COMPETITIVE DIFFERENTIAL EVOLUTION AND GENETIC ALGORITHM IN GA-DS TOOLBOX

J. Tvrdík

University of Ostrava

1 Introduction

The global optimization problem with box constrains is formed as follows: for a given objective function $f : D \to \mathbb{R}, \quad D = \prod_{i=1}^{d}[a_i, b_i], \quad a_i < b_i, \quad i = 1, 2, \ldots, d$ the point $\boldsymbol{x}^*$ is to be found such that $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in D} f(\boldsymbol{x})$. The point $\boldsymbol{x}^*$ is called the global minimum point and D is the search space.

The problem of the global optimization is hard and plenty of stochastic algorithms were proposed for its solution, see e.g. [1], [7]. The authors of many such stochastic algorithms claim the efficiency and the reliability of searching for the global minimum. The reliability means that the point with minimal function value found in the search process is sufficiently close to the global minimum point and the efficiency means that the algorithm finds a point sufficiently close to the global minimum point at reasonable time. However, the efficiency and the reliability of many stochastic algorithms is strongly dependent on the values of control parameters. Self-adaptive algorithms reliable enough at reasonable time-consumption without the necessity of fine tuning their input parameters have been studied in recent years, e.g. Winter et al. [12] proposed a flexible evolutionary agent for real-coded genetic algorithms. The proposal of an adaptive generator of robust algorithms is described in Deb [2]. Theoretical analysis done by Wolpert and Macready [13] implies, that any search algorithm cannot outperform the others for all objective functions. In spite of this fact, there is empirical evidence, that some algorithms can outperform others for relatively wide class of problems both in the convergence rate and in the reliability of finding the global minimum point. Thus, the way to the adaptive algorithms leads rather trough the experimental research than trough purely theoretical approach.

2 Differential Evolution and its Control Parameters

The differential evolution (DE) is an evolutionary heuristic search for the global minimum in box-constrained search space. It was proposed by Storn and Price [8] in nineties. The differential evolution has become one of the most popular algorithms for the continuous global optimization problems in last decade [3].

Basic idea of DE is very simple. The differential evolution works with two population P (old generation) and Q (new generation) of the same size N. A new trial point $\boldsymbol{y}$ is composed of the current point $\boldsymbol{x_i}$ of old generation and the point $\boldsymbol{u}$ obtained by using mutation. If $f(\boldsymbol{y}) < f(\boldsymbol{x_i})$ the point $\boldsymbol{y}$ is inserted into the new generation Q instead of $\boldsymbol{x_i}$. After completion of the new generation Q the old generation P is replaced by Q and the search continues until stopping condition is fulfilled. The differential evolution in pseudo-code is written as Algorithm 1 in more detail.

There are several variants how to generate the mutant point $\boldsymbol{u}$. Most frequently used one (called DERAND in this text) generates the point $\boldsymbol{u}$ by adding the weighted difference of two points

$$\boldsymbol{u} = \boldsymbol{r}_1 + F\left(\boldsymbol{r}_2 - \boldsymbol{r}_3\right), \tag{1}$$

where $\boldsymbol{r}_1, \boldsymbol{r}_2$ and $\boldsymbol{r}_3$ are three distinct points taken randomly from P (not coinciding with the current $\boldsymbol{x}_i$) and $F > 0$ is an input parameter. Another variant called DEBEST generates the

point $\boldsymbol{u}$ according to formula

$$\boldsymbol{u} = \boldsymbol{x}_{\min} + F\,(\boldsymbol{r}_1 + \boldsymbol{r}_2 - \boldsymbol{r}_3 - \boldsymbol{r}_4)\,, \tag{2}$$

where $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4$ are four distinct points taken randomly from P (not coinciding with the current $\boldsymbol{x}_i$), $\boldsymbol{x}_{\min}$ is the point of P with minimal function value, and $F > 0$ is an input parameter.

Algorithm 1. Differential evolution

```
1   generate P = (x_1, x_2, ..., x_N); (N points in D)
2   repeat
3       for i := 1 to N do
4           compute a mutant vector u;
5           create y by the crossover of u and x_i;
6           if f(y) < f(x_i) then insert y into Q
7                            else insert x_i into Q
8           endif;
9       endfor;
10      P := Q;
11  until stopping condition;
```

The elements y_j, $j = 1, 2. \ldots, d$ of trial point $\boldsymbol{y}$ are built up by the crossover of its parents $\boldsymbol{x}_i$ and $\boldsymbol{u}$ using the following rule

$$y_j = \begin{cases} u_j & \text{if} \quad U_j \le C \quad \text{or} \quad j = l \\ x_{ij} & \text{if} \quad U_j > C \quad \text{and} \quad j \ne l\,, \end{cases} \tag{3}$$

where l is a randomly chosen integer from $\{1, 2, \ldots, d\}$, $U_1, U_2, \ldots, U_d$ are independent random variables uniformly distributed in $[0,\ 1)$, and $C \in [0, 1]$ is an input parameter influencing the number of elements to be exchanged by crossover. Eq. (3) ensures that at least one element of $\boldsymbol{x}_i$ is changed even if $C = 0$.

The main advantage of the differential evolution consist in its simplicity. It has only three input parameters controlling the search process, namely the size of population N, the mutation parameter F and the crossover parameter C. However, it is also known that the efficiency of the search for the global minimum is very sensitive to the setting the control parameters and it becomes the disadvantage of DE in some global optimization problems. The recommended values are $F = 0.8$ and $C = 0.5$, but even Storn and Price in their principal paper [8] use $0.5 \le F \le 1$ and $0 \le C \le 1$ depending on the results of preliminary tuning. They also set the size of population less than the recommended $N = 10\,d$ in many of their test tasks. Many papers deal with the setting of the control parameters for differential evolution. Recent state of adaptive parameter control in differential evolution is summarized by Liu and Lampinen [4]. The setting of the control parameters can be made self-adaptive trough the implementation of a competition into the algorithm [11]. This idea is similar to the competition of local-search heuristics in evolutionary algorithm [9] or in controlled random search [10].

Let us have H settings (different values of F and C used in the statements on line 4 and 5 of Algorithm 1) and choose among them at random with the probability $q_h,\ h = 1, 2, \ldots, H$. The probabilities can be changed according to the success rate of the setting in preceding steps of search process. The h-th setting is successful if it generates such a trial point $\mathbf{y}$ that $f(\mathbf{y}) < f(\mathbf{x}_i)$. When n_h is the current number of the h-th setting successes, the probability q_h can be evaluated simply as the relative frequency

$$q_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(n_j + n_0)}\,, \tag{4}$$

where $n_0 > 0$ is a constant. The setting of $n_0 \ge 1$ prevents a dramatic change in q_h by one random successful use of the h-th parameter setting. In order to avoid the degeneration of

process the current values of q_h are reset to their starting values ($q_h = 1/H$) if any probability q_h decreases bellow a given limit $\delta > 0$.

It is supposed that such a competition of different settings will prefer successful settings. The competition will serve as an self-adaptive mechanism of setting control parameter appropriate to the problem actually solved.

3 Experiments and Results

The algorithms were tested on the benchmark of six functions commonly used in experimental tests. Two of them are unimodal (first De Jong, Rosenbrock), the other test functions are multimodal. Definitions of the test functions can be found e.g. in [8] and they are also described in Appendix as well as the definition of the search search space for all the functions. The search spaces D in all test tasks were symmetric and with the same interval in each dimension, $a_i = -b_i,\ b_i = b_j\ i, j = 1, 2, \ldots, d$.

The test were preformed for all the functions at four levels of dimension d of search spaces, namely $d = 2$, $d = 5$, $d = 10$ and $d = 30$. One hundred of repeated runs were carried out for each function and level of d.

The `ga` function from Genetic Algorithm and Direct Search Toolbox [5] for the global minimization were compared differential evolution in numerical tests. The calling statement of `ga` function was the same for all the parameter settings tested in `ga`:
`[xstar,fnstar,reason,output]=ga(fhand,d,[],[],[],[],a,b,[],options);`
Four variants of settings for GA were tested. The sequence of these settings in the tests was as follows:

- GA1 – as many default options as possible for the box-constrained problems
  ∗ initial population was generated in the whole search space D
  `options= gaoptimset('PopInitRange', [a;b]) {default PopInitRange=[0;1]}`
  ∗ the mutation for the box-constrained problems was set up
  `options=gaoptimset(options,'MutationFcn',@mutationadaptfeasible)`

- GA2 – due to the poor reliability of the search by GA1, some additional options were changed in order to make the results more reliable:
  ∗ population size was increased for higher search space dimensions
  `options= gaoptimset(options,'PopulationSize',10*d)`
  `{default PopulationSize=20}`
  ∗ maximum number of generation was increased
  `options= gaoptimset(options,'Generations', 10000) {default Generations=100}`
  ∗ more benevolent stopping condition
  `options= gaoptimset(options,'TolFun',1e-12) {default TolFun=1e-6}`

- GA3 – comparing with GA2 some options were changed in order to get time demand comparable with standard DE
  ∗ size of the population modified
  `options= gaoptimset(options,'PopulationSize', max([20;5*d]))`
  ∗ number of surviving parent-population individuals increased for higher d
  `options= gaoptimset(options,'EliteCount', d) {default EliteCount=2}`
  ∗ slightly more strict stopping condition
  `options= gaoptimset(options,'TolFun',1e-10)`

- GA3hyb – the same setting as in GA3, but the search process continues by the local constrained optimizer after *ga* terminates
  `options= gaoptimset(options,'HybridFcn',@fmincon)`

The results of these setting of *ga* were compared with two variants differential evolution:

- DERAND – the standard DE with recommended values $F = 0.8$ and $C = 0.5$, the mutant vector $\boldsymbol{u}$ generated according to (1),
- DEBR18 – 18 competing control-parameter settings, the mutant vector $\boldsymbol{u}$ is generated according to (1) in nine settings and according to (2) in remaining nine settings. Combinations of F values ($F = 0.5$, $F = 0.8$, $F = 1$) and C values ($C = 0$, $C = 0.5$, $C = 1$) creates nine parameter settings for each variant of the mutant vector $\boldsymbol{u}$ [11].

Population size for both variants of DE was set to $N = \max(20, 2\,d)$, parameters for competition control in DEBR18 were set to $n_0 = 2$, and $\delta = 1/(5\,H)$ in all the test tasks. The search for the global minimum was stopped if $f_{\max} - f_{\min} < 1E-07$ or the number of objective function evaluations exceeds the input upper limit $20000\,d$.

Table 1: Comparison of DE and best ga setting

Algorithm		DERB18				DERAND				GA3hyb			
Function	d	λ_f	λ_m	ne	R	λ_f	λ_m	rne	R	λ_f	λ_m	rne	R
ackley	2	7.1	6.8	2409	100	7.3	6.9	-2	100	6.2	5.9	-7	95
dejong1	2	8.4	3.7	1162	100	8.4	3.7	-1	100	11.0	8.1	29	100
griewangk	2	8.5	3.5	2876	100	6.8	2.7	25	78	4.5	1.8	-39	53
rastrig	2	8.5	4.9	1778	100	8.5	4.9	-2	99	9.0	5.7	18	97
rosen	2	8.3	4.6	1956	100	8.3	4.7	105	100	7.5	3.7	142	100
schwefel	2	7.5	5.5	1640	100	7.5	5.5	-3	100	4.6	3.1	12	57
ackley	5	6.4	6.2	6401	100	6.3	6.1	1	99	4.1	3.8	-24	67
dejong1	5	7.2	3.2	3176	100	7.1	3.2	-3	100	11.0	8.1	13	100
griewangk	5	7.2	2.5	8686	100	5.2	1.7	14	70	3.7	1.2	-61	53
rastrig	5	7.2	4.4	4989	100	6.7	4.1	16	95	7.5	5.0	0	92
rosen	5	6.9	4.2	6256	100	7.2	4.4	528	100	6.4	3.3	414	89
schwefel	5	7.4	5.4	4564	98	7.4	5.4	-3	98	1.5	0.5	-15	9
ackley	10	6.1	5.9	13569	100	5.9	5.7	14	99	3.2	2.9	-9	52
dejong1	10	6.7	3.0	6973	100	6.5	3.0	6	100	11.0	8.1	39	100
griewangk	10	6.6	2.1	13153	99	5.3	1.6	18	78	4.0	1.2	-40	68
rastrig	10	6.7	4.2	10711	100	5.3	3.4	104	82	6.3	4.3	26	82
rosen	10	6.3	4.0	20524	100	6.7	4.3	429	100	5.8	3.1	10	84
schwefel	10	7.4	5.4	9964	99	7.3	5.2	9	96	1.4	0.3	1	6
ackley	30	5.9	5.8	142208	100	5.6	5.6	164	100	0.2	0.1	-78	2
dejong1	30	6.4	3.0	78664	100	6.1	2.9	141	100	11.0	8.1	-51	100
griewangk	30	6.4	1.6	103095	100	6.0	1.5	174	100	3.4	0.6	-84	51
rastrig	30	6.4	4.1	110071	100	0.0	0.0	445	0	1.7	1.2	-51	23
rosen	30	6.3	4.3	381972	100	0.0	0.0	57	0	0.2	0.1	-87	4
schwefel	30	7.5	5.4	108050	100	7.5	5.4	206	100	1.0	0.0	-65	0

The accuracy of the result obtained by the search for the global minimum was evaluated according to the number of duplicated digits when compared with the correct certified result. The number of duplicated digits λ can be calculated via *log relative error* [6]:

- If $c \neq 0$, the λ is evaluated as

$$\lambda = \begin{cases} 0 & \text{if } \frac{|m-c|}{|c|} \geq 1 \\ 11 & \text{if } \frac{|m-c|}{|c|} < 1 \times 10^{-11} \\ -\log_{10}\left(\frac{|m-c|}{|c|}\right) & \text{otherwise}\,, \end{cases} \tag{5}$$

  where c denotes the right certified value and m denotes the value obtained by the search.

- If $c = 0$, the λ is evaluated as

$$\lambda = \begin{cases} 0 & \text{if } |m| \geq 1 \\ 11 & \text{if } |m| < 1 \times 10^{-11} \\ -\log_{10}(|\ m\ |) & \text{otherwise}\,. \end{cases} \tag{6}$$

Two values of the number of duplicated digits are reported in the results: λ_f for the function value, and λ_m, which is minimal λ for the global minimum point $(x_1, x_2, \ldots, x_d)$ found by the search.

The results of the comparison are presented in Table 1 and 2 . The time consumption is expressed as the average number (ne) of the objective function evaluations needed to reach the stopping condition. Columns of Table 1 (and also Table 2) denoted rne contain the relative change of ne in percents when compared with DEBR18 in Table 1. Therefore, the negative values of rne mean less time consumption with respect to DEBR18, the positive values bigger one, the value $rne = 100$ means that ne is twice bigger, the value $rne = -50$ means half ne in comparison with DEBR18. The reliability of search is reported in the columns λ_f and λ_m, where are the average values of one hundred runs, and also in the columns denoted R, where the percentage of runs with $\lambda_f > 4$ is given.

4 Conclusions

The results demonstrate the disadvantage of real-coded genetic algorithm implemented in `ga` function. The search for the global minimum is strongly dependent on the control-parameter setting. It is not easy to select the proper set up for the solved problem. There are more than ten control parameters (options) of `ga` influencing the search for the global minimum in the box-constrained problem and it is difficult to guess the influence of their changes on the search process in different global optimization tasks.

As it is clear from Table 2, the performance of the default setting (GA1) is very poor, very early premature convergence occurs in all the tasks with higher dimension of the search space, and even in the tasks with $d = 2$ the premature convergence was very frequent. The second parameter setting (GA2) brought slight (but insufficient) improvement in the reliability for the tasks with low dimension. The early premature convergence persists in all the tasks with $d = 30$ and the premature convergence also remains in most tasks of $d = 5$ and $d = 10$, even if the time demand was significantly higher in comparison with DEBR18 in Table 1, see all the tasks with $d = 10$. The third setting (GA3) decreased the time demand but also decreased the reliability comparing with GA2.

The most reliable results of `ga` among the tested parameter settings are the results obtained by the hybrid procedure GA3hyb, see Table 1, but the reliability of the search for the global minimum achieved by this setting is also not satisfactory. Full reliability ($R = 100$) was got only in the easiest tasks of first DeJong function (unimodal convex function, the search for the minimum is easy for each algorithm) and in the Rosenbrock function with $d = 2$. Moreover, it is surprising, that the full reliability was not achieved by GA3hyb in the other test tasks using Rosenbrock function, although this function is unimodal. The best parameter setting GA3hyb of `ga` found in the tests does not outperform nor standard DE in most tasks.

Table 2: Results for GA1, GA2 and GA3 and comparison of ne with DEBR18

Algorithm		GA1				GA2				GA3			
Function	d	λ_f	λ_m	ne	R	λ_f	λ_m	rne	R	λ_f	λ_m	rne	R
ackley	2	1.8	1.7	-51	0	3.6	3.3	14	12	3.2	2.9	-7	3
dejong1	2	4.5	2.3	-12	67	7.0	3.6	74	100	5.7	2.9	28	100
griewangk	2	2.3	0.7	-63	10	2.7	0.9	-24	26	2.7	0.9	-36	24
rastrig	2	2.5	2.4	-34	5	5.5	4.0	44	100	4.9	3.6	19	88
rosen	2	1.5	0.5	-40	3	3.8	1.6	378	33	2.3	0.9	142	4
schwefel	2	2.3	1.4	-36	9	4.4	2.6	44	62	3.6	2.1	11	59
ackley	5	0.4	0.4	-79	0	3.5	3.2	87	2	2.1	1.8	-27	0
dejong1	5	2.0	1.1	-62	1	6.2	3.2	159	100	4.6	2.4	12	81
griewangk	5	1.4	0.2	-88	0	2.8	0.8	-1	36	2.5	0.6	-61	9
rastrig	5	0.6	1.1	-65	0	5.1	3.8	120	98	3.5	2.9	1	31
rosen	5	0.2	0.1	-75	0	1.8	0.7	3417	0	0.7	0.2	439	0
schwefel	5	1.0	0.2	-69	0	2.3	1.0	94	26	1.6	0.5	-17	16
ackley	10	0.0	0.0	-90	0	3.3	3.1	149	0	1.5	1.3	-11	0
dejong1	10	0.9	0.6	-76	0	5.2	2.8	219	98	3.8	2.1	38	31
griewangk	10	1.1	0.0	-92	0	3.0	0.8	53	37	2.1	0.4	-44	0
rastrig	10	0.0	0.1	-83	0	4.4	3.5	185	76	2.6	2.4	32	3
rosen	10	0.0	0.0	-91	0	0.7	0.2	2199	0	0.1	0.0	27	0
schwefel	10	0.7	0.0	-83	0	1.3	0.1	139	3	1.1	0.0	2	1
ackley	30	0.0	0.0	-99	0	1.3	1.3	-29	0	0.1	0.0	-77	0
dejong1	30	0.0	0.0	-98	0	2.7	1.7	21	2	1.5	1.0	-50	0
griewangk	30	0.8	0.0	-99	0	1.7	0.1	-61	0	1.3	0.0	-87	0
rastrig	30	0.0	0.0	-99	0	1.8	2.3	22	0	0.3	0.5	-51	0
rosen	30	0.0	0.0	-99	0	0.0	0.0	-67	0	0.0	0.0	-88	0
schwefel	30	0.3	0.0	-98	0	1.2	0.0	-18	0	0.9	0.0	-67	0

The DEBR18 algorithm was the most reliable among all the tested algorithms and it is also faster when comparing with the standard DERAND and four `ga` parameter settings, DEBR18 is outperformed in the convergence rate by GA3hyb only in one task (Dejong1, $d = 30$), where both the DEBR18 and the GA3hyb achieve the full reliability.

The competitive setting of the control parameters F and C in DE proposed in [11] proved to be an useful tool for self-adaptation of differential evolution, which can help to solve the box-constrained global optimization tasks without necessity of fine parameter tuning. The source code of the DEBR18 algorithm in Matlab is available at author's web site (http://albert.osu.cz/tvrdik).

References

[1] Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)

[2] Deb, K.: A population-based algorithm-generator for real parameter optimization. Soft Computing **9** (2005) 236 – 253

[3] Lampinen, J.: A Bibliography of Differential Evolution Algorithm. Technical Report. Lappeenranta University of Technology, Department of Information Technology. http://www.lut.fi/∼jlampine/debiblio.htm (2002)

[4] Liu, J., Lampinen, J.: A Fuzzy Adaptive Differential Evolution Algortithm. Soft Computing **9** (2005) 448 – 462

[5] MATLAB, version 7 (R2006a), The MathWorks, Inc. (2006)

[6] McCullough, B.D.,Wilson, B.: On the accuracy of statistical procedures in Microsoft Excel 2003. Comput. Statist. and Data Anal. **49** (2005) 1244 – 1252

[7] Spall, J. C.: Introduction to Stochastic Search and Optimization, Wiley-Intersience (2003)

[8] Storn, R., Price, K.: Differential evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. J. Global Optimization **11** (1997) 341 – 359

[9] Tvrdík, J., Mišík, L., Křivý, I.: Competing Heuristics in Evolutionary Algorithms. 2nd Euro-ISCI, Intelligent Technologies - Theory and Applications (Sinčák P. et al. eds.), IOS Press, Amsterdam (2002) 159 – 165

[10] Tvrdík, J.: Generalized controlled random search and competing heuristics. MENDEL 2004, 10th International Conference on Soft Computing, (Matoušek R. and Ošmera P. eds). University of Technology, Brno (2004) 228 – 233

[11] Tvrdík, J.: Competitive Differential Evolution. MENDEL 2006 , 12th International Conference on Soft Computing (Matoušek R. and Ošmera P. eds). University of Technology, Brno (2006) 7 – 12

[12] Winter, G., Galvan, B., Alonso, S., Gonzales, B., Jimenez, J. I., Greimer, D.: A flexible evolutionary agent: cooperation and competition among real-coded evolutionary operators. Soft Computing **9** (2005) 299 – 323

[13] Wolpert, D. H., Macready, W.G.: No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation **1** (1997) 67 – 82

Author's address:
Department of Computer Science, University of Ostrava, 30.dubna 22, 701 03 Ostrava, tvrdik@osu.cz

The work was supported by the grant 201/05/0284 of the Czech Grant Agency and by the research scheme MSM 6198898701 of the Institute for Research and Applications of Fuzzy Modeling, University of Ostrava.

Appendix – Test functions

- Ackley's function

$$f(\boldsymbol{x}) = -20\exp\left(-0.02\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos 2\pi x_i\right) + 20 + \exp(1)$$

$$x_i \in [-30, 30],\ \boldsymbol{x}^* = (0, 0, \ldots, 0),\ f(\boldsymbol{x}^*) = 0$$

- First De Jong's function

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} x_i^2$$

$$x_i \in [-5.12, 5.12],\ \boldsymbol{x}^* = (0, 0, \ldots, 0),\ f(\boldsymbol{x}^*) = 0$$

- Griewangk's function

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} \frac{{x_i}^2}{4000} - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$x_i \in [-400, 400],\ \boldsymbol{x}^* = (0, 0, \ldots, 0),\ f(\boldsymbol{x}^*) = 0$$

- Rastrigin's function

$$f(\boldsymbol{x}) = 10d + \sum_{i=1}^{d} [{x_i}^2 - 10\cos(2\pi x_i)]$$

$$x_i \in [-5.12, 5.12],\ \boldsymbol{x}^* = (0, 0, \ldots, 0),\ f(\boldsymbol{x}^*) = 0$$

- Rosenbrock's function (banana valley)

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

$$x_i \in [-2.048, 2.048],\ \boldsymbol{x}^* = (1, 1, \ldots, 1),\ f(\boldsymbol{x}^*) = 0$$

- Schwefel's function

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} x_i \sin(\sqrt{|\ x_i\ |})$$

$$x_i \in [-500, 500],\ \boldsymbol{x}^* = (s, s, \ldots, s),\ s = 420.97,\ f(\boldsymbol{x}^*) = -418.9829d$$