ON VISUALIZATION OF DIVERSITY IN EVOLUTIONARY ALGORITHMS BASED ON SIMULATION OF PHYSICAL SYSTEM

Jan Drchal[1], **Miroslav Šnorek**[1]

[1]Czech Technical University, Faculty of Electrical Engineering, Department of Computer Science and Engineering
Karlovo náměstí 13, 121 35, Prague, Czech Republic

drchaj1@fel.cvut.cz(Jan Drchal)

Abstract

Evolutionary Computation is a popular soft-computing approach to solve problems in a nature-inspired way. Evolutionary Algorithms (EAs) are well-known optimization methods that belong to this domain. All EAs work with population of organisms which represent solutions of optimized problem. Organisms compete for survival and gradually evolve to higher fitness. EAs can be seen as algorithms which traverse the search-space in a parallel way. Diversity is an essential aspect of each EA. It describes the variability of organisms in the population. The lack of diversity is a common problem. It means that all organisms are very similar to each other and that they are located around a single point in the search-space. Diversity should be preserved in order to evade local extremes (premature convergence). Niching algorithms are modifications of classical EAs. Niching is based on dividing the population into separate subpopulations - it spreads the organisms effectively all over the search-space and hence making the overall population diverse. Using niching methods also requires setting of their parameters, which can be very difficult. This paper presents a novel way of diversity visualization based on physical system simulation. It is inspired by intermolecular forces and employs overall energy minimization. This minimization is done via known unconstrained optimization numerical methods, namely, Steepest Descent, Conjugated Gradients or Quasi-Newton. The visualization is helpful when designing and tuning niching algorithms, but it has also other uses. The visualization will be presented on NEAT - the evolutionary algorithm which optimizes both the topology and the parameters of neural networks. We compare our approach with a related method of dimension reduction devised by Sammon.

Keywords: Diversity, Evolutionary algorithms, Niching, Visualization.

Presenting Author's Biography

Jan Drchal is a postgradual student and a research worker on Czech Technical University, Faculty of Electrical Engineering in Prague. His main interest in computer science is soft computing, particularly neural networks and evolutionary computation. He is a member of Neural-Computing Group (NCG) led by his tutor doc. Miroslav Šnorek.

ISBN 978-3-901608-32-2

1 Introduction

We have organized the paper as follows. In Section 2 diversity and niching algorithms are briefly described. In Section 3 diversity visualization problem is defined and a new method based on simulation of physical system model is proposed. The visualization itself is performed via physical system energy minimization. The choice of an optimization algorithm for this minimization is covered in Section 4. In Section 5 the experimental results are presented.

2 Diversity and niching algorithms

Evolutionary Algorithms (EAs) are general optimization methods based on evolution processes as we know from the Nature. EAs always work over a population of models (solutions, organisms, individuals). Each individual is described by a genome (chromosome). The *genotype* of a particular organism is its exact *genome (chromosome)* setup – it is one specific *genome* instance. The *phenotype* of an organism represents its actual physical properties (height, weight, eye color, etc.).

Typically EA run converges when the population of individuals accumulates nearby a certain point of a search space. When this point represents a local optimum then we speak about the premature convergence. This phenomenon is often explained by the lack of diversity in the population, in other words, all organisms in the population become similar. The diversity has to be to some extent preserved in order to avoid local extremes.

Niching algorithms were originally designed to simultaneously search for multiple global optima in multimodal functions, however, it has been shown that they are generally a good way to solve the problem of premature convergence for any EA. There exist many niching techniques, some of them are fitness sharing [1] or crowding [2]; a good discussion can be found in [3]. Their common problem is that they add new parameters to EA which are often very hard to find. This difficulty motivated the search for a new diversity visualization algorithm which should be helpful when incorporating a niching method into any existing EA, comparing different niching algorithms, etc.

3 Diversity visualization

As we have stated in Section 2, diversity describes the variability of organisms in the population, therefore a similarity measure (distance) d_{ij} of organisms i and j can be defined. Distance can be based on genotypic or phenotypic similarity. For genetic algorithm as a special type of EA, where chromosomes are encoded in binary strings, genotypic distance d_{ij} is often defined as a Manhattan distance of organisms i and j. Phenotypic distance is often difficult to define as it is problem-dependent, nevertheless, it is generally preferred for niching. Clearly

$$d_{ii} = 0, \quad (1)$$

$$d_{ij} = d_{ji} \quad (2)$$

for $i, j = 1..N$. Distances can be represented in the matrix form. The distance matrix is symmetric with zero diagonal.

3.1 Problem definition

The distance matrix can be trivially visualized in the $N-1$ dimensional Euclidean space. Three points that represent solutions can be projected on a 2–D plane forming a triangle. Solutions have Euclidean distances $l_{ij} = d_{ij}$ on the plane. Four points can be similarly projected to the 3–D space. Larger distance matrices cannot be visualized in 2–D or 3–D. The problem of visualization in 2–D can be then defined as:

For a given distance matrix D find representative points (x_i, y_i) in the plane such that the 2–D distances $l_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ (Euclidean distance) *approximate as closely as possible* the original distances d_{ij}. Figure 1 depicts an example situation.

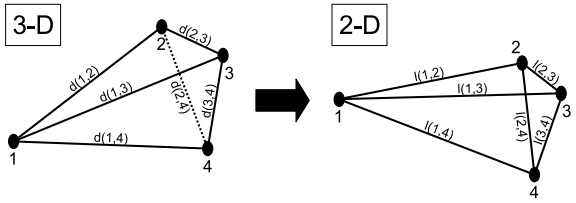

Fig. 1 Diversity visualization example. Population of size $N = 4$ is projected from 3–D to a 2–D plane. New distances l_{ij} approximate original distances d_{ij}.

The niching algorithms divide the population into several species. These species (almost) independently traverse the search-space. The visualization should show different species as clusters in 2–D. Stanley in [4] proposed such visualization, it was however limited to the so-called explicit fitness sharing. Explicit fitness sharing is a modified fitness sharing, where species cannot overlap.

3.2 Solution inspired by physics

The solution we propose is based on modeling of physical phenomena. It is inspired by intermolecular forces [5]. Atoms in molecules are exposed to attractive and repulsive forces. Attractive forces prevail for long distances while repulsive forces for short.

We used the above scheme, associating solutions with atoms, and modified it to:

1. the shorter the distance d_{ij} is, the stronger the attractive force is,
2. the shorter the 2–D distance l_{ij} is, the stronger the repulsive force is,
3. *1* prevails for longer distances, *2* for shorter.

For such system energy can be computed (see next subsection) and the minimum energy can be found (Section 4). The item 2 prevents the system from collapsing into a single point.

3.3 Energy and force equations

The total energy to be minimized is defined as

$$\Phi_{TOTAL} = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \Phi_{ij}, \tag{3}$$

where Φ_{ij} is an energy contribution from the solutions i and j. The force contribution F_{ij} can be evaluated as a gradient of Φ_{ij} taken with minus sign. We have developed two different sets of energy and force equations (**A** and **B**). The first denoted set **A** is

$$\Phi_{ij}^{A} = \left(\frac{l_{ij}}{d_{ij} + d_0} - \frac{\arctan\left(\frac{l_{ij}}{a}\right)}{a} \right), \tag{4}$$

$$F_{ij}^{A} = -\frac{\partial \phi_{ij}^{A}}{\partial l_{ij}} = -\frac{1}{d_{ij} + d_0} + \frac{1}{l_{ij}^2 + a^2}, \tag{5}$$

where d_0 and a are constants. The equations of set **B** read

$$\Phi_{ij}^{B} = \frac{l_{ij}}{(d_{ij} + a)^2} + \frac{1}{l_{ij} + a} - \frac{2}{d_{ij} + a}, \tag{6}$$

$$F_{ij}^{B} = -\frac{\partial \phi_{ij}^{B}}{\partial l_{ij}} = -\frac{1}{(d_{ij} + a)^2} + \frac{1}{(l_{ij} + a)^2}, \tag{7}$$

where a is a small positive constant which prevents division by zero. The difference between these sets is in the influence of d_{ij} for longer distances (in **A** the influence is stronger). Both **A** and **B** were found empirically, however, they are very similar to the real-world equations [5]. The elements d_{ij} of the distance matrix D are always normalized such that $\max d_{ij} = 1$ in order to keep parameter settings uniform.

The Figure 2 gives a summary of the whole visualization process.

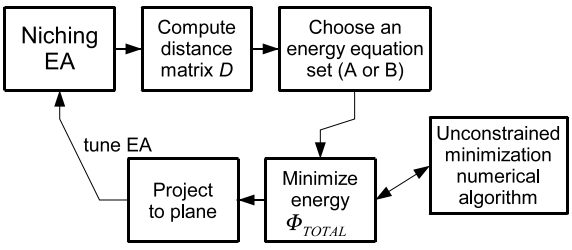

Fig. 2 Tuning niching EA using the proposed visualization. We extract population data from the niching EA and use them to compute a distance matrix D. We choose a set of equations (**A** or **B**) and continue with minimization of energy Φ_{TOTAL}. The minimization is done via an unconstrained minimization algorithm (See Section 4). At last the solution is visualized on a plane. Species are visualized as clusters. This information can be used to tune the niching EA.

3.4 Sammon's projection

The well known Sammon's projection algoritm [6] can make similar job for diversity visualisation. Sammon's projection minimizes the quadratic error between original distances d_{ij} and projected distances l_{ij}. The error (or energy) can be obtained from the following equation:

$$\Phi_{TOTAL}^{S} = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{(d_{ij} - l_{ij})^2}{d_{ij}}}{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}}. \tag{8}$$

The problem of Sammon's projection is that solutions with very small d_{ij} are projected into a single point. This behavior is not suitable for visualization purposes and therefore the positions of projected solutions must be artificialy moved.

3.5 Interactive distance information

The proposed algorithm performs a dimension reduction, and in many cases this cannot be done perfectly. Therefore the visualization was extended by a possibility of selecting a single solution. The intensities of colors of the other solutions are then adjusted using the following equation:

$$I_j = \frac{1}{1 + \varepsilon^{-\alpha\left(\frac{1}{2} - d_{ij}\right)}}. \tag{9}$$

The function is depicted in the Figure 3. Again the equation was devised empirically. It renders solutions similar to i with brighter colors and fades dissimilar. We found that useful settings of α lay in the range of $\langle 1..100 \rangle$ when D is normalized. This extension is very useful when tuning energy and force equations but also when distinguishing between overlapping clusters.

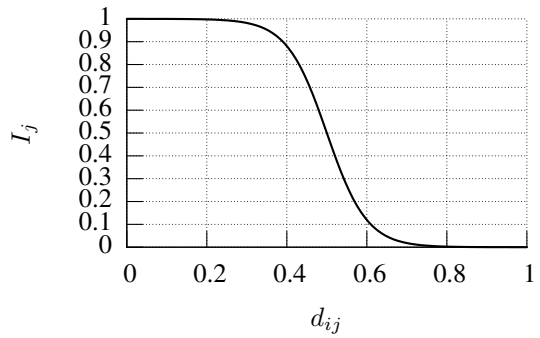

Fig. 3 Color intensity adjustment function. Similar solutions to the selected one are brighter, dissimilar are faded.

4 Optimization algorithms involved in visualization

This section describes optimization algorithms used to minimize the energy (3):

$$\min_{\mathbf{X} \in \mathbb{R}^{2N}} \Phi_{TOTAL}(\mathbf{X}), \tag{10}$$

where **X** is a vector containing coordinates (x_i, y_i) for all $i = 1 \dots N$ solutions (again, N is the size of the population). In ordinary EAs population sizes range from tens to thousands of organisms. It is clear that a special attention should be paid to the choice of optimization algorithm for (10). We made several experiments with numerical methods for unconstrained optimization, starting with simple Steepest Descent to complex Quasi-Newton method. This section briefly describes methods used, the experiments are described in Section 5.

4.1 Steepest Descent with fixed step and momentum

Steepest Descent is a simple gradient method in which steps are taken in the direction of a negative gradient, see [7]. At first we have implemented the Steepest Descent method with a fixed step (SDF) - no line minimization was used. The properties of the method were greatly improved by using a momentum (with $m = 0.9$) which is a well-known technique in optimization of neural networks [8].

4.2 Steepest Descent with line search

We have also implemented a Steepest Descent with a line search (SD). Two line search methods were tested: Brent's algorithm using derivatives [9] and a method satisfying Strong Wolfe conditions using cubic interpolation [10] (which was a default in further tests).

4.3 Conjugated Gradients

As the next method we have implemented much more sophisticated Conjugated Gradient method (CG) [7, 11, 12]. The line searches were the same as in Subsection 4.2. Our experiments have shown that Hestenes–Stiefel [7] update works best (compared to Fletcher–Reeves and Polak–Ribière).

We have also experimented with Phylogenetic Analysis Library (PAL) implementation of Conjugated Gradients (CG-PAL) [13]. Here the line search uses numerical derivatives, while the algorithm itself uses analytic gradient. Again, Hestenes–Stiefel update worked the best.

4.4 Quasi-Newton

The last method which we have tested was the Quasi-Newton method implemented in UNCMIN package [14] (QN-UNCMIN). The More-Hebdon trust region update was found to be the best.

5 Results

This section contains results of our experiments. It is divided as follows: the first Subsection shows the properties of equations (4), (5), (6) and (7) on artificial data and compares them to Sammon's projection (8). The second Subsection shows the visualization properties on real-world data. The third Subsection compares optimization algorithms used for energy minimization as presented in Section 4. The last Subsection shows that our algorithm can be used not only for diversity visualization but also as a general method for visualization of classification datasets. The initial coordinates (x_i, y_i) were randomly distributed in a square with side $a = 0.1$ centered at origin and with a uniform distribution. Parameter settings for the equation set **A** were $d_0 = 1.1$, $a = 1.005$, for the equation set **B** $a = 1.10^{-6}$.

5.1 Artificially generated clusters

Prior to the experiments on the real-world data the experiments with artificially generated data were performed. For this purpose we have used an algorithm which generates the distance matrix D. The input of the algorithm is the set of cluster sizes C and two real intervals $X = \langle x_{min}, x_{max} \rangle$ and $Y = \langle y_{min}, y_{max} \rangle$. The algorithm then creates $|C|$ clusters of given sizes, where the d_{ij}'s inside a single cluster are random numbers from X. The distances d_{ij} for i and j belonging to different clusters are random numbers from Y. We have used uniformly distributed random numbers.

For example, the following matrix represents the cluster set $C = \{2, 3\}$:

$$d_{ij} = \begin{pmatrix} 0 & x & y & y & y \\ x & 0 & y & y & y \\ y & y & 0 & x & x \\ y & y & x & 0 & x \\ y & y & x & x & 0 \end{pmatrix}, \qquad (11)$$

where $x \in X$ and $y \in Y$.

In the first place we made experiments with "coarse" clusters - with $X = \langle 0.1, 0.1 \rangle, Y = \langle 0.9, 0.9 \rangle$. Figure 4 shows the situation for the distance matrix of $N = 50$ and the cluster set $C = \{2, 3, 15, 30\}$ generated for **A**. The Figure 5 shows the same using equation set **B**. Note that the resulting figures need not to be similar as they depend on details of acting forces and on random initialization.

The second series of figures shows the situation for medium-sized clusters $C = \{2, 3, 15, 30, 75, 175\}$ ($N = 300$) - Figure 6 was generated by equations **A**, while Figure 7 by equations **B**. We can see that the the smallest clusters projected by equations **A** were disrupted by larger clusters. This problem disappeared when equations **B** were used. It is clear that for **B** equations large distances d_{ij} have small influence. In the Figure 8 we show the visualization of the same data using Sammons projection.

Figures 9 and 10 depict the similar situation where X and Y were changed to $X = \langle 0.1, 0.5 \rangle$ and $Y = \langle 0.5, 0.9 \rangle$. Tthe distinct clusters are still clearly visible.

5.2 Real world application - NEAT niching algorithm

The real-world experiments were performed on data from NEAT (NeuroEvolution of Augmenting Topologies) algorithm [4]. NEAT is a system which evolves neural network parameters and topology simultaneously. It uses a niching algorithm as its integral part. Here, we have used the modified NEAT with deterministic crowding [2]. Neural networks were learned to the classic XOR classification problem. The popula-

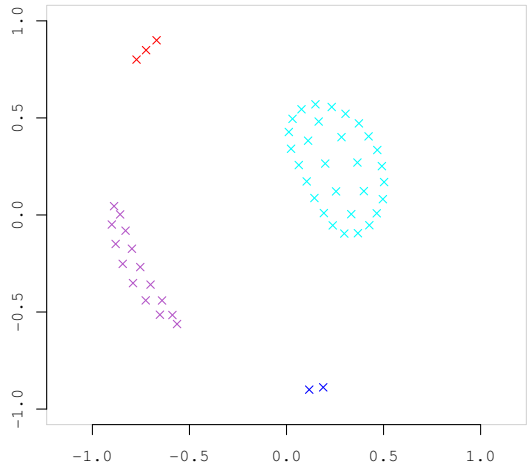

Fig. 4 Small coarse clusters, equation set **A**. Each cross represents a single organism in the population. The clusters were artificially generated in order to imitate species evolved by a niching evolutionary algorithm. Population size was $N = 70$ creating four clusters of different sizes. See text for more details.

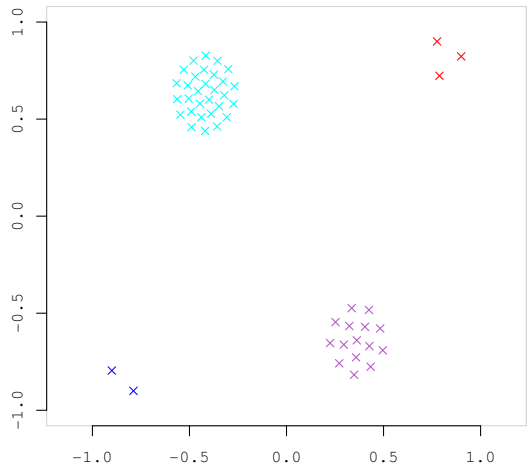

Fig. 5 Small coarse clusters, equation set **B**. Same data as in the previous Figure. Notice that influence of d_{ij} for longer distances is smaller (in comparison with the set **A**).

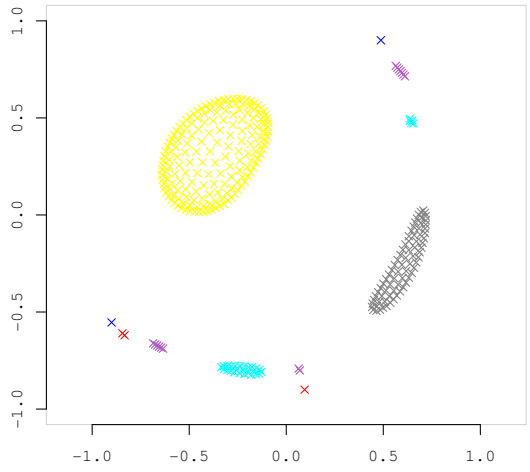

Fig. 6 Medium-sized coarse clusters, equation set **A**. Each cross represents a single organism in the population. Population size was $N = 300$ creating six clusters of different sizes. See text for more details. Notice that small clusters are broken apart.

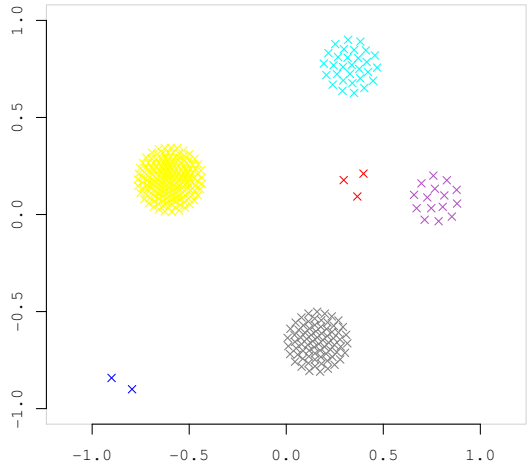

Fig. 7 Medium-sized coarse clusters, equation set **B**. Same data as in the previous Figure. Equation set **B** has better visualization capabilities as it is more resistant to disruption of small clusters.

tion size was $N = 100$. NEAT uses genotypic distance measure (for details see [4]).

Figures 11 and 12 show the visualization of D using equation set B for the first and for the last generation of the evolutionary run. In the first generation all solutions are randomized and the whole population forms a single cluster as expected, although similar solutions are nearby (as can be seen using color intensity corrections from Subsection 3.5). The last generation shown in Figure 12 clearly depicts distinct species that emerged through the evolution. Thus the visualization confirms

ISBN 978-3-901608-32-2

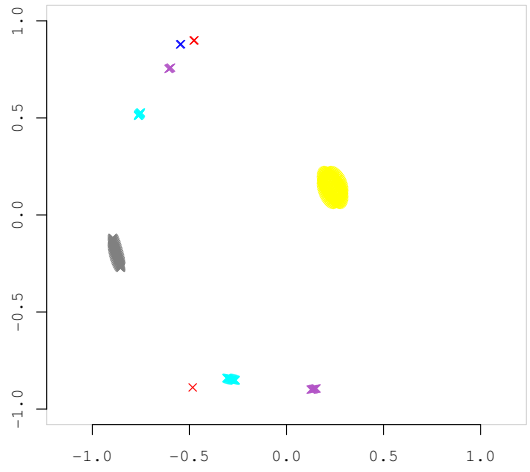

Fig. 8 Medium-sized coarse clusters, Sammon's projection. Same data as in the previous two Figures. Similarly to equation set **A**, Sammon's projection tends to break apart small clusters.

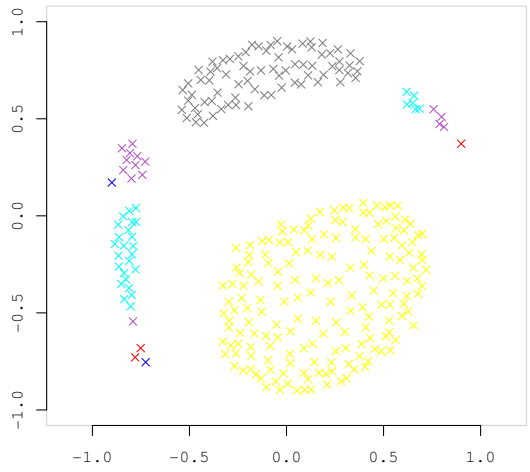

Fig. 9 Medium-sized randomized clusters, equation set **A**. Population size was $N = 300$ creating six clusters of different sizes with randomized inter- and intra-cluster distances.

that niching algorithm was correctly searching for more optima simultaneously. Comparing similar figures can bring much more information, e.g., species number, sizes or solution distribution inside species.

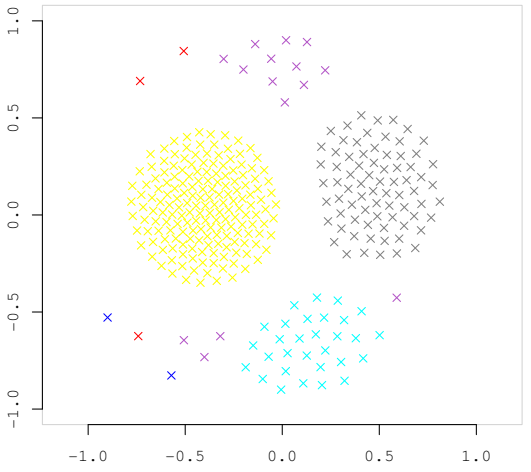

Fig. 10 Medium-sized randomized clusters, equation set **B**. Same data as in the previous Figure.

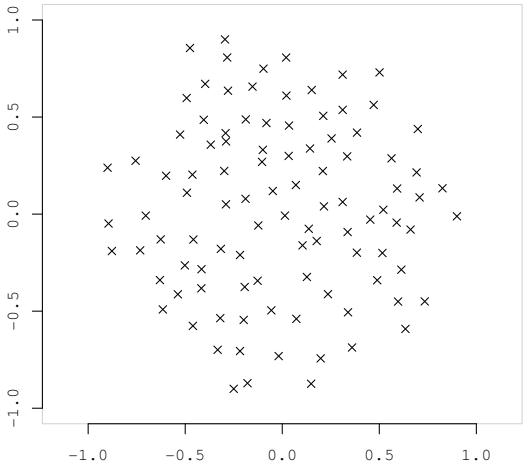

Fig. 11 NEAT XOR training first generation. Equation set **B**. In the first generation the population of EA is fully randomized forming only one large species which was visualized as a single cluster. Population size $N = 100$.

5.3 Projection of classification datasets

The use of our method is not limited only to visualization of diversity of EA populations. It can be used as a general dataset visualization method. We are currently making research in this area. One of our results can be seen in the Figure 13. It shows the visualization of a well-known Iris classification dataset [15]. All three classes (Setosa, Versicolor and Virginica) form separate

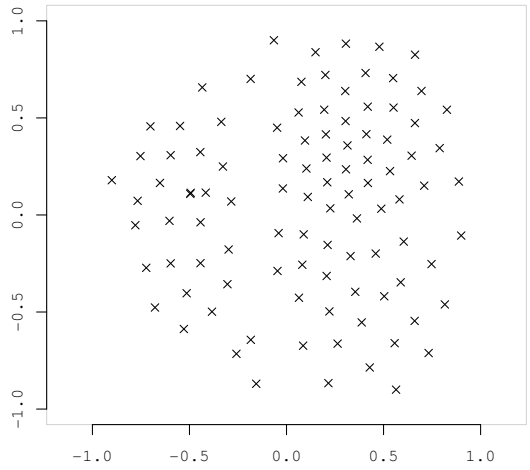

Fig. 12 NEAT XOR training last generation. Equation set **B**. During the evolutionary process the species were developed. They were visualized as different clusters. Population size $N = 100$.

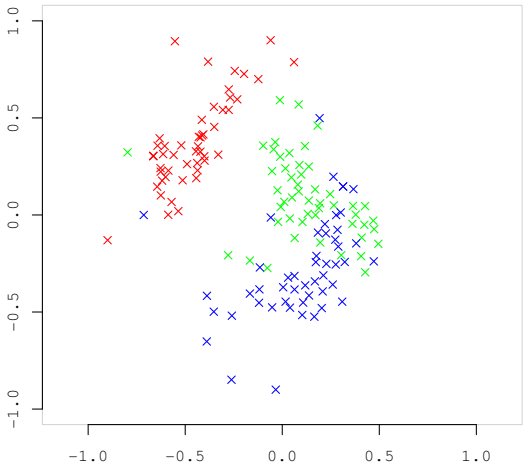

Fig. 13 Use of our method for a general classification dataset visualization - Iris dataset, equation set **B**. Three classes: Setosa (red), Versicolor (green) and Virginica (blue) form clusters. Versicolor and Virginica are known to be not linearly separable which is clearly reflected in the visualization (green and blue clusters overlap).

clusters. Versicolor and Virginica are known to be not linearly separable which is clearly reflected in the visualization.

5.4 Optimization algorithms comparison

This Subsection contains experiments with optimization algorithms described in Section 4. We have made tests on three sizes of population: small ($N = 50$), medium ($N = 300$) and large ($N = 1000$). That makes 100, 600 and 2000 dimensions to optimize. The data were generated as described in the Subsection 5.1. The optimization algorithms are compared using the average energy reached **AVG** (the lower the better), the number of functional value evaluations (energy) **FE** and the number of gradient evaluations (force) **GE**.

The comparison of optimization algorithms is done for the equation set **A**, equation set **B** and Sammon's projection separately. Originally Sammon's projection used only Steepest Descent with fixed step which is, as it will be shown, the worst possible choice.

The results on small populations can be found in tables 1, 2 and 3 for equations sets **A**, **B** and for Sammon's projecion. It is clear that the equation set **A** is the fastest of all, while equation set **B** is much slower than both equation set **A** and Sammon (on the other hand it has better visualization capabilities as it tends to keep smaller clusters unbroken).

Tables 4, 5 and 6 show the results for medium-sized populations. It can be seen that equations **A** speedup against Sammon's projection is even greater than on small populations.

In tables 7 and 8 the results for large populations are shown. Here, we ommited the equation set **B** because of the long computational times.

Tab. 1 Comparison of optimization algorithms on a small population. Equation set **A**. Parameter settings were $N = 50$, $C = \{2, 3, 15, 30\}$, $X = \langle 0.1, 0.1 \rangle$, $Y = \langle 0.9, 0.9 \rangle$. Tested on 100 runs for different randomly initialized **X**. **AVG** stands for average energy reached (the lower the better). **FE** for number of functional value evaluations and **GE** for gradient evaluations.

METHOD	AVG	FE	GE
CG	-4.4133	186.63	186.63
CG-PAL	-4.4137	79.22	179.78
QN-UNCMIN	-4.4138	97.91	96.25
SD	-4.4135	499.04	499.04
SDF	-4.1234	125377.74	125377.74

Tab. 2 Comparison of optimization algorithms on a small population. Equation set **B**.

METHOD	AVG	FE	GE
CG	1141.84	2843.82	2843.82
CG-PAL	-	-	-
QN-UNCMIN	1141.21	792.78	739.53
SD	1142.76	16392.78	16392.78
SDF	-	-	-

It is clear from the tables that Quasi-Newton (QN-UNCMIN) gave the unmatched results for the small

Tab. 3 Comparison of optimization algorithms on a small population. Sammon's projection.

METHOD	AVG	FE	GE
CG	8.96	347.47	347.47
CG-PAL	9.01	104.1	236.01
QN-UNCMIN	8.92	177.19	173.99
SD	9.08	827.34	827.34
SDF	9.29	3251.51	3251.51

Tab. 4 Comparison of optimization algorithms on a medium population. Equation set **A**. Parameter settings were $N = 300$, $C = \{2, 3, 15, 30, 75, 175\}$. Tested on 100 runs for different randomly initialized **X**.

METHOD	AVG	FE	GE
CG	-27.64	242.36	242.36
CG-PAL	-27.65	140.9	316.11
QN-UNCMIN	-27.64	50.84	48.02
SD	-27.62	663.92	663.92
SDF	-	-	-

Tab. 5 Comparison of optimization algorithms on a medium population. Equation set **B**, 10 runs.

METHOD	AVG	FE	GE
CG	59501.43	9501.6	9501.6
CG-PAL	-	-	-
QN-UNCMIN	59526.91	2945.3	2771.5
SD	59580.50	61352.4	61352.4
SDF	-	-	-

Tab. 6 Comparison of optimization algorithms on a medium population. Sammon's projection, 100 runs.

METHOD	AVG	FE	GE
CG	877.19	1544.39	1544.39
CG-PAL	888.17	430.49	945.31
QN-UNCMIN	887.76	155.65	155.65
SD	922.84	9398.82	9398.82
SDF	-	-	-

Tab. 7 Comparison of optimization algorithms on a large population. Equation set **A**. Parameter settings were $N = 1000$, $C = \{25, 25, 50, 100, 200, 600\}$. Tested on 20 runs for different randomly initialized **X**.

METHOD	AVG	FE	GE
CG	-91.71	239.15	239.15
CG-PAL	-91.75	142.1	322.65
QN-UNCMIN	-	-	-
SD	-91.65	564.35	564.35
SDF	-	-	-

population in terms of function and gradient evaluations. However, the Quasi-Newton algorithm had a big overhead which made it unusable for larger populations due to unacceptable computational time. Both Conjugated Gradient methods gave stable results for all sizes of population, except for equation set **B** where CG-PAL was too slow. Steepest Descent method with line search (SD) performed well on small populations. However, it became too slow for medium and large populations. SDF was found quite useful when constructing the energy and force equations, which was done on small populations - we were able to animate the optimization process and tune parameter settings intuitively. We have also performed experiments with populations of $N > 2000$; it turned out that only Conjugated Gradient methods were able to find reasonable solution.

Tab. 8 Comparison of optimization algorithms on a large population. Sammon's projection, 100 runs.

METHOD	AVG	FE	GE
CG	12544.59	7682.4	7682.4
CG-PAL	12053.91	928.8	2030.85
QN-UNCMIN	-	-	-
SD	12674.16	24551.05	24551.05
SDF	-	-	-

6 Conclusion

This article proposes a novel method of diversity visualization in evolutionary algorithms based on modeling of intermolecular forces. We have created two sets of energy equations (sets **A** and **B**) to simulate the system. and compared various optimization methods for minimization of energy of the system. We recommend to use Conjugated Gradients methods (CG or CG-PAL) as they worked best for all sizes of populations. Other experiments have shown visualization properties on both artificial and real-world data.

We have also shown that the use of this method is not limited to diversity visualization. It is able to visualize any clustered data or, in other words, any multidimensional data with defined distances (similarity measure).

The comparison with Sammon's projection method showed that our equation set **A** is faster to optimize (the speedup is more evident for larger populations). On the other hand the proposed equation set **B** gives better visualization results as it does not tend to disrupt the smaller clusters.

There are many areas to be explored. First, it is the creation of a new set of equations overcoming the problem of small cluster disruption which could be optimized faster than equations **B**. It might be also possible to show the whole evolutionary process from the first generation to the last (by adding the time axis thus making the visualization 3-D).

Acknowledgments
This research is partially supported by the grant Automated Knowledge Extraction (KJB201210701) of the Grant Agency of the Academy of Sciences of the Czech Republic, the research program "Transdisciplinary Research in the Area of Biomedical Engineering II" (MSM6840770012) sponsored by the Ministry of Education, Youth and Sports of the Czech Republic and

by the CTU IGS under grant CTU0707013.

7 References

[1] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, 1987.

[2] S.W. Mahfoud. Crowding and preselection revisited. *Evolutionary Computation*, 2(1):37–66, 1992.

[3] S.W. Mahfoud. A comparison of parallel and sequential niching methods. In Morgan Kaufmann, editor, *Proceedings of the Sixth ICGA*, pages 136–143, 1995.

[4] K.O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. Technical Report TR-AI-01-290, Department of Computer Sciences, The University of Texas at Austin, 2001.

[5] W.J. Moore. *Physical Chemistry*. Prentice Hall, New York, USA, 1972.

[6] J.W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[7] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer-Verlag, New York, USA, 1999.

[8] L.H. Tsoukalas and R.E. Uhrig. *Fuzzy and neural aproaches in engineering*. John Wiley & Sons, New York, USA, 1997.

[9] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: the art of scientific computing 2nd ed.* Cambridge University Press, Cambridge, 1992.

[10] M. Al-Baali and R Fletcher. An efficient line search for nonlinear least squares. *J. Optim. Theory Appl.*, 48(3):359–377, 1986.

[11] R. Fletcher. *Practical Methods of Optimization Vol.1: Unconstrained Oprimization*. John Wiley & Sons, New York, USA, 1980.

[12] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. Springer-Verlag, Berlin, Heidelberg, Germany, 2003.

[13] PAL: Phylogenetic analysis library. http://www.cebl.auckland.ac.nz/pal-project/index.html.

[14] R.B. Schnabel, J.E. Koontz, and B.E. Weiss. A modular system of algorithms for unconstrained minimization. *ACM Transactions on Mathematical Software*, 11(4):419–440, December 1985.

[15] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

