applied sciences

MDPI

Article

An Improved Tunicate Swarm Algorithm for Solving the MultiObjective Optimisation Problem of Airport Gate Assignments

Yu Zhang [1], Qing He [1,*], Liu Yang [2,*] and Chenghan Liu [1]

[1] College of Big Data & Information Engineering, Guizhou University, Guiyang 550025, China
[2] School of Public Administration, Guizhou University, Guiyang 550025, China
* Correspondence: qhe@gzu.edu.cn (Q.H.); lyang3@gzu.edu.cn (L.Y.)

Abstract: Airport gate assignment is a critical issue in airport operations management. However, limited airport parking spaces and rising fuel costs have caused serious issues with gate assignment. In this paper, an effective multiobjective optimisation model for gate assignment is proposed, with the optimisation objectives of minimising real-time flight conflicts, maximising the boarding bridge rate, and minimising aircraft taxiing fuel consumption. An improved tunicate swarm algorithm based on cosine mutation and adaptive grouping (CG-TSA) is proposed to solve the airport gate assignment problem. First, the Halton sequence is used to initialise the agent positions to improve the initial traversal and allocation efficiency of the algorithm. Second, the population as a whole is adaptively divided into dominant and inferior groups based on fitness values. To improve the searchability of the TSA for the dominant group, an arithmetic optimisation strategy based on ideas related to the arithmetic optimisation algorithm (AOA) is proposed. For the inferior group, the global optimal solution is used to guide the update to improve the convergence speed of the algorithm. Finally, the cosine mutation strategy is introduced to perturb the optimal solution and prevent the target from falling into the local extrema as a way to efficiently and reasonably allocate airport gates. The CG-TSA is validated using benchmark test functions, Wilcoxon rank-sum detection, and CEC2017 complex test functions and the results show that the improved algorithm has good optimality-seeking ability and shows high robustness in the multiobjective optimisation problem of airport gate assignment.

Keywords: airport gate assignment; tunicate swarm algorithm; Halton sequence; arithmetic optimisation strategy; adaptive grouping; cosine mutation strategy

Citation: Zhang, Y.; He, Q.; Yang, L.; Liu, C. An Improved Tunicate Swarm Algorithm for Solving the MultiObjective Optimisation Problem of Airport Gate Assignments. *Appl. Sci.* **2022**, *12*, 8203. https://doi.org/10.3390/app12168203

Academic Editors: Jamie W.G. Turner, Giovanni Vorraro, Hui Liu, Toby Rockstroh and Giancarlo Mauri

Received: 8 July 2022
Accepted: 14 August 2022
Published: 17 August 2022

1. Introduction

The growing size and population of cities with limited resources and services for healthcare, education, the environment, and transportation will make the development of smart cities even more difficult. Smart transportation contributes to the design of smart cities that meet user needs in terms of transportation network efficiency and social sustainability. In this paper, improving urban transportation services is the main goal and the intelligent management of public transportation is the fundamental aspect. With the growing demand for air transportation and the rapid increase in the number of flights, the limited airport resources are already overloaded. Airport parking space is an essential resource for airports and is one of the most critical factors for achieving the fast and safe stopping of flights, ensuring the compelling connection between flights, and improving the operational efficiency and resource use efficiency of the whole airport [1]. There are two methods for alleviating the strain on airport parking resources. One approach is to begin directly on the hardware side by expanding infrastructure such as parking spaces, corridor bridges, and airport aprons. However, increasing hardware equipment and expanding airports will necessitate significant investment in terms of money, time, workforce, and large-scale planning, which will be a long-term strategic consideration. The second goal

is to optimise the software aspect, allocate gate resources more efficiently, and improve airport gate utilisation efficiency. In comparison to hardware expansion, the latter is more appropriate for solving problems such as insufficient parking spaces, with low investment costs, low risk, and quick results. As a result, studying the optimal configuration of airport gates is extremely practical.

The Airport Gate Assignment Problem (AGAP) [2] is a multiobjective optimisation problem that involves allocating resources while taking into account the various stakeholders and the opposing goals for the same stakeholder. Its goal is to distribute a large number of flights among a small number of gates in order to minimise the amount of time passengers must spend walking to their gates. Airport operators strive to decrease resource requirements, resource congestion, disturbances, and delays while also maximising the effectiveness of airport resources. Therefore, this paper provides a review of the literature from two main perspectives: that of the passenger and that of airport operations. According to the first perspective, Bihr [3] proposed a 0–1 linear planning method to minimise the passenger travel distance by reducing the distance passengers walk from one gate to another. Although some practical ideas are provided in a dynamic airport environment, the solutions are idealised. Xu et al. [4] proposed a simple tabu search metaheuristic to minimise the time for passengers to change flights. Drexl et al. [5] proposed a Pareto-simulated annealing strategy to reduce the number of flights and passengers' walking distances. Xie et al. [6] proposed an improved simulated annealing algorithm based on a cluster search, introducing the neighbourhood construction of the variable neighbourhood search idea to achieve a 0–1 integer planning model for aircraft gate assignment while minimising the number of gates used and taking transit passengers' transfer tensions into account. Yuan et al. [7] proposed a hybrid particle swarm algorithm based on the combination of a genetic algorithm and a PSO algorithm to achieve the minimisation of passenger walking distance and parking space idle time equalisation. Dell et al. [8] proposed a new fuzzy swarm optimisation algorithm to reduce total passenger walking distance. CeCen [9] proposed mixed integer linear programming (MILP) and the simulated annealing (SA) algorithm to reduce the total walking distance of passengers from the gate to the baggage conveyor as well as the total fuel consumption of the aircraft taxiing from the gate to the baggage conveyor. From the perspective of airport operators, Yan et al. [10] proposed a heuristic and a simulation framework to assist airport gates in the gate assignment for delayed flights. Hassounah et al. [11] investigated random flight delays to improve static gate assignment performance. Yan et al. [12] used a fixed buffer time between two consecutive flights assigned to the same gate to minimise random flight delays. Dorndorf et al. [13] proposed a recovery planning procedure to maximise the total aircraft assignment preference while minimising the number of constrained expected gate stops. Zhang et al. [14] proposed a biogeography-based optimisation algorithm and a new method for estimating conflict probability to solve a multiobjective gate assignment model that minimises the flight conflict probability and the number of flights allocated to the ramp, thereby improving gate assignment robustness.

Of the types of methods, mathematical planning methods are popular: these systems assign flights to gates based on specific rules. Cheng [15] proposed an airport gate assignment system based on mathematical planning techniques, resulting in an assignment scheme that satisfies both static and dynamic cases at a reasonable computation time. Jaehn [16] proposed a dynamic planning scheme with the sole goal of maximising the aircraft/gate preference fraction in order to solve the gate assignment problem in linear time relative to the number of flights. Yan et al. [17] developed a network model and proposed a Lagrangian relaxation-based subgradient method algorithm, shortest path algorithm, and Lagrangian heuristic algorithm to help airports allocate gates efficiently.

Another class of methods can be broadly categorised as "heuristic" methods. Bi et al. [18] proposed a label search-based algorithm to create a unique neighbourhood structure and tabu search strategy for effectively and rationally allocating gates in order to reduce airport operation costs and improve passenger satisfaction. Hu et al. [19] constructed chromosomes

using relative aircraft positions rather than absolute aircraft positions in the gate queue and proposed an infeasibility-free genetic algorithm for multiobjective gaps. Asadi et al. [20] proposed a hybrid metaheuristic algorithm based on the shuffle frog leap algorithm (SFLA) and grasshopper optimisation algorithm (GOA) to solve the problem of joint aircraft turnaround time and airport gate assignment. Deng et al. [21] proposed an improved ant colony optimisation algorithm that used a collaborative ant colony strategy and a pheromone update strategy to quickly achieve a reasonable and efficient airport gate assignment.

Kaur et al. [22] proposed a new tunicate swarm algorithm (TSA) in 2020, which was inspired by tunicate swarm behaviour. Tunicates are the only animals that move through the ocean using fluid jets for propulsion and they survive by using jet propulsion and swarm intelligence. The TSA is easier to implement than other optimisation algorithms, has better average robustness and global search capabilities, and has been successfully applied in a variety of fields. For example, Li et al. [23] proposed a tent mapping-based tunicate swarm optimisation algorithm that generates initialisation using tent mapping and uses the grey wolf algorithm to generate global search vectors to improve the algorithm's global exploration capability. The Levy flight is also introduced to broaden the search range and provide an optimal economic and environmental dynamic scheduling scheme. Houssein et al. [24] proposed a TSA based on local search improvements and introduced a local search algorithm (LEO) [25] to improve the TSA's convergence speed and local search efficiency, which they used for global optimisation and image segmentation. Sharma et al. [26] used a TSA to implement a local search algorithm to estimate the optimised values of unknown PV cell parameters at standard temperatures. It has been discovered through the study of TSA that using a TSA's global search capability can avoid the problem of falling into local optimal solutions in the process of finding the optimal solution for traditional multiobjective optimisation problems and can keep the diversity of individual solutions.

In summary, the decision to set walking distance objectives for passengers has increased in popularity in recent years, driven by the significance of robust gate schedules for airports and airlines and a growing number of robustness-oriented objectives. Few instances have addressed the requirement for airport operators to simultaneously pursue reduced fuel costs and shorter passenger walking distances. Therefore, in this paper, the AGAP is essentially divided into four objectives: minimising real-time flight conflicts, maximising the boarding bridge rate, minimising the fuel used by aircraft during taxiing, and simultaneously addressing the overall goal of increasing the boarding bridge rate and minimising the fuel used by aircraft. By increasing the effectiveness of airport operations, the pressure on airport fuel prices will be reduced. A survey of the literature on airport gate assignment [27] issues revealed that a number of heuristic and metaheuristic methods have been employed to find solutions. Few have, however, suggested multiobjective precise approaches, and some metaheuristics might generate infeasible solutions. Therefore, an improved tunicate swarm algorithm based on cosine mutation and adaptive grouping (CG-TSA) is proposed in this paper. The Halton sequence initialisation algorithm was introduced to allow the TSA to traverse all possible gates within the entire airport during the initial search in order to improve assignment efficiency. A cosine mutation strategy is used for the gate assignment multiobjective problem to prevent the algorithm from being limited to the optimisation performance of one objective at the expense of the solvency of the others and to prevent the single-objective optimisation from falling into the local extrema, thereby improving the robustness of the gate multiobjective problem solution. The proposed CG-TSA generates partially feasible solutions and improves them during the iterative process. The efficiency of the optimisation procedure can thus be improved.

The remainder of the paper is organised as follows. A multiobjective optimisation model for gate assignments is developed in Section 2. In Section 3, an improved TSA based on cosine mutation and adaptive grouping is proposed. Section 4 evaluates the performance of CG-TSA using the benchmark test function, the benchmark test function Wilcoxon rank-sum detection, and the CEC2017 test function. Section 5 presents the application of CG-TSA to the gate assignment multiobjective optimisation problem and simulates and analyses the

data, and these results are compared with the TSA and GA. The conclusion of this work is provided in Section 6.

2. Optimisation Modelling of Gate Assignments

2.1. Description of the Gate Assignment Problem

Airport gate assignments can be subject to a variety of conditions. To achieve fast, safe, and effective flight docking, reasonable gate assignments play a vital role. The following is a detailed analysis of the gate and flight characteristics. For gates, there are three important features as follows.

1. Matching characteristics of gates

Gates can be divided into three types according to size: large, medium, and small. Large gates can park large and small aircraft, medium gates can park small and medium aircraft, and small gates can only park small aircraft.

2. Parking uniqueness

Only one aircraft can be assigned to the same gate at any one time during a time slot.

3. Auxiliary facilities

Due to geographical and land constraints, an airport has a limited number of boarding bridges. In addition to the bridges, passengers can board from the apron by transfer bus.

For the landing characteristics of flights, the following types of characteristics exist.

1. Flight No.

To facilitate the organisation of transport production and differentiate management, each flight is coded with different numbers according to certain rules.

2. Flight types

This refers to the types of aircraft owned by the airlines. The aircraft type is determined by the number of seats and the layout of the cabin. Different performance variables (such as range, fuel consumption, etc.) mean the aircraft will vary in fuselage size and operating costs. Common flight models are Airbus 320, Boeing 737, Boeing 767, etc.

3. Dynamic characteristics

Each flight has an arrival time and a departure time, with prearranged runways for take-off and landing.

2.2. Determine the Objective Function

Airports, passengers, and airlines all have a stake in the allocation of airline slots. The preassignment of airline slots should take into account flight delays, weather effects, and other disruptions from the perspective of airport operations management. To improve the robustness of the preallocation, the gate assignment model must be solved with the ability to handle dynamic changes, minimise the number of idle slots, and minimise the number of conflicts between real-time operating flights. As a result, minimising real-time flight conflicts is chosen as an optimisation goal. Because the number of airport boarding bridges is limited, as the number of flights increases, some flights will be assigned to more distant aprons. To access the boarding area, passengers must take a bus. This causes inconvenience to passengers while also increasing the workload for dispatchers. As a consequence, optimising flight occupancy will reduce wasted boarding bridge slots and improve passenger satisfaction. Furthermore, due to the promotion of "green transportation" and the ongoing rise in international oil prices, airlines have faced pressures due to fuel consumption costs in recent years. The cost of aircraft fuel accounts for approximately 15.5 percent of all airline operating costs [28]. As a result, reducing fuel consumption and lowering transportation costs have been critical issues for airlines to address. The choice of gate directly affects the taxiing costs of the aircraft from the landing runway to the gate in the gate assignment problem so minimising fuel consumption is used as an optimisation objective. At the same

time, although the aircraft typically wants to arrive at the assigned gate using the least amount of fuel, the gate reached at this point may be a more distant gate. As a result, the overall goal is to maximise the boarding bridge rate while minimising aircraft taxiing fuel consumption. In summary, the detailed description of the identified optimisation objectives is as follows.

1. Minimise real-time flight conflicts

In this paper, the multiobjective function optimisation uses the linear weighting method. Minimising the number of real-time running flight conflicts during gate preallocation can improve the flight punctuality rate. The mathematical model of the conflicting objective function is shown below.

$$F_1 = \min(\sum_{k\in N}\sum_{u\in M} x_{ku} + \sum_{k\in N}\sum_{u\in M} \psi_{ku}) \tag{1}$$

where N is the set of flights, M is the set of aircraft slots, and x_{ku} is a variable taken as 0 or 1. $x_{ku} = 1$ if flight k is assigned to slot u and $x_{ku} = 0$ otherwise. ψ_{ku} is the penalty factor when a flight is assigned to a distant slot and is 0 if assigned to a near slot and 1 when assigned to a distant slot.

The robustness of the preassignment scheme can be improved by minimising the number of conflicts in real-time flights, which can equalise the efficiency of the utilisation of aircraft resources, and by improving the normalisation rate of flight releases in terms of minimising idle time. The minimisation of the slot idle time objective function is shown as follows.

$$\bar{t}_k = \frac{\sum_{k=1}^{N} t_k}{m} \tag{2}$$

$$\min F_2 = \frac{\sum_{k=1}^{N} (t_k - \bar{t}_k)^2}{m} \tag{3}$$

where t_k denotes the slot idle time between flight k and flight $k-1$. The dataset in this paper contains m flights and assuming that the slot is closed after the arrival of the last flight, there are a total of m slot idle times.

2. Maximise the boarding bridge rate

The objective function for maximising the number of flight stops at the boarding bridge is shown below.

$$F_3 = \max \sum_{k\in N}\sum_{u\in M} x_{ku} p_u \tag{4}$$

where p_u is a variable that takes the value of 0 or 1. It takes the value of 1 when the flight is assigned to a near gate and 0 when it is assigned to a far gate.

3. Minimise fuel consumption

The minimised fuel consumption objective function for a flight landing on the runway and taxiing from the runway to the preassigned gate is shown below.

$$F_4 = \min\left[C \cdot \frac{\sum_{k\in N}\sum_{u\in M} x_{ku} o_k d_{ku}}{v}\right] \tag{5}$$

where C is the unit cost of aircraft fuel, o_k is the fuel consumption of flight k during taxiing (fuel consumption is related to the type of aircraft), d_{ku} is the distance from aircraft k landing on the runway to gate u, and v is the average taxiing speed of the aircraft on the runway.

4. Overall target model

Although the aircraft wants to arrive at the assigned gate using the least amount of fuel, the gate reached at this time may be a more distant gate. As a result, to maximise the flight-to-bridge rate and the lowest fuel consumption, nonlinear normalisation must be used to compute the fast assignment of each aircraft from the landed runway to a free near-boarding bridge with relatively low fuel consumption. The objective function of the total objective model is as follows.

$$fit = \max(\left[\text{atan}(F_3) \cdot \frac{2}{\pi}, \text{atan}(F_4) \cdot \frac{2}{\pi}\right]) \tag{6}$$

$$F_5 = \min(\omega_1 \cdot \text{atan}(F_3) \cdot \frac{2}{\pi \cdot fit} + \omega_2 \cdot \text{atan}(F_4) \cdot \frac{2}{\pi \cdot fit}) \tag{7}$$

where ω_1 and ω_2 are the weight coefficients, which take the value of $\omega_1 = \omega_2 = 0.5$ in this paper.

2.3. Constraints

The related constraints are illustrated as follows:

$$\sum_{u \in M} x_{ku} = 1, \forall k \in N \tag{8}$$

$$f_q \leq g_l + (1 - x_{ku})\delta, \forall k \in N, u \in M, l \in M, \delta \in R^+ \tag{9}$$

$$\Delta t_{\text{ku}} = a_k + x_{ku} \cdot (\frac{d_{iu}}{v}), \forall k \in N, u \in M, i = 1, 2, 3 \tag{10}$$

$$\left|A_k - D_j\right| \geq T, \forall k, j \in N \tag{11}$$

Equation (8) maintains that only one aircraft can be assigned to a gate at any given moment. Equation (9) assures that the aircraft receives service from the gate according to the flight type, where f_q is the aircraft type and g_l is the large gate, i.e., the type of aircraft that can accommodate the largest size. Equation (10) calculates the entrance and exit times for gate u and aircraft k using the taxiing distance and arrival time, where Δt_{ku} is expressed as the entry time of aircraft k to gate u, a_k is the expected arrival time of aircraft k, and d_{ku} is the distance from aircraft k landing on the runway to gate u. Equation (11) means that the time interval between two adjacent flights at the same gate must be greater than the safety interval time, where A_k is the scheduled arrival moment of aircraft k, D_j is the scheduled departure moment of flight j, and T is the safety interval time.

3. An Improved TSA Based on Cosine Mutation and Adaptive Grouping

3.1. Tunicate Swarm Algorithm (TSA)

Tunicates are bright bio-luminescent animals and produce a pale blue-green light. Each tunicate draws water from the surrounding sea and generates jet propulsion via atrial syphons at its open end. Tunicates are the only animals in the ocean that move by using fluid jets for propulsion and they survive by using jet propulsion and group behaviour. The TSA primarily simulates two tunicate behaviours: jet propulsion and swarm intelligence. Jet propulsion behaviour relies primarily on gravity, the advection of deep-sea currents, and the interaction forces between individuals to achieve conflict avoidance between individual searches. For jet propulsion behaviour, the tunicate must meet three conditions. Before jet propulsion, the first condition is to avoid conflicts between the searching agents. The second requirement is that the searching agents move towards the best searching agent. The third requirement is to keep a safe distance from the best searching agent. Swarm intelligence is concerned with updating the location of the search agent's optimal solution, determining the location of companions through changes in the neural perception of water flow around the tunicate and companion luminescence, and cooperating to congregate towards the location of the target food source for group foraging. The TSA-specific implementation principle is illustrated below.

3.1.1. Jet Propulsion Behaviour

Before performing the jet propulsion behaviour, the tunicate needs to avoid conflicts with other searching agents using the vector A to calculate the new position, then

$$A = \frac{G}{M} \tag{12}$$

$$G = c_1 + c_2 - F \tag{13}$$

$$F = 2 \cdot c_3 \tag{14}$$

where G denotes gravity, c_1, c_2, and c_3 are random numbers taking the values [0, 1], and F denotes the velocity of the current in the deep sea. M denotes the social force between the searching agents, and its mathematical model description is shown.

$$M = \lfloor Pos_{\min} + c_3(Pos_{\max} - Pos_{\min}) \rfloor \tag{15}$$

where $Pos_{\min}$ and $Pos_{\max}$ are the initial and slave velocities of the attraction, respectively.

After avoiding the conflict between neighbours, the search agents move in the direction of the best neighbour.

$$dpos_t = |P_{best} - rand \cdot x_t| \tag{16}$$

where $dpos_t$ is the distance between the search agents and the food source. P_{best} denotes the location of the food source, i.e., the optimal location. x_t denotes the current location of the search agents and *rand* is a random number in the range [0, 1].

Thereafter, the search agent moves towards the best search agent. The mathematical model of its movement is shown below.

$$x_t = \begin{cases} P_{best}^t + A \cdot d\mathrm{pos_t}, rand \geq 0.5 \\ P_{best}^t - A \cdot d\mathrm{pos_t}, rand \leq 0.5 \end{cases} \tag{17}$$

where P_{best}^t is the position of the food source at the tth iteration, i.e., the optimal position at the tth iteration.

3.1.2. Swarm Intelligence

The swarm intelligence behaviour of the tunicate is to update the other search agents by the optimal search agent positions and to continuously update the positions of the other search agents by the positions of the first two best search agents. Its mathematical model is as follows.

$$x_{t+1} = \frac{x_t + x_{t-1}}{2 + c_1} \tag{18}$$

where x_{t-1} denotes the position of the closest food source of the previous generation of the tunicate. c_1 is a random number taking the value [0, 1].

3.2. Ideas and Strategies for Improved Adaptive TSA

3.2.1. Halton Sequence Initialisation

The tunicate is not socially organised while searching for food. The tunicate is centred on its current position and searches randomly within a sector constructed by the difference between the current optimal position (food source position) and its current position. The TSA uses random population initialisation, resulting in highly random populations. However, when compared to the uniform method, random initialisation causes initial population aggregation and overlap, resulting in a slow population search and insufficient algorithmic diversity. This paper introduces the Halton sequence to initialise the population in order for the TSA to traverse all possible gates within the entire airport during the initial search. The Halton sequence [29] is a low-difference sequence sampling method with good uniform distribution properties, allowing the initialised agents to be distributed evenly across the search space. In the literature [30], the uniformity of sequences generated by

Halton sequences and Rand functions was compared, and the algorithm using Halton sequence initialisation had a more stable convergence time and better traversal.

Halton sequence sampling belongs to the extension of the proposed Monte Carlo sequence, whose main idea is to select a prime as the base and then continuously slice it to form a series of uniform and nonrepeating points, each with coordinates between 0 and 1. Its mathematical formula is expressed as follows.

$$n = \sum_{i=0}^{j} p_i \cdot q^i = p_j q^j + p_{j-1} q^{j-1} + \cdots + p_0 \tag{19}$$

where $n \in [1, N]$ is an arbitrary integer, $p \geq 2$, is prime, $p \in \{0, 1, 2 \ldots, q-1\}$ is a constant, and q is the basic quantity of the Halton sequence. Define the sequence function $\alpha(n)$ as follows.

$$\alpha(n) = p_0 q^{-1} + p_1 q^{-2} + \cdots + p_j q^{-j-1} \tag{20}$$

The final two-dimensional uniform Halton sequence $H(n)$ can be obtained as follows.

$$H(n) = [\alpha_1(n), \alpha_2(n)] \tag{21}$$

The distribution of agents initialised with the random population generated by the basic TSA is given in Figure 1. Figure 2 shows the distribution of the population after the initialisation using the Halton sequence, where the base quantities are taken as 2 and 3.

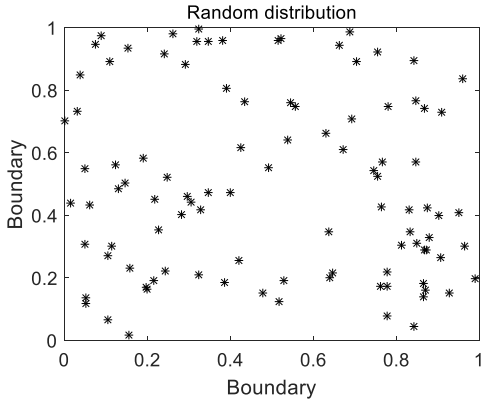


Figure 1. Random distribution map.

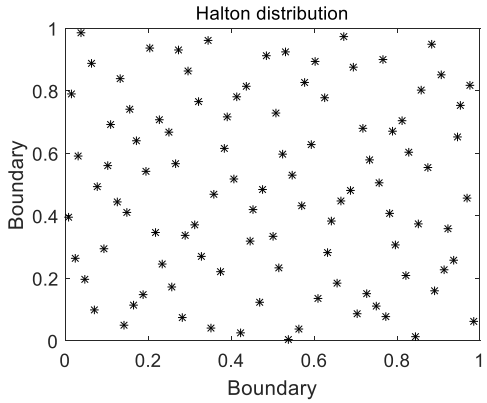


Figure 2. Distribution map of the Halton sequence.

From the comparison of Figures 1 and 2, the population distribution obtained by the Halton sequence is more uniform and there is no overlapping of the agents. Thus, the initialisation with the Halton sequence can make the population quality higher and thus improve the diversity of the algorithm.

3.2.2. TSA Adaptive Grouping Strategy

The TSA is a group intelligence search algorithm based on the habitat of jet propulsion and swarm intelligence. The jet propulsion phase corresponds to the global development of the algorithm, whereas the swarm intelligence phase corresponds to the algorithm's local search. Because of the different positions that the tunicates are in, there is a possibility that these agents are attracted to the local optimal solution and are prone to falling into the local optimal solution as the number of iterations increases. To avoid this possibility, an adaptive grouping strategy is proposed, where agents are divided into two groups according to their fitness values from largest to smallest, with the first half of the agents being the dominant group of the tunicate and the remaining individuals being the inferior group of the tunicate. For the group with the lower fitness values, that is, the group dominated by saccades, indicating a better position in the population, it is necessary to enhance their search ability and strengthen the information exchangeability among the agents at this time. Therefore, in this paper, inspired by the position updating method of the arithmetic optimisation algorithm, an arithmetic optimisation strategy is proposed to combine the TSA's jet propulsion behaviour and swarm intelligence behaviour for one-to-one optimisation. The group with the higher fitness values, which is the inferior saccade group, is in a worse position. Hence, the global optimal solution is adopted to guide the agent position update as a way to speed up the convergence of the algorithm. At the same time, the ability of the algorithm to jump out of the local optimum is improved.

Algorithmic Optimisation Strategy

The dominant group of tunicate swarms employs arithmetic optimisation strategies for updating position and improving interindividual information transfer. The AOA is a metaheuristic algorithm proposed in 2021 by Abualigah et al. [31] that primarily uses the calculation of arithmetic operators in mathematics, such as multiplication and division operators, as well as addition and subtraction operators. The specific implementation process is divided into two phases: exploration and development.

In the exploration phase, the AOA uses highly decentralised multiplication or division operators to extensively explore the search space and find a better position. The mathematical model of the AOA exploration phase implementation is described as follows.

$$x_{t+1}^{i} = \begin{cases} P_{best}^{i} \div (MOP + \alpha) \cdot (\lambda(UB_i - LB_i) + LB_i), r_2 < 0.5 \\ P_{best}^{i} \times MOP \cdot ((UB_i - LB_i) \times \lambda + LB_i), otherwise \end{cases} \tag{22}$$

where x_{t+1} denotes the agent solution after the next iteration, P_{best} denotes the optimal solution of the current iteration, UB_i and LB_i denote the upper and lower bounds of the ith position, respectively, α is the defined integer, λ is the control parameter to adjust the search process, and the value of λ in the basic AOA is 0.5. MOP is the probability function, where the mathematical model is described as follows.

$$MOP = 1 - \frac{t^{\varepsilon}}{T_{\max}^{\varepsilon}} \tag{23}$$

where ε = 5, a sensitivity factor, defines the search accuracy of the iterations.

During the development phase, AOA are developed using high-density additive or subtractive arithmetic. The additive and subtractive operators can easily approach the target due to their low spatial span. Therefore, it is easier to derive the optimal solution after several iterations. The mathematical model of the AOA development process implementation is as follows.

$$x_{t+1}^{i} = \begin{cases} P_{best}^{i} - MOP \times ((UB_i - LB_i) \times \lambda + LB_i), r_3 < 0.5 \\ P_{best}^{i} + MOP \times ((UB_i - LB_i) \times \lambda + LB_i), otherwise \end{cases} \tag{24}$$

where r_3 takes the value [0, 1] as a random number.

In the exploration and exploitation phases, the arithmetic optimisation algorithm performs position updating using operators with different characteristics for different states of the population. This process not only aids in the discovery of the optimal solution during the exploration phase but also preserves the diversity of other candidate optimal solutions. The TSA is a random search based on jet propulsion and swarms intelligence behaviour centred on agent tunicates and because of this single search mode, the TSA lacks dynamism, easily falls into a local optimum, and has low solution accuracy. Therefore, the AOA multiplication and division operator is introduced in the TSA jet propulsion behaviour phase to assist the TSA in increasing the search range, improving the mining ability of the TSA and the global search ability during TSA jet propulsion. The AOA multiplication and division operator is introduced in the group behaviour phase of the TSA to target the algorithm's local exploitation ability and improve the algorithm's overall search accuracy and convergence speed.

Global Optimal Solution Guidance Strategy

For the inferior group of the tunicates with relatively poor exploration and exploitation abilities, the global optimal solution guidance strategy is used. By keeping the current solution away from a randomly selected candidate solution from the agents, the guidance-based global optimal solution is generated, and the captive group's search opportunity for the optimal solution in the neighbourhood is increased, speeding up the algorithm's convergence. The formula for updating the inferior group position after the introduction of the global optimal solution guidance strategy is shown below.

$$x_{t+1} = x_t + rand \times (P_{best}^t - x_t) \tag{25}$$

where x_t is the current position of the capsule agent and P_{best} is the optimal position of the current capsule agent, which is the global optimal position. The change in the number of iterations can guide the capsule group to perform a locality search, which is beneficial for increasing the algorithm diversity and improving the convergence accuracy.

3.2.3. Cosine Mutation Strategy

This paper, which was inspired by the literature [32], proposes a sinusoidal mutation strategy based on the cosine function to guide the current optimal agent to make a local jump in order to improve the ability of the agents after TSA grouping to jump out of the local optimal solution at a later stage. When dealing with the boarding gate assignment problem, it also prevents the TSA from being limited to the optimisation performance of one objective while ignoring the solving ability of the other objectives and it moves closer to the theoretical optimum. For the cosine function, its value on the range interval $[0, 2\pi]$ is $[-1, 1]$, which can make the grouped algorithm perform a small search for the current global optimum position. Thus, it can make the algorithm tend to search better for the optimum in a certain search space. The TSA position update formula with the introduction of the cosine mutation is as follows.

$$P_{best}^{t+1} = \cos(2\pi \times rand(1,30)) \cdot P_{best}^t \tag{26}$$

$$P_{best}^t = \begin{cases} P_{best}^{t+1}, if(P_{best}^t) < T \\ P_{best}^t, otherwise \end{cases} \tag{27}$$

where P_{best} is the global optimal position as shown in Equation (26) and Equation (27). If the adaptation of the agent after cosine sinusoid is better, the mutation result is selected. Conversely, the original position is chosen to be kept.

3.3. CG-TSA Implementation Process

The flow chart of the CG-TSA implementation is shown below, see Figure 3.

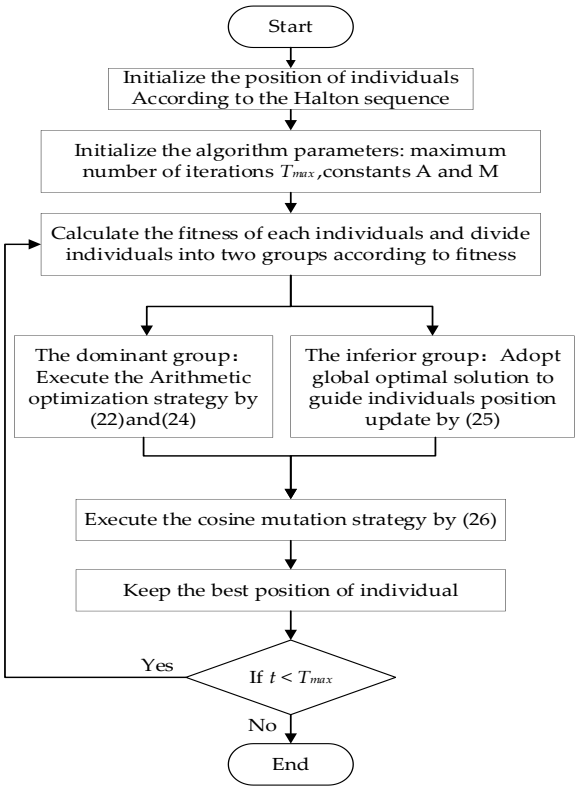


Figure 3. Flowchart of CG-TSA.

4. Simulation Experiment Analysis

This section describes the test functions used to evaluate the performance of the CG-TSA. The Whale Optimisation Algorithm (WOA) [33], Grey Wolf Optimiser (GWO) [34], Sine Cosine Algorithm (SCA) [35], AOA, and the basic TSA were selected for the function-finding comparison for the CG-TSA. To ensure the fairness of the comparison between the algorithms, the basic parameters of the algorithms were set to the same values including the population size $N = 30$, the maximum number of iterations $T_{\max} = 500$, and the dimensionality $d = 30/500/1000$. The internal parameters of each algorithm were set as shown in Table 1. The experiments were conducted with 10 benchmark test functions for the CG-TSA to compare the search performance, including single-peak test functions F1–F6 and complex multipeak test functions F7–F10, and the description of the benchmark test function information is shown in Table 2. The experimental running environment was Windows 10, the CPU was a 3.4 GHz Intel Core i7-6500U with 8 GB of memory and a 64-bit OS, and the model was built on MATLAB R2021(a).

Table 1. Algorithm parameter setting.

Algorithm	Parameter
WOA	\
GWO	\
SCA	\
AOA	$MOP\_Max = 1$, $MOP\_Min = 0.2$, $\alpha = 5$, $\mu = 0.49$
TSA	$P_{\max} = 4$, $P_{\min} = 1$
CG-TSA	$P_{\max} = 4$, $P_{\min} = 1$

Table 2. Introduction to benchmark functions.

Fun No.	Name	Range	Dim	Optimal Value
F1	Sphere	[−100, 100]	30/500/1000	0
F2	Schwefel.2.22	[−10, 10]	30/500/1000	0
F3	Schwefel.1.2	[−100, 100]	30/500/1000	0
F4	Schwefel.2.21	[−100, 100]	30/500/1000	0
F5	Step Function	[−100, 100]	30/500/1000	0
F6	Quartic Function	[−1.28, 1.28]	30/500/1000	0
F7	Rastrigin	[−5.12, 5.12]	30/500/1000	0
F8	Ackley	[−32, 32]	30/500/1000	0
F9	Criewank	[−600, 600]	30/500/1000	0
F10	Penalized 1	[−50, 50]	30/500/1000	0

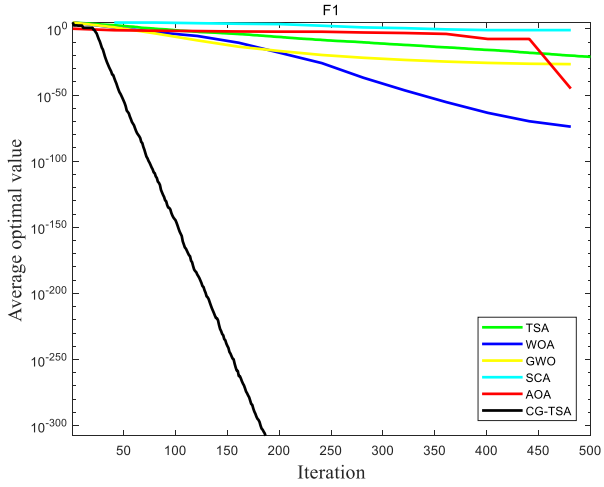
4.1. Comparative Analysis of CG-TSA Benchmarking Function Search

To verify the advantages of the CG-TSA for benchmark test function seeking and to analyse the simulation experimental results more intuitively, the WOA, GWO algorithm, SCA, AOA, and basic TSA were selected for the benchmarking function finding performance comparison experiments with the CG-TSA. Each algorithm was run 30 times independently. Figure 4 shows the average convergence curves of some of the benchmark functions. The horizontal coordinate is the number of iterations, and the vertical coordinate is the fitness value.

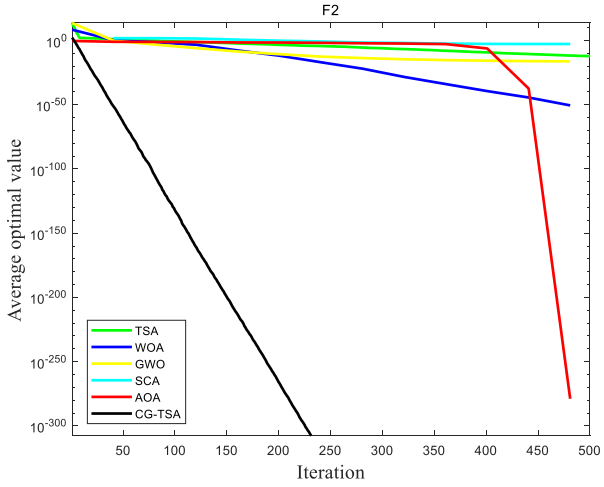
As seen in Figure 4, the CG-TSA achieved theoretical optima for both unimodal and complex multimodal functions. In particular, for the unimodal F3 function, the CG-TSA expanded the search range in the early stage and accelerated the convergence of the algorithm in later stages, indicating that the CG-TSA introducing the Halton sequence had advantages in finding the optimal accuracy of the basic function. For the single-peaked F6 function, the CG-TSA jumped out of the local optimal solution, which indicated that the introduced cosine mutation effectively helped the algorithm to jump out of the local extremum. As seen in the multipeaked F7–F10, the CG-TSA converged to the theoretical optimum quickly compared with the other algorithms under the same number of iterations. Based on the above analysis, the grouping improvement strategy of the cosine mutation introduced in this paper effectively helped the TSA to show obvious advantages in the basic function optimisation and improve the convergence accuracy of the algorithm.

4.2. CG-TSA High-Dimensional Function Finding Performance Comparison Analysis

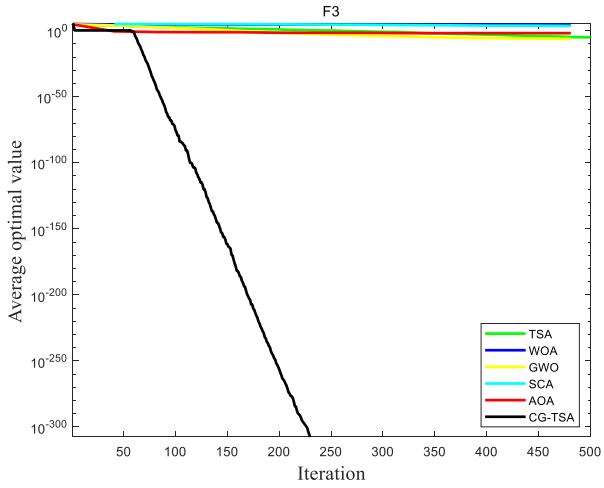
To further validate the processing capability of the CG-TSA for high-dimensional optimisation problems, the basic TSA, WOA, GWO algorithm, and SCA were selected and compared with the CG-TSA for 500-dimensional and 1000-dimensional simulation experiments of the benchmark test functions. For each test function, the dimensionality d = 500/1000, the population size N = 30, and the number of iterations $T_{\max}$ = 500. Each algorithm was run 30 times independently, and the mean and standard deviation were taken to judge the performance of the optimisation algorithm. The optimisation search results of each algorithm for the high-dimensional benchmark test functions are shown in Table 3.



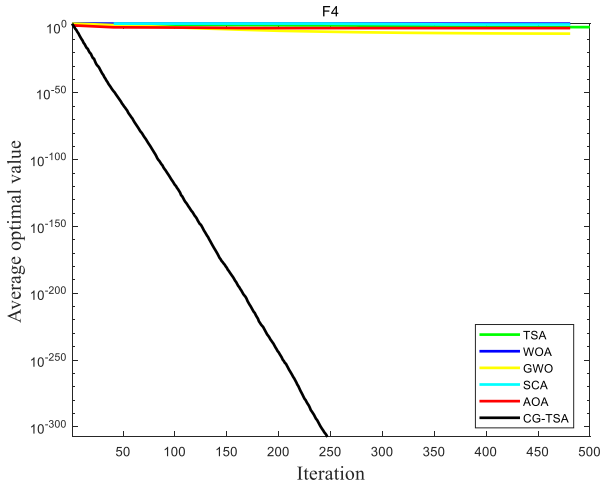
(**a**)



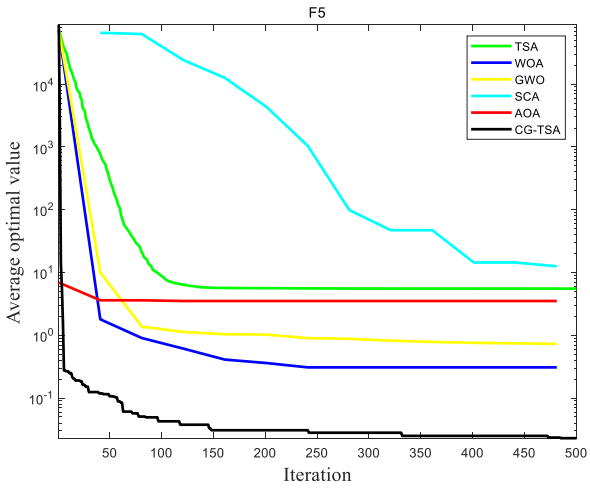
(**b**)



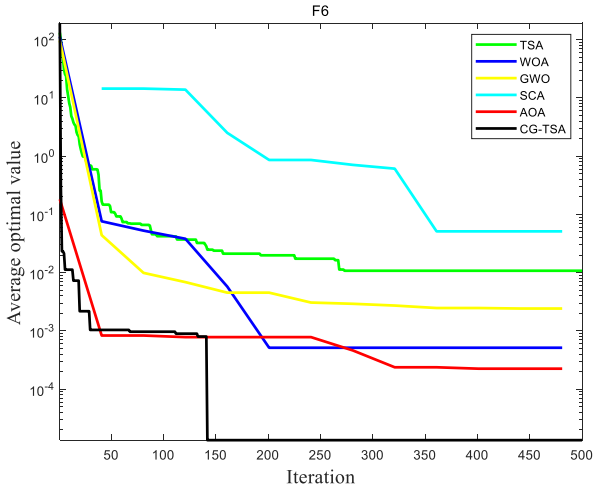
(**c**)



(**d**)



(**e**)



(**f**)

Figure 4. *Cont.*

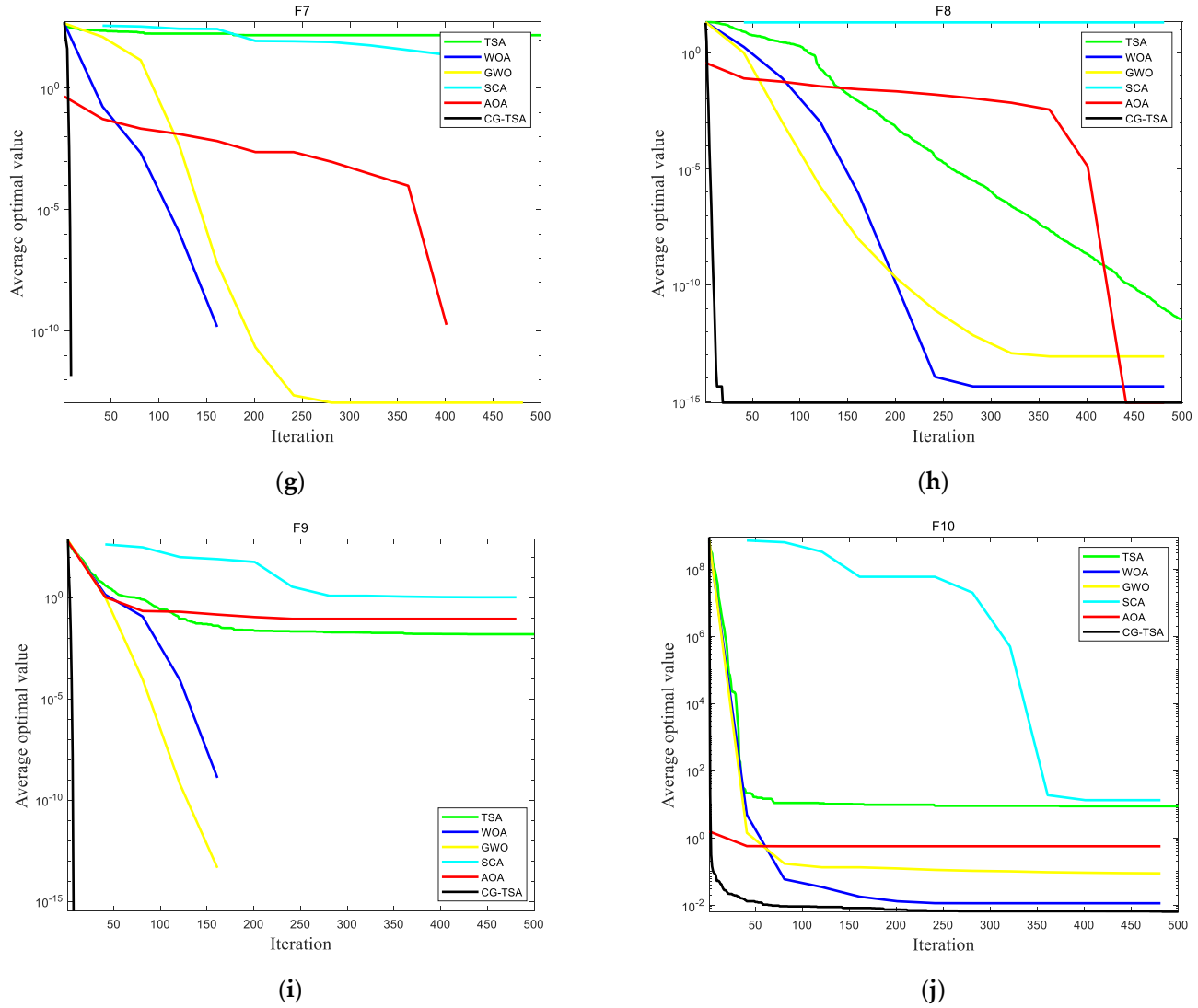


Figure 4. Partial convergence curves of the proposed CG-TSA and other metaheuristic algorithms of the benchmark test functions: (**a**) Convergence curve of F1; (**b**) Convergence curve of F2; (**c**) Convergence curve of F3; (**d**) Convergence curve of F4; (**e**) Convergence curve of F5; (**f**) Convergence curve of F6; (**g**) Convergence curve of F7; (**h**) Convergence curve of F8; (**i**) Convergence curve of F9; (**j**) Convergence curve of F10.

It is clear from the comparison results in Table 3 that the CG-TSA had clear advantages when handling high-dimensional optimisation problems. The CG-TSA results were similar in the 500-dimensional and 1000-dimensional search spaces for the single-peaked test functions F1–F4 and the multipeaked test functions F7 and F8, and globally optimal values were found for F1–F4 and F7. The search results for the other 1000-dimensional test functions were remarkably similar to the data at 500 dimensions. The search results were superior to those of other algorithms, despite the fact that the theoretical optimal value was not found. It can be concluded that the CG-TSA had superior search capability and better robustness than the other algorithms for the search for high-dimensional functions. The proposed CG-TSA is a better optimiser for solving global optimisation problems.

Table 3. Comparison of optimisation results of each algorithm.

Fun No.	Dim	TSA		WOA		GWO		SCA		CG-TSA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F_1	$d = 500$	2.89×10^{-21}	5.33×10^{-21}	1.86×10^{-73}	5.72×10^{-73}	5.54×10^{-28}	6.86×10^{-28}	1.30×10^{1}	2.20×10^{1}	**0**	**0**
	$d = 1000$	7.74×10^{-22}	6.69×10^{-22}	3.73×10^{-73}	1.16×10^{-72}	1.38×10^{-27}	3.34×10^{-27}	1.72×10^{1}	4.13×10^{1}	**0**	**0**
F_2	$d = 500$	1.27×10^{-13}	8.68×10^{-13}	1.69×10^{-52}	2.45×10^{-52}	8.12×10^{-17}	7.31×10^{-17}	2.64×10^{-2}	3.86×10^{-2}	**0**	**0**
	$d = 1000$	7.62×10^{-14}	8.92×10^{-14}	7.32×10^{-52}	1.78×10^{-51}	1.63×10^{-16}	8.99×10^{-17}	1.83×10^{-2}	3.14×10^{-2}	**0**	**0**
F_3	$d = 500$	3.35×10^{-4}	1.01×10^{-3}	3.87×10^{4}	1.16×10^{4}	8.15×10^{-6}	1.55×10^{-5}	9.61×10^{3}	5.84×10^{3}	**0**	**0**
	$d = 1000$	2.59×10^{-5}	3.53×10^{-5}	4.06×10^{4}	9.04×10^{3}	4.14×10^{-6}	5.80×10^{-6}	6.76×10^{3}	7.16×10^{3}	**0**	**0**
F_4	$d = 500$	3.16×10^{-1}	2.66×10^{-1}	5.47×10^{1}	3.61×10^{1}	5.64×10^{-7}	3.76×10^{-7}	3.41×10^{1}	1.30×10^{1}	**0**	**0**
	$d = 1000$	2.77×10^{-1}	2.55×10^{-1}	5.62×10^{1}	2.06×10^{1}	5.96×10^{-7}	3.81×10^{-7}	3.65×10^{1}	1.34×10^{1}	**0**	**0**
F_5	$d = 500$	3.72×10^{0}	4.01×10^{-1}	5.58×10^{-1}	2.93×10^{-1}	6.26×10^{-1}	4.75×10^{-1}	1.50×10^{1}	1.20×10^{1}	$\mathbf{2.01 \times 10^{-2}}$	$\mathbf{3.24 \times 10^{-3}}$
	$d = 1000$	3.95×10^{0}	6.12×10^{-1}	4.10×10^{-1}	2.12×10^{-1}	1.05×10^{0}	3.69×10^{-1}	2.10×10^{1}	1.99×10^{1}	$\mathbf{2.07 \times 10^{-2}}$	$\mathbf{2.56 \times 10^{-3}}$
F_6	$d = 500$	1.19×10^{-2}	5.22×10^{-3}	1.32×10^{-3}	1.48×10^{-3}	2.03×10^{-3}	1.17×10^{-3}	1.16×10^{-1}	5.17×10^{-2}	$\mathbf{5.04 \times 10^{-5}}$	$\mathbf{7.83 \times 10^{-5}}$
	$d = 1000$	8.48×10^{-3}	4.53×10^{-3}	6.08×10^{-3}	4.53×10^{-3}	2.58×10^{-3}	1.18×10^{-3}	1.51×10^{-1}	2.11×10^{-1}	$\mathbf{6.46 \times 10^{-5}}$	$\mathbf{4.73 \times 10^{-5}}$
F_7	$d = 500$	2.00×10^{2}	3.43×10^{1}	**0**	**0**	3.62×10^{0}	3.54×10^{0}	5.57×10^{1}	3.23×10^{0}	**0**	**0**
	$d = 1000$	1.95×10^{2}	4.75×10^{1}	**0**	**0**	2.75×10^{0}	3.82×10^{0}	4.62×10^{1}	4.58×10^{1}	**0**	**0**
F_8	$d = 500$	1.86×10^{0}	1.62×10^{0}	4.08×10^{-15}	4.25×10^{-15}	9.75×10^{-14}	1.18×10^{-14}	1.26×10^{1}	9.77×10^{0}	$\mathbf{8.88 \times 10^{-16}}$	**0**
	$d = 1000$	1.58×10^{0}	1.67×10^{0}	5.50×10^{-15}	2.39×10^{-15}	1.02×10^{-13}	1.83×10^{-14}	1.47×10^{1}	8.80×10^{0}	$\mathbf{8.88 \times 10^{-16}}$	**0**
F_9	$d = 500$	1.16×10^{-2}	9.26×10^{-3}	3.82×10^{-2}	8.07×10^{-2}	3.77×10^{-3}	8.68×10^{-3}	8.85×10^{-1}	3.17×10^{-1}	$\mathbf{1.46 \times 10^{-5}}$	$\mathbf{9.57 \times 10^{-6}}$
	$d = 1000$	6.56×10^{-3}	8.60×10^{-3}	2.46×10^{-2}	7.79×10^{-2}	1.38×10^{-3}	4.37×10^{-3}	7.39×10^{-1}	3.77×10^{-1}	**0**	**0**
F_{10}	$d = 500$	6.81×10^{0}	3.64×10^{0}	2.64×10^{-2}	1.92×10^{-2}	9.72×10^{-1}	1.63×10^{-1}	1.11×10^{4}	3.49×10^{4}	$\mathbf{4.46 \times 10^{-3}}$	$\mathbf{8.43 \times 10^{-3}}$
	$d = 1000$	5.51×10^{0}	4.45×10^{0}	7.22×10^{-1}	1.61×10^{-1}	4.23×10^{0}	2.58×10^{0}	1.02×10^{4}	3.19×10^{4}	$\mathbf{1.21 \times 10^{-2}}$	$\mathbf{1.10 \times 10^{-2}}$

4.3. Wilcoxon Rank-Sum Test

From the analysis of each experiment above, it can be concluded that the CG-TSA performed well in the benchmark test function search. To reflect the algorithm's performance more comprehensively, the CG-TSA was compared with the basic TSA and the other algorithms in terms of the optimisation-seeking effect. Wilcoxon nonparametric statistical tests were performed at the 0.05 significance level.

The results of the CG-TSA run for the 10 benchmark test functions in Table 2 were selected to perform the Wilcoxon rank-sum tests and calculate the p values with the TSA after adding the Halton initialisation (HTSA), TSA after grouping improvements (FTSA), TSA after adding the cosine variation (CTSA), and the results of the GWO algorithm and SCA runs. When $p < 5\%$, it was considered a strong validation of the rejection of the null hypothesis [36]. The experimental results NaN indicated that there were no data for comparison and +, =, and − indicated that the CG-TSA outperformed, equalled, and underperformed the compared algorithms, respectively. The results of the Wilcoxon rank-sum tests are shown in Table 4.

Table 4. Wilcoxon rank-sum test results.

Fun No.	TSA (p_1)	HTSA (p_2)	FTSA (p_3)	CTSA (p_4)	GWO (p_5)	SCA (p_6)
F_1	1.34×10^{-16}	1.34×10^{-16}	3.31×10^{-20}	3.31×10^{-20}	9.91×10^{-16}	7.06×10^{-18}
F_2	3.31×10^{-20}	3.31×10^{-20}	3.31×10^{-20}	NaN	3.31×10^{-20}	3.31×10^{-20}
F_3	4.55×10^{-4}	3.46×10^{-14}	3.06×10^{-10}	3.31×10^{-20}	6.85×10^{-4}	7.06×10^{-18}
F_4	2.94×10^{-13}	7.67×10^{-15}	7.06×10^{-18}	3.31×10^{-20}	7.77×10^{-4}	7.06×10^{-18}
F_5	7.55×10^{-6}	5.37×10^{-10}	7.06×10^{-18}	1.09×10^{-1}	7.50×10^{-18}	7.50×10^{-18}
F_6	7.06×10^{-18}	7.06×10^{-18}	3.83×10^{-1}	2.26×10^{-2}	7.06×10^{-18}	7.06×10^{-18}
F_7	3.31×10^{-20}	3.31×10^{-20}	2.80×10^{-11}	NaN	3.17×10^{-20}	3.31×10^{-20}
F_8	3.31×10^{-20}	2.62×10^{-23}	3.27×10^{-1}	9.46×10^{-9}	2.97×10^{-20}	3.31×10^{-20}
F_9	1.51×10^{-14}	4.07×10^{-15}	3.31×10^{-20}	3.31×10^{-20}	2.33×10^{-17}	7.96×10^{-18}
F_{10}	7.06×10^{-18}	1.34×10^{-16}	7.06×10^{-18}	1.63×10^{-17}	8.00×10^{-17}	7.96×10^{-18}
+/=/−	10/0/0	10/0/0	8/0/2	7/2/1	10/0/0	10/0/0

According to the results of the Wilcoxon rank-sum statistics in Table 4, except for the absence of data comparison, the Wilcoxon rank-sum test p values of the CG-TSA were less than 5%, and the performance of the CG-TSA was better than the various algorithms compared. This shows that the optimal performance advantage of the CG-TSA for basic functions was obvious.

4.4. Experimental Analysis of the CEC2017 Test Function

To verify the processing capability of the CG-TSA for complex feature functions and its applicability to different complex optimisation problems, this study used the CEC2017 test function [37] to test its performance. The CEC2017 test function was introduced as shown in Table 5, where the function types contain Simple Multimodal Functions (SMF), Hybrid Functions (HF), and Composition Functions (CF). In this paper, the proposed CG-TSA was compared with the standard PSO algorithm, GWO, SCA, and TSA for the performance of the search. The experimental parameters were taken as population size N = 100, dimension d = 10, and the maximum number of iterations $T_{\max}$ = 5000, and each function was run 50 times independently to obtain the mean and standard deviation. The comparison of the results of each algorithm run is shown in Table 6.

Table 5. Part of the CEC2017 test function.

Fun No.	Dim	Function Type	Range	Optimal Value
CEC04	10	SMF	[−100, 100]	400
CEC05	10	SMF	[−100, 100]	500
CEC06	10	SMF	[−100, 100]	600
CEC07	10	SMF	[−100, 100]	700
CEC08	10	SMF	[−100, 100]	800
CEC09	10	SMF	[−100, 100]	900
CEC10	10	SMF	[−100, 100]	1000
CEC11	10	HF	[−100, 100]	1100
CEC13	10	HF	[−100, 100]	1300
CEC16	10	HF	[−100, 100]	1600
CEC17	10	HF	[−100, 100]	1700
CEC20	10	HF	[−100, 100]	2000
CEC21	10	CF	[−100, 100]	2100
CEC22	10	CF	[−100, 100]	2200
CEC23	10	CF	[−100, 100]	2300
CEC24	10	CF	[−100, 100]	2400
CEC25	10	CF	[−100, 100]	2500
CEC26	10	CF	[−100, 100]	2600
CEC27	10	CF	[−100, 100]	2700
CEC28	10	CF	[−100, 100]	2800
CEC29	10	CF	[−100, 100]	2900

As seen in Table 6, the standard PSO algorithm performed well on single peaks, but for multipeak, mixed, and composite functions, the CG-TSA had a clear advantage. In particular, for CEC09, CEC11, CEC16, CEC17, and CEC20, the CG-TSA was able to converge to near the theoretical value. The CG-TSA mean was lower and closer to the theoretical value for higher dimensions on CEC23, CEC24, CEC25, CEC26, CEC28, and CEC29 than the other algorithms. Overall, the proposed algorithm had a more prominent advantage in the CEC2017 test function compared to the other four algorithms.

Table 6. CEC2017 function optimisation comparison.

Fun No.	Dim	TSA		GWO		PSO		SCA		CG-TSA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
CEC04	$d = 10$	4.58×10^{2}	6.21×10^{1}	4.77×10^{2}	7.72×10^{1}	4.54×10^{2}	6.86×10^{2}	4.52×10^{2}	5.92×10^{1}	$\mathbf{4.32 \times 10^{2}}$	$\mathbf{4.92 \times 10^{1}}$
	$d = 30$	2.58×10^{3}	1.03×10^{3}	5.63×10^{2}	4.39×10^{1}	$\mathbf{4.76 \times 10^{2}}$	$\mathbf{4.25 \times 10^{0}}$	1.47×10^{3}	2.18×10^{2}	6.58×10^{2}	$\mathbf{3.49 \times 10^{1}}$
	$d = 50$	9.14×10^{3}	1.50×10^{3}	9.70×10^{2}	1.45×10^{2}	5.96×10^{2}	5.16×10^{1}	7.22×10^{3}	8.48×10^{2}	$\mathbf{4.93 \times 10^{2}}$	$\mathbf{3.93 \times 10^{2}}$
CEC05	$d = 10$	5.96×10^{2}	3.52×10^{1}	5.98×10^{2}	3.46×10^{1}	$\mathbf{5.16 \times 10^{2}}$	$\mathbf{4.85 \times 10^{0}}$	5.34×10^{2}	6.49×10^{1}	5.38×10^{2}	1.44×10^{1}
	$d = 30$	8.07×10^{2}	4.62×10^{1}	5.91×10^{2}	1.73×10^{1}	6.07×10^{2}	2.82×10^{0}	7.68×10^{3}	2.56×10^{2}	$\mathbf{5.58 \times 10^{2}}$	$\mathbf{1.91 \times 10^{1}}$
	$d = 50$	1.09×10^{3}	7.24×10^{1}	6.89×10^{2}	2.33×10^{1}	7.02×10^{2}	3.16×10^{1}	1.07×10^{3}	3.02×10^{1}	$\mathbf{6.69 \times 10^{2}}$	$\mathbf{2.94 \times 10^{1}}$
CEC06	$d = 10$	6.31×10^{2}	1.15×10^{1}	6.92×10^{2}	3.47×10^{1}	$\mathbf{6.00 \times 10^{2}}$	$\mathbf{3.71 \times 10^{0}}$	6.49×10^{2}	1.63×10^{1}	6.25×10^{2}	1.19×10^{1}
	$d = 30$	6.70×10^{2}	1.46×10^{1}	6.74×10^{2}	2.26×10^{1}	6.48×10^{2}	5.67×10^{0}	6.78×10^{2}	4.60×10^{2}	$\mathbf{6.24 \times 10^{2}}$	$\mathbf{8.43 \times 10^{0}}$
	$d = 50$	6.82×10^{2}	1.38×10^{1}	6.93×10^{2}	3.00×10^{1}	6.94×10^{2}	5.15×10^{0}	6.79×10^{2}	3.35×10^{0}	$\mathbf{6.75 \times 10^{2}}$	$\mathbf{7.14 \times 10^{0}}$
CEC07	$d = 10$	7.85×10^{2}	2.61×10^{1}	7.94×10^{2}	1.67×10^{1}	7.47×10^{2}	4.69×10^{0}	7.99×10^{2}	2.86×10^{0}	$\mathbf{7.17 \times 10^{2}}$	$\mathbf{1.67 \times 10^{1}}$
	$d = 30$	1.19×10^{3}	7.53×10^{3}	8.43×10^{2}	3.51×10^{1}	8.17×10^{2}	2.88×10^{1}	1.12×10^{4}	3.30×10^{2}	$\mathbf{7.28 \times 10^{2}}$	$\mathbf{2.96 \times 10^{1}}$
	$d = 50$	1.80×10^{3}	1.79×10^{2}	1.03×10^{3}	4.64×10^{1}	1.01×10^{3}	4.71×10^{1}	1.57×10^{3}	4.02×10^{1}	8.75×10^{2}	4.96×10^{1}
CEC08	$d = 10$	8.35×10^{2}	1.30×10^{1}	8.89×10^{2}	5.82×10^{0}	8.89×10^{2}	2.99×10^{-1}	8.53×10^{2}	2.64×10^{0}	$\mathbf{8.29 \times 10^{2}}$	$\mathbf{6.66 \times 10^{1}}$
	$d = 30$	1.06×10^{3}	4.61×10^{3}	8.74×10^{2}	1.16×10^{1}	8.81×10^{2}	1.69×10^{1}	1.04×10^{4}	1.92×10^{2}	$\mathbf{8.66 \times 10^{2}}$	$\mathbf{3.58 \times 10^{1}}$
	$d = 50$	1.44×10^{3}	8.13×10^{2}	9.56×10^{3}	2.38×10^{1}	9.92×10^{2}	1.89×10^{1}	1.34×10^{3}	2.94×10^{1}	$\mathbf{9.11 \times 10^{2}}$	$\mathbf{3.16 \times 10^{1}}$
CEC09	$d = 10$	1.03×10^{3}	1.54×10^{2}	9.87×10^{2}	5.98×10^{0}	9.89×10^{2}	1.99×10^{-3}	9.44×10^{2}	1.64×10^{1}	$\mathbf{9.03 \times 10^{2}}$	$\mathbf{4.91 \times 10^{0}}$
	$d = 30$	9.72×10^{3}	3.43×10^{3}	1.38×10^{4}	3.34×10^{2}	1.83×10^{4}	4.61×10^{2}	5.92×10^{4}	9.48×10^{2}	$\mathbf{1.19 \times 10^{4}}$	$\mathbf{5.43 \times 10^{2}}$
	$d = 50$	3.74×10^{4}	1.17×10^{4}	4.12×10^{3}	2.03×10^{3}	8.06×10^{3}	1.50×10^{3}	2.31×10^{4}	6.03×10^{3}	$\mathbf{1.52 \times 10^{3}}$	$\mathbf{2.59 \times 10^{2}}$
CEC10	$d = 10$	1.85×10^{3}	1.99×10^{2}	2.47×10^{4}	2.67×10^{2}	1.56×10^{3}	2.46×10^{2}	1.98×10^{3}	1.15×10^{2}	$\mathbf{1.41 \times 10^{3}}$	$\mathbf{2.25 \times 10^{2}}$
	$d = 30$	6.71×10^{3}	6.70×10^{2}	4.63×10^{3}	3.34×10^{2}	1.83×10^{3}	4.61×10^{2}	5.92×10^{3}	9.48×10^{2}	$\mathbf{1.19 \times 10^{3}}$	$\mathbf{5.43 \times 10^{2}}$
	$d = 50$	1.37×10^{4}	9.74×10^{4}	6.50×10^{3}	5.93×10^{2}	6.94×10^{3}	9.46×10^{2}	1.49×10^{4}	3.76×10^{2}	$\mathbf{1.33 \times 10^{3}}$	$\mathbf{8.90 \times 10^{2}}$
CEC11	$d = 10$	3.41×10^{3}	4.04×10^{1}	$\mathbf{1.11 \times 10^{3}}$	$\mathbf{9.98 \times 10^{0}}$	1.20×10^{4}	3.65×10^{0}	1.14×10^{3}	7.59×10^{0}	$\mathbf{1.11 \times 10^{3}}$	$\mathbf{4.04 \times 10^{0}}$
	$d = 30$	4.87×10^{3}	1.17×10^{3}	1.36×10^{3}	8.55×10^{2}	1.19×10^{4}	2.83×10^{1}	2.12×10^{3}	2.06×10^{2}	$\mathbf{1.17 \times 10^{3}}$	$\mathbf{9.95 \times 10^{0}}$
	$d = 50$	1.21×10^{4}	2.97×10^{3}	3.59×10^{3}	1.23×10^{3}	1.31×10^{3}	2.41×10^{2}	6.86×10^{3}	8.59×10^{2}	$\mathbf{1.23 \times 10^{3}}$	$\mathbf{4.69 \times 10^{2}}$
CEC13	$d = 10$	5.99×10^{3}	3.74×10^{3}	5.39×10^{3}	2.53×10^{3}	4.21×10^{3}	1.92×10^{3}	6.21×10^{3}	2.80×10^{4}	$\mathbf{1.34 \times 10^{3}}$	$\mathbf{4.04 \times 10^{3}}$
	$d = 30$	1.31×10^{9}	3.81×10^{9}	4.02×10^{5}	1.00×10^{6}	2.20×10^{4}	2.02×10^{4}	4.25×10^{8}	1.40×10^{8}	$\mathbf{1.65 \times 10^{4}}$	$\mathbf{2.54 \times 10^{5}}$
	$d = 50$	1.21×10^{4}	2.97×10^{3}	3.59×10^{3}	1.23×10^{3}	1.31×10^{3}	2.41×10^{2}	6.86×10^{3}	8.59×10^{2}	$\mathbf{1.23 \times 10^{3}}$	$\mathbf{4.69 \times 10^{2}}$
CEC16	$d = 10$	1.86×10^{3}	2.31×10^{2}	1.72×10^{3}	1.31×10^{2}	1.69×10^{3}	8.33×10^{1}	1.65×10^{3}	1.74×10^{1}	$\mathbf{1.67 \times 10^{3}}$	$\mathbf{7.33 \times 10^{1}}$
	$d = 30$	3.33×10^{3}	5.42×10^{2}	2.37×10^{3}	3.57×10^{2}	2.52×10^{3}	1.87×10^{2}	3.67×10^{3}	1.64×10^{2}	$\mathbf{1.75 \times 10^{3}}$	$\mathbf{6.92 \times 10^{2}}$
	$d = 50$	4.54×10^{3}	5.72×10^{2}	2.78×10^{3}	4.12×10^{2}	2.94×10^{3}	2.63×10^{2}	5.56×10^{3}	6.15×10^{2}	$\mathbf{2.11 \times 10^{3}}$	$\mathbf{5.57 \times 10^{2}}$
CEC17	$d = 10$	1.91×10^{3}	1.10×10^{2}	1.73×10^{3}	1.27×10^{1}	$\mathbf{1.72 \times 10^{3}}$	$\mathbf{1.41 \times 10^{1}}$	1.75×10^{3}	7.17×10^{0}	$\mathbf{1.72 \times 10^{3}}$	$\mathbf{1.34 \times 10^{0}}$
	$d = 30$	2.33×10^{3}	2.59×10^{2}	1.91×10^{3}	1.45×10^{2}	2.05×10^{3}	1.89×10^{2}	2.47×10^{3}	1.25×10^{2}	$\mathbf{1.81 \times 10^{3}}$	$\mathbf{2.78 \times 10^{2}}$
	$d = 50$	4.08×10^{3}	7.34×10^{2}	2.58×10^{3}	2.16×10^{2}	2.80×10^{3}	1.59×10^{2}	4.31×10^{3}	4.30×10^{2}	$\mathbf{1.96 \times 10^{3}}$	$\mathbf{2.80 \times 10^{2}}$
CEC20	$d = 10$	2.14×10^{3}	6.62×10^{1}	2.05×10^{3}	5.01×10^{1}	2.04×10^{3}	5.67×10^{0}	2.06×10^{3}	9.19×10^{0}	$\mathbf{2.02 \times 10^{3}}$	$\mathbf{4.64 \times 10^{0}}$
	$d = 30$	2.69×10^{3}	1.72×10^{2}	2.35×10^{3}	1.45×10^{2}	2.38×10^{3}	1.31×10^{2}	2.69×10^{3}	1.35×10^{2}	$\mathbf{2.03 \times 10^{3}}$	$\mathbf{2.72 \times 10^{2}}$
	$d = 50$	3.53×10^{3}	4.10×10^{2}	2.82×10^{3}	1.98×10^{2}	3.04×10^{3}	4.51×10^{2}	3.79×10^{3}	1.52×10^{2}	$\mathbf{2.11 \times 10^{3}}$	$\mathbf{3.88 \times 10^{2}}$
CEC21	$d = 10$	2.31×10^{3}	6.44×10^{1}	2.05×10^{3}	5.01×10^{1}	2.04×10^{3}	5.67×10^{0}	$\mathbf{2.06 \times 10^{3}}$	$\mathbf{9.19 \times 10^{0}}$	2.02×10^{3}	4.64×10^{0}
	$d = 30$	2.60×10^{3}	3.67×10^{1}	2.37×10^{3}	1.28×10^{2}	2.39×10^{3}	1.71×10^{2}	2.55×10^{3}	2.35×10^{2}	$\mathbf{2.17 \times 10^{3}}$	$\mathbf{6.17 \times 10^{1}}$
	$d = 50$	2.94×10^{3}	8.86×10^{1}	2.47×10^{3}	2.95×10^{1}	2.52×10^{3}	1.81×10^{1}	2.87×10^{3}	3.62×10^{1}	$\mathbf{2.23 \times 10^{3}}$	$\mathbf{4.42 \times 10^{1}}$
CEC22	$d = 10$	2.39×10^{3}	1.07×10^{2}	2.30×10^{3}	5.34×10^{1}	2.25×10^{3}	4.52×10^{1}	2.32×10^{3}	2.49×10^{1}	$\mathbf{2.29 \times 10^{3}}$	$\mathbf{4.72 \times 10^{2}}$
	$d = 30$	7.68×10^{3}	7.25×10^{1}	3.96×10^{3}	1.69×10^{3}	2.30×10^{3}	1.22×10^{0}	8.95×10^{3}	1.91×10^{3}	$\mathbf{2.28 \times 10^{3}}$	$\mathbf{8.08 \times 10^{1}}$
	$d = 50$	1.47×10^{4}	6.36×10^{2}	8.67×10^{3}	8.25×10^{2}	8.50×10^{3}	9.69×10^{2}	1.63×10^{4}	3.71×10^{2}	$\mathbf{5.97 \times 10^{3}}$	$\mathbf{9.92 \times 10^{1}}$
CEC23	$d = 10$	2.66×10^{3}	2.65×10^{2}	2.61×10^{3}	5.86×10^{0}	2.63×10^{3}	9.25×10^{0}	2.64×10^{3}	7.38×10^{0}	$\mathbf{2.41 \times 10^{3}}$	$\mathbf{3.28 \times 10^{1}}$
	$d = 30$	3.06×10^{3}	3.44×10^{1}	2.72×10^{3}	2.06×10^{1}	2.89×10^{3}	4.43×10^{1}	3.01×10^{3}	2.96×10^{1}	$\mathbf{2.33 \times 10^{3}}$	$\mathbf{8.46 \times 10^{1}}$
	$d = 50$	3.89×10^{3}	1.28×10^{2}	2.91×10^{3}	2.92×10^{2}	3.29×10^{3}	1.46×10^{2}	3.53×10^{3}	1.00×10^{2}	$\mathbf{2.37 \times 10^{3}}$	$\mathbf{9.83 \times 10^{1}}$
CEC24	$d = 10$	2.69×10^{3}	1.14×10^{2}	2.74×10^{3}	1.05×10^{1}	2.60×10^{3}	1.31×10^{2}	2.77×10^{3}	9.92×10^{0}	$\mathbf{2.53 \times 10^{3}}$	$\mathbf{1.84 \times 10^{0}}$
	$d = 30$	3.31×10^{3}	8.87×10^{1}	2.93×10^{3}	5.28×10^{1}	3.13×10^{3}	7.74×10^{1}	3.16×10^{3}	3.30×10^{1}	$\mathbf{2.49 \times 10^{3}}$	$\mathbf{1.12 \times 10^{2}}$
	$d = 50$	3.98×10^{3}	1.56×10^{2}	3.06×10^{3}	3.06×10^{2}	3.46×10^{3}	1.25×10^{2}	3.69×10^{3}	3.47×10^{2}	$\mathbf{2.45 \times 10^{3}}$	$\mathbf{1.76 \times 10^{1}}$
CEC25	$d = 10$	3.04×10^{3}	1.93×10^{2}	2.91×10^{3}	2.11×10^{1}	2.91×10^{3}	2.21×10^{1}	2.94×10^{3}	1.61×10^{1}	$\mathbf{2.60 \times 10^{3}}$	$\mathbf{3.20 \times 10^{1}}$
	$d = 30$	3.39×10^{3}	1.62×10^{2}	2.94×10^{3}	3.21×10^{1}	2.89×10^{3}	1.07×10^{1}	3.29×10^{3}	1.09×10^{2}	$\mathbf{2.78 \times 10^{3}}$	$\mathbf{3.27 \times 10^{2}}$
	$d = 50$	6.33×10^{3}	1.23×10^{3}	3.33×10^{3}	2.12×10^{2}	3.01×10^{3}	2.51×10^{1}	6.08×10^{3}	6.50×10^{2}	$\mathbf{2.68 \times 10^{3}}$	$\mathbf{8.46 \times 10^{2}}$
CEC26	$d = 10$	3.33×10^{3}	4.55×10^{2}	2.92×10^{3}	3.61×10^{1}	2.87×10^{3}	9.48×10^{1}	3.09×10^{3}	1.51×10^{1}	$\mathbf{2.63 \times 10^{3}}$	$\mathbf{5.95 \times 10^{1}}$
	$d = 30$	8.13×10^{3}	6.45×10^{2}	4.38×10^{3}	2.14×10^{2}	4.13×10^{3}	1.69×10^{3}	7.08×10^{3}	2.29×10^{2}	$\mathbf{2.64 \times 10^{3}}$	$\mathbf{1.20 \times 10^{2}}$
	$d = 50$	1.30×10^{4}	1.85×10^{3}	5.93×10^{3}	4.83×10^{2}	9.26×10^{3}	7.46×10^{1}	1.18×10^{4}	6.15×10^{2}	$\mathbf{3.31 \times 10^{3}}$	$\mathbf{1.51 \times 10^{3}}$
CEC27	$d = 10$	3.14×10^{3}	3.19×10^{2}	3.09×10^{3}	3.42×10^{0}	3.11×10^{3}	2.62×10^{0}	3.10×10^{3}	1.35×10^{0}	$\mathbf{2.78 \times 10^{3}}$	$\mathbf{4.70 \times 10^{1}}$
	$d = 30$	3.60×10^{3}	2.00×10^{2}	3.23×10^{3}	1.56×10^{1}	3.33×10^{3}	1.00×10^{2}	3.42×10^{3}	5.79×10^{1}	$\mathbf{2.89 \times 10^{3}}$	$\mathbf{1.53 \times 10^{2}}$
	$d = 50$	4.32×10^{3}	2.53×10^{1}	3.51×10^{3}	4.83×10^{2}	4.20×10^{3}	3.67×10^{2}	4.53×10^{3}	1.14×10^{2}	$\mathbf{2.76 \times 10^{3}}$	$\mathbf{5.97 \times 10^{2}}$
CEC28	$d = 10$	3.41×10^{3}	1.59×10^{2}	3.29×10^{3}	1.24×10^{2}	3.11×10^{3}	5.91×10^{1}	3.23×10^{3}	1.83×10^{1}	$\mathbf{2.85 \times 10^{3}}$	$\mathbf{8.59 \times 10^{1}}$
	$d = 30$	4.27×10^{3}	4.10×10^{2}	3.37×10^{3}	4.41×10^{1}	3.20×10^{3}	7.86×10^{0}	3.97×10^{3}	1.82×10^{2}	$\mathbf{3.11 \times 10^{3}}$	$\mathbf{2.08 \times 10^{2}}$
	$d = 50$	6.75×10^{3}	1.25×10^{1}	4.07×10^{3}	4.85×10^{2}	3.34×10^{3}	1.22×10^{1}	6.81×10^{3}	7.49×10^{2}	$\mathbf{3.41 \times 10^{3}}$	$\mathbf{9.09 \times 10^{2}}$
CEC29	$d = 10$	3.25×10^{3}	5.37×10^{1}	3.17×10^{3}	3.79×10^{1}	3.18×10^{3}	2.96×10^{1}	3.19×10^{3}	1.14×10^{1}	$\mathbf{3.11 \times 10^{3}}$	$\mathbf{2.06 \times 10^{2}}$
	$d = 30$	4.58×10^{3}	3.62×10^{2}	3.63×10^{3}	1.05×10^{2}	3.81×10^{3}	1.76×10^{2}	4.68×10^{3}	1.66×10^{2}	$\mathbf{3.52 \times 10^{3}}$	$\mathbf{4.08 \times 10^{2}}$
	$d = 50$	6.48×10^{3}	4.07×10^{1}	4.25×10^{3}	1.22×10^{2}	4.60×10^{3}	1.11×10^{1}	7.54×10^{3}	6.45×10^{2}	$\mathbf{2.99 \times 10^{3}}$	$\mathbf{8.93 \times 10^{2}}$

5. Data Simulation and Analysis

5.1. Test Environment and Experimental Data

The proposed CG-TSA was used to solve the problem of airport gate assignment. To test and simulate the scenario, 10 gates for 40 flights between 0:00 and 24:00 were used based on a typical airport flight schedule for a day. Gates 1–10 were numbered. Gate 1, Gate 6, Gate 7, and Gate 9 were the four big gates. The five medium gates were Gate 2–Gate 3, Gate 5, Gate 8, and Gate 10, and the one small gate was Gate 4. Gates 1–4 and Gates 6–7 were located near the bridge. Gate 5 and Gates 8–10 were located off the bridge. Large gates can be assigned to all types of aircraft, medium gates can be assigned to medium and small aircraft, and small gates can only be assigned to small aircraft. Table 7 contains information about the boarding gate (where "1" indicates that the gate is close to the boarding bridge and "2" indicates that the gate is far away from the boarding bridge). Table 8 contains details of flight information. The information described includes the flight number, the

serial number of the flight number in order of arrival time, the aircraft type (represented by 1–8 in this document), the matching gate type (1–3 for small, medium, and large gates, respectively), and the landing runway (there are three landing runways represented by A, B, and C, respectively). The safe time between two adjacent flights at the same gate is 20 min. Table 9 contains information on fuel consumption. The described information includes the fuel-consuming reference models as well as the average fuel consumption per minute. Table 10 contains information on runway-to-gate distances. The distance between the aircraft number and each runway is specified (in this paper, runways A, B, and C).

Table 7. Detailed information on gates.

Gates	Type	Near the Boarding Bridge
Gate1	large	1
Gate2	medium	1
Gate3	medium	1
Gate4	small	1
Gate5	medium	2
Gate6	large	1
Gate7	large	1
Gate8	medium	2
Gate9	large	2
Gate10	medium	2

5.2. Experimental Results and Analysis of Airport Gate Assignment

In this paper, the proposed CG-TSA was run 20 times independently to solve the gate assignment problem. To assess the efficacy of the CG-TSA and the optimisation model, an optimal gate assignment system was used. Figure 5 depicts the Gantt chart of the optimal gate assignment results. The horizontal coordinate in Figure 5 represents the moment, the vertical coordinate the gate serial number, and the rectangle a designated flight. The corresponding flight, which is used to indicate the positioned flight, is marked on the aircraft number. The plan is visible for parking at Gates 1–10 with no overlap. The robust multiobjective optimisation model for maximising real-time flights for boarding gate assignments can handle dynamic changes such as flight conflicts, delays, and so on. The proposed CG-TSA solved the multiobjective optimisation model for the gate assignment problem quickly and efficiently. It demonstrated better optimisation capability in complex optimisation problems.

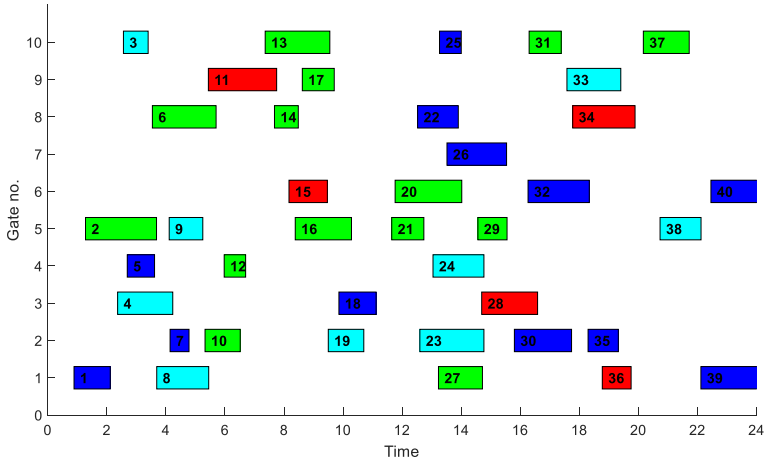


Figure 5. Gantt chart of airport gate assignment.

Table 8. Detailed information on flights.

Flight No.	Flight Serial No.	Flight Type	Match Type	Landing Strip
4317	24	3	1	C
1580	37	1	1	C
3402	8	4	1	A
4369	35	7	2	B
3688	39	7	2	C
3225	33	6	2	C
1983	28	2	1	B
1685	21	7	2	C
3811	31	8	3	B
2067	20	4	1	A
1515	15	6	2	A
3490	1	5	2	A
4288	13	5	2	A
1188	14	2	1	A
3234	3	2	1	B
1078	5	4	1	A
3991	19	6	2	A
4892	36	8	3	C
2624	26	4	1	C
3538	10	7	2	A
3739	32	7	2	C
1762	18	7	2	C
4154	7	2	1	C
4349	16	7	2	C
4193	6	2	1	C
3503	29	2	1	A
3607	25	2	1	A
3577	34	5	2	C
3557	30	5	2	C
3507	12	4	1	C
4645	23	7	2	B
3218	27	2	1	C
2799	2	8	3	C
1819	38	7	2	A
4407	22	1	1	A
2028	17	6	2	C
2657	2	7	2	C
2788	4	2	1	C
4927	11	5	2	B
1111	40	5	2	A

Table 9. Detailed information on fuel consumption.

Flight Type	Fuel Consumption (kg/min)
Flight 1	500
Flight 2	738
Flight 3	928
Flight 4	1506
Flight 5	1850
Flight 6	2438
Flight 7	2627
Flight 8	3137

Table 10. Detailed information on runway-to-gate distances.

Gates	Runway A	Runway B	Runway C
Gates 1	1.355	8.156	1.668
Gates 2	8.137	3.312	1.467
Gates 3	2.470	2.254	7.107
Gates 4	3.143	8.039	7.857
Gates 5	9.086	0.452	0.688
Gates 6	3.657	1.013	7.732
Gates 7	9.088	6.734	5.557
Gates 8	4.145	8.802	2.079
Gates 9	4.027	2.574	7.904
Gates 10	1.171	9.112	5.951

Considering passenger convenience and satisfaction, as well as airport fuel costs, the boarding bridge rate and burning fuel costs were optimised as the objective functions. The GA [38], basic TSA, and CG-TSA were used to test and compare the objective optimisation model. The experimental results are shown in Figures 6 and 7. In Figure 6, the horizontal coordinate indicates the number of iterations, and the vertical coordinate indicates the boarding bridge rate. As shown in Figure 6, the basic TSA helped flights to stop at the boarding bridge with a boarding bridge rate of approximately 82% as the number of iterations increased. The GA commonly used for gate assignment achieved an approximately 91.5% bridging rate, and the proposed CG-TSA helped the bridging rate of flights to reach approximately 97.5%. Thus, the proposed CG-TSA in this paper had high optimisation efficiency and better search results than the traditional GA and basic TSA. The results obtained by CG-TSA were very close to the ideal solution to meet the needs of airport authorities. In Figure 7, the horizontal coordinate indicates the number of iterations, and the vertical coordinate indicates the total fuel consumption (kg). As shown in Figure 7, as a continuum of the iterative process, the convergence rate of the algorithm gradually decreased, and the minimised fuel consumption was continuously optimised. After 140 iterations, the basic TSA curve stabilised, and the minimum optimal value point of fuel consumption explored was approximately 2.19×10^6 kg. After 150 iterations, the conventional GA curve stabilised, and the minimised fuel consumption value reached was approximately 1.95×10^6 kg. The proposed CG-TSA stabilised after 190 iterations, and the achieved minimised fuel consumption value was approximately 1.61×10^6 kg. The proposed CG-TSA jumped out at the local minima and effectively reduced fuel consumption in the gate assignment.

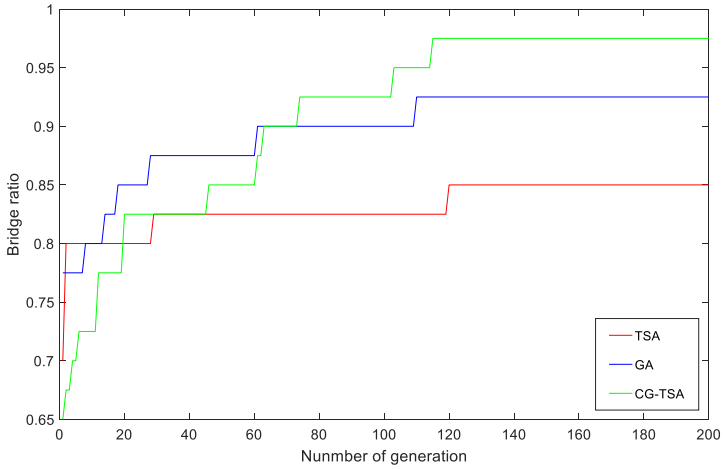


Figure 6. Comparison of airport gate assignment with bridge rate.

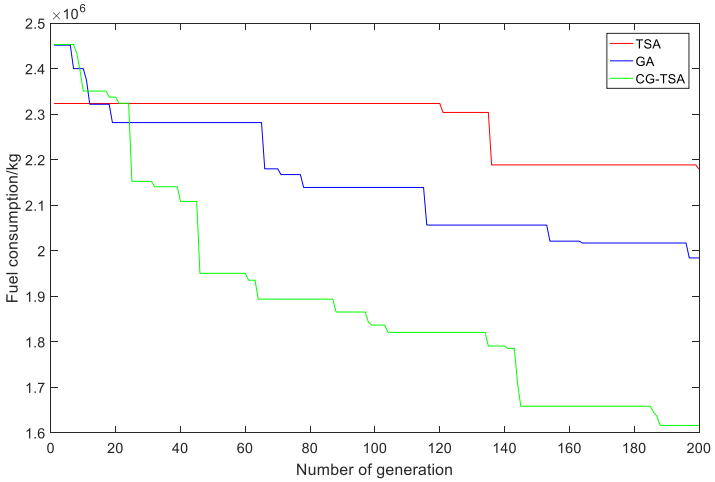


Figure 7. Total fuel consumption comparison chart.

To further analyse the effectiveness of the CG-TSA for multiobjective optimisation problems, the boarding bridge rate and the cost of burning fuel were optimised as a total objective function based on the maximum probability of the flight being allocated to the bridge and saving fuel consumption. The multiobjective optimisation model was tested and compared using GA, basic TSA, and CG-TSA and the experimental results are shown in Figure 8. In Figure 8, the horizontal coordinate indicates the number of iterations, and the vertical coordinate indicates the total objective fitness value. As shown in Figure 8, after 80 iterations, the basic TSA curve fell into the local extrema and found an optimal value of 0.86 after 195 iterations. After 120 iterations, the genetic algorithm curve converged and found an optimal value of 0.86. The proposed CG-TSA curve converged and found an optimal value of 0.85 after 140 iterations. Compared with the GA and basic TSA, the CG-TSA had better multiobjective optimisation capability and was able to provide an effective and reasonable method for airport gate assignments.

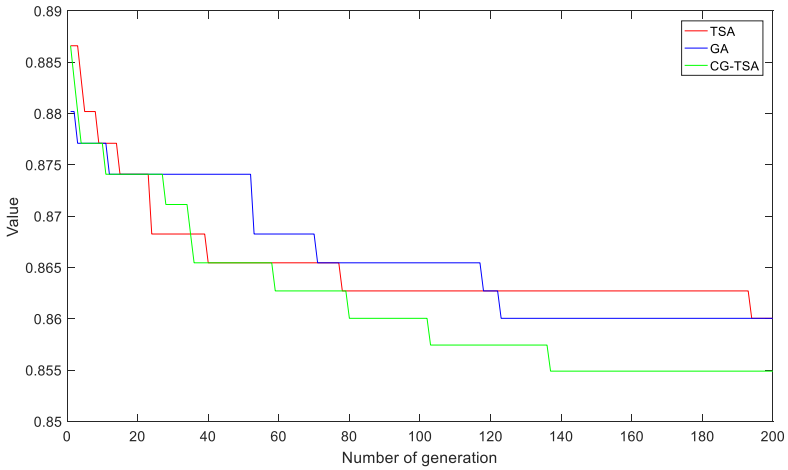


Figure 8. General objectives.

6. Conclusions

In this paper, a new multiobjective optimised gate assignment problem model is developed by minimising real-time flight conflicts, maximising the boarding bridge rate, and minimising aircraft taxiing fuel consumption. Furthermore, an improved tunicate swarm algorithm based on cosine mutation and adaptive grouping (CG-TSA) is proposed, which uses Halton sequences to initialise agent positions, improves the algorithm's initial traversal and allocation efficiency, and classifies agent adaptations into dominant and inferior groups based on the size of the fitness value. Using the AOA concept, a cosine

mutation strategy is introduced to solve the multiobjective optimisation model for gate assignments using the global optimal solution as a guide to avoid the objective falling into the local extrema in order to efficiently and reasonably allocate gates, improve airport operational efficiency, and relieve airport fuel cost pressures. The CG-TSA is validated using benchmark test functions, Wilcoxon rank-sum detection, and CEC2017 complex test functions, with the results compared to other metaheuristic and improved algorithms. The experimental results show that the improved CG-TSA is competitive and superior in terms of search accuracy, convergence speed, and stability. Finally, the genetic algorithm, basic TSA, and CG-TSA are chosen to solve the gate assignment optimisation model. The experimental results show that the CG-TSA outperforms the GA and basic TSA in solving the single-objective learning rate, fuel consumption, and multiobjective problems and that it is better suited for solving the gate assignment problem. Other airport gate assignment factors will be considered in future work, and the performance of the TSA in solving multiobjective problems will be improved.

Author Contributions: Conceptualization, Y.Z.; methodology, Y.Z.; software, Y.Z.; validation, Y.Z., Q.H., L.Y. and C.L.; formal analysis, Q.H. and L.Y.; writing—original draft preparation, Y.Z.; writing—review and editing, Y.Z., Q.H. and C.L.; resources, Q.H.; visualization, C.L. All authors have read and agreed to the published version of the manuscript.

Funding: The research was funded by the National Natural Science Foundation of China's "Research on the Evidence Chain Construction from the Analysis of the Investigation Documents (62166006)", supported by Guizhou Provincial Science and Technology Projects (Guizhou Science Foundation -ZK [2021] General 335), and the National Natural Science Foundation of China's "Rural spatial restructuring in poverty-stricken mountainous areas of Guizhou based on Spatial equity: A case study of Dianqiangui Rocky Desertification Area (41861038)".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rajapaksha, A.; Jayasuriya, N. Smart airport: A review on future of the airport operation. *Glob. J. Manag. Bus. Res.* **2020**, *20*, 25–34. [CrossRef]
2. Bouras, A.; Ghaleb, M.A.; Suryahatmaja, U.S.; Salem, A.M. The airport gate assignment problem: A survey. *Sci. World J.* **2014**, *2014*, 923859. [CrossRef] [PubMed]
3. Bihr, R.A. A conceptual solution to the aircraft gate assignment problem using 0, 1 linear programming. *Comput. Ind. Eng.* **1990**, *19*, 280–284. [CrossRef]
4. Xu, J.; Bailey, G. The airport gate assignment problem: Mathematical model and a tabu search algorithm. In Proceedings of the 34th Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 6 January 2001; IEEE: Piscataway, NJ, USA, 2001; p. 10.
5. Drexl, A.; Nikulin, Y. Multicriteria airport gate assignment and Pareto simulated annealing. *IIE Trans.* **2008**, *40*, 385–397. [CrossRef]
6. Xie, W.; Guan, J.X.; Zhou, Y.; Zhu, W.B. Gate Distribution Problem Based on Improved Simulated Annealing Algorithm. *Comput. Syst. Appl.* **2021**, *30*, 157–163.
7. Yuan, S.; Qiuzhao, H. Airport Gate Assignment Optimization Based on Hybrid Particle Swarm Algorithm. *J. Civ. Aviat. Flight Univ. China* **2013**, *24*, 24–28.
8. Dell'Orco, M.; Marinelli, M.; Altieri, M.G. Solving the gate assignment problem through the fuzzy bee colony optimization. *Transp. Res. Part C Emerg. Technol.* **2017**, *80*, 424–438. [CrossRef]
9. Cecen, R.K. Multi-objective optimization model for airport gate assignment problem. *Aircr. Eng. Aerosp. Technol.* **2021**, *93*, 311–318. [CrossRef]
10. Yan, S.; Tang, C.H. A heuristic approach for airport gate assignments for stochastic flight delays. *Eur. J. Oper. Res.* **2007**, *180*, 547–567. [CrossRef]
11. Hassounah, M.I.; Steuart, G.N. Demand for aircraft gates. *Transp. Res. Rec.* **1993**, *1423*, 26–33.
12. Yan, S.; Huo, C.M. Optimization of multiple objective gate assignments. *Transp. Res. Part A Policy Pract.* **2001**, *35*, 413–432. [CrossRef]

13. Dorndorf, U.; Jaehn, F.; Pesch, E. Flight gate assignment and recovery strategies with stochastic arrival and departure times. *OR Spectr.* **2017**, *39*, 65–93. [CrossRef]
14. Zhang, H.H.; Xue, Q.W.; Jiang, Y. Multi-objective gate assignment based on robustness in hub airports. *Adv. Mech. Eng.* **2017**, *9*, 1687814016688588. [CrossRef]
15. Cheng, Y. A knowledge-based airport gate assignment system integrated with mathematical programming. *Comput. Ind. Eng.* **1997**, *32*, 837–852. [CrossRef]
16. Jaehn, F. Solving the flight gate assignment problem using dynamic programming. *Z. Betr.* **2010**, *80*, 1027–1039. [CrossRef]
17. Yan, S.; Chang, C.M. A network model for gate assignment. *J. Adv. Transp.* **1998**, *32*, 176–189. [CrossRef]
18. Bi, J.; Wu, Z.; Wang, L.; Xie, D.; Zhao, X. A tabu search-based algorithm for airport gate assignment: A case study in Kunming, China. *J. Adv. Transp.* **2020**, *2020*, 8835201. [CrossRef]
19. Hu, X.B.; Paolo, E.D. An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In *Multi-Objective Memetic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 71–89.
20. Asadi, E.; Schultz, M.; Fricke, H. Optimal schedule recovery for the aircraft gate assignment with constrained resources. *Comput. Ind. Eng.* **2021**, *162*, 107682. [CrossRef]
21. Deng, W.; Sun, M.; Zhao, H.; Li, B.; Wang, C. Study on an airport gate assignment method based on improved ACO algorithm. *Kybernetes* **2017**, *47*, 20–43. [CrossRef]
22. Kaur, S.; Awasthi, L.K.; Sangal, A.L.; Dhiman, G. Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103541. [CrossRef]
23. Li, L.L.; Liu, Z.F.; Tseng, M.L.; Zheng, S.J.; Lim, M.K. Improved tunicate swarm algorithm: Solving the dynamic economic emission dispatch problems. *Appl. Soft Comput.* **2021**, *108*, 107504. [CrossRef]
24. Houssein, E.H.; Helmy, B.E.D.; Elngar, A.A.; Abdelminaam, D.S.; Shaban, H. An improved tunicate swarm algorithm for global optimization and image segmentation. *IEEE Access* **2021**, *9*, 56066–56092. [CrossRef]
25. Ahmadianfar, I.; Bozorg-Haddad, O.; Chu, X. Gradient-based optimizer: A new metaheuristic optimization algorithm. *Inf. Sci.* **2020**, *540*, 131–159. [CrossRef]
26. Sharma, A.; Dasgotra, A.; Tiwari, S.K.; Sharma, A.; Jately, V.; Azzopardi, B. Parameter extraction of photovoltaic module using tunicate swarm algorithm. *Electronics* **2021**, *10*, 878. [CrossRef]
27. Daş, G.S.; Gzara, F.; Stützle, T. A review on airport gate assignment problems: Single versus multi objective approaches. *Omega* **2020**, *92*, 102146. [CrossRef]
28. Kang, L.; Hansen, M. Improving airline fuel efficiency via fuel burn prediction and uncertainty estimation. *Transp. Res. Part C Emerg. Technol.* **2018**, *97*, 128–146. [CrossRef]
29. Wang, X.; Hickernell, F.J. Randomized halton sequences. *Math. Comput. Model.* **2000**, *32*, 887–899. [CrossRef]
30. Zhang, T.; Zhou, Z.L.; An, S.Z.; Zhang, J. On the proposed Monte Carlo method for computing multiple integrals. *J. Wenzhou Univ. (Nat. Sci. Ed.)* **2012**, *33*, 33–38.
31. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
32. Chen, L.; Tian, Y.; Ma, Y. An improved grasshopper optimization algorithm based on dynamic dual elite learning and sinusoidal mutation. *Computing* **2022**, *104*, 981–1015. [CrossRef]
33. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
34. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
35. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
36. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
37. Wu, G.; Mallipeddi, R.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization*; Technical Report; Kyungpook National University: Daegu, Korea, 2017.
38. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]