Comparative Analysis of Various Evolutionary and Memetic Algorithms

Krisztián Balázs[1], János Botzheim[2], László T. Kóczy[1,3]

[1] Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Hungary
[2] Department of Automation, Széchenyi István University, Győr, Hungary
[3] Institute of Informatics, Electrical and Mechanical Engineering, Faculty of Engineering Sciences, Széchenyi István University, Győr, Hungary
E-mail: {balazs, koczy}@tmit.bme.hu, {botzheim, koczy}@sze.hu

Abstract: Optimization methods known from the literature include gradient techniques and evolutionary algorithms. The main idea of gradient methods is to calculate the gradient of the objective function at the actual point and then to step towards better values according to this value. Evolutionary algorithms imitate a simplified abstract model of evolution observed in nature. Memetic algorithms traditionally combine evolutionary and gradient techniques to exploit the advantages of both methods. Our current research aims to discover the properties, especially the efficiency (i.e. the speed of convergence) of particular evolutionary and memetic algorithms. For this purpose the techniques are compared by applying them on several numerical optimization benchmark functions and on fuzzy rule base identification.

Keywords: evolutionary algorithms, memetic algorithms, fuzzy rule based learning

1 Introduction

The scope of engineering applications based on soft computing methods is continuously expanding in the field of complex problems, because of their favorable properties. Evolutionary computation (and evolutionary based, e.g. memetic) methods form a huge part of these techniques. However, both theory and application practice still contain many unsolved questions, hence researching the theory and applicability of these methods is obviously an important and actual task. As the results of the investigations on applicability mean some kind of labeling for the involved methods, there are two outcomes of these investigations. One is the fact that they result in practical knowledge for industrial users concerning which techniques offer better possibilities and which ones are worth to be selected for integration into their respective products. The other one is feedback to the researchers regarding to in which direction they should continue their work.

Our work aims at the investigation of such methods. Evolutionary computation algorithms are numerical optimization techniques, so their efficiency can be characterized by the speed of convergence to the global optimum. Since so far there have not been invented any methods to obtain this property exactly, it can be figured out mostly by simulation. Therefore this investigation is based on simulation carried out using a modular software system that we implemented in C language, introduced in [1] and discussed deeper in [2].

It contains two larger units: a machine learning frame and a main optimization module. Thus the system is able to deal with both optimization and machine learning problems.

The learning frame implements fuzzy rule based learning with two inference methods, one using dense while the other one spare rule bases. The former method is called Mamdani-inference [3] and the latter one is the stabilized KH-interpolation technique [4], [5].

The optimization main module contains various sub-modules, each one implementing an optimization method, such as steepest descent [6] and Levenberg-Marquardt [7], [8] from the family of gradient techniques, furthermore genetic algorithm [9] and bacterial evolutionary algorithm [10] both being evolutionary methods. Obviously, memetic techniques [11] are also available in the software as the combination of the previous four algorithm types.

The methods have been compared by their respective performance on various optimization benchmark functions (that are typically used in the literature to 'evaluate' global optimization algorithms) and on machine learning problems.

Although many results have been published comparing particular evolutionary and memetic techniques (see e.g. [10], [12]), these discussions considered only a few methods and mainly focused on the convergence of the algorithms in terms of number of generations. However, different techniques have very differing computational demands. This difference is sometimes two or three orders of magnitude in computational time. Therefore the question arises: what is the relation of these methods compared to each other in terms of time? This has been set as the main question of this research.

Actually, our work is far from being complete, because we definitely have not implemented and compared all optimization and inference methods that can be found in the literature. This paper first of all tries to give a concept how such comparative investigations can be carried out.

The next section gives a brief overview of the algorithms and techniques used. After that, the benchmark functions and machine learning problems will be described, that are applied in the simulations. The simulation results and the observed behavior will be discussed in the fourth section. Finally, we summarize our work and draw some conclusions.

2 Overview of the Algorithms and Techniques Used

In order to carry out this investigation, it is necessary to overview two related theoretical topics and to point at the connection between them. One of these is numerical optimization and the other one is supervised machine learning.

The following subsections aim to give a brief overview of some important points of these theoretical aspects, which will be referred to later repeatedly in the paper.

2.1 Numerical Optimization

Numerical optimization [6] is a process, where the (global) optimum of an objective function $f_{obj}(\mathbf{p})$ is being searched for by choosing the proper variable (or parameter) vector $\mathbf{p}$. The optimum can be the maximum or the minimum of the objective function depending on the formulation of the problem.

There are several deterministic techniques as well as stochastic algorithms for optimization. Some of them will be presented below; these are the ones that were investigated in our work.

2.1.1 Gradient Methods

A family of iterative deterministic techniques is called gradient methods. The main idea of these methods is to calculate the gradient of the objective function at the actual point and to step towards better (greater if the maximum and smaller if the minimum is being searched) values using it by modifying $\mathbf{p}$. In case of advanced algorithms additional information about the objective function may also be applied during the iterations.

One of the most frequently used methods of this type is the steepest descent algorithm (SD) [6]. Each of its iterations contains the following steps: the gradient vector is computed, it is multiplied with a so-called bravery factor and finally, it is added (in case of searching for the maximum) or subtracted (in case of searching for the minimum) from the actual position to obtain the new position. If the gradient vector function is given, the vector can be obtained by calling this function, otherwise by a pseudo-gradient computation.

A more advanced and effective technique is the Levenberg-Marquardt algorithm (LM) [7], [8]. The steps applied during the iterations are based on a Jacobian matrix. Each row of this matrix contains the gradient vector of the so-called residual functions, whose sum of squares is being minimized. If the Jacobian matrix computing function is given, the matrix can be obtained by calling this function, otherwise by a pseudo-Jacobian computation.

After a proper amount of iterations, as a result of the gradient steps, the algorithms find the nearest local minimum quite accurately. However, these techniques are

very sensible to the location of the starting point. In order to find the global optimum, the starting point must be located close enough to it, in the sense that no local optima separate these two.

2.1.2 Evolutionary Computation Methods

A family of iterative stochastic techniques is called evolutionary algorithms. These methods, like the genetic algorithm (GA) [9] or the bacterial evolutionary algorithm (BEA) [10], imitate the abstract model of the evolution observed in the nature. Their aim is to change the individuals in the population by the evolutionary operators to obtain better and better ones. The goodness of an individual can be measured by its 'fitness'. If an individual represents a solution for a given problem, the algorithms try to find the optimal solution for the problem. Thus, in numerical optimization the individuals are potentially optimal parameter vectors and the fitness function is a transformation of the objective function. If an evolutionary algorithm uses an elitist strategy, it means that the best ever individual will always survive and appear in the next generation. As a result, at the end of the algorithm the best individual will hold the (quasi-) optimal values for $\mathbf{p}$, i.e. the best individual will represent the (quasi-) optimal parameter vector.

2.1.3 Memetic Algorithms

Evolutionary computation techniques explore the whole objective function, because of their characteristic, so they find the global optimum, but they approach to it slowly, while gradient algorithms find only the nearest local optimum, however, they converge to it faster.

Avoiding the disadvantages of the two different technique types, evolutionary and gradient methods may be combined [11], [12], for example, if in each iteration for each individual some gradient steps are applied. Expectedly, this way the advantages of both gradient and evolutionary techniques can be exploited: the local optima can be found quite accurately on the whole objective function, i.e. the global optimum can be obtained quite accurately.

There are several results in the literature confirming this expectation in the following aspect. Usually, the more difficult the applied gradient step is, the higher convergence speed the algorithm has in terms of number of generations. It must be emphasized, that most often these results discuss the convergence speed in terms of number of generations. However, the more difficult an algorithm is, the greater computational demand it has, i.e. each iteration takes longer.

Therefore the question arises: how does the speed of the convergence change in terms of time if the gradient technique applied in the method is changed?

Apparently, this is a very important question of applicability, because in real world applications time as a resource is a very important and expensive factor, but the number of generations the algorithm executes does not really matter.

This is the reason why the efficiency in terms of time was chosen to be investigated in this paper.

Our work considers six algorithms:

- Genetic algorithm, GA (without gradient steps)
- Genetic steepest descent, GSD (GA using SD steps)
- Genetic memetic algorithm, GMA (GA using LM steps)
- Bacterial evolutionary algorithm, BEA (without gradient steps)
- Bacterial steepest descent, BSD (BEA using SD steps)
- Bacterial memetic algorithm, BMA (BEA using LM steps)

2.2 Supervised Machine Learning

Supervised machine learning [13] means a process where parameters of a 'model' are being adjusted so that its behavior becomes similar to the behavior of the 'system', which is to be modeled. Since the behavior can be characterized by input-output pairs, the aim of the learning process can be formulated so that the modeling system should give similar outputs for the input as the original system does.

The model can be for example a simple function (e.g. a polynomial function), where the parameters are the coefficients, or it can be a neural network, where the parameters are the weights, or it can be a fuzzy rule base together with an inference engine [14]. In this case the parameters can be the characteristic points of the membership functions of the rules in the rule base. In our work we applied a fuzzy rule base combined with an inference engine using both Mamdani-inference [3] and stabilized KH-interpolation techniques [4], [5].

If a function $\phi(\mathbf{x})$ denotes the system and $f(\mathbf{x},\mathbf{p})$ denotes the model, where $\mathbf{x}\in\mathbf{X}$ is the input vector and $\mathbf{p}$ is the adjustable parameter vector, the previous requirement can be expressed as follows:

$$\forall \mathbf{x} \in \mathbf{X} : \phi(\mathbf{x}) \approx f(\mathbf{x}, \mathbf{p})$$

In a supervised case the learning happens using a set of training samples (input-output pairs). If the number of samples is m, the input in the i^{th} sample is $\mathbf{x}_i$, the desired output is $d_i = \phi(\mathbf{x}_i)$ and the output of the model is $y_i = f(\mathbf{x}_i,\mathbf{p})$, the following formula can be used:

$$\forall i \in [1, m] : d_i \approx y_i$$

The error (ε) shows how similar the modeling system to the system to model is. (See figure 1.) It is the function of the parameter vector, so it can be denoted by $\varepsilon(\mathbf{p})$. Let us use a widely applied definition for the error, the Mean of Squared Errors (MSE):

$$\varepsilon(\mathbf{p}) = \frac{\sum_{i=1}^{m} (d_i - y_i)^2}{m}$$

Obviously, the task is to minimize this $\varepsilon(\mathbf{p})$ function. It can be done by numerical optimization algorithms.

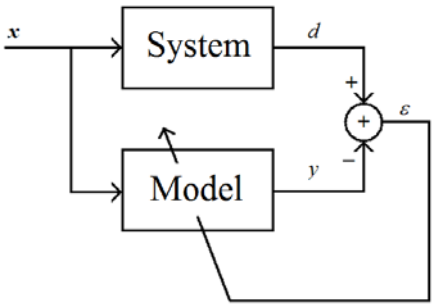

Figure 1
Supervised machine learning

This way machine learning problems can be traced back to optimization problems, furthermore they can be applied to discover the efficiency of evolutionary and memetic algorithms.

3 Benchmark Functions and Machine Learning Problems

The optimization benchmark functions and machine learning problems, which were used during our investigations, will be described in this section.

In case of the benchmark functions the minimum value, in case of the learning problems the minimum error is searched.

3.1 Benchmark Functions

In our investigations five benchmark functions were applied: Ackley's [15], Keane's [16], Rastrigin's [17], Rosenbrock's [18] and Schwefel's function [19]. These functions are widely used in the literature to evaluate global optimization methods, like evolutionary techniques. They are generic functions according to dimensionality, i.e. the number of dimensions of these functions can be set to an arbitrary positive integer value. In our simulations this value was set to 30, because it is a typical value in the literature for these functions.

Ackley's, Keane's, Rastrigin's and Schwefel's functions are multimodal, i.e. they have more than one local optima (actually they have a number of local optimum).

Rastrigin's function is separable as well as Schwefel's function. This means that the minimization along each dimensions result the minimum.

For example Ackley's benchmark function is given as follows (*k* denotes the number of dimensions):

$$f_{Ack}(\mathbf{x}) = 20 + e - 20e^{\left(-0.2\sqrt{\frac{1}{k}\sum_{i=1}^{k} x_i^2}\right)} - e^{\left(\frac{1}{k}\sum_{i=1}^{k}\cos(2\pi x_i)\right)}, \quad \forall x_i \in [-30,30]$$

3.2 Machine Learning Problems

In our investigations three machine learning problems were applied: the one dimensional pH [12], the two dimensional ICT [12] and a six dimensional problem that was also used by Nawa and Furuhashi to evaluate the performance of bacterial evolutionary algorithm [10].

For example this six dimensional function is defined as follows:

$$f_{6\dim} = x_1 + \sqrt{x_2} + x_3 x_4 + 2e^{2(x_5 - x_6)},$$
$$x_1, x_2 \in [1,5], \quad x_3 \in [0,4], \quad x_4 \in [0,0.6], \quad x_5 \in [0,1], \quad x_6 \in [0,1.2]$$

4 Results and Observed Behavior

In the simulations the parameters had the following values. The number of rules in the rule base was 4, the number of individuals in a generation was 14. In case of genetic techniques the selection rate was 0.3 and the mutation rate was 0.1, in case of bacterial techniques the number of clones was 5 and 4 gene transfers were carried out in each generation. The genetic methods applied elitist strategy. In case of memetic algorithms 8 iteration long gradient steps were applied. The gradient vector and the Jacobian matrix computing functions were not given, hence

pseudo-gradients and pseudo-Jacobians were computed where steepest descent or Levenberg-Marquardt gradient steps were used.

The numbers of training samples were between 100 and 200 in the learning processes.

During the runs the fitness values of the best individuals were monitored in terms of time. This fitness value was calculated based on the MSE value (measured on the training samples) as follows:

$$F = \frac{10}{MSE + 1} = \frac{10m}{\sum_{i=1}^{m} (d_i - y_i)^2 + m}$$

In case of all algorithms for all benchmark functions and learning problems 10 runs were carried out. Then we took the mean of the obtained values. These means were presented in figures, to get a better overview. The horizontal axes show the elapsed computation time in seconds and the vertical axes show the fitness values of the best individuals at the current time.

On the figures (see later) dashed lines show the result of the pure evolutionary algorithms (GA and BEA), dotted lines denote the techniques using steepest descent gradient steps and solid lines present the graphs of methods using Levenberg-Marquard technique.

The results of the runs and their short explanations follow in the next subsections. Every simulation will not appear though, because their great number does not allow it, rather some example results will be presented in the next three subsections. The results for the other optimization benchmark functions and learning problems are mostly, but not always similar, therefore these examples present only a behavior that were observed most often.

In subsection 4.4 conclusions will be drawn about the behavior of the methods considering all of the simulations.

4.1 Results for Ackley's Benchmark Function

This example presents the performance of the compared techniques on Ackley's benchmark function.

Figure 2 shows the fitness values of the best individuals in terms of time.

As it can be observed, bacterial algorithms gave better results than genetic techniques. Actually bacterial methods found the global optimum (the 10 fitness value indicates this fact). At the beginning BEA and BSD were better than BMA, but after an adequate time BMA was not worse than any other algorithm.

Bacterial techniques were better than the corresponding genetic methods (i.e. BEA was better than GA, BSD was better than GSD, etc.).

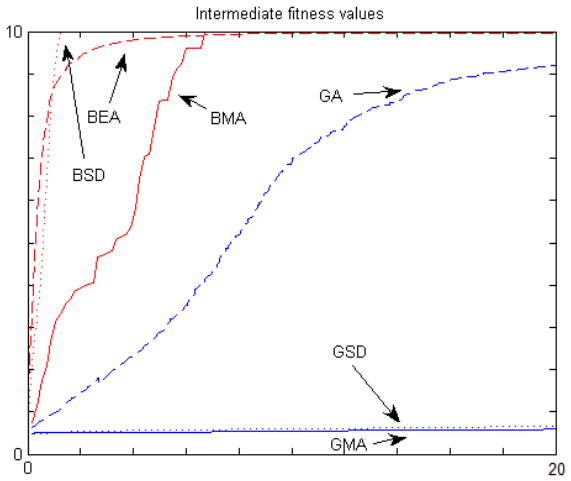

Figure 2
Results for Ackley's benchmark function

4.2 Results in Case of Applying Mamdani-Inference-based Learning for the ICT Problem

This example presents the performance of the compared techniques on the two dimensional ICT problem, when Mamdani-inference based learning was applied.

Like in the previous case, there is an adequate time again from when BMA was not worse than any other technique; however at the beginning BEA was better.

At the end GMA gave the second best result. Thus it was better than any other genetic algorithms (see Figure 3).

The methods using steepest descent gradient steps were the worst among both bacterial and genetic algorithms.

Again, bacterial techniques were always better than the corresponding genetic methods.

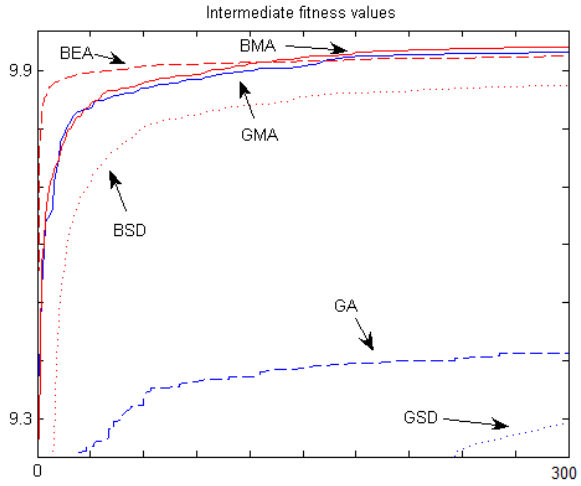

Figure 3
Results for the ICT learning problem applying Mamdani-inference based learning

4.3 Results in Case of Applying Stabilized KH-interpolation-based Learning for the Six Dimensional Problem

This example presents the performance of the compared techniques on the six dimensional learning problem, when stabilized KH-interpolation based learning was applied.

In this case from the beginning BMA and GMA gave the best fitness values, and there is also a time limit from when BMA was the best of all algorithms.

At the end GMA gave the second best results, thus it was better than any other genetic algorithm (see Figure 4).

The methods using steepest descent gradient steps were the worst among both bacterial and genetic algorithms.

Bacterial techniques were better than the corresponding genetic methods.

4.4 Observed behavior

Based on the simulation results we observed the following behaviors:

- Generally, bacterial techniques seemed to be better than the corresponding genetic methods.
- Generally, for learning problems GMA had the highest convergence speed among genetic algorithms.

- Usually (but not always), after a sufficient time, BMA was not worse than any other algorithms. The more difficult the problem is, the better the advantage of the technique appears.

It might be said that BMA advances 'slowly but surely' to the optimum. 'Slowly', because in most of the cases at the beginning it did not have the highest convergence speed. 'Surely', because during the simulations it did not lose so much from its efficiency than the other techniques.

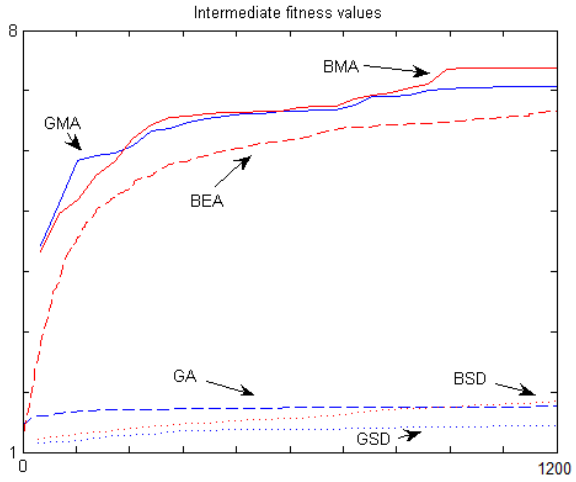

Figure 4
Results for the 6 dimensional learning problem applying stabilized KH-interpolation based learning

Conclusions

In our work various evolutionary and memetic algorithms have been compared on general optimization benchmark functions and on machine learning problems using a software that was developed by ourselves and that can be found in the referred works.

After we carried out simulation runs and investigated the obtained results, we drew some conclusions about the behavior of the various techniques compared to each other. They can be shortly summed up as follows. Generally, bacterial techniques seemed to be better than the corresponding genetic methods. Usually, algorithms applying LM technique seemed to be better for learning problems than methods not using gradient steps or using SD.

To reinforce these tendencies, as a continuation of this work, carrying out more simulations is necessary. Further research may aim to compare other global optimization algorithms that can be found in the literature.

Acknowledgements

This paper was supported by the National Scientific Research Fund Grant OTKA K75711, a Széchenyi István University Main Research Direction Grant and the Social Renewal Operation Programme TÁMOP–4.2.2 08/1–2008–0021.

References

[1] K. Balázs, L. T. Kóczy, J. Botzheim: Comparison of Fuzzy Rule-based Learning and Inference Systems, Proceedings of the 9th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, CINTI 2008, Budapest, Hungary, 2008, pp. 61-75

[2] K. Balázs: Comparative Analysis of Fuzzy Rule-based Learning and Inference Systems, Master's Thesis, Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary, 2009, 97 p.

[3] E. H. Mamdani: Application of Fuzzy Algorithms for Control of Simple Dynamic Plant, IEEE Proc., Vol. 121, No. 12, 1974, pp. 1585-1588

[4] L. T. Kóczy, K. Hirota: Approximate Reasoning by Linear Rule Interpolation and General Approximation, Internat. J. Approx. Reason., 9, 1993, pp. 197-225

[5] D. Tikk, I. Joó, L. T. Kóczy, P. Várlaki, B. Moser, T. D. Gedeon: Stability of Interpolative Fuzzy KH-Controllers, Fuzzy Sets and Systems, 125, 2002, pp. 105-119

[6] Jan A. Snyman: Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-based Algorithms, Springer, New York, 2005

[7] K. Levenberg: A Method for the Solution of Certain Non-Linear Problems in Least Squares. Quart. Appl. Math., 2(2), 1944, pp. 164-168

[8] D. Marquardt: An Algorithm for Least-Squares Estimation of Nonlinear Parameters. J. Soc. Indust. Appl. Math., 11(2), 1963, pp. 431-441

[9] J. H. Holland: Adaption in Natural and Artificial Systems, The MIT Press, Cambridge, Massachusetts, 1992

[10] N. E. Nawa, T. Furuhashi: Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm, IEEE Transactions on Fuzzy Systems, 7(5), 1999, pp. 608-616

[11] P. Moscato: On evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989

[12] J. Botzheim, C. Cabrita, L. T. Kóczy, A. E. Ruano: Fuzzy Rule Extraction by Bacterial Memetic Algorithms, Proceedings of the 11th World Congress

of International Fuzzy Systems Association, IFSA 2005, Beijing, China, 2005, pp. 1563-1568

[13] E. Alpaydin: Introduction to Machine Learning, The MIT Press, ISBN 978-0-262-01211-9, 2004, 445 p.

[14] D. Driankov, H. Hellendoorn, M. Reinfrank: An Introduction to Fuzzy Control, Springer, New York, 316 p.

[15] D. Ackley: An Empirical Study of Bit Vector Function Optimizacion, Genetic Algorithms and Simulated Annealing, 1987, pp. 170-215

[16] A. Keane: Experiences with Optimizers in Structural Design, in Parmee, IC (ed) Proceedings of the 1st Conf. on Adaptive Computing in Engineering Design and Control, University of Plymouth, UK, 1994, pp. 14-27

[17] L. A. Rastrigin: Extremal Control Systems, Theoretical Foundations of Engineering Cybernetics Series, Moscow, Russian, 1974

[18] H. H. Rosenbrock: An Automatic Method for Finding the Greatest or Least Value of a Function, Computer Journal, (3), 1960, pp. 175-184

[19] H. P. Schwefel: Numerical Optimization of Computer Models, John Wiley & Sons, 1981, English translation of Numerische Optimierung von Computer-modellen mittels der Evolutionsstrategie, 1977