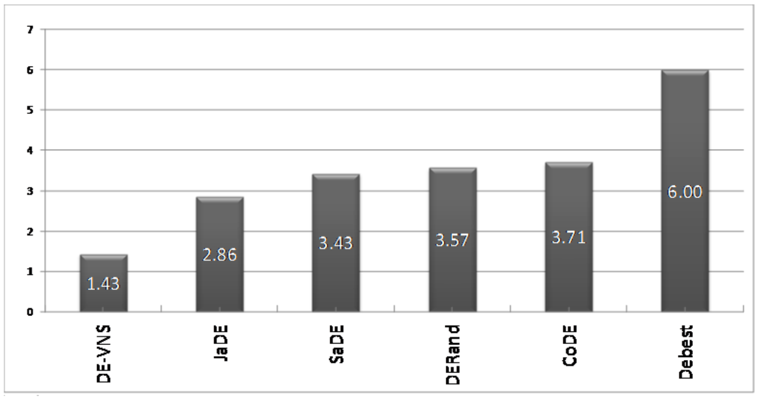
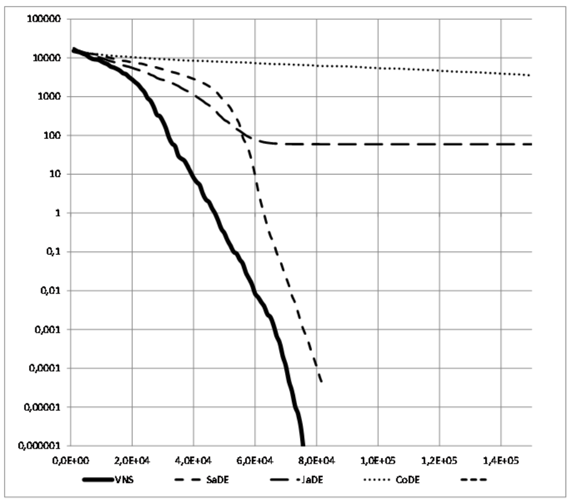
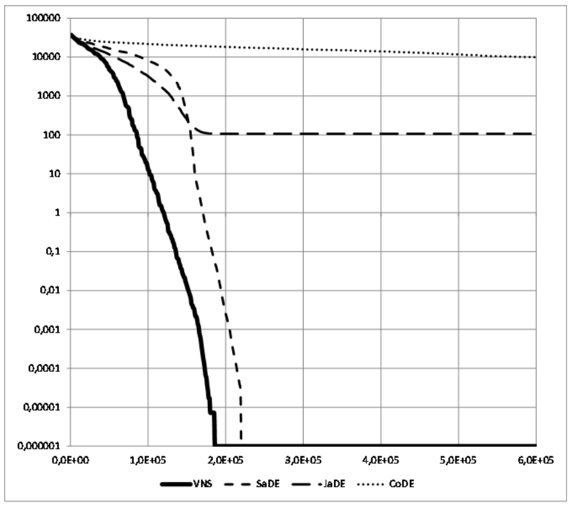


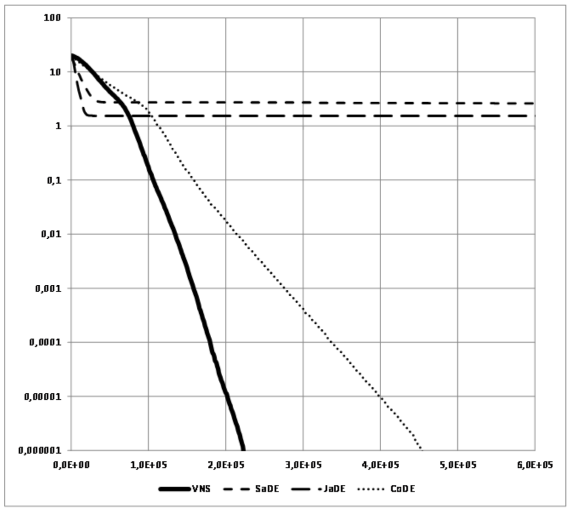
Figure 10: Average rankings of compared metaheuristics



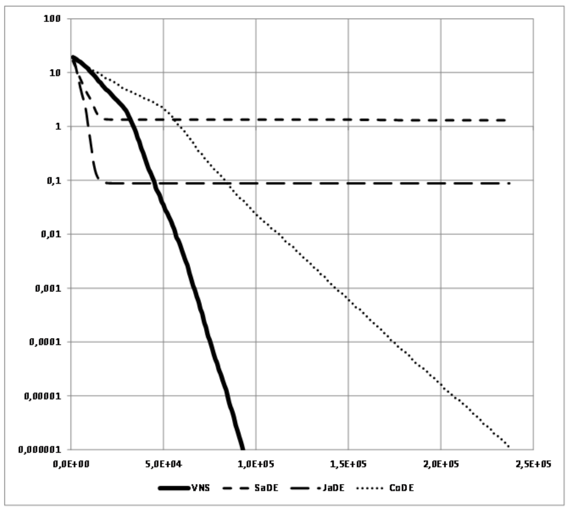
(a)



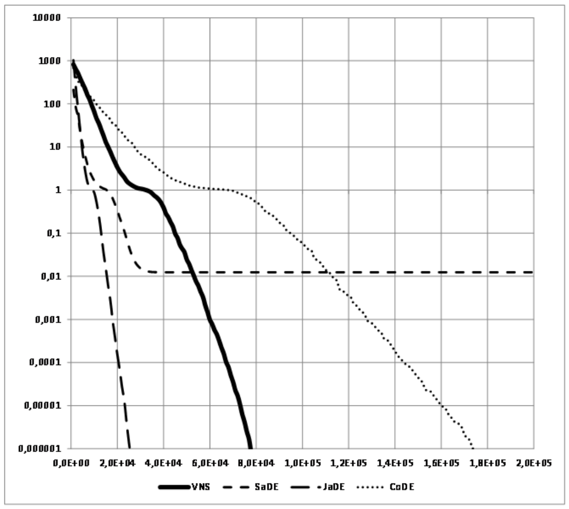
(b)


VNS
SaDE
JaDE
CoDE

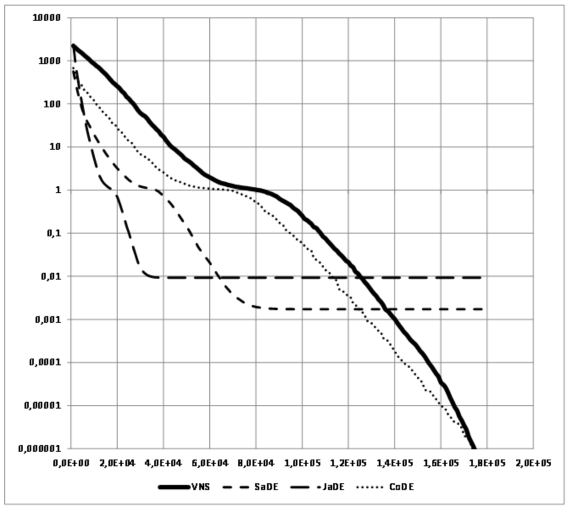
(c)


VNS
SaDE
JaDE
CoDE

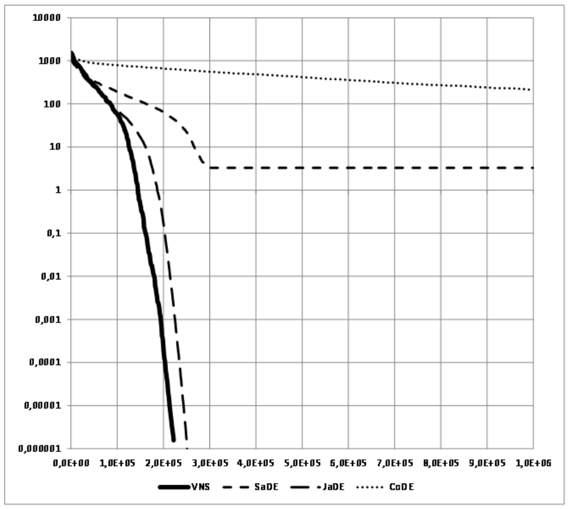
(d)


VNS
SaDE
JaDE
CoDE

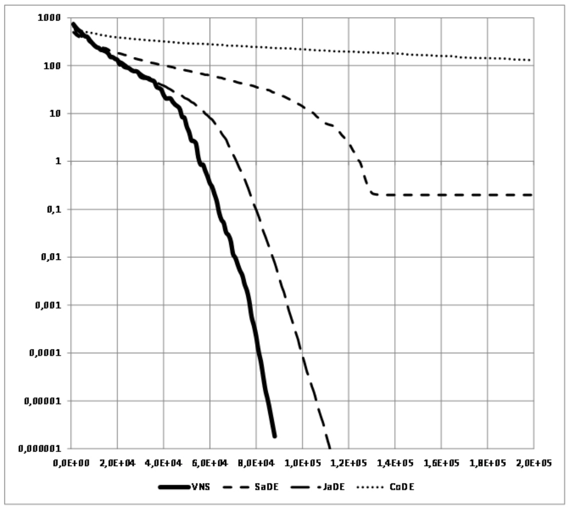
(e)


VNS
SaDE
JaDE
CoDE

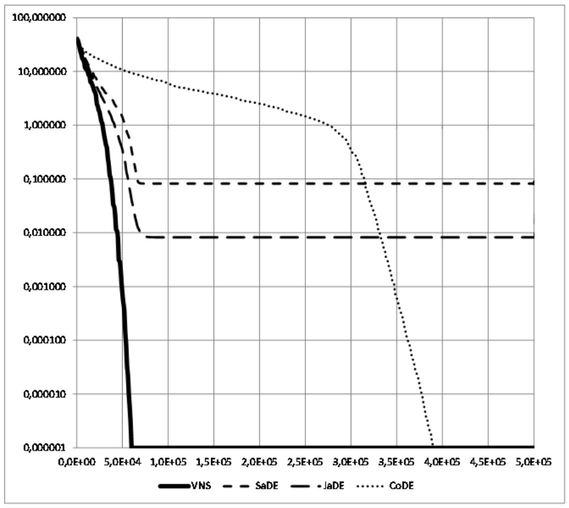
(f)


VNS
SaDE
JaDE
CoDE

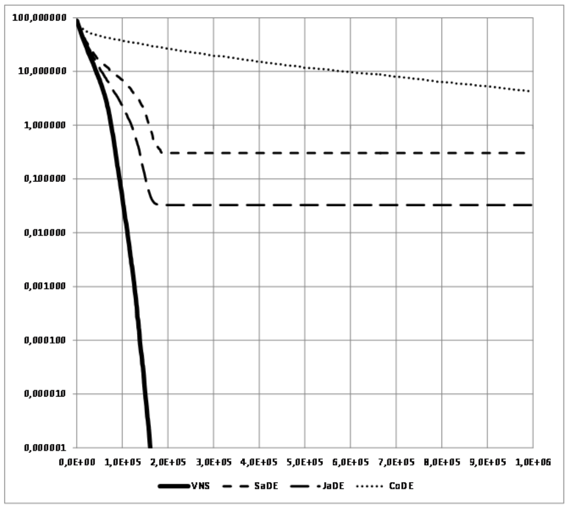
(g)


VNS
SaDE
JaDE
CoDE

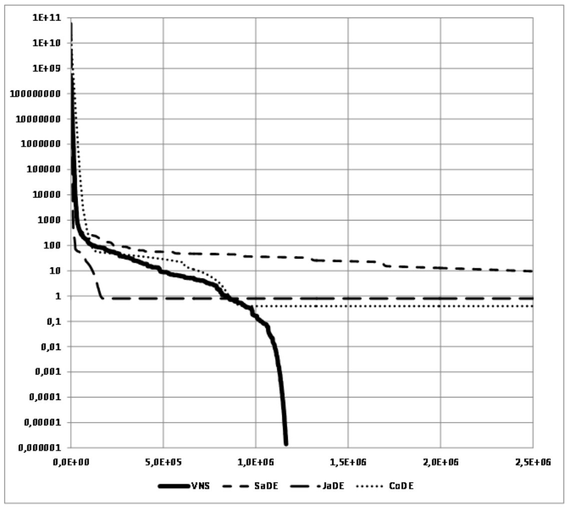
(h)



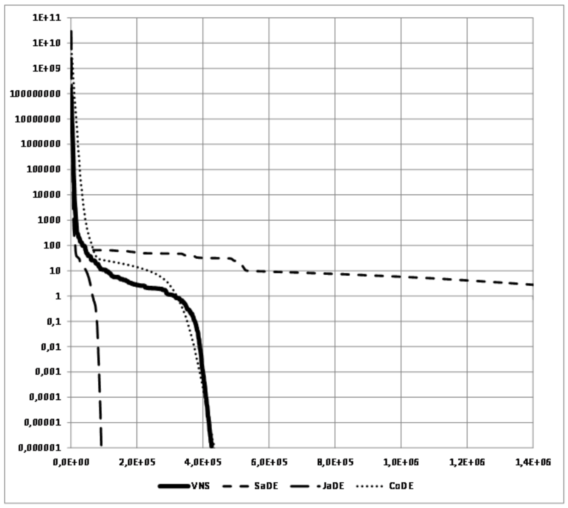
(i)



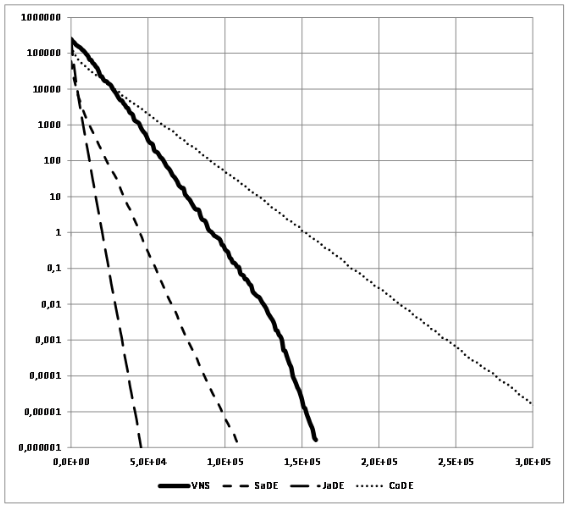
(j)



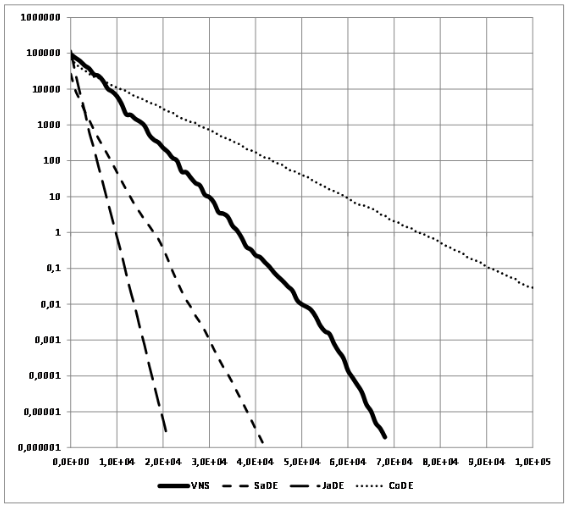
(k)



(l)



(m)



(n)

Figure 11: Convergence for a) Schwefel 50D; b) Schwefel 100 D; c) Ackley 50D; d) Ackley 100 D; e) Greiwank 50D; f) Greiwank 100 D; g) Rastrigin 50D; h) Rastrigin 100 D; i) MPE 50D; j) MPE 100 D; k) Rosenbrock 30D; l) Rosenbrock 50 D; m) Sphere 50D; n) Sphere 100 D

Appendix

A Maximum likelihood estimation for TSP distribution

Let X be a random variable with probability density function given by:

$$f\left(x|a,m,b,par\right)=\begin{cases}\frac{(1/par)}{b-a}\left(\frac{x-a}{m-a}\right)^{\frac{1-par}{par}}, & a<x<m\\ \frac{(1/par)}{b-a}\left(\frac{b-x}{b-m}\right)^{\frac{1-par}{par}}, & m<x<b\end{cases} \tag{A.1}$$

The random variable X is said to follow a TSP distribution $F(a,b,m,par)$ where $a\le m\le b$, $par>0$ and a correspondent cumulative distribution function can be expressed as:

$$F\left(x|a,m,b,par\right)=\begin{cases}\frac{m-a}{b-a}\left(\frac{x-a}{m-a}\right)^{\frac{1}{par}}, & a<x<m\\ 1-\frac{b-m}{b-a}\left(\frac{b-x}{b-m}\right)^{\frac{1}{par}}, & m<x<b\end{cases} \tag{A.2}$$

For a random sample $X=(x_1,x_2,\ldots,x_s)$ with size s from TSP distribution, let the order statistics be $x_{(1)}<x_{(2)}<\ldots<x_{(s)}$. Utilizing expression (26), the likelihood for X is by definition:

$$L\left(X;a,m,b,par\right)=\left(\frac{1}{par(b-a)}\right)^s H\left(X;a,m,b\right)^{\frac{1}{par}-1} \tag{A.3}$$

where

$$H\left(X;a,m,b\right)=\frac{\prod_{i=1}^{r}\left(x_{(i)}-a\right)\prod_{i=r+1}^{s}\left(b-x_{(i)}\right)}{\left(m-a\right)^r\left(b-m\right)^{s-r}} \tag{A.4}$$

$X_{(0)}=a, X_{(s)}=b$ and r is implicitly defined by $X_{(r)}\le m\le X_{(r+1)}$. From (28) follows that two-parameter MLE procedure maximizing equation as a function of m and par (with a and b fixed) is two stage optimization problem, namely we may first determine $\hat{m}$ at which equation (29) attains its maximum as a function of m. Next utilizing $\hat{m}$, we may calculate $\hat{par}$, maximizing $L(X;a,m,b,par)$ as a function of par. Van Dorp and Kotz [25] proved that equation (29) attains its maximum, for fixed a and b, at one of the order statistics $\left(X_{(1)},X_{(2)},\ldots,X_{(s)}\right)$. Specifically,

$$\hat{m}\left(a,b\right)=X_{(\hat{r}(a,b))} \tag{A.5}$$

$$\hat{r}\left(a,b\right)=\arg\max_{r\in\{1,2,\ldots,s\}}\left\{M\left(a,b,r\right)\right\} \tag{A.6}$$

$$M\left(a,b,r\right)=\prod_{i=1}^{r-1}\frac{X_{(i)}-a}{X_{(r)}-a}\prod_{i=r+1}^{s}\frac{b-X_{(i)}}{b-X_{(r)}} \tag{A.7}$$

Utilizing $H\left\{X;a,\hat{m}(a,b),b\right\}=M\left\{a,b,\hat{r}(a,b)\right\}$, the maximum likelihood estimator of n, $\hat{n}(a,b)$, maximizing (28) together with the maximum likelihood estimator of m, $\hat{m}(a,b)$ in (30) is:

$$\hat{par}\left(a,b\right)=-\frac{\log\left[M\left\{a,b,\hat{r}(a,b)\right\}\right]}{s} \tag{A.8}$$

References

[1] Storn R, Price K. Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. J Glob Optim 1997; 11:341–359.

[2] Vesterstroem J, Thomsen R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In Proc. IEEE Congr Evolu Comput, Portland, 2004; 1980–1987.

[3] Sun, J., Q. Zhang, and E. Tsang, DE/EDA: A new evolutionary algorithm for global optimization. Info. Sci. 2004; 169:249–262.

[4] Liu J, Lampinen J. On setting the control parameter of the differential evolution method. In Proc. 8th Int Conf Soft Comput 2002; 11–18.

[5] Yao X, Liu Y, Lin G. Evolutionary programming made faster. IEEE Trans Evol Comput 1999; 3(2):82-102.

[6] Lee CY, Yao X. Evolutionary programming using mutations based on the Levy probability distribution, IEEE Trans Evol Comput 2004; 8(1):1–13.

[7] Pan QP, Suganthan PN, Wang L, Gao L, Mallipeddi R. A differential evolution algorithm with self-adapting strategy and control parameters. Comput and Oper Res 2011; 38(1):394–408.

[8] Brest J, Greiner S, Boskovic B, Mernik M, Zumer V. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Trans Evol Comput 2006; 10(6):646–657.

[9] Omran MGH, Salman A, Engelbrecht AP. Self-adaptive Differential Evolution. Comput Intell and Secur, Lect Notes in Comput Sci 2005; 37(12):192–199.

[10] Wang L, Pan QP, Suganthan PN, Wang WH, Wang YM. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. Comput and Oper Res 2010; 37(3):509–520.

[11] Pan QP, Wang L, Qian B. A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. Comput and Oper Res 2009; 36(8):2498–2511.

[12] Salman A, Ahmad I, Omran MGH, Mohammad MG. Frequency assignment problem in satellite communications using differential evolution. Comput and Oper Res 2010; 37(12):2152–2163.

[13] Mladenovic N, Hansen E. Variable neighborhood search. Comput and Oper Res 1997; 24(1):1097–1100.

[14] Mladenovic N, Drazic M, Kovacevic-Vujcic V, Cangalovic M. General variable neighborhoods search for the continuous optimization. Eur J of Oper Res 2008; 191(3):753–770.

[15] Carrizosa E, Drazic M, Drazic Z, Mladenovic N. Gaussian variable neighborhood search for continuous optimization. Comput and Oper Res 2012; 39:2206–2213.

[16] Yang Z, Tang K, Yao X. Self-adaptive differential evolution with neighborhood search. In Proc. IEEE Congr on Evol Comput (CEC-2008), Hong Kong, 2008; 1110–1116.

[17] Mezura-Montes E, Velazquez-Reyes J, Coello Coello CA. A comparative study of differential evolution variants for global optimization. In Proc. Genet Evol Comput Conf (GECCO '06), Seattle, 2006; 485–492.

[18] Price K, Storn R, Lampinen J. Differential Evolution: A Practical Approach to Global Optimization. Berlin:Springer-Verlag; 2005.

[19] Tvrdik J. Di?erential Evolution with Competitive Setting of its Control Parameters. TASK q 2007; 11:169–179.

[20] Salomon R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions: a survey of some theoretical and practical aspects of genetic algorithms. Biosystems 1996; 39(3):263–278.

[21] Gensane T. Dense Packings of Equal Spheres in a Cube. Electronic J Comb 2004; 11(1):1–17.

[22] Johnson D. The triangular distribution as a proxy for the beta distribution in risk analysis. The Statistician 1997; 46(3):387–398.

[23] Johnson NL, Kotz S. Nonsmooth sailing or triangular distributions revisited after some 50 years. The Statistician 1999; 48(2):179–187.

[24] Williams TM. Practical use of distributions in network analysis. J Oper Res Soc 1992; 43(3):265–270.

[25] Van Dorp J, Kotz S. A Novel Extension of the Triangular Distribution and its Parameter Estimation. The Statistician 2002; 51(1):63–79.

[26] Lavor C, Maculan N. A function to test methods applied to global minimization of potential energy of molecules. Num alg 2004; 35:287–300.

[27] Smith JE, Fogarty TC. Operator and parameter adaptation in genetic algorithms. Soft Comput 1997; 1(2):81–87.

[28] Qin AK, Huang VL, Suganthan PN. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans on Evol Comput 2009; 13(2):398–417.

[29] Zhang J, Sanderson AC. JADE: Adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 2009; 13(5):945–958.

[30] Wang Y, Cai Z, Zhang Q. Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans on Evol Comput 2011; 15(1):55–66.