# Improved Wolf Pack Algorithm for the Traveling Salesman Problem (Postprint)

**Authors:** Huang Haisong, Ren Zhupeng, Wei Jian' an

**Date:** 2018-10-11T00:00:00+00:00

## Abstract

To find a shortest path and overcome issues such as the unsuitability of traditional algorithms for discrete domain solving in path planning and their slow convergence speed, an improved wolf pack algorithm is proposed. By introducing a position-order encoding method in the initialization phase, discrete domain path optimization is investigated; simultaneously, a secondary search is introduced in the iteration process to enhance the algorithm' s solution speed and accuracy, thereby achieving the emergence of an optimal solution before reaching the maximum iteration count. Results demonstrate that the improved wolf pack algorithm exhibits higher solution accuracy, faster convergence speed, and more effectively avoids falling into local optima compared to existing algorithms. It is thus evident that the improved wolf pack algorithm can be well applied to solving optimal path planning problems.

## Full Text

## Preamble

### Improved Wolf Pack Algorithm for Solving the Traveling Salesman Problem

*Huang Haisong, Ren Zhupeng, Wei Jian' an*
(Key Laboratory of Advanced Manufacturing Technology of Ministry of Education, Guizhou University, Guiyang 550025, China)

**Abstract:** To find the shortest path and overcome the limitations of traditional algorithms in path planning—namely their unsuitability for discrete domain solutions and slow convergence—this paper proposes an improved wolf pack algorithm. By introducing a position-order encoding method during the initialization phase, the algorithm addresses path optimization in discrete domains. Simultaneously, a secondary search mechanism is incorporated into the iterative process to enhance solution speed and precision, enabling the optimal solution to

emerge before reaching the maximum iteration count. Results demonstrate that the improved wolf pack algorithm achieves higher solution accuracy and faster convergence compared to existing algorithms, while more effectively avoiding local optima. The findings indicate that the improved wolf pack algorithm can be successfully applied to optimal path planning problems.

**Keywords:** improved wolf pack algorithm; discrete domain; secondary search; path planning

## 0 Introduction

The Traveling Salesman Problem (TSP) is a classic path optimization problem aimed at finding the shortest possible route that visits each city exactly once. Many real-world problems are abstracted as TSP instances, including robot control, workshop scheduling, UAV trajectory planning, computer networking, and network router deployment. Although TSP is an NP-hard combinatorial optimization problem that is difficult to solve, its study holds significant practical value and theoretical importance.

In theory, small-scale TSP instances can be solved optimally using traditional algorithms, while large-scale problems typically employ heuristic methods. However, heuristic algorithms have limitations in obtaining optimal solutions within finite time. Population-based intelligent algorithms derived from natural biological systems—such as genetic algorithms, particle swarm optimization, ant colony algorithms, artificial bee colony algorithms, firefly swarm optimization, biogeography-based migration algorithms, and imperialist competitive algorithms—provide alternative approaches for solving complex path optimization problems. Emerging swarm intelligence and bionic algorithms have improved the probability of obtaining optimal solutions within limited time and reduced the solution space. Nevertheless, many algorithms suffer from shortcomings such as susceptibility to local optima and slow convergence. Consequently, path planning remains a challenging and valuable research area.

Building upon the wolf pack algorithm, numerous scholars have proposed various improvements. Reference [8] optimized the traditional algorithm by designing a new position update formula for fierce wolves and improving the step size. Reference [9] enhanced optimization performance by introducing crossover, mutation, and selection operations from differential evolution. However, these methods still exhibit slow convergence and a tendency to fall into local optima, particularly failing to address the wolf pack algorithm' s limitations in discrete domain solutions. To achieve discrete domain path optimization, this paper adopts a position-order encoding method. To improve the speed and accuracy of the wolf pack algorithm, random single-point insertion and multi-point insertion operators are introduced during iteration, implementing a secondary search strategy to achieve optimal solutions before reaching the maximum iteration count.

## 1 Overview of the Wolf Pack Algorithm

The wolf pack algorithm comprises three intelligent behaviors: scouting wolf wandering, head wolf summoning, and fierce wolf besieging. The algorithm follows "survival of the fittest" rules for pack updating and "winner takes all" rules for head wolf competition. The procedure is as follows:

a) Wolf pack spatial coordinates are randomly initialized in the solution space, with the head wolf representing the artificial wolf with the optimal objective function value.

b) During prey search, scouting wolves begin wandering. If a scouting wolf detects a scent concentration greater than that found by the head wolf, it assumes dominance and updates the head wolf position. Otherwise, the scouting wolf continues wandering until it either discovers a scent concentration exceeding the head wolf' s or reaches the maximum wandering count, at which point the head wolf proceeds from its current position.

c) After the head wolf issues a summons, fierce wolves move toward the head wolf with maximum step size. If a fierce wolf' s objective value proves superior to the head wolf' s during this process, the head wolf is updated. Otherwise, the fierce wolves continue moving until entering the siege range.

d) Collaborating with scouting wolves, fierce wolves near the head wolf besiege the prey (treating the head wolf position as the prey). If an artificial wolf' s objective value surpasses the head wolf' s after besieging, the head wolf position is updated and hunting continues; otherwise, it remains unchanged.

e) Artificial wolves with inferior objective function values are eliminated, and new artificial wolves are randomly generated in the solution space, continuously updating the wolf pack.

f) Finally, the head wolf' s objective value is evaluated against convergence criteria or maximum iteration count. If neither condition is met, iteration continues until satisfied. Upon meeting requirements, the output objective function value represents the optimal solution, with the head wolf's spatial coordinates indicating the function' s optimal position.

## 2 Improved Wolf Pack Algorithm for Path Planning

### 2.1 Principle of the Improved Wolf Pack Algorithm

The correspondence between the wolf pack algorithm and path planning problems is established as follows:
a) Wolf pack algorithm  Path planning
b) Individual wolf  Feasible path
c) Wolf position change  Path order change
d) Wolf fitness  Path length

Although the wolf pack algorithm can handle simple path planning problems, it cannot directly achieve discrete domain path planning and suffers from slow iteration speed. Therefore, this paper proposes an improved wolf pack algorithm that introduces a position-order encoding method for discrete domain path optimization and employs secondary search to enhance solution speed and precision.

#### 2.2.1 Wolf Position-Order Encoding Process

The wolf pack algorithm can be applied to continuous domain function optimization but is unsuitable for direct application to discrete domain TSP solving. The primary reason is that TSP solutions are typically represented by natural numbers, while search mode mutations occur in the real number domain. To overcome this issue, position-order encoding is introduced.

The $t$-th generation $i$-th individual wolf in the wolf pack algorithm is represented as shown in the encoding process. When sorted in ascending order, the resulting sequence' s order yields the corresponding feasible path encoding. This transformation constitutes the position-order encoding process.

During search mode, the wolf pack defines three parameters:
- CDC: Number of genes changed per individual
- SMP: Memory pool storing mutation states
- SDR: Change range for each gene on an individual

The specific search mode proceeds as:
a) Place the current-state wolf individual from the pack into SMP and copy it;
b) Modify the copied individual in the memory pool after mutation occurs and update to a new position;
c) Calculate the fitness value of the copy, with the wolf' s final updated state being the optimal state among them.

Under this search mode, all wolf individuals gradually converge toward the global optimal solution.

#### 2.2.2 Fitness Determination

Let $p_1, p_2, \ldots, p_n$ represent city numbers forming a feasible tour path. The total path length is calculated as shown in the fitness function. In the wolf pack

algorithm, individuals are evaluated based on path length: larger $d$ indicates poorer individual fitness, where the reciprocal of path length defines individual fitness (fitness $= 1/d$).

### 2.2.3 Secondary Search Strategy

The search mode constitutes the main optimization process of the wolf pack algorithm but suffers from issues such as relatively monotonous mutation strategies and suboptimal convergence. When optimizing sufficiently large populations, limited maximum iteration counts may prevent the emergence of superior solutions after mutation. To better address TSP when paths fail to obtain improved solutions after search mode optimization, a secondary search optimization strategy with different mutation operators but similar to the wolf pack search mode is introduced.

Two approaches for the secondary search optimization strategy are as follows:

**1) Random Single-Point Insertion Operator Description**
After encoding all individual wolves, let $(i, j)$ be randomly selected gene positions with corresponding cities. The distance between two cities is denoted as $d(c_i, c_j)$. To improve the overall path distance, if $\Delta d = d(c_i, c_{j+1}) + d(c_j, c_{i+1}) - d(c_i, c_{i+1}) - d(c_j, c_{j+1}) < 0$, then swap the genes at positions $i$ and $j$. This method arbitrarily selects two positions for gene exchange, minimally affecting "pattern transformation" with low computational complexity. Through this approach, the nearest and path-optimal point can be selected during each iteration. [Figure 2: see original paper]

**2) Random Multi-Point Insertion Operator Description**
After encoding all wolf individuals, let $k$ be the city position number. Gene position $i$ is randomly selected, and city $t$ nearest to city $k$ is located among the subsequent $n - i$ positions with gene position $j$. The segment after position $j$ is then inserted after gene position $i$. This segment insertion operator can significantly reduce computation time and find optimal solutions with minimal iteration count. [Figure 3: see original paper]

The advantages of this insertion method are:
a) It can improve algorithm fitness by selecting the nearest point, providing local optimization;
b) It can enrich feasible solution diversity through random insertion point selection;
c) It can avoid destroying existing globally optimal path segments through whole-segment insertion.

---

## 3 Simulation Experiments and Analysis

Experiments were conducted using several instances from the TSPLIB standard library to verify the accuracy, effectiveness, and feasibility of the discrete

wolf pack algorithm. Results were compared with optimization algorithms from references [14–16].

#### 3.1.1 Dantzig42 Problem

For the Dantzig42 problem, the optimization path obtained by the proposed algorithm is shown in [Figure 4: see original paper], with ten optimization results presented in the first column of Table 1.

#### 3.1.2 Eil51 Problem

For the Eil51 problem, the optimization path obtained by the proposed algorithm is shown in [Figure 5: see original paper], with ten optimization results presented in the second column of .

#### 3.1.3 Berlin52 Problem

For the Berlin52 problem, the optimization path obtained by the proposed algorithm is shown in [Figure 6: see original paper], where black circles indicate locally magnified portions demonstrating no intersections. Ten optimization results are presented in the third column of .

### 3.2 Experimental Analysis

As shown in Figures 4–6, the optimal paths obtained by the proposed IWPA algorithm for the three TSPLIB instances exhibit no path crossings, with lengths generally superior to those from other literature algorithms. demonstrates that the improved wolf pack algorithm outperforms both the unimproved version and algorithms from references. The optimization results for Dantzig42, Eil51, and Berlin52 instances surpass the optimal results provided by the international TSPLIB website. In multiple optimization experiments, the algorithm converges to global optimal solutions. Comparative analysis with previously improved wolf pack algorithms in shows significant improvements across six experimental groups in terms of best solution, worst solution, and iteration count. Regarding average time consumption, the random insertion operator approach yields shorter average times compared to reference [17], demonstrating the superiority of the improved wolf pack algorithm over similar methods.

---

## 4 Conclusion

This paper employs a position-order encoding method to achieve discrete domain path optimization. During the search process, random insertion operators are used to improve solution speed and precision, enabling path optimization before reaching maximum iteration counts and avoiding local optima. Compared with other algorithms and improved wolf pack algorithms, this work represents

significant progress in discrete domain solving, as evidenced by the experiments showing substantial leadership in both solution speed and accuracy. Furthermore, a remaining challenge involves applying this method to practical engineering problems, particularly AGV transportation in workshop settings, which represents the next research direction for our group.

The improved wolf pack algorithm can effectively escape local optima and obtain optimal solutions before reaching iteration limits. Consequently, the discrete wolf pack algorithm can be successfully applied to path optimization problems.

---

## References

[1] Luo W, Lin D, Feng X. An improved ant colony optimization and its application on TSP problem [C]// Proc of IEEE International Conference on Internet of Things. 2017: 136-141.
[2] Liu Jie, Zhao Haifang, Zhou Delian. Mobile robot path planning based on improved quantum behavior particle swarm optimization algorithm [J]. Computer Science, 2017, 44 (Z11): 123-128.
[3] Zheng Jian, Huang Min, Zhang Teng, et al. Improved artificial bee colony algorithm for solving path planning problem of directional signs [J]. Application Research of Computers, 2017, 34 (8): 2355-2359.
[4] Deb S, Fong S, Tian Z, et al. Finding approximate solutions of NP-hard optimization and TSP problems using elephant search algorithm [J]. Journal of Supercomputing, 2016, 72 (10): 1-33.
[5] Fang Qinghua, Ni Liping, Li Yiming, et al. Two-stage multi-objective ant colony algorithm for solving logistics Web service composition problem [J]. China Mechanical Engineering, 2016, 27 (10): 1327-1336.
[6] Yan Xiangbo, Zhu Yunlong, Zhang Dingyi. Dual-population collaborative learning algorithm for solving PFSP [J]. Control and Decision, 2017, 32 (1): 12-20.
[7] Yan Yanqin. Research on AGV path planning algorithm based on swarm intelligence optimization [D]. Changchun: Jilin University, 2017.
[8] Ye Yong, Zhang Huizhen. Wolf group algorithm for multi-distribution center vehicle routing problem [J]. Journal of Computer Applications, 2017, 34 (9): 2590-2593.
[9] Wang Yingxiang, Chen Minyu, Cheng Tingli, et al. Research on improved wolf group algorithm based on differential evolution [J//OL]. Journal of Computer Applications, 2019, 36 (8): 1-10.
[10] Jin Shan, Jin Zhigang. Multi-objective convergence node covering algorithm based on quantum wolf group evolution [J]. Journal of Electronics & Information Technology, 2017, 39 (5): 1178-1184.
[11] Hui Xiaobin, Guo Qing, Wu Pingping. An improved wolf group algorithm [J]. Control and Decision, 2017, 32 (7): 1163-1172.
[12] Yang Shuying, Zhang Wei. Group intelligence and bionic computing: MATLAB technology implementation [M]. Beijing: Publishing House of

Electronics Industry, 2012: 180-189.
[13] Song Shuanjun, Yang Peili, Shi Wenli. Integrated optimization of flexible process and shop scheduling based on artificial bee colony algorithm [J]. Journal of Computer Applications, 2017, 37 (2): 523-529.
[14] You Xiaoming, Liu Sheng, Lyu Jinqiu. An ant colony algorithm for dynamic search strategy and its application in robot path planning [J]. Control and Decision, 2017, 32 (3): 552-556.
[15] Ge Haiming. Application and research of improved genetic algorithm for solving TSP problem [D]. Ganzhou: Jiangxi University of Science and Technology, 2016.
[16] He Qing, Wu Yile, Xu Tongwei. Application of improved genetic simulated annealing algorithm in TSP optimization [J]. Control and Decision, 2018 (2): 219-22.
[17] Wu Husheng, Zhang Fengming, Li Hao, et al. Discrete wolf group algorithm for solving TSP problem [J]. Control and Decision, 2015, 30 (10): 1861-1867.

*Note: Figure translations are in progress. See original paper for figures.*