An Improve Grey Wolf Optimizer Algorithm for Traveling Salesman Problems

Zhinan Xu, Xiaoxia Zhang*

Abstract—**The Traveling Salesman Problem (TSP) seeks the shortest closed tour that visits each city once and returns to the starting city. This problem is NP-hard, so it is not easy to solve using conventional methods. The grey wolf optimization (GWO) algorithm has shown outstanding per- formance in many practical applications. However, it is inclined towards premature convergence. This paper proposes an im- proved GWO (I-GWO) algorithm, which hybridizes GWO with genetic algorithms (GA) for the TSP. The main feature of the I-GWO algorithm is that it can make full use of the advantages of the GWO algorithm and the GA algorithm to make up for their respective shortcomings. Moreover, to make the GWO suitable for solving the TSP, both the 2-opt operator strategy and hamming distance h ave been designed to implement the discrete GWO directly. Additionally, to increase the diversity of solutions by expanding the search space, we present a new population update strategy with crossover and mutation operations in the next iteration. Meanwhile, the integration of the Simulated Annealing (SA) algorithm into the Improved Grey Wolf Optimizer (I-GWO) enhances its local search capabilities. Experimental results show that the I-GWO algorithm competes with established optimal methods for solving the TSP, suggesting its potential for different TSP variants and logistic transport domains.**

Index Terms—**Traveling salesman problem, 2-opt, Grey wolf optimization, Crossover and mutation**

I. Introduction

THE logistics transportation plays an indispensable role in modern society and permeates various aspects of our lives. It is worth noting that the logistics transportation problem can actually be considered as one of the practical applications of the traveling salesman problem. Logistics transportation requires finding the shortest path connecting different delivery points to minimize cost and time. Therefore, logistics and transportation are closely related to the traveling salesman problem, and algorithms and optimization methods for the traveling salesman problem play a key role in logistics planning, helping us to manage logistics networks more intelligently and thus improve our daily lives.

The traveling salesman problem (TSP) [1] is a classical combinatorial optimization problem, and its aim is to find a travel route that visits each city exactly once, returning to the starting city to form a closed tour with the minimum cost. However, with the number of cities increasing, the computational time required to solve the problem will grow exponentially. It means that it is impossible to find the optimal solution in polynomial time. Some classical TSP solutions including integer linear programming [2], dynamic programming [3], branch and cut [4] etc., have been widely used in many fields such as logistics planning, circuit design, bioinformatics and tourism planning. However, TSP belongs to the np-hard problem, and these classical algorithms take a lot of time to search for the optimal solution. Even when the data are too large, they can fall into a local optimum or fail to search for the optimal solution. For this reason, researchers use heuristic and metaheuristic algorithms to solve the TSP.

Metaheuristic algorithm (MHA) is an advanced optimization algorithm for solving complex combinatorial optimization problems with a large search space and multiple local optimal solutions. Metaheuristic algorithms have advantages over traditional algorithms in terms of generalization, global search capability, iterative improvement and adaptability to find better solutions to problems. The algorithm's potent global search capability can discover feasible solutions to intricate problems, and the iterative improvement strategy aids in enhancing the solutions' quality gradually. Furthermore, metaheuristic algorithms are typically not reliant on a specific problem, and they can be effortlessly used with different problems without making significant modifications to the algorithmic structure. These aspects help to extend algorithms' application areas and increase the flexibility and efficiency of solving problems. However, metaheuristic algorithms have some limitations. On np-hard problems with a large search space, metaheuristic algorithms can usually only obtain approximate solutions and are not guaranteed to find a globally optimal solution. This means that meta-heuristic algorithms may not be able to provide the best solution to the problem. Therefore, researchers have subsequently proposed many improved metaheuristic algorithms. For the large data level TSP problem, Skinderowicz [5] proposed the use of ACO variant (FACO) for optimization. Huang et al [6] proposed the use of discrete frog jumping algorithm for solving the TSP problem. Gao [7] proposed a new ACO optimization algorithm for the TSP problem. Wang et al [8] proposed the use of ACO algorithm for TSP problem's parameter optimization. Zhang [9] proposed a genetic algorithm for TSP problem based on jumping genes and heuristic operators (GA-JGHO). Kanna et al [10] introduced the Deer Hunting Linked Earthworm Optimization Algorithm for solving large travelling salesman problems.

The grey wolf optimization (GWO) algorithm [11] is a heuristic algorithm for solving optimization problems based on the behavior of grey wolf packs in nature proposed by Mirjalili in 2014. The algorithm simulates the social structure and behavioral characteristics of grey wolves, particularly the

Manuscript received November 11, 2023; revised April 19, 2024.

Z. N. Xu is a Postgraduate in School of Computer Science and Software Engineering, University of Science and Technology LiaoNing, Anshan, 114051, China. (e-mail: xuzhinan@ustl.edu.cn).

X. X. Zhang is a Professor of School of Computer Science and Software Engineering, University of Science and Technology LiaoNing, Anshan, 114051, China. (Corresponding author, phone:86-0412-5929812, e-mail: aszhangxx@163.com).

collaboration and competition between leaders and followers in a grey wolf population. From the invention of GWO algorithm untill now, it is widely used in various fields such as optimization problems, machine learning, signal processing, image processing, logistics and transportation and economics and other related work. Zamfirache et al. [12] used the GWO algorithm for the training of neural networks. The GWO algorithm obtained better solutions than PD and PSO algorithms. Ahmed et al. [13] proposed the usage of support vector regression (SVR) and GWO for predicting the compressive strength of GGBFS-based polymer concrete. Additionally, Altan [14] proposed a hybrid WSF model that ensures reliable and precise wind speed prediction for wind power generation through the optimization of intrinsic mode function (IMF) estimation output via GWO.

In addition, the researchers have proposed different versions of GWO to solve the application problem in a variety of scenarios. Nadimi-Shahraki et al. [15-16] proposed a GWO algorithm based on gaze cue learning to solve global optimization and engineering design problems. This method can reduce the problems of premature convergence, local optimal trap and stagnation caused by high selection pressure and low diversity of GWO algorithm. Ghalambaz [17] proposed an improved grey wolf optimization algorithm (GGWO), which is used to minimize the damage caused by the Seattle weather to office buildings to maximize the lifetime of the building. Bardhan et al [18] proposed AGWO and EGWO to estimate the load carrying capacity of steel pipe concrete columns by combining these two optimization methods with artificial neural networks (ANN). Elsisi et al. [19] proposed an MGWO algorithm applied to self-driving cars and combined it with an AMPC controller. Experimental results show that the MGWO-based AMPC controller has higher effective performance compared to other controllers. Ala et al. [20] proposed the use of multi-objective grey wolf optimization (MOGWO) and lexicon weighted tchebycheff (LWT) methods to solve sustainable energy problems. In addition, GWO is combined with various heuristic algorithms, such as the cuckoo search (CS) algorithm, the flower pollination optimization algorithm (FPA), the bat algorithm (BA), etc. [21-23].

The GWO algorithm performs well on small-scale problems with continuous type optimization and is able to search for optimal solutions quickly. However, there are still some limitations when dealing with large-scale data and avoiding local optimal solutions. In particular, there may be some disadvantages when dealing with discrete problems. As a classical combinatorial optimization problem, TSP is often used by researchers to evaluate and test the performance of various discrete algorithms. In recent years, many researchers have been keen on using heuristic algorithms to solve TSP problems, such as ACO, DSSA, SA, HHO, etc. [24-27]. Although many researchers have put a lot of effort into solving the TSP problem, there are very few GWO algorithms for TSP.

Since the GWO algorithm was originally designed for problems with continuous parameters, it cannot solve discrete problems directly. Then, the grey wolf optimization algorithm models the collaborative and competitive strategies of grey wolves, which may not be applicable to all discrete problems and may lead to premature falling into local optimal solutions. To address the above problems, Panwar [28] discretized the solution space of the grey wolf optimization algorithm in 2021, specifically for dealing with discrete problems. The discrete grey wolf optimization algorithm shows better performance when dealing with small to medium sized problems and searching for globally optimal solutions. However, there are still some limitations in some cases. Since the search space for discrete problems is typically more complex, the algorithmic search complexity may increase at larger data sizes. This substantially raises the search time and the risk of falling into a local optimum. These challenges mean that discrete grey wolf optimization algorithms still require further enhancement and optimization in solving complex discrete problems to meet the demands of practical applications.

In this paper, an improved GWO (I-GWO) algorithm is proposed to combine GWO and genetic algorithm (GA) for TSP. I-GWO highlights the advantages of fully utilizing the GWO and GA algorithms and compensates for their shortcomings. To adapt the TSP solution, two strategies, the 2-opt operator and Hamming distance, are introduced to implement the discrete GWO. The search space is expanded with a newly updated population strategy to enhance the diversity of solutions, and crossover and mutation operations are executed in the following iteration. Meanwhile, the I-GWO is embedded with the simulated annealing (SA) algorithm to better the local search performance.

The rest of the article is organized as follows: chapter 2 provides a description of the problem, chapter 3 details the basic algorithms involved, chapter 4 describes the structure of the I-GWO algorithm, chapter 5 presents the experimental data of I-GWO and analyses it in comparison with other algorithms, and, finally, chapter 6 concludes the paper.

II. Problems Description

The travelling salesman problem (TSP) can be mathematically modelled as a complete undirected graph $G = (N, C)$, where N is the set of city locations, $C = \{d(i, j) : i, j \in N\}$ is the set of edges, and d_{ij} is the distance between two city locations i and j. The objective of the TSP is to find the shortest Hamiltonian loop that starts from a certain city and traverses all the cities such that each city can be visited only once by the traveler before returning to the starting city.

For ease of description, a mapping $\pi:=i \rightarrow j$ is used to represent the one-to-one correspondence between the sequential numbering of the cities visited by the traveler and the locations of the cities traversed. Let $X= (\pi(1), \pi(2), ..., \pi(n))$ be a mapping solution, where $i =1, 2, ..., n$, represents the ith sequentially visited city location number. For example, a solution is $X= \{3, 5, 2, 4, 1\}$, which represents that the traveler starts from city 3, visits cities 5, 2, 4, and 1 sequentially, and finally goes back to city 3. The mathematical model of the TSP problem is demonstrated as follows:

$$Z = \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} d_{ij} \cdot x_{ij} \tag{1}$$

$$\sum_{j \neq i, j=1}^{n} x_{ij} = 1, \forall i \in N \tag{2}$$

$$\sum_{i \neq j, i=1}^{n} x_{ij} = 1, \forall j \in N \tag{3}$$

Eq. (1) defines the total distance of the traveler's path and ensures that each city is visited only once , as specified by Eq. (2) and (3). Here, x_{ij} represents the selection of the path from city i to city j, where x_{ij}=1 if the path is selected and x_{ij}=0 otherwise.

III. Grey wolf optimization and genetic Algorithms

Before introducing the I-GWO algorithm, this chapter will first elaborate on the related basic algorithms. This will provide the reader with an in-depth understanding. These basic algorithms provide a solid foundation for the innovative techniques in this paper. Our improvement work is carried out within the framework of these algorithms to better meet the needs of the research.

A. Grey Wolf Optimizer Algorithm

Inspired by the hunting strategy and leadership hierarchy of grey wolves, the Grey Wolf Optimizer (GWO) is a recently developed population-based algorithm. In this algorithm, each potential solution in the solution space is treated as a 'grey wolf' that competes and cooperates with others based on its fitness value to find the optimal solution. At the start of the algorithm, a group of individual grey wolves is randomly generated, with each representing a potential solution to the problem. Mathematically, the most optimal solution is referred to as alpha (α), with beta (β) and delta (δ) representing the second and third best solutions, respectively. The remaining candidate solutions are classified as omega (ω). The alpha grey wolf leads the evolution of the whole group towards a better solution, while the beta and delta grey wolves influence the movement of the other grey wolves.

The mathematical model is displayed below.

$$\vec{A} = 2\cdot\vec{a}\cdot\vec{r}_1 - \vec{a} \tag{4}$$

$$\vec{C} = 2\cdot\vec{r_2} \tag{5}$$

$$a = 2\cdot(1-\frac{t}{\text{max}-iteration}) \tag{6}$$

This formulation uses coefficient vectors $\vec{A}$ and $\vec{C}$, along with random values r_1 and r_2 from the interval [0,1]. The parameter $\vec{a}$ regulate the convergence of the GWO algorithm. Its value linearly decreases from 2 to 0. t and *max-iteration* represent the current and maximum number of iterations, respectively.

$$\vec{D} = |\vec{C}\cdot\vec{X_p}(t) - \vec{X_i}(t)| \tag{7}$$

$$\vec{X}(t+1) = |\vec{X}_p(t) - \vec{A}\cdot\vec{D}| \tag{8}$$

Eq. (7) and (8) simulate wolves encircling their prey, with $\vec{D}$ denoting the distance between wolf i and target p at round t, and $\vec{X}(t+1)$ denoting the position of the wolf update in the next round.

$$\begin{aligned} \vec{D_\alpha} &= |\vec{C}\cdot\vec{X_\alpha} - \vec{X_i}(t)| \\ \vec{D_\beta} &= |\vec{C}\cdot\vec{X_\beta} - \vec{X_i}(t)| \\ \vec{D_\delta} &= |\vec{C}\cdot\vec{X_\delta} - \vec{X_i}(t)| \end{aligned} \tag{9}$$

$$\begin{aligned} \vec{X_1} &= |\vec{X_\alpha} - \vec{A}\cdot\vec{D_\alpha}| \\ \vec{X_2} &= |\vec{X_\beta} - \vec{A}\cdot\vec{D_\beta}| \\ \vec{X_3} &= |\vec{X_\delta} - \vec{A}\cdot\vec{D_\delta}| \end{aligned} \tag{10}$$

$$\vec{X_i}(t+1) = \frac{\vec{X_1}+\vec{X_2}+\vec{X_3}}{3} \tag{11}$$

Eq. (9)-(11) depict the scenario where three alpha wolves lead the pack in the pursuit of prey. Eq. (9) is employed to compute the distance between wolf i and the three alpha wolves during the tth iteration, while Eq. (10) determines the values of $\vec{X_1}$, $\vec{X_2}$ and $\vec{X_3}$. These values are then integrated into Eq. (11) to determine the position of wolf i during the t+1 rounds.

B. Genetic Algorithm

The Genetic Algorithm (GA) [29] is a heuristic search and optimization algorithm that was proposed by Holland in 1992. It is inspired by natural selection and genetic mechanisms in biology. The basic idea of the genetic algorithm is to continuously evolve and improve candidate solutions to find the optimal or near-optimal solution to a problem. First, an initial set of individuals is generated, each representing a potential solution; second, the quality of the individuals is evaluated by assessing their fitness using a function dependent on the problem; third, the individuals with the higher fitness are selected as parents and generated by crossover and mutation; and finally, some or all of the parents are replaced by the offspring to form a new generation of the population. This iterative process is repeated until the stopping condition is met.

The pseudo-code of the genetic algorithm is shown as follows:

Algorithm 1 Genetic Algorithm

```
Input N: Population size
Output $X_{Best}$: the best solution
Initialize P, P= {$X_1, X_2, ..., X_N$}: Population
while the termination condition is not met
    Two individuals X were randomly selected as parents.
    Exchange some chromosomes among parents to produce new individuals.
    Mutational manipulation of the zygote chromosomes to produce a new individual $X_{new}$.
    for $X_i$, i =1, 2, ..., N
        According to the selection operation, well-adapted individuals are retained as $X_{Best}$.
    end for
end while
```

IV. Improve Grey Wolf Optimization Algorithm

This chapter presents our adaptation of the GWO for solving the TSP. The D-GWO algorithm has some imperfections in modifying the original algorithm, causing it to easily fall into the local optimal solution and limiting its performance on complex optimization problems. Therefore, there is a significant need to innovate the D-GWO algorithm. The traditional GWO algorithm is effective because it establishes a relationship between the three head wolves by means of which these head wolves work together to steer the whole pack towards a more optimal direction. However, the three head wolves in the D-GWO algorithm iterate independently of each other, and there is a lack of logical relationship between them, which may cause the algorithm to fall into different local optimal solutions without being able to search globally. Such incoherence makes the D-GWO algorithm perform poorly in high-dimensional, complex optimization problems and fails to realize its full potential. By improving the D-GWO algorithm, we propose the I-GWO algorithm. The I-GWO algorithm can introduce more cooperative mechanisms, which makes the individual head wolves work better together and ensures the ability of global search. This improvement is expected to reduce the risk of falling into local optimal solutions and improve the global search ability of the algorithm, thus making it more suitable for solving complex optimization problems. Therefore, we introduce crossover and mutation operations of the genetic algorithm to mix the genes of the main grey wolves. If the mixed solution works well, it will replace the original main grey wolf as the new main grey wolf; otherwise, it will be eliminated. Furthermore, the 2-opt algorithm has been integrated into the I-GWO algorithm using hybrid simulated annealing. This extension of the search area helps to avoid the problem of local optimal solutions.

A. 2-opt

The 2-opt algorithm is a local search algorithm for solving the traveling salesman problem (TSP). The basic idea of the 2-opt algorithm is to continually improve the quality of the current solution by continually exchanging two edges in the path to reduce the total length of the path. As shown in the figure, the path $R_1 =< A,C,E,B,D >$ is updated by the 2-opt algorithm to get a shorter path $R_2 =< A,B,C,D,E >$.

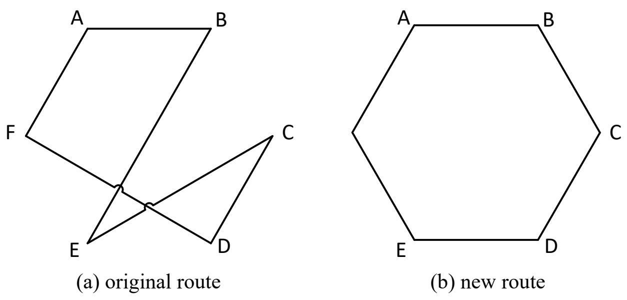


Fig. 1. Simulate 2-opt algorithm example

B. Discrete Grey Wolf Optimization Algorithm

Since the classical GWO algorithm cannot be applied to discrete problems, Karuna proposed a discrete GWO algorithm [12] to solve the TSP problem. In this discrete grey wolf optimization algorithm, he uses the Hamming distance instead of the difference vector $\vec{D}$ eq. (12).

$$\overrightarrow{D_j} = random[1, hd(X_{i\in N}, X_{j\in\alpha,\beta,\delta})] \tag{12}$$

$$X_{ij} = 2-opt(X_{i\in N}, D_{j\in\alpha,\beta,\delta}) \tag{13}$$

Then, Eq. (13) describes that a better solution is obtained by iterating the 2-opt algorithm according to the value of the distance parameter.

C. The improvement method

After solutions generated, we attempt to improve the quality of the solutions by applying simulated annealing strategies. Simulated annealing, originally proposed by Kirkpatrick [30], is a global optimization algorithm mainly used to solve complex combinatorial optimization problems. The uniqueness of this algorithm lies in its ability to effectively avoid falling into local optimal solutions and helps to obtain better global solutions. Its principle is derived from the physical phenomenon of the annealing process of solid materials, in which the material gradually cools down at high temperatures and reaches a stable state. Analogous to this process, the simulated annealing algorithm avoids falling into a local optimum by accepting a worse solution with a certain probability. That is, starting from an initial solution X_0 and an initial temperature T, iterates by repeatedly generating a new solution, calculating the objective function difference, and accepting or discarding for the current solution.

Although the 2-opt algorithm exhibits strong search capabilities when dealing with discrete problems, it is difficult for the 2-opt algorithm to generate a better solution when the algorithm falls into a local optimum. Therefore, to solve the discrete problem, we choose to combine simulated annealing with the 2-opt algorithm to overcome this limitation. This combination can effectively balance global and local search by avoiding falling into a local optimum through the global search capability of simulated annealing and rapidly improving the current solution through the efficient local search of the 2-opt algorithm, which accelerates the convergence of the algorithm, is applicable to a wide range of discrete optimization problems, and improves the chances of finding a more optimal solution.

In order to more effectively combine simulated annealing and the 2-opt algorithm to solve the TSP problem, we have made an improvement by relating the annealing rate to the path distance. The improved annealing rate formula is shown below:

$$probability = e^{\frac{(D(x)-D(x'))/D(x)}{T}} \tag{14}$$

Where *D(x)* is the distance of the shortest path, *D(x')* is the distance of the new path and *T* is the current temperature.

If *D(x') < D(x)* then this solution is necessarily accepted, otherwise the differential solution is accepted according to some probability.

$$probability = \begin{cases} 1, & D(x') < D(x) \\ e^{\frac{(D(x)-D(x'))/D(x)}{T}}, & D(x') \geq D(x) \end{cases} \tag{15}$$

The update process for the hybrid simulated annealing 2-opt algorithm is as follows. First, we screen and rank all possible solutions to select the top three best solutions, which will be the main members of the wolf pack, denoted x_α, x_β and x_δ, respectively. Next, we divide the set of solutions into three parts, representing the top 5%, the 5%-40%, and the remaining 60%, respectively. Each of these three parts corresponds to the main member wolves, and then the distance parameter D is calculated using Eq. (12), and the SA-2-opt algorithm Eq. (16) is applied to update all the solutions.

$$\begin{aligned} x_i &= SA-2-opt(x_i, \vec{D}_\alpha) \\ x_j &= SA-2-opt(x_j, \vec{D}_\beta) \\ x_k &= SA-2-opt(x_k, \vec{D}_\delta) \end{aligned} \tag{16}$$

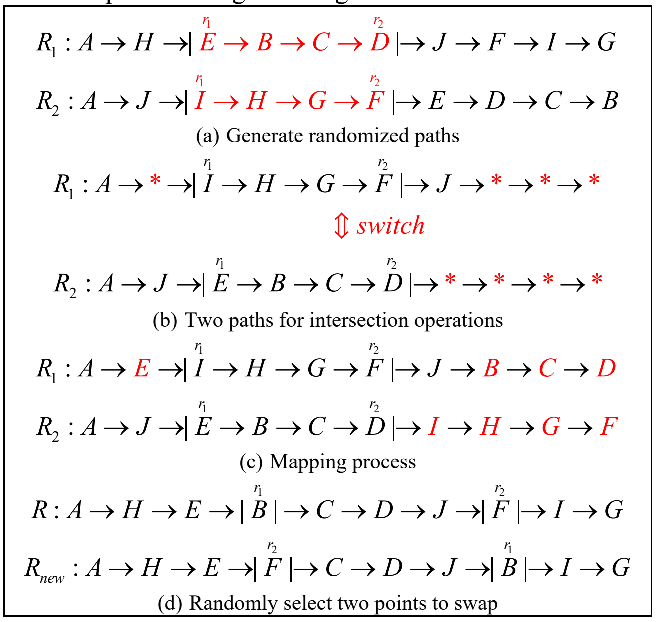
The hybrid simulated annealing 2-opt algorithm pseudocode is shown below:

Algorithm 2 Hybrid SA and 2-opt Algorithm

```
Input $T$: initial temperature $CR$: cooling rate
    $F$: final temperature $\vec{D}$: Distance vector
Initialize The path of the $i$th wolf in the pack $R_i$=<$A_1$, $A_2$, ..., $A_N$>
for $i = 1, 2, ..., \vec{D}$ do
  for $j = 1, 2, ..., N-1$ do
    for $k = j+1, j+2, ..., N$ do
      $Dis(R_i)$ = Total distance of path $R_i$
      $R_{new}$ = 2-opt exchange for path $R_i$
      $Dis(R_{new})$ = Total distance of path $R_{new}$
      if $Dis(R_i) > Dis(R_{new})$ then
        $R_i = R_{new}$
      else
        Accepting the worse solution according to the probability of Eq. 15
        $R_i = R_{new}$
      if $T > F$ then
        $T = T * CR$
      end if
    end for
  end for
end for
```

D. Crossover and Mutation Operations

In the improved grey wolf optimization (I-GWO) algorithm, crossover and mutation operations are used to enhance search diversity and find better solutions. By selecting three head wolves and crossover and mutate their genes, we aim to increase the exploration range of the solution space in the expectation of obtaining better solutions. The crossover operation promotes the combination of the strengths of different head wolves, while the mutation operation introduces stochasticity to help the algorithm jump out of the local optimal solution, thus improving the adaptability of GWO and making it more competitive in a variety of optimization problems. Fig. 2 illustrates the crossover and mutation parts of the genetic algorithm.



$R_1 : A \to H \to | \overset{r_1}{E} \to B \to C \to \overset{r_2}{D} | \to J \to F \to I \to G$
$R_2 : A \to J \to | \overset{r_1}{I} \to H \to G \to \overset{r_2}{F} | \to E \to D \to C \to B$
(a) Generate randomized paths
$R_1 : A \to * \to | \overset{r_1}{I} \to H \to G \to \overset{r_2}{F} | \to J \to * \to * \to *$
⇕ switch
$R_2 : A \to J \to | \overset{r_1}{E} \to B \to C \to \overset{r_2}{D} | \to * \to * \to * \to *$
(b) Two paths for intersection operations
$R_1 : A \to E \to | \overset{r_1}{I} \to H \to G \to \overset{r_2}{F} | \to J \to B \to C \to D$
$R_2 : A \to J \to | \overset{r_1}{E} \to B \to C \to \overset{r_2}{D} | \to I \to H \to G \to F$
(c) Mapping process
$R : A \to H \to E \to | \overset{r_1}{B} | \to C \to D \to J \to | \overset{r_2}{F} | \to I \to G$
$R_{new} : A \to H \to E \to | \overset{r_2}{F} | \to C \to D \to J \to | \overset{r_1}{B} | \to I \to G$
(d) Randomly select two points to swap

Fig. 2. Genetic algorithm crossover and mutation operations

First two random paths are generated and two points r_1 and r_2 are randomly generated in the interval [1.L] according to the length of the paths L. Let r_1=3, r_2=6. The process is shown as (a) in Fig. 2

Then the crossover operation is performed between the two points, after the crossover operation is finished, there will be duplicated points under the same path, the non-duplicated points will be retained and the conflicting points will be replaced with *. The exchanged path is shown as (b) in Fig. 2

Finally, the [r_1, r_2] intervals of the corresponding paths are used for mapping, i.e., the intervals [1, r_1) and (r_2, L] of *Route1* are mapped with the intervals [r_1, r_2] of Route2. *Route2* ditto. The process is shown in (c) of Fig. 2.

The selection of initial random points in the mutation operation is consistent with the selection of random points operation in the crossover operation, where two points r_1 and r_2 are randomly selected between the intervals [1, L], which are then exchanged for position. Let r_1=4, r_2=8. The process is shown in (d) in Fig. 2.

The crossover and mutation modifications to the head wolf gene operate as follows. After updating all the solutions using Eq. (16) in the previous section, we select the three best solutions from them, denoted x_α, x_β, and x_δ. Next, we recombine these three solutions through the crossover and mutation operations of the genetic algorithm of Eq. (17) and decide whether to keep the newly generated solutions or not according to Eq. (18), retaining them only if the new solutions are better and discarding them otherwise.

$$\begin{aligned} x_{\alpha new} &= GA(x_\alpha, x_\beta) \\ x_{\beta new} &= GA(x_\alpha, x_\delta) \\ x_{\delta new} &= GA(x_\beta, x_\delta) \end{aligned} \tag{17}$$

$$x_{new} = \begin{cases} x_{old}, & Dis(x_{new}) > Dis(x_{old}) \\ x_{new}, & Dis(x_{new}) \le Dis(x_{old}) \end{cases} \tag{18}$$

The pseudo-code of the crossover and mutation part of the genetic algorithm is shown below:

Algorithm 1 The Crossover and Mutation Of GA

```
Input $N$: Population size
Output $X_{Best}$: the best solution
Initialize $P$, $P = \{X_1, X_2, ..., X_N\}$: Population
while the termination condition is not met
  for $X_i$, $i = 1, 2, ..., N$
    Exchange some chromosomes among pairs of individuals to produce new individuals.
    Mutation operators act on the population to obtain new individuals $X_{new}$.
  end for
  for $X_i$, $i = 1, 2, ..., N$
    Well adapted individuals are retained according to the Selection operation.
  end for
end while
```

In each iteration, we introduce a competitive mechanism in the wolf pack to ensure that the better wolves remain in the pack, while the relatively weaker wolves have a higher probability of elimination, which can be determined by Eq. (19), where i represents the ranking of an individual in the wolf population and N denotes the total population size.

$$\text{rate} = \frac{i}{N} \tag{19}$$

E. Procedure of I-GWO Algorithm

This section shows pseudocode to improve the grey wolf optimization algorithm. It is divided into three main stages. First, the wolves are initialized to generate the initial population. In the second stage, the genes of the wolf population are modified using the crossover and mutation operations of the genetic algorithm to expand the search space and increase the diversity of solutions. Finally, a hybrid SA-2-opt algorithm is embedded in I-GWO to improve the local search performance. The pseudocode for improving the grey wolf optimization algorithm is as follows:

```
Algorithm 3 Improve Grey Wolf Optimization
  Input $N$: population size $M$: max iteration
        $R$: randomized route of TSP
  Initialize population of wolves $x_i$ ($i = 1, 2, ..., N$) $x_i \in R$
  for $i = 1, 2, ..., N$ do
    $D_i$= Calculate the length of $x_i$
    $D$: The full distance of the wolf's path
  end
  Choose three optimal solutions
    $x_\alpha$: the shortest path $D_\alpha$ of $D_{i\in N}$
    $x_\beta$: second shortest path $D_\beta$ of $D_{i\in N}$
    $x_\delta$: third shortest path $D_\delta$ of $D_{i\in N}$
  for $t = 1, 2, ..., M$ do
    According to the ratio of the first 5%, the first 5% -
 40%, and the last 60% will all be dissolved into three
 zones a, b, c.
      for $i = 1, 2, ..., a$ do
        update the path of wolf $x_i$ according to Eq. (12)
        and Eq. (16)
        Elimination of poor solutions according to the
        probability of Eq. (19)
      end for
      for $j = a, a+1, ..., b$ do
        update the path of wolf $x_j$ according to Eq. (12)
        and Eq. (16)
        Elimination of poor solutions according to the
        probability of Eq. (19)
      end for
      for $k = b, b+1, ..., N$ do
        update the path of wolf $x_k$ according to Eq. (12)
        and Eq. (16)
        Elimination of poor solutions according to the
        probability of Eq. (19)
      end for
      update $x_\alpha$, $x_\beta$, $x_\delta$ using Eq. (17) and (18)
  end for
  return the best route $x_\alpha$
```

V. Experiment

In this section, experiments will be conducted on the research points presented in this paper, as well as comparisons with other algorithms. This chapter is divided into two main parts, in the first part, the comparison between the proposed improved discrete grey wolf optimization algorithm and the discrete grey wolf optimization algorithm is described. Then in the second part, the experimental results of the improved discrete grey wolf optimization algorithm are shown and compared with other algorithms. A total of 26 TSP instances were used in the experimental comparisons, and the TSP data used for the experiments were obtained from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95 /tsp/. As well as in this study all test behaviors were done on an AMDRyzen7 5800h laptop with 3.20 GHz and 16 GB of RAM. The programming language used is Python.

A. Experimental comparison between I-GWO and D-GWO algorithms

In order to test that I-GWO performance, we conduct a comparison test between I-GWO and D-GWO in this chapter. To ensure that I-GWO can perform optimally, we conducted several parameter tests, including the initial solution, the SA-2-opt algorithm, the cross- mutation of the GA, and the number of iterations. The final parameters are those that perform best in the experiments. Table 1 summarizes the relevant parameters used for I-GWO and D-GWO. To keep the experiments fair, we conducted 20 independent experiments for each instance, using the same number of runs to be consistent with the D-GWO algorithm. We used 17 instances of TSP in our experiments, and the experimental data for D-GWO was taken from a recently published study [12]. Table 2 presents the results of the comparison between the algorithms, focusing on the optimal solution, the average solution, the average difference value, and the average running time. When the algorithms are run, we set $N + \sum_{I=1}^{N}$ as the termination condition, i.e., the algorithm ends when the optimal solution obtained by the algorithm in N generations is not updated. Where N is the number of nodes in the TSP instance. In the next sections, we will continue with comparisons with other heuristic algorithms and apply the above configurations to these comparisons.

In Table 2, we compare the performance of I-GWO and D-GWO in terms of mean, best solution, mean difference and running time. As shown in (a) in Fig. 3, I-GWO is significantly better than D-GWO in obtaining the mean, which means that the results of I-GWO are more stable and less fluctuating when obtaining the best solution. Figure 3 in (b) shows the running time spent by the two algorithms in solving the TSP instances, which shows that I-GWO has higher efficiency in solving the TSP problem, which helps to cope with the increasing data volume and complexity. In addition, I-GWO shows improved performance in terms of the best solution, which is able to explore a wider search space even under the same initial population condition. Taken together, I-GWO clearly outperforms D-GWO, showing superior performance.

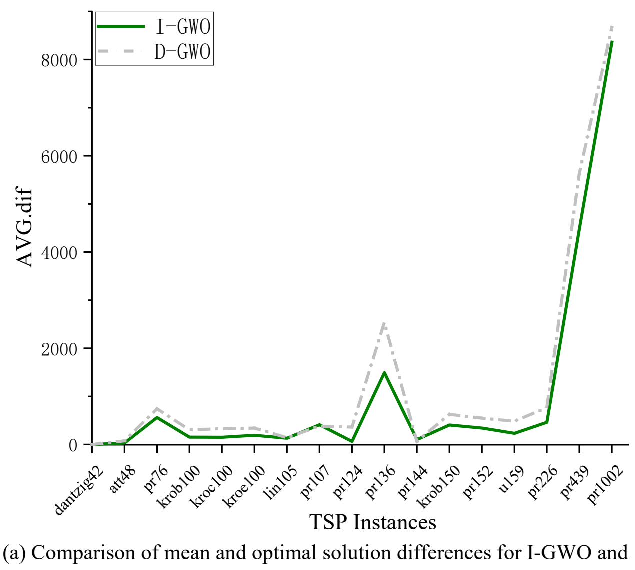
Figure 4 illustrates the comparative results of the convergence of the I-GWO and D-GWO algorithms. In our experiments, we used four different TSP instances including Pr76, KroB100, Pr152, and Pr266.First, we observe that the I-GWO performs more extensively in terms of the solution search range relative to the D-GWO algorithm. This is manifested in the graphs, where the fold fluctuations of the I-GWO algorithm are significantly more pronounced, indicating its larger search range. This is an important advantage for solving the TSP problem. Next, we focused on the convergence speed. From Figures a, b and c we can clearly observe that the I-GWO algorithm converges significantly faster than the D-GWO algorithm on these three instances. This means that I-GWO is able to find solutions faster, especially when solving small TSP instances. However, in Figure d we note that the I-GWO algorithm converges relatively slowly. Nonetheless, it is able to jump out of the local optimum and end up with a better solution by combining it with the hybrid simulated annealing 2-opt algorithm. This suggests that the I-GWO algorithm may have a better global search capability when dealing with large TSP instances, and can more easily overcome the limitation of local optimal solutions and improve the quality of the solution compared to the D-GWO algorithm. In summary, the I-GWO algorithm has a larger search area, faster convergence speed, and potential performance in jumping out of the local optimum when solving TSP problems.

TABLE I
PARAMETERIZATION OF I-GWO, DGWO AND IBA

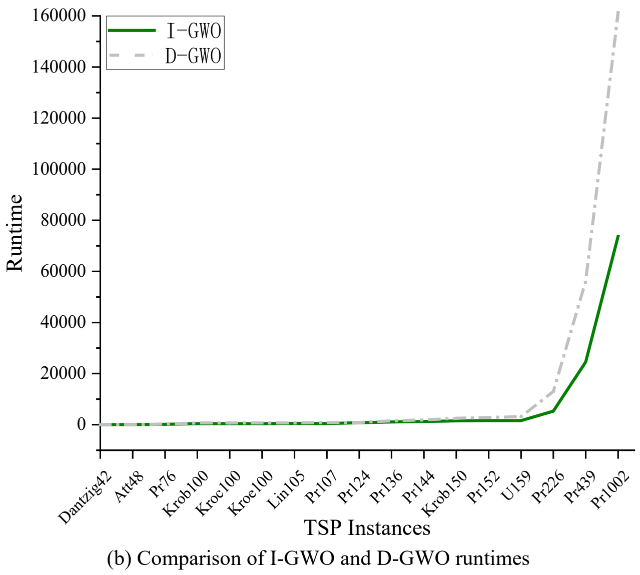
I-GWO		D-GWO	
Parameter	Value	Parameter	Value
Population size	50	Population size	50
Iteration	20	Iteration	20
Update method	SA-2-opt&GA	Update method	2-opt
Temperature	t=100*0.95^{iter},$t\in$[1,100]		
Crossover function	Ordered Crossover		
Mutation function	Insertion		

TABLE II
RESULT OF PROPOSED I-GWO AND D-GWO FOR THE TSP INSTANCES.

Instance		I-GWO				D-GWO			
Name	Optima	Avg	Best	A.dif	Time(s)	Avg	Best	A.dif	Time(s)
Dantzig42	679	**679.0**	679	**0**	29.6	680	679	1	40.69
Att48	33,523	**33,552.6**	33,523	**29.6**	43.08	33,600	**33,522**	77	60.07
Pr76	108,159	**108,719.4**	**108,159**	**560.4**	161.82	108,900	**108,159**	741	264.23
Krob100	22,140	**22,292.2**	**22,140**	**152.2**	389.82	22,444.6	22,159	304.6	689.34
Kroc100	20,749	**20,899.8**	**20,749**	**150.8**	375.2	21,078	**20,749**	329	688.43
Kroe100	22,068	**22,260.0**	**22,068**	**192**	394.13	22,410	22,131	342	685.46
Lin105	14,379	**14,510.2**	**14,379**	**131.2**	520.32	14,520	14,382	141	685.33
Pr107	44,303	44,713.2	44,303	410.2	454.25	**44,685.1**	**44,301**	**382.1**	830.54
Pr124	59,030	**59,092.8**	**59,030**	**62.8**	767.5	59,390.9	**59,030**	360.9	888.3
Pr136	96,772	**98,266.3**	**97,532**	**1494.3**	1055	99,310.5	97,826	2538.5	1486.25
Pr144	58,537	58,637.8	58,537	100.8	1238.53	**58,600.5**	**58,535**	**63.5**	1866
Krob150	26,130	**26,535.4**	**26,231**	**405.4**	1473.81	26,756.2	26,320	626.2	2503.2
Pr152	73,682	**74,022.6**	**73,687**	**340.6**	1579.32	74,230	73,690	548	2856.22
U159	42,080	**42,312.4**	**42,133**	**232.4**	1603.85	42,563.3	42,142	483.3	3108.6
Pr226	80,369	**80,830.5**	**80,551**	**461.4**	5303.5	81,135.7	80,648	766.7	12,971.29
Pr439	107,217	**111,705.0**	**109,925**	**4488**	24,523.4	112,850.3	110,415	5633.3	56,225.8
Pr1002	259,047	**267410.2**	**264,513**	**8363.2**	73,685.34	267,713.2	264,922	8666.2	161,240.9



(a) Comparison of mean and optimal solution differences for I-GWO and D-GWO



(b) Comparison of I-GWO and D-GWO runtimes

Fig. 3. I-GWO and D-GWO performance comparison chart

B. Comparison of I-GWO and other heuristic algorithms

In this chapter, we intend to conduct a comparative study using 18 TSP instances with the aim of fully evaluating the performance differences between the I-GWO algorithm and a number of other heuristic algorithms.

As mentioned above, the TSP serves as a standard benchmark for evaluating discrete optimization algorithms. While I-GWO achieves commendable results for the TSP, reaching the optimal solution in 13 out of 17 cases, it is important to emphasize that the primary goal of this study is not to find optimal solutions. Rather, both problems are used as benchmarks to demonstrate the adaptability of GWO to routing problems and to show the potential of I-GWO as an effective approximation method for solving the TSP.

To achieve this, we aim to show that the I-GWO can outperform well-known metaheuristics in the literature using standard non-heuristic functions. For comparison, we've chosen three historically successful techniques (GA, SA, IDGA) and two recent ones (DFA, DICA, BA).

It's important to emphasize that we have tried to use identical operators and parameters in all implemented algorithms for consistency in our experiments. It's also worth noting that all individuals are randomly generated.

These algorithms represent different problem solving strategies, including GA, ESA, IDGA, DFA, DICA and BA. Through these comprehensive comparative experiments, we aim to gain insights into the performance of these algorithms in solving real-world problems in terms of their robustness, search capability, and potential for global optimal solution discovery. This will help to provide deeper insights and better performance evaluation for problem solving.

The parameterization for I-GWO corresponds to Table 1 from the previous section. The results for I-GWO, ESA, GA and IDGA over 18 instances are presented in Table 3, showing average and best results along with average differences. Best average results are highlighted and optimal solutions are noted. In particular, to avoid duplication, some of the I-GWO results in Table 4 are the same as those in Table 2.

The experimental data of these heuristic algorithms come from the study published by Osaba in 2016 [31], which provides us with a valuable reference. In the comparison study, we paid special attention to several key performance indicators, including the average value, the optimal solution, and the average difference, in order to comprehensively evaluate the performance of each algorithm. The detailed comparison data is presented in Tables 3 and 4, and the average performance difference between these algorithms is visualized in Figure 5.

On the other hand, Table 4 compares the results obtained by I-GWO in the 15 cases with those obtained by a BA, a DFA and a DICA. As before, we have used similar parameters and functions in these techniques in order to obtain fair conclusions. I-GWO, BA, DFA and DICA all used the same number of individuals and relied on the Hamming distance function for their movements. Similar conclusions can be drawn from this table: I-GWO performed best, achieving better results in 83.3% of cases (15 out of 18), with only three cases where it performed worse. Overall, I-GWO outperformed DFA and DICA in all 18 TSP cases and BA in 15 TSP cases.

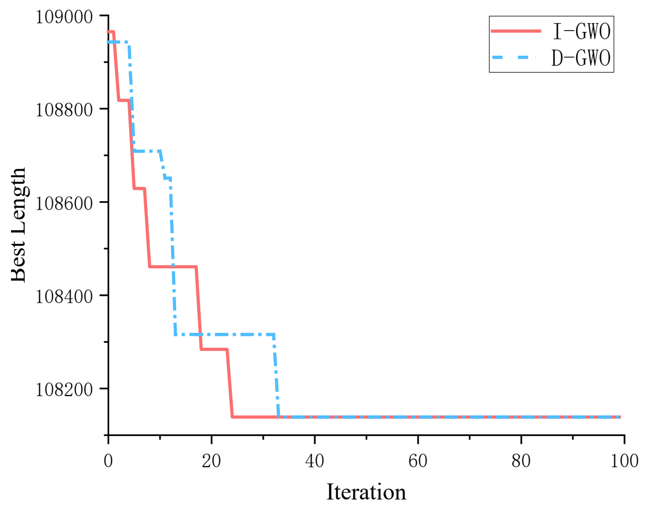
From the data in Tables 3 and 4, it is clear that the I-GWO algorithm performs well in handling these TSP instances. Although it is slightly inferior to the BA algorithm in several instances, namely Eil101, Pr264 and Pr299, I-GWO is more consistent in terms of average performance. This advantage is not only reflected in the optimal solution, but also in the key of metric mean difference. Looking at the mean difference comparisons in Figure 5, it is clear that the I-GWO algorithm exhibits greater stability when dealing with multiple instances, which means that it is able to maintain a consistent level of excellent performance across different problems.

VI. Discussion

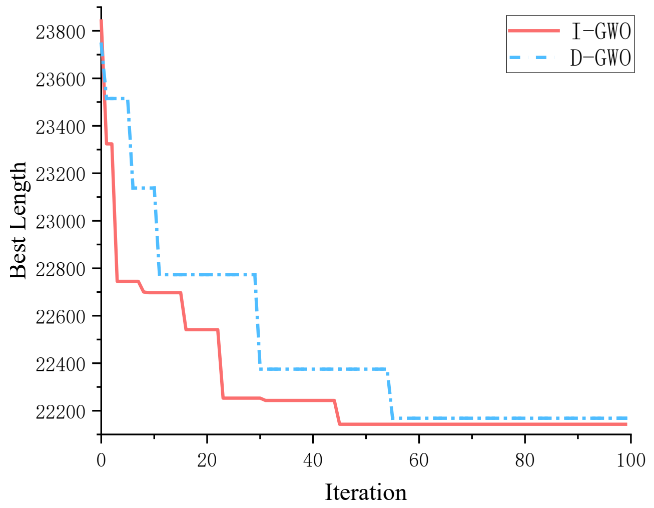
A comparison of the Improved Grey Wolf Optimization (I-GWO) algorithm with D-GWO and other heuristics highlights the superior performance of I-GWO in solving the Travelling Salesman Problem (TSP) and its potential to address a wider range of optimization problems. Compared to D-GWO, I-GWO exhibits greater stability in obtaining average solutions and shorter run times, suggesting that it is more efficient in solving TSP instances. In addition, I-GWO is able to explore a larger search space and obtain better solutions, especially for smaller TSP instances. Although convergence is relatively slow in some cases, I-GWO's ability to get rid of local optima through hybridization highlights its potential in larger TSP instances. Across multiple TSP instances, I-GWO consistently outperformed established metaheuristics and demonstrated overall consistency and stability, making it ideal for TSP optimization. Future research could focus on further refining I-GWO through algorithmic tuning and hybrid strategies to exploit its advantages. In addition, exploring scalability and parallelization techniques will help improve the applicability of I-GWO to larger optimization problems, making it an important tool for various practical applications. Overall, I-GWO is a promising optimization technique that outperforms traditional and contemporary heuristic algorithms in the field of TSP optimization. Its demonstrated superior performance and future research prospects make it the method of choice for a variety of complex optimization challenges.

VII. Conclusion

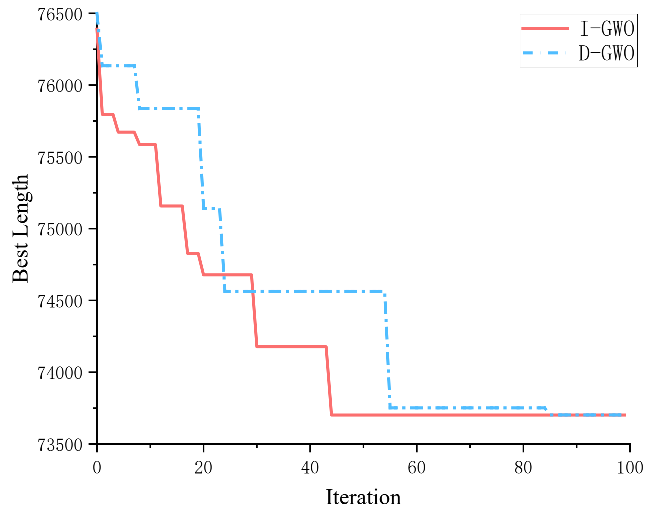
In this study, an improve grey wolf optimization algorithm I-GWO is proposed as a new approach to solve the TSP problem. Since the discrete grey wolf optimization algorithm has the defects of falling into local optimum and poor performance release slow convergence speed when facing the TSP problem. Therefore, the D-GWO algorithm is modified by adding simulated annealing for updating iterations, as well as adding genetic algorithm to control the genetic changes of the population. To verify the performance of I-GWO, 26 TSP instances are used to compare with other algorithms. From the experiments, it can be concluded that the improvement of I-GWO is beneficial, and the robustness of the algorithm, the faster running efficiency and the convergence speed can be improved by these changes. It has a larger search space in the same running time.



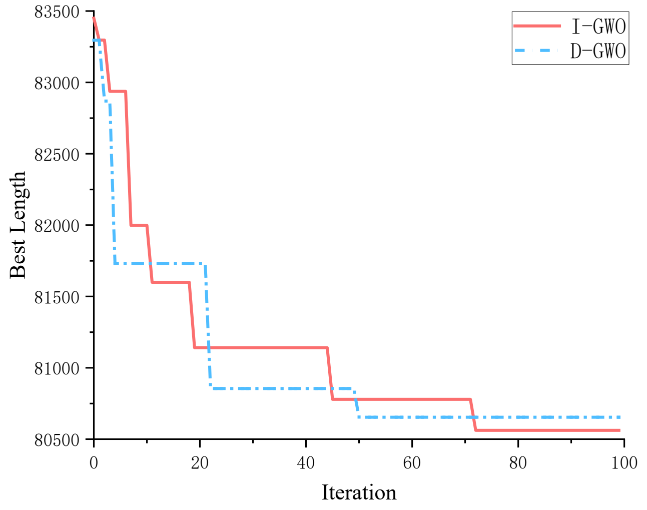
(a) TSP instance Pr76



(b) TSP instance KroB100

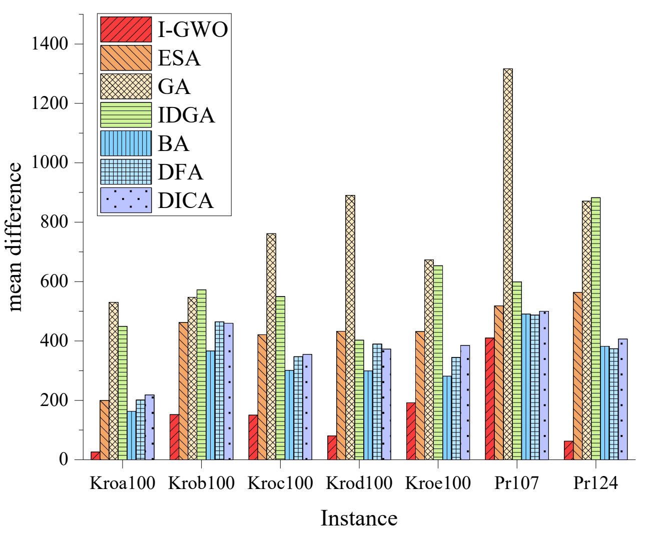


(c) TSP instance Pr152



(d) TSP instance Pr226

Fig. 4. Convergence of I-GWO algorithm demonstrated



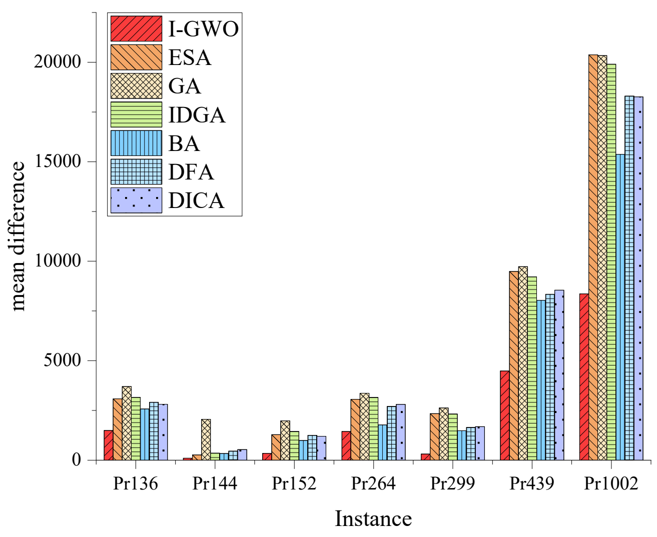


Fig. 5. Comparison of algorithmic mean difference

TABLE III
RESULT OF PROPOSED I-GWO AND ESA, GA AND IDGA FOR THE TSP INSTANCES.

Instance		I-GWO			ESA			GA			IDGA		
Name	Optima	Avg	Best	A.dif	Avg	Best	A.dif	Avg	Best	A.dif	Avg	Best	A.dif
Eil51	426	**426.8**	**426**	**0.8**	431.6	**426**	5.6	440.8	427	14.8	434.4	**426**	8.4
Berlin52	7542	**7542**	**7542**	**0**	**7542.0**	**7542**	**0**	**7542.0**	**7542**	**0**	**7542.0**	**7542**	**0**
Eil76	538	**541**	**538**	**3**	553.7	546	15.7	565.4	545	27.4	557.7	545	19.7
Kroa100	21,282	**21,308.2**	**21,282**	**26.2**	21,481.7	**21,282**	199.7	21,812.4	21,350	530.4	21,731.8	21,345	449.8
Krob100	22,140	**22,292.2**	**22,140**	**152.2**	22,602.2	22,202	462.2	22,687.4	22,176	547.4	22,712.6	22,208	572.6
Kroc100	20,749	**20,899.8**	**20,749**	**150.8**	21,170.4	**20,749**	421.4	21,510.4	20,861	761.4	21,298.7	20,830	549.7
Krod100	21,294	**21374.8**	**21,294**	**80.8**	21,726.5	21,500	432.5	22,184.6	21,492	890.6	21,696.9	21,582	402.9
Kroe100	22,068	**22,260**	**22,068**	**192**	22,499.7	22,099	431.7	22,741.3	22,150	673.3	22,721.9	22,110	653.9
Eil101	629	**646.2**	**635**	**17.2**	658.4	650	29.4	673.8	655	44.8	660.7	650	31.7
Pr107	44,303	**44,713.2**	**44,303**	**410.2**	44,821.5	44,413	518.5	45,619.6	44,392	1316.6	44,902.5	44,428	599.5
Pr124	59,030	**59,092.8**	**59,030**	**62.8**	59,593.6	**59,030**	563.6	59,901.0	**59,030**	871	59,912.8	59,072	882.8
Pr136	96,772	**98,266.3**	**97,532**	**1494.3**	99,858.3	98,499	3086.3	100,472.4	98,432	3700.4	99,932.7	98,532	3160.7
Pr144	58,537	**58,637.8**	**58,537**	**100.8**	58,807.3	58,574	270.3	60,591.4	58,599	2054.4	58,893.0	58,581	356
Pr152	73,682	**74,022.6**	**73,687**	**340.6**	74,969.5	74,172	1287.5	75,658.3	74,520	1976.3	75,126.7	74,249	1444.7
Pr264	49,135	**50,581.4**	**49,864**	**1446.4**	52,198.5	51,603	3063.5	52,499.8	51,712	3364.8	52,290.0	51,653	3155
Pr299	48,191	**48,501.2**	**48,323**	**310.2**	50,532.3	49,242	2341.3	50,817.1	49,659	2626.1	50,513.3	49,572	2322.3
Pr439	107,217	**111,705.0**	**109,925**	**4488**	116,706.9	113,497	9490	116,943.4	113,576	9726.4	116,436.1	113,207	9219.1
Pr1002	259,047	**267410.2**	**264,513**	**8363.2**	279,419.7	273,496	20,372.7	279,384.7	273,001	20,337.7	278,951.4	272,893	19,904.4

TABLE IV
RESULT OF PROPOSED I-GWO AND BA, DFA AND DICA FOR THE TSP INSTANCES.

Instance		I-GWO			BA			DFA			DICA		
Name	Optima	Avg	Best	A.dif	Avg	Best	A.dif	Avg	Best	A.dif	Avg	Best	A.dif
Eil51	426	**426.8**	**426**	**0.8**	428.1	**426**	2.1	430.8	**426**	4.8	432.3	**426**	6.3
Berlin52	7542	**7542**	**7542**	**0**	**7542.0**	**7542**	**0**	**7542.0**	**7542**	**0**	**7542.0**	**7542**	**0**
Eil76	538	**541**	**538**	**3**	548.1	539	10.1	556.8	543	18.8	557.6	544	19.6
Kroa100	21,282	**21,308.2**	**21,282**	**26.2**	21,445.3	**21,282**	163.3	21,483.6	**21,282**	201.6	21,500.3	21,282	218.3
Krob100	22,140	**22,292.2**	**22,140**	**152.2**	22,506.4	**22,240**	366.4	22,604.8	22,183	464.8	22,599.7	22,180	459.7
Kroc100	20,749	**20,899.8**	**20,749**	**150.8**	21,050.0	**20,749**	301	21096.3	20,756	347.3	21,103.9	20,756	354.9
Krod100	21,294	**21374.8**	**21,294**	**80.8**	21,593.4	**21,294**	299.4	21,683,8	21,408	389.8	21,666.8	21,399	372.8
Kroe100	22,068	**22,260**	**22,068**	**192**	22,349.6	**22,068**	281.6	22413	22,079	345	22,453.3	22,083	385.3
Eil101	629	**646.2**	635	**17.2**	646.4	**634**	17.4	659.0	643	30	663.8	644	34.8
Pr107	44,303	**44,713.2**	**44,303**	**410.2**	44,793.8	**44,303**	490.8	44790.4	**44,303**	487.4	44,803.3	**44,303**	500.3
Pr124	59,030	**59,092.8**	**59,030**	**62.8**	59,412.1	**59,030**	382.1	59,404.3	**59,030**	374.3	59,436.9	**59,030**	406.9
Pr136	96,772	**98,266.3**	**97,532**	**1494.3**	99,351.2	97,547	2579.2	99,683.7	97,716	2911.7	99,583.7	92,736	2811.7
Pr144	58,537	**58,637.8**	**58,537**	**100.8**	58,876.2	**58,537**	339.2	58993.3	58,546	456.3	59,070.9	58,563	533.9
Pr152	73,682	**74,022.6**	**73,687**	**340.6**	74,676.9	73,921	994.9	74,934.3	74,033	1252.3	74,886.7	74,052	1204.7
Pr264	49,135	**50,581.4**	49,864	**1446.4**	50,908.3	**49,756**	1773.3	51.837.0	50,491	2702	51,943.6	50,553	2808.6
Pr299	48,191	**48,501.2**	48,323	**310.2**	49,674.1	**48,310**	1483.1	49,839.7	48,579	1648.7	49,880.3	48,600	1689.3
Pr439	107,217	**111,705.0**	**109,925**	**4488**	115,256.4	111,538	8039.4	115,558.2	111,967	8341.2	115,763.1	111,983	8546.1
Pr1002	259,047	**267,410.2**	**264,513**	**8363.2**	274,419.7	270,016	15,372.7	277,344.7	272,003	18,297.7	277,308.1	272,082	18,261.1

REFERENCES

[1] D. B. Shmoys, J. K. Lenstra, et al. "The traveling salesman problem," *John Wiley & Sons, Incorporated*, vol. 12, 1985.

[2] I. Kara, T. Bektas, "Integer linear programming formulations of multiple salesman problems and its variations." *European Journal of Operational Reseasrch*, vol. 174, no. 3, pp1449-1458, 2006.

[3] A. G. Chentsov, L. N. Korotayeva, "The dynamic programming method in the generalized traveling salesman problem." *Mathematical and computer modelling* vol. 25, no. 1, pp93-105, 1997.

[4] M. Padberg, G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems." *SIAM review* vol. 33, no. 1, pp60-100, 1991.

[5] R. Skinderowicz, "Improving Ant Colony Optimization efficiency for solving large TSP instances." *Applied Soft Computing*, 120, pp108653 2022.

[6] Y. Huang, X. N. Shen, and X. You, "A discrete shuffled frog-leaping algorithm based on heuristic information for traveling salesman problem." *Applied Soft Computing* vol. 102, pp107085, 2021.

[7] W. Gao, "Modified ant colony optimization with improved tour construction and pheromone updating strategies for traveling salesman problem." *Soft Computing* vol. 25, no. 4, pp3263-3289, 2021.

[8] Y. Wang, Z. Han, "Ant colony optimization for traveling salesman problem based on parameters optimization." *Applied Soft Computing* vol. 107, pp107439, 2021.

[9] P. Zhang, J. Wang, "A genetic algorithm with jumping gene and heuristic operators for traveling salesman problem." *Applied Soft Computing* vol. 127, pp109339, 2022.

[10] S. K. R. Kanna, K. Sivakumar, and N. Lingaraj, "Development of deer hunting linked earthworm optimization algorithm for solving large scale traveling salesman problem." *Knowledge-Based Systems* vol. 227, pp107199, 2021.

[11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer." *Advances in engineering software* vol. 69, pp46-61, 2014.

[12] I. A. Zamfirache, et al. "Policy iteration reinforcement learning-based control using a grey wolf optimizer algorithm." *Information Sciences* vol. 585, pp162-175, 2022.

[13] H. U. Ahmed, et al. "Support vector regression (SVR) and grey wolf optimization (GWO) to predict the compressive strength of GGBFS-based geopolymer concrete." *Neural Computing and Applications* vol. 35, no. 3, pp2909-2926, 2023.

[14] A. Altan, S. Karasu, and E. Zio, "A new hybrid model for wind speed forecasting combining long short-term memory neural network, decomposition methods and grey wolf optimizer." *Applied Soft Computing* vol. 100, pp106996, 2021.

[15] M. H. Nadimi-Shahraki, S. Taghian, and S. Mirjalili, "An improved grey wolf optimizer for solving engineering problems." *Expert Systems with Applications* vol. 166, pp113917, 2021.

[16] M. H. Nadimi-Shahraki, S. Taghian, and S. Mirjalili, "GGWO: Gaze cues learning-based grey wolf optimizer and its applications for solving engineering problems." *Journal of Computational Science* vol. 61, pp101636, 2022.

[17] M. Ghalambaz, R. J. Yengejeh, and A. H. Davami, "Building energy optimization using grey wolf optimizer (GWO)." *Case Studies in Thermal Engineering* vol. 27 pp101250, 2021.

[18] A. Bardhan, et al. "A novel integrated approach of augmented grey wolf optimizer and ANN for estimating axial load carrying-capacity of concrete-filled steel tube columns." *Construction and Building Materials*, vol. 337, pp127454, 2022.

[19] M. Elsisi, "Improved grey wolf optimizer based on opposition and quasi learning approaches for optimization: Case study autonomous vehicle including vision system." *Artificial intelligence review*, vol. 55, no. 7, pp5597-5620, 2022.

[20] A. Ala, et al. "A Novel Neutrosophic-Based Multi-Objective Grey Wolf Optimizer for Ensuring the Security and Resilience of Sustainable Energy: A Case Study of Belgium." S*ustainable Cities and Society*, pp104709, 2023.

[21] V. Bathina, R. Devarapalli, and F. P. García Márquez, "Hybrid approach with combining cuckoo-search and grey-wolf optimizer for solving optimal power flow problems." *Journal of Electrical Engineering & Technology*, vol. 18, no. 3, pp1637-1653, 2023.

[22] S. Hemalatha, G. Banu, and K. Indirajith, "Design and investigation of PV string/central architecture for bayesian fusion technique using grey wolf optimization and flower pollination optimized algorithm." *Energy Conversion and Management*, vol. 286, pp117078, 2023.

[23] K. Balasubramanian, K. Ramya, and K. Gayathri Devi. "Optimized adaptive neuro-fuzzy inference system based on hybrid grey wolf-bat algorithm for schizophrenia recognition from EEG signals." *Cognitive Neurodynamics*, vol. 17, no. 1, pp133-151, 2023.

[24] P. Stodola, P. Otřísal, and K. Hasilová, "Adaptive ant colony optimization with node clustering applied to the travelling salesman problem." *Swarm and Evolutionary Computation*, vol. 70, pp101056, 2022.

[25] Z. Zhang, and Y. Han, "Discrete sparrow search algorithm for symmetric traveling salesman problem." *Applied Soft Computing* vol. 118, pp108469, 2022.

[26] İ. İlhan, and G. Gökmen, "A list-based simulated annealing algorithm with crossover operator for the traveling salesman problem." *Neural Computing and Applications*, pp1-26, 2022.

[27] F. S. Gharehchopogh, and B. Abdollahzadeh. "An efficient harris hawk optimization algorithm for solving the travelling salesman problem." *Cluster Computing*, vol. 25, no.3, pp1981-2005, 2022.

[28] K. Panwar, and K. Deep. "Discrete Grey Wolf Optimizer for symmetric travelling salesman problem." *Applied Soft Computing*, vol. 105, pp107298, 2021.

[29] J. H. Holland, "Genetic algorithms." *Scientific american* vol. 267, no.1, pp66-73, 1992.

[30] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies." *Journal of statistical physics* vol. 34, pp975-986, 1984.

[31] E. Osaba, et al. "An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems." *Engineering Applications of Artificial Intelligence* vol. 48, pp59-71, 2016.

Zhinan Xu is currently a Postgraduate in School of Computer Science and Software Engineering at University of Science and Technology Liaoning, Anshan, China. His major research is Intelligent optimization algorithms.

Xiaoxia Zhang is currently a Professor in School of Computer Science and Software Engineering at University of Science and Technology Liaoning, Anshann China. Her major research is Intelligent optimization algorithms.