A Population-Based Optimization Method Using Newton Fractal

Soyeong Jeong and Pilwon Kim

School of Technology Management and Department of Mathematical Sciences, Ulsan National Institute of Science and Technology (UNIST), Ulsan Metropolitan City 689-798, Republic of Korea

Correspondence should be addressed to Pilwon Kim; pwkim@unist.ac.kr

Received 29 May 2018; Revised 24 August 2018; Accepted 17 September 2018; Published 3 February 2019

Guest Editor: Wenbo Wang

Copyright © 2019 Soyeong Jeong and Pilwon Kim. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We propose a deterministic population-based method for a global optimization, a Newton particle optimizer (NPO). The algorithm uses the Newton method with a guiding function and drives particles toward the current best positions. The particles’ movements are influenced by the fractal nature of the Newton method and are greatly diversified in the approach to the temporal best optimums. As a result, NPO generates a wide variety of searching paths, achieving a balance between exploration and exploitation. NPO differs from other metaheuristic methods in that it combines an exact mathematical operation with heuristics and is therefore open to more rigorous analysis. The local and global search of the method can be separately handled as properties of an associated multidimensional mapping.

1. Introduction

The most critical component of a random search algorithm is balancing the trade-off between exploration and exploitation. Exploration is to ensure that the algorithm examines a rather wide region far from the current optimum in the search space to avoid getting trapped in a local minimum. Exploitation is the ability of the algorithm to search the surrounding area nearby the current optimum, attempting to improve it. Since these two objectives are to some extent in conflict, finding an algorithm that could compromise both is challenging [1, 2].

The population-based methods are characterized by a set of multiple potential solutions to the problem, moving towards a near-optimal solution area. These algorithms increase the capability of finding the global minimum since they make a simultaneous search in many directions by using a population of temporal solutions. Particle swarm optimization (PSO) is a popular method of choice in the field of heuristic search, which tunes up between the best known position and each particle’s local best optimum [3–6].

In this work, we introduce a new population-based metaheuristic method, the Newton particle optimizer (NPO), which achieves a natural balance between exploration and exploitation in global optimization problems. The algorithm drives the particles (searching agents) in the search space using the Newton method and therefore is deterministic in contrast to most population-based methods. In those conventional methods, each particle combines the currently available information with a random walk to decide the next movement. Hence, the final results may vary even with an identical initial particle distribution. On the other hand, NPO uses the Newton method and determines the searching path exactly from the current positions of the particles.

While the searching paths are completely reproducible from the initial distribution, they are very irregular, reflecting the fractal nature of the Newton method. It is this complexity of the paths that enables us to develop an efficient optimization method. The Newton method with a proper guiding function makes the particles search close around the current optimums and intermittently pushes them away so that the particles can explore distant potential optimums. However, every particle eventually converges to one of the optimums sooner or later. If a better optimum is found during this process, it will be reflected in the guiding function accordingly.

The metaheuristic approach in optimization has been often criticized for lack of scientific rigor [7]. There are many “nature-inspired” methods for global optimizations, which have been introduced without a detailed study of...
the mechanics underlying the methods. Since NPO combines a well-established deterministic operation with heuristics, it allows a more rigorous analysis.

2. Properties of Newton Paths

The Newton method is a fundamental numerical scheme for finding successively better approximations to the roots of a nonlinear function [8, 9]. For a real/complex-valued function \( f \), the corresponding scheme to find its zeros is

\[
x \leftarrow x - m \frac{f(x)}{f'(x)}. \tag{1}
\]

If \( x \) starts close enough to one of the zeros of \( f \), it converges to the zero rapidly. While the sequence generated from (1) eventually converges to the zero, its detailed behaviour can be very sensitive to the initial point and the values of the degree coefficient \( m \). For example, if \( f : \mathbb{C} \to \mathbb{C} \) is a complex function, then the corresponding Newton iteration exhibits a fractal structure in the complex plane [10]. This implies that the trace of \( x \), which we call the Newton path, can be very irregular with a wide variety.

Consider a polynomial function

\[
f(x) = (x - p_1)(x - p_2) \cdots (x - p_n), \tag{2}
\]

where \( p_1, p_2, \ldots, p_n \) are \( n \) locations of interest in \( \mathbb{R} \). Applying the Newton method in (1) to (2) iteratively yields the dynamics of the particle \( x \) wandering around \( p_1, p_2, \ldots, p_n \). Figure 1 illustrates such dynamics around a root.

The behavioural tendency of the sequence sensitively depends on the choice of \( m \). Refer to Figure 2. While the sequence of a particle with \( m = 1 \) rapidly converges to one of the roots, \( x = 2 \), in Figure 2(a), the sequences with \( m = 2.5 \) and \( m = 4 \) in Figures 2(b) and 2(c), respectively, seem to keep wandering around the roots, \( x = 0, 1, \) or \( 2 \), along the iterations. The particle with a higher value of \( m \) makes a more irregular movement around the target, even taking a jump as high as 60. However, it should be noted that the sequences are bounded and never get away from \( x = 2 \).

The dynamics of a particle in the 2-dimensional search space is even more diversified under the guiding function, \( f = f(z) \), \( z \in \mathbb{C} \). Sensitive dependence of the Newton method on the initial point divides the domain into complicated regions, called the Julia sets, according to where the resulting sequence converges. Figure 3 compares three Newton paths with distinct values of \( m \) in the complex plane. Three particles are initiated at the same point \((5, 5)\) and are led to the origin under the same guiding function \( f(z) = z(z - 2i)(z + 1 - i) \). While the particle with \( m = 1 \) makes a gradual search toward the origin, the ones with \( m = 2.5 \) and \( m = 3.5 \) show rather wild swings around it. It is notable that the movements of the latter two particles are an irregular mix of jumping and mincing. Figure 4 shows that the Newton paths sensitively change with the initial point near \((3, 3)\), while they eventually converge to the same point. This

\[\text{Figure 1: Illustration of the particle dynamics under a guiding function in the 1-dimensional search space: a particle is searching around a current optimum (boxed point) to find a better one, moving from 1 to 5.}\]

\[\text{Figure 2: Sequences generated with a different degree coefficient } m \text{ for the guiding function } f(x) = x(x - 1)(x - 2).\]
“diversely convergent” searching paths promote a balanced search between exploration and exploitation.

It is the complexity of the Newton paths that enables us to develop an efficient optimization method. Suppose $g$ is the fitness function to be optimized. One constructs a guiding function $f$ such that the roots of $f$ coincide with the current best optimums of $g$. Then all particles are driven by (1) along the Newton paths to the zeros of $f$, which are also the candidate optimum of $g$. The particles are likely to move close around the current optimums, or sometimes take a sudden jump away from them. If a better optimum is found during this process, the guiding function $f$ is updated accordingly.

3. Method

3.1. Algorithm. This section introduces the Newton particle optimizer based on the Newton paths dealt in Section 2.

We begin with 2-dimensional examples to illustrate the algorithm of NPO clearly. Suppose $g : \mathbb{C} \to \mathbb{R}$ is the fitness function, or the cost function to be optimized. We adopt $N$ particles $z_i, 1 \leq i \leq N$, as “searching agents,” assigning them with the degree coefficient $m_i \in \mathbb{C}$. At each iteration of the scheme, $n$ best fitters with respect to the fitness function $g$ are selected as leading particles. That is, we pick $p_1, \ldots, p_n$, such that $p_i = z_{k_i}$, $g_{z_{k_i}}$ is the $i$th minimum among $g_{z_1}, \ldots, g_{z_N}$ for $i = 1, \ldots, n$.

A guiding function $f(z)$ is constructed as a polynomial of degree $n$ whose roots coincide with $p_1, \ldots, p_n$, as

$$f(z) = (z - p_1)(z - p_2) \cdots (z - p_n).$$

Then, we update the particle’s position $z_1, \ldots, z_n$ by applying the scheme

$$z_i \leftarrow z_i - m_i \frac{f(z_i)}{f'(z_i)}.$$  \hspace{1cm} (5)

Note that all the particles except the leading particles are attracted toward leading particles. Once the position of all particles are updated, we examine their fitness to choose the next leading particles $p_1, p_2, \ldots, p_n$. Then, we refresh the guiding function $f$ accordingly, and reapply (5) to the particles. As long as the guiding function is a polynomial function, the convergence of the algorithm is guaranteed from the convergence of Newton’s method and the monotone convergence theorem. However, like other heuristic optimization methods, convergence here means convergence of the sequence of solutions in which all particles have converged to points of leading particles. In the search space, those points may or may not be the optimum.

The pseudocode of the procedure is as described in Algorithm 1.

In $\mathbb{R}^d$, $d \geq 3$, we set a guiding function $f(x) = (f(x), \ldots, f_d(x))^T$ where $x \in \mathbb{R}^d$ and $f_i(x)$ is a real valued
Figure 5 depicts that the optimal values of \( z_i \) may vary. In practice, we assign di

\[ \text{for each particle do} \]
\[ \text{Initialize particle } z_i \text{ with } m_i. \]
\[ \text{end for} \]
\[ \text{while maximum iterations or minimum error criteria is not attained do} \]
\[ \text{for each particle do} \]
\[ \text{Calculate fitness value } g. \]
\[ \text{end for} \]
\[ \text{Choose } n \text{ best members } p_1, p_2, \ldots, p_n \text{ in the search domain.} \]
\[ \text{Set the guiding function } f(z) = (z - p_1)(z - p_2) \cdots (z - p_n). \]
\[ \text{for each particle do} \]
\[ z_i \leftarrow z_i - m \frac{f(z_i)}{f'(z_i)} \]
\[ \text{end for} \]
\[ \text{end while} \]

Algorithm 1: Newton Particle Optimization (in C).

![Algorithm diagram](image)

Figure 5: Performance variation according to \( m_{\text{max}} \). 200 particles with 4 leading particles are applied to find the minimum of the Rosenbrock function.

function in \( \mathbb{R}^d \). Note that the updated scheme in (5) becomes the following multidimensional Newton scheme:

\[ x \leftarrow x - MDf(x)^{-1}f(x), \]  

(6)

where \( M \) is a \( d \)-by-\( d \) constant matrix and \( Df^{-1} \) is an inverse of the Jacobian matrix of \( f \) at \( x \). The eigenvalues of \( M \) are generally associated with a local search tendency, \( m \).

3.2. Criteria for Choice of \( m \) and \( M \). In Section 3.1, it is illustrated that the parameter \( m \) in (5) determines the diversity of the searching paths of NPO. We can show that particles driven by NPO remain bounded if \( m \) satisfies \( 0 < m < 2n \), where \( n \) is the number of leading particles, or equivalently, the degree of a polynomial guiding function \( f \). For a small \( m \), the movement of particles slows down. On the other hand, it becomes very irregular if \( m \) is close to \( 2n \). We can also find a more practical criterion for \( m \) from experiments. In practice, we assign different values of \( m_i \) to each particle to diversify the search. We usually pick them uniformly from the interval \([0, m_{\text{max}}]\). Figure 5 depicts that the optimal values for \( m_{\text{max}} \) tend to be around the middle of the range, that is, \( m_{\text{max}} = n \). When using a lesser number of particles, one may choose \( m_{\text{max}} > n \), so that the scheme generates more irregular searching paths and therefore compensates for a lack of diversity. In a higher dimensional NPO in (6), the eigenvalues of \( M \) play a similar role of \( m \) that decides the local search tendency. Practically, one can simply use a diagonal matrix of which diagonal entries follow the distribution of \( m_i \) as mentioned above.

From the above observation, we can see that adjusting \( m_{\text{max}} \) is one way to control the trade-off between exploitation and exploration in NPO. Let us define an exploration indicator (EI) as

\[ \text{EI}_1 = \frac{m_{\text{max}}}{2n}. \]  

(7)

Note that \( 0 < \text{EI}_1 < 1 \). If \( \text{EI}_1 \) is small, the particles tend to search near the current optimums. If \( \text{EI}_1 \) is close to 1, they widely explore away from the current optimums along irregular Newton paths. We usually set \( \text{EI}_1 = 0.5 \).

3.3. Construction of Guiding Functions. The essence of NPO lies in that the diversely exploring paths toward the temporal optimums can be easily generated from the simple deterministic iteration working on a guiding function. Generating a proper guiding function is therefore the main issue to balance between the exploration and the exploitation. The three desirable properties that a guiding function is expected to have are as follows:

(i) The function has zeros at the designated points and no zeros elsewhere

(ii) The function is symmetric

(iii) The inverse of its Jacobian is easy to compute

Condition (i) is necessary for the guiding function to drive particles to a target. Condition (ii) is for unbiased search. Polynomial functions are likely to satisfy condition (iii) and reduce the computational cost. One can easily confirm that a factored polynomial such as (4) in 1- and 2-dimensional search spaces naturally satisfies (i), (ii), and (iii). Unfortunately, it is well known that such factored polynomials do not exist in more than three-dimensional space.
We therefore need to sacrifice condition (i), at least partly, to construct a polynomial guiding function in $\mathbb{R}^d$, $d \geq 3$.

In order to find a proper extension of (4) in a higher dimensional domain, we rewrite (4) in the complex form as $f(x + iy) = u(x, y) + iv(x, y)$ and $p_l = q_l + ir_l$. Then the corresponding component functions are

$$
\begin{align*}
    u(x, y) &= \prod_{k=1}^{n} (x - q_k) - \sum_{l=1}^{n} (x - q_l) \prod_{m=1, m \neq l}^{n} (y - r_m), \\
    v(x, y) &= -\prod_{k=1}^{n} (y - r_k) + \sum_{l=1}^{n} (y - r_l) \prod_{m=1, m \neq l}^{n} (x - q_m).
\end{align*}
$$

(8)

Now we propose, as an extension of (8), a general guiding function in $\mathbb{R}^d$ as

$$
\begin{align*}
    f_j(x) &= (-1)^{i+1} \left\{ \prod_{k=1}^{n} (x_{j+2} - p_{j+2,k}) \\
    &- \sum_{l=1}^{n} (x_l - p_{j,l}) \prod_{m=1, m \neq l}^{n} (x_{j+1} - p_{j+1,m}) \right\},
\end{align*}
$$

(9)

where $p_l = (p_{l,1}, \ldots, p_{l,d})^T$, $1 \leq i \leq n$ denotes a leading particle. Here, we use circular indexes like $x_{j+1} = x_j$ in (9). One can confirm that $f(x)$ vanishes if $x = p_l$, $1 \leq i \leq n$. Note that $f(x)$ may have extra zero vectors at other than $p_1, \ldots, p_n$, in $\mathbb{R}^d$, $d \geq 3$. However, it is interesting that such extra zeros do not lower the searching performance. Indeed, such “phantom roots” occur irregularly and lead the particles to more extensive search. This finding naturally brings us to the issue of how to choose the leading particles for more efficient exploration.

3.4. Criteria for Choice of Leading Particles. The choice of leading particles $p_1, \ldots, p_n$ is crucial in searching performance in that their relative positions determine other particles’ movement. One immediate idea is to adopt the current best optimums as suggested in (3). That is, we simply choose top $n$ rank fitters from the population. However, this may cause in many cases overly fast convergence to current optimums, giving up further exploration.

To avoid being trapped at local minimums too soon and to diversify global searching paths, we select leading particles not only from the best fitters, but also from mediocre ones. These less efficient performers prevent a situation where a few similar local minimums happen to dominate all the other particles. Figure 6 compares two cases of NPO, one of which uses the top 5 rank fitters and the other uses the top 4 rank fitters and 1 mediocre fitter with a low rank for leading particles. Interestingly, while the case with this “idle leader” seems to be slow at first, it eventually becomes more successful in finding the minimum value. This can be explained by a balance between exploitation and exploration: when the best fitters lead other particles to the temporal best, they may cause overexploitation and make the particles neglect further exploration. However, the existence of the idle leader may distract the particles’ attention from the temporal best and have them search for better optimums. That is, the idle leader enhances the exploration of the search.

From the above observation, we can develop another exploration indicator that is associated with the balance between exploitation and exploration. Let us define

$$
\text{EI}_2 = \frac{1}{2} - \frac{n(n+1)}{4\sigma} + \frac{n(n+1)}{4(nN + n - \sigma)},
$$

(10)

where $\sigma$ is the summation of the ranks of the leading particles. One can see that $0 < \text{EI}_2 < 1$ and it becomes close to 0 when all leading particles are chosen from the top best fitters. Hence, $\text{EI} = 0$ means low exploration (high exploitation). On the contrary, if all leading particles come from lowest-rank particles, the indicator increases to 1, implying high exploration (low exploitation). One may adjust $\text{EI}_2$ in the middle range for a balanced search. In the examples in Figure 6(a) $\text{EI}_2 = 0.015$ and in Figure 6(b) $\text{EI}_2 = 0.366$.

4. Numerical Results

This section compares the numerical performances of NPO with those of PSO and the firefly algorithm. The firefly algorithm (FA) is another stochastic population-based method which has been applied in most areas of optimization [11, 12]. We tested with 28 benchmark functions suggested from the CEC 2013 competition for real-parameter optimization [13]. The functions were 10-dimensional and 51 optimization tests were done for each problem. We kept the same 100,000 times of evaluation of functions for all methods.

NPO uses 4 leading particles, 3 from the top best and one from the 50% performer. The matrix $M$ for each particle was chosen such that its eigenvalues lie between 0 and 4. The parameters for PSO were set to the recommended values which are widely used in the benchmark tests [14–16]. The parameters for FA were taken from [17, 18].
Table 1: Benchmark for NPO, PSO, and FA: tested with 28 10-dimensional functions from CEC 2013 competition.

| Function | Best       | Worst      | Median     | Mean       | Variance    |
|----------|------------|------------|------------|------------|-------------|
| NPO      | 1.000e−08  | 1.000e−08  | 1.000e−08  | 1.000e−08  | 1.000e−08   |
| PSO      | 1.000e−08  | 1.000e−08  | 1.000e−08  | 1.000e−08  | 1.000e−08   |
| FA       | 2.413e−06  | 1.526e−05  | 8.771e−06  | 9.311e−06  | 8.862e−12   |
| NPO      | 1.935e+04  | 1.056e+06  | 2.269e+05  | 3.291e+05  | 2.670e+05   |
| PSO      | 3.686e+04  | 2.911e+06  | 2.767e+05  | 4.520e+05  | 5.253e+05   |
| FA       | 2.720e+03  | 2.255e+05  | 3.900e+05  | 5.487e+04  | 2.480e+09   |
| NPO      | 9.035e−01  | 1.234e+08  | 9.889e+05  | 5.881e+06  | 1.765e+07   |
| PSO      | 4.078e−03  | 4.513e+07  | 1.077e+05  | 1.631e+06  | 6.624e+06   |
| FA       | 2.429e+00  | 1.877e+03  | 6.572e+01  | 2.622e+02  | 1.854e+05   |
| NPO      | 4.565e+01  | 1.671e+03  | 2.231e+02  | 3.531e+02  | 3.360e+02   |
| PSO      | 3.972e+02  | 3.814e+03  | 1.223e+03  | 1.340e+03  | 6.479e+02   |
| FA       | 1.162e+02  | 5.479e+03  | 1.311e+03  | 1.655e+03  | 1.514e+06   |
| NPO      | 2.130e−05  | 3.202e−04  | 1.141e−04  | 1.232e−04  | 6.380e−05   |
| PSO      | 1.000e−08  | 1.000e−08  | 1.000e−08  | 1.000e−08  | 1.000e−08   |
| FA       | 7.248e−04  | 2.908e−03  | 1.652e−03  | 1.709e−03  | 2.628e−07   |
| NPO      | 4.910e−06  | 7.939e+01  | 9.853e+00  | 7.926e+00  | 1.107e+01   |
| PSO      | 4.753e−02  | 7.699e+01  | 1.013e+01  | 8.818e+00  | 1.053e+01   |
| FA       | 3.314e−03  | 9.820e+00  | 9.814e+00  | 9.440e+00  | 3.505e+00   |
| NPO      | 8.153e−03  | 1.650e+01  | 1.586e+00  | 3.488e+00  | 4.058e+00   |
| PSO      | 4.003e−01  | 8.370e+01  | 2.790e+00  | 7.079e+00  | 1.384e+01   |
| FA       | 7.785e−03  | 1.504e−01  | 2.231e−02  | 3.281e−02  | 8.395e−04   |
| NPO      | 2.008e+01  | 2.047e+01  | 2.032e+01  | 2.030e+01  | 9.520e−02   |
| PSO      | 2.009e+01  | 2.047e+01  | 2.033e+01  | 2.031e+01  | 8.881e−02   |
| FA       | 2.014e+01  | 2.050e+01  | 2.038e+01  | 2.036e+01  | 6.315e−03   |
| NPO      | 1.324e+00  | 6.142e+00  | 3.495e+00  | 3.493e+00  | 1.098e+00   |
| PSO      | 7.067e−01  | 5.864e+00  | 3.123e+00  | 3.045e+00  | 1.214e+00   |
| FA       | 4.928e−02  | 2.492e+00  | 1.644e+00  | 1.500e+00  | 3.757e−01   |
| NPO      | 9.353e−02  | 1.202e+00  | 4.733e−01  | 5.136e−01  | 2.461e−01   |
| PSO      | 3.693e−02  | 1.321e+00  | 3.151e−01  | 3.828e−01  | 2.482e−01   |
| FA       | 1.751e−02  | 2.833e−01  | 8.426e−02  | 9.841e−02  | 3.595e−03   |
| NPO      | 8.060e−09  | 2.989e+00  | 4.997e−01  | 6.962e−01  | 7.574e−01   |
| PSO      | 1.000e−08  | 4.975e+00  | 1.990e+00  | 1.974e+00  | 1.272e+00   |
| FA       | 9.950e−01  | 1.492e+01  | 4.975e+00  | 5.736e+00  | 8.497e+00   |
| NPO      | 2.984e+00  | 2.984e+01  | 1.293e+01  | 1.356e+01  | 5.561e+00   |
| PSO      | 2.985e+00  | 3.927e+01  | 1.293e+01  | 1.435e+01  | 7.071e+00   |
| FA       | 9.950e−01  | 1.691e+01  | 5.970e+00  | 6.438e+00  | 1.150e+01   |
| NPO      | 7.426e+00  | 3.828e+01  | 2.293e+01  | 2.271e+01  | 8.047e+00   |
| PSO      | 4.003e+00  | 3.909e+01  | 2.258e+01  | 2.215e+01  | 7.815e+00   |
| FA       | 1.990e+00  | 2.745e+01  | 1.046e+01  | 1.129e+01  | 4.502e+01   |
| NPO      | 2.995e−01  | 2.894e+02  | 7.649e+01  | 1.087e+02  | 8.823e+01   |
| PSO      | 3.747e−01  | 5.912e+02  | 1.018e+02  | 1.257e+02  | 1.114e+02   |
| FA       | 1.189e+01  | 9.208e+02  | 4.004e+02  | 4.424e+02  | 3.466e+04   |
### Table 1: Continued.

| Function | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| NPO      | 2.676e + 01 | 1.442e + 03 | 8.231e + 02 | 8.216e + 02 | 3.104e + 02 |
| PSO      | 2.846e + 02 | 1.555e + 03 | 7.610e + 02 | 7.990e + 02 | 2.823e + 02 |
| FA       | 1.303e + 02 | 8.364e + 02 | 2.537e + 02 | 2.984e + 02 | 2.719e + 04 |
| NPO      | 5.424e - 02 | 1.021e + 00 | 4.327e - 01 | 4.365e - 01 | 2.474e - 01 |
| PSO      | 5.023e - 01 | 1.634e + 00 | 9.331e - 01 | 9.553e - 01 | 2.372e - 01 |
| FA       | 1.304e - 02 | 3.961e - 01 | 8.392e - 02 | 1.140e - 01 | 7.635e - 03 |
| NPO      | 1.097e + 01 | 1.985e + 01 | 1.405e + 01 | 1.464e + 01 | 2.113e + 00 |
| PSO      | 2.308e + 00 | 2.157e + 01 | 1.440e + 01 | 1.376e + 01 | 4.639e + 00 |
| FA       | 1.242e + 01 | 3.003e + 01 | 1.757e + 01 | 1.808e + 01 | 1.674e + 01 |
| NPO      | 1.400e + 01 | 4.754e + 01 | 2.411e + 01 | 2.558e + 01 | 7.811e + 00 |
| PSO      | 7.196e + 00 | 5.724e + 01 | 3.009e + 01 | 3.051e + 01 | 9.823e + 00 |
| FA       | 1.207e + 01 | 2.984e + 01 | 1.687e + 01 | 1.770e + 01 | 1.698e + 01 |
| NPO      | 2.148e - 01 | 1.299e + 00 | 5.489e - 01 | 5.837e - 01 | 2.068e - 01 |
| PSO      | 3.495e - 01 | 1.106e + 00 | 6.702e - 01 | 6.785e - 01 | 1.950e - 01 |
| FA       | 4.368e - 01 | 1.703e + 00 | 8.771e - 01 | 9.387e - 01 | 8.351e - 02 |
| NPO      | 9.732e - 01 | 3.876e + 00 | 2.744e + 00 | 2.733e + 00 | 6.217e - 01 |
| PSO      | 9.494e - 01 | 3.621e + 00 | 3.202e + 00 | 2.979e + 00 | 6.406e - 01 |
| FA       | 1.826e + 00 | 5.000e + 00 | 3.269e + 00 | 3.261e + 00 | 4.416e - 01 |
| NPO      | 3.000e + 02 | 4.001e + 02 | 4.001e + 02 | 3.982e + 02 | 1.389e + 01 |
| PSO      | 2.000e + 02 | 4.002e + 02 | 4.002e + 02 | 3.649e + 02 | 7.370e + 01 |
| FA       | 4.002e + 02 | 4.002e + 02 | 4.002e + 02 | 4.002e + 02 | 2.201e - 14 |
| NPO      | 1.250e + 01 | 4.068e + 02 | 1.682e + 02 | 1.741e + 02 | 9.757e + 01 |
| PSO      | 2.879e + 01 | 4.089e + 02 | 2.098e + 02 | 2.039e + 02 | 1.028e + 02 |
| FA       | 2.532e + 01 | 1.424e + 03 | 5.736e + 02 | 5.582e + 02 | 8.571e + 04 |
| NPO      | 3.999e + 02 | 1.837e + 03 | 1.090e + 03 | 1.097e + 03 | 3.703e + 02 |
| PSO      | 2.271e + 02 | 1.575e + 03 | 8.299e + 02 | 8.140e + 02 | 2.908e + 02 |
| FA       | 3.874e + 01 | 1.993e + 03 | 4.077e + 02 | 4.849e + 02 | 1.019e + 05 |
| NPO      | 1.106e + 02 | 2.175e + 02 | 2.079e + 02 | 2.022e + 02 | 2.309e + 01 |
| PSO      | 2.001e + 02 | 2.211e + 02 | 2.098e + 02 | 2.099e + 02 | 5.524e + 00 |
| FA       | 2.001e + 02 | 2.166e + 02 | 2.002e + 02 | 2.022e + 02 | 2.163e + 01 |
| NPO      | 1.149e + 02 | 2.202e + 02 | 2.059e + 02 | 2.059e + 02 | 1.409e + 01 |
| PSO      | 1.205e + 02 | 2.214e + 02 | 2.101e + 02 | 2.087e + 02 | 1.349e + 01 |
| FA       | 2.001e + 02 | 2.111e + 02 | 2.002e + 02 | 2.009e + 02 | 5.615e + 00 |
| NPO      | 1.039e + 02 | 2.000e + 02 | 2.000e + 02 | 1.736e + 02 | 3.938e + 01 |
| PSO      | 1.060e + 02 | 3.176e + 02 | 2.000e + 02 | 1.864e + 02 | 7.060e + 01 |
| FA       | 1.020e + 02 | 3.144e + 02 | 2.002e + 02 | 1.969e + 02 | 8.862e + 03 |
| NPO      | 3.002e + 02 | 5.414e + 02 | 3.059e + 02 | 3.409e + 02 | 6.418e + 01 |
| PSO      | 3.003e + 02 | 6.393e + 02 | 4.935e + 02 | 4.373e + 02 | 1.224e + 02 |
| FA       | 3.010e + 02 | 4.000e + 02 | 4.000e + 02 | 3.559e + 02 | 2.416e + 03 |
| NPO      | 1.000e + 02 | 6.426e + 02 | 3.000e + 02 | 3.423e + 02 | 1.239e + 02 |
| PSO      | 1.000e + 02 | 3.000e + 02 | 3.000e + 02 | 2.882e + 02 | 4.706e + 01 |
| FA       | 3.001e + 02 | 4.001e + 02 | 3.001e + 02 | 3.217e + 02 | 1.727e + 03 |
Table 1 shows that the performances of the three schemes are comparable. However, the summarized ranking of NPO, PSO, and FA in Table 2 are, respectively, 1.821, 2.107, and 2.01, which indicates that NPO is practically better than PSO and FA with these benchmark functions.

5. Discussion

NPO takes a deterministic approach based on a mathematical operation, the Newton method. Due to the inherent fractal properties and strong convergence of the method, NPO seems to enjoy both features of exploration and exploitation, making effective optimizations for a wide range of functions. One of major differences between NPO and other random-walk-based metaheuristic methods is that NPO is a combination of a well-established mathematical operation and heuristics: its local and global search abilities can be controlled as a property of a multidimensional mapping. Since such mapping can be easily created like a polynomial function that we suggested in this paper, the convergent/divergent tendency of searching paths can also be analysed and finely tuned accordingly. This opens the door to the future study of customizing the guiding functions so that they can reflect the properties of a given set of problem instances.

Data Availability

The numerical results are all included in the article.

Disclosure

A part of this work on a 2-dimensional case has been presented in the 2017 International Conference on Intelligent Systems, Metaheuristics, and Swarm Intelligence. The funder had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2017R1D1A1B04032921).

References

[1] Q. He, L. Wang, and B. Liu, “Parameter estimation for chaotic systems by particle swarm optimization,” Chaos, Solitons & Fractals, vol. 34, no. 2, pp. 654–661, 2007.
[2] Z. Sheng, J. Wang, S. Zhou, and B. Zhou, “Parameter estimation for chaotic systems using a hybrid adaptive cuckoo search with simulated annealing algorithm,” Chaos: An Interdisciplinary Journal of Nonlinear Science, vol. 24, no. 1, article 013133, 2014.
[3] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of ICNN’95—International Conference on Neural Networks, pp. 1942–1948, Perth, WA, Australia, 1995.
[4] Q. Liao, S. Zhou, H. Shi, and W. Shi, “Parameter estimation of nonlinear systems by dynamic cuckoo search,” Neural Computation, vol. 29, no. 4, pp. 1103–1123, 2017.
[5] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization,” Swarm Intelligence, vol. 1, no. 1, pp. 33–57, 2007.
[6] Y. Shi and R. C. Eberhart, “Parameter selection in particle swarm optimization,” in International Conference on Evolutionary Programming of Lecture Notes in Computer Science, pp. 591–600, San Diego, CA, USA, 1998Springer.
[7] K. Sörensen, “Metaheuristics—the metaphor exposed,” International Transactions in Operational Research, vol. 22, no. 1, pp. 3–18, 2015.
[8] J. Hubbard, D. Schleicher, and S. Sutherland, “How to find all roots of complex polynomials by Newton’s method,” Inventiones Mathematicae, vol. 146, no. 1, pp. 1–33, 2001.
[9] T. Sauer, Numerical Analysis: Pearson New International Edition, Pearson Higher Ed, 2013.
[10] B. B. Mandelbrot, The Fractal Geometry of Nature/Revised and Enlarged Edition, WH Freeman and Co., New York, NY, USA, 1983.
[11] I. Fister, I. Fister Jr., X. S. Yang, and J. Brest, “A comprehensive review of firefly algorithms,” *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.

[12] X.-S. Yang, “Firefly algorithms for multimodal optimization,” in *International Symposium on Stochastic Algorithms of Lecture Notes in Computer Science*, pp. 169–178, Sapporo, Japan, 2009, Springer.

[13] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Diaz, “Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization,” Tech. Rep. 201212, Computational Intelligence Laboratory, Zhengzhou University, and Nanyang Technological University, 2013.

[14] C.-W. Chou, J.-H. Lin, and R. Jeng, “Markov chain and adaptive parameter selection on particle swarm optimizer,” *International Journal on Soft Computing*, vol. 4, no. 2, pp. 1–15, 2013.

[15] F. A. Guerra and L. S. Coelho, “Multi-step ahead nonlinear identification of Lorenz’s chaotic system using radial basis neural network with learning by clustering and particle swarm optimization,” *Chaos, Solitons & Fractals*, vol. 35, no. 5, pp. 967–979, 2008.

[16] M. Khodier, N. Dib, and J. Ababneh, “Design of multi-band multi-section transmission line transformer using particle swarm optimization,” *Electrical Engineering*, vol. 90, no. 4, pp. 293–300, 2008.

[17] Y.-B. Mo, Y.-Z. Ma, and Q.-Y. Zheng, “Optimal choice of parameters for firefly algorithm,” in *2013 Fourth International Conference on Digital Manufacturing & Automation*, pp. 887–892, Qingdao, China, 2013.

[18] M. Takeuchi, H. Matsushita, Y. Uwate, and Y. Nishio, “Firefly algorithm existing leader fireflies,” in *2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 313–316, Jeju, Republic of Korea, 2016.
