# An Enhanced Bio-Inspired Optimization Approach for Solving Path Selection Challenges in Network System

Liu Qiang[1]

1. Computer Basic Teaching Department, College of Computer Science and Artificial Intelligence, Hunan University of Technology, China

Correspondence: Liu Qiang, Computer Basic Teaching Department, College of Computer Science and Artificial Intelligence, Hunan University of Technology, No. 88, Taishan West Road, Zhuzhou City, Hunan Province, China

**Abstract**

The shortest path problem stands as a critical issue in transportation network analysis, serving as the foundation for numerous optimization tasks such as resource allocation, route design, and traffic management. This paper conducts an in-depth analysis of the basic Artificial Fish Swarm Algorithm (AFSA) and identifies its limitations in terms of accuracy and processing time when solving the shortest path problem between two points in a traffic network. To address these drawbacks, an improved AFSA is proposed, which optimizes the initialization of artificial fish populations and modifies their behaviors. Simulation experiments demonstrate that the improved algorithm can find the shortest path between any two points in a traffic network more accurately and efficiently compared to the original AFSA, verifying its feasibility and superiority in practical applications.

**Keywords:** transportation network analysis, shortest path, Artificial Fish Swarm Algorithm

## 1. Introduction

In the context of rapid urbanization and the continuous expansion of transportation networks, the efficient solution to the shortest path problem has become increasingly crucial for the operation of Intelligent Transportation Systems (ITS). The shortest path problem, which involves finding the path with the minimum total weight (such as distance, time, or cost) between two specific nodes in a network, is not only a fundamental research topic in graph theory but also a key technology in various real-world applications. For instance, in navigation systems, it directly affects the accuracy and efficiency of route guidance; in logistics and distribution, it determines the rationality of delivery routes and the reduction of operational costs; and in urban traffic management, it plays a vital role in alleviating traffic congestion and optimizing traffic flow.

Over the past few decades, researchers have made extensive efforts to develop algorithms for solving the shortest path problem, resulting in a wide range of solutions. Traditional algorithms, such as Dijkstra's algorithm and the A* algorithm, have laid the theoretical foundation for this field. Dijkstra's algorithm, proposed by Edsger W. Dijkstra in 1956, is a classic method for finding the shortest path from a single source node to all other nodes in a graph with non-negative edge weights. It works by iteratively selecting the node with the smallest tentative distance and updating the distances to its neighbors. However, Dijkstra's algorithm has limitations in handling large-scale networks due to its high time complexity ($O(n^2)$ for adjacency matrices and $O(m + n \log n)$ for adjacency lists, where n is the number of nodes and m is the number of edges), which makes it inefficient in real-time applications with dynamic traffic conditions.

The A* algorithm, an extension of Dijkstra's algorithm, introduces a heuristic function to guide the search towards the target node, reducing the number of nodes that need to be explored. While the A* algorithm improves efficiency in many cases, its performance heavily depends on the quality of the heuristic function. A poorly designed heuristic function can lead to suboptimal performance or even failure to find the shortest path.

In recent years, with the development of swarm intelligence and evolutionary computation, a new class of heuristic intelligent algorithms has emerged, providing alternative approaches to solving complex optimization problems. These algorithms, inspired by natural phenomena such as the foraging behavior of ants, the flocking of birds, and the evolution of species, exhibit strong global search capabilities and adaptability. Examples include genetic algorithms, simulated annealing algorithms, tabu search algorithms, and ant colony algorithms. Genetic algorithms, based on the principles of natural selection and genetics, use operations such as mutation, crossover, and selection to evolve solutions. However,

they often suffer from slow convergence and a tendency to fall into local optima. Simulated annealing algorithms, inspired by the annealing process in metallurgy, gradually reduce the temperature to escape local optima, but their performance is sensitive to cooling schedules. Tabu search algorithms use a memory mechanism to avoid revisiting previously explored solutions, but they may struggle with large solution spaces. Ant colony algorithms, which simulate the foraging behavior of ants laying and following pheromones, have shown promising results in path optimization problems, but they can be slow in convergence and require careful parameter tuning.

The Artificial Fish Swarm Algorithm (AFSA), a relatively new swarm intelligence optimization algorithm, has attracted attention due to its simplicity, flexibility, and fast convergence speed. Proposed by Li Xiaolei et al., AFSA simulates the collective behavior of fish schools, including foraging, swarming, and following behaviors, to achieve global optimization through local interactions among individuals. However, like other intelligent algorithms, the basic AFSA has limitations in solving the shortest path problem, such as low accuracy and long processing time. To overcome these shortcomings, this paper proposes an improved AFSA by optimizing the initialization of artificial fish populations and modifying their behaviors. The simulation results demonstrate that the improved algorithm can find the shortest path more accurately and efficiently in traffic networks.

## 2. Description of the Shortest Path Problem in Transportation Networks

Transportation networks, such as urban road networks, consist of physical components such as road segments and intersections, as well as logical attributes such as travel time, distance, and road conditions. To mathematically model these networks for shortest path analysis, we can abstract them into weighted directed graphs. In graph theory, a weighted directed graph G is represented as a binary tuple G = (V, E), where:

V is a set of nodes representing intersections or key points in the transportation network. For example, in a simple weighted network (as typically illustrated in graph theory for shortest path analysis), V could be defined as {1, 2, ..., 9}, where each number corresponds to a distinct intersection.

E is a set of directed edges (arcs) representing road segments between nodes. Each edge <i, j> ∈ E indicates a direct road from node i to node j.

Each edge <i, j> is associated with a non-negative weigh $w_{ij}$, which represents the cost of traveling from node i to node j. The weight can be defined based on various attributes, such as travel time (in minutes), distance (in kilometers), or a composite index considering road conditions, traffic volume, and other factors. For instance, in a typical small-scale weighted network (often used for algorithm illustration in graph theory), the weight of edge <1, 2> could be set as $w_{12} = 1$ indicating that the cost from node 1 to node 2 is 1 unit (the specific unit depends on the attribute being measured, such as 1 minute for travel time or 1 kilometer for distance).

The shortest path problem in a transportation network can be defined as follows: Given a weighted directed graph G=(V,E), a starting node S∈V, and a target node T∈V, find a path from S to T such that the sum of the weights of the edges along the path is minimized. For example, in a typical small-scale weighted directed graph (often used for algorithm illustration), suppose there exists a path from node 1 to node 9 consisting of edges

{<1,2>,<2,3>,<3,8>,<8,9>}, where the weights of these edges are 1, 1, 1, and 1 respectively. Then the total weight of this path is 4, and if no other path from node 1 to node 9 has a smaller total weight, this path is considered the shortest path.

It is important to note that the shortest path problem can be extended to various scenarios, such as undirected graphs (where edges have no direction, i.e., $w_{ij} = w_{ji}$, time-dependent graphs (where weights change over time, e.g., due to traffic congestion), and multi-objective optimization (where multiple criteria, such as time and distance, need to be considered simultaneously). In this paper, we focus on static weighted undirected graphs, where the weights are constant and edges are bidirectional, which is a common abstraction of urban road networks in many practical applications.

To better illustrate the problem, consider a more complex example as shown in Figure 1, which depicts an undirected weighted network with 30 nodes. Each edge is labeled with a weight representing the travel time in minutes. The goal is to find the shortest path from node 1 to node 30, i.e., the path with the minimum total travel time. This example mimics a real-world urban traffic network, where nodes represent intersections and edges represent road segments with varying travel times due to factors such as road length, speed limits, and traffic flow.

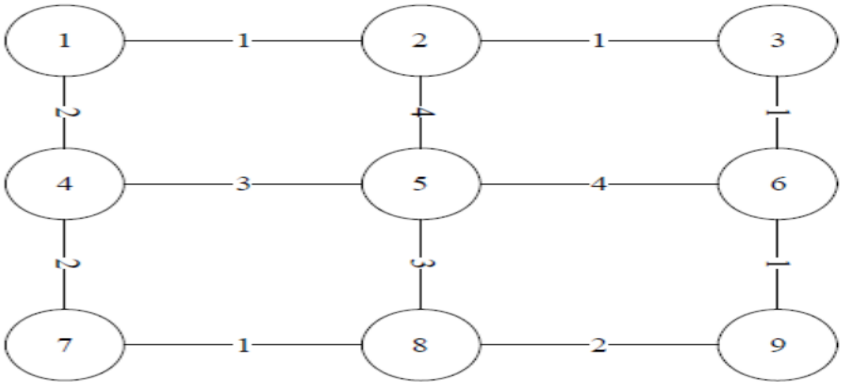


**Figure 1. Network with weights**

## 3. Implementation of the Improved Artificial Fish Swarm Algorithm for the Shortest Path Problem

### *3.1 The Basic Artificial Fish Swarm Algorithm*

#### 3.1.1 Algorithm Principles

The Artificial Fish Swarm Algorithm (AFSA) is a population-based intelligent optimization algorithm that simulates the social behavior of fish schools. Unlike traditional optimization algorithms that rely on gradient information, AFSA achieves global optimization through the interaction and cooperation of multiple artificial fish individuals, each of which performs local search based on simple behavioral rules. The core idea of AFSA is to construct the low-level behaviors of individual fish (such as foraging, swarming, and following) and leverage the collective intelligence of the fish school to find the global optimal solution.

AFSA has several advantages, including:

a) Simplicity and ease of implementation: The algorithm uses simple behavioral rules and does not require complex mathematical derivations.

b) Strong robustness: It is less sensitive to initial conditions and parameter settings, making it suitable for various optimization problems.

c) Fast convergence: The collective search behavior allows the algorithm to quickly approach the optimal solution.

d) Versatility: It can be applied to both continuous and discrete optimization problems with appropriate modifications.

Due to these advantages, AFSA has been successfully applied in various fields, such as continuous optimization, combinatorial optimization, online identification of time-varying systems, parameter tuning of robust PID controllers, optimization of feedforward neural networks, reactive power optimization in power systems, multi-user detectors, information retrieval, and positioning of multi-stage stations in oil fields.

#### 3.1.2 Model Definitions

To apply AFSA to the shortest path problem, we need to define the state of artificial fish, distance metrics, neighborhood concepts, and other key elements:

a) State of artificial fish: The state of an artificial fish individual $x_i$ is represented as a vector $X_i$= ($X_{i1}$, $X_{i2}$, ...,$X_{in}$), where each $x_{ik}$ is a node in the transportation network. The vector represents a path from the starting node to the target node, with each element indicating a node along the path.

b) Food concentration: The food concentration at the state $x_i$, denoted as y = f($X_i$), is defined as the reciprocal of the total weight of the path represented by $x_i$. For the shortest path problem (a minimization problem), a higher food concentration indicates a shorter path. Thus, the goal is to find the artificial fish with the highest food concentration.

c) Distance between artificial fish: The distance $d_{ij}$ between two artificial fish $x_i$ and $x_j$ is defined to measure the similarity between their paths. In this paper, we use a modified Hamming distance: $dij = |X_i - X_j| + |X_j - X_i|$ This distance metric considers the symmetric difference

between the two paths, i.e., the number of nodes that are in one path but not the other. A smaller distance indicates more similar paths.

d) Visual range: The visual range (Visual) is the maximum distance within which an artificial fish can perceive other individuals. Artificial fish only interact with others within their visual range.

e) Step size: The step size (step) determines the maximum distance an artificial fish can move in one iteration. It is usually a random value within a predefined range (rand() × step).

f) Crowding factor (δ): The crowding factor controls the balance between exploration and exploitation. It is used to determine whether the fish school is too crowded to move towards the center.

g) Center position of the fish school: The center position $X_c$ of a group of artificial fish is defined as the intersection of all paths, representing the common nodes in most paths:

i. $$X_c(k) = \bigcup_{i=1}^{n} \bigcup_{\substack{j=1 \\ j \neq i}}^{n} (X_i \bigcap X_j)$$

ii. Movement rule: When an artificial fish $x_i$ moves towards another fish $x_j$ by a random step, its new state $X_j$ (k) is calculated as:

iii. $$X_j(k) = X_i(k) + \text{rand}() \cdot \text{step} \cdot \frac{X_j(k) - X_i(k)}{\|X_j(k) - X_i(k)\|} \quad ; k = 1, \cdots, n$$

This formula ensures that the movement is in the direction of $X_j$ with a random step length.

3.1.3 Behavioral Descriptions

The behavior of artificial fish in AFSA is governed by three primary rules: foraging, swarming, and following. These behaviors are designed to simulate the natural behavior of fish schools and guide the search process.

a) Foraging behavior: Foraging behavior simulates the process of fish searching for food. For an artificial fish $X_i$:

i. Randomly select another artificial fish $X_j$ within its visual range.

ii. Compare the food concentrations: if f($X_j$) > f($X_i$:) (indicating a better path), move towards $X_j$ by a random step.

iii. If no better $X_j$ is found after several attempts (TryNumber), move randomly within the visual range.

b) Swarming behavior: Swarming behavior mimics the tendency of fish to gather in groups to avoid predators. For an artificial fish $X_i$:

i. Count the number of neighboring fish ($n_f$) within its visual range and calculate their center position $X_c$.

ii. If the average food concentration at the center ($Y_c$ / $n_f$) is higher than δ × $Y_i$ (indicating sufficient food and low crowding), move towards $X_c$ by a random step.

iii. Otherwise, perform foraging behavior.

c) Following behavior: Following behavior simulates fish following the leader with the highest food concentration. For an artificial fish $X_i$:

i. Identify the neighboring fish X<sub>j</sub> with the highest food concentration within its visual range.

ii. If the food concentration of $X_j$ ($Y_j$) is higher than δ × $Y_i$ (indicating a better path and low crowding), move towards $X_j$ by a random step.

iii. Otherwise, perform foraging behavior.

Behavior selection: Artificial fish select their next behavior based on the current environment. For minimization problems, the selection is often based on a trial-and-error approach: simulate each possible behavior (foraging, swarming, following) and choose the one that yields the best result. If no better behavior is found, foraging is performed by default.

*3.2 The Improved Artificial Fish Swarm Algorithm*

The basic AFSA has limitations in solving the shortest path problem, such as uncertain initial population distribution and slow convergence. To address these issues, we propose the following improvements:

3.2.1 Initialization Optimization

The initial distribution of artificial fish significantly affects the global convergence of the algorithm. In the basic AFSA, artificial fish are randomly initialized, which may result in uneven coverage of the solution space. If the initial population does not cover the global optimal solution and the behavioral operators fail to expand the search space sufficiently, the algorithm may prematurely converge to a local optimum.

To ensure uniform coverage of the solution space, we propose a grid-based initialization method. The solution space (all possible paths from the start to the target node) is divided into a grid, and artificial fish are placed at grid points to ensure even distribution. The mathematical formulation is:

$$X_i(k) = D_{\text{Min}}(k) + \frac{D_{\text{Max}}(k) - D_{\text{Min}}(k)}{n-1} \times i$$

where:

a) $X_i(K)$ is the k-th component of the i-th artificial fish.

b) $D_{Min}(k)$ and $D_{Max}(k)$ are the minimum and maximum values of the k-th component in the solution space.

c) n is the number of artificial fish.

This method ensures that artificial fish are evenly distributed across the solution space, enhancing the diversity of the initial population and reducing the risk of premature convergence.

3.2.2 Behavioral Improvements

a) Foraging Behavior

i. **Boundary handling**: When an artificial fish moves beyond the solution space (e.g., a path contains invalid nodes), we use a reflection mechanism to bring it back within bounds:

ii.

$$X_j(k) = \begin{cases} 2D_{\text{Max}}(k) - X_j(k) & \text{if } X_j(k) > D_{\text{Max}}(k) \\ 2D_{\text{Min}}(k) - X_j(k) & \text{if } X_j(k) < D_{\text{Min}}(k) \end{cases}$$

This ensures that all paths remain valid (i.e., consist of existing edges in the graph).

iii. Global search trigger: If foraging fails to find a better solution after a predefined number of attempts (TryNumber), a global search is performed to escape local optima:

$$X_{\text{next}} = X_i + \text{rand}() \cdot \text{global}$$

iv. where "global" is a parameter defining the range of the global search. If the global search yields a worse solution than the current state, the artificial fish remains stationary. This mechanism helps preserve good solutions while enabling exploration of new areas.

b) Swarming and Following Behavior Optimization

In the basic AFSA, swarming and following behaviors involve random movement towards the center or the leader, which may slow down convergence. We introduce a adaptive step size (S) to balance exploration and exploitation:

$$\text{exploitation:} S = \begin{cases} \text{rand}() \cdot \text{step} & \text{if } F_{\text{step}} > F_{X_c} \\ X_c = \bigcup_{i=1}^{n} \bigcup_{\substack{j=1 \\ j \neq i}}^{n} (X_i \bigcap X_j) & \text{if } F_{\text{step}} < F_{X_c} \end{cases}$$

where:

$F_{step}$ is the food concentration after a random step.

$F_{X_c}$ is the food concentration at the center position.

If the food concentration after a random step is higher than that at the center, a random step is taken to explore new areas. Otherwise, the artificial fish moves directly to the center to exploit promising regions. This adaptive step size accelerates convergence by reducing unnecessary random movements.

The bulletin board mechanism remains unchanged: it records the best solution found, and artificial fish update it whenever a better solution is discovered.

*3.3 Implementation of the Improved AFSA for the Shortest Path Problem*

3.3.1 Algorithm Idea

The improved AFSA solves the shortest path problem by treating each path as an artificial fish in a "water area," where the food concentration is the reciprocal of the path's total weight (shorter paths have higher concentrations). The algorithm initializes a uniformly distributed population of artificial fish, then iteratively improves the paths through modified foraging, swarming, and following behaviors, ultimately finding the path with the highest food concentration (shortest total weight).

3.3.2 Algorithm Steps

a) Parameter initialization: Set the number of artificial fish (FishNumber), crowding factor (δ), maximum number of attempts (TryNumber), maximum iterations (Max_gen), step size (step), stagnation parameter (tag, i.e., stop if no improvement for tag iterations), and other parameters.

b) Population initialization: Define the number of nodes (n) as the dimension of artificial fish. The first element is the start node, and the remaining elements are randomly generated non-repeating nodes. Ensure the target node is included, and set all nodes after the target node to 0. Remove invalid paths (those containing non-existent edges) and calculate the initial food concentration for each artificial fish. The initial best solution is stored in the bulletin board.

c) Behavior execution: Each artificial fish performs improved foraging, swarming, or following behaviors. After each behavior, update the bulletin board if a better solution is found.

d) Termination check: If the maximum iterations (Max_gen) are reached or no improvement is observed for tag iterations, stop the algorithm.

**4. Algorithm Simulation and Result Analysis**

*4.1 Simulation Setup*

To evaluate the performance of the improved AFSA, we conducted simulations on the undirected weighted network shown in Figure 2, which consists of 30 nodes with edge weights representing travel time in minutes. The goal is to find the shortest path from node 1 to node 30.

The network is treated as a symmetric directed graph (edge weights are the same in both directions), mimicking real urban traffic networks. The simulation was conducted using MATLAB 7.0, and the parameters were determined through extensive experiments to ensure optimal performance:

a) Number of artificial fish (FishNumber) = 25

b) Crowding factor (δ) = 0.8

c) Maximum attempts (TryNumber) = 20

d) Maximum iterations (Max_gen) = 100

e) Step size (step) = Vision / 4

f) Stagnation parameter (tag) = 5

g) Vision = average distance between artificial fish

*4.2 Simulation Results*

The improved AFSA found the shortest path from node 1 to node 30 as:
1 - 7 - 13 - 19 - 25 - 26 - 27 - 28 - 29 - 30

The total travel time (sum of weights) is 11 minutes.

To verify the superiority of the improved algorithm, we compared it with other algorithms, including Dijkstra's algorithm, the ant colony algorithm, and the basic AFSA, under the same accuracy requirements. The results are shown in Table 1.

| Algorithm | Shortest Path Weight (min) | Convergence Speed (ms) |
|---|---|---|
| Dijkstra's algorithm | 11 | 2160 |
| Ant colony algorithm | 11 | 760 |
| Basic AFSA | 11 | 640 |
| Improved AFSA | 11 | 380 |

Table 1: Comparison of simulation results

The results indicate that all algorithms found the shortest path with a total weight of 11 minutes, but the improved AFSA achieved the fastest convergence (380 ms), outperforming Dijkstra's algorithm (2160 ms), the ant colony algorithm (760 ms), and the basic AFSA (640 ms).

*4.3 Result Analysis*

a) Accuracy: All algorithms correctly identified the shortest path, confirming the validity of the improved AFSA.

b) Convergence speed: The improved AFSA outperformed other algorithms due to:
   i. Uniform initial distribution ensuring broader exploration.
   ii. Adaptive step size reducing unnecessary movements.
   iii. Global search mechanism escaping local optima.

c) Robustness: The improved AFSA showed stable performance across multiple runs, with small variations in convergence time, indicating strong robustness.

**5. Conclusion**

This paper addresses the shortest path problem in transportation networks by proposing an improved Artificial Fish Swarm Algorithm (AFSA). The improvements include grid-based initialization for uniform population distribution and adaptive behavioral rules to accelerate convergence. Simulation results demonstrate that the improved AFSA outperforms traditional algorithms (Dijkstra's), the basic AFSA, and the ant colony algorithm in terms of convergence speed while maintaining accuracy.

The algorithm's advantages, such as adaptability to search spaces, parallel search capability, and fast tracking of changing optima, make it suitable for dynamic transportation networks. Future work will focus on extending the algorithm to time-dependent networks and integrating real-time traffic data for more practical applications.

## References

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. MIT Press.

Di Lecce, V., & Amato, A. (2011). Route planning and user interface for an advanced intelligent transport system. *Intelligent Transport Systems*, *5*(3), 149–158. https://doi.org/10.1504/ITS.2011.041243

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. https://doi.org/10.1007/BF01386390

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

He, S., Belacel, N., Hamam, H., & Bouslimani, Y. (2009, April). Fuzzy clustering with improved artificial fish swarm algorithm. In *Proceedings of the International Conference on Computational Sciences and Optimization* (Vol. 2, pp. 317–321). IEEE. https://doi.org/10.1109/CSO.2009.141

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.

Kang, H., Lee, B., & Kim, K. (2008, November). Path planning algorithm using the particle swarm optimization and the improved Dijkstra algorithm. In *Proceedings of the International Conference on Computational Intelligence and Industrial Application* (Vol. 2, pp. 1002–1004). IEEE.

Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks* (Vol. 4, pp. 1942–1948). IEEE. https://doi.org/10.1109/ICNN.1995.488968

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Matsuda, Y., Nakamura, M., Kang, H., & Miyagi, H. (2004, October). An optimal routing problem for sightseeing with fuzzy time-varying weights. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* (pp. 3665–3669). IEEE. https://doi.org/10.1109/ICSMC.2004.1400846