Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2021.020307
ISSN 2790-0916 Vol. 2, Issue 3: 58-65

Adaptive Integration Algorithm for Distributed System Based on Particle Swarm Optimization

Mishra Manohar*

Radboud Univ Nijmegen, Donders Inst Brain Cognit & Behav, Nijmegen, Netherlands

**corresponding author*

Keywords: Particle Swarm Optimization Algorithm, DS, Information Processing, Distributed Nodes

Abstract: With the development of society, various optimization problems continue to emerge, making optimization a very applicable research field and attracting the attention of many researchers. Particle swarm optimization (PSO) algorithm algorithm is a technology proposed for optimization problems, so this paper studies DSs based on PSO algorithm. This paper firstly discusses the basic concepts of PSO algorithm and DS, and then designs the DS.

1. Introduction

With the rapid development of artificial intelligence technology, many tasks or needs in the fields of financial management, data mining, and biomedicine can be attributed to optimization problems, and evolutionary optimization algorithms are used to solve them. Evolutionary optimization technology is a kind of iterative search algorithm that imitates the evolutionary behavior and mechanism of biology. The population evolves in a more suitable direction, and through continuous iterative evolution, the optimal solution or satisfactory solution of the target problem is finally obtained. Considering the complexity of practical optimization problems, this technology has attracted more and more attention and research by scholars at home and abroad [1-2].

At present, many scholars have carried out in-depth research on PSO algorithm and distributed system(DS), and have achieved good results. For example,Vijayakumar T and others proposed the use of multi-core CPU clusters to divide the model to be trained into multiple parts, and then put them into computing nodes for calculation, which can effectively overcome the problem that model training requires a large amount of computing resources. Split, so the time required for training will increase [3]. Combining genetic algorithm, particle swarm algorithm and ant colony algorithm based on compensation, Maghayreh E A et al. proposed a two-stage hybrid swarm intelligent solution algorithm. The algorithm divides the whole solution process into two stages. In the first stage, the genetic algorithm is used. and the randomness, rapidity and completeness of PSO to obtain a series of sub-optimal solutions (preliminary search), and use them to initialize the pheromone of the ant colony algorithm, and in the second stage, use high-precision solutions for detailed search [4]. Although the PSO problem is widely used in various practical optimization problems, there are relatively few studies applying it to the DS optimization problem. The research is of great significance [5-6].

This paper studies DSs based on particle optimization algorithms. The structure of this paper can be divided into three parts; the first part is the introduction of concepts related to DSs and particle optimization algorithms, and the second part is the design of DSs. The main part is the design of information processing and distributed lock, and finally the analysis of system implementation. In this part, this paper analyzes the information processing update strategy, and tests the system performance. Through the test of the system performance, it is found that The system performs well.

2. Related Concepts

2.1. Distributed System

The development of DS services has expanded from a single architecture to a cluster architecture model, and then to today's distributed cluster architecture, which has undergone tremendous changes [7-8]. Different modules of a single system are used at different frequencies, and different modules can be deployed with different amounts of servers to better adapt to different business scenarios [9]. Therefore, today's distributed architecture has been developed. A service is divided into multiple microservices that can be called each other, and multiple nodes are deployed between an area, a city or even multiple cities to form a complete set of distributed node models[10]. Distributed nodes can not only greatly improve the pressure load of the system, but also ensure the disaster tolerance between nodes. Even if a module has a problem, it will not affect the overall use [11].

2.2. PSO Algorithm

PSO is a recognized swarm intelligence optimization algorithm, which uses the information exchange ability of the population to search for the optimal solution of the problem. Particles are defined in the search space by their position and velocity [12-13]. The optimal P of the node itself in the DS represents the optimal information searched by the node i after t iterations, and P is defined as the following formula (1):

$$P = \{(X_i(1), f(X_i(2),..., f(X_i(t)\} \tag{1}$$

3. System Design

3.1. Scheduling Algorithm

Because the quantum particle swarm algorithm does not expand the search range to infinity, but only searches for particles that satisfy the aggregation property in the entire solution space, so by using this algorithm to solve the resource scheduling problem of edge nodes, in order to obtain the local optimal solution [14-15]. The scheduling center is responsible for collecting the information load status $\varnothing_f$ of the relevant target nodes and the amount of offloading tasks. According to the model, the computing resource g_{cf} of the offloading node can be determined as:

$$g_{cf} = \frac{b_{cf}\sum_{Ij} sc}{\varnothing_f + b_{cf}\sum_{Ij} sc} g_{\max} \tag{2}$$

Define the node group M, the position vector corresponding to the node is:

$$\lambda_m = \{\lambda_{j1}, \lambda_{j2}, \ldots, \lambda_{cf}\} \tag{3}$$

After n iterations, the best position of the M node is $P_m(n)$, and the best position of the current node is D(n), then in the next iteration:

$$P = \varnothing P_m(n) + (1-\varnothing)D(n) \tag{4}$$

3.2. Distributed Lock Design

In a DS, preemptive task execution is performed for multiple nodes. In order to ensure the principle of single execution of tasks, a lock mechanism needs to be used. This paper considers that preemptive task execution is required when the computing strategies are equal, so it is necessary to ensure the principle of unique execution of distributed tasks.

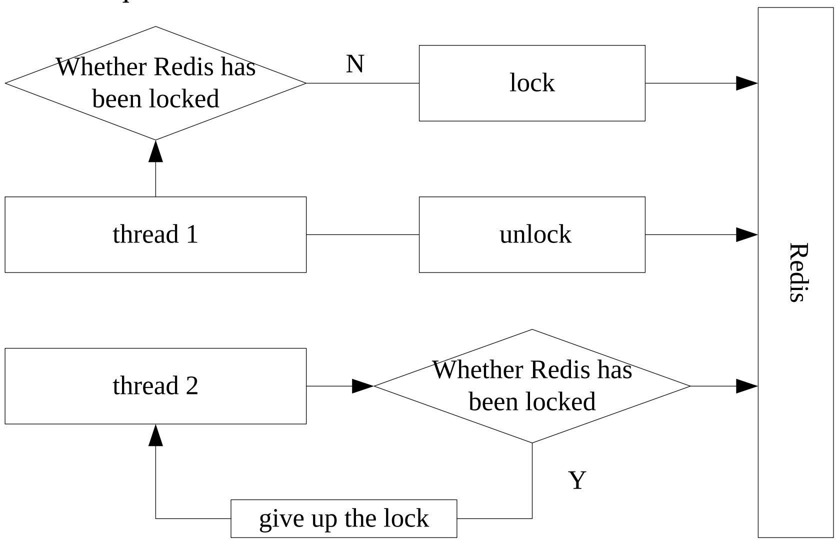

Figure 1. Distributed lock model diagram

As shown in Figure 1, this article uses distributed locks implemented by Redis. The first is to determine whether there is a value, if not, set the value, then set the key time by calling expire, and finally perform the unlock release lock operation in finally. The key time is set to prevent deadlock. When the second thread performs the locking operation again, if it finds that the lock already exists in redis, it will give up the locking task.

3.3. Design of Distributed Node Information Processing

Both the central node and the edge node have this module, which is the central module of the system and involves the processing of information and the training of the model in the process.

(1) Edge node information processing

The edge node in the DS is responsible for sensing information. After sensing the information, the edge node encodes the sensing information. The data format is sensing information + node IP, and shares it with the central node connected to itself [16-17]. After the edge node directly shares the perception information to the central node, the central node will manage the information in a unified manner [18]. After receiving the multicast information from the central node, the edge node firstly identifies the information to identify whether it is the information shared by the central node., the specific process is shown in Figure 2.

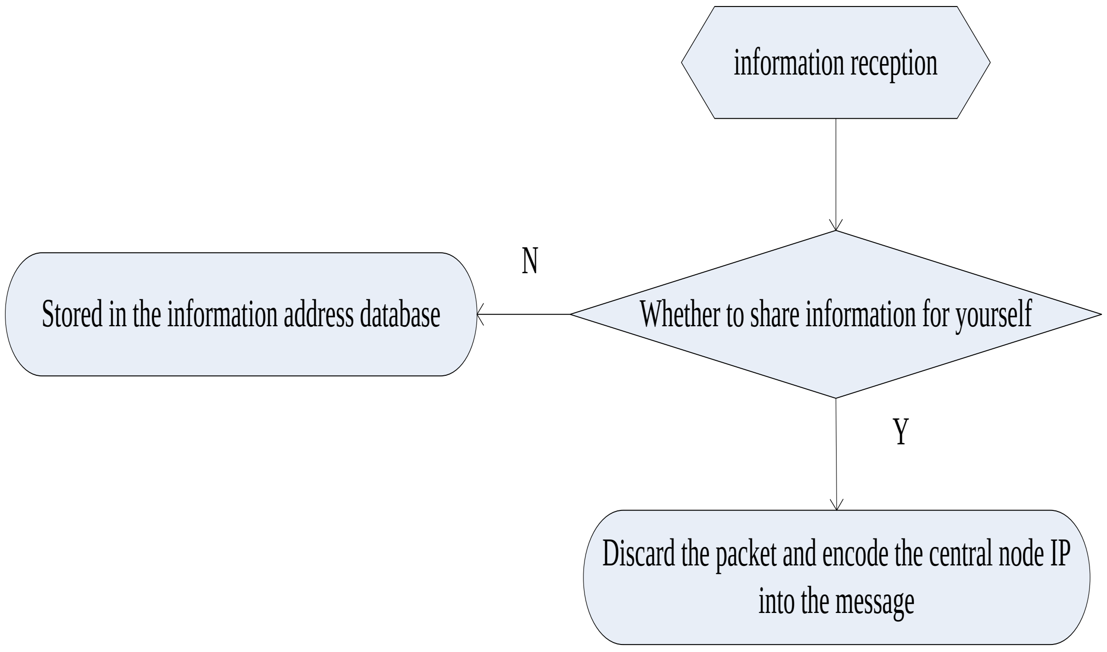
information reception
Whether to share information for yourself
N
Stored in the information address database
Y
Discard the packet and encode the central node IP into the message

Figure 2. Flow chart of edge node information processing

Finally, after passing the model training, the central node of the DS will select an edge node to backup and cache the information. The central node will completely share this information with the edge node, and the edge node will store the information in the database. In addition to the central node, other nodes do not know that the node has this piece of information, so a locally confidential information backup and cache process is achieved.

(2) Information processing of the central node

The central node will first receive the perception information shared by the edge nodes, and after saving the information in the Redis database, the central node will receive the encoded information shared by the edge nodes, and the central node will further encode the information. Due to the encoding of the edge nodes, the central node will continue to operate on the information encoded by the edge nodes, that is, add its own IP.

Then, the central node needs to multicast the information, so that the external nodes know what information is stored on those nodes, which is convenient for system information optimization later. The multicast method mainly scans the communication channels directly connected to itself, and then sends all channels Save it, traverse these channels during multicast, and send the constructed data packet through this channel, as shown in Figure 3.

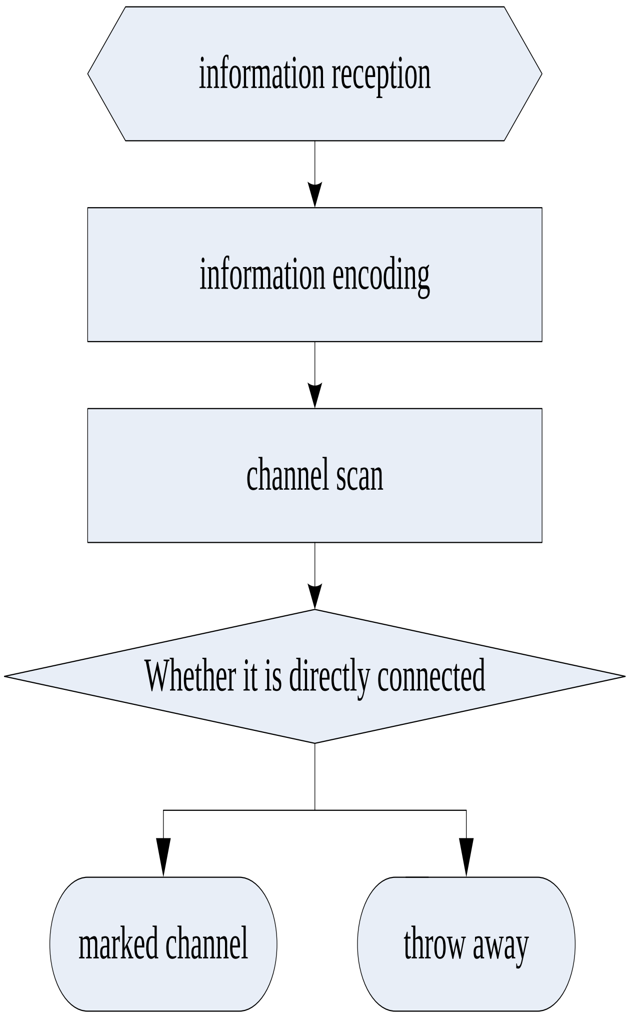

Figure 3. Multicast processing flow chart

4. System Implementation

4.1. Analysis of Different Node Update Strategies

This section will verify the effectiveness of DS information processing by comparing different node information optimization updates. The specific experimental arrangement is as follows: the central node and edge node information processing systems are selected for testing, and the standard PSO algorithm is used to solve the information optimization problem. The basic crossover strategy is: for the current node, randomly select the information crossover area in another node, insert the crossover information area into the current node, delete the information area between the current node and the crossover area; the basic mutation strategy is: for the current node, randomly select a certain information area as the area to be mutated, randomly select the mutation position, insert the node to be mutated into the position, and the rest of the information nodes remain unchanged.Figure 4 is an evolutionary graph of different node update strategies.

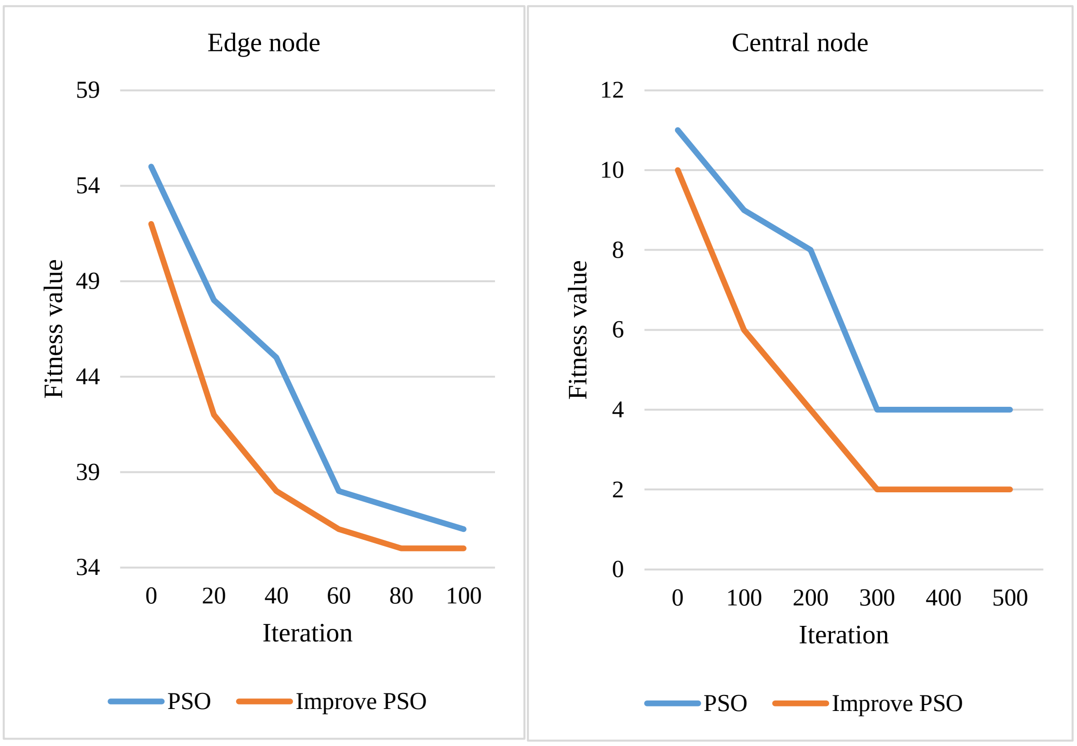

Figure 4. Evolutionary graph of different node update strategies

Aiming at the two test problems of central node and edge node, each node update strategy is run 30 times respectively, among which PSO is the evolution curve designed for basic node information update, and improved PSO is the evolution curve designed for improving node information processing in this paper. It can be seen that: this paper The proposed information processing design has better convergence speed than the basic node update strategy in the early stage of the algorithm, and the algorithm has better convergence performance. In the later stage of the evolution process, compared with the simple mutation strategy, the mutation probability is adjusted according to the evolutionary algebra and the fitness value, and the operation of selecting the mutation position by clustering increases the ability to jump out of the local optimum. It can be seen from the evolution curve that the final result of the information processing design strategy proposed in this paper is obvious.

4.2. System Performance Test

Create 30 information processing timing tasks, which are selected from edge collaboration strategy, polling strategy, consistent HASH strategy, random strategy, and adjacent node strategy. The execution interval of the scheduled task is set within 5 minutes and varies from one to another. Start the task to observe the execution status. Carry out the information processing task test according to the steps in Table 1. The DS is tested through Table 1, and it is found that the performance of the system is well tested.

Table 1. Performance test table

Test Task	ClassifyByStockCode
Test Environment	Three different registered nodes, one dispatch center
Test Steps	Start 30 scheduled tasks and run for half a day
Operation Hours	The running time interval is set within 5 minutes, and the running time is more than 8 hours
Expected Outcome	Each scheduled task basically runs normally

5. Conclusion

The PSO algorithm can find the optimal solution through iteration. This paper studies and analyzes the DS based on the PSO algorithm, and expands the scope of information search through the scheduling of the PSO algorithm. In the system design part, this paper mainly focuses on the design of distributed locks and the design of system information processing. In the system implementation part, it mainly analyzes different node update strategies and tests the performance of DSs. Through the analysis of different node update strategies, it is found that the information processing design strategy designed in this paper can jump out of the local optimum and the information processing effect is remarkable. The system passed the performance test. Although this paper designs a DS based on the PSO algorithm, there are still shortcomings, and the DS needs to be improved. Research on DSs based on PSO algorithm is a worthy direction.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Nabavi S R, Eraghi N O, Torkestani J A. Wireless Sensor Networks Routing Using Clustering Based on Multi-Objective Particle Swarm Optimization Algorithm. Journal of Intelligent Procedures in Electrical Technology (JIPET), 2021, 12(47:49-67.

[2] Abolhoseini S, Mesgari S M, Mohamadi R. Modified particle swarm optimization algorithm to solve location problems on urban transportation networks (Case study: Locating traffic police kiosks) (in Persian). Journal of Geospatial Information Technology, 2021, 8(3):1-16. https://doi.org/10.52547/jgit.8.3.1

[3] Vijayakumar T, Vinothkanna R. Efficient Energy Load Distribution Model using Modified

Particle Swarm Optimization Algorithm. Journal of Artificial Intelligence and Capsule Networks, 2021, 2(4):226-231. https://doi.org/10.36548/jaicn.2020.4.005

[4] Maghayreh E A, Dhahiri H, Albogamy F, et al. Particle Swarm Optimization Algorithm for Detecting Distributed Predicates. IEEE Access, 2021, 9:105286-105296.

[5] Alimardani M, Almasi M. Investigating the application of particle swarm optimization algorithm in the neural network to increase the accuracy of breast cancer prediction. International Journal of Computer Trends and Technology, 2020, 68(4):65-72.

[6] Mauliddina A N, Saifuddin F A, Nagari A L, et al. Implementation of discrete particle swarm optimization algorithm in the capacitated vehicle routing problem. Jurnal Sistem dan Manajemen Industri, 2020, 4(2):117-128. https://doi.org/10.30656/jsmi.v4i2.2607

[7] Kumari R, Gupta N, Kumar N. Cumulative Histogram based Dynamic Particle Swarm Optimization Algorithm for Image Segmentation. Indian Journal of Computer Science and Engineering, 2020, 11(5):557-567.

[8] V S. Optimized Edge Detection Method Using Particle Swarm Optimization Algorithm: An Analysis For Image Processing Applications. International Journal Of Advanced Research In Engineering & Technology, 2020, 11(6):1210-1222.

[9] Hameed F A, Hasan H R, Ahmed A A, et al. Using the Cuckoo Search for Generating New Particles in Particle Swarm Optimization Algorithm. Journal of Computer Science, 2020, 16(4):430-438. https://doi.org/10.3844/jcssp.2020.430.438

[10] Ghathwan K I, Mohammed A J, Yusof Y. Optimal Robot Path Planning using Enhanced Particle Swarm Optimization algorithm. Iraqi Journal of Science, 2020, 61(1):178-184. https://doi.org/10.24996/ijs.2020.61.1.20

[11] Elsayed E, Salem D, Aly M. A Fast Quantum Particle Swarm Optimization Algorithm for Image Denoising Problem. International Journal of Intelligent Engineering and Systems, 2020, 13(1):98-112. https://doi.org/10.22266/ijies2020.0229.10

[12] Mato J, Duster A, Guidez E, et al. Adaptive-Partitioning Multilayer Dynamics Simulations: 1. On-the-Fly Switch between Two Quantum Levels of Theory.. Journal of chemical theory and computation, 2021, 17(9):5456-5465. https://doi.org/10.1021/acs.jctc.1c00556

[13] Das S, Karfa C, Biswas S. Formal Modeling of Network-on-Chip Using CFSM and its Application in Detecting Deadlock. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020, 28(99):1016-1029.

[14] Fay F X, Robles E, Marcos M, et al. Sea trial results of a predictive algorithm at the Mutriku Wave power plant and controllers assessment based on a detailed plant model. Renewable energy, 2020, 146(2):1725-1745.

[15] Sakir R, Bhardwaj S, Kim D S. Enhanced faulty node detection with interval weighting factor for distributed systems. Journal of Communications and Networks, 2021, 23(1):34-42. https://doi.org/10.23919/JCN.2021.000002

[16] Saraswat B K, Suryavanshi R, Yadav D. Formal Specification & Verification of Checkpoint Algorithm for Distributed Systems using Event - B. International Journal of Engineering Trends and Technology, 2021, 69(4):1-9.

[17] Champagnat N, Schott R, Villemonais D. Analysis of distributed systems via quasi-stationary distributions. Stochastic Analysis and Applications, 2021(1729):1-20.

[18] Sarkar K, Chakraborty S, Bonnerjee D, et al. Distributed Computing with Engineered Bacteria and Its Application in Solving Chemically Generated 2 × 2 Maze Problems.. ACS synthetic biology, 2021, 10(10):2456-2464. https://doi.org/10.1021/acssynbio.1c00279