Handling outliers in swarm algorithms: a review

Dmytro Uzlov[1], Yehor Havryliuk[1], Ivan Hushchyn[1], Volodymyr Strukov[1], and Sergiy Yakovlev[1]

[1] *V.N. Karazin Kharkiv National University. 4 Svobody, Sq., Kharkiv, 61022, Ukraine*

Abstract

Swarm optimization algorithms, inspired by the collective behavior of biological swarms, are a promising tool for solving the problem of optimizing complex systems where traditional methods are often ineffective. However, the problem of outliers can significantly affect the process of finding an optimal solution. Therefore, the study of methods for detecting and processing outliers in swarm algorithms, such as the particle swarm optimization (PSO), is an urgent task that has significant potential to improve the efficiency and reliability of these algorithms in various practical applications, such as drone control systems, financial systems, environmental control and modeling systems. The article deals with the problem of outliers in swarm optimization algorithms such as PSO. An overview of existing approaches to managing outliers, including adaptive methods, methods using swarm topologies, hybrid algorithms, and others, is provided. The advantages and disadvantages of each approach are analyzed. Particular attention is paid to new promising areas, such as the combination of neural networks and reinforcement learning, to develop more efficient and adaptive swarm algorithms. The article is aimed at researchers and practitioners in the field of optimization who are interested in improving the efficiency and reliability of swarm algorithms.

Keywords

Swarm algorithms, particle swarm optimization, deep learning. handling outliers.

1. Introduction

Swarm optimization algorithms, inspired by biological swarms, are crucial for solving complex problems in fields like engineering, economics, and medicine. Despite their power, these algorithms often suffer from early convergence, where particles get stuck in local optima due to outliers. This article reviews methods for detecting and managing outliers in swarm systems on the example of PSO, analyzing adaptive methods, swarm topologies, and hybrid algorithms. It also explores emerging solutions like neural networks and reinforcement learning hybridization, which could enhance swarm algorithms' ability to avoid local optima and find global solutions.

2. Problem statement and state of the arts

In [1], the authors emphasize that an "outlier is strange data values that stand out from datasets". From this definition, outliers in swarm systems can be represented as particles in the swarm that do not follow the expected swarm behavior, such as particles that move much faster or slower than other particles in the swarm. Such particles have all the characteristics inherent in the standard definition of an outlier: they can deviate significantly from the swarm trajectory, interfere with other particles, and prevent them from moving toward the optimal solution. This leads to slower swarm convergence or suboptimal results.

The issue of outliers in swarm algorithms is underexplored but crucial, as optimizing them could greatly enhance swarm convergence, benefiting many modern applications. Swarm systems, such as drone control systems, are widely used in a variety of areas, including transportation systems [2],

ProfIT AI 2024: 4th International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2024), September 25–27, 2024, Cambridge, MA, USA

dmytro.uzlov@karazin.ua (D. Uzlov); gavrilyk2903@gmail.com (Y. Havryliuk); gushchin.iv@gmail.com (I.Hushchyn); struk_vm@ukr.net (V. Strukov); svsyak7@gmail.com (S. Yakovlev)

0000-0003-3308-424X (D. Uzlov); 0000-0002-4392-2000 (Y. Havryliuk); 0000-0002-1917-716X (I. Hushchyn); 0000-0003-4722-3159 (V. Strukov); 0000-0003-1707-843X (S. Yakovlev)

search and rescue operations [3], and other industries. In such systems, rogue drones can not only slow down the convergence of the swarm, but also lead to a loss of control over individual drones, which can cause safety hazards and involve damage to property or people.

In financial systems where swarm systems such as PSO are used to optimize investment portfolios, outliers can cause unpredictable fluctuations in portfolio value [4]. These fluctuations can negatively affect the stability and predictability of investments, creating additional risks for financial markets.

Outliers can have severe consequences in control systems [5], leading to unpredictable behavior and potential damage. In environmental models, they can cause inaccurate predictions. There is no universal solution for handling outliers in swarm systems, as hyperparameters in swarm algorithms greatly impact their efficiency and stability. Tailoring strategies to specific problems and exploring alternative outlier control methods are key research priorities. Developing effective detection and management techniques for outliers in swarm systems is crucial and requires innovative approaches.

3. State of the arts

The outlier problem, though well-known in statistics, is especially crucial in swarm algorithms. Viewing swarm particles as data points frames the outlier issue as a statistical anomaly detection problem, where outliers signal an imbalance in exploration and exploitation. Traditional methods, like removing outliers [6], aren't always viable, especially in applications like drone systems where losing a unit isn't acceptable. Thus, exploring alternative outlier management strategies is essential.

Recent studies have proposed a wide range of variations of the PSO algorithm aimed at solving various problems and improving the basic algorithm. These variations include adaptive approaches, where hyperparameters dynamically change during the optimization process, considering the current state of the swarm and the characteristics of the problem. Researchers also consider methods that utilize swarm topologies (in a standard PSO, one of the topologies shown in Figure 1 is often used) [7], which affect the information exchange between particles, allowing for more efficient exploration of the solution space and avoiding early convergence. In addition, hybrid algorithms combining PSO with other optimization methods, such as genetic algorithms or bee swarm methods, are actively being investigated to combine the advantages of different approaches [7, 8].

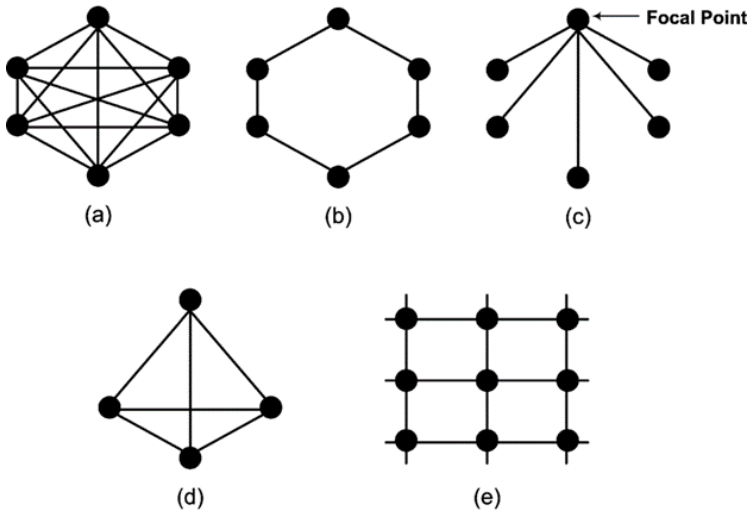

Figure 1: Standard swarm topologies: (a) Global-Best, (b) Ring, (c) Wheel, (d) Pyramid, (e) Von Neumann

The research papers [7, 8] provide an overview of such modern variations of PSO, their processing principles and purpose. Further analysis of these variations, presented in this paper, will allow to deeper understand their advantages and disadvantages, as well as to determine the optimal vector for future research on the outlier problem not only in PSO, but also in the domain of swarm

algorithms in general, and finding a solution that will increase the efficiency and reliability of swarm algorithms in various practical problems.

4. Managing outliers in swarm algorithms

4.1. Outliers causes

Outliers in swarm systems, as in any other systems where they occur, are stochastic in nature. It is proposed to study outliers in swarm systems on the example of one of the most common and simplest implementations of swarm intelligence algorithms for function optimization – PSO [9]. Determining PSO hyperparameters involves dealing with outlier particles that deviate from the swarm's general trend, making PSO ideal for studying outliers in swarm algorithms.

Among the main reasons for the occurrence of outliers in PSO algorithms are the following:

1. Initialization: some particles may start moving with initial values of position and velocity that are far from optimal
2. Particle divergence: particles explore the search space and may deviate from the swarm
3. Stochasticity: the algorithm is inherently random, potentially leading to some particles having significantly different positions or velocities than the rest of the swarm
4. Inappropriate hyperparameters: the behavior of PSO is strongly influenced by the choice of its hyperparameters

The initial position and velocity of particles in the simplest implementation are determined according to a uniform distribution in the search area. They also affect the speed of finding the optimal solution by a swarm. There are effective methods for solving this problem [10, 11].

Such causes of outliers as particle divergence, stochasticity, and inappropriate hyperparameters are related to the choice of parameters such as inertia weights and acceleration coefficients [9] - they affect the degree of exploration and exploitation of particles, thereby changing the behavior of agents. The choice of such parameters is usually a separate task when implementing PSO to optimize the objective function. For a static end state, it is possible to select such parameters. When the objective function changes dynamically, this approach becomes suboptimal, because each change in the objective function can potentially lead to unexpected results due to the static choice of initial parameters. To solve the problem of optimizing swarm outliers, as well as to overcome the need to find a compromise between the exploratory and exploitative behavior of a particle, many variations of PSO, were developed. For example, one of them is adaptive PSO.

Adaptive PSO (APSO) [12] has better search efficiency than the standard PSO, can perform a global search over the entire search space with a higher convergence rate, and can automatically control inertia weights, acceleration factors, and other algorithmic parameters at runtime, thereby improving search efficiency and performance simultaneously. In addition, this algorithm can influence a single swarm particle with the global best found value to "force" it out of the likely local optimum.

But as any other variation mentioned in this article, APSO has its own set of limitations and potential problems. Some of the most common problems include [12]:

- Over-adaptation
- Complexity
- Convergence
- Performance

Although APSO is widely utilized among PSO variants, it may not represent the most optimal approach for all application scenarios.

4.2. An overview of existing PSO variations

APSO is by far the most common modification of the standard algorithm, being more flexible and more versatile, but it does not solve all problems and creates new disadvantages. We will consider other modifications of PSO further.

Scientists Meetu Jain, Vibha Saihjpal Narinder Singh, Satya Bir Singh noted the following variations of the PSO algorithm in their work, each aimed at solving the specific standard PSO limitation [8]:

1. Fuzzy adaptive PSO algorithm – improves PSO optimization capabilities
2. Homogeneous particle swarm optimizer (HPSO) – modified version for solving multi-objective optimization problems
3. Hybrid PSO with ranking, selection and mean square error criterion (STPSO) – combines PSO with statistical methods to solve stochastic optimization problems
4. Evolutionary modified PSO –improves search efficiency
5. Improved PSO algorithm (IPSO) – improves the search efficiency
6. Fully informed particles in PSO –improves performance

The authors of another paper [7], in addition to a general review of the PSO algorithm and its principles, provide a brief overview of the most recent PSO review documents, as well as a list of recent publications with PSO variations and their limitations. The main PSO variants the article focuses include the following:

1. Cooperative PSO – solving the outliers' problem through particle cooperation
2. Multi-swarm PSO – improves exploration avoiding local optima convergence
3. Hybrid PSO – improves performance in dynamic object layout problems
4. Binary PSO – can optimize both continuous and discrete functions

The article also provides an additional list of other PSO variations that are less popular or less efficient. These variations of PSO are usually aimed at eliminating a specific drawback of the standard algorithm, for example, the possibility of hitting a local optimum. But each of them also has its own drawbacks. For example, adding new parameters to the algorithm increases the complexity of the initial model setup and generally complicates the system with more hyperparameters, and, in addition, requires additional computational costs [7].

The PSO variations in [7, 8] show ongoing evolution and improvement for solving diverse optimization problems, but their multitude suggests that each may offer a less universal solution for specific problem types.

4.3. Comparative analysis of the PSO modifications

The analysis of recent publications on the topic [7, 8] shows that many researchers' efforts are focused on the development and improvement of the PSO algorithm. As a result of intensive scientific activity, a wide range of modifications and hybridizations of the basic PSO algorithm have been proposed, each of which has its own strengths and weaknesses. Based on the analysis of [7, 8], the authors of this article present a qualitative comparison of PSO variations (Table 1).

While these PSO variations offer improvements over the standard PSO, there is limited evidence of their effectiveness in real-world applications. Current literature indicates that no PSO variant is universally optimal or improves PSO without introducing new constraints. Thus, selecting a specific algorithm requires careful consideration of the problem's specifics, convergence speed, and solution accuracy, necessitating further research and experimentation.

Table 1
Qualitative comparison of different PSO modifications

Name	Description	Limitations
Standard PSO	Simple and resource efficient.	Particles can get stuck in local optima.
Adaptive PSO	Balances exploration and exploitation by dynamically adjusting the inertial weight, improving convergence speed and solution quality.	Increased complexity due to dynamic inertial weight, which requires additional tuning compared to fixed-parameter PSO options.
Cooperative PSO (CPSO)	Aimed at solving the problem of outliers through the cooperation of the particles.	Particles can get stuck in local optima.
Gaussian PSO (GPSO)	It only requires specifying the number of particles before use.	Not suitable for tasks where setting specific parameters is critical for optimal performance.
Concurrent PSO (CONPSO)	Improves convergence performance compared to the original PSO.	Increased computational complexity due to parallel operations.
Binary PSO (BPSO)	It can optimize both continuous and discrete functions.	Not always as effective as specialized algorithms for specific types of tasks (e.g., continuous vs. discrete).
Bare-Bones PSO	Eliminates the speed formula, making it simpler.	It is not always as effective in complex, multidimensional search spaces where speed control is important.
Fully Informed PSO (FIPS)	Particles are influenced by all neighbors, not just the best one.	Increased computational costs due to the consideration of information from all neighbors.
Binary PSO for Classification	Designed for classification tasks, showing promising results compared to machine learning methods.	Not suitable for other types of optimization problems.
Fuzzy Adaptive PSO (FAPSO)	Uses a fuzzy system to adapt the inertia weight, improving convergence.	Increased complexity due to the fuzziness of the system, requiring additional customization and expertise.
Guided PSO (GPSO)	Specially designed to recognize facial emotions, it demonstrates promising accuracy.	Limited applicability to other problem areas other than facial emotion detection.
Self-Regulating PSO (SRPSO)	Includes human learning strategies to improve exploration and exploitation processes.	Requires careful adjustment of self-regulation mechanisms for optimal performance.
Improved PSO (IPSO)	Solves the problems of slow convergence and limitations of the basic PSO when planning the trajectory of a mobile robot.	Tends to generalize poorly to other problem areas or demonstrate stable performance in different scenarios.
Genotype Phenotype Modified Binary PSO (GPMBPSO)	Designed to solve the Knapsack problem, offering improved performance compared to BPSO.	Increased complexity due to genotype-phenotype mapping, which may require additional computing resources.
Modified Binary PSO (MBPSO)	Outperforms the original BPSO algorithm.	It is not always suitable for other types of optimization problems and requires adaptation to a specific task.
Hybrid PSO (HPSO)	Combines PSO with other algorithms (e.g., simulated annealing) to improve performance in dynamic object layout problems.	Increased complexity due to its hybrid nature, requiring multiple algorithms to be configured.
STPSO (Stochastic PSO)	Hybridizes PSO with statistical methods to solve stochastic optimization problems.	Increased complexity due to its hybrid nature, requiring multiple algorithms to be configured.

While these PSO variations offer improvements over the standard PSO, there is limited evidence of their effectiveness in real-world applications. Current literature indicates that no PSO variant is universally optimal or improves PSO without introducing new constraints. Thus, selecting a specific

algorithm requires careful consideration of the problem's specifics, convergence speed, and solution accuracy, necessitating further research and experimentation.

Having examined this, the authors believe that it is advisable to consider other ways to improve the PSO algorithm, other than the above approaches. One of these alternatives is to integrate PSO with neural networks (NN), which will allow to use the advantages of both approaches. PSO can be used to optimize the NN architecture, find optimal values of weights and thresholds, or find optimal hyperparameters. In turn, NNs can be used to model complex nonlinear dependencies and improve PSO's ability to find solutions.

In addition, combining PSO with NNs can be especially useful in problems where many parameters need to be considered or where the objective function is complex and multimodal. In such cases, NNs can help PSO avoid local optima and find more accurate solutions (resolving outliers' problem as well).

The use of hybrid approaches that combine PSO with NN may grant new opportunities for solving complex optimization problems and improving the efficiency of existing solutions. To confirm this hypothesis, additional research and experiments are needed to assess the potential of this approach and determine its advantages and disadvantages.

5. Deep learning models used over PSO

Modifying and hybridizing PSO isn't the only way to improve it. Paper [8] reviews how integrating collective intelligence, like self-organization and swarm intelligence, can enhance deep learning. The authors explore using these principles to address deep learning challenges, such as combining cellular automata with neural networks for image processing and rethinking reinforcement learning with self-organizing agents. The authors identify four main areas of deep learning that have begun to incorporate the ideas of collective intelligence:

1. Image processing
2. Deep Reinforcement Learning (DRL)
3. Multi-agent learning
4. Meta learning

Based on these studies, it can be assumed that the introduction of a reinforcement learning model for outlier's optimization in PSO has prerequisites for future research to overcome the limitations of other algorithm modifications. For instance, it can be predicted that one of the potential advantages of integrating reinforcement learning with PSO is that the introduction of neural network models into swarm operation will solve the problem of overfitting. Reinforcement learning algorithms are designed to learn optimal policies that generalize well to new environments [13], so it is reasonable to consider such integration as conducive to an effective process of adaptation to various optimization problems.

Let us consider an example of a possible potential application of the hybrid PSO-DRL approach. The GPT models, such as ChatGPT by OpenAI, which have become a modern breakthrough in the field of artificial intelligence, use deep neural networks with many parameters, which makes their training and tuning a complex and resource-intensive process [14]. Using PSO to optimize the GPT model architecture can help find the optimal number of layers, neurons in each layer, and types of connections between them. This will reduce the number of model parameters, speed up its training, and improve its ability to generate text.

In addition, PSO can be used to optimize hyperparameters like learning rate, data set size, and regularization, balancing training speed and model accuracy. DRL can help model complex relationships between parameters and performance, enhancing PSO's efficiency in finding optimal solutions. Some studies, like [15], have explored combining these methods, showing that the introduced parameter adaptation method based on reinforcement learning (RLAM) improves PSO's convergence rate and outperforms other variants. However, RLAM increases computational complexity, complicates implementation, and risks overfitting. Despite these challenges, combining PSO and DRL could effectively optimize GPT models, improving performance, reducing resource use, and speeding up development. It is also promising to combine the modified PSO algorithm with reinforcement learning models. Such integration has the prerequisites for further improving the convergence property of the modified algorithm. Optimal policies in reinforcement learning

algorithms are obtained by maximizing the reward signal, which can be used to control the search process in an adaptive algorithm [16].

However, in the context of such integration, it is also worth noting potential limitations:

1. Additional complexity of initialization
2. Setting additional hyperparameters
3. Over-fitting of the RL model

In addition, determining the best strategy for integrating the reinforcement learning algorithm with PSO for a particular optimization problem, as well as finding another possible method for combining the two algorithms that could reduce the number of algorithm limitations while improving the performance of the standard PSO, requires additional research. The implementation of this approach, as well as experimental confirmation or refutation of its advantages and disadvantages, is the subject of the author's future research.

6. Conclusions

Outliers in particle swarm optimization are a major challenge, influenced by factors like velocity, position, and acceleration coefficients. Addressing their causes can improve convergence speed, accuracy, stability, and reliability in complex search spaces. This article examines the causes, concepts, and solutions to outlier issues in swarm optimization on the example of PSO, focusing on methods that enhance convergence and reduce outliers, including adaptive methods, swarm topologies, and hybrid algorithms.

For example, the adaptive particle swarm method balances exploration and fast convergence but is more complex than standard PSO. Cooperative PSO aids particle cooperation but can get stuck in local optima, while binary PSO handles both continuous and discrete parameters but may be less efficient than specialized algorithms. It has been determined that none of the existing variations of PSO is a universal solution; each comes with its own limitations.

It has been proposed to use of hybrid approaches combining PSO with neural networks and reinforcement learning, which will grant new opportunities for solving complex optimization problems and improving the efficiency of existing solutions. In contrast to algorithmic solutions, the use of neural networks in combination with the particle swarm method (or its variations) would be appropriate to obtain a positive practical result when applied to drone control systems or financial systems for which other variations of algorithms are not optimal for one reason or another.

Further research will be aimed at studying and experimentally confirming or refuting the advantages and disadvantages of the proposed approach, as well as developing a new method for effective detection and management of outliers in swarm systems on the example of PSO.

References

[1] Misinem, A. A. Bakar, A. R. Hamdan, M. Z. A. Nazri. "A rough set outlier detection based on particle swarm optimization", 10th international conference on intelligent systems design and applications, Cairo, Egypt, 2010. pp. 1021-1025. doi: 10.1109/ISDA.2010.5687054.

[2] F. Schiano, P. M. Kornatowski, L. Cencetti, D. Floreano. "Reconfigurable drone system for transportation of parcels with variable mass and size." IEEE Robotics and Automation Letters, vol. 7, no. 4, pp. 12150-12157. Oct. 2022. doi: 10.1109/LRA.2022.3208716.

[3] A. A. Nathan, J. Rakesh, I. Kurmi, O. Bimber. "Drone swarm strategy for the detection and tracking of occluded targets in complex environments." Communications Engineering, vol. 2, 55, 2023. doi: 10.1038/s44172-023-00104-0.

[4] S. G. Reid, K. M. Malan, A. P. Engelbrecht. "Carry trade portfolio optimization using particle swarm optimization." 2014 IEEE Congress on Evolutionary Computation (CEC). Beijing, China, pp. 3051-3058. 2014. doi: 10.1109/CEC.2014.6900497.

[5] J. Hamidi. "Control system design using particle swarm optimization (PSO)." International Journal of Soft Computing and Engineering: Blue Eyes Intelligence Engineering & Sciences Publication, vol. 1, issue 6, pp. 2231-2307. 2012. URL: https://www.ijsce.org/wp-content/uploads/papers/v1i6/F0280111611.pdf.

[6] T.W. Gress, J. Denvir, J.I. Shapiro. "Effect of removing outliers on statistical inference: implications to interpretation of experimental data in medical research." Marshall Journal of Medicine, vol. 4, issue 2.9. 2018. doi: 10.18590/mjm.
[7] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, S. Mirjalili. "Particle swarm optimization: A comprehensive survey." IEEE Access, vol. 10, pp. 10031-10061. 2022. doi: 10.1109/ACCESS.2022.3142859.
[8] M. Jain, V. Saihjpal, N. Singh, S.B. Singh. "An overview of variants and advancements of PSO algorithm." Appl. Sci. vol. 12, no. 17: 8392. 2022. doi: 10.3390/app12178392.
[9] J. Kennedy, R. Eberhart. "Particle swarm optimization." Proceedings of IEEE International Conference on Neural Networks, vol. IV. pp. 1942–1948. 1995. doi:10.1109/ICNN.1995.488968.
[10] X. Hu, R. Shonkwiler, M. Spruill. "Random Restarts in Global Optimization." Georgia Tech Library. Georgia Institute of Technology: School of Mathematics. 2009. URL: http://hdl.handle.net/1853/31310.
[11] M. Barad. "Design of Experiments (DOE) — A Valuable Multi-Purpose Methodology." Applied Mathematics, vol. 5, no 14, pp. 2120-2129. 2014. doi: 10.4236/am.2014.514206.
[12] Z-H. Zhan, J. Zhang, Y. Li, H.S-H. Chung. "Adaptive Particle Swarm Optimization." IEEE Transactions on Systems, Man, and Cybernetics, vol. 39 (6), pp. 1362-1381. doi:10.1109/TSMCB.2009.2015956.
[13] J. Shuford. "Deep Reinforcement Learning: Unleashing the Power of AI in Decision-Making." Journal of Artificial Intelligence General Science JAIGS, vol. 1, issue 1. 2024. URL: https://www.researchgate.net/publication/378335647_ARTICLE_INFO_Deep_Reinforcement_Learning_Unleashing_the_Power_of_AI_in_Decision-Making
[14] A. Birhane, A. Kasirzadeh, D. Leslie. "Science in the age of large language models." Nature Reviews Physics, vol. 5, pp. 277–280. doi: 10.1038/s42254-023-00581-4.
[15] S. Yin, M. Jin, H. Lu. "Reinforcement-learning-based parameter adaptation method for particle swarm optimization." Complex Intell. Syst, vol. 9, pp. 5585–5609. 2023. doi: 10.1007/s40747-023-01012-8.
[16] L. P. Kaelbling, M. L. Littman, A. W. Moore. "Reinforcement Learning: A Survey." Journal of Artificial Intelligence Research, vol. 4. pp. 237–285. 1996. doi: 10.1613/jair.301.