Acta
Technica
Jaurinensis

Vol. 8, No. 3, pp. 257–266, 2015
DOI: 10.14513/actatechjaur.v8.n3.381
Available online at acta.sze.hu

A New Metaheuristic Optimization Algorithm, the Weighted Attraction Method

G. Friedl[1], M. Kuczmann[2]

Széchenyi István University, Department of Automation
Egyetem 1, 9026, Győr, Hungary
E-mail: {friedl.gergely[1], kuczmann[2]}@sze.hu

Abstract: The paper presents a novel, particle behavior–based metaheuristic global optimization method. The idea behind the algorithm is based on attraction between particles, and in some aspects it is similar to the particle swarm optimization, but the interaction between particles is realized in a completely different way. The paper shows the main steps of the technique and some possible modifications. After that the comparison of efficiency and the speed of convergence with different well-known algorithms on two objective functions will be shown.

Keywords: *Weighted Attraction Method, metaheuristics, global optimization, swarm intelligence*

1. Introduction

Nowadays the metaheuristic algorithms are one of the most important tools for global optimization. These algorithms do not set almost any requirements to the objective functions, which have to be minimized. These methods can be applied to find the approximation of the optima for not continuous and not differentiable objective functions, even if they cannot be expressed in closed-form. The members of this algorithm family are usually inspired by nature, e.g. the genetic algorithm [1, 2] (GA), the bacterial evolutionary algorithm [3, 4] (BEA), and the particle swarm optimization [5–7] (PSO). These techniques deal with many disadvantages as well, e.g. the convergence speed of GA and BEA can be very slow or extremely decreasing, on the other hand, the PSO can *stuck* at local minimum. It is not exact which algorithm is better than the others, the efficiency is hardly depending on the objective function [8]. In some cases the calculation of the fitness function is very time consuming, the number of function evaluations has to be minimized to access fast runtime.

During our research the main goal was to develop an algorithm that is able to reach fast convergence with minimal number of function evaluations without being stuck. This qualities are very important for an optimization technique, because in a lot of cases the

fitness value can be calculated numerically, and the optimization process can take a huge amount of time. The main idea behind this newly proposed technique is the gravitational attraction between particles. In some aspects it is similar to the PSO, but the determination process of the next searching points are different. The following section introduces the main steps and the modification possibilities. After that, the third section shows the comparison of this method with some another techniques on two test objective function, the Ackley function and the Rosenbrock function.

2. The Weighted Attraction Method

2.1. Basic Concepts of Optimization

The general mathematical form of an optimization process [9] of a *constrained* optimization problem is formulated in the following way:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(\boldsymbol{x}), && \boldsymbol{x} = [x_1, x_2, \ldots, x_n]^\mathsf{T} \in \mathbb{R}^n, \\ \text{subject to} \quad & g_l(\boldsymbol{x}) \leq 0, && l = 1, 2, \ldots, r, \\ & h_l(\boldsymbol{x}) = 0, && l = 1, 2, \ldots, s. \end{aligned} \tag{1}$$

Here $f(\boldsymbol{x})$ is the *objective function*, which has to be minimized, the vector $\boldsymbol{x}$ contains the *design variables*, $g_j(\boldsymbol{x})$ is for the *inequality constraint functions*, and $h_j(\boldsymbol{x})$ denotes the *equality constraint functions*. The functions f, g, and h are scalar functions, more precisely: $\{f, g, h\} : \mathbb{R}^n \to \mathbb{R}$. If the feasible space for the minimum is the whole search space, the problem is called *unconstrained*. The solution vector that exactly satisfies (1) will be denoted as $\boldsymbol{x}^*$, where $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$.

The metaheuristic optimization begins with a set of points on the search space with parameters selected randomly. These points are usually called as *individuals* or *particles*. Let us denote the number of the particles by k. The set of all individuals are called *population* or *swarm*, denoted by $\mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^{k \times n}$. Based on the concept of the applied method, these individuals are moved to another place, or (if the applied method is the GA) replaced by the so–called *offspring*. The goodness of each individual is defined by the *fitness function* as $\boldsymbol{F} = \mathscr{F}\{\mathbf{X}\}$, where $\mathscr{F}$ is the fitness operator, and $\boldsymbol{F} \in \mathbb{R}^k$ is a vector containing the fitness values for all particles. The definition of the fitness function is not straightforward, for a single problem it is possible to define many fitness function from a lot of aspects, but for the simplest cases the fitness function can be equal to the objective function. The interaction between the individuals is based on their fitness value. The better individuals are taken into account better by these methods. These methods often deal with random searching operation, and if the applied method is some of the evolutionary type methods, this is usually called as *mutation*, otherwise it is just called as random search.

2.2. The Proposed Method

The main goal was to develop a new algorithm that is capable of global optimization using less number of individuals without decreasing the runtime. The main idea behind the proposed method is the gravitational attraction between particles. The power of the attraction is a function of the fitness values. This power generates a global particle movement, and this global movement can be used as an efficient global searching technique. The main steps of this method are the following:

1. **Initialization:** Creating an initial, randomly placed particle dispersion on the whole search space;
2. **Calculation of attraction:** Calculating the fitness function, and based on the fitness values of each particle, assigning attraction factor to each one of them;
3. **Moving:** Choosing search direction for each particle based on their movement in the past, and the actual resultant attraction direction;
4. **Explosion:** Scattering the particles if they are too close to each other, and their fitness value is almost equal.

The 1^{st} step is evaluated only once at the beginning of the optimization procedure. The steps 2–4 are running iteratively until the best particle meets the predefined requirements, or the iteration number reaches its maximum. These steps are described in detail in the followings.

2.2.1. Initialization

The 1^{st} step of the weighted attraction method is creating uniformly placed initial searching particles. The easiest case, when the design variables are simply bounded, that is, $a_i \leq x_i \leq b_i$, $i = 1, \ldots, n$, $a_i, b_i \in \mathbb{R}$, and $\boldsymbol{x} \in \mathbf{X}$. Let us denote the j^{th} particle as $\boldsymbol{x}_j$. Now, the matrix $\mathbf{X}$ that represents the swarm of randomly chosen particles can be built up as

$$\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_j, \ldots, \boldsymbol{x}_k], \tag{2}$$

where

$$x_{j,\,i} = a_i + (b_i - a_i)r, \tag{3}$$

and $r \in [0, 1]$ is a random number. The initialization procedure is more difficult if the constraining of the search space is more specific, e.g. the feasible space is an n dimensional sphere or an abstract body. The search space can be limited with the definition of the constraint functions mentioned in (1).

2.2.2. Calculation of the attraction function

After the fitness values are calculated, an attraction factor should be given to each particles. This factor is defined by the *attraction function*. For the determination procedure this algorithm recommends the following steps:

- Sorting the particles according to their fitness value;
- Mapping the swarm to the domain $[0, 1]$ ($\boldsymbol{F} \rightarrow \boldsymbol{F'}$). The extreme values of this interval are assigned to the best and the worst particles, and the other particles got proportionally determined values;
- An attraction function w has to be determined. If this function is linear and its slope is unit, the attraction factors $w(\boldsymbol{F'})$ are equal to the assigned values. Otherwise, the attraction factors of the particles are the related values of the attraction function wF', as can be seen in Fig. 1;

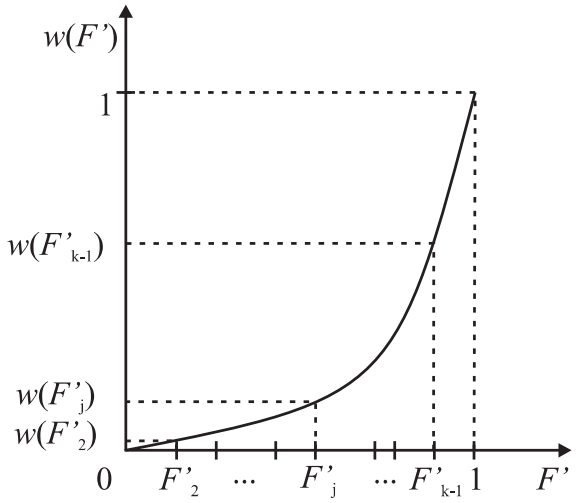


Figure 1. Explanation for the calculation of the attraction factors.

Because with the attraction function the factors of the attraction can be weighted, the algorithm will search more likely the better areas in the search space. The attraction function can be defined different ways. One of the ways can be seen in 1, where $w(0) = 0$ and $w(1) = 1$, and the function is monotonous. The other way is, when $w(0) < 0$. In this case the worst particles will repel the other particles from the bad parts of the objective function.

2.2.3. Moving

Based on the actual position of the particles and the corresponding attraction factors, the position of the *center of the mass* $\boldsymbol{c}$ in the m^{th} iteration can be defined as

$$\boldsymbol{c}^{(m)} = \frac{\sum_{j=1}^{k} \boldsymbol{x}_j^{(m)} w_j^{(m)}}{\sum_{j=1}^{k} w_j^{(m)}}. \tag{4}$$

The direction vectors pointing from the j^{th} particle to the center of mass can be easily calculated as

$$\boldsymbol{d}_j^{(m+1)} = \boldsymbol{c}^{(m)} - \boldsymbol{x}_j^{(m)}. \tag{5}$$

The new position of the j^{th} particle in the $(m+1)^{th}$ iteration is determined with the formula

$$\boldsymbol{x}_j^{(m+1)} = \boldsymbol{x}_j^{(m)} + \varphi_a \boldsymbol{d}_j^{(m+1)} + \varphi_b \boldsymbol{d}_j^{(m)}, \tag{6}$$

where φ_a and φ_b are random numbers. The initial searching directions $\boldsymbol{d}_j^{(0)}$ are optional, but mostly equal to zero. It can be useful if the algorithm takes into account the former searching directions in a better way. It can be easily realized (along with the update of the value $\boldsymbol{d}_j^{(m)}$) as

$$\boldsymbol{d}_j^{(m)} = \frac{u\boldsymbol{d}_j^{(m)} + v\boldsymbol{d}_j^{(m+1)}}{u+v}, \tag{7}$$

where u,$v \in \mathbb{R}^+$, and usually $v \geq u$.

In the early iterations this step usually provides fast convergence for the algorithm, but after awhile, if the particles are stuck in a local (or in fortunate cases global) minimum place, or the particles are in a *flat* area of the objective function, the convergence slows down, or even stops.

2.2.4. Explosion

The algorithm presented above can lead the global searching to stuck, because if every particle fall into the same local valley, the algorithm cannot jump out from here. It can be prevented by dispersing the particles, if a predefined *closeness* in position and fitness value has been reached. The authors recommends the dispersion process to be a normal distributed random dispersion that depends on the actual position and the extreme values of the design variables. A new value of the i^{th} design variable of the j^{th} particle can be calculated using the *Box–Muller–Marsaglia polar method* [10]. In algorithm 1, the value of α is predefined by the user. The greater value of α expands the expected radius of the dispersion.

The closeness condition is user–defined, but an optimal formula for it is unknown yet. It is clear that it depends on the variance of the fitness values and on the variance of the positions of the particles.

Algorithm 1 Box–Muller–Marsaglia polar method

procedure BMM(μ,σ) ▷ Normal random number generation.
  $\mu \leftarrow x_{j,\,i}$ ▷ Mean of the normal distribution is the actual value.
  $\sigma \leftarrow \alpha(\max(\mathbf{X}_{all,\,i}) - \min(\mathbf{X}_{all,\,i}))$ ▷ Formula for the variance.
  repeat
    $p, q \leftarrow [-1, 1]$ ▷ Uniformly distributed random number from the interval.
    $z \leftarrow p^2 + q^2$
  until $0 < z < 1$ ▷ Not to take the square root of a negative number.
  $x_{j,\,i_1} \leftarrow \mu + p\sigma\sqrt{-2\frac{\ln z}{z}}$
  $x_{j,\,i_2} \leftarrow \mu + q\sigma\sqrt{-2\frac{\ln z}{z}}$
  $r \leftarrow a \bmod b$
  return $x_{j,\,i_1}$ or $x_{j,\,i_2}$ ▷ Two random number generated, only one needed.
end procedure

3. Testing and comparison

This section is showing the comparison of the proposed algorithm with GA, BEA and PSO using the Ackley function and Rosenbrock function. The Ackley and Rosenbrock function are two well known test functions to evaluate properties of optimization methods.

The 2D Ackley function can be formulated as [12]:

$$f(x_1,x_2) = -20 \cdot \mathrm{e}^{-0.2\sqrt{0.5(x_1^2+x_2^2)}} - \mathrm{e}^{0.5(\cos(2\pi x_1)+\cos(2\pi x_2))} + 20 + \mathrm{e}. \tag{8}$$

This function is symmetric, has a numerous local minima, and a global minimum at the $x_1 = x_2 = 0$ point, where the function value is $f(0, 0) = 0$.
The most common form of the 2D Rosenbrock function can be formulated as [13]:

$$f(x_1,x_2) = (x_1 - 1)^2 + 100(x_2 - x_1^2)^2. \tag{9}$$

This function is asymmetric, has only one local (which is also a global) minimum, and the global minimum is at the $x_1 = x_2 = 1$ point, where the function value is $f(1, 1) = 0$.

On the objective functions, two different tests were made:

1. Optimization with 10 searching particles/individuals on the $-10 \leq x_1, x_2 \leq 10$ domain;
2. Optimization with 100 searching particles/individuals on the $-100 \leq x_1, x_2 \leq 100$ domain.

To test the robustness of the algorithm, every optimization process were made 100 times

and lasted 1000 period. In these cases the fitness functions were equal to the objective functions.

In Fig. 2 and 3, the comparison of the algorithms searching the minimum of the Ackley function is presented, in addition to the settings mentioned above.

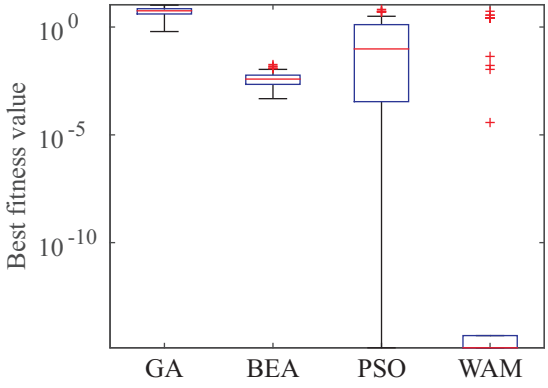


Figure 2. Optimization on the Ackley function, first case.

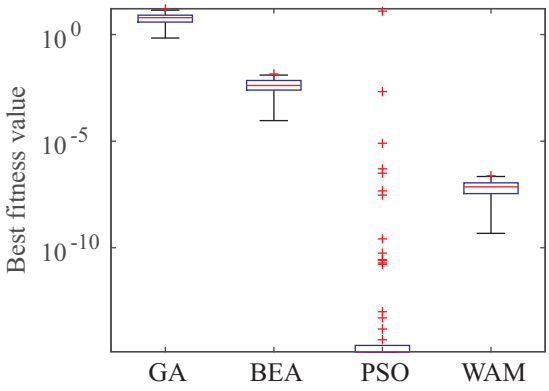


Figure 3. Optimization on the Ackley function, second case.

The presented figures were created by the `boxplot` function of `Matlab`. On each box the central mark is the median, the edges of the box are the 25^{th} and 75^{th} percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually [11]. In the first case, the WAM outperforms the other algorithms, in the worst processes it provided results at least as good as the other algorithms, bit in

the second case, with a lot of searching particles, the algorithm could not get any closer to the global minimum. The optimization processes were usually started with very fast convergence, but in the area of the minimum, the particles were jumping over the $[0, 0]$ point.

The comparisons of the algorithms on the Rosenbrock function in the two cases mentioned above can be seen in Fig. 4 and 5.

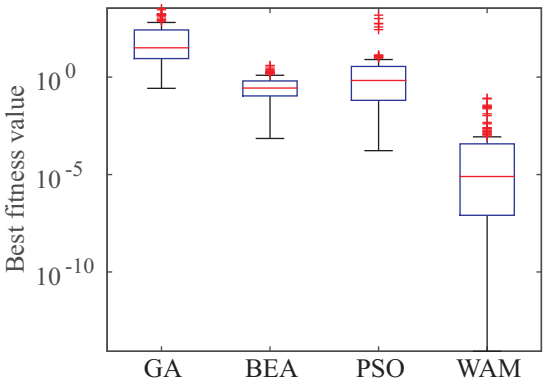


Figure 4. Optimization on the Rosenbrock function, first case.

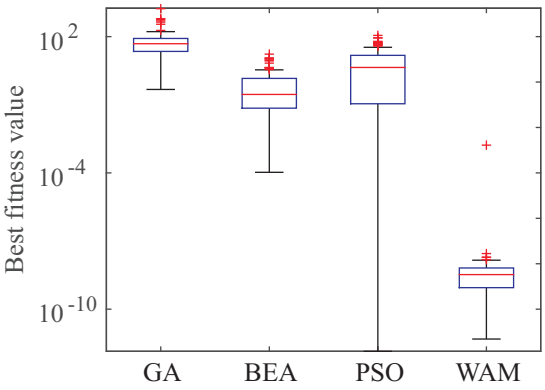
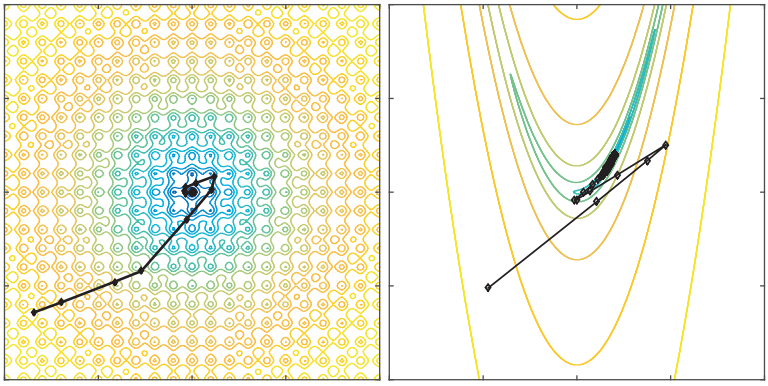


Figure 5. Optimization on the Rosenbrock function, second case.

Now in both cases, the WAM provided better solutions than the other applied algorithms, and in the second case, with more particle on greater domain the WAM could perform better approximation of the global minimum, and this contradicts the previous experiences.

The following figures show the typical trajectory of a particle during the optimization process. The empirical experiments showed that if the objective function is symmetrical like the Ackley function, the particles follow a spiral orbit during the searching process, but in the non-symmetrical case this behaviour was not experimented (see in Fig. 6).



(a) Ackley function *(b) Rosenbrock function*

Figure 6. Typical trajectory of a searching particle.

4. Conclusions

The paper proposed a metaheuristic optimization tool named as Weighted Attraction Method. The main idea behind the algorithm was the gravitational attraction between particles. The concept is supplemented with a step that is responsible for the scattering of the particles preventing the algorithm being stuck. The comparison of the algorithm with other well known techniques were presented in Section 3. The Weighted Attraction Method were able to outperform the other techniques in the most cases, especially when the number of the searching particles were low. This property makes the algorithm even more effective, because it deals with fast convergence while the number of the fitness function evaluation remains low. The main disadvantage of the algorithm is that an exact optimal formula for the closeness of the particles does not exist yet. Now it seems that the best opportunity for this could be an adaptive *closeness condition* with adaptive dispersion scale.

The proposed algorithm will used to optimize antenna geometries, when the objective function is a strictly specified radiation pattern. The algorithm is believed to accelerate this optimization task, because it appears to be suitable for problems, where the evaluation of the fitness function is the most time consuming step.

References

[1] Holland JH.: Adaption in Natural and Artificial Systems, University of Michigan Press, Massachusetts, 1975

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

[3] Nawa NE, 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

[4] Nawa NE, 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

[5] 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

[6] del Valle Y, Venayagamoorthy GK, Mohagheghi S, Hernandez J-C, Harley RG: 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

[7] Neri F, Mininno E, Iacca G: Compact Particle Swarm Optimization, Information Sciences, vol. 239, pp. 96–121, 2013
DOI: 10.1016/j.ins.2013.03.026

[8] Wolpert DH, Macready WG:No Free Lunch Theorems for Optimization, IIEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67–82, 1997
DOI: 10.1109/4235.585893

[9] Snyman JA: Practical Mathematical Optimization, Springer, New York, 2005

[10] Luke S: Essentials of Metaheuristics, Lulu, second edition, 2013
available at http://cs.gmu.edu/∼sean/book/metaheuristics/

[11] Matlab, www.mathworks.com, (last visited: 06 May 2015)

[12] Ackley DH: A connectionist machine for genetic hillclimbing, Kluwer Academic Publishers, Boston, 1987

[13] Rosenbrock HH: An automatic method for finding the greatest or least value of a function, The Computer Journal, vol. 3, pp. 175–184, 1960
DOI: 10.1093/comjnl/3.3.175