Assignment3
Particle Swarm Optimization

Xiyao Huang
Department of Computer Science
St. Francis Xavier University
Antigonish, Canada
x2023fkv@stfx.ca
202306631

I. Introduction

The basic task of this job is to implement particle swarm optimization (PSO) on a series of test optimization functions, add a series of enhancements to it, and finally use PSO to solve multi-objective problems based on the basic tasks. Since the particle swarm optimization (PSO) algorithm was proposed by Kennedy and Eberhart in 1995 [1], it has been widely studied and applied to various optimization problems. PSO is one of the most important swarm intelligence methods and evolutionary computing algorithms. This is due to its ability to adapt to dynamic environments and its low constraints on the continuity of the objective function and the connectivity of the search space. In short, PSO explores the global optimal solution by utilizing the memory of particles and groups. [2] In 2019, Jia Guo and Yuji Sato proposed a fission-fusion hybrid bare bones particle swarm optimizer (FHBBPSO). The algorithm combines the fission strategy and the aggregation strategy to sample the new position of particles. They divide the search space through the fission strategy so that particles are assigned to different local groups, and use the aggregation strategy to narrow the search space and approach the theoretical optimal value, gradually merging the edge groups into the center group, so that only one group remains in the end. [3] In 2023, Zhan, Jianjun, et al. proposed an improved particle swarm optimization algorithm (IPSO) to solve global optimization and hyperparameter optimization problems in order to reduce the probability of particles falling into local optimality and alleviate the premature convergence of the particle swarm optimization algorithm (PSO) and the imbalance between development and exploration. [4] However, PSO still faces challenges such as decreased high-dimensional search performance and parameter sensitivity. Therefore, improving its algorithm performance and balancing exploration and development capabilities are of great significance for solving dynamic, multi-objective or constrained optimization problems.

II. Problem description

A. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a powerful optimization technique mainly used for single-objective problems. [3] PSO is very efficient in dealing with real-valued or continuous optimization problems. Compared with other evolutionary computing techniques such as GA and ACO, it is relatively simple to implement and requires fewer hyperparameters to be tuned. [5]

B. Multi-Objective Optimization

In many practical scenarios, optimization is not just about one goal, but often involves multiple conflicting goals. For example, in scheduling problems, both efficiency and resource utilization must be improved. Using multi-objective test problems ensures that the algorithm can find a set of diverse and high-quality solutions while evaluating the algorithm's ability to handle the problem.

III. Algorithm Description

A. Particle Swarm Optimization

1) Implementation:

- Test Functions: There are two functions for single-objective optimization and mutiple-objective optimization respectively. The sphere function is used for single-objective optimization to minimize the sum of squares of all variables.
- Initialization: A Particle class is created to manage the properties of particles, such as position, speed, personal optimal solution and fitness value. During initialization, the particle swarm size, that is, number of particles, problem dimension and search boundary are defined to ensure that the particle distribution covers the entire search space, thereby increasing the possibility of finding the global optimal solution. Each particle represents a potential solution, and its position and speed are randomly initialized and their range is guaranteed to be within the specified boundary. For multi-objective optimization, the Pareto set is additionally initialized.
- Fitness Evaluation: To calculate its fitness, the position of each particle is evaluated by the objective function, and the fitness is a scalar value. If the current fitness is better than the personal best value recorded by the particle, the particle's personal best position and value are updated.
- Update: There are two parts involved in the update, particle speed and position. The speed will be updated

through the inertia term, cognitive term and social term. The inertia term maintains the original speed of the particle, the cognitive term pulls the particle to its personal optimal position, and the social term pulls the particle to the global optimal position. Randomness is introduced through random coefficients (r1 and r2) to prevent the particle's movement path from being completely determined. The particle's position is updated with its updated velocity. If the particle is outside the specified bounds, its position is adjusted to fit within the bounds.

- Visualization: For single-objective optimization, a convergence curve is plotted to show the change in optimal fitness during the iteration process, so as to intuitively display the algorithm optimization process.

2) Enhancements:

- Speed Limits: Use the np.clip function to limit the speed range to prevent instability in the search process and to avoid convergence out of control due to excessive particle speed.
- Dynamic Search Space: In order to ensure the validity of the solution and improve the search efficiency, the particles are forced to remain within the search space boundary and their velocity is reset to zero at the boundary to prevent the particles from exceeding the boundary, which leads to invalid search.
- Adaptive Parameters: By dynamically adjusting the cognitive coefficient c1 and social coefficient c2, the algorithm performance at different stages can be improved. This effectively reduces the burden of manual parameter adjustment and improves the algorithm adaptability and convergence effect to a certain extent.
- Inertia Weight Decay: By gradually reducing the inertia weight with iterations, the goal of balancing global search and local search is achieved. This makes the entire algorithm more inclined to global search in the early stage and will not fall into the local optimum too early; in the later stage, the local search accuracy is enhanced because of the reduction of the weight. This enhancement improves the convergence efficiency of the algorithm and balances the global and local search capabilities.

B. Multi-Objective Optimization

1) Implementation:

- Test Functions: For multi-objective optimization, a zdt1 function is set to test the multi-objective optimization performance through two objective values f1 and f2.
- Initialization: For multi-objective optimization, based on the initialization of the single-objective problem, the Pareto set is additionally initialized.
- Fitness evaluation: For multi-objective optimization, the fitness is composed of multiple objectives. The Pareto dominance rule is used to compare solutions and update the Pareto frontier of each particle, ensuring that non-dominated solutions are retained.
- Update: The same with the single-objective one.
- Selection: In multi-objective optimization, to update the Pareto frontier, it is necessary to discard the dominated solutions and only keep the solutions with the optimal trade-off between the objectives, which is achieved by selecting non-dominated solutions through the Pareto dominance rule.
- Visualization: For multi-objective optimization, a scatter plot of the Pareto frontier is drawn to show the trade-off relationship between the objectives. It helps to evaluate the performance of the algorithm

IV. Results

Fig. 1. shows the results obtained using the Sphere function in a single-objective optimization task. From this Convergence Curve, we can see that the PSO algorithm successfully minimizes the fitness value in about 98 iterations. The final optimal fitness value is about 0.1488, showing the ability of the algorithm to converge to a near-optimal solution. The convergence curve of the Sphere function shows that the fitness value decreases rapidly in the early stage, indicating that the algorithm can efficiently find high-quality solutions in early iterations. This is in line with expectations for PSO's global search capabilities. The fitness value in the later period tends to be stable, indicating that the algorithm shifts to local search.

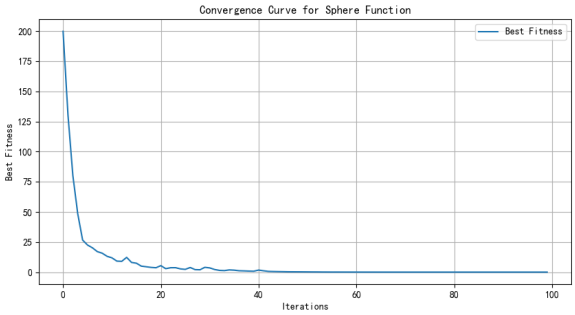

Fig. 1. Sphere Function

In the multi-objective optimization task, the ZDT1 function is used. From Fig. 2. It can be seen that the algorithm generates a Pareto front containing 9 non-dominated solutions and visualizes the trade-off relationship between objectives f1 and f2, which illustrates that the algorithm can Effectively approximates the theoretical Pareto optimal solution set. The results were also statistically analyzed using two PSO variants with different speed constraints (-1,1) and (-2,2) for statistical comparison. The average fitness value is calculated: PSO1 (speed constraint: -1,1) has an average fitness value of 0.2669, while PSO2 (speed constraint: -2,2) has an average fitness value of 0.5400. And the p value of the T-test is 0.0048, while the p value of the Mann-Whitney U test is 0.0018. The p values of both results indicate that the difference in the mean values of the two algorithms is statistically significant. With Cohen's d value of -0.7771, it can be concluded that there is a medium effect size between the two methods, and PSO1 performs better. Overall, the algorithm PSO1 with a smaller

speed constraint range has better optimization performance on the Sphere function.

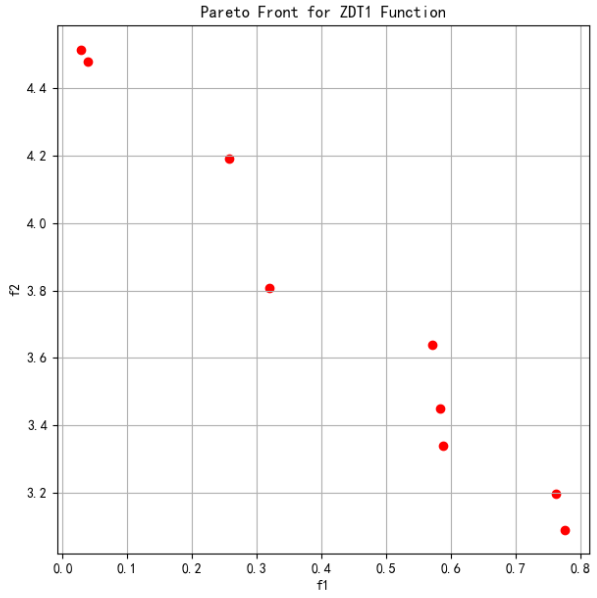

Fig. 2. ZDT1 Function

V. Discussion

Compared with random search, PSO significantly speeds up convergence and improves the quality of solutions because of its ability to exploit memory and group interactions between particles, while random search lacks these properties. Compared with other algorithms, such as GA, PSO converges faster because it does not require complex crossover and mutation operations and requires fewer hyperparameters to be adjusted. The four enhancement functions adopted in the algorithm all have positive effects. First, limiting the speed range plays a key role in maintaining search stability and accelerating convergence. From the results of statistical analysis, smaller speed ranges performed better in this study. Secondly, dynamic search space adjustment ensures the effectiveness of understanding, especially near the boundaries, improving the stability of understanding. The inertia weight attenuation achieves a balance between early global search and late local search by gradually reducing the inertia weight during the iteration process. Finally, the adaptive parameters improve the performance of the algorithm at different stages by dynamically adjusting the cognitive coefficient c1 and social coefficient c2, and enhance the adaptability and convergence effect of the algorithm.

VI. Conclusion

The PSO algorithm implemented in this task can efficiently solve single-objective problems, and can effectively achieve multi-objective optimization tasks, and achieved competitive results. A total of four enhancements have been implemented, namely Speed Limits, Dynamic Search Space, Adaptive Parameters, and Inertia Weight Decay. The implemented enhancements have significantly improved algorithm performance, especially in terms of the balance between global exploration and local development. Statistical analysis shows that the choice of parameters is sometimes crucial, and the results show that a smaller speed range often works better on the Sphere function. The algorithm performs well overall, showing efficient convergence in both optimization tasks. Its capabilities in multi-objective optimization are also demonstrated by the Pareto front generated for the ZDT1 function. There are still several aspects that can be improved for continued research: First, more advanced enhancement functions can be performed, such as applying adaptive speed constraints or some learning-based parameter tuning. Secondly, you can also try to test the algorithm on high-dimensional problems to evaluate its scalability and robustness. Of course, PSO can also be combined with other technologies to take advantage of complementary advantages. In summary, the PSO algorithm and its enhanced functions implemented this time have achieved good optimization results and effectively solved the multi-objective problems raised in the assignment.

References

[1] Kennedy, J., and R. Eberhart. “Particle Swarm Optimization.” Proceedings of ICNN’95 - International Conference on Neural Networks, vol. 4, IEEE, 1995, pp. 1942–48 vol.4, https://doi.org/10.1109/ICNN.1995.488968.

[2] Mei-Ping Song, and Guo-Chang Gu. “Research on Particle Swarm Optimization: A Review.” Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), vol. 4, IEEE, 2004, pp. 2236–41 vol.4, https://doi.org/10.1109/ICMLC.2004.1382171.

[3] Guo, Jia, and Yuji Sato. “A Fission-Fusion Hybrid Bare Bones Particle Swarm Optimization Algorithm for Single-Objective Optimization Problems.” Applied Intelligence (Dordrecht, Netherlands), vol. 49, no. 10, 2019, pp. 3641–51, https://doi.org/10.1007/s10489-019-01474-9.

[4] Zhan, Jianjun, et al. “Improved Particle Swarm Optimization Algorithm Based on Grouping and Its Application in Hyperparameter Optimization.” Soft Computing (Berlin, Germany), vol. 27, no. 13, 2023, pp. 8807–19, https://doi.org/10.1007/s00500-023-08039-6.

[5] Song, Ming Li. “A Study of Single-Objective Particle Swarm Optimization and Multi-Objective Particle Swarm Optimization.” Applied Mechanics and Materials, vol. 543–547, no. Vehicle, Mechatronics and Information Technologies II, 2014, pp. 1635–38, https://doi.org/10.4028/www.scientific.net/AMM.543-547.1635.