mathematics

Article

A New Gaining-Sharing Knowledge Based Algorithm with Parallel Opposition-Based Learning for Internet of Vehicles

Jeng-Shyang Pan [1,2], Li-Fa Liu [1], Shu-Chuan Chu [1,3,*], Pei-Cheng Song [1] and Geng-Geng Liu [4]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; jspan@cc.kuas.edu.tw (J.-S.P.); llf@sdust.edu.cn (L.-F.L.); spc@sdust.edu.cn (P.-C.S.)
[2] Department of Information Management, Chaoyang University of Technology, 168, Jifeng E. Rd., Wufeng District, Taichung 41349, Taiwan
[3] College of Science and Engineering, Flinders University, Bedford Park, SA 5042, Australia
[4] College of Computer and Data Science, Fuzhou University, Xueyuan Road No.2, Fuzhou 350116, China; liugenggeng@fzu.edu.cn
* Correspondence: scchu0803@sdust.edu.cn

Abstract: Heuristic optimization algorithms have been proved to be powerful in solving nonlinear and complex optimization problems; therefore, many effective optimization algorithms have been applied to solve optimization problems in real-world scenarios. This paper presents a modification of the recently proposed Gaining–Sharing Knowledge (GSK)-based algorithm and applies it to optimize resource scheduling in the Internet of Vehicles (IoV). The GSK algorithm simulates different phases of human life in gaining and sharing knowledge, which is mainly divided into the senior phase and the junior phase. The individual is initially in the junior phase in all dimensions and gradually moves into the senior phase as the individual interacts with the surrounding environment. The main idea used to improve the GSK algorithm is to divide the initial population into different groups, each searching independently and communicating according to two main strategies. Opposite-based learning is introduced to correct the direction of convergence and improve the speed of convergence. This paper proposes an improved algorithm, named parallel opposition-based Gaining–Sharing Knowledge-based algorithm (POGSK). The improved algorithm is tested with the original algorithm and several classical algorithms under the CEC2017 test suite. The results show that the improved algorithm significantly improves the performance of the original algorithm. When POGSK was applied to optimize resource scheduling in IoV, the results also showed that POGSK is more competitive.

Keywords: Gaining–Sharing Knowledge-based algorithm; Taguchi method; opposition-based learning; resource scheduling; parallel mechanism

MSC: 68W50

Citation: Pan, J.-S.; Liu, L.-F.; Chu, S.-C.; Song, P.-C.; Liu, G.-G. A New Gaining-Sharing Knowledge Based Algorithm with Parallel Opposition-Based Learning for Internet of Vehicles. *Mathematics* **2023**, *11*, 2953. https://doi.org/10.3390/math11132953

Academic Editor: Ioannis G. Tsoulos

Received: 3 June 2023
Revised: 28 June 2023
Accepted: 29 June 2023
Published: 2 July 2023

1. Introduction

Heuristic optimization algorithms have been studied and discovered over the past three decades and have been fully proven to solve a variety of complex, nonlinear optimization problems. These methods are both user-friendly and do not necessitate mathematical analysis of the optimization problem. Compared with the traditional methods, they have the advantages of flexibility, no gradient mechanism and avoiding being trapped in the local optimum [1]. These features drew a large number of researchers to participate in the design. Most heuristic algorithms are inspired by the author's observation of animal and plant phenomena in nature. To solve optimization problems, algorithms are used to simulate growth and evolution. In real scenarios, heuristics are also widely used, such as path planning [2], wireless sensor network localization problem [3], wireless sensor network routing problem [4], airport gate assignment [5] and cloud computing workflow scheduling [6].

Heuristic algorithms can be divided into four categories [7]. The first category is algorithms based on swarm intelligence techniques. Much of the inspiration for such algorithms comes from observations of social animals. In a population, each individual exhibits a certain degree of independence while still interacting with the entire group. The main representative algorithms are particle swarm optimization (PSO) [8], the phasmatodea population evolution algorithm (PPE) [9], the Gannet optimization algorithm (GOA) [10], the grey wolf optimizer (GWO) [11], cat swarm optimization (CSO) [12], etc. The second category is algorithms based on evolutionary techniques. Such algorithms are inspired by developments in biology. The initial random population is gradually iterated to achieve the final optimization purpose through crossover, mutation, selection and other operations. The main representative algorithms are the genetic algorithm (GA) [13], the differential evolution algorithm (DE) [14], the quantum evolutionary algorithm (QEA) [15], etc. The third category is the algorithm based on physical phenomena. This kind of algorithm simulates the law of some natural phenomena. The main representative algorithms are the Archimedes optimization algorithm (AOA) [16], the simulated annealing algorithm (SAA) [17], the sine cosine algorithm (SCA) [18], etc. The fourth category of algorithms is those based on human-related technology. As an independent intelligent and rational individual, each person has unique physical and psychological behavior. The main representative algorithms are the Teaching–Learning-Based Optimization Algorithm (TLBO) [19], the Gaining–Sharing Knowledge-based algorithm (GSK) [7], etc.

The original GSK simulates the behavior of acquiring and sharing knowledge throughout a person's life, culminating in the maturation of the individual [20]. The author divides human life into two distinct phases: the junior phase, which corresponds to childhood, and the senior phase, which corresponds to adulthood. The strategies for knowledge acquisition and sharing are different in these two phases. At the beginning of the algorithm, individuals tend to use a relatively naive method to acquire and share knowledge. However, not all disciplines (on all dimensions of the solution) use this naive method. In some disciplines, individuals will also use relatively advanced methods for knowledge acquisition and sharing. With the growth of individuals, the algorithm enters the middle stage and the learning of knowledge is more inclined to use the advanced method, while a few disciplines still use the naive method. Individuals go through two stages, alternating between naive or advanced strategies to update their knowledge in each discipline. Individuals eventually reach maturity, which is when they find their optimal position. Ali Wagdy Mohamed demonstrated its powerful optimization capabilities on the CEC2017 test suite when he presented the GSK algorithm. Although the GSK algorithm demonstrates excellent convergence in solving the optimization problem, there is room for improvement in avoiding locally optimal solutions and convergence speed. To further improve the performance of the GSK algorithm, we propose several approaches to be incorporated into the GSK algorithm, which are described next in turn. Experiments have been conducted to demonstrate that these approaches are effective in improving the performance of the GSK algorithm.

Parallel processing is concerned with producing the same results using multiple processors with the goal of reducing the running time [21]. Because physical parallel processing cannot be used in the optimization algorithm, we adopt an alternative approach. The main idea of the parallel mechanism is to divide the initial population into several different groups. Each group performs iterative updates independently and communicates regularly between groups. The parallel mechanism has been applied widely, including to the parallel particle swarm algorithm (Chu S C 2005) [21] and parallel genetic algorithms [22]. In addition, parallel strategy is also used in multi-objective optimization algorithms. Cai D proposed an evolutionary algorithm based on uniform and contraction for many-objective optimization [23], which uses a parallel mechanism to enhance the local search ability.

The communication strategies between groups can be varied for optimizing different algorithms. This paper presents a communication strategy using the Taguchi method. The main idea is to use a pre-designed orthogonal table for crossover experiments. Compared with the traditional experimental method, it can achieve almost the same effect while obviously reducing the number of experiments. The Taguchi method has the advantages of reducing the number of experiments, reducing the cost of experiments and improving the efficiency of experiments [24]. It has been successfully applied to improve the genetic algorithm (Jinn-Tsong Tsai 2004) [25], the Archimedes optimization algorithm (Shi-Jie Jiang 2022) [26], the cat swarm optimization algorithm (Tsai P W 2012) [24], etc. In addition, opposition-based learning (OBL) was also incorporated into GSK. The concept of OBL was proposed by Tizhoosh (2005) [27]. After that, some classical optimization algorithms started to introduce this idea. It has been successfully applied to improve grey wolf optimization (Souvik Dhargupta 2020) [28] (Dhargupta S 2020) [29], the differential evolution algorithm (Rahnamayan S 2008) [30] (Wu Deng2021) [31], particle swarm optimization (Wang H 2011) [32], the grasshopper optimization algorithm (Ahmed A. Ewees 2018) [33], etc.

In order to reach a convincing conclusion, the performance of any optimization or evolutionary algorithm can only be judged via extensive benchmark function tests [34]. Some diverse and difficult test problems are required for this purpose and the CEC2017 test suite [35] is a widely accepted test problem. In order to apply the optimization algorithm to the complex real-world optimization problem, it is necessary that the optimization algorithm can effectively solve the single objective optimization problem. CEC2017 test suite contains 30 single-objective real-parameter numerical optimization questions. Compared with CEC2013 and CEC2014, in CEC2017, several test problems with new characteristics are proposed, such as new basic problems, composing test problems by extracting features dimension-wise from several problems, graded level of linkages, rotated trap problems and so on. In this paper, the CEC2017 test suite is utilized to evaluate the proposed algorithm (POGSK), the original algorithm and several classical algorithms.

The IoV enables vehicles on the road to exchange information with roadside units (RSUs) [36]. Therefore, users can expect quick, comprehensive and convenient services, such as road condition information, traffic jam section information, traffic condition information, city entertainment information, etc. However, traditional resource-allocation strategies may not be able to provide satisfactory Quality of Service (QoS) due to several factors. These include resource constraints, network transmission delays and the deployment of RSUs. Scheduling problems can be solved using various methods, which can be roughly grouped into three categories: exact, approximate and heuristic [37]. The exact method is to calculate all the solutions of the whole search space to find the optimal solution, which is obviously only suitable for small-scale problems. The approximation method uses certain mathematical rules to find the optimal solution, which requires different analyses for different problems. However, in most cases, the mathematical analysis of the problem is difficult. Therefore, the heuristic approach is a decent option. The main objective of scheduling algorithms is to find the best resources in the cloud for the applications (tasks) of the end user. This improves the QoS parameters and resource utilization [38]. In order to solve this optimization problem, this paper proposes using the heuristic algorithm POGSK to complete resource scheduling.

The main contributions of this paper are as follows:

1. An improved Gaining–Sharing Knowledge-based algorithm (POGSK) is proposed, which uses parallel strategy and OBL strategy. The use of parallel strategy increases the diversity of the population so that the algorithm can effectively avoid local optimal solutions. The OBL strategy can correct the convergence direction and improve the convergence accuracy.

2. A new inter-group communication strategy is designed. Specifically, the Taguchi communication strategy and the population-merger communication strategy were used. This enables efficient exchange of information between subpopulations and avoids the weakening of algorithm performance caused by the reduction of the number of individuals in subpopulations.

3. POGSK is used in the resource-scheduling problems of the IoV to improve QoS, which can reflect the performance of the algorithm in real scenarios. Simulation results show that POGSK is more competitive than other algorithms.

2. Related Works

2.1. GSK Algorithm

GSK is a human-based heuristic algorithm that gradually updates knowledge (corresponding to the solution of the algorithm) by simulating the process of knowledge sharing and acquisition in human life. The algorithm mainly consists of two phases: the junior phase and the senior phase, for knowledge gaining and sharing in phases have different processes [7].

When individuals are young, they prefer to interact with individuals who are similar to themselves. Despite their immature ability to distinguish right from wrong, they are willing to communicate with unfamiliar people, which represents curiosity in the junior phase. People who are similar to themselves correspond to their relatives, friends and other small surrounding groups. People who are unfamiliar correspond to the stranger. The above scheme is the junior gaining–sharing scheme. In the junior phase, more dimensions are updated using this scheme than using the other (senior gaining–sharing) scheme.

After updating and iteration, individuals reach middle age gradually. As the capacity to distinguish between right and wrong gradually increases, individuals are more willing to divide the crowd into three different populations: the advantaged, the disadvantaged and the general population. Individuals improve themselves by interacting with these three groups. The above scheme is the senior gaining–sharing scheme. In the senior phase, more dimensions are updated using the senior scheme than using the junior scheme. In the following, we describe in detail the dimensions in which the two schemes are utilized and their respective processes.

Let x_i, i = 1, 2, 3, . . . , N; N is population size, x_i represents the individual members of the population. $x_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iD})$, D is the size of the problem and x_{ij} represents the value of an individual in this dimension. Before the renewal of each generation, we need to determine which dimensions use the senior scheme and which dimensions use the junior scheme for each individual. Based on the concept of constant human growth, these dimensions are determined to use the following nonlinear increasing or decreasing empirical equation [7].

$$D(juniorphase) = (problemsize) * (1 - \frac{G}{GEN})^k \tag{1}$$

$$D(seniorphase) = (problemsize) - D(juniorphase) \tag{2}$$

where K is knowledge rate of a real number, K > 0, G is generation number and GEN is the maximum number of generations.

The steps for the junior scheme are as follows:

1. The fitness of all individuals is calculated and the sequence that follows is generated by ranking the individual from high to low fitness: $(x_{best}, x_2, x_3, \ldots, x_{n-1}, x_{worse})$. Each individual chooses three individuals to communicate with according to step 2.

2. For each individual x_i which is not the best and the worst, select x_{i-1} and x_{i+1}. For the best individual x_{best}, select x_{best+1} and x_{best+2}. For the worst individual x_{worse}, select $x_{worse-1}$, $x_{worse-2}$. In addition, an individual x_r is selected in a random way. These three individuals are sources of information. The pseudo-code for the above junior scheme is presented in Algorithm 1:

Algorithm 1: Junior scheme pseudo-code

```
for i = 1: N do
    for j = 1: D do
        if rand<=K_r(Konwledge ratio) then
            if f(x_i) > f(x_r) then
                x_ij^new = x_i+k_f*[(x_{i-1} − x_{i+1}) + (x_r − x_i)]
            else
                x_ij^new = x_i+k_f*[(x_{i-1} − x_{i+1}) + (x_i − x_r)]
            end
        else
            x_ij^new = x_ij^old
        end
    end
end
```

Here, k_f represents the knowledge factor ($k_f > 0$).

The steps for the senior scheme are as follows:

1. The fitness of all people is calculated and the sequence that follows is generated by ranking the individual from high to low fitness: $(x_{best}, x_2, x_3, \ldots, x_{n-1}, x_{worse})$.

2. The population is divided into three parts: the first 100p% with better fitness, the last 100p% with poor fitness and the middle NP − (2 * 100p%), where p is the proportion of a population division. For example, if p = 0.1, NP = 100, the best group is the top 10 peoples with better fitness, the worst group is the bottom 10 peoples with poor fitness and the middle group is the middle 80 peoples. From these three groups, x_{p_best}, x_{p_worse} and x_{p_middle} are selected as sources of information. The pseudo-code for the above senior scheme is presented in Algorithm 2:

Algorithm 2: senior scheme pseudo-code

```
for i = 1: N do
    for j = 1: D do
        if rand<=K_r(Konwledge ratio) then
            if f(x_i) > f(x_m) then
                x_ij^new = x_i + k_f*[(x_{p_best} − x_{p_worse}) + (x_m − x_i)]
            else
                x_ij^new = x_i + k_f*[(x_{p_best} − x_{p_worse}) + (x_i − x_m)]
            end
        else
            x_ij^new = x_ij^old
        end
    end
end
```

There are several important parameters in GSK, respectively knowledge rate K that controls the proportion of junior and senior schemes in the individual renewal scheme, knowledge factor K_f that controls the total amount of knowledge currently learned by the individual from others, knowledge ratio K_r that controls the ratio between the current and acquired experience. The pseudo-code is presented in Algorithm 3:

Algorithm 3: GSK pseudo-code

G = 0, initialize parameters: N, K_f, K_r, K and p
Creat a random initial population x_I, $i = 1, 2, \ldots, N$
Evaluate $f(x_i)$, $i = 1, 2, \ldots, N$
for *G = 1: GEN* **do**
  computer the number of(Gained and Sherad dimensions of both phase) using experinence Equations (1) and (2);
  //Junior gaining–sharing knowledge phase//
  //Senior gaining–sharing knowledge phase//
  Evaluate $f(x_i^{new})$, $i = 1, 2, \ldots, N$
  for *i = 1: N* **do**
    if $f(x_i^{new}) < f(x_i^{old})$ **then**
      $x_i^{old} = x_i^{new}, f(x_i^{old}) = f(x_i^{new})$//update each vector
    end
    if $f(x_i^{new}) < f(x_G^{best})$ **then**
      $x_G^{best} = x_i^{new}, f(x_G^{best}) = f(x_i^{new})$//updateglobal best
    end
  end
end

2.2. Taguchi Method and Parallel Mechanism

Every engineer wants to design a satisfactory product with minimum cost and minimum time. However, many factors can impact the quality of the product and it will take a long time to test one by one. The Taguchi method was developed by Dr. Genichi Taguchi in Japan after World War II [25]. When this method was used in practical production, it greatly promoted Japan's economic recovery. The orthogonal matrix experiment is one of the important tools of the Taguchi method. Suppose a product has K influencing factors, each of which has Q levels. If the influence of each factor level on product quality is tested one by one, K^Q experiments are required [24]. This will consume a lot of time and production costs are difficult to control. The orthogonal matrix experiment uses the pre-designed orthogonal matrix to conduct a small number of experiments on each factor, which can achieve almost the same effect while greatly reducing the number of experiments. Assume that there are seven factors affecting the product, each of which has two levels, then we can use the $L_8(2^7)$ shown in Equation (3). Each row in the table represents an experiment, where the values represent the current level of factor adoption. It can be observed that in each column, the number of occurrences is the same for both levels, which guarantees the fairness of the experiment.

$$L_8(2^7) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 2 & 1 & 2 & 1 \\ 2 & 2 & 1 & 1 & 2 & 2 & 1 \\ 2 & 2 & 1 & 2 & 1 & 1 & 2 \end{bmatrix} \tag{3}$$

This paper mainly uses two levels orthogonal matrix. For the 10-dimensional experiment uses $L_{12}(2^{11})$. For the 30-dimensional experiment uses $L_{32}(2^{31})$.

The parallel mechanism, also known as multi-population strategy, is a popular algorithm-optimization method for increasing population diversity. The main idea is to divide the initial population into several subpopulations. The subpopulations were searched independently after initialization and communicated with other subpopulations according to certain conditions. For parallel strategies, the inter-group communication strategy is critical. Because the number of individuals in the subpopulation is reduced, the algorithm easily falls into the local optimum via independent search. Excellent inter-group communication strategies can make subpopulations gain the ability to escape the local optimum and effective communication strategies enable populations to exchange a small amount of information while improving their search ability. Chai proposed the tribal annexation communication strategy and herd mentality communication strategy to improve the searching ability of the whale optimization algorithm [3]. Tsai used the Taguchi communication strategy to enhance the search capability of the cat swarm optimization algorithm [24]. The above scheme demonstrates the diversity of communication strategies for different algorithms. In this paper, a suitable communication strategy is proposed according to the characteristics of the GSK algorithm. Specifically, the Taguchi communication strategy and the population-merger communication strategy were used.

2.3. Opposition-Based Learning

OBL was proposed by Tizhoosh (2005) [27], which is fundamentally based on estimates and counter estimates. OBL can modify the convergence direction of the algorithm and improve the search accuracy of the algorithm. In order to get close to the optimal position quickly, we generally want the population to be near it. However, the initial populations are generated randomly and it may be far from the optimal position or even the exact opposite. As a result the algorithm converges so slowly that it does not converge to near the optimal solution under the specified conditions. The main idea of OBL is to find the opposite position of the random initial population, evaluate it and select the better position to replace the initial population. Furthermore, during the population updating process, the position of the current individual and its opposite are evaluated and the better individuals are left.

Definition 1 (Opposite Number [27])**.** *Let x be a real number defined in the interval [a,b], then the Opposite Number x^{op} of x is defined by the following equation:*

$$x^{op} = a + b - x \tag{4}$$

Definition 2 (Opposite Point [27])**.** *Let P $(x_1, x_2, \ldots, x_n)$ be a point in an n-dimensional coordinate system with $(x_1, \ldots, x_n)$ being real numbers, where each of x_j is defined in the interval $[a_j, b_j]$. Then, the Opposite Point P^{op} $(x_1^{op}, x_2^{op}, \ldots, x_n^{op})$ is defined by the following equation:*

$$x_j^{op} = a_j + b_j - x_j \tag{5}$$

In the actual search process, the search space usually changes dynamically, the Opposite Point P_i^{op} = $((x_{i1}^{op}, x_{i2}^{op}, \ldots, x_{in}^{op}))$ of P_i is defined by the following equation; P_i belongs to the population P = $(P_1, P_2, \ldots, P_n)$.

$$a_j^{min} = min(P_{i,j}) \qquad i = 1, 2, \ldots, n \tag{6}$$

$$b_j^{max} = max(P_{i,j}) \qquad i = 1, 2, \ldots, n \tag{7}$$

$$x_{i,j}^{op} = a_j^{min} + b_j^{max} - x_{i,j} \tag{8}$$

2.4. Resource-Scheduling Problem of the IoV

With intelligent transportation and smart city development, the IoV has received more and more attention. Based on the information interaction between the on-board unit and the roadside unit, a successful architecture of the IoV is formed. In this process, due to roadside unit deployment, own resource limitation and network delay, it may not guarantee satisfactory QoS for users. In this case, the most likely bottleneck is the proper scheduling of resources [39]. Therefore, we need a scheduling algorithm to distribute the user workload into the RSUs. The algorithm must be based on resource capacity and solve the problem of over- and underutilization [6]. The algorithm should take into account the available resources and work to improve QoS. Based on practical considerations, the performance of the algorithm can be evaluated by resource utilization, load balancing, maximum completion time, execution cost, power consumption, reliability and other indicators.

To improve the computing capabilities of mobile devices, edge computing transfers computation-intensive applications from resource-constrained smart mobile devices to nearby edge servers with computational capabilities [40]. Bin Cao proposed a space–air–ground-integrated network (SAGIN)-IoV edge-cloud architecture based on software-defined networking (SDN) and network function virtualization (NFV) [36] that takes into consideration that the actual user needs to establish an optimization model. Yao proposed a big data-based heterogeneous Internet of Vehicles engineering cloud system resource allocation optimization algorithm [39]. Filip proposed a new model for scheduling microservices over heterogeneous cloud-edge environments [41]. This model aims to improve the resource utilization of edge computing equipment and reduce cost. Farid M proposed a new multi-objective scheduling algorithm with fuzzy resource utilization (FR-MOS) for scheduling scientific workflow based on the PSO method [6]. This algorithm's primary objective is to minimize cost and makespan in consideration of reliability constraints, where the constraint coefficient is determined by cloud resource utilization.

Existing resource-scheduling algorithms generally account for a variety of usage scenarios. In our proposed case study, we recognize that it is practical to employ these existing methods. In this article, service delay, resource utilization, load balancing and security are considered simultaneously and an optimization model is constructed.

3. Proposed Algorithm and Its Application

3.1. Parallel Communication Strategy

Choosing a suitable communication strategy is critical in parallel strategy since it facilitates information exchange between two groups, thereby enhancing their search capabilities. In this paper, two primary communication strategies are used. The first primary communication strategy is controlled by the communication control factor R. If the random number generated in the communication is greater than R, all groups will be matched in pairs and the following Taguchi communication will be performed:

1. Select the optimal solution of the two groups and select the appropriate orthogonal table based on the number of levels and factors. For example, the experiment is two-level if it contains two candidates and is seven-factor if each candidate in the experiment contains seven influencing factors. Conduct the experiment according to the orthogonal table and calculate the fitness value for each new individual.

2. In each dimension, calculate the fitness sum of the two levels separately. For each dimension, select the level with better fitness as a candidate. Combine the candidates to produce an optimal individual.

When the random number is less than R, the optimal solutions within all groups are compared with the global optimal solution using the following steps:

1. If it is worse than the global optimal solution, the intra-group optimal solution is replaced by the global optimal solution.

2. If it is better than or equal to the global optimal solution, a random mutation operation is performed.

Every individual has the potential to excel in some dimensions. The Taguchi method can efficiently excavate these excellent dimensions and then combine them together. The communication process described above will be visualized through an example. The fitness function is assumed to be:

$$f(x) = \sum_{i=1}^{n} x^2 \tag{9}$$

Suppose the search goal is to find the smallest fitness value. The two candidate individuals are shown in Table 1. The Taguchi orthogonal experiment is a two-level seven-factor experiment that employs the $L_8(2^7)$ orthogonal table shown in Equation (3). Table 2 depicts the specific operation process. The cumulative fitness value of the two candidate solutions in the table is calculated based on whether the solution is selected in the orthogonal table. In the first dimension of Table 2, the candidate solution x_2 was used in experiments 5 to 8, the cumulative fitness value for the first dimension of x_2 is the sum of the fitness values from 5 to 8 experiments.

Table 1. The position of the candidate individuals.

Position	**Dimension**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**Fitness Value**
x_1	0	1	1	0	0	1	0	3
x_2	1	0	0	1	1	0	1	4

Table 2. The Taguchi method is used to produce better individuals.

Experiment Number	**Dimension**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**Fitness Value**
1	0	1	1	0	0	1	0	3
2	0	1	1	1	1	0	1	5
3	0	0	0	0	0	0	1	1
4	0	0	0	1	1	1	0	3
5	1	1	0	0	1	1	1	5
6	1	1	0	1	0	0	0	3
7	1	0	1	0	1	0	0	3
8	1	0	1	1	0	1	1	5
x_1 cumulative fitness value	12	16	16	12	12	16	12	-
x_2 cumulative fitness value	16	12	12	16	16	12	16	-
Selected dimension source	x_1	x_2	x_2	x_1	x_1	x_2	x_1	-
Position of the new individual	0	0	0	0	0	0	0	0

In addition, this paper employs the population-merger communication strategy as the second primary communication strategy. In a swarm intelligence algorithm with parallel strategies, multiple subpopulations search independently and communicate with each other at intervals. The parallel strategy increases the diversity of the algorithms, but reduces the number of individuals in each subpopulation and some algorithms require more individual data for search. This conflict weakens the performance of the algorithm. After testing, the search performance of the original GSK algorithm was significantly reduced when the algorithm was divided into several subpopulations. In order to solve the above problem, the population-merger communication strategy was adopted. Specifically, each subpopulation searched independently in the early stage of the algorithm. Once a specific condition is met, the two adjacent subpopulations combine into one population. The newly formed population incorporates all the information from both subpopulations. Finally, before the end of the algorithm, all the individuals are merged into one population, which contains all the information of the original subpopulation. In the first stage of this paper, the initial GSK population was divided into four groups. In the second stage, the four groups were merged into two groups. In the third stage, the two groups are merged into one group. The condition of merger refers to the number of fitness functions.

3.2. Incorporate OBL into GSK

There are two primary steps involved in adding OBL to GSK. Using OBL, optimize the initial population as the first step. In the second step, the opposite population is generated to correct the convergence direction.

For the original algorithm (GSK), the initial population is randomly generated within a defined range. The initial individuals thus generated may be too far away from the global optimal position. The utilization of the OBL strategy enables the generation of a population closer to the optimal position, thereby facilitating more effective algorithm optimization. The steps in detail are as follows:

1. Initialize the population X = $\{x_1, x_2, \ldots, x_n\}$ randomly according to the defined range, n denotes the number of individuals in the population. Generate the opposing population $X^{op} = \{x_1^{op}, x_2^{op}, \ldots, x_n^{op}\}$ with the following formula:

$$x_{i,j}^{op} = a_j + b_j - x_{i,j} \qquad i = 1, 2, \ldots n; j = 1, 2, \ldots, D; \tag{10}$$

where a_j represents the upper bound of the current dimension and b_j represents the lower bound.

2. Select individuals with excellent fitness from $\{x_i, x_i^{op}\}$ and combine them into NP, taking NP as the initial population.

In the process of population updating, by using similar methods to generate the opposite population and for evaluation, the current population can be guaranteed to be closer to the global optimal position. The probability of generating an opposing population can be controlled by adjusting the jump rate r. The steps in detail are as follows:

1. After each update of the population, a random number is generated to compare with the jump rate r. If the random number is less than r, the opposite population of the current population will be generated, the following formula shows the process:

$$x_{i,j}^{op} = max_j + min_j - x_{i,j} \qquad i = 1, 2, \ldots n; j = 1, 2, \ldots, D; \tag{11}$$

where max_j represents the maximum value of the j dimension in the current population and min_j represents the minimum value.

2. Current and opposing populations are combined, and the fitness is tested separately. Then, the fittest individuals are selected from $\{x_i, x_i^{op}\}$.

In this paper, several different approaches are considered for integration with GSK and Figure 1 illustrates the process. In POGSK, the initial population is divided into four subpopulations after OBL optimization and the four subpopulations independently conduct GSK and communicate with each generation according to the Taguchi method. After updating the current population, the OBL operation is performed. The pseudocode for POGSK is shown in Algorithm 4. Moreover, in order to demonstrate the POGSK process more visually, Figure 2 shows its main processes.

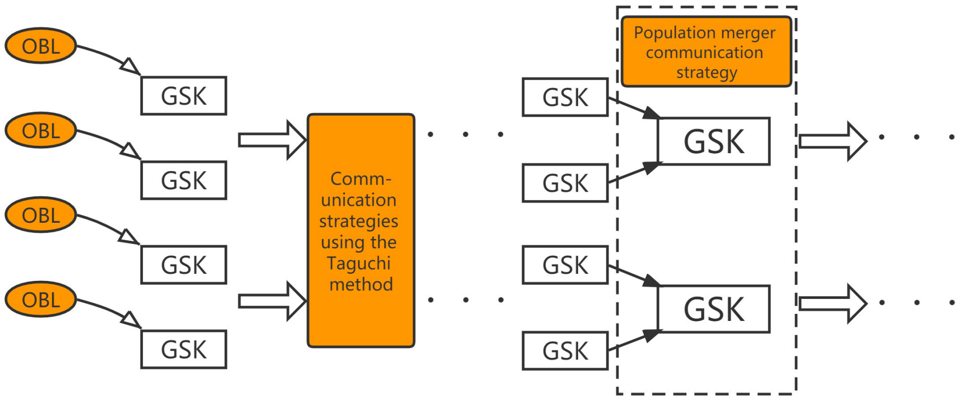

Figure 1. The approaches of POGSK.

Algorithm 4: POGSK pseudo-code

nfes = 0, t = 0, initialize parameters: MAX_{nefs}, K_f, K_r, K, R, L, r and G.
Creat a random initial population $\{x_i\}$, i = 1, 2, . . . , N.
Calculate the opposition population $\{x_i^{op}\}$ according to Equation (10), i = 1, 2, . . . , N.
Evaluate $f(x_i)$ and $f(x_i^{op})$, i = 1, 2, . . . , N.
Select NP fittest solutions from $\{x_i, x_i^{op}\}$ as the initial population. The population is divided into G groups. Update intra-group best and update global best.
while *nfes* < MAX_{nfex} **do**
  t = t + 1;// into the next generation
  for *g = 1: G* **do**
    compute the number of (gained and shared dimensions of both phases) using experience Equation (1) and Equation (2);
    //Junior gaining–sharing knowledge phase//
    //Senior gaining–sharing knowledge phase//
    Evaluate $f(x_i^{new})$, i = 1, 2, . . . , N
    for *i = 1: N/G* **do**
      if $f(x_i^{new}) < f(x_i^{old})$ **then**
        $x_i^{old} = x_i^{new}, f(x_i^{old}) = f(x_i^{new})$//update intra-group best
      end
      if $f(x_i^{new}) < f(x_G^{best})$ **then**
        $x_G^{best} = x_i^{new}, f(x_G^{best}) = f(x_i^{new})$//update global best
      end
    end
  end
  if *t = nL (n = 1, 2, 3, . . .)* **then**
    communicate between groups.
  end
  if *the first reduction group number condition is met* **then**
    four groups are combined into two groups
  end
  if *the second reduction group number condition is met* **then**
    two groups are combined into one group
  end
  if *rand < r* **then**
    Calculate the opposition population $\{x_i^{op}\}$ according to Equation (11), i = 1, 2, . . . , N.
    Select the fittest solutions from $\{x_i, x_i^{op}\}$ as the current population.
    Update intra-group best and update global best
  end
end

3.3. Apply the POGSK to Solve the Resource-Scheduling Problem in IoV

In order to reasonably allocate the resources of RSUs in IoV and enhance the QoS, this paper proposes the following mathematical model. Assume that there are multiple vehicles on the road and a total of n tasks are submitted simultaneously and each task contains four attributes: (1) the size of the task; (2) deadlines for tasks; (3) type and quantity of resources required; (4) the transfer time of the submitted work. The n tasks are represented as follows [36]:

$$T = \{T_i\}(i = 1, 2, .., n) \tag{12}$$

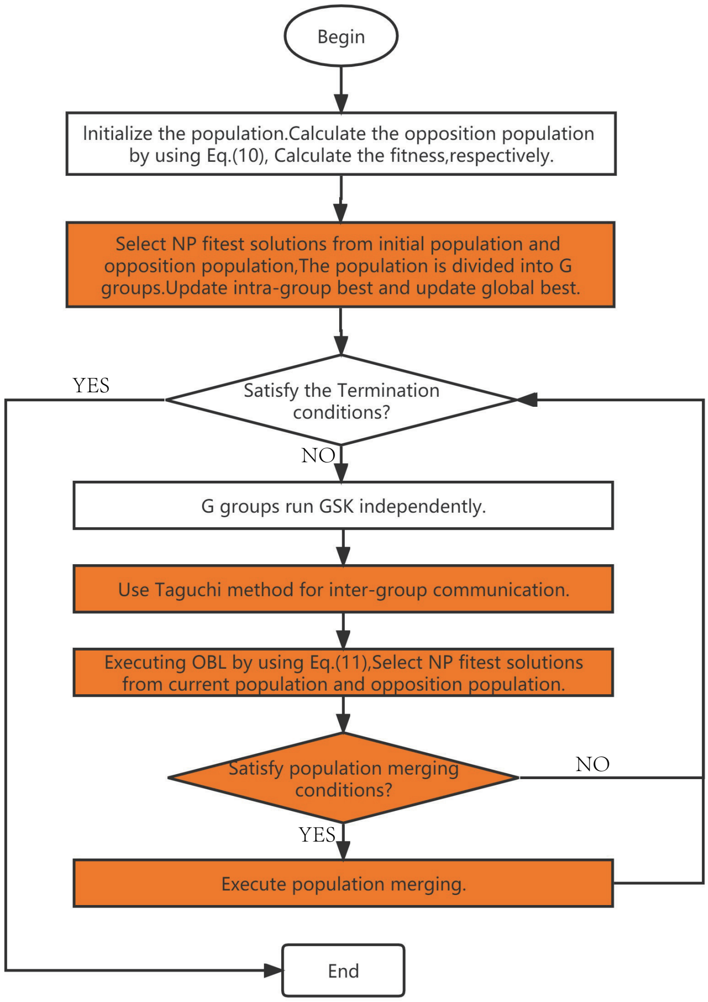

Figure 2. The flowchart of POGSK.

Suppose that there are m processing nodes in the current scenario. Each processing unit contains two attributes: (1) type and quantity of resources owned by the processing unit and (2) the processing capacity of the processing unit. The m processing nodes are represented as follows [36]:

$$P = \{P_j\}(i = 1, 2, .., m) \quad (13)$$

- Service delay
  In order to provide users with faster services, service latency should be as short as possible. A processing node can handle multiple tasks simultaneously, with different processing capabilities for each node. Then, the processing time of a task on the processing node is [36]

$$DOP_i^j = \frac{S_i}{H_j} \quad (14)$$

  where S_i represents the size of the task T_i and H_j represents the processing capability of the processing node P_j. Then, the time required for a processing node to complete all the tasks assigned to it is

$$PT_j = \sum_{i=1}^{n} DOP_i^j * C(i, j) \quad (15)$$

where C(i,j) is a binary value and indicates whether task T_i is assigned to node P_j, denoted as

$$C(i,j) = \begin{cases} 0, otherwise. \\ 1, T_i \ is \ assigned \ to \ P_j \end{cases} \tag{16}$$

The sum of the processing delays of all nodes is

$$F_T = \sum_{j=1}^{m} PT_j \tag{17}$$

- Resource utilization
  According to research, the energy consumption of the server in the idle state can account for more than 60% of the full load operation [42], which leads to a large amount of energy wasted on the idle server. So we want the roadside unit to be as resource-efficient as possible. The service request in the IoV requires the support of four kinds of computer resources, namely CPU, memory, disk and bandwidth. We need to pay attention to all four sources. Then, the total resource utilization is

$$F_U = \frac{\sum_{i=1}^{n} \sum_{k=1}^{4} CR(i,j)}{\sum_{j=1}^{m} \sum_{k=1}^{4} (PN_j * PR(j,k))} \tag{18}$$

  where CR(i,k) represents the number of $resources_k$ required by T_i, PR(j,k) represents the number of $resources_k$ owned by P_j and PN_j is a binary value indicating whether P_j is turned on or not (k = 1,2,3,4 represents four resources).

$$PN_j = \begin{cases} 0, otherwise. \\ 1, P_j \ is \ running \end{cases} \tag{19}$$

- Load balancing
  The service request in the IoV has different demands on different resources; it is easy to cause the load of different types of resources to be unbalanced. A valuable load-balancing technique in cloud computing can enhance the accuracy and efficiency of cloud computing performance [43]. So we want the processing unit to be as load-balanced as possible. Then, the utilization of $resource_k$ in P_j is

$$u_j^k = \frac{\sum_{i=1}^{n} (CR(i,j) * C(i,j))}{PR(j,k)} \tag{20}$$

  where C(i,j) is calculated using Equation (16). The mean of resource utilization of P_j is

$$MU_j = \frac{\sum_{k=1}^{4} u_j^k}{4} \tag{21}$$

  The variance of resource utilization of P_j is

$$VU_j = \frac{\sum_{k=1}^{4} (u_j^k - MU_j)}{4} \tag{22}$$

  The average resource utilization variance for all processing units is

$$F_N = \frac{\sum_{j=1}^{m} (VU_j^k * PN_j)}{z} \tag{23}$$

  where PN_j is calculated by Equation (19) and z represents the number of processing units opened.
- Security
  In the IoV, tasks must be completed on schedule to ensure safety, as service requests are made at high speeds. In real scenarios, the network latency and security of the task is important [6,44]. Task deadlines will be sent with task submissions and we want

as many tasks as possible to be completed on time. Then, the actual time required to complete the task is

$$ps_i = DOP_i^j + TL(i, j) \tag{24}$$

where DOP_i^j is calculated with Equation (14) and TL(i,j) represents the transmission buffer time from T_i to P_j. Then, whether the task is completed on time is expressed as a binary value:

$$S_i = \begin{cases} 0, cs_i < ps_i. \\ 1, cs_i \geq ps_i \end{cases} \tag{25}$$

where cs_i indicates the deadtime of T_i, which is uploaded when the task is submitted. Then, we express the degree of security as the successful execution rate of the task.

$$F_S = \frac{\sum_{i=1}^{n} S_i}{n} \tag{26}$$

Considering the above four objectives, we propose the following fitness function:

$$fitness = a * F_T + \frac{b}{F_U} + c * F_N + \frac{d}{F_S} \tag{27}$$

For processing unit P_j, the number of various resources required by all the tasks running on it is not permitted to exceed the number of resources owned by the unit. The workflow is shown in Figure 3. The constraint conditions are:

$$\begin{gathered} \sum_{i=1}^{n} (C(i, j) * CR(j, k)) < PR(j, k) \\ j = 1, 2, \ldots, m;\ k = 1, 2, 3, 4 \end{gathered} \tag{28}$$

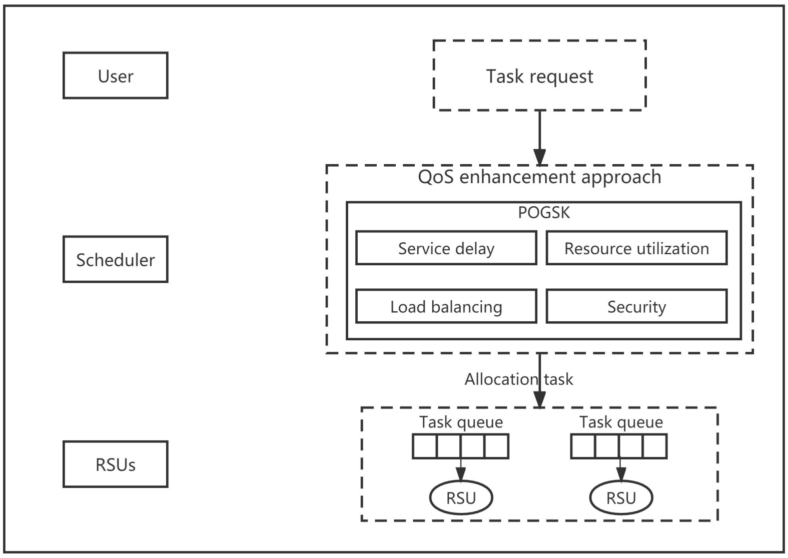

Figure 3. Scheduling model.

4. Results

4.1. Simulation Results on CEC2017

Single objective optimization algorithms are the basis of the complex optimization algorithm. It is considered effective to test with some classical mathematical functions. CEC2017 contains 30 benchmark functions to test the optimization ability of the algorithm. The F2 function was abandoned because of the dimension-setting problem. F1 and F3 are Unimodal Functions, F4–F10 are Simple Multimodal Functions, F11–F20 are Hybrid Functions, F21–F30 are Composition Functions. Set error = $(f_i - f_i^*)$ as the objective function, where f_i is the actual value of the ith test function and f_i^* is the minimum value of the ith test function. The optimization goal is to make the error as small as possible. Values of error and standard deviations less than 10^{-8} are considered as zero [35].

In this paper, POGSK is compared with the original algorithm GSK, PSO, DE and GWO. The Taguchi strategy in POGSK results in additional fitness function calls in each population generation. So for the sake of fairness, in this paper, the termination condition of the algorithm is set to the maximum number of function evaluations (NEFS) which is set to 10,000*problem_size. For example, the NEFS in a test with 30 variables is 300,000 times. The population size is set to 100 and the range of all test function solutions is set to [−100, 100]. Conduct 31 independent experiments each time to avoid special circumstances. The best results are marked in bold for all problems. The basic parameter settings of each algorithm are shown in Table 3.

Table 4 shows experimental results of POGSK, GSK and DE over 31 independent runs on 29 test functions of 10 variables under CEC2017. Table 5 shows the experimental results of POGSK, GWO and PSO. Compared with the original algorithm GSK, POGSK obtains excellent results in 26 test functions, five of which reach the minimum value of the test function. In contrast to DE, POGSK obtains excellent results on 23 functions. In contrast to GWO, POGSK obtains excellent results on 25 functions. In contrast to PSO, POGSK obtains excellent results on 25 functions. In addition, it is worth noting that POGSK obtained 9 times better results on functions 21–30. This shows that it has excellent search ability on Composition Functions.

Table 3. Parameter settings of each algorithm.

Algorithms	Parameters Settings
POGSK	G = 4, R = 0.5, L = 1, r = 0.1, K_f = 0.5, K_r = 0.9, K = 1
GSK	K_f = 0.5, K_r = 0.9, K = 1
PSO	V_{max} = 6, V_{min} = −6, wMax = 0.9, wMin = 0.2, c1 = c2 = 2
DE	$beta_{min}$ = 0.2, $beta_{max}$ = 0.8, pCR = 0.2
GWO	$\vec{a}$ = 2 (linearly decreased over iterations)

Table 4. Experimental results of POGSK, GSK and DE over 31 independent runs on 29 test functions of 10 variables under CEC2017.

Function	POGSK		GSK		DE	
	Mean	Std	Mean	Std	Mean	Std
1	**0**	0	**0**	0	1243.046	917.0439
3	**0**	0	**0**	0	1521.115	622.8467
4	**0**	0	**0**	0	6.037244	0.307901
5	17.48551	4.653999	20.50791	3.041351	**9.710514**	1.981759
6	**0**	0	0	0	**0**	0
7	29.31611	4.914915	30.33268	3.706216	**20.85221**	1.880978
8	17.00134	4.4744	19.45566	3.871004	**9.794002**	1.752965
9	**0**	0	**0**	0	**0**	0
10	930.7502	187.4053	1022.488	101.7212	492.0772	107.547
11	0.288859	0.459088	**0**	0	3.254768	0.650172
12	102.7016	87.50214	**80.9387**	62.50609	162,272.8	84,043.21
13	**4.39938**	2.902828	6.489874	1.622716	1247.31	1044.067
14	**0.87396**	0.883004	5.982104	2.986793	26.23902	20.65982
15	**0.15139**	0.27689	0.313974	0.29665	28.17154	24.51716
16	**0.99709**	2.036369	2.917096	4.299084	3.314234	1.910263
17	**1.82295**	3.702029	9.250251	6.796455	1.89604	0.899154

Table 4. *Cont.*

Function	POGSK		GSK		DE	
	Mean	Std	Mean	Std	Mean	Std
18	**1.67142**	4.206737	1.673515	5.02943	967.1823	585.524
19	**0.0497**	0.056623	0.104905	0.114554	27.60279	33.09174
20	0.446523	0.312835	0.414759	0.211807	**0**	0
21	176.3012	57.57345	193.0073	51.14463	**161.4336**	36.74808
22	**95.8506**	16.53658	100.3894	0.814027	97.40993	9.366758
23	**306.122**	2.840924	317.7665	3.307184	312.0515	2.093624
24	**306.697**	81.28064	341.4343	21.0113	316.2438	39.04365
25	**399.464**	8.161678	427.1743	21.27511	410.4391	9.406591
26	**296.774**	17.96053	300	2.99E−13	300.5042	52.62727
27	**389.217**	0.239349	389.5044	0.052422	389.5575	0.2579
28	**300**	2.67E−13	303.1151	17.34415	430.3714	69.26502
29	**243.926**	5.174707	248.3445	5.043812	263.9279	8.011281
30	**445.736**	38.26841	7363.998	38,494.24	13,511.91	7149.584
Win	-	-	26	15	23	18
Lose	-	-	3	14	6	11

Table 5. Experimental results of POGSK, PSO and GWO over 31 independent runs on 29 test functions of 10 variables under CEC2017.

Function	POGSK		GWO		PSO	
	Mean	Std	Mean	Std	Mean	Std
1	**0**	0	4,497,091	11,601,735	1159.397	1605.775
3	**0**	0	259.1001	389.8524	**0**	0
4	**0**	0	9.201156	5.959095	3.91603	12.5333
5	17.48551	4.653999	13.25237	7.436721	28.50068	9.743601
6	**0**	0	0.464612	0.881753	4.077067	3.713828
7	29.31611	4.914915	26.10298	8.270842	21.02474	6.645744
8	17.00134	4.4744	11.49256	4.369928	15.46999	6.8877
9	**0**	0	3.344128	8.094291	**0**	0
10	930.7502	187.4053	**455.7428**	250.51	793.989	304.649
11	0.288859	0.459088	19.59614	25.7201	23.76794	11.81411
12	102.7016	87.50214	480,552.9	684,130.6	12,243.2	11,168.19
13	**4.39938**	2.902828	8697.586	4995.097	7018.655	5919.406
14	**0.87396**	0.883004	1108.531	1650.036	78.09055	101.3466
15	**0.15139**	0.27689	1291.43	1574.817	238.5524	322.1582
16	**0.99709**	2.036369	77.88443	69.76017	212.4675	118.4518
17	**1.82295**	3.702029	44.01848	19.88944	43.69786	25.65489
18	**1.67142**	4.206737	28,538.6	14,703.86	8380.526	5920.313
19	**0.0497**	0.056623	4300.029	5439.223	619.0984	767.2034
20	0.446523	0.312835	51.83766	39.56463	63.11953	46.19121
21	176.3012	57.57345	208.0848	19.82006	173.3363	63.70194
22	**95.8506**	16.53658	103.3866	17.3914	102.3462	0.964004
23	**306.122**	2.840924	315.7259	7.263185	361.6049	72.55207
24	**306.697**	81.28064	337.3761	43.27372	348.4116	106.5231
25	**399.464**	8.161678	439.6042	13.4442	425.0482	23.00597
26	**296.774**	17.96053	355.5426	169.6211	367.0571	174.7731
27	**389.217**	0.239349	397.2865	17.07097	432.788	42.47196
28	**300**	2.67E−13	537.1552	99.1675	377.7298	48.39093
29	**243.926**	5.174707	275.9368	34.50598	308.0478	38.2051
30	**445.736**	38.26841	601,218.1	684,207.6	4893.086	3573.952
Win	-	-	25	26	25	28
Lose	-	-	4	3	4	1

Table 6 shows the experimental results of POGSK, GSK and DE over 31 independent runs on 29 test functions of 30 variables under CEC2017. Table 7 shows the experimental results of POGSK, GWO and PSO. Compared with the original algorithm GSK, POGSK obtains excellent results in 20 test functions, five of which reach the minimum value of the test function. In contrast to DE, POGSK obtains excellent results on 26 functions. Compared with GWO, POGSK obtains excellent results on 28 functions. In contrast to PSO, POGSK obtains excellent results on 25 functions. Furthermore, it is worth noting that POGSK obtained 7 times better results on functions 21–30. This again validates its excellent search ability on combinatorial functions.

Table 6. Experimental results of POGSK, GSK and DE over 31 independent runs on 29 test functions of 30 variables under CEC2017.

Function	POGSK		GSK		DE	
	Mean	Std	Mean	Std	Mean	Std
1	**0**	0	**0**	0	565.1619	467.5469
2	0.1710225	0.951944	**1.27E−07**	3.59E−07	78,485.43	11,969.56
3	**2.3896**	2.00382	8.0680906	15.72576	88.87305	1.39142
4	**32.70072**	9.603168	157.06186	10.58822	129.9884	9.275313
5	4.78E−05	7.06E−05	6.48E−07	1.44E−06	**0**	0
6	**110.9293**	59.25905	184.5384	10.80525	163.7576	10.61515
7	**31.70902**	9.949111	157.84376	8.553038	130.7675	8.930799
8	1.3961111	0.964445	**0**	0	**0**	0
9	6640.146	352.1547	6773.1382	280.2139	5270.445	300.3687
10	**15.3527**	8.172784	32.010214	37.36949	110.2576	10.49883
11	7473.1514	4191.488	**5872.033**	3954.488	4,064,732	1,326,017
12	**68.10246**	33.29562	97.603211	38.84956	146,977.1	70,134.12
13	**47.58591**	13.26488	56.580342	4.594124	59,985.92	29,927.38
14	32.795131	16.27245	**16.78781**	10.92508	20,442.66	11,240.16
15	**225.2992**	208.1208	795.64371	166.3631	588.2006	138.6957
16	**47.13605**	16.82453	198.98593	92.83664	162.9097	38.25502
17	104.56587	64.00951	**36.67658**	8.700766	424,098.9	151,960.3
18	22.242391	8.28202	**10.97738**	4.416147	19,255.23	9427.15
19	**62.56111**	50.53387	68.739206	65.88684	173.8984	49.83044
20	**234.2304**	20.45764	349.17774	8.751155	332.6374	8.650852
21	100.07935	0.441787	**100**	4.52E−13	1202.919	888.6275
22	**374.781**	6.959946	463.52232	59.16068	480.7536	7.607155
23	**444.1957**	6.710317	567.06927	29.46917	582.855	8.472587
24	385.91772	1.780967	386.92029	0.19508	387.3266	0.082705
25	**685.0133**	500.7473	1035.9259	330.0292	2326.135	82.41891
26	496.61134	7.400898	**492.5301**	7.147451	509.6858	2.319662
27	**303.3373**	18.54975	321.05968	43.75981	427.0414	13.69322
28	**460.2316**	41.83653	563.47372	111.3137	729.1611	76.50307
29	**2045.674**	80.2348	2080.6714	121.1253	20,973.93	6734.785
Win	-	-	20	12	26	15
Lose	-	-	9	17	3	14

To demonstrate the algorithm performance of each CEC2017 test problem for multiple numbers of objective function evaluation allowances, we conducted further experiments setting the algorithm termination conditions to 0.1*maximum NFES, 0.3*maximum NFES and 0.5*maximum NFES. Continue using the basic parameters of each algorithm shown in Table 3 without change. Table 8 shows experimental results of POGSK, GSK and PSO over 31 independent runs on 10 variables for multiple numbers of objective function evaluation. Table 9 shows experimental results of POGSK, GWO and DE. For presentation purposes, only the mean fitness values for 31 independent runs of the algorithm are shown in Tables 8 and 9. It can be observed that in comparison with the 1*max NFES termination condition, POGSK still shows a strong optimization performance when NFES is reduced. It is worth noting that POGSK shows a slight decrease in optimization performance compared to the PSO algorithm. In particular, when NFES = 0.1*Max NFES, POGSK outperforms PSO for only 18 functions. We believe this performance is reasonable because the optimization capability of the POGSK algorithm is not fully utilized when NFES is reduced.

Table 7. Experimental results of POGSK, GWO and PSO over 31 independent runs on 29 test functions of 30 variables under CEC2017.

Function	POGSK		GWO		PSO	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
1	**0**	0	1.065E+09	9.63E+08	2399.513	3445.498
3	0.1710225	0.951944	28,199.902	10,867.38	0.059275	0.066189
4	**2.3896**	2.00382	160.71215	47.20044	60.0721	28.92923
5	**32.70072**	9.603168	78.084404	18.62267	152.9662	25.34413
6	4.78E−05	7.06E−05	4.5010838	2.508565	31.18878	7.532825
7	**110.9293**	59.25905	136.85245	23.06388	117.7735	23.32769
8	**31.70902**	9.949111	73.037706	18.75822	115.3826	25.07128
9	1.3961111	0.964445	524.43328	280.1444	2084.344	424.3998
10	6640.146	352.1547	**2822.749**	522.9234	3384.133	730.3928
11	**15.3527**	8.172784	296.75347	137.9677	90.02398	17.27491
12	7473.1514	4191.488	28,314,824	40,351,462	49,597.69	29,308.71
13	**68.10246**	33.29562	5,377,176.9	23,990,313	11,223.13	12,434.89
14	**47.58591**	13.26488	109,516.46	312,614.9	7271.957	5769.247
15	32.795131	16.27245	309,301.92	786,455	6660.235	7995.343
16	**225.2992**	208.1208	673.1389	239.9596	1021.81	208.1095
17	**47.13605**	16.82453	224.34632	107.809	518.3412	162.6259
18	104.56587	64.00951	449,259.69	471,076.9	113,207.1	84,886.55
19	22.242391	8.28202	680,051.23	1,796,562	10,046.08	14,864.67
20	**62.56111**	50.53387	316.20714	113.958	425.1631	129.9895
21	**234.2304**	20.45764	269.79579	15.67731	324.6484	26.97333
22	100.07935	0.441787	2207.2882	1473.069	1253.874	1862.064
23	**374.781**	6.959946	430.27783	38.0415	708.1221	116.4942
24	**444.1957**	6.710317	514.80412	50.32647	778.1768	72.14411
25	385.91772	1.780967	455.40908	24.09939	**379.621**	6.204345
26	**685.0133**	500.7473	1924.7685	304.0471	3503.61	1534.257
27	496.61134	7.400898	533.23588	12.11012	494.1057	71.15496
28	**303.3373**	18.54975	562.25313	62.77035	371.5156	61.83288
29	**460.2316**	41.83653	781.81747	149.2681	917.2247	292.7391
30	**2045.674**	80.2348	4,740,432.9	4,505,133	3024.257	4562.477
Win	-	-	28	26	25	26
Lose	-	-	1	3	4	3

To better visualize the performance of POGSK, the convergence curves of the nine benchmark functions on 10 variables are shown in Figure 4 and the convergence curves of the nine benchmark functions on 30 variables are shown in Figure 5. The convergence curves of POGSK on functions 1–10 of 10 variables are not shown much because unimodal functions and simple multimodal functions are too simple to distinguish the search capability in the case of a few variables. We can see that in the middle and late stages of the algorithm, POGSK shows its powerful search ability to effectively avoid local optima. This reflects the fact that the addition of the OBL strategy and parallel strategy significantly enhances the search capability of the original algorithm. Through the above experimental comparison, it can be determined that POGSK has better capability in the CEC2017 test suite compared with GSK, PSO, DE and GWO.

Table 8. Experimental results of POGSK, GSK and PSO over 31 independent runs on 10 variables for multiple numbers of objective function evaluation.

Function	POGSK			GSK			PSO		
	0.1*Max	**0.3*Max**	**0.5*Max**	**0.1*Max**	**0.3*Max**	**0.5*Max**	**0.1*Max**	**0.3*Max**	**0.5*Max**
1	65,168.64	**0.008729**	**9.00E−08**	198,621.8	0.148295	1.24E−07	**1507.631**	1841.959	1481.528
3	1072.874	0.020955	1.37E−09	1258.366	0.150523	3.43E−08	**3.622942**	**1.32E−06**	**0**
4	**4.46869**	0.156832	0.000642	4.782125	**0.0252**	**1.26E−05**	6.788089	9.600704	4.619372
5	35.91597	28.17189	22.91988	35.00533	29.01222	23.6099	31.58184	30.49059	29.62402
6	**0.67712**	**0.000231**	**3.46E−07**	1.131835	0.001475	9.12E−06	9.346549	4.068655	5.220518
7	46.40205	36.92188	34.41898	50.00724	37.96985	34.8906	**28.87082**	**24.42772**	**23.73282**
8	34.3193	26.88238	22.69993	36.83458	28.55074	26.63988	17.84513	15.91932	**14.53922**
9	**0.335181**	**0**	**0**	1.190911	0	**0**	17.39665	6.587552	6.853679
10	1532.095	1315.312	1128.588	1525.435	1311.749	1192.912	837.2588	806.28	762.448
11	**11.17021**	**5.209862**	**1.532807**	12.14115	5.79704	3.51912	26.16613	28.79452	25.05342
12	91,453.6	**486.7377**	199.4322	164,573.9	617.8236	**179.6806**	**30,604.58**	11,899.81	16,792.79
13	**47.41327**	12.20971	**8.403044**	60.55603	**11.71106**	9.488802	6398.473	9347.321	6564.624
14	**27.46006**	19.86859	**11.84031**	28.9843	**18.29269**	15.361	1203.128	558.986	266.1244
15	**9.511309**	2.529362	**0.612543**	10.67048	**2.489827**	0.667059	2532.754	1061.556	548.5125
16	**83.47293**	**19.34939**	**4.684275**	91.10372	36.5609	13.88171	216.4358	208.4391	229.0174
17	71.5188	30.70673	18.47544	85.25013	44.12263	29.57548	**49.02151**	50.02125	45.54325
18	**74.44131**	**14.67361**	5.338491	162.1569	15.29351	**4.686504**	13,029.71	8474.698	8372.767
19	**6.425225**	**1.749615**	**0.682905**	7.216083	1.944351	0.892702	3491.462	2753.934	2199.245
20	68.77453	11.38535	1.397734	81.44863	21.28738	3.679158	97.71285	83.91263	75.89307
21	188.5297	184.346	189.9476	209.0319	202.3022	178.1039	**171.7462**	**178.0085**	189.2755
22	**103.8606**	100.5495	**100.2226**	105.9777	102.3459	101.2098	131.0425	**96.96854**	135.8181
23	335.3295	325.8999	317.8954	335.7887	327.2087	323.999	377.3523	369.7106	371.4575
24	357.042	341.9053	316.7562	361.736	347.599	348.096	**330.8915**	**320.122**	347.5205
25	**403.4286**	**409.5803**	**404.0327**	420.6563	426.9033	422.8455	415.486	420.797	423.8288
26	302.3367	**296.7742**	296.7742	**301.0423**	300	300	395.6739	392.4169	469.6308
27	391.4686	**389.2144**	**389.2638**	**391.1973**	389.4444	389.4419	446.6022	436.6223	439.4359
28	**383.6587**	**300.0077**	303.8159	432.3408	312.2833	**300**	405.3951	389.2501	350.7108
29	310.7228	**273.0201**	**257.9113**	311.5873	277.9013	266.0836	319.5175	320.2717	321.4858
30	11,6124.7	**735.7863**	508.8163	169,420.9	26,998.33	**474.5937**	**73,193.34**	14,707.45	9126.257
win	-	-	-	25	24	23	18	22	24
lose	-	-	-	4	5	6	11	7	5

Table 9. Experimental results of POGSK, GWO and DE over 31 independent runs on 10 variables for multiple numbers of objective function evaluation.

Function	POGSK			GWO			DE		
	0.1*Max	**0.3*Max**	**0.5*Max**	**0.1*Max**	**0.3*Max**	**0.5*Max**	**0.1*Max**	**0.3*Max**	**0.5*Max**
1	65,168.64	**0.008729**	**9.00E−08**	2,346,406	1417856	9.565626	16,135,020	20,544.91	4898.193
3	1072.874	0.020955	1.37E−09	2051.369	948.2263	0.752611	14,530.99	8845.123	5727.654
4	**4.46869**	0.156832	0.000642	22.33487	11.35453	29.34763	12.85429	6.929505	6.505718
5	35.91597	28.17189	22.91988	**18.45571**	**12.63573**	**13.74093**	30.60479	18.81837	14.40469
6	**0.67712**	**0.000231**	**3.46E−07**	1.228019	0.823463	3.763687	1.810073	0.001324	6.07E−07
7	46.40205	36.92188	34.41898	37.21266	29.32247	506.0761	44.70667	30.20338	25.16062
8	34.3193	26.88238	22.69993	**15.91906**	**15.30446**	21.20127	31.77074	20.53481	14.54278
9	**0.335181**	**0**	**0**	6.672216	8.604824	522,798.1	41.7464	0.017794	2.17E−07
10	1532.095	1315.312	1128.588	**677.9308**	**573.4273**	11,452.42	1210.891	851.5756	**702.4145**

Table 9. *Cont.*

Function	POGSK			GWO			DE		
	0.1*Max	**0.3*Max**	**0.5*Max**	**0.1*Max**	**0.3*Max**	**0.5*Max**	**0.1*Max**	**0.3*Max**	**0.5*Max**
11	**11.17021**	**5.209862**	**1.532807**	30.60828	31.73361	1246.452	47.17272	8.057127	5.097437
12	91,453.6	**486.7377**	199.4322	1,407,448	643,050.9	2140.969	6,603,038	1,351,308	594,344.9
13	**47.41327**	12.20971	**8.403044**	11065.19	12,549.47	90.00417	16,654.35	4276.951	1909.666
14	**27.46006**	19.86859	**11.84031**	2450.4	1269.145	52.02613	961.5601	184.9314	110.3554
15	**9.511309**	2.529362	**0.612543**	5154.414	3513.667	25,350.52	1426.507	236.2673	121.9112
16	**83.47293**	**19.34939**	**4.684275**	118.3694	120.0137	4220.433	96.57352	22.84995	9.823819
17	71.5188	30.70673	18.47544	68.63565	61.1111	60.51609	53.16403	29.56227	**14.18701**
18	**74.44131**	**14.67361**	5.338491	24,579.32	25,987.05	201.3137	41,806.47	6779.277	2723.725
19	**6.425225**	**1.749615**	**0.682905**	9683.89	7409.389	106.6633	2030.003	296.4361	199.3466
20	68.77453	11.38535	1.397734	88.42235	71.2303	317.6138	**37.92879**	**4.11068**	**0.002431**
21	188.5297	184.346	189.9476	211.797	205.8708	342.6138	209.201	179.1673	**172.5472**
22	**103.8606**	100.5495	**100.2226**	111.9155	108.5442	436.9162	126.1442	102.526	101.7367
23	335.3295	325.8999	317.8954	**323.4811**	**317.4113**	369.5282	332.2919	319.7871	**316.0538**
24	357.042	341.9053	316.7562	354.3611	335.1408	398.7922	361.1658	352.334	343.8135
25	**403.4286**	**409.5803**	**404.0327**	439.5899	433.9101	552.888	450.7069	431.3259	420.7792
26	302.3367	**296.7742**	296.7742	370.8195	374.8314	**275.9333**	541.4581	399.4515	350.68
27	391.4686	**389.2144**	**389.2638**	396.9011	394.4558	852,804.2	397.4228	391.6064	390.2805
28	**383.6587**	**300.0077**	303.8159	551.6053	539.1753	3358.777	549.7038	511.7699	488.9492
29	310.7228	**273.0201**	**257.9113**	**299.4216**	291.2249	3191.632	345.5497	296.2041	280.3842
30	116,124.7	**735.7863**	508.8163	491,702	795,289.2	255,507.2	285,465.2	70,612.38	32,384.37
win	-	-	-	21	23	26	22	21	21
lose	-	-	-	8	6	3	7	8	8

4.2. Simulation Results on Resource-Scheduling Problems

In this paper, we used the fitness function proposed in Section 3.3 to test the optimized performance of POGSK in real scenarios. We consider the construction of an edge processing system consisting of ten processing units. The size of the tasks to be processed is a random distribution in the interval $(0, 5] \times 10^6$ instructions. The maximum evaluation times were set as 300,000 times and the population size as 100. In order to avoid exceeding the constraint, the constraint test is carried out when the solution of the algorithm enters the fitness function. Specific constraint testing steps are as follows:

1. Each individual is represented as $X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$, the assignment list $\{k_1, k_2, \ldots, k_m\}$ is obtained by rounding each dimension of the individual. K_i = b indicates that task i is assigned to node b. All nodes are traversed to find idle nodes and all nodes whose resource utilization is lower than 50%. The idle queues F and low resource utilization queues L are established, respectively. The node is traversed and the over-allocated node is found. Queue E_j is established for the tasks on this node.

2. The tasks in queue E_j are redistributed to the nodes in queue L or to F if L is empty. After each redistribution, the current node is checked to see if it is over-allocated. If not, the current operation is stopped and the process returns to step 3 until all nodes have been traversed. Based on the above results, the individuals need to be adjusted. For example, $x_{i,3} = 2.1$ and $k_3 = 2$. After the constraint test, k_3 is adjusted to 6, then $x_{i,3}$ is updated randomly in the range [5.6, 6.4].

In this paper, we randomly generated 11 independent scenarios which submitted 30 tasks to the processing unit simultaneously. The best results are marked in bold for all scenarios. Table 10 shows other experimental parameters. These experimental parameters control the scene setting in the experiment. Each experiment was independently run 20 times. Table 11 shows the experimental performance of POGSK with GSK, GWO and PSO. You can see that out of the 11 experiments, POGSK won nine times. POGSK achieved excellent results in 9 scenarios compared to GSK. Compared with PSO, POGSK achieved excellent results in 9 scenarios. Compared with GWO, excellent results were obtained in 11 scenarios. This is due to the use of the parallel strategy and the OBL strategy. The Taguchi communication strategy allows the original GSK to effectively avoid falling into a local optimum. The population-merging communication strategy allows POGSK to not weaken algorithm performance due to the reduction in the number of individuals in the subpopulation. The use of the OBL strategy corrects the direction of convergence of the algorithm and increases the speed of convergence.

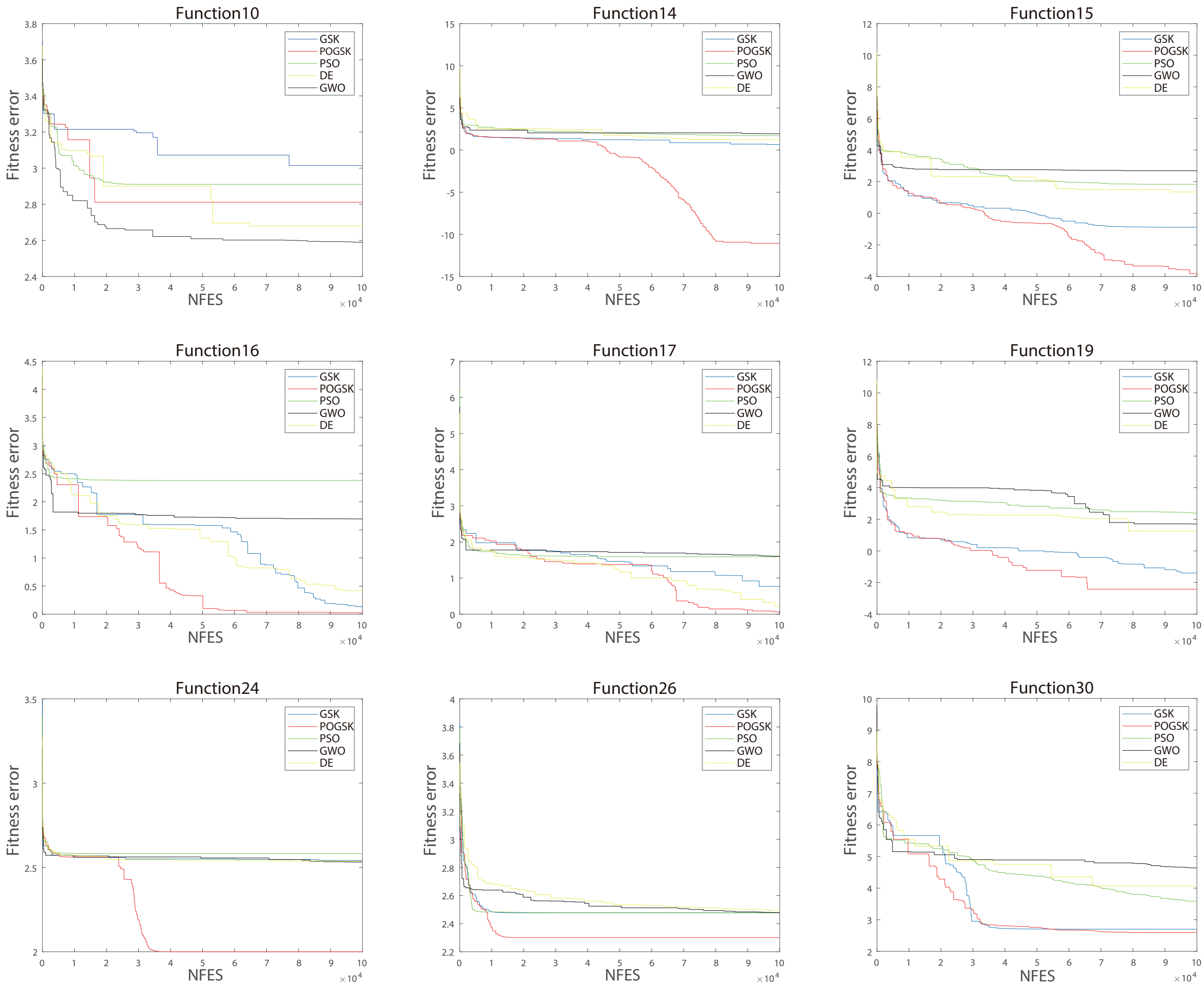

Figure 4. Convergence curves of 9 functions on 10 variables.

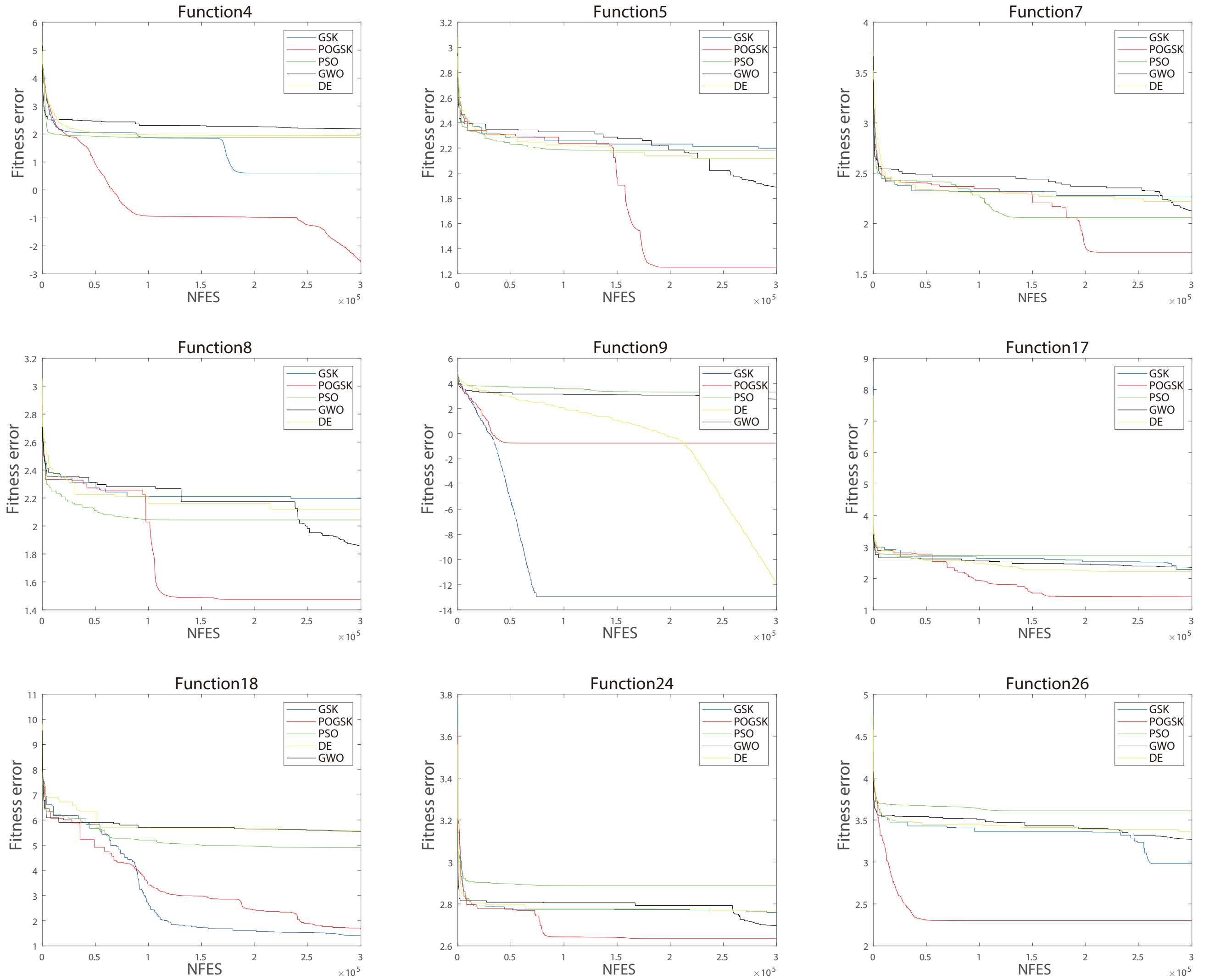

Figure 5. Convergence curves of 9 functions on 30 variables.

Figure 6 shows the fitness function value as the number of test function call changes. We can still see that POGSK performs well in avoiding local optimality. It shows that POGSK also performs well in constrained realistic optimization problems.

Table 10. Experimental parameter settings.

Symbols	Descriptions	Values
S_i	Size of the ith task	$(0, 5] \times 10^6$ instr
H_j	Processing capacity of the jth processing unit	$[0.5, 2] \times 10^6$ instr/ms
CR(i,k)	The number of $resources_k$ required by T_i	[0, 5]
PR(i,k)	The number of $resources_k$ owned by P_j	[5, 25]
TL(i,j)	The transmission buffer time from T_i to P_j	[0, 3] ms
cs_i	Deadtime of T_i	S_i + [0, 3] ms

Table 11. A total of 30 tasks were assigned to 10 processing unit experiments.

Scenario	POGSK		GSK		PSO		GWO	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
1	**5.718209**	0.144952	5.855939	0.135115	6.053149	0.248256	8.991358	0.330173
2	**6.329143**	0.340846	6.485126	0.241881	6.265463	0.270268	9.997938	0.317971
3	**5.981696**	0.163071	6.003088	0.158414	6.339609	0.287226	9.344461	0.246512
4	**5.212316**	0.204313	5.36799	0.227842	5.917752	0.393545	8.974101	0.311377
5	**5.245472**	0.162959	5.326294	0.247776	5.428665	0.832814	9.157856	0.251572
6	**5.624188**	0.115448	5.789302	0.231226	6.21364	0.559447	9.555559	0.421648
7	**6.661226**	0.325418	6.793765	0.249867	6.975366	0.409818	10.00448	0.244958
8	5.158754	0.278578	**4.992622**	0.156462	5.53684	0.151473	8.233507	0.457459
9	**5.567888**	0.096427	5.70977	0.105788	5.874641	0.139983	8.946471	0.32081
10	**5.828361**	0.208058	5.929272	0.293122	6.330107	0.47567	9.343315	0.307779
11	5.677546	0.008842	5.558177	0.175736	**5.274045**	0.673208	9.222462	0.412783
Win	-	-	9	6	9	9	11	9
Lose	-	-	2	5	2	2	0	2

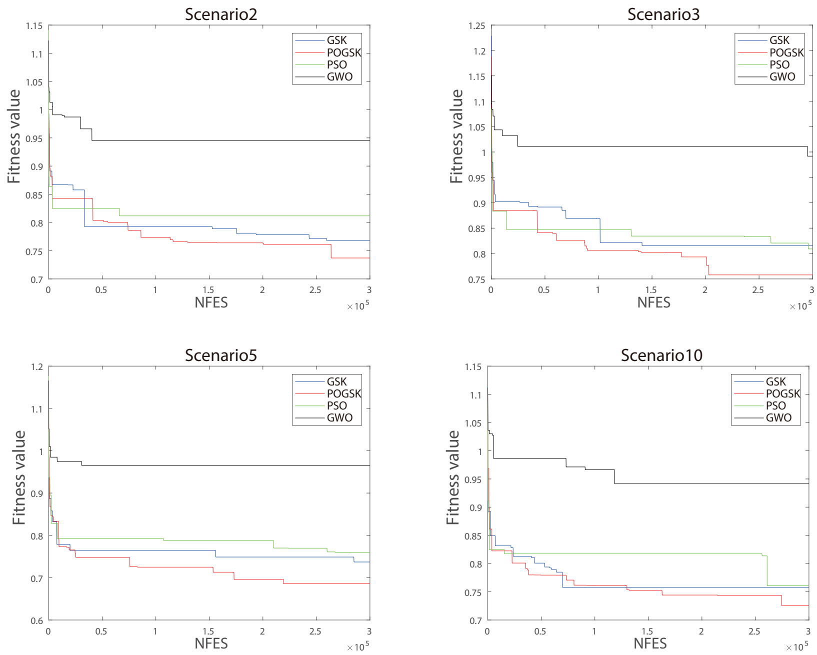

Figure 6. Convergence curves of four situations.

5. Conclusions

In this paper, the POGSK algorithm is proposed to solve the resource-scheduling problem in the IoV. Based on the original algorithm, POGSK uses OBL and parallel strategy. The information exchange of subpopulations uses the Taguchi strategy and the population-merger strategy. By testing with the original algorithm and some classical algorithms on CEC2017, it is shown that the new algorithm has stronger searching ability. Then, we applied POGSK to the resource-scheduling problem and carried out the simulation test, which also showed better results.

In the future, we can continue to improve the inter-group communication strategy and enhance the search capability of the algorithm. We can also study the application of POGSK in multi-objective problems, engineering optimization problems and binary optimization problems. We believe the new algorithm can also achieve better results.

Author Contributions: Conceptualization, J.-S.P.; methodology, J.-S.P. and L.-F.L.; software, J.-S.P. and L.-F.L.; validation, L.-F.L. and S.-C.C.; investigation, L.-F.L.; resources, J.-S.P. and P.-C.S.; data curation, G.-G.L. and P.-C.S.; writing—original draft preparation, L.-F.L.; writing—review and editing, J.-S.P. and P.-C.S.; supervision, G.-G.L.; project administration, S.-C.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets generated for this study are available on request to the corresponding author.

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

References

1. Wang, H.; Wang, W.; Cui, Z.; Zhou, X.; Zhao, J.; Li, Y. A new dynamic firefly algorithm for demand estimation of water resources. *Inf. Sci.* **2018**, *438*, 95–106. [CrossRef]
2. Song, B.; Wang, Z.; Zou, L. An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve. *Appl. Soft Comput.* **2021**, *100*, 106960. [CrossRef]
3. Chai, Q.w.; Chu, S.C.; Pan, J.S.; Hu, P.; Zheng, W.M. A parallel WOA with two communication strategies applied in DV-Hop localization method. *EURASIP J. Wirel. Commun. Netw.* **2020**, *2020*, 50. [CrossRef]
4. Wu, J.; Xu, M.; Liu, F.F.; Huang, M.; Ma, L.; Lu, Z.M. Solar Wireless Sensor Network Routing Algorithm Based on Multi-Objective Particle Swarm Optimization. *J. Inf. Hiding Multim. Signal Process.* **2021**, *12*, 1–11.
5. Deng, W.; Xu, J.; Song, Y.; Zhao, H. Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem. *Appl. Soft Comput.* **2021**, *100*, 106724. [CrossRef]
6. Farid, M.; Latip, R.; Hussin, M.; Hamid, N.A.W.A. Scheduling scientific workflow using multi-objective algorithm with fuzzy resource utilization in multi-cloud environment. *IEEE Access* **2020**, *8*, 24309–24322. [CrossRef]
7. Mohamed, A.W.; Hadi, A.A.; Mohamed, A.K. Gaining–sharing knowledge based algorithm for solving optimization problems: A novel nature-inspired algorithm. *Int. J. Mach. Learn. Cybern.* **2020**, *11*, 1501–1529. [CrossRef]
8. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the MHS'95, Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
9. Song, P.C.; Chu, S.C.; Pan, J.S.; Yang, H. Simplified Phasmatodea population evolution algorithm for optimization. *Complex Intell. Syst.* **2022**, *8*, 2749–2767. [CrossRef]
10. Pan, J.S.; Zhang, L.G.; Wang, R.B.; Snášel, V.; Chu, S.C. Gannet optimization algorithm: A new metaheuristic algorithm for solving engineering optimization problems. *Math. Comput. Simul.* **2022**, *202*, 343–373. [CrossRef]
11. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
12. Chu, S.C.; Tsai, P.W.; Pan, J.S. Cat swarm optimization. In Proceedings of the PRICAI 2006: Trends in Artificial Intelligence: 9th Pacific Rim International Conference on Artificial Intelligence, Guilin, China, 7–11 August 2006; Proceedings 9; Springer: Berlin/Heidelberg, Germany, 2006; pp. 854–858.
13. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
14. Storn, R.; Price, K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341. [CrossRef]
15. Han, K.H.; Kim, J.H. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Trans. Evol. Comput.* **2002**, *6*, 580–593. [CrossRef]
16. Hashim, F.A.; Hussain, K.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W. Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. *Appl. Intell.* **2021**, *51*, 1531–1551. [CrossRef]
17. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
18. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
19. Rao, R.V.; Savsani, V.J.; Vakharia, D. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput.-Aided Des.* **2011**, *43*, 303–315. [CrossRef]
20. Mohamed, A.W.; Abutarboush, H.F.; Hadi, A.A.; Mohamed, A.K. Gaining–sharing knowledge based algorithm with adaptive parameters for engineering optimization. *IEEE Access* **2021**, *9*, 65934–65946. [CrossRef]
21. Chu, S.C.; Roddick, J.F.; Pan, J.S. A parallel particle swarm optimization algorithm with communication strategies. *J. Inf. Sci. Eng* **2005**, *21*, 809–818.
22. Harada, T.; Alba, E. Parallel genetic algorithms: A useful survey. *ACM Comput. Surv. CSUR* **2020**, *53*, 1–39. [CrossRef]
23. Cai, D.; Lei, X. A New Evolutionary Algorithm Based on Uniform and Contraction for Many-objective Optimization. *J. Netw. Intell.* **2017**, *2*, 171–185.
24. Tsai, P.W.; Pan, J.S.; Chen, S.M.; Liao, B.Y. Enhanced parallel cat swarm optimization based on the Taguchi method. *Expert Syst. Appl.* **2012**, *39*, 6309–6319. [CrossRef]
25. Tsai, J.T.; Liu, T.K.; Chou, J.H. Hybrid Taguchi-genetic algorithm for global numerical optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 365–377. [CrossRef]

26. Jiang, S.J.; Chu, S.C.; Zou, F.M.; Shan, J.; Zheng, S.G.; Pan, J.S. A parallel Archimedes optimization algorithm based on Taguchi method for application in the control of variable pitch wind turbine. *Math. Comput. Simul.* **2023**, *203*, 306–327. [CrossRef]
27. Mahdavi, S.; Rahnamayan, S.; Deb, K. Opposition based learning: A literature review. *Swarm Evol. Comput.* **2018**, *39*, 1–23. [CrossRef]
28. Yu, X.; Xu, W.; Li, C. Opposition-based learning grey wolf optimizer for global optimization. *Knowl.-Based Syst.* **2021**, *226*, 107139. [CrossRef]
29. Dhargupta, S.; Ghosh, M.; Mirjalili, S.; Sarkar, R. Selective opposition based grey wolf optimization. *Expert Syst. Appl.* **2020**, *151*, 113389. [CrossRef]
30. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M. Opposition-based differential evolution. *IEEE Trans. Evol. Comput.* **2008**, *12*, 64–79. [CrossRef]
31. Deng, W.; Shang, S.; Cai, X.; Zhao, H.; Song, Y.; Xu, J. An improved differential evolution algorithm and its application in optimization problem. *Soft Comput.* **2021**, *25*, 5277–5298. [CrossRef]
32. Wang, H.; Wu, Z.; Rahnamayan, S.; Liu, Y.; Ventresca, M. Enhancing particle swarm optimization using generalized opposition-based learning. *Inf. Sci.* **2011**, *181*, 4699–4714. [CrossRef]
33. Ewees, A.A.; Abd Elaziz, M.; Houssein, E.H. Improved grasshopper optimization algorithm using opposition-based learning. *Expert Syst. Appl.* **2018**, *112*, 156–172. [CrossRef]
34. Mohamed, A.W.; Hadi, A.A.; Mohamed, A.K.; Awad, N.H. Evaluating the performance of adaptive gainingsharing knowledge based algorithm on CEC 2020 benchmark problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.
35. Awad, N.; Ali, M.; Liang, J.; Qu, B.; Suganthan, P. *Problem Definitions and Evaluation Criteria for The CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*; Technical Report; Nanyang Technological University: Singapore, 2016.
36. Cao, B.; Zhang, J.; Liu, X.; Sun, Z.; Cao, W.; Nowak, R.M.; Lv, Z. Edge–Cloud Resource Scheduling in Space–Air–Ground-Integrated Networks for Internet of Vehicles. *IEEE Internet Things J.* **2022**, *9*, 5765–5772. [CrossRef]
37. Ðurasević, M.; Jakobović, D. Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: A survey. *Artif. Intell. Rev.* **2022**, *56*, 3181–3289. [CrossRef]
38. Singh, S.; Chana, I. QoS-aware autonomic resource management in cloud computing: A systematic review. *ACM Comput. Surv. CSUR* **2015**, *48*, 1–46. [CrossRef]
39. Yao, J. Research on Optimization Algorithm for Resource Allocation of Heterogeneous Car Networking Engineering Cloud System Based on Big Data. *Math. Probl. Eng.* **2022**, *2022*, 1079750. [CrossRef]
40. Wang, Q.; Guo, S.; Liu, J.; Yang, Y. Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing. *Sustain. Comput. Inform. Syst.* **2019**, *21*, 154–164. [CrossRef]
41. Filip, I.D.; Pop, F.; Serbanescu, C.; Choi, C. Microservices scheduling model over heterogeneous cloud-edge environments as support for IoT applications. *IEEE Internet Things J.* **2018**, *5*, 2672–2681. [CrossRef]
42. Guo, M.; Li, L.; Guan, Q. Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems. *IEEE Access* **2019**, *7*, 78685–78697. [CrossRef]
43. Ullah, A.; Nawi, N.M.; Ouhame, S. Recent advancement in VM task allocation system for cloud computing: Review from 2015 to2021. *Artif. Intell. Rev.* **2022**, *55*, 2529–2573. [CrossRef]
44. Cao, B.; Sun, Z.; Zhang, J.; Gu, Y. Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3832–3840. [CrossRef]