Leveraging Local Optima Network Properties for Memetic Differential Evolution

Viktor Homolya[1] and Tamás Vinkó[2]

[1] University of Szeged, Department of Computational Optimization, Szeged, Hungary
homolyav@inf.u-szeged.hu
[2] University of Szeged, Department of Computational Optimization, Szeged, Hungary
tvinko@inf.u-szeged.hu

Abstract. Population based global optimization methods can be extended by properly defined networks in order to explore the structure of the search space, to describe how the method performed on a given problem and to inform the optimization algorithm so that it can be more efficient. The memetic differential evolution (MDE) algorithm using local optima network (LON) is investigated for these aspects. Firstly, we report the performance of the classical variants of differential evolution applied for MDE, including the structural properties of the resulting LONs. Secondly, a new restarting rule is proposed, which aims at avoiding early convergence and it uses the LON which is built-up during the evolutionary search of MDE. Finally, we show the promising results of this new rule, which contributes to the efforts of combining optimization methods with network science.

Keywords: global optimization · memetic differential evolution · local optima network · network science

1 Introduction

Consider the global optimization problem

$$\min_{\mathbf{x}\in D\subset\mathbb{R}} f(\mathbf{x}), \tag{1}$$

where f is a continuous function, which we aim to solve by the usage of memetic differential evolution (MDE) [10]. Recent benchmarking results [1, 5] show the promising efficiency of MDE over challenging optimization problems. Differential evolution (DE) is a well known iterative, population based algorithm [12] using only the function value of f as information. Memetic approaches use local optimization method in each and every iteration, hence the population members are always local optima of the objective function [7, 9]. MDE is a simple extension of DE, the formal description of the algorithm is the following.

1. Start with a random population $\{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$ $\quad(\mathbf{p}_i \in \mathcal{R}^n)$.
2. For each $\mathbf{p}_i$ iterate until the stopping conditions hold:

(a) Select three pairwise different elements from the population: $\mathbf{p}_j, \mathbf{p}_k, \mathbf{p}_l$, all different from $\mathbf{p}_i$.
(b) Let $\mathbf{c} = \mathbf{p}_j + F \cdot (\mathbf{p}_k - \mathbf{p}_l)$ be a candidate solution.
(c) Modify vector $\mathbf{c}$ applying a CR-crossover using vector $\mathbf{p}_i$.
(d) Execute a local search from vector $\mathbf{c}$.
(e) Replace vector $\mathbf{p}_i$ with vector $\mathbf{c}$ if $f(\mathbf{c}) \leq f(\mathbf{p}_i)$ holds.

As it can be seen, MDE has some parameters: m is the population size, $F \in (0, 2)$ is the differential weight and $CR \in (0, 1)$ is the crossover probability. In Step 2(c) the CR-crossover for the candidate solution $\mathbf{c} \in \mathcal{R}^n$ means that for all dimensions of $\mathbf{c}$ a number r is generated uniform at random in $(0, 1)$. If $r > CR$ then the dimension of $\mathbf{c}$ is made equal to the same dimension of $\mathbf{p}_i$. To guarantee getting a new vector $\mathbf{c}$, the CR-crossover is skipped for a randomly selected dimension, so the linear combination of the three other vectors in this dimension is kept.

Our contributions can be summarized as follows. First, we numerically investigate the classical $x/y/z$ variants in the context of MDE. Then the MDE algorithm gets extended by the concept of local optima network (LON). In general, LONs are graphs, in which the nodes correspond to local optima of the optimization problem and the edges represent useful information either related to the problem (e.g. critical points of f) or to the optimization method in use. Similarly to our earlier work [4], the directed edges of MDE LONs are formed in such a way that they represent parent-child relation. Thus at the end of the MDE run, we obtain a graph representation of how the method discovered the landscape of the optimization problem. Apart from the standard performance metrics, we also report and compare certain characteristics of the resulting LONs using some global metrics. One of the detailed analysis is to show the relation between the function values of nodes and the function values of their out-neighbors. Based on this and some graph properties we propose an extension to the MDE which can lead to better performance on the test functions used in this paper.

2 Definitions

2.1 Strategies

The most popular DE variants which apply different strategies are distinguished by the notation *DE/x/y/z*, where

- x specifies the solution to be perturbed, and it can be either *rand* or *best*, i.e., a random one or the current best solution.
  In the above algorithm description it defines the way to choose $\mathbf{p}_j$ in Step 2(a).
- y specifies the number of difference vectors (i.e. the difference between two randomly selected and distinct population members) to be used in the perturbation done in Step 2(b), and its typical values are either 1 or 2.
  The choice $y = 1$ is considered as default and hence Step 2(a) and 2(b) are as already given in the description. In case of $y = 2$, then besides $\mathbf{p}_k$ and $\mathbf{p}_l$, further two vectors, $\mathbf{p}_m$ and $\mathbf{p}_n$ are also selected in order to create another differential vector.

– z identifies which probability distribution function to be used by the crossover operator: either *bin* (as binomial) or *exp* (as exponential).
In *bin* choose randomly a dimension index d. In Step 2(c) the vector $\mathbf{c}$ modified as for every $e \neq d$ index let $\mathbf{c}_e := \mathbf{p}_{i_e}$ with CR probability.
In *exp* choose randomly a dimension index d. Starting from d, step over every e dimension and modify $\mathbf{c}_e$ to $\mathbf{p}_{i_e}$. In every step with $1 - CR$ probability finish the modification.

2.2 Local optima network

As it was already briefly described in the Introduction, given problem (1) a local optima network (LON) is a graph in which the vertices are local optima of function f and the edges are defined between vertices separated by a critical point [13]. It is important to note that other kind of LONs can also be introduced which then specifically depend on the optimization method in use as well. In our work two vertices (local optimizers) are connected if they are in parent-child relation, i.e., the parent vertex is the target vector, the base vector or a member from the differential vector(s) and the child vertex is the result of the MDE iteration with the mentioned vectors. The edges are directed to the children. Loops are allowed, and the LON can be weighted to represent multi-edges.

Another possibility has been developed and analyzed in [11] for DE in which the nodes are the population members and the weighted edges also represent parent-child relation. However, the resulting network captures the evolution of the population members, rather than the detection of the local optima.

2.3 Network measures

It is expected that different MDE variants lead to different LONs at the end of their runs. In order to characterize these differences we can use global measures to characterize the entire graph, which are the followings.

– The number of nodes (N) and edges (M);
– the diameter (D) is the length of the longest of all directed shortest paths;
– and the average degree (d) (the average in-degree is equal to the average out-degree).

Larger N value means more local optima found. The diameter corresponds to the maximal number of times when Step 2(e) gets fulfilled for a given population member. Finally, for the average degree, for the $y = 1$ and $y = 2$ variants $d < 3.5$ and respectively $d < 5$ is an indication of early convergence.

3 Benchmarking the classic variants

The MDE and the LON creator and analyzer was implemented in Python with Pyomo [3] and NetworkX [2] packages. The local solver in MDE was MINOS [8].

3.1 Test functions

Following the numerical experiments done in [1, 5] we tested the MDE variants on two test functions:

- Rastrigin:

$$f_R(\mathbf{x}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)), \quad \mathbf{x} \in [-5.12, 5.12]^n,$$

  which is a single-funnel function with 10^n local minimizers, and its global minimum value is 0.
- Schwefel:

$$f_S(\mathbf{x}) = \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|}), \quad \mathbf{x} \in [-500, 500]^n,$$

  which is a highly multi-funnel function with 2^n funnel bottom, and its global minimum value is $-418.98129n$.

In fact, we used modified versions of these functions, namely we applied shifting and rotation of Rastrigin: $f_R(\mathbf{W}(\mathbf{x} - \overline{\mathbf{x}}))$, and rotation on Schwefel: $f_S(\mathbf{W}(\mathbf{x}))$, where $\mathbf{W}$ is an n-dimensional orthogonal matrix, and $\overline{\mathbf{x}}$ is an n-dimensional shift vector. These transformations result in even more challenging test functions, as they are non-separable and their global minimizer points do not lie in the center of their search space (as in the original versions).

3.2 Performance Metrics

After fixing the shift vectors and the rotation matrices for the test functions we executed $K = 50$ independent runs for all MDE variations. The performance metrics used to compare their efficiency are the followings.

- S is the percentage of success, i.e. how many times we reached the global minimizer;
- 'Best' is the best function value found out of K runs;
- 'Avg' is the average of function values;
- 'Adf' is the average distance between the found function value and the global optimum value in those runs where a failure occurred [5];
- 'LS' is the average number of local searches per successful run;
- 'SP' is the success performance [1], which is calculated as

$$\text{mean}(\#\text{ local searches over successful runs}) \times \frac{K}{\#\text{ successful runs}}.$$

Note that for all metrics, but for S, lower number indicates better performance.

3.3 Stop Conditions

The following stopping conditions were used:

- the sum of pairwise differences of the current population members' values less than 10^{-4};
- the population members had not replaced over the last 100 iterations;
- the best founded value did not changed during the last 20,000 local searches (# iterations $\times$ m, where m is the population size).

3.4 Results

As it was already mentioned, we executed $K = 50$ independent runs for every variants. Both the dimension and the population size was fixed to 20. The MDE parameters were set up as $F = 0.5$ and $CR = 0.1$ in all experiments.

For the Rastrigin function the tested strategies resulted in different performance metrics as it can be seen in Table 1. The most successful is the *rand/2/bin* variant as it was able to find the global optimum in all cases. Overall the *rand/y/z* strategies did quite well, except the *rand/1/exp* which resulted in the highest SP value. Among the *best/x/y* ones the *best/2/bin* got the highest success rate and the lowest SP value, whereas the *best/1/exp* did not succeed at all. Regarding the LONs we can notice that the *x/2/z* strategies led to larger graphs, as expected. This is a clear indication that these versions discover wider regions during the optimization runs. Note that larger LONs, such as *rand/2/z* have not resulted in larger diameters. The small size LONs of the *best/1/z* strategies and their low average degree are evidences of early convergence to local optima.

Table 1. Performance and graph metrics for rotated and shifted Rastrigin-20

rule	S	Best	Avg	Adf	LS	SP	N	M	D	d
best/1/bin	4	0	6.10	6.35	490	12250	358.5	1345.7	10.8	3.74
best/1/exp	0	1.98	10.51	10.5	∞	∞	224.3	837.3	9.24	3.72
best/2/bin	44	0	0.73	1.31	1462.7	**3324**	1370.5	7809.7	12.52	5.69
best/2/exp	14	0	1.48	1.72	1042.8	7449	879.6	5002	12.02	5.68
rand/1/bin	54	0	0.69	1.51	1938.5	3590	1721.8	6954.0	14.38	4.03
rand/1/exp	12	0	2.54	2.88	1393	11611	1106.1	4504.6	14.22	4.07
rand/2/bin	**100**	0	**0**	**0**	6325	6325	6203.7	37212.3	14	5.99
rand/2/exp	92	0	0.09	1.24	3964.3	4309	3817.6	22901.3	13.38	5.99

As it was expected the Schwefel problem turned out to be much more challenging for the MDE versions, see Table 2. Only three out of eight strategies were able to find the global optimum at least once. For this function *rand/2/bin* has the largest success rate and the lowest Adf and SP values, being essentially better than any other variants. However, the relative good performance of *rand/2/bin* is related to the highest number of nodes and edges in its LONs, hence it spends considerably more computational time than the others. An overall observation

Table 2. Performance and graph metrics for rotated Schwefel-20

rule	S	Best	Avg	Adf	LS	SP	N	M	D	d
best/1/bin	0	-7905.9	-7371.6	1007.9	∞	∞	176.1	633.3	7.1	3.57
best/1/exp	0	-8142.7	-7204.9	1174.6	∞	∞	100	341.7	5.9	3.39
best/2/bin	0	-8261.2	-7886.8	492.8	∞	∞	1857.5	10573.4	7.5	5.64
best/2/exp	0	-8024.3	-7629.7	749.9	∞	∞	821.7	4658.6	7.1	5.62
rand/1/bin	2	-8379.6	-7875.2	514.6	3520	176000	2186.8	8676.4	10.8	3.97
rand/1/exp	0	-8024.3	-7639.5	740.1	∞	∞	931.7	3754.9	10.1	4.03
rand/2/bin	**20**	-8379.6	**-8202.2**	**221.7**	15408	**77040**	14946.1	88927.2	9.8	5.94
rand/2/exp	4	-8379.6	-8114.2	276.4	5530	138250	8763.3	52254.4	9.6	5.96

is that the diameters are certainly lower for the Schwefel problem than for the Rastrigin. On the other hand, the average degree values are very similar for the two problems.

4 MDE supported by Network Analysis

Apart from reporting the LONs and analyzing their basic characteristics, we aim at extending the MDE algorithm with rules exploiting network properties which provide us with rich amount of information about how the execution of the optimization method was done. There are lots of possibilities to do so, here we report on one of them, which turns out to be useful to guide MDE towards better performance. Based on the analysis reported below we can propose a modified version of MDE.

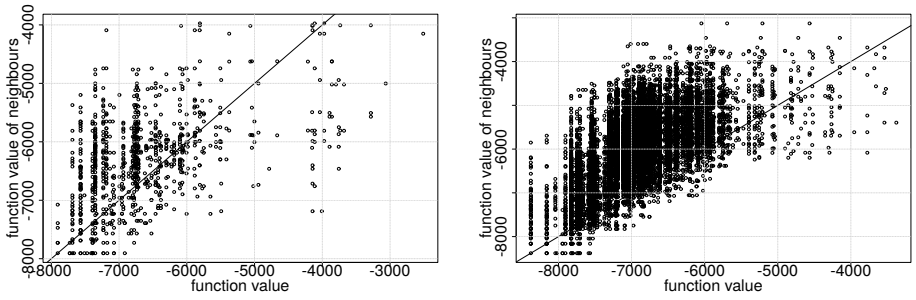

Fig. 1. Function values of out-neighbors for f_S with $n = 20$; the most successful runs for: *best/1/bin* (left) and *rand/1/bin* (right)

During the MDE run the corresponding LON gets built-up and it is possible to store the function values of the nodes. We can investigate the out-neighbors of node u and compare their function values against u. Figure 1 contains two plots of this kind, showing two different runs of two MDE variants. The x-axis

contains the function values of LON nodes with positive out-degree. Each dot shows the function values of the out-neighbors. The straight line helps us to notice the amount of neighbors with higher and lower function values for each node. Having more dots above the line indicates that the MDE variant created more children with worse function value from a given node. The side effect of this behavior is the wider discovery of the search space, which can be quite beneficial especially on multi-funnel functions such as Schwefel.

Although the *rand/1/bin* variant resulted in much larger LONs than the *best/1/bin* ones, Figure 1 clearly shows that *rand/1/bin* has relatively much more dots *above* the line than below. For the other *rand/y/z* variants we obtained similar figures, and we know from Table 2 that some of these variants were able to find the global minimizer. On the other hand, *best/1/bin* got stuck in a local minimizer point and from the plot we can see the sign of greedy behavior.

The fact that more successful variants can show similar behavior for the single-funnel Rastrigin function is shown on Figure 2. Greedy behavior for this function could lead to better performance, nevertheless, even the most successful run (in terms of best function value reached) of *best/1/exp* converged to a local minimizer (left hand side on Figure 2).

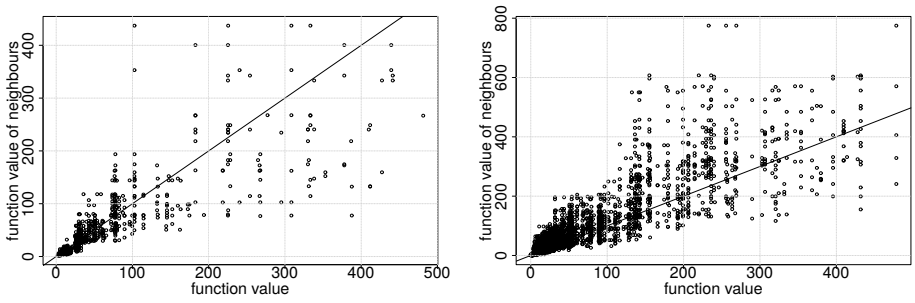

Fig. 2. Function values of out-neighbors for f_R with $n = 20$; the most successful runs for: *best/1/exp* (left) and *rand/1/exp* (right)

Based on these observations we are ready to propose an extension to MDE using LON.

4.1 Above-Below Rule

To avoid the early converge we propose a restart-rule to be applied some members of the population. Only the ones which generated the convergence are the problems while the MDE did not explore enough parts of the search space, so the nodes which have more neighbors below the line. Remove these members from the population and add new random ones. Permanent restarting would prevent the convergence, so the restart is applied only in every α-th iteration

of the MDE. We noticed that when the diameter of the LON is high enough, the population visited fairly large part of the space, so it has good chances to converge to the global optimum if we use the MDE without this modification.

We propose to extend the MDE algorithm in its Step 2 with the following rule, which has three integer parameters, $\delta > 0, \alpha > 0$ and $\theta \leq 0$. If the diameter of the current LON is lower than δ then in every α-th iteration for all $\mathbf{p}_i$ do the followings:

- collect the out-neighbors of $\mathbf{p}_i$ into the set N_i,
- calculate the function values of the elements of N_i,
- let $N_i^a := \{\mathbf{q} \,:\, f(\mathbf{q}) > f(\mathbf{p}_i)\}$, and $N_i^b := \{\mathbf{q} \,:\, f(\mathbf{q}) < f(\mathbf{p}_i)\}$
- if $|N_i^a| - |N_i^b| < \theta$ then replace $\mathbf{p}_i$ by a newly generated random vector.

Note that function values of the nodes are stored directly in the LON, so practically they need to be calculated only once.

4.2 Numerical experiment

Using the above introduced rule we have done extensive benchmarking in order to see the performance indicators. Our aim was to find a combination of the three parameters which leads to improved efficiency. Hence we did a parameter sweep: $\delta \in [6, 9]$, $\alpha \in [3, 6]$, and $\theta \in [-2, 0]$. The choice for the interval from which the values of δ are taken is motivated by the fact that, according to Tables 2 and 1 the diameters of the LONs for a given MDE variant are much larger for Rastrigin than for Schwefel, and it never goes beyond 10 for f_S. On the other hand, when population members for f_R are already having function values close to 0, then it is unwise to make MDE exploring the search space.

We report the results of the experiments for $n = 20$ only. According to our findings, the combination $\delta = 7, \alpha = 3, \theta = -1$ led to the best performance improvements for the tested functions. The indicators are reported in Tables 3 and 4, where improved metrics are highlighted by underline.

Table 3. Performance and graph metrics for rotated and shifted Rastrigin-20 using the new rule

rule	S	Best	Avg	Adf	LS	SP	N	M	D	d
best/1/bin	0	0.99	5.56	5.56	∞	∞	378.1	1424.3	11.52	3.76
best/1/exp	0	1.99	17.38	17.38	∞	∞	226.4	843.5	9.52	3.71
best/2/bin	60	0	0.61	1.54	1449.3	**2415**	1349.8	7677.1	12.88	5.68
best/2/exp	18	0	2.39	2.92	993.3	5519	879.8	4989.7	12.16	5.66
rand/1/bin	60	0	0.55	1.39	1996.0	3327	1794.4	7244.1	14.96	4.03
rand/1/exp	14	0	2.21	2.57	1411.4	10082	1110.1	4521.1	14.1	4.07
rand/2/bin	**100**	0	**0**	**0**	6341.6	6342	6225.1	37318.3	14.64	5.99
rand/2/exp	98	0	0.03	1.78	3897.1	3977	3776.1	22641.9	14.04	5.99

We can see that our rule improved the percentage of success (S) for the single-funnel Rastrigin function in up to 16%, and resulted in lower average

Table 4. Performance and graph metrics for rotated and shifted Schwefel-20 using the new rule

rule	S	Best	Avg	Adf	LS	SP	N	M	D	d
best/1/bin	0	-8142.7	-7685.6	693.9	∞	∞	278.1	1014.3	7.3	3.63
best/1/exp	0	-8024.3	-7464.8	914.8	∞	∞	149.8	532.8	6.3	3.51
best/2/bin	0	-8261.2	-7993.3	386.2	∞	∞	1991.5	11323.6	7.4	5.66
best/2/exp	0	-8261.2	-7834.0	545.6	∞	∞	1731.6	9869.2	7.5	5.67
rand/1/bin	0	-8142.7	-7899.7	479.8	∞	∞	2370.9	9408.0	11.2	3.97
rand/1/exp	0	-8261.2	-7639.2	740.4	∞	∞	1065.6	4270.5	10.2	4.01
rand/2/bin	**26**	-8379.6	**-8188.6**	**258.1**	13524	**52017**	16304.1	96986.8	9.9	5.94
rand/2/exp	4	-8379.6	-8111.9	278.8	5850	146250	8384.2	50048.2	9.8	5.96

function values for six out of eight variants. We obtained 7% improvement in success performance with *best/2/bin*.

For the multi-funnel Schwefel function the new rule does not help for the variants which were unsuccessful in the original versions to find the global optimum. However, it made them finding local optimum with lower function value on average and hence decreased their 'average difference failure' measure. The most efficient *rand/2/bin* variant got better in its SP measure by 32%.

5 Conclusions

To the best of our knowledge our paper is the first one reporting benchmarking results on MDE variants. According to the numerical experiments, the *rand/2/bin* strategy provides overall the best percentage of success metric, especially when it is applied on multi-funnel problem. This is somewhat in line with the results reported in [6] for DE. For a single-funnel function the *best/2/bin* variant can be advantageous if one needs good success performance, i.e. lower computational time.

We have shown that incorporating certain knowledge on the local optima network of the MDE to the evolutionary procedure can lead us to formalize restarting rules to enhance the diversification of the population. Our numerical tests indicates that the proposed restarting rule is beneficial on average for most of the MDE variants.

In this work we have developed a computational tool in Python using Pyomo and NetworkX packages which provide us with a general framework to discover further possibilities on the field of (evolutionary) global optimization and network science. We plan to extend our codebase with further MDE rules, in particular with those involve network centrality measures as selection [4].

Acknowledgment

This research has been partially supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no

EFOP-3.6.3-VEKOP-16-2017-0002. The project has been supported by the European Union and co-funded by the European Social Fund. Ministry of Human Capacities, Hungary grant 20391-3/2018/FEKUSTRAT is acknowledged.

References

1. Cabassi, F., Locatelli, M.: Computational investigation of simple memetic approaches for continuous global optimization. Computers & Operations Research **72**, 50 – 70 (2016)
2. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using NetworkX. Tech. rep., Los Alamos National Lab. (LANL), Los Alamos, NM (United States) (2008)
3. Hart, W.E., Laird, C.D., Watson, J.P., Woodruff, D.L., Hackebeil, G.A., Nicholson, B.L., Siirola, J.D.: Pyomo-optimization modeling in python, vol. 67. Springer (2012)
4. Homolya, V., T.Vinkó: Memetic differential evolution using network centrality measures. In: AIP Conference Proceedings 2070, 020023 (2019)
5. Locatelli, M., Maischberger, M., Schoen, F.: Differential evolution methods based on local searches. Computers & Operations Research **43**, 169 – 180 (2014)
6. Mezura-Montes, E., Velázquez-Reyes, J., Coello Coello, C.A.: A comparative study of differential evolution variants for global optimization. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. pp. 485–492. ACM (2006)
7. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech concurrent computation program, C3P Report **826** (1989)
8. Murtagh, B.A., Saunders, M.A.: MINOS 5.5.1 user's guide. Technical Report SOL 83-20R (2003)
9. Neri, F., Cotta, C.: Memetic algorithms and memetic computing optimization: A literature review. Swarm and Evolutionary Computation **2**, 1–14 (2012)
10. Piotrowski, A.P.: Adaptive memetic differential evolution with global and local neighborhood-based mutation operators. Information Sciences **241**, 164–194 (2013)
11. Skanderova, L., Fabian, T.: Differential evolution dynamics analysis by complex networks. Soft Computing **21**(7), 1817–1831 (2017)
12. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**, 341–359 (1997)
13. Vinkó, T., Gelle, K.: Basin hopping networks of continuous global optimization problems. Central European Journal of Operations Research **25**, 985–1006 (2017)