ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/287427201

Population and Gradient Based Optimization Techniques, a Theoretical Overview

Article · October 2014

DOI: 10.14513/actatechjaur.v7.n4.342

CITATION
1

READS
142

2 authors:

Gergely Friedl
Széchenyi István University, Gyor
7 PUBLICATIONS **12** CITATIONS
SEE PROFILE

Miklos Kuczmann
Széchenyi István University, Gyor
108 PUBLICATIONS **472** CITATIONS
SEE PROFILE

All content following this page was uploaded by Miklos Kuczmann on 01 August 2016.

The user has requested enhancement of the downloaded file.

Acta
Technica
Jaurinensis

Vol. 7., No. 4., pp. 378–387, 2014
DOI: 10.14513/actatechjaur.v7.n4.342
Available online at acta.sze.hu

Population and Gradient Based Optimization Techniques, a Theoretical Overview

G. Friedl, M. Kuczmann

Széchenyi István University, Department of Automation
9026, Győr, Hungary
E-mail: friedl@maxwell.sze.hu

Abstract: Many practical problems can be modeled only as a nonlinear continuous global optimization problem. It is usually impossible and impractical to solve them exactly. Evolutionary and hybrid algorithms are modern techniques to find optima for complex search spaces. This paper is a short summary about the optimization techniques, especially about the evolutionary based global searching algorithms, and the gradient based local searching algorithms. After the introduction, the second chapter gives a brief summary about population based techniques, especially about genetic and bacterial evolutionary algorithm, and particle swarm optimization. The next chapter discusses two gradient based searching technique, the *steepest descent* and the *Levenberg-Marquardt* method. This paper is a theoretical overview of the basics of my future Ph.D. dissertation.

Keywords: *optimization, soft computing, evolutionary algorithms, memetic algorithms*

1. Introduction

Soft computing techniques are characterised to provide an inexact solution to computationally hard problems, for which there is no known algorithm, what could calculate the exact solution, and the computation time is non-deterministic. Evolutionary algorithm is a branch of them, and represents many type of global searching techniques. This optimization technique family is named *evolutionary*, because the ideas behind them are drawn from the biological evolution. It is a population (or multi-population) based metaheuristic optimization algorithm. For a problem in analytically indescribable search field with limited computation time, metaheuristics provide a suitable result, a so-called quasi-optimum.

These methods are useful to find the area of global maximum (or minimum) of the search field, but the convergence becomes slower, when the task is to find the local maximum (or minimum) of this area. There are many searching algorithms, which are developed to find

these points on an unknown function as fast, as it is possible. These methods are called gradient based methods, e.g. the steepest descent method, or the Levenberg-Marquardt method. These methods are very useful to find local extreme values within a reasonable time, but if they founded this point, they can not find an another, maybe better point in the search field, these algorithms get stuck at the local extreme values.

To take the advantage of these two types of searching algorithms, hybrid, so-called memetic algorithms were developed. These algorithms use population based algorithms to find a point next to the global maximum or minimum, then apply the gradient based algorithms to achieve fast convergence.

2. Population based searching algorithms

Population based algorithms, like genetic algorithm [1], bacterial evolutionary algorithm [2, 3] and particle swarm optimization [4] are developed to find global quasi-optimum of any complex search field. They are all inspired by a natural phenomena. These algorithms operate with a population of possible solutions of the search field. These solutions are called *individuals*. The population is then arranged in pairs as parents, the children or the mutation of them may be a better solution than the parents. The goodness of these individuals are described by their fitness function. The fitness function is an objective function, that summarises the properties of an individual in a single number. There is no unified description of the fitness function, because its definition is hardly depends on the type of the task.

2.1. Genetic algorithm

The first evolutionary type, and one of the most applied method is the genetic algorithm [1]. This technique imitates the process of natural selection by modeling the crossover of individuals and randomly occurring mutations. Every individual has a chromosome, that represents a set of properties. A property can be a bit, a numeric value, a color, etc. During the evolution, these chromosomes are continuously changing to create better and better individuals. The genetic algorithm has five main steps [5, 6]. These are the following:

Initialization: Creating an initial population of randomly chosen individuals in the search field. This step has to be done only once at the beginning of the algorithm.

Selection: Sorting the individuals according to their fitness value, then selecting some of them. The chance of the selection is proportional to the fitness value. The selected ones are called as *parents*.

Crossover: Arranging the selected part of the population in pairs. For each pair a random point of their chromosome is selected, and the parents change every properties between each other after this point (see in Fig. 1). The newly created individuals

are called *offspring*. There are modified algorithms, that work with two or more point crossover, e.g. in the case of two point crossover the changed part of the chromosomes between these two point.

Mutation: Changing a random property of the chromosomes of each offspring with small probability (see in Fig. 2). The new, mutated value of this point is chosen randomly.

Substitution: Substituting some selected members of the population by the offspring. The chance of the selection is inversely proportional to their fitness value. The *strong* individuals survive with a bigger chance. If there is no acceptably good individual, or the iteration/time limit is not reached, the algorithm steps back to the *selection* step.

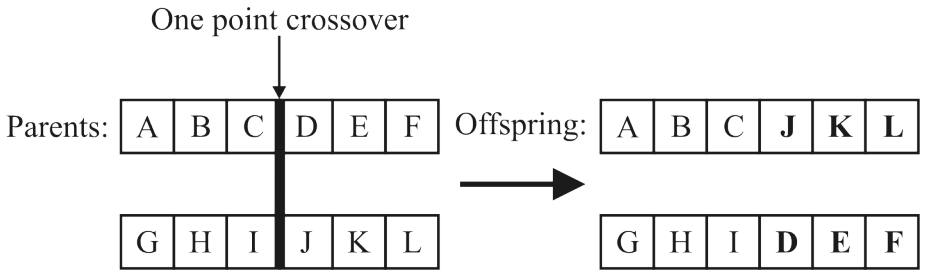

Figure 1. One point crossover.

There are different methods to execute the selection step. The two most widely used methods are the *roulette wheel technique* and the *stochastic universal sampling* [6]. If the so-called *elitist strategy* is implemented, the best individual always survives.

The genetic algorithm finds the area of global optimum very effectively. On the other side, the algorithm is very time consuming, the calculation of the fitness function can be very difficult and the convergence of the algorithm gets slower.

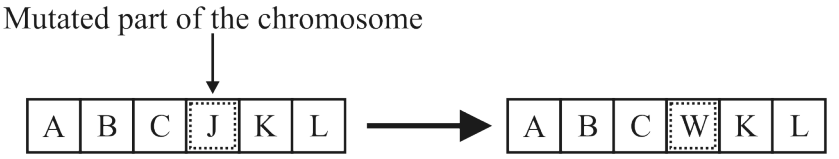

Figure 2. Mutation of a single property of the chromosome of an individual.

2.2. Bacterial evolutionary algorithm

Another widely used evolutionary type searching method is the bacterial evolutionary algorithm[2, 3], which is based on the microbial evolution. There are two main difference between the bacterial and genetic algorithm:

1. The *crossover* and *selection* step is substituted by a new operator called gene transfer.
2. The *mutation* step is modified as bacterial mutation.

The implementation of this method is easier than the genetic algorithm, but usually it is also more efficient. The bacterial evolution algorithm has three main steps [5, 7]. These are the following:

Initialization: Creating an initial population of randomly chosen individuals in the search field. This step has to be done only once at the beginning of the algorithm.

Bacterial mutation: Mutating every property of every bacteria in a random order even multiple times. If the original bacterium is a better solution to the problem, then it is restored, otherwise the new individual substitutes the original (see in Fig. 3).

Gene transfer: Separating the population into two groups according to the fitness values (superior and inferior group), arranging the members in pairs (one from the superior and one from the inferior group), then transferring genes from the superior individual to the inferior individual.

The usage bacterial evolutionary algorithm is also an effective way to find quasi-optimum. This method usually faster, than the genetic algorithm, but the convergence also gets slower, when the actual best result is close to the global optimum. This slowing is caused by the randomness of the evolution.

2.3. Particle swarm optimization

The idea behind the particle swarm optimization method is the social behavior of bird flocking or fish schooling [4]. The individuals are called as *particles*, and the population is called as a *swarm*. The particles placed randomly on the search field are continuously moving. The direction of moving of each particle depends on their actual position, the location of their *local best value*, and the location of the *global best value*. Every particle has its own local best place, which is the place where its fitness value was the best during the moving. The global best place is the place of the best local best place. These particles can be ordered in different topologies. These topologies describe, which other particles will affect the moving direction.

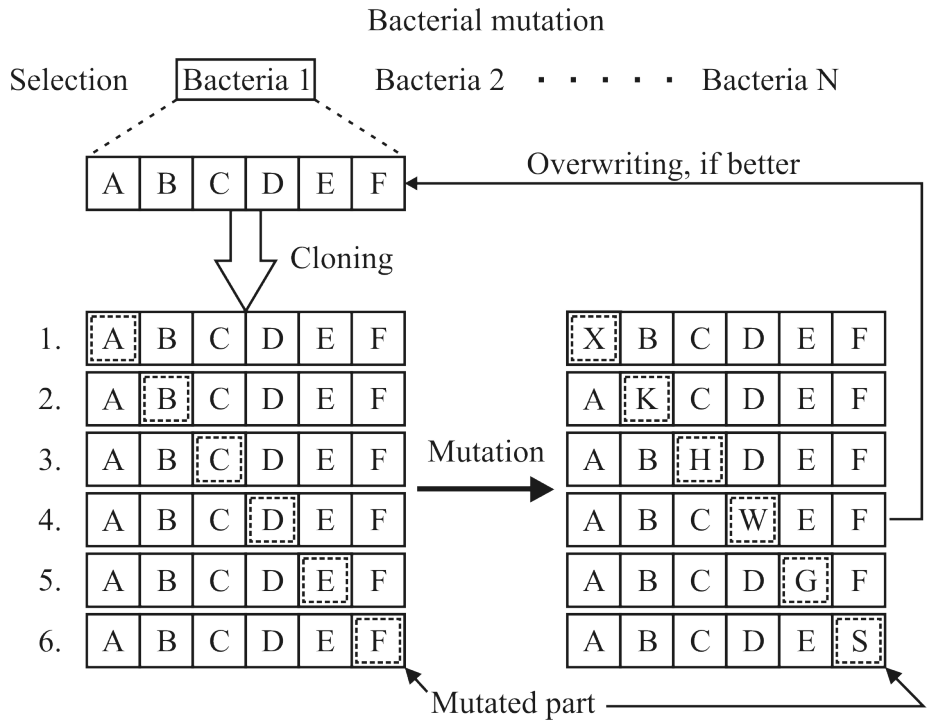

Figure 3. A step of the bacterial mutation

In the simplest case, the particle swarm optimization method has three main step. These are the following[5, 8]:

Initialization: Creating an initial swarm (or multi-swarm) placing particles in random positions.

Moving: The particles are moving in directions affected by their local best values $\boldsymbol{l}$ and the global best value $\boldsymbol{g}$. If i is the iteration number, and the values of the set of parameters at actual location of a particle is denoted as $\boldsymbol{x}_i$, the direction of the moving, and the next place of the particle can be determined by the following equation:

$$\begin{aligned} \boldsymbol{d}_0 &= 0, \\ \boldsymbol{d}_{i+1} &= \varphi_d \boldsymbol{d}_i + \varphi_l r_l (\boldsymbol{l}_i - \boldsymbol{x}_i) + \varphi_g r_g (\boldsymbol{g}_i - \boldsymbol{x}_i), \\ \boldsymbol{x}_{i+1} &= \boldsymbol{x}_i + \boldsymbol{d}_{i+1}, \end{aligned} \tag{1}$$

where φ_d, φ_l and φ_g are parameters, r_l and r_g are random values. An explanation for this step can be seen in Fig. 4.

Update: Calculating the fitness value of each particle, and if the actual value is better, than at its local best place, the actual place is set as the new local best place. If there is any particle, that is in a better position, than the actual global best place, the global best place is set as the place of the actual best particle.

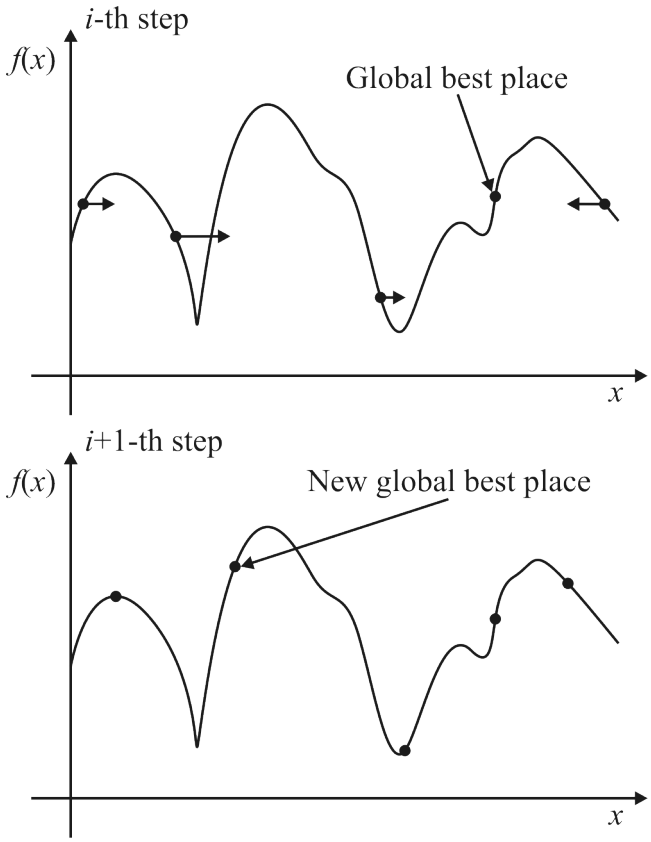

Figure 4. Moving of the particles to the direction of the global best place.

The iteration above runs, until a particle is in a better position, than the required criteria, or until the time limit or generation limit exceeded.

2.4. Advantages and disadvantages

The population based algorithms can be widely applied, when the complexity of the search field prevents of the usage of other analytical or numerical techniques. They can provide for the most task a quasi-optimal solution. A moderate optimal solution can be find relatively

fast. The algorithms can be easily implemented at low cost. The algorithms are iterative, and the model parameters can be changed adaptively between two iteration.

These algorithms have also some disadvantages as well. The global best solution is not guaranteed in a finite time, and the convergence speed is continuously slowing, because of the randomness of the evolution. In some cases, the model parameters can be only modified by trying.

3. Gradient based searching algorithms

Gradient based algorithms, like steepest descent/ascent method [7] and Levenberg-Marquardt method [9, 10] are developed to find the nearest local optimum. These algorithms work with calculating the gradient of the search field at a point in question. The gradient can be calculated analytically from the gradient vector function, otherwise a pseudo-gradient is needed. The pseudo-gradient can be calculated experimentally by producing the derivative of the search field with very small steps. The searching is moving towards based on the gradient, where the optimization needs (to the maximum or minimum).

3.1. Steepest descent/ascent method

The steepest descend (or also mentioned as gradient descend) [7, 11, 12] method is a first order local minimum searching technique. The direction of the search is described by the negative gradient of the search field at the current point. If the interest is in finding the local maximum, the step is proportional to the positive of the gradient. In this case, this method is called as steepest ascent method. The size and the direction of the step is the gradient vector multiplied by a so-called bravery factor. The bigger the γ bravery factor is, the bigger the chance is to missing/jumping over a local minimum. If the search field is denoted as F, and the values of the set of variables in the actual place is denoted as $\boldsymbol{x}_i$, the next place of the searching $\boldsymbol{x}_{i+1}$ from the actual place x_i can be calculated as:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma \nabla F(\boldsymbol{x}_i). \tag{2}$$

This searching procedure on a single random two dimensional F function can be seen on Figure 5.

This optimization technique is very sensitive to the starting position, because it can stuck at every local minimum, instead of finding the real minimum position of the search field.

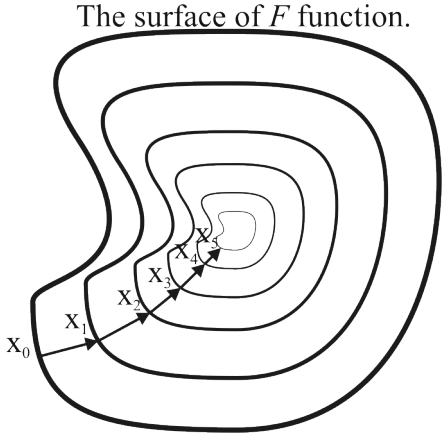

Figure 5. Minimum searching with steepest descent method.

3.2. Levenberg-Marquardt method

The Levenberg-Marquardt method is developed to find the minimum of a multivariate function $f(\boldsymbol{x})$ that is expressed as the sum of squared errors

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{j=1}^{m} \boldsymbol{r}_j^2(\boldsymbol{x}), \tag{3}$$

where $\boldsymbol{r}_j(\boldsymbol{x})$ is a residual function for $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ parameter set. The method uses the *Jacobian-matrix* to minimize the error function. Each row of the Jacobian is the gradient vector of the residual functions corresponding to $\boldsymbol{x}$, and can be written as:

$$\mathbf{J} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}} = \begin{pmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_1}{\partial x_n} & \cdots & \frac{\partial r_m}{\partial x_n} \end{pmatrix}. \tag{4}$$

If the Jacobian can not be calculated calculated analytically, because the formula of the gradient vector function is unknown, a pseudo-Jacobian can be calculated experimentally by determining each pseudo-gradient vector function. The basic of the Levenberg-Marquardt method is the linear approximation of the error function f in the neighborhood of $\boldsymbol{x}$. The core equation of parameter changing between two iteration steps of the method is the following:

$$\Delta(\boldsymbol{x}) = -\left(\mathbf{J}\mathbf{J}^T + \lambda \mathbf{I}\right)^{-1} \mathbf{J}^T \boldsymbol{r}. \tag{5}$$

In (5) λ is a damping parameter, which can be adaptively changed. Using large values for λ parameter results in the gradient descent method, and small values for λ parameter results in the *Gauss-Newton* method.

The Levenberg-Marquardt method is the most widely used optimization technique. In the most case, this method has the best convergence properties, but if the analytical formula of the gradient vector functions are unknown, the calculation of the Jacobian can be very time consuming.

3.3. Advantages and disadvantages

The rapid convergence is the main advantage of the gradient based methods. This methods find the nearest local minimum/maximum effectively with adaptively variable model parameters.

These methods are not sufficient finding global optima, the effectiveness of them are strongly depends on where the searching algorithm starts. Because of the small steps, they can easily stuck at a single local maximum/minimum value, and can only explore the neighborhood of the starting position.

4. Memetic algorithms

The population based algorithms are very effective finding the area of global optima. They provide quasi-optimal solutions within acceptable time, but the speed of the convergence becomes very slow because of the randomness of the evolution, finding the exact global optima is hopeless within finite time. The gradient based methods causing minor modifications of the input parameters, so they can provide fast and accurate result of the place of local optimum, but as it was mentioned before, these algorithms are very sensitive of the starting point, and can easily stuck at a local optima.

Memetic algorithms are hybrid algorithms developed to avoid the disadvantages above by applying them combined. This can be done easily, e.g. applying local searching techniques when the fitness value of the individuals reached a critical value, or using local searches between each generation of the population based algorithms. There are many ways to combine these algorithms, e.g. the combination of genetic algorithm and the steepest descent method is referred as *genetic steepest descent*, the combination of the bacterial evolutionary algorithm and the Levenberg-Marquardt method results in *bacterial memetic algorithm*, etc. The family of memetic algorithms are often referred as *Baldwinian* or *Lamarckian* evolutionary algorithms.

5. Conclusion

In this review paper, I summarized the most widely used population and gradient based searching algorithms. These methods will form the theoretical basics of my future Ph.D. dissertation. The main goal of my work is developing evolutionary and hybrid algorithms to design antennas for strictly specified radiation pattern.

References

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

[2] Nawa, N. E., Hashiyama, T., Furuhashi, T., Uchikawa, Y.: *Fuzzy Logic Controllers Generated by Pseudo-Bacterial Genetic Algorithm*, In Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 2408–2413, Houston, 1997
DOI: 10.1109/ICNN.1997.614446

[3] Nawa, N. E., Furuhashi T.: *Fuzzy system parameters discovery by bacterial evolutionary algorithm*, IEEE Transactions on Fuzzy Systems, vol. 7, no. 5, pp. 608–616, 1999
DOI: 10.1109/91.797983

[4] Kennedy, J., Eberhart R.: *Particle swarm optimization* In Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948, Perth, 1995
DOI: 10.1109/ICNN.1995.488968

[5] Balázs, K.: *Advanced Approaches in the Application Methodologies of Evolutionary Algorithms*. Ph.D. dissertation, Budapest, Hungary, 2013

[6] Sumathi, S., Hamsapriya, T., Surekha P.: *Evolutionary Intelligence - An Introduction to Theory and Applications with Matlab*, Springer, India, 2008

[7] Gál, L.: *Optimalization of Fuzzy Models by Bacterial Type Algorithms (in Hungarian)*, Ph.D. dissertation, Győr, Hungary, 2012

[8] del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J.-C., Harley, R. G.: *Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems*, IEEE Transactions on Evolutionary Computation, vol. 12, no. 2, pp. 171-195, 2008
DOI: 10.1109/TEVC.2007.896686

[9] Levenberg, K.: *A method for the solution of certain non-linear problems in least squares*, Quarterly Journal of Applied Mathematics, vol. 2, no. 2, pp. 164–168, 1944

[10] Marquardt, D.: *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*, SIAM J. Appl. Math., vol. 11, no. 2, pp. 431-441, 1963

[11] Yamada, I.: *The Hybrid Steepest Descent Method for the Variational Inequality Problem over the Intersection of Fixed Points Sets of Nonexpansive Mapping*, Studies in Computational Mathemathics, vol. 8, pp. 473–504, 2001
DOI: 10.1016/S1570-579X(01)80028-8

[12] Snyman, J.: *Practical Mathematical Optimization*, Springer, New York, 2005

View publication stats