Ionospheric Disturbance Optimization

Jincheng Zhang*1[0009-0005-1860-0009]

[1] Faculty of Science and Technology, Rajabhat Maha Sarakham University, Maha Sarakham 44000, Thailand

zjc1639834588@gmail.com

Abstract. This paper proposes a heuristic optimization algorithm based on the mechanism of planetary ionospheric perturbations, namely the Ionospheric Disturbance Optimization (IDO) algorithm. This algorithm simulates the physical process of electron density perturbations in the planetary ionosphere, introduces local perturbations and propagation mechanisms, and achieves an organic combination of global and local search. The algorithm is modeled and analyzed using plain text mathematical formulas, including core mechanisms such as individual initialization, perturbation updates, propagation attenuation, and fitness selection. This paper describes in detail the algorithm design principles, mathematical model, iterative strategy, and pseudocode implementation. The algorithm's heuristic properties and global convergence capability are analyzed, and its potential application value in continuous optimization problems is explored. This algorithm can be theoretically analyzed without relying on complex experimental data and is suitable for solving multidimensional continuous optimization problems.

Keywords: Planetary ionospheric disturbances, heuristic optimization, mathematical modeling, disturbance propagation, global convergence.

I. Introduction

The planetary ionosphere is a charged layer in a planet's atmosphere, formed by high-energy solar radiation and cosmic rays. Its electron density exhibits complex variations with altitude, time, and space. Ionospheric disturbances primarily originate from solar flares, coronal mass ejections (CMEs), atmospheric gravity waves, and the interaction between the planetary magnetosphere and the solar wind. Disturbances manifest as localized increases or decreases in electron density and can propagate through space, affecting the entire ionospheric structure. Their randomness, locality, and propagation provide a natural inspiration for heuristic optimization algorithms: perturbations simulate random exploration in the search space, propagation mechanisms simulate information transfer between individuals, and global convergence corresponds to the search for the optimal solution to the optimization problem.

In recent years, heuristic optimization algorithms have been widely used in continuous optimization, combinatorial optimization, and multi-objective optimization problems. Classical algorithms, such as genetic algorithms, particle swarm optimization algorithms, and ant colony algorithms, achieve global search by simulating natural evolution, swarm behavior, or physical phenomena. However, when solving multimodal complex functions, traditional algorithms are prone to falling into local optima and lack exploration capabilities [1-51]. The IDO algorithm proposed in this paper simulates the physical processes of planetary ionospheric disturbances, combining local perturbations with global guidance to enhance search space coverage.

This paper focuses on:

Designing an optimization strategy using the physical mechanisms of planetary ionospheric disturbances.

Constructing a plain-text mathematical formula to describe the algorithmic process, including perturbation updates, propagation attenuation, and fitness selection.

Analyzing the algorithm's heuristic properties and global convergence capabilities, its theoretical application value is explored.

II. The Physical Mechanism of Planetary Ionospheric Disturbances

The core characteristics of planetary ionospheric disturbances include multi-source perturbations, local electron density variations, and perturbation propagation. Taking the Earth as an example, the ionosphere is divided into the D, E, F1, and F2 layers, and their electron density, n_e, varies significantly with altitude, time, and the solar activity index. The perturbation can be described by the following physical equation:

Electron density continuity equation:

$$\partial n_e/\partial t + \nabla\cdot(n_e * v_e) = P - L$$

where n_e is the electron density, v_e is the electron drift velocity, P is the generation rate, and L is the loss rate.

Ion momentum equation:

$$m_i * (\partial v_i/\partial t + v_i \cdot \nabla v_i) = q_i * (E + v_i \times B) - \nabla p_i - m_i * \nu_{in} * (v_i - v_n)$$

where m_i is the ion mass, q_i is the ion charge, E and B are the electric and magnetic fields, p_i is the ion pressure, ν_{in} is the frequency of ion-neutral particle collisions, and v_n is the neutral gas velocity.

Perturbation Propagation Equation:

$$\partial^2(\delta n_e)/\partial t^2 + \omega_{pe}^2 * \delta n_e - c_s^2 * \nabla^2(\delta n_e) = S(t, x)$$

Where δn_e is the electron density perturbation, ω_{pe} is the plasma frequency, c_s is the speed of sound, and $S(t, x)$ is the perturbation source term. The propagation of perturbations in space follows an exponential decay law, and their local effects diminish with increasing distance.

The above equation provides a theoretical basis for the IDO algorithm, mapping the perturbation, propagation, and global equilibrium mechanisms into the individual update and information transfer processes of the optimization algorithm.

III. Design of Planetary Ionospheric Perturbation Optimization Algorithm
3.1 Algorithm Framework

The IDO algorithm consists of the following steps:

Initialize candidate solutions: Randomly generate N individuals within the search space.

Local Perturbation Update: Simulate local perturbation sources to introduce random perturbations to increase exploration capabilities.

Perturbation Propagation: Simulates the transmission of disturbances between individuals, gradually attenuating the influence of information between neighboring individuals.

Fitness Evaluation and Selection: Calculates the objective function value for each individual and retains the optimal solution.

Global Guided Update: Uses the current optimal solution to guide individuals toward the global optimal region.

Iterative Convergence: Repeats the above steps until the iteration limit or convergence condition is reached.

3.2 Mathematical Modeling

Initialize Candidate Solutions:

$$X_i^0 = X_{min} + rand() * (X_{max} - X_{min}), \quad i = 1..N$$

Where N is the number of individuals, X_{min} and X_{max} are the search space boundaries, and rand() is a uniform random number in the range [0, 1].

Perturbation Update Formula:

$$X_i^{t+1} = X_i^t + \alpha * \mathrm{randn}() * D_i + \beta * (X_{best}^t - X_i^t)$$

Where X_i^t is the position of the i-th individual in the t-th generation, X_{best}^t is the current optimal solution, α is the perturbation intensity coefficient, β is the global bootstrap coefficient, randn() is a standard normal distribution random number, and D_i is the local perturbation amplitude.

Local Perturbation Amplitude D_i:

$$D_i = \gamma * \exp(-\|X_i^t - X_j^t\| / \sigma)$$

Where γ is the perturbation amplitude reference, X_j^t is the position of the neighboring individual, and σ is the perturbation propagation attenuation parameter. This formula simulates the attenuation of perturbations in the ionosphere as they propagate across space.

Fitness selection rule:

if $f(X_i^{t+1}) < f(X_i^t)$:
$X_i^{t+1} = X_i^{t+1}$
else:
$X_i^{t+1} = X_i^t$

f(x) is the objective function (minimization problem).

Termination condition:

$$|f(X_{best}^{t+1}) - f(X_{best}^t)| < \varepsilon$$

where ε is the convergence threshold.

3.3 Pseudocode Implementation
Input: Objective function f(x), search space [X_{min}, X_{max}], number of individuals N, maximum iterations T_{max}
Output: Global optimal solution X_{best}, optimal fitness f_{best}

1: Initialize X_i^0, i=1..N
2: Calculate initial fitness $f(X_i^0)$
3: $X_{best}^0 = \mathrm{argmin}(f(X_i^0))$

```
for t = 1 to T_max do
for i = 1 to N do
Select neighboring individuals X_j^t
Calculate local perturbation D_i = γ * exp(-||X_i^t - X_j^t|| / σ)
Update position X_i^{t+1} = X_i^t + α * randn() * D_i + β * (X_best^t - X_i^t)
Constraint bounds: X_i^{t+1} ∈ [X_min, X_max]
Calculate fitness f(X_i^{t+1})
If f(X_i^{t+1}) < f(X_i^t), retain X_i^{t+1}; otherwise, X_i^{t+1} = X_i^t
end for
Update global optimal value X_best^t
end for

return X_best, f(X_best)
```

IV. Algorithm Characteristics Analysis

Heuristic Exploration Capability

Local perturbations D_i simulate random fluctuations in the local electron density of the ionosphere, guiding individuals out of local optima and enabling multi-region exploration.

Adaptive Propagation Mechanism

The exponential decay in the perturbation propagation formula simulates the strength of information transfer between individuals, enabling the search process to balance local utilization with global exploration.

Global Convergence Ability

The global guidance term $\beta * (X_{best}^t - X_i^t)$ ensures that individuals converge toward the optimal solution, theoretically achieving global convergence.

Scalability and Adaptability

The algorithm framework is flexible and can be extended to multi-objective optimization, constrained optimization, and dynamic environment optimization problems.

V. Mathematical Formula Extension Analysis

To gain a deeper understanding of the algorithm's dynamic behavior, the perturbation update formula can be linearized:

Perturbation Propagation Attenuation Term:

$D_i = \gamma * \exp(-\|X_i^t - X_j^t\| / \sigma) \approx \gamma * (1 - \|X_i^t - X_j^t\| / \sigma)$ (when $\|X_i^t - X_j^t\| \ll \sigma$)

This approximation can be used to analyze the impact of a local perturbation on neighboring individuals.

Iterative Convergence Condition:

$$X_i^{t+1} - X_i^t = \alpha * \mathrm{randn}() * D_i + \beta * (X_{best}^t - X_i^t)$$

Taking the expected value E[ ] yields:

$$E[X_i^{t+1} - X_i^t] = \beta * (X_{best}^t - X_i^t)$$

This indicates that the expectation of the random perturbation is zero, primarily serving an exploratory purpose. The global guidance term ensures the direction of the search.

Analysis of Global Optimal Solution Trend

If $t \to \infty$ and $\beta > 0$, the individual tends to X_{best}. The perturbation term ensures that the search space is fully explored before convergence, reducing the probability of falling into a local optimum.

VI. Potential Applications and Discussion

The IDO algorithm is applicable to:

Continuous optimization problems: Minimization of functions in multidimensional real space.

Multimodal optimization problems: Local perturbation mechanisms help escape local optima.

Constrained optimization problems: These can be solved by combining penalty functions or boundary constraints.

Dynamic optimization problems: Perturbation mechanisms simulate environmental changes and can adapt to dynamic search spaces.

Compared to traditional particle swarm optimization algorithms or genetic algorithms, the perturbation propagation mechanism introduced by IDO provides a unique search strategy. Combining global guidance with local exploration, it theoretically improves convergence speed and global search capability.

VII. Conclusion

The Planetary Ionospheric Perturbation Optimization (IDO) algorithm proposed in this paper:

Learning from the multi-source perturbation and propagation characteristics of planetary ionospheric perturbations, a heuristic search mechanism is designed.

The algorithm's initialization, perturbation update, propagation attenuation, fitness selection, and iterative convergence are systematically modeled using plain text mathematical formulas.

The algorithm's heuristic properties, adaptive propagation, global convergence capability, and scalability are analyzed.

This algorithm provides a new heuristic solution method for continuous optimization problems. It can theoretically handle multimodal and dynamic optimization problems, providing new insights for the development of heuristic optimization algorithms.

References

1. Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The arithmetic optimization algorithm. Computer methods in applied mechanics and engineering, 376, 113609.
2. Khishe, M., & Mosavi, M. R. (2020). Chimp optimization algorithm. Expert systems with applications, 149, 113338.
3. Juneja, M., & Nagar, S. K. (2016, October). Particle swarm optimization algorithm and its parameters: A review. In 2016 International conference on control, computing, communication and materials (ICCCCM) (pp. 1-5). IEEE.
4. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. Advances in engineering software, 95, 51-67.
5. Rajabioun, R. (2011). Cuckoo optimization algorithm. Applied soft computing, 11(8), 5508-5518.
6. Rahkar Farshi, T. (2021). Battle royale optimization algorithm. Neural Computing and Applications, 33(4), 1139-1157.
7. Zhang, J. Supernova Explosion Optimization Algorithm. https://doi.org/10.13140/RG.2.2.16625.75365
8. Faramarzi, A., Heidarinejad, M., Stephens, B., & Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. Knowledge-based systems, 191, 105190.
9. Ghaemi, M., & Feizi-Derakhshi, M. R. (2014). Forest optimization algorithm. Expert systems with applications, 41(15), 6676-6687.
10. Zhang, Y., Wang, S., & Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. Mathematical problems in engineering, 2015(1), 931256.
11. Zhou, F. H., & Liao, Z. Z. (2013). A particle swarm optimization algorithm. Applied Mechanics and Materials, 303, 1369-1372.
12. Jiang, Y., Hu, T., Huang, C., & Wu, X. (2007). An improved particle swarm optimization algorithm. Applied Mathematics and Computation, 193(1), 231-239.

13. Du, H., Wu, X., & Zhuang, J. (2006, September). Small-world optimization algorithm for function optimization. In International conference on natural computation (pp. 264-273). Berlin, Heidelberg: Springer Berlin Heidelberg.
14. Hashim, F. A., & Hussien, A. G. (2022). Snake Optimizer: A novel meta-heuristic optimization algorithm. Knowledge-Based Systems, 242, 108320.
15. Rao, R. V. (2019). Jaya: an advanced optimization algorithm and its engineering applications.
16. Zhang, J. Lunar Crater Projection Optimization. https://doi.org/10.13140/RG.2.2.35513.22889
17. Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-Qaness, M. A., & Gandomi, A. H. (2021). Aquila optimizer: a novel meta-heuristic optimization algorithm. Computers & Industrial Engineering, 157, 107250.
18. Ahmadianfar, I., Bozorg-Haddad, O., & Chu, X. (2020). Gradient-based optimizer: A new metaheuristic optimization algorithm. Information Sciences, 540, 131-159.
19. Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. Future generation computer systems, 97, 849-872.
20. Hu, G., Zhong, J., Du, B., & Wei, G. (2022). An enhanced hybrid arithmetic optimization algorithm for engineering applications. Computer Methods in Applied Mechanics and Engineering, 394, 114901.
21. Zhang, J. Lunar Occultation Optimization. https://doi.org/10.13140/RG.2.2.28802.34243
22. Abdel-Basset, M., Mohamed, R., Jameel, M., & Abouhawwash, M. (2023). Spider wasp optimizer: A novel meta-heuristic optimization algorithm. Artificial Intelligence Review, 56(10), 11675-11738.
23. Zervoudakis, K., & Tsafarakis, S. (2020). A mayfly optimization algorithm. Computers & Industrial Engineering, 145, 106559.
24. Masadeh, R., Mahafzah, B. A., & Sharieh, A. (2019). Sea lion optimization algorithm. International Journal of Advanced Computer Science and Applications, 10(5).
25. Ahmadianfar, I., Heidari, A. A., Noshadian, S., Chen, H., & Gandomi, A. H. (2022). INFO: An efficient optimization algorithm based on weighted mean of vectors. Expert Systems with Applications, 195, 116516.
26. Sharma, H., Hazrati, G., & Bansal, J. C. (2018). Spider monkey optimization algorithm. In Evolutionary and swarm intelligence algorithms (pp. 43-59). Cham: Springer International Publishing.
27. Betts, J. T., & Frank, P. D. (1994). A sparse nonlinear optimization algorithm. Journal of Optimization Theory and Applications, 82(3), 519-541.
28. Ghaemi, M., & Feizi-Derakhshi, M. R. (2016). Feature selection using forest optimization algorithm. Pattern Recognition, 60, 121-129.
29. Wang, F., Wang, X., & Sun, S. (2022). A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization. Information Sciences, 602, 298-312.
30. Kaur, G., & Arora, S. (2018). Chaotic whale optimization algorithm. Journal of Computational Design and Engineering, 5(3), 275-284.
31. Qais, M. H., Hasanien, H. M., & Saad, A. (2020). Transient search optimization: a new meta-heuristic optimization algorithm. Applied Intelligence, 50(11), 3926-3941.
32. Karaboga, D., & Basturk, B. (2007, June). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In International fuzzy systems association world congress (pp. 789-798). Berlin, Heidelberg: Springer Berlin Heidelberg.

33. Ahmadianfar, I., Heidari, A. A., Gandomi, A. H., Chu, X., & Chen, H. (2021). RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method. Expert Systems with Applications, 181, 115079.
34. Gou, J., Lei, Y. X., Guo, W. P., Wang, C., Cai, Y. Q., & Luo, W. (2017). A novel improved particle swarm optimization algorithm based on individual difference evolution. Applied Soft Computing, 57, 468-481.
35. Ab Wahab, M. N., Nefti-Meziani, S., & Atyabi, A. (2015). A comprehensive review of swarm optimization algorithms. PloS one, 10(5), e0122827.
36. Chen, C. Y., & Ye, F. (2012, May). Particle swarm optimization algorithm and its application to clustering analysis. In 2012 Proceedings of 17th Conference on Electrical Power Distribution (pp. 789-794). IEEE.
37. Rao, R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, 7(1), 19-34.
38. Bandyopadhyay, S., Saha, S., Maulik, U., & Deb, K. (2008). A simulated annealing-based multiobjective optimization algorithm: AMOSA. IEEE transactions on evolutionary computation, 12(3), 269-283.
39. Ray, T., & Liew, K. M. (2003). Society and civilization: an optimization algorithm based on the simulation of social behavior. IEEE Transactions on Evolutionary Computation, 7(4), 386-396.
40. Liao, T., Stützle, T., de Oca, M. A. M., & Dorigo, M. (2014). A unified ant colony optimization algorithm for continuous optimization. European Journal of Operational Research, 234(3), 597-609.
41. Lange, K. (2016). MM optimization algorithms. Society for Industrial and Applied Mathematics.
42. Zhang, J. Planetary Rotation Day-Length Optimization. https://doi.org/10.13140/RG.2.2.26692.08328
43. Moosavi, S. H. S., & Bardsiri, V. K. (2019). Poor and rich optimization algorithm: A new human-based and multi populations algorithm. Engineering applications of artificial intelligence, 86, 165-181.
44. Onwunalu, J. E., & Durlofsky, L. J. (2010). Application of a particle swarm optimization algorithm for determining optimum well location and type. Computational Geosciences, 14(1), 183-198.
45. Bartz-Beielstein, T., Chiarandini, M., Paquete, L., & Preuss, M. (Eds.). (2010). Experimental methods for the analysis of optimization algorithms (pp. 311-336). Berlin: Springer.
46. Tanweer, M. R., Suresh, S., & Sundararajan, N. (2015). Self regulating particle swarm optimization algorithm. Information Sciences, 294, 182-202.
47. Karaboğa, D., & Ökdem, S. (2004). A simple and global optimization algorithm for engineering problems: differential evolution algorithm. Turkish Journal of Electrical Engineering and Computer Sciences, 12(1), 53-60.
48. Črepinšek, M., Liu, S. H., & Mernik, L. (2012). A note on teaching – learning-based optimization algorithm. Information Sciences, 212, 79-93.
49. Amiri, M. H., Mehrabi Hashjin, N., Montazeri, M., Mirjalili, S., & Khodadadi, N. (2024). Hippopotamus optimization algorithm: a novel nature-inspired optimization algorithm. Scientific Reports, 14(1), 5032.
50. Civicioglu, P. (2013). Backtracking search optimization algorithm for numerical optimization problems. Applied Mathematics and computation, 219(15), 8121-8144.

51. Kaveh, A., & Dadras, A. (2017). A novel meta-heuristic optimization algorithm: thermal exchange optimization. Advances in engineering software, 110, 69-84.