A Simplex Search-Based Particle Swarm Optimization Algorithm Postprint

Authors: Hu Jinfan, Zhang Xiaowei, Yuan Qijiang, Zhang Weijun, Cheng Chongdong

Date: 2018-11-29T00:00:00+00:00

Abstract

To improve the solution performance of the particle swarm optimization algorithm, a hybrid algorithm based on simplex search and particle swarm optimization is proposed. On the one hand, the algorithm adaptively determines the inertia weight, cognitive, and social parameters to achieve a parameter-free purpose; on the other hand, it utilizes simplex search to guide the search direction of some particles, thereby accelerating algorithm convergence. Numerical experimental results show that, compared with the traditional particle swarm algorithm and other simplex-based particle swarm algorithms, the algorithm performs well in terms of the number of evaluations and solution accuracy.

Full Text

Preamble

Vol. 37 No. 1
Application Research of Computers
Accepted Paper

A Particle Swarm Optimization Algorithm Based on Simplex Search

Hu Jinfan, Zhang Xiaowei†, Yuan Qijiang, Zhang Weijun, Cheng Chongdong
(School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract: To improve the performance of particle swarm optimization, this paper proposes a hybrid algorithm based on simplex search and particle swarm optimization. On the one hand, the algorithm adaptively determines the inertia weight, cognitive, and social parameters to achieve a parameter-free design. On the other hand, it utilizes simplex search to guide the search direction of

some particles, thereby accelerating convergence. Numerical experimental results demonstrate that compared with traditional particle swarm algorithms and other simplex-based particle swarm algorithms, the proposed algorithm performs well in terms of function evaluations and solution accuracy.

Keywords: direct search; simplex search; particle swarm optimization

0 Introduction

Particle swarm optimization (PSO), proposed by Kennedy and Eberhart in 1995, simulates the migration and flocking behavior of bird swarms foraging for food to find global optima. In 1998, Shi et al. introduced an inertia weight mechanism that further improved PSO performance. Compared with traditional optimization methods, PSO offers several advantages: it does not require derivative information of the objective function, providing excellent generality; it is less prone to local optima and demonstrates strong robustness; and its simple process is easy to implement. However, the heuristic nature of PSO often leads to premature convergence, and its performance heavily depends on parameter settings for specific problems.

Many improved algorithms have failed to adequately address the issues of excessive parameters and reliance on empirical settings, while simple parameter-free improved algorithms tend to suffer from premature convergence. Simplex search can leverage weak analytical properties of problems, has rich theoretical foundations, conducts targeted searches with fast convergence, and shares natural commonalities with PSO, making them complementary. Considering these characteristics, numerous simplex-based PSO algorithms have emerged.

In recent years, many studies have investigated simplex search-based PSO algorithms, which have achieved good results and wide application in engineering. In 2007, Fan and Zahira proposed a hybrid algorithm for unconstrained optimization that combined simplex search with PSO to improve convergence rate. While this algorithm structure provided ideas for subsequent designs, it had many parameters and its performance on higher-dimensional test functions needed improvement. In 2008, Hsu and Gao embedded a simplex search mechanism into PSO that executed one simplex search every k iterations, reducing function evaluations and balancing exploration-exploitation, but their experiments evaluated only 2-5 dimensional functions. In 2010, Ren et al. proposed a quantum PSO based on the simplex method that used quantum PSO for global search first, then determined whether to apply simplex for local search based on population fitness standard deviation. However, judging premature convergence from fitness standard deviation is difficult, limiting further promotion. In 2018, Wu et al. added simplex search to PSO initialization for preprocessing initial values, testing on 20 10-dimensional functions. Results showed higher accuracy than standard PSO, but the algorithm suffered from difficult parameter settings and many evaluations.

To improve PSO performance, this paper proposes a simplex search-based PSO algorithm. On one hand, it adaptively determines inertia weight, cognitive, and social parameters using principles similar to electromagnetism-like mechanisms to achieve parameter-free operation. On the other hand, it uses simplex search to guide search directions and step sizes, accelerating convergence.

1 Particle Swarm Algorithm

The particle swarm optimization concept originates from bird flocking behavior during flight or foraging. Birds only track a limited number of neighbors, yet the overall result appears centrally controlled—complex global behavior emerges from simple rule interactions. Unlike Darwinian "survival of the fittest"evolution, PSO finds optimal solutions through individual cooperation.

The PSO algorithm proceeds as follows:

a) Initialize population $P(t)$, cognitive coefficient c_1, social coefficient c_2, inertia weight w, and set $t = 0$.

b) Calculate each particle $X_i(t) \in P(t)$' s fitness value, individual best position $p_i(t)$, and global best position $G(t)$.

c) Update each particle $X_i(t) \in P(t)$' s velocity and position using equations (1) and (2).

d) If termination conditions are met, stop; otherwise, set $t = t+1$ and return to step b).

In step c), equation (1) shows particle velocity depends on previous velocity, its own historical best position, and the swarm' s global best position. Equation (2) shows how each particle' s position updates in the search space.

2 Simplex Search

A simplex is a fundamental concept in algebraic topology, generalizing triangles and tetrahedrons. An n-dimensional simplex is the convex hull of $n + 1$ points. For example, a 1D simplex is a line segment, a 2D simplex is a triangle, and a 3D simplex is a tetrahedron.

The Nelder-Mead simplex search (NM) seeks optimal solutions through elementary geometric transformations including reflection, expansion, contraction, and shrink. Each transformation attempts to replace the worst vertex with a better one. Taking the minimization problem $\min_{X\in\mathbb{R}^n} f(X)$ as an example, we calculate function values at three vertices X_{low}, X_{sec}, X_{high} as f_{low}, f_{sec}, f_{high}, assuming $f_{high} > f_{sec} > f_{low}$.

The basic operations (Figures 1 [Figure 1: see original paper] to 4) are:

a) **Reflection**: Calculate midpoint $X_{cent} = 0.5\times(X_{low}+X_{sec})$ and reflection point $X_{ref} = X_{cent} + (X_{cent} - X_{high})$. If $f_{low} > f_{ref}$, calculate expansion

point $X_{exp} = 2 \times X_{cent} - X_{ref}$.

b) **Expansion**: If $f_{exp} < f_{ref}$, accept X_{exp}; otherwise accept X_{ref}.

c) **Contraction**: If $f_{ref} \geq f_{sec}$:

- (a) If $f_{ref} < f_{high}$, calculate outside contraction point $X_{cont} = 0.5 \times (X_{cent} + X_{ref})$.
- (b) If $f_{ref} \geq f_{high}$, calculate inside contraction point $X_{cont} = 0.5 \times (X_{cent} + X_{high})$.

d) **Shrink**: If $f_{high} > f_{cont}$, replace all vertices except X_{low} with $X_i = 0.5 \times (X_i + X_{low})$.

The NM algorithm description is:

a) Calculate function values at $n+1$ vertices X_i $(i = 1, \ldots, n+1)$, identify worst X_{high}, second-worst X_{sec}, best X_{low}, and midpoint X_{cent}.

b) Calculate reflection point X_{ref} and its function value.

c) Perform simplex transformation:

- If $f_{ref} < f_{low}$, calculate expansion point X_{exp}. If $f_{exp} < f_{ref}$, replace X_{high} with X_{exp}; otherwise replace with X_{ref}.
- Else if $f_{ref} < f_{sec}$, replace X_{high} with X_{ref}.
- Else if $f_{ref} < f_{high}$, calculate outside contraction point X_{cont}. If $f_{cont} < f_{ref}$, replace X_{high} with X_{cont}; otherwise perform shrink.
- Else calculate inside contraction point X_{cont}. If $f_{cont} < f_{high}$, replace X_{high} with X_{cont}; otherwise perform shrink.

3.1 PSO Algorithm Improvement

In traditional PSO, the i-th particle' s position update depends on three factors (equations (1) and (2)). However, in the Nelder-Mead simplex particle swarm optimization (NM-PSO), for particle i in generation t population $P(t)$, we update using equations (3)-(5):

The candidate particle $T_i(t)$ is calculated as:

$$T_i(t) = X_i(t) + V_i(t) \quad (3)$$

where the velocity $V_i(t)$ is:

$$V_i(t) = F_0 + F_1 + F_2 \quad (4)$$

with:

$$F_0 = \left[\frac{f(X_{r_3}(t)) - f(X_{r_1}(t))}{D} \right] \cdot (X_{r_3}(t) - X_{r_1}(t))$$

$$F_1 = \left[\frac{f(g(t)) - f(X_{r_2}(t))}{D}\right] \cdot (g(t) - X_{r_2}(t))$$

$$F_2 = \left[\frac{f(G(t)) - f(X_{r_3}(t))}{D}\right] \cdot (G(t) - X_{r_3}(t))$$

Here r_1, r_2, r_3 are three distinct particles randomly selected from the population, $D = f_{high} - f_{low}$ records the difference between worst and best fitness values, $g(t)$ is the individual best, and $G(t)$ is the global best.

Equation (5) uses a greedy selection mechanism to update particle $X_i(t)$: it replaces $X_i(t)$ with candidate $T_i(t)$ only when $f(T_i(t)) < f(X_i(t))$. This approach uses information sharing between the base point $X_{r_3}(t)$ and reference points to construct candidate particles.

When $f(X_{r_1}(t)) > f(X_{r_3}(t))$, F_0 guides $X_{r_3}(t)$ to search outward. The larger the fitness difference, the greater the step size in that direction. Conversely, when $f(X_{r_1}(t)) < f(X_{r_3}(t))$, F_0 guides $X_{r_3}(t)$ to search inward, with step size characterizing neighborhood size. Larger fitness differences create smaller neighborhoods, while smaller differences create larger neighborhoods. This mechanism balances local search and global exploration while adaptively switching between accelerating convergence and maintaining population diversity.

Similarly, F_1 records information between the reference point $X_{r_3}(t)$ and individual best $g(t)$. F_2 guides the base point $X_{r_3}(t)$ toward the global best $G(t)$. This two-particle information exchange mechanism resembles electromagnetism, where each particle acts as a point charge—better (lower) function values correspond to higher charge and stronger attraction. A particle' s next search direction is the "resultant force" from other particles, improving performance and achieving parameter-free operation.

3.2 Algorithm Description

a) Initialize population with size POP $= 3n+1$, mutation coefficient $Cr = 0.1$, and set $t = 0$.

b) Sorting: Calculate fitness values $f_i(t)$ and sort particles by fitness ascending.

c) Simplex search: Use the best $n + 1$ particles as simplex vertices, execute NM algorithm to improve the worst vertex.

d) Particle swarm optimization: For the remaining $2n$ particles forming population $P(t)$, each particle $X_i(t) \in P(t)$:

- (a) Calculate candidate particle $T_i(t)$ using equation (4) and update via equation (3).
- (b) Boundary check: For any component j, if $T_{ij}(t) < LB$, set $T_{ij}(t) = LB$; if $T_{ij}(t) > UB$, set $T_{ij}(t) = UB$.

- (c) Greedy selection: If $f(T_i(t)) < f(X_i(t))$, set $X_i(t+1) = T_i(t)$; otherwise $X_i(t+1) = X_i(t)$.
- (d) Mutation: For any component j, if rand $> Cr$, set $X_{ij}(t+1) = X_{ij}(t)$.

e) If termination conditions are met, stop; otherwise set $t = t+1$ and return to step b).

To increase population diversity and avoid premature convergence, NM algorithm executes only one iteration in step c, and mutation is added in step d.

4.1 Control Algorithm Introduction

This section compares numerical experimental results of five algorithms including the proposed NM-PSO. The four control algorithms are:

1) **Degressive Inertia Weight PSO (DPSO)**: In standard PSO, inertia weight w dominates velocity updates. Larger w values produce longer "step sizes" beneficial for global search, while smaller w enhances local search precision. A decreasing w with iterations strengthens global search early and local search later, improving overall performance. The decreasing formula is:
$$w = w_{max} - (w_{max} - w_{min}) \times \frac{iter}{maxiter}$$
where w_{max} and w_{min} are set to 1.4 and 0 respectively as recommended by Shi et al. Other steps remain consistent with standard PSO.

2) **Random Inertia Weight PSO (RPSO)**: This modifies PSO by using a uniformly distributed random inertia weight $w = \text{rand}(0.5, 1)$. This simple adaptive mechanism serves as a baseline to demonstrate performance improvements from more sophisticated adaptive mechanisms in NM-PSO.

3) **Hybrid Simplex PSO (HSPSO)**: In each iteration, HSPSO updates the best $N+1$ particles using simplex search and improves the remaining $2N$ particles using PSO. Simplex search executes only one iteration before replacing the worst vertex, reducing computational complexity while maintaining performance.

4) **Simplex Method PSO (SMPSO)**: After random initialization, SMPSO preprocesses initial values using simplex search before PSO iterations. If processed initial values are near the global optimum, PSO converges quickly. Even if trapped in local optima, PSO can escape with high probability. This leverages both algorithms' strengths effectively.

4.2 Numerical Experiment Data

Five algorithms were independently tested 50 times on 10 30-dimensional test functions (see appendix). Results are shown in Tables 1-4 and Figures 6-15. Algorithm parameters are:

1) **DPSO**: POP $= 3 \times 30 + 1$, $c_1 = c_2 = 1.494$, $w_{max} = 1.4$, $w_{min} = 0$
2) **RPSO**: POP $= 3 \times 30 + 1$, $c_1 = c_2 = 1.494$, $w = \text{rand}(0.5, 1)$
3) **HSPSO**: POP $= 3 \times 30 + 1$, $c_1 = 0.6$, $c_2 = 1.6$, $w = \text{rand}(0.5, 1)$, mutation coefficient $\lambda = 0.85$, reflection coefficient $r = 1.75$, expansion coefficient $e = 2.75$, contraction coefficient $c = 0.75$
4) **SMPSO**: POP $= 3 \times 30 + 1$, $c_1 = 1.5$, $c_2 = 2.5$, $w = \text{rand}(0.5, 1)$

Tables 1-3 record best, worst, and mean solution errors versus theoretical optima across 50 runs. Table 4 records solution variance. Figures 6-15 show mean values versus evaluation counts (logarithmic y-axis for clarity).

4.3.1 Table Data Analysis

Table 4 shows NM-PSO achieves significantly lower variance than other algorithms, except slightly higher than DPSO on Sphere, indicating superior stability. Table 3 demonstrates NM-PSO leads in mean error for most functions, often by multiple orders of magnitude, except on Rosenbrock where performance is comparable. Tables 1-2 reveal NM-PSO produces the best solutions on Rastrigin, Styblinski-Tang, Exponential, and Schwefel 2.22. For worst-case solutions across 8 functions, NM-PSO leads, particularly by 1+ orders on Rastrigin, Styblinski-Tang, Levy, Exponential, Zakharov, and Schwefel 2.22 (9 orders on Levy), demonstrating strong local optima escape capability. Overall, NM-PSO performs best on most test functions, never being completely disadvantaged even when not optimal.

4.3.2 Chart Data Analysis

For Rastrigin, Styblinski-Tang, and Levy, NM-PSO shows overwhelming advantage in average solution precision. After approximately 20,000-40,000 function evaluations, the other four algorithms become trapped in local optima with minimal subsequent improvement, while NM-PSO achieves final values 2-4 orders of magnitude better. For Ackley, Griewank, Exponential, Zakharov, and Schwefel 2.22, NM-PSO demonstrates superior convergence speed (faster y-axis descent). Only HSPSO matches NM-PSO briefly on Griewank early on; NM-PSO consistently achieves best results. For Sphere, three simplex-based PSO algorithms outperform RPSO but trail DPSO, possibly due to hybrid structure characteristics. On Rosenbrock, all five algorithms fail to reach global optimum (errors around 10^2). Comparing simplex-based algorithms on Ackley, all three outperform traditional PSO in final results and convergence speed, with NM-PSO being optimal. For Rastrigin, Griewank, Levy, Exponential, and Zakharov, at least two simplex-based algorithms outperform traditional methods, confirming simplex search benefits, with NM-PSO being the best among them.

In conclusion, NM-PSO demonstrates high solution precision and superior performance on most test functions.

5 Conclusion

To improve PSO performance, this paper designed a PSO iteration method inspired by electromagnetism-like mechanisms and used simplex search to accelerate convergence. Numerical experiments show the parameter-free NM-PSO effectively balances local search and global exploration. The original intention was to leverage mature theories from traditional optimization to guide evolutionary algorithm design. Future work will focus on designing better and more concise algorithm fusion methods.

References

[1] Kennedy J. Particle swarm optimization [M]// Encyclopedia of machine learning. Boston: Springer, 2011: 760-766.

[2] Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space [J]. IEEE Trans on Evolutionary Computation, 2002, 6 (1): 58-73.

[3] Kennedy J. Swarm intelligence [M]// Handbook of nature-inspired and innovative computing. Boston: Springer, 2006: 187-219.

[4] Shi Yuhui, Eberhart R. A modified particle swarm optimizer [C]// Proc of IEEE International Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence. Piscataway, NJ: IEEE Press, 1998: 69-73.

[5] Eberhart R C, Shi Yuhui. Tracking and optimizing dynamic systems with particle swarms [C]// Proc of Congress on Evolutionary Computation. 2001: 94-100.

[6] Shi Yuhui, Eberhart R C. Parameter selection in particle swarm optimization [C]// Proc of International Conference on Evolutionary Programming. Berlin: Springer, 1998: 591-600.

[7] Eberhart R C, Shi Yuhui. Comparison between genetic algorithms and particle swarm optimization [C]// Proc of International conference on evolutionary programming. Berlin: Springer, 1998: 611-616.

[8] Eberhart R C, Shi Yuhui. Comparing inertia weights and constriction factors in particle swarm optimization [C]// Proc of Congress on Evolutionary Computation. 2000: 84-88.

[9] Yang Chunhua, Qian Xiaoshan, Gui Weihua. A hybrid algorithm for chaotic difference evolution and particle swarm optimization [J]. Application Research of Computers, 2011, 28 (2).

[10] Li Hongliang, Hou Chaozhen, Zhou Shaosheng. An efficient improved particle swarm optimization algorithm [J]. Computer Engineering and Applications, 2008, 44 (1): 14-16.

[11] Kennedy J. Bare bones particle swarms [C]// Proc of Swarm Intelligence Symposium. 2003: 80-87.

[12] Birbil Ş İ, Fang S C. An electromagnetism-like mechanism for global optimization [J]. Journal of Global Optimization, 2003, 25 (3): 263-282.

[13] Loengarov A, Tereshko V. A minimal model of honey bee foraging [C]// Proc of IEEE Swarm Intell. Symp. 2006: 175-182.

[14] Marini F, Walczak B. Particle swarm optimization (PSO): a tutorial [J]. Chemometrics and Intelligent Laboratory Systems, 2015, 149: 153-165.

[15] Zahara E, Kao Y T. Hybrid Nelder–mead simplex search and particle swarm optimization for constrained engineering design problems [J]. Expert Systems with Applications, 2009, 36 (2): 3880-3886.

[16] Fan S K S, Zahara E. A hybrid simplex search and particle swarm optimization for unconstrained optimization [J]. European Journal of Operational Research, 2007, 181 (2): 527-548.

[17] Hsu C C, Gao C H. Particle swarm optimization incorporating simplex search and center particle for global optimization [C]// Proc of IEEE Conference on Soft Computing in Industrial Applications. 2008: 26-31.

[18] Ren XiaoKang, Hao Ruizhi, Sun Zhengxing, et al. Quantum particle swarm optimization algorithm based on simplex method [J]. Microelectronics and Computer Science, 2010 (1): 154-157.

[19] Wu Kedong, Wei Zengxin, Yang Tianshan. A particle swarm optimization algorithm using simplex method [J]. Journal of Mathematics in Practice and Theory, 2018, 48 (7): 199-205.

[20] Conn A R, Scheinberg K, Vicente L N. Introduction to derivative-free optimization [M]// MOS-SIAM Series on Optimization, 2009: 141-162.

[21] Trelea I C. The particle swarm optimization algorithm: convergence analysis and parameter selection [J]. Information processing letters, 2003, 85 (6): 317-325.

Appendix: Test Functions

Table 5 Information of 10 Test Functions

Function	Dimension	Range	Optimum
Ackley	30	[-32, 32]	0
Sphere	30	[-100, 100]	0
Rosenbrock	30	[-30, 30]	0
Rastrigin	30	[-5.12, 5.12]	0
Griewank	30	[-600, 600]	0

Function	Dimension	Range	Optimum
Styblinski-tang	30	[-5, 5]	-39.166
Exponential	30	[-10, 10]	-1
Zakharov	30	[-1, 1]	0
Schwefel 2.22	30	[-5, 10]	0

Table 6 Test Functions

- **Ackley**: $f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$
- **Sphere**: $f(x) = \sum_{i=1}^{n} x_i^2$
- **Rosenbrock**: $f(x) = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$
- **Rastrigin**: $f(x) = \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
- **Griewank**: $f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
- **Styblinski-tang**: $f(x) = \frac{1}{2}\sum_{i=1}^{n}\left(x_i^4 - 16x_i^2 + 5x_i\right)$
- **Exponential**: $f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$
- **Zakharov**: $f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$
- **Schwefel 2.22**: $f(x) = \sum_{i=1}^{n}|x_i| + \prod_{i=1}^{n}|x_i|$

Note: Figure translations are in progress. See original paper for figures.