Memetic Differential Evolution using Network Centrality Measures

Viktor Homolya[1,b)] and Tamás Vinkó[1,a)]

[1] *University of Szeged, Department of Computational Optimization, Szeged, Hungary*

[a)]Corresponding author: tvinko@inf.u-szeged.hu
[b)]homolyav@inf.u-szeged.hu

Abstract. The concept of using network centrality measures in the selection process of memetic differential evolution algorithm is proposed. The usual aim for introducing changes in global optimization algorithms is to make it perform better. This short paper does not intend to provide enough experimental details to decide upon the performance of the new method, nevertheless, we definitely obtain interesting insights on how the discovery of local optima were done.

INTRODUCTION

The intersection of two highly relevant and active fields, global optimization and network science provides great opportunities for interesting research works. This paper reports on the preliminary results obtained by experimenting with memetic differential evolution driven by centrality measures defined on an auxiliary dynamic graph. The method is iterative, population based and belongs to the derivative-free algorithms, i.e., it uses only the function value of the global optimization problem of form $f(\mathbf{x}) : \mathcal{R}^n \to \mathcal{R}$ which needs to be minimized. As in the original version of the differential evolution (DE), superficially, in every iteration steps three elements from the population are chosen, from which, using specific formula, a new candidate solution is produced [1]. We propose and investigate new methods for this selection procedure, which are inspired by network science [2].

ALGORITHMS

Memetic Differential Evolution (MDE) Memetic algorithms are population based approaches involving local search method. Simply, all the individual elements in the population hold a locally optimal solution of the problem (except perhaps at the very beginning). A high level description of the memetic differential evolution 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.

Parameters of MDE: m is the population size, $F \in (0, 2)$ is the differential weight and $CR \in (0, 1)$ is the crossover probability. The CR-crossover for the candidate solution $\mathbf{c} \in \mathcal{R}^n$ in Step 2(c) is done in the following way. For all of its dimensions generate uniformly a random number r from the interval $(0, 1)$. If r is larger than the given parameter CR then the dimension of $\mathbf{c}$ is made equal to the same dimension of $\mathbf{p}_i$. In order to certainly get new vector $\mathbf{c}$: select and fix a dimension for which this CR-crossover is skipped (we keep the linear combination of the three other vectors in this dimension).

Local Optima Network (LON) LONs are interesting graphs which are associated with optimization problems and practically also with optimization methods. Informally, the local optimizer points are the vertices and the edges are defined between vertices separated by a critical point. General definitions can be found in [3]. In this work we propose a modified version, where the vertices are still local optimizer points found by MDE, but the edges are capturing the information generated by the iterative part of MDE.

We need to mention here that a different approach has been developed in [4] for DE, where the vertices are the members of the population, and two vertices are connected with a weighted edge if one is a parent of the other. Hence, the generated network describes the evolution of the population members, rather than the discovery of the local optima.

MDE LONs We propose the following approach that leads to an edge-weighted network. The vertices of the network are the solutions (local optima) obtained by the execution of MDE. Two vertices are connected if the corresponding vectors were in parent-child relation (as in Step 2(c) of the algorithm description given above). This means that at the end of every iteration step vector $\mathbf{c}$ (which by definition contains a local optimum) becomes a vertex in our graph, and it gets connected by $\mathbf{p}_j, \mathbf{p}_k$ and $\mathbf{p}_l$ (which also contain local optima). Obviously, a local optimum can be found multiple times. In this case new vertex is not created, but the weights of the edges get increased. Albeit the size of the population in MDE is fixed, our network is dynamically growing during the optimization procedure.

Leveraging MDE LONs The construction of the above discussed network can open up the possibilities for constructing new selection rules in the creation of a candidate solution $\mathbf{c}$ using the formula $\mathbf{c} = \mathbf{p}_j + F \cdot (\mathbf{p}_k - \mathbf{p}_l)$. In the original DE algorithm, the three parents are selected uniformly at random from the population. Now, as all the members are assigned to a vertex of the built network, one can use the so-called centrality measures of networks.

Centrality, in general, is a real valued function of the vertices of a graph. A centrality measure quantifies the importance of a vertex by ranking. The vast literature of network science proposes various definitions leading to different ranking of the vertices. We use the following measures for our experiments:

- degree centrality, which measures the degree of a vertex;
- betweenness centrality gives a score to vertices by measuring the extent to which a vertex lies on paths between other vertices;
- PageRank score of a vertex is derived from the scores of its network neighbors and it is proportional to their centrality divided by their out-degree;
- closeness centrality is the sum of the length of the shortest paths between a vertex and all other vertices.

These tools give us the possibility to define new selection rules applied in Step 2(a). In the selection of the parents, give higher chances to a vertex to be selected which has higher (or smaller) centrality measure. In practice, this selection rule should be applied only after some initial amount of iterations in order to let the MDE network grow in a uniformly random fashion. Although the size of the network we build gets increased, the selection rule is only applied to the vertices which are corresponding to any actual member of the current population of fixed size.

PRELIMINARY NUMERICAL RESULTS

In this section we show our preliminary results obtained by using the first prototype implementation. This was done in AMPL [5], which has rich enough language to handle set of vectors (as population) and to code the simple loop of differential evolution. A big advantage here is the easy usage of any AMPL compatible solver, which can serve as the local optimizer called in Step 2(d) of MDE. The question remains, how the centrality measures should be calculated? In our opinion, it would make no sense to implement them in AMPL, so we used the `R/igraph` package. Though this made the experimenting easier, this approach is obviously not the way to go. Nevertheless, this prototype was developed to provide us with the proof of concept results.

For the testing we chose the Schwefel benchmarking test function from the literature, defined as:

$$\text{Schwefel}_n(\mathbf{x}) = 418.9829n + \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|}) \qquad x_i \in [-500, 500].$$

This function has exponentially growing number of local minimizer points whose values are very close to the global optimum and on top of that, they are located at different regions of the search domain. As it is reported in [6], MDE

can find the globally optimal solution of Schwefel$_n$ with relatively high success rate for dimensions $n = 10, 50$. In this short paper, we do not aim at going for even higher percentage of success. Instead, we would like to demonstrate the fact that the proposed selection rule by using centrality measures leads to diverse solutions.

We ran MDE using the different selection rules for $n = 2$ and $n = 10$, where the size of the population was 10 and 20, respectively. Further parameters were set up as $F = 0.5$ and $CR = 0.1$. The number of iterations were intentionally kept as low as 20 steps. The new selection rules were applied only after the first 10 iteration steps.

In Table 1 the reported results correspond to the case where the new selection rules were applied for all the members of the population, ranked by the different centrality measures. In case of the lower dimension, all versions were able to find the global optimum at least once. On average, closeness centrality (cc) performed the best. For the larger problem the original DE provided the average and absolute best solution, however, was unable to find the global one. The results shown in the last row were achieved by always selecting the top three highly ranked vertices. As we can see, this lead to better solutions, especially cc obtained as good result as the original DE.

TABLE 1: Average (and best) function values achieved. The last row reports the results when the top3 most central vertices were considered.

dim	–	bc	degree	pagerank	cc
2	26.32 (0)	13.16 (0)	50.44 (0)	52.63 (0)	0 (0)
10	366.62 (118.43)	783.77 (356.83)	748.95 (572.45)	852.03 (690.89)	594.31 (236.87)
10	366.62 (118.43)	555.75 (236.87)	464.64 (236.87)	503.82 (335.57)	387.26 (118.44)

Obviously, one should not draw general conclusions from the numerical experiments done in this section. Whether MDE with the network-driven selection rules can give better results compared to the standard version remains a question, which will certainly be answered in the full version of this paper. Nevertheless, it is still interesting to investigate what are the properties of the networks obtained at the end of the optimization.

Properties of MDE LONs Investigating the features of the local optima networks built by the MDE algorithm can reveal lots of interesting results regarding how the optimizer explored the search space of the optimization problem. We give here a brief analysis of the networks obtained by the execution of our MDE. Some of the basic properties are given in Table 2. The first column contains the properties of the graph obtained by not using any centrality measure. We can see that the networks are different in the two simple aspects indicated here. The number of vertices refer to the number of different local optima found. The original MDE explored the most locally optimal solutions. Among the new methods betweenness centrality found the most local optima, however, it was unable to find the best one. Note that if the different methods found the same or distinct local optima cannot be seen from this indicator. The average degree might be a simple measure hiding many details about its distribution, it indicates that how many times a vertex, on average, got selected to be a parent. We do not see much difference here with respect to the centrality based methods, whereas the original one stands out. Knowing the results we conclude that this effect was beneficial for the original MDE.

TABLE 2: Properties of the MDE LONs corresponding to the best solutions given in the last row of Table 1.

	–	bc	degree	pagerank	cc
number of vertices	179	162	116	145	152
average degree	3.11	2.72	2.78	2.76	2.81

Visualizations of the four networks are shown in Figures 1 and 2. Note that larger vertex size represent larger centrality value measured by degree and betweenness, PageRank and closeness, respectively. Rigorous analysis should not be made on graph drawings as the appearance depends on the layout. Nevertheless, we can indeed see the differences between the four graphs, as it was already indicated by Table 2. Maybe the closeness centrality based network (on the right in Figure 2) stands out having relatively lots of vertices with higher centrality. According to Table 1, this version achieved the best solution among the network driven MDEs.

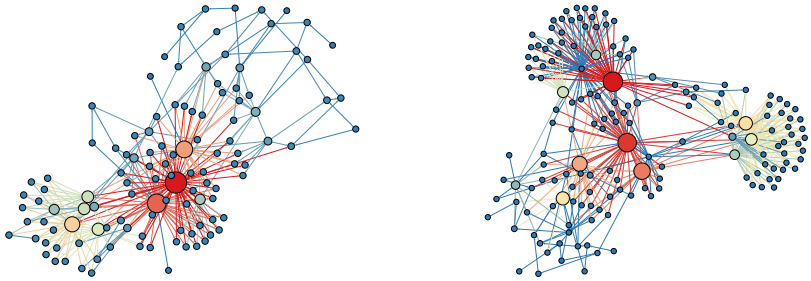

FIGURE 1: MDE LONs: using degree centrality (left) and betweenness centrality (right). Vertex size represents the corresponding centrality value.

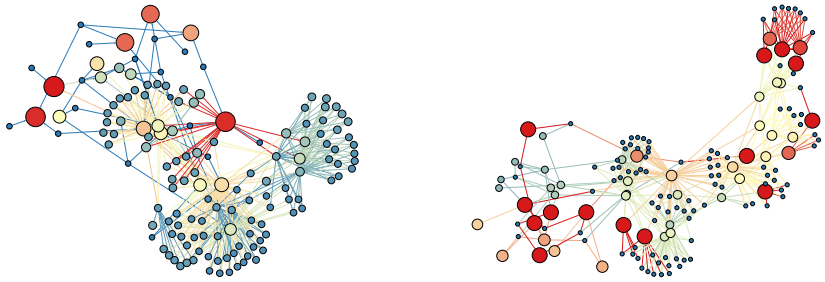

FIGURE 2: MDE LONs: using PageRank centrality (left) and closeness centrality (right). Vertex size represents the corresponding centrality value.

Final remarks

At the time of writing this short paper, we have been working on an implementation done in Python using the Pyomo open source optimization modeling language, and the `NetworkX` package for calculating the centrality measures. This environment makes it possible to keep everything in memory, hence radically shortening the execution time, compared to the prototype implementation we used earlier. We have seen that different measures lead to different results even after small amount of iterations. Whether any of these ideas lead to more efficient version of MDE will be answered in the full version of this paper using a set of challenging benchmark problems and detailed analysis of the local optima networks.

ACKNOWLEDGMENTS

This research has been 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.

REFERENCES

[1] R. Storn and K. Price, Journal of Global Optimization **11**, 341–359 (1997).
[2] M. Newman, *Networks: An Introduction* (Oxford University Press, 2010).
[3] T. Vinkó and K. Gelle, Central European Journal of Operations Research **25**, 985–1006 (2017).
[4] L. Skanderova and T. Fabian, Soft Computing **21**, 1817–1831 (2017).
[5] R. Fourer and B. Kernighan, *AMPL: A Modeling Language for Mathematical Programming* (Duxbury Press, 2002).
[6] M. Locatelli, M. Maischberger, and F. Schoen, Computers & Operations Research **43**, 169 – 180 (2014).