symmetry

MDPI

Article

An Intelligent Genetic Scheme for Multi-Objective Collaboration Services Scheduling

Wei Guo [1,2], Lanju Kong [1,2,*], Xudong Lu [1,2] and Lizhen Cui [1,2]

[1] Research Center of Software and Data Engineering, School of Software, Shandong University, Jinan 250101, China
[2] Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR), Shandong University, Jinan 250101, China
* Correspondence: klj@sdu.edu.cn

Abstract: The optimization of collaborative service scheduling is the main bottleneck restricting the efficiency and cost of collaborative service execution. It is helpful to reduce the cost and improve the efficiency to deal with the scheduling problem correctly and effectively. The traditional genetic algorithm can solve the multi-objective problem more comprehensively than the optimization algorithm, such as stochastic greedy algorithm. But in the actual situation, the traditional algorithm is still one-sided. The intelligent genetic scheme (IGS) proposed in this paper enhances the expansibility and diversity of the algorithm on the basis of traditional genetic algorithm. In the process of initial population selection, the initial population generation strategy is changed, a part of the population is randomly generated and the selection process is iteratively optimized, which is a selection method based on population asymmetric exchange to realize selection. Mutation factors enhance the diversity of the population in the adaptive selection based on individual innate quality. The proposed IGS can not only maintain individual diversity, increase the probability of excellent individuals, accelerate the convergence rate, but also will not lead to the ultimate result of the local optimal solution. It has certain advantages in solving the optimization problem, and provides a new idea and method for solving the collaborative service optimization scheduling problem, which can save manpower and significantly reduce costs on the premise of ensuring the quality. The experimental results show that Intelligent Genetic algorithm (IGS) not only has better scalability and diversity, but also can increase the probability of excellent individuals and accelerate the convergence speed.

Keywords: genetic algorithm; multi-objective optimization; collaboration services scheduling; self-adaptation; symmetry and asymmetry; collaborative computing

Citation: Guo, W.; Kong, L.; Lu, X.; Cui, L. An Intelligent Genetic Scheme for Multi-Objective Collaboration Services Scheduling. *Symmetry* **2022**, *14*, 2037. https://doi.org/10.3390/sym14102037

Academic Editor: José Carlos R. Alcantud

Received: 28 August 2022
Accepted: 19 September 2022
Published: 29 September 2022

1. Introduction

With the rapid development of Web service technology and the popularity of collaborative services within and between enterprises, traditional intelligent applications transfer big data to the cloud for computing, which causes pressure on network bandwidth, large transmission delay, and reduced service quality. Considering time, cost and service, a cooperative optimization model is established to achieve a reasonable matching of supply and demand production tasks, which requires efficient cooperative service scheduling scheme. An efficient scheme in collaboration service execution needs to be obtained according to the current available resources, so that the execution efficiency and cost of collaboration services can satisfy the needs of users [1–3]. And with the development of microservices and Docker technology, service providers can flexibly and dynamically cache microservices at the edge, so as to respond efficiently with limited resources. Therefore, in the process of collaboration service execution, it is necessary to schedule the collaborative service according to the user's quality of service (QoS) requirements to obtain the optimal execution scheme [4–7].

Through the analysis of the above application deployment problem, it is not difficult to conclude that the problem belongs to the category of the Nondeterministic Polynomially

(NP) problem [8,9], because it is similar to knapsack problem, that is, no algorithm can solve the problem in polynomial time. NP problems are usually solved by dynamic programming algorithm [10], simulated annealing algorithm [11], neural network algorithm [12], particle swarm algorithm [13], genetic algorithm [14–16]. The genetic algorithm is an algorithm that draws on global evolution to obtain the optimal solution to the global search. It is very effective in solving random and nonlinear problems, and widely used in various fields, especially for NP problems. However, the genetic algorithm does not have many mathematical requirements for the optimization problem to be solved. By using the concept of evolution, we can get a better solution. Because of its evolutionary characteristics, the solution process does not need to understand the inherent properties of the problem. Since it can deal with any form of objective function and constraint, it is more suitable for solving this kind of optimal combination problem [17–19].

Therefore, genetic algorithm is a suitable algorithm for solving this kind of optimal combination problem. However, one of the main disadvantages of this method is that the implementation of traditional genetic algorithm is very time-consuming. The main reason is that the initial population generation time is too long and the convergence speed is too slow. Moreover, the treatment of low-complexity individuals in traditional genetic algorithms leads to the loss of superior genes in the population. This in turn leads to the reduction of the optimization degree of the final solution and the decrease of the convergence speed. In addition, the fixed mutation rate in traditional genetic algorithm keeps the evolution rate unchanged, that is, it evolves at a fixed rate regardless of the current population status, and it has the same probability variation regardless of the quality of the gene. As a result, the superior and inferior genes show a symmetrical pattern; in other words, the mutation probabalities of the superior and inferior genes are the same. In addition, the more superior genes exist in the population, the more difficult it is to optimize [20–22].

In view of the above limitations, this paper proposes an intelligent genetic scheme (IGS)-based application deployment algorithm, which improves the genetic algorithm on the basis of the traditional genetic algorithm as follows:

- The initial population generation strategy has changed. The proposed IGS reduces the generation time of the initial population by generating all the population randomly to generate a part of the population randomly, and then generating the remaining population in the way of approximately randomly generated chromosomes.
- In order to ensure the diversity of the population, the selection method was changed from one based on symmetric exchange to one based on asymmetric exchange.
- Some individuals with lower physical fitness are retained. Individuals may have low adaptability due to one or two inferior genes. However, these genes may have better gene fragments.
- Adaptive mutation rate is used. In this way, the mutation rate of newly generated individuals is dynamically determined according to the fitness of their parents.

The IGS can maintain the diversity among individuals, increase the probability of excellent individuals, speed up the convergence rate, and does not cause the final results to tend towards the local optimum. This approach has certain advantages in solving such optimization problems. Optimizing the execution of business processes within the service architecture can significantly [23] reduce costs while maintaining quality of service.

The remainder of this paper is organized as follows. Section 2 of this paper briefly introduces the related work on this subject. Section 3 discusses the proposed genetic scheme in detail, including the coding mode, cost model, cross-selection method, individual selection method, mutation strategy and fitness function. Section 4 introduces the simulation experiment and discusses the simulation results. Finally, Section 5 presents the conclusions and points out some directions for future work.

2. Related Work

To obtain the optimal execution plan for the QoS-aware deployment of IoT applications [24–27], the ant colony algorithm, particle swarm optimization algorithm and genetic algorithm can be used to solve multi-objective optimization problems of this kind.

Solnon et al. [28] used the ant colony algorithm to solve multi-objective optimization problems. On this basis, Stutzle et al. [29] proposed the MAX-MIN ant system algorithm to limit the amount of pheromone on the path. These authors set MAX and MIN to update the pheromone on the path, only updated the path of the optimal solution at any time, and set the pheromone reduction of other paths. Therefore, the pheromone of the quality path increases rapidly and the inferior path decreases slowly, which overcomes the problem of easy stagnation. Xia et al. [30] proposed a multi-pheromone, dynamically updating ant colony optimizing algorithm (MPDACO). By altering the dynamic transition probability and pheromone change rules in the ant colony algorithm, MPDACO implements the dynamic selection optimal scheme under multiple constraints. However, these algorithms incorporate only one pheromone, which means that they cannot solve the problem of finding the optimal solution under multiple constraints.

Coello et al. [31] proposed a multi-objective particle swarm optimization (MOPSO) algorithm, which introduced Pareto dominance into particle swarm optimization (PSO). This algorithm uses an external repository of particles, which is then used by other particles to guide their own flight. Furthermore, these authors also incorporate a special mutation operator that enriches the algorithm's exploratory capabilities. Through the verification of the standard method used in the field of evolutionary multi-objective optimization, it is proved that the method has strong competitiveness and can be regarded as a feasible method to solve multi-objective optimization problems. What should be noted here is that due to the inherent correlation between variables, the decomposition of fixed variables leads to the loss of a large part of the internal correlation information. The CCMOPSO algorithm proposes a random decomposition method of variables. The purpose of this method is to mine the internal association information among variables, and put the associated variables into the same group as much as possible, so as to avoid the problem of inline data loss caused by the decomposition of fixed variables. However, large-scale variables can give rise to procedural disasters and cannot solve our current multivariate optimization problems. At the same time, when the objective function is multi-modal, the traditional particle swarm optimization algorithm is still prone to fall into the local optimization.

Yang et al. [32] proposed a compressed-encoding PSO with fuzzy learning (CEPSO-FL) algorithm to solve the problem of poor search performance when PSO was processing large-scale features. Experiments proved that this method could solve the large-scale feature selection problem well. The solutions obtained by CEPSO-FL contain small feature subsets and have an excellent performance in classification problems. However, it is difficult to reduce the computational cost of evaluation effectively.

Guo et al. [33] proposed a self-adapted task scheduling strategy for wireless sensor networks (WSNs). First, a multi-agent-based architecture for WSN is proposed and a mathematical model of dynamic alliance is constructed for the task allocation problem. Next, an effective discrete particle swarm optimization algorithm for the dynamic alliance is proposed, featuring a well-designed particle position code and fitness function. A mutation operator that can effectively improve the algorithm's ability to conduct global search and improve population diversity is also introduced in this algorithm.

Wang et al. [34] proposed a collaboration differential evolution algorithm with multiple populations for multi-objective optimization [14]. This algorithm has M single-objective optimization subpopulations and an archive population to solve the M-objective optimization problems. An adaptive differential evolution algorithm is applied to each subpopulation in order to optimize the corresponding objectives of the multi-objective optimization problem. The proposed algorithm handles multiple targets using multiple subpopulations; each of these subpopulations only processes one target, and the subpopulations then cooperate to approximate the overall optimal solution. Antonio et al. [35] also proposed

applying scalable decision variables to multi-objective optimization problems, which is both an effective and efficient means of solving large-scale multi-objective optimization problems. However, the issue here is that the services must be executed in sequence, meaning that, these methods are not able to solve such problems well.

In brief, the adaptation function is optimized on the basis of the basic genetic algorithm. Not only is the completion time of the total task included in the adaptation function, but the average time taken to complete the task is also included in the adaptation function; this helps to improve the convergence speed of the algorithm and decrease the time taken to find the optimal solution. The roulette strategy is used as a selection operator when screening individuals, while the probability is selected according to the individual's fitness when performing crossover and mutation. Although the algorithm has some advantages when compared with the basic genetic algorithm, the roulette strategy for screening individuals will cause the convergence speed of the algorithm to slow down, meaning that the probability of getting the local optimal solution will increase accordingly.

Kuo et al. [36] proposed a novel selection method based on the issue of poor individual diversity in the evolution of traditional genetic algorithms, referred to as the disruptive genetic algorithm (DGA). This method uses a non-monotonic fitness function that differs completely from the traditional monotonic fitness function, and is beneficial to both superior and inferior individuals. DGA effectively alleviates this problem by first demonstrating that DGA can be used to solve non-stationary search problems, i.e. those in which the goal is to track time-varying optima. DGA tracks the best values immediately after environmental changes occur. DGA immediately tracks the optimum after the change of environment, and it has an advantage in solving spike functions; however, it is not applicable to the multi-objective optimization problem studied in this paper.

Li et al. [37] optimized the traditional genetic algorithm by defining a local search operator to detect community structure within a network. However, while this method has obvious advantages for solving network problems, it is not suitable for solving optimization problems.

Pereira et al. [38] studied and explored the adaptive mutation rate, proved its validity, and verified it via experiments. However, the mutation rate only changes with the number of iterations and does not consider the characteristics of the individual; thus, it may tend to be a local optimal solution under the multiple constraints described in this paper.

Zhang et al. [39] proposed a synergistic adaptive genetic algorithm (SAGA). It is proved that SAGA has better global optimal solution search ability. Although the proposed algorithm improves the traditional genetic algorithm and maintains the diversity of the population to a certain extent by adjusting the range of fitness values, the probability of obtaining excellent individuals needs to be verified.

To sum up, the existing methods mentioned above have certain limitations in solving the business process optimization problem described in this paper. When these methods are used to solve multi-objective sequential optimization problems, they will fall into local optimal solutions and fail to solve multivariable [40]. Therefore, the above multi-objective optimization scheduling method is not suitable for our purpose.

3. Our Proposed Intelligent Genetic Scheme

3.1. Problem Formulation

Definition 1. *Application S. The application S consists of a set of web services W and a set of partial order relations R on the set of web services, where:* $W = \{s_i | i = 1, 2, ..., n\}, R = \{< s_i, s_j > | 1 <= i <= n, 1 <= j <= n\}$.

Definition 2. *Available resource set R. The available resource set R consists of available resources N and constraint set C for each of the available resources, where* $N = \{n_i | i = 1, 2, ..., m\}, C = \{(n_i, s_j) | 1 <= i <= m, 1 <= j <= n\} \cup \{max(n_i) <= k\}$, *and* $max(n_i)$ *represents the maximum number of services that resource i can process.*

Definition 3. *User QoS requirements. User QoS requirements $Q = \{T, C, A\}$ for application execution, where T is the time required for application execution, C is the cost required for application execution, and A is the usability of application execution. Here, usability is used to measure whether the solution satisfies the constraints of the problem and can thus be used for practical implementation. In the following sections, the usability of application S will be measured by the quality of service.*

Definition 4. *QoS results for an application applying $Q_s = \{T_s, C_s, A_s\}$ and user QoS requirements $Q = \{T, C, A\}$. If $T >= T_s$, $C >= C_s$, and $A <= A_s$, then this is recorded as $Q <= Q_s$, which is referred to as a case where Q_s satisfies Q.*

Definition 5. *The usability $D = \{(s_i, n_j)|i = 1, 2, ..., n, j = 1, 2, ..., m\}$; i.e., for any web service s_i in an application S, there is always a resource n_j responsible for the execution of the web service and that satisfies the constraints of the resource n_j. Thus, D is referred as applicable for the application S on the available resource set R.*

3.1.1. Representation of Application Deployment

Assumption: the number of web services in an application is n, and there is a partial order relationship between services; the number of available resources is m; the number of web services that can be executed by each available resource is different from the maximum number of executable web services; the time, cost and availability of each available resource executing the same web service are different; the feasible solution to the application deployment problem is a one-dimensional array of n elements, called application array X. Then, i can be defined as the number of resources required to execute the j-th web service so that any element X_j of the application array can be represented as:

$$X_j = i. \tag{1}$$

For each application array, X has the following restrictions:

Condition 1. *The maximum number of web services that can be executed by the available resources is represented as a set M, where M_i represents the maximum number of services able to be processed by the available resources i. Moreover, the set S is used to represent the number of services processed by each available resource in the current application array X, where S_i represents the total number of services able to be carried out using the available resources i. Furthermore, the number of web services executed by i must satisfy the following condition:*

$$S_i < M_i. \tag{2}$$

Condition 2. *Since the application array X is a linear structure, there is only one element per location, which directly satisfies the constraint that each service is executed by only one available resource. Therefore, the constraint of this condition on the results will not be considered below.*

Condition 3. *The quality of the web service executed by the available resources is expressed as a matrix T, where T_{ij} represents the QoS for service j when executed by available resource i. Users' minimum requirements for service quality are expressed as set Y, where Y_j represents users' minimum service quality for service j. To define service j as executed by i in the application array X, the following conditions need to be met to successfully execute the service from available resources:*

$$Y_j <= T_{ij}. \tag{3}$$

3.1.2. Chromosome Coding

In this paper, an improved genetic scheme is used to optimize the application-deploying problem.

- Each available resource needs to be chromosomally encoded.

If N is defined as the number of available resources, the number of gene positions c for each available resource on the chromosome is determined as follows:

$$c = \lceil log_2 N \rceil. \tag{4}$$

In the traditional expression, the above chromosomes require N bits to represent each available resource. By contrast, using Eq. (4) can compress the length of the chromosome based on the traditional expression, reduce the memory consumption required by the experiment, and improve the speed of the evolution.

- Chromosomal encoding for each application array X.
  After encoding the available resources, the chromosomes are encoded according to the assignment of each element in X. If M is defined as the number of web services in the application, the length L of the chromosome is as follows:

$$L = M \times c = M \times \lceil log_2 N \rceil. \tag{5}$$

For example, if the number of available resources is $N = 6$, then the number of gene bits per available resource on the chromosome is determined according to Eq. (4), $C = 3$, and the coding for the available resources is shown in Table 1.

Table 1. An example of available resources coding.

Available resources	1	2	3	4	5	6
Coding	000	001	010	011	100	101

If the number of web services in the application is defined as $M = 9$, then the overall chromosome length is 27, and the chromosome coding is as follows: 000101000011010001000101100.

In the process of application deployment, users need to comprehensively evaluate the composite scheme based on the three aspects of cost, time and service quality. Therefore, this section includes three parts of the application deployment model: the cost model, time model and service quality model.

- Cost model
  The total cost of application execution is the sum of the cost of executing each web service. In turn, the cost of executing each web service on all available resources can be represented as a matrix $Quote$, where $Quote_{ij}$ represents the cost of executing a web service j on the available resource i. If the cost of executing web service is defined as j on the available resource i in the synergistic service array X, the final total cost of application execution $Cost$ can be obtained as follows:

$$Cost = \sum_{j=1}^{M} Quote_{ij}, X_j = i. \tag{6}$$

When an application array X is generated and the above operation is performed, the total cost of executing the application can be obtained. According to the $Cost$ obtained from each synergy service array X, the minimum cost and the optimal application-deploying scheme corresponding to the minimum cost can be determined.

- Time model
  The start execution time T_s of the first web service in the application is defined as 0. According to the application array X and the partial order relationship between web services, along with the length of time required by the available resources to execute different web services T_{ij}, the completion time of the last web service T_f can be obtained. Matrix T stores the time cost of each web service executed by each available resource; this is an $N \times M$ matrix, where T_{ij} is the time cost of web service j being

executed by available resource i; moreover, the M column of the matrix corresponds to M web services, and these M web services have a temporal topology sequence. Therefore, the total time (defined as *Time*) required to execute the entire application can be obtained as follows:

$$Time = T_f - T_s. \tag{7}$$

- Service usability model
  Given the widespread differences in computing power, memory, network bandwidth, and many other aspects of available resources, the QoS for each available resource is different when the same web service is being performed. The QoS of the available resources when performing web services is expressed as matrix A, where A_{ij} represents the QoS of the available resources i when executing web services j. The QoS corresponding to the synergetic service array X can thus be expressed as follows:

$$Usability = \sum_{j=1}^{M} A_{ij}, X_j = i. \tag{8}$$

- Target of model
  The aim of deploying these applications is a) to reduce the cost as much as possible and b) to reduce the execution time of the application with the goal of ensuring service usability. Therefore, the objective function can be established as follows:

$$\begin{aligned} Fitness = min(a * norm\_Cost + b * norm\_Time + \\ (1 - a - b) * norm\_Usability). \end{aligned} \tag{9}$$

$$norm\_X = \frac{X - min(X)}{max(X) - min(X)}. \tag{10}$$

  The *norm_Cost*, *norm_Time* and *norm_Usability* in Eq. (9) is the normalized value of *Cost*, *Time* and *Usability* which are calculated using Eq. (10). Eq. (9) needs to satisfy Eqs. (2) and (3). Thus, the term constraint described below refers to Eqs. (2) and (3). The parameters a and b of *Fitness* in Eq. (9) are formulated according to the emphasis placed by the user on each factor.

3.2. Intelligent Genetic Scheme

The traditional genetic algorithm (GA) generates the initial population and converges slowly to solve the application-deploying problem. As a result, the traditional genetic algorithm becomes time-consuming and inefficient in the late evolutionary stage. In addition, in the application of traditional genetic algorithm, it has always been a difficult problem which selection method can be used to maintain both the superior individuals and the diversity of the population, so that the good genes can be preserved.

Therefore, a new intelligent genetic scheme is proposed to solve the above-mentioned problems. IGS is not only applicable to collaborative service scheduling scenarios, but also to other application scenarios.

The IGS consists of four parts: (1) The initial ratio of the initial point is specified in order to generate the initial population, which is changed from random to the generated proportion of the specified individuals; the remainder are close to the randomly generated individuals. (2) The proposed scheme retains individuals with low fitness and reuses their superior genes. (3) Based on the selection of cross-population, the populations are divided into multiple sub-populations, which are selected and crossed in different ways. (4) Individual mutation is carried out with an adaptive mutation rate, and the mutation rate of newly produced individuals is determined dynamically according to the fitness of their parents.

3.2.1. Generation of Initial Populations

The number of initial populations is set to *population_Size*. Firstly, random initial points are generated according to the percentage M of random initial points (random initial points refer to the individuals randomly generated under initial conditions that satisfy the constraints), that is, the number of random initial points is $population\_Size * M$. For each random initial point, *entity_size* individuals are randomly generated. From these individuals, the individuals satisfying the constraints remain in the initial population, while the others are close to the generated individuals until the constraints are satisfied and remain in the initial population. This process continues until the number of individuals generated is *population_Size*. The way to bring an individual that does not satisfy all constraints closer to an individual that satisfies all constraints is to take the midpoint of both as the new value of the individual; subsequently, if all constraints are still not satisfied, the midpoint of both is taken as the new value again, and this process is repeated until all constraints are met. Although initializing the population by randomly generating some initial points and ensuring that other nodes approach them will reduce the diversity of the initial population to a certain extent, the influence of this initialization mode on the diversity of the population can be modified by adjusting proportion of random initialization points, which will not reduce the diversity of the initial population, and greatly improve the initialization time of the population.

A midpoint operation is adopted in this scheme. When taking the midpoint of two individuals to form a new individual, it is necessary to count the number of services provided by each service provider among the two individuals, then take the midpoint operation for the number of services provided by each service provider. For example, the service provider numbered 1 in the initial individual A provides a total of 5 services, while the service provider numbered 1 in the individual B approaching the initial individual A provides a total of 3 services; the service provider numbered 1 in the newly formed individual B provides a total of 4 services, and the numbers of these 4 services are randomly generated. Finally, for the unassigned tasks, the service provider is generated by randomly generating a service provider number.

Algorithm 1 greatly reduces the complexity of initial population generation by randomly generating initial interior points in a certain proportion. By generating a specified number of random points for each interior point and approaching the initial interior point, more individuals can be obtained, thus ensuring the diversity of the initial population. Algorithm 2 can ensure that the diversity of the initial population is more ideal based on the complexity of $O(n)$, where n refers to the number of individuals in the population.

Algorithm 1 IGS algorithm

Input: initial population number $population$
Output: individual fitness $fitness$

Calculating individual fitness $fitness$;
while $fitness$ does not meet the convergence condition **do**
    if $fitness$ Satisfies all constraints **then**
        Keep excellent individuals;
    else Keep a small number of individuals with low fitness;
    end if
    Cross progeny generated based on the inter population;
    Variability using adaptive mutation rate;
end while
return $fitness$;

Algorithm 2 Initialize the population algorithm

Input: initial population number *population_Size*, percentage of initial inner point M, number of randomly generated individuals *entity_Size*
Output: initial population *population*

```
i = 0, j = 0;
for i < population_Size * M do
    Generate an individual initial_Entity that satisfies all constraints;
    Put initial_Entity into the population;
    i++;
    for j < entity_Size do
        Randomly generate a new individual Entity;
        while entity does not satisfy all constraints do
            Perform the midpoint operation of initial_Entity and Entity to become the new Entity;
        end while
        Entity is retained in the population;
    end for
end for
return population;
```

3.2.2. Selection Based on Crossing between Populations

The selection method based on inter-population crossing involves dividing the original population into sub-populations and making different types of selection and crossing operations according to the population, shown in Algorithm 3. The generated initial population is divided into two groups on average: population A and population B. Population A is then divided into population $A1$ and population $A2$ on average, while population B is divided into population $B1$ and population $B2$ on average. Population $A1$ and $B1$ then select individuals $a1$ and $b1$ by roulette strategy, respectively, and remove the selected individuals from the population. Individuals $a1$ and $b1$ cross to generate an offspring , which is stored in the offspring population *new_population*. According to the above method, individuals are selected from populations $A1$ and $B1$ to be crossed in order to generate offspring until all individuals in the population are selected. Populations $A2$ and $B2$ respectively select individuals $a2$ and $b2$ using the tournament method and remove the selected individuals from their populations. Individuals $a2$ and $b2$ then cross to produce an offspring, which is stored in the offspring population *new_population*.

3.2.3. Adaptive Mutation Rate

The traditional genetic algorithm has a fixed mutation rate in the generation of progeny, which limits the diversity of its population. In the IGS proposed by us, the adaptive mutation rate method dynamically determines the mutation rate of individuals and further improves the diversity and evolution rate of populations. In order to determine the mutation rate of an individual, the individual's father and mother (i.e. the two individuals who cross-produce the individual) are searched first, and then the mutation rate of the individual is determined according to the fitness of the parents.

For example, for any individual *Entity* in a population, if the initial mutation rate is defined as S, while the father and mother are *F_Entity* and *M_Entity*, then the mutation rate of *Entity* obtained according to the adaptive mutation rate method.

The fixed mutation rate adopted in the traditional genetic algorithm has another defect. If the value of the mutation rate is set too small, the evolution rate will be extremely low, resulting in a lengthy process. Moreover, if the value of the mutation rate is set too high, an individual with high fitness can easily proceed in a bad direction; this will make the direction of the evolution difficult to control, and the resulting length of the result cannot be determined. The mutation rate is dynamically determined by means of the adaptive mutation rate, and the mutation rate can also be determined according to the

actual situation of different individuals. Individuals with high fitness can retain good genes to the greatest extent and evolve faster. Low-adaptation individuals can increase the probability of gene mutation into good gene, and the evolution can be accelerated by increasing the mutation rate.

Algorithm 3 Inter-population cross-method individual selection algorithm

Inter-population cross-method individual selection algorithm
Input: initial *population* and its size *population_Size*
Output: child generation *new_population*

The population is divided into populations $A1\_population$, $A2\_population$, $B1\_population$, $B2\_population$;
for $z = 0; z < A1\_population\_Size; z++$ **do**
    Remove an individual $a1$ from the $A1\_population$ using the roulette strategy;
    Remove an individual $b1$ from the $B1\_population$ using the roulette strategy;
    $a1$ and $b1$ cross-generate the individual $a1'$;
    while $a1'$ does not satisfy all constraints **do**
        $a1$ and $b1$ cross-generate the individual $a1'$;
    end while
    Save $a1'$ to the population *new_population*;
end for
for $t = 0; t < A2\_population\_Size; t++$ **do**
    Remove an individual $a2$ from the $A2\_population$ using the tournament method;
    Remove an individual $b2$ from the $B2\_population$ using the tournament method;
    $a2$ and $b2$ cross-generate the individual $a2'$;
    while $a2'$ does not satisfy all constraints **do**
        $a2$ and $b2$ cross-generate the individual $a2'$;
    end while
    Save $a2'$ to the population *new_population*;
end for
return *new_population*;

While the traditional genetic algorithm eliminates individuals with low fitness, some of them may have low overall fitness because of one or two bad genes, although they may also have good genes. In order to maintain the population diversity and continue the transmission of good genes, we retained them and determined the mutation rate by determining the adaptive mutation rate. According to the adaptive mutation rate method, individuals with low fitness have higher mutation rates, which increases the probability of improving the fitness of individual offspring, shown in Algorithm 4.

Algorithm 4 Preserving low fitness individual algorithms

Input: initial population and its size *population_Size*, retained low fitness individual ratio *Entity_rate*
Output: selected population *New_population*

Initialize a low fitness individual set *Entities*;
for $i = 0; i < population\_Size * (1 - Entity\_rate); i++$ **do**
    Use uniform sorting selection to select *Entity*;
    Put *Entity* into *New_population*;
end for
for $j = 0; j < population\_Size * Entity\_rate; j++$ **do**
    Randomly select individual *L_Entity* from *Entities* and put it into *New_population*;
end for
return *New_population*;

By retaining some of the low-complexity individuals, the excellent genes contained in the low-compatibility individuals can be retained to some extent, thus accelerating the evolution speed.

Through the above optimization, the optimization of the initial population running time will be analyzed, as well as the influence of evolution algebra and fitness in the following chapters.

4. Simulation Experiments and Results Analysis

4.1. Experimental Setup Instructions

This section will compare the IGS algorithm proposed in this paper with traditional genetic algorithms. The experimental environment is Intel(R) Core(TM) i7-12700 4.90 GHz, RAM 16 GB, hard disk 500 GB, 1000 MB network bandwidth. Each comparative experiment was run 20 times and the results were. The time marked in the figure is measured in seconds. Since the unit of fitness cannot be expressed by an actual cost unit, the unit of cost indicated in the experiment is expressed in analog units. Moreover, the value of fitness is negatively correlated, that is, the smaller the fitness data, the better the algorithm.

- GA: Genetic algorithm, implemented using the classical genetic algorithm; genetic operator adopts crossover and mutation evolution [14].
- IGS: The improved intelligent genetic scheme proposed in this paper.
- GA+1: Based on the genetic algorithm GA, the algorithm is optimized for the initial population in Section 3.2.1 [20].
- GA+2: Based on the genetic algorithm GA, the algorithm is optimized for the crossover between populations in Section 3.2.2 [21].
- GA+3: Based on the genetic algorithm GA, the algorithm optimizes the adaptive mutation rate in Section 3.2.3 [22].
- CGA: The genetic algorithm that uses the adaptive mutation rate in [24].

In the optimization of the initial population, the percentage of initial interior points is set to 0.2, while the number of individuals in the population is 100. The maximum number of services performed by each available resource within a resource constraint is an integer between 5 and 7; moreover, the quality of service is an integer between 1 and 7, and the available resources of each task is an integer between 25 and 60. For individuals with low fitness, the retention rate is 0.1.

This is an example of a quote.

4.2. Experimental Parameter Determination

In this section, three experimental parameters are set for mutation rate, crossover rate and number of variant genes. In the IGS, different values are set for experimental comparison. First, a crossover rate of 0.8, a variant gene number of 1, and mutation rates of 0.03, 0.05, 0.07 and 0.09 for the comparison experiments are chosen. The results are shown in Figure 1.

From Figure 1 , it can be clearly seen that the *Fitness* value reduces as the evolution time increases. At the same time, it can be seen that the *Fitness* value is optimal when the mutation rate is 0.07. Subsequent experiments in this paper were carried out in an environment with a mutation rate of 0.07.

In the next step, a mutation rate of 0.07 is chosen. The number of variant genes was 1, and the crossover rates were 0.6, 0.7, 0.8 and 0.9 for the comparison experiments. The results are shown in Figure 2.

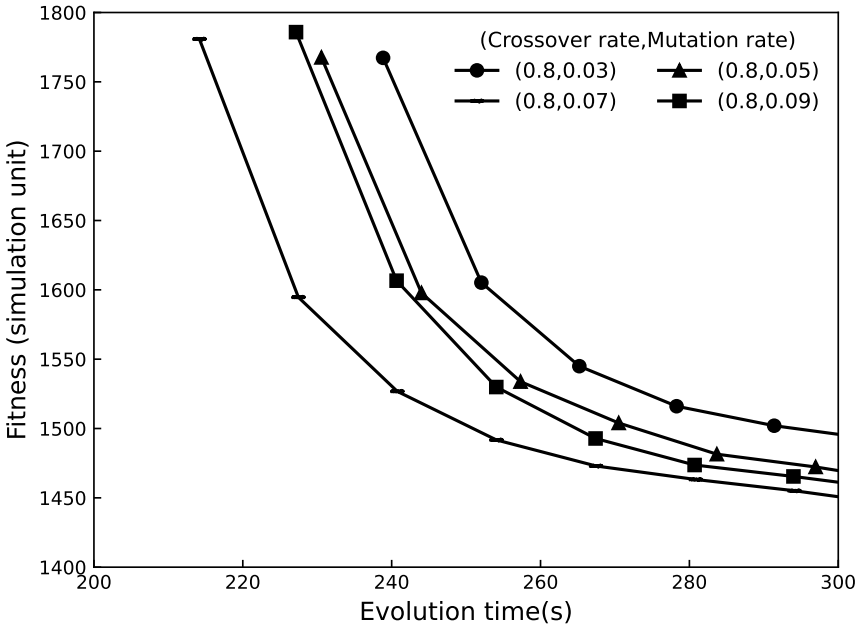

Figure 1. The effect of different mutation rates on *Fitness* under IGS.

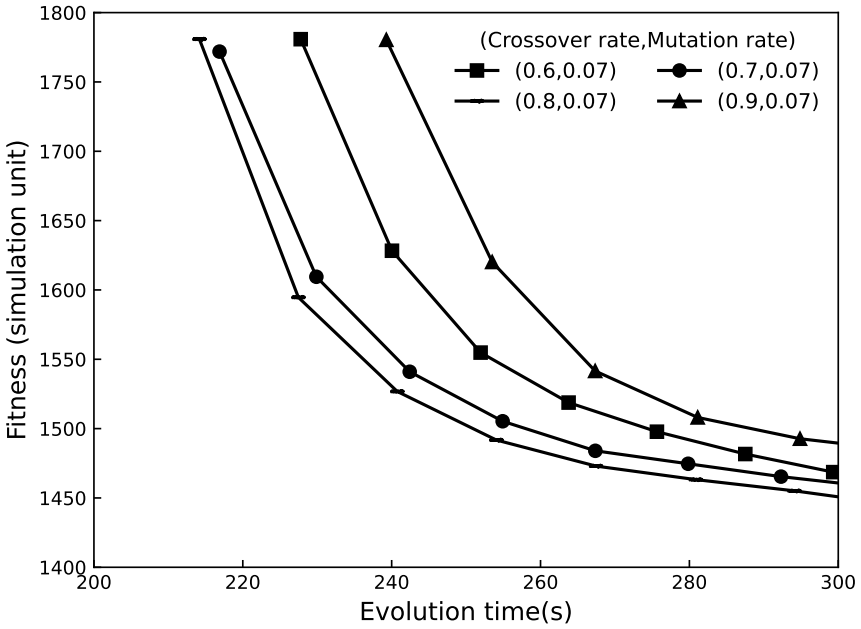

Figure 2. The effect of different crossover rates on *Fitness* under IGS.

As shown in Figure 2, the *Fitness* value decreases with the increase of evolution time. At the same time, it can be seen that the *Fitness* value is optimal when the crossover rate is 0.8. Subsequent experiments in this paper were carried out in an environment with a crossover rate of 0.8.

In the next step, mutation rate of 0.07, crossover rate of 0.8, and some mutation genes with values of 1, 2, and 3 for comparative experiment are selected. The results are shown in Figure 3.

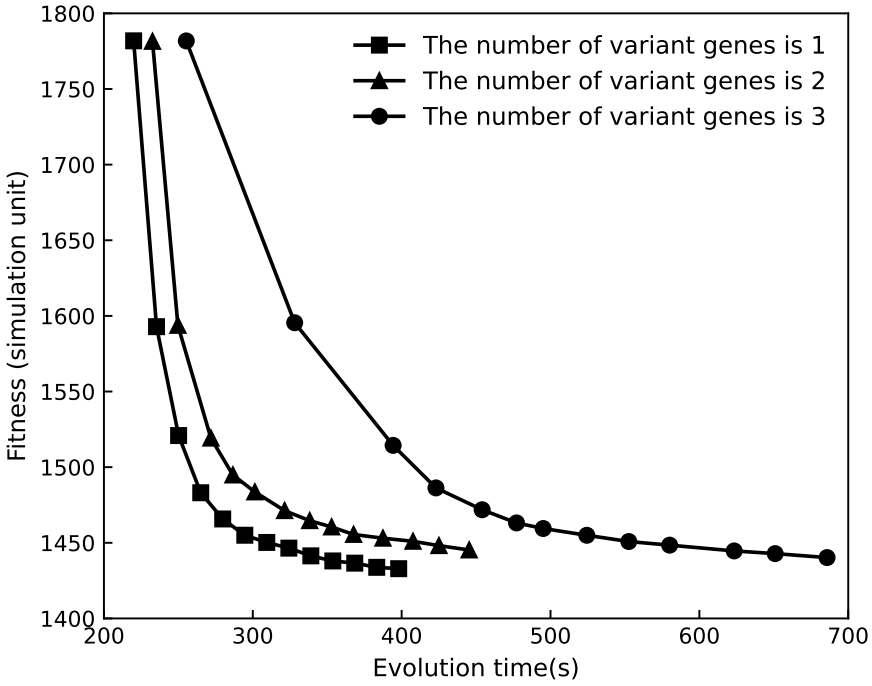

Figure 3. The effect of the number of different variant genes under IGS.

Figure 3 clearly shows that the *Fitness* value decreases with the increase of evolution time. It is also evident that when the number of mutant genes is 1, the *Fitness* value is optimal. Subsequent experiments in this paper were carried out in an environment where the crossover rate was set to 1.

Following comparison of the above experimental results, it can be concluded that IGS achieves the best performance when the mutation rate is 0.07, the crossover rate is 0.8, and the number of variant genes is 1. Therefore, for the subsequent experiments, the above values will be taken as the default values.

4.3. Performance Analysis: Comparison with Existing Main Schemes

This section uses five algorithms for comparative experiments: GA, IGS, GA+1, GA+2 and GA+3. These experiments compare the number of services that need to be coordinated as 15, 30, 45 and 60.

When comparing the evolution time lengths of different numbers of services, it is necessary to select the target *Fitness* value separately according to the number of services, and then obtain the evolution time of different algorithms according to the experiment to achieve the target *Fitness* value. The service times of this experiment were 15, 30, 45 and 60, and the specified fitness were 500, 650, 1150 and 1500. The evolution time of different algorithms under different service numbers is shown in Figure 4.

It can be clearly seen from Table 2 that as the number of applications continues to increase, the evolutionary duration of each algorithm also increases. Under the same target *Fitness* value, the IGS is significantly superior to other algorithms in terms of the evolution time for different numbers of coordinated services. Moreover, in the comparison between IGS and GA, IGS reduced the evolution time by 43%.

Table 2. Running time of different algorithms when the number of collaborative applications changes.

Fitness name \ Number of tasks(a)	60	45	30	15
IGS	**251.43**	**20.62**	**6.61**	**1.21**
GA	439.10	54.98	23.81	5.61
GA+1	273.98	31.43	10.48	1.68
GA+2	420.26	47.21	24.49	4.58
GA+3	430.08	42.20	23.74	4.15

Due to the different numbers of applications, the evolutionary duration of each generation is different. It needs to determine the evolution time value of the target, specify it according to the evolution time of each application, and then determine the fitness of different algorithms after reaching the target evolution time. When the number of coordinated services is 15, 30, 45 and 60, the corresponding target evolution times are 1, 3, 6 and 50, respectively. In addition, actual experiments are carried out according to these conditions. The *Fitness* values obtained are shown in Figure 5.

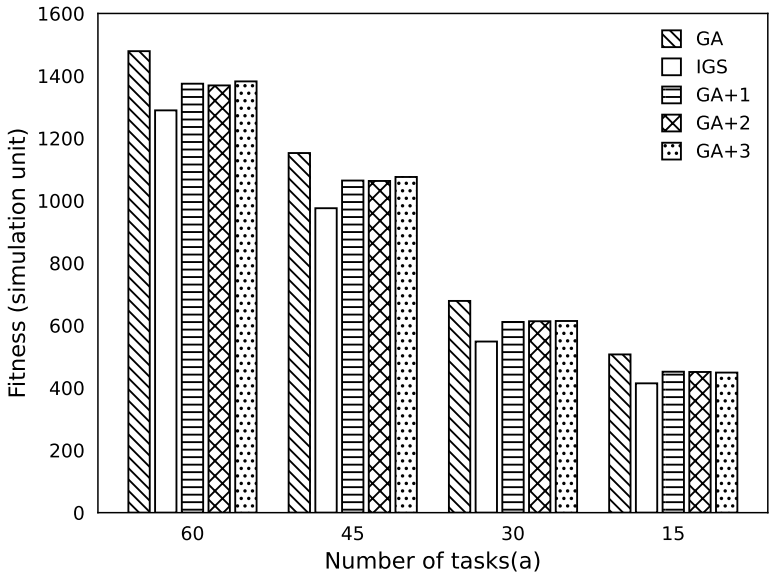

Figure 4. The *Fitness* value of different algorithms when the number of collaborative applications changes.

It can be clearly seen from Figure 4 that under the same evolutionary time, the IGS achieves better fitness than other algorithms for different numbers of applications, while the *Fitness* value is 10% higher than that of GA. The number of applications is set to 60. Experiments are carried out on the above five algorithms to analyze the influence of evolution time changes on the *Fitness* value under different algorithms, as shown in Figure 5.

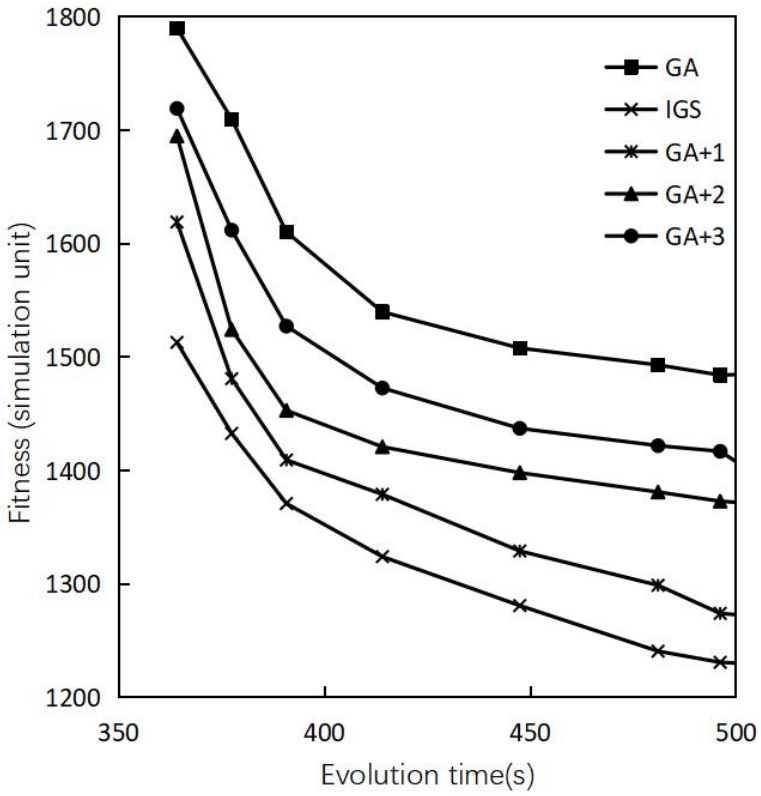

Figure 5. The *Fitness* value of IGS and GA,GA+1,GA+2,GA+3 as the evolutionary duration changes.

As can be seen from Figure 5, the *Fitness* values of IGS algorithm and GA [20,40], GA+1 [20,40], GA+2 [21,38], GA+3 [22,37] algorithm are optimized for the initial population, and the evolution starts in 200s and ends in 350s. Moreover, although the other three algorithms started to evolve after 350s, there are great differences in the evolution time. As is shown in the experiments, under the same evolution time, the *Fitness* value of the GA+1 is much smaller than the GA, GA+2 and GA+3. In summary, the IGS achieves best *Fitness* value.

It can be seen from the above comparison experiments that the IGS achieves obvious improvement effects relative to the other algorithms, and both the evolution time and *Fitness* value are significantly improved. Compared with the traditional genetic algorithm using the method based on population symmetric exchange, which has the disadvantage of time-consuming, IGS using the method based on population asymmetric exchange can reduce the cost under the premise of ensuring the quality, and has higher probability of excellent individuals and faster convergence speed, IGS outperforms other algorithms.

5. Discussion and Limitations

Although the genetic algorithm IGS based on population asymmetric switching has lower cost, faster convergence speed, and can find the optimal solution. At the same time, IGS has improved over traditional genetic algorithms, but the overall speed of IGS is still unavoidable and the running time of IGS increases when the population is large. Experimental results show that the IGS algorithm has a clear advantage in cooperative service scheduling, which also implies that the IGS algorithm can be used to solve the problem of optimizing the continuous deployment of applications. Donca et al. [41] proposed to optimize deployment through an automated pipeline generator based on agile practices, which shed light on the application direction of IGS algorithms in continuous deployment.

6. Conclusions and Future Work

The optimization problem of collaborative service scheduling is an important factor that restricts the efficiency and cost of collaborative service execution. It is helpful to reduce the cost and improve the efficiency of collaborative service execution to deal with the scheduling problem correctly and effectively. In order to solve this multi-objective optimization problem, an intelligent genetic scheme (IGS) is proposed based on traditional genetic algorithm, which improves the expansibility and diversity of the algorithm. On the basis of the traditional genetic algorithm, the initial population selection, adaptive selection of the mutation factor and individual population selection are improved. IGS can not only maintain the diversity of individuals, increase the probability of excellent individuals and accelerate the convergence speed, but also does not lead to the final result tending to a local optimal solution. It has some advantages in solving such optimization problems. When solving the problem of collaborative service scheduling, the cost can be reduced and the execution efficiency of collaborative services can be improved on the premise of ensuring service quality.

In future work, we need to further consider the case that the running time of the algorithm increases with the increase of the population, and make further optimization of the algorithm speed. At the same time, further research is made on how to apply IGS algorithm in continuous program deployment.

Author Contributions: This is a joint work and the authors were in charge of their expertise and capability: W.G.: data curation and writing original draft; L.K.: formal analysis and editing; X.L.: investigation and validation; L.C.: resources and supervision. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Nature Science Foundation of China No.91846205, 61772316, Science and Technology Development Plan Project of Shandong Province No.2018YFJH0506, 2018CXGC0706, Key Research & development Project of Shandong Province No.2019GGX101009, Shandong-Chongqing Science and Technology Cooperation Project No. cstc2020jscx-lyjsAX0010.

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

References

1. Zeng, Z.; Miao, W.; Li, S.; Liao, X.; Zhang, M.; Zhang, R.; Teng, C. Adaptive Task Scheduling in Cloud-Edge System for Edge Intelligence Application. In Proceedings of the 2021 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), New York City, NY, USA, 30 September–3 October 2021; pp. 1682–1689. https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00226.
2. Wan, L.; Wei, L.; Xiong, N.; Yuan, J.; Xiong, J. Pareto optimization for the two-agent scheduling problems with linear non-increasing deterioration based on internet of things. *Future Gener. Comput. Syst.* **2017**, *76*, 293–300. https://doi.org/10.1016/j.future.2016.09.004.
3. Li, F.; Li, X.; Yang, Y.; Xu, Y.; Zhang, Y. Collaborative Production Task Decomposition and Allocation among Multiple Manufacturing Enterprises in a Big Data Environment. *Symmetry* **2021**, *13*, 2268. https://doi.org/10.3390/sym13122268.
4. Cheng, Y.; Bi, L.; Tao, F.; Ji, P. Hypernetwork-based manufacturing service scheduling for distributed and collaborative manufacturing operations towards smart manufacturing. *J. Intell. Manuf.* **2020**, *31*, 1707–1720. https://doi.org/10.1007/s10845-018-1417-8.
5. Xu, Y.; Chen, L.; Lu, Z.; Du, X.; Wu, J.; Hung, P.C. An Adaptive Mechanism for Dynamically Collaborative Computing Power and Task Scheduling in Edge Environment. *IEEE Internet Things J.* **2021**. https://doi.org/10.1109/JIOT.2021.3119181.
6. Shi, L.; Fu, X.; Li, J. Mobility prediction-based service scheduling optimization algorithm in cloudlets. In Proceedings of the 3rd International Conference on Cloud Computing and Security, Nanjing, China, 16–18 June 2017; pp. 619–630. https://doi.org/10.1007/978-3-319-68542-7-53.
7. Yin, J.; Lo W.; Deng, S.; Li, Y.; Wu, Z.; Xiong, N. Colbar: A collaborative location-based regularization framework for QoS prediction. *Inf. Sci.* **2014**, *265*, 68–84. https://doi.org/10.1016/j.ins.2013.12.007.
8. Yuan, D.; Yang, Y.; Liu, X.; Chen, J. A data placement strategy in scientific cloud workflows. *Future Gener. Comput. Syst.* **2010**, *26*, 1200–1214. https://doi.org/10.1016/j.future.2010.02.004.
9. Cheng, H.; Su, Z.; Xiong, N.; Xiao, Y. Energy-efficient node scheduling algorithms for wireless sensor networks using Markov random field model. *Inf. Sci.* **2016**, *329*, 461–477. https://doi.org/10.1016/j.ins.2015.09.039.
10. Benayoun, R.; De Montgolfier, J.; Tergny, J.; Laritchev, O. Linear programming with multiple objective functions: Step method (stem). *Math. Program.* **1971**, *1*, 366–375. https://doi.org/10.1007/BF01584098.

11. Meng, F.C.; Chu, D.H.; Li, K.Q.; Zhou, X.Q. Solving SaaS components optimization placement problem with hybrid genetic and simulated annealing algorithm. *J. Softw.* **2016**, *27*, 916–932. https://doi.org/10.13328/j.cnki.jos.004965.
12. Zhang, Y.W.; Xiang, T.; Guo, X.; Jia, Z.H.; He, Q. Quality prediction for services based on SOM neural network. *J. Softw.* **2018**, *29*, 3388–3399. https://doi.org/10.13328/j.cnki.jos.005474.
13. Hu, W.; Yen, G.G.; Zhang, X. Multiobjective particle swarm optimization based on pareto entropy. *J. Softw.* **2014**, *25*, 1025–1050. https://doi.org/10.13328/j.cnki.jos.004496.
14. Keshanchi, B.; Souri, A.; Navimipour, N.J. An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing. *J. Syst. Softw.* **2017**, *124*, 1–21. https://doi.org/10.1016/j.jss.2016.07.006.
15. Jiang, Y.; Tong, G.; Yin, H.; Xiong, N. A pedestrian detection method based on genetic algorithm for optimize XGBoost training parameters. *IEEE Access* **2019**, *7*, 118310–118321. https://doi.org/10.1109/ACCESS.2019.2936454.
16. Yu, X.; Xiong, N.; Zhang, W. Research on mining rules from multi-criterion group decision making based on genetic algorithms. In Proceedings of the 13th IEEE International Conference on Computational Science and Engineering, Hong Kong, China, 11–13 December 2010; pp. 302–307. https://doi.org/10.1109/CSE.2010.45.
17. de Oliveira, L.L.; Freitas, A.A.; Tinós, R. Multi-objective genetic algorithms in the study of the genetic code's adaptability. *Inf. Sci.* **2018**, *425*, 48–61. https://doi.org/10.1016/j.ins.2017.10.022.
18. Bello-Orgaz, G.; Salcedo-Sanz, S.; Camacho, D. A multi-objective genetic algorithm for overlapping community detection based on edge encoding. *Inf. Sci.* **2018**, *462*, 290–314. https://doi.org/10.1016/j.ins.2018.06.015.
19. Toroslu, I.; Arslanoglu, Y. Genetic algorithm for the personnel assignment problem with multiple objectives. *Inf. Sci.* **2007**, *177*, 787–803. https://doi.org/10.1016/j.ins.2006.07.032.
20. Goren, H.; Tunali, S.; Jans, R. A review of applications of genetic algorithms in lot sizing. *J. Intell. Manuf.* **2008**, *21*, 575–590. https://doi.org/10.1007/s10845-008-0205-2.
21. Hyun, C.; Kim, Y.; Kim, Y. A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines computers. *Comput. Oper. Res.* **1998**, *25*, 675–690. https://doi.org/10.1016/S0305-0548(98)00026-4.
22. Qu, H.; Xing, K.; Alexander, T. An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots. *Neurocomputing* **2013**, *120*, 509–517. https://doi.org/10.1016/j.neucom.2013.04.020.
23. Górski, T.; Oźniak, A.P.W. Optimization of Business Process Execution in Services Architecture: A Systematic Literature Review. *IEEE Access* **2021**, *9*, 111833–111852. https://doi.org/10.1109/ACCESS.2021.3102668.
24. Fang, W.; Yin, X.; An Y.; Xiong, N.; Guo, Q.; Li, J. Optimal scheduling for data transmission between mobile devices and cloud. *Inf. Sci.* **2015**, *301*, 169–180. https://doi.org/10.1016/j.ins.2014.12.059.
25. Lin, B.; Guo, W.; Chen, G.; Xiong, N.; Li, R. Cost-driven scheduling for deadline-constrained workflow on multi-clouds. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshop, Hyderabad, India, 25–29 May 2015; pp. 1191–1198. https://doi.org/10.1109/IPDPSW.2015.56.
26. Wan, J.; Yang, L.T.; Li, Y.; Xu, X.; Xiong, N. An adaptive management mechanism for resource scheduling in multiple virtual machine system. In Proceedings of the 8th International Conference on Autonomic and Trusted Computing, Banff, AB, Canada, 2–4 September 2011; pp. 60–74. https://doi.org/10.1007/978-3-642-23496-5-5.
27. Xu, J.; Liu, A.; Xiong, N.; Wang, T.; Zuo, Z. Integrated collaborative filtering recommendation in social cyber-physical systems. *Int. J. Distrib. Sens. Netw.* **2017**, *13*, 1550147717749745. https://doi.org/10.1177/1550147717749745.
28. Solnon, C. Ants can solve constraint satisfaction problems. *IEEE Trans. Evol. Comput.* **2002**, *6*, 347–357. https://doi.org/10.1109/TEVC.2002.802449.
29. Stutzle, T.; Hoos, H. MAX-MIN ant system and local search for the traveling salesman problem. In Proceedings of the IEEE International Conference on Evolutionary Computation, (ICEC '97), Indianapolis, IN, USA, 13–16 April 1997; pp. 309–314. https://doi.org/10.1109/ICEC.1997.592327.
30. Xia, Y.M.; Cheng, B.; Chen, J.L.; Meng, X.W.; Liu, D. Optimizing services composition based on improved ant colony algorithm. *Chin. J. Comput.* **2012**, *35*, 270–281. https://doi.org/10.3724/SP.J.1016.2012.00270.
31. Coello, C.; Pulido, G.; Lechuga, M. Handling multiple objectives with particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 256–279. https://doi.org/10.1109/TEVC.2004.826067.
32. Yang, J.-Q.; Chen, C.-H.; Li, J.-Y.; Liu, D.; Li, T.; Zhan, Z.-H. Compressed-Encoding Particle Swarm Optimization with Fuzzy Learning for Large-Scale Feature Selection. *Symmetry* **2022**, *14*, 1142. https://doi.org/10.3390/sym14061142.
33. Guo, W.; Xiong, N.; Chao, H.C.; Hussain, S.; Chen, G. Design and analysis of self-adapted task scheduling strategies in wireless sensor networks. *Sensors* **2011**, *11*, 6533–6554. https://doi.org/10.3390/s110706533.
34. Wang, J.; Zhang, W.; Zhang, J. Cooperative differential evolution with multiple populations for multiobjective optimization. *IEEE Trans. Cybern.* **2015**, *46*, 2848–2861. https://doi.org/10.1109/TCYB.2015.2490669.
35. Antonio, L.; Coello, C. Use of cooperative coevolution for solving large scale multiobjective optimization problems. In Proceedings of the IEEE Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 2758–2765. https://doi.org/10.1109/CEC.2013.6557903.
36. Kuo, T.; Hwang, S. Using disruptive selection to maintain diversity in genetic algorithms. *Appl. Intell.* **1997**, *7*, 257–267. https://doi.org/10.1023/A:1008276600101.
37. Li, S.; Chen, Y.; Du, H.; Feldman, M.W. A genetic algorithm with local search strategy for improved detection of community structure. *Complexity* **2010**, *15*, 53–60. https://doi.org/10.1002/cplx.20300.

38. Pereira, A.; Andrade, B. On the genetic algorithm with adaptive mutation rate and selected statistical applications. *Comput. Stat.* **2014**, *30*, 131–150. https://doi.org/10.1007/s00180-014-0526-x.
39. Zhang, D.; Li, P.; Wulamu, A. An Improved Multi-Label Learning Method with ELM-RBF and a Synergistic Adaptive Genetic Algorithm. *Algorithms* **2022**, *15*, 185. https://doi.org/10.3390/a15060185.
40. Ide, J.; Köbis, E.; Kuroiwa, D.; Schöbel, A.; Tammer, C. The relationship between multi-objective robustness concepts and set-valued optimization. *Fixed Point Theory Appl.* 2014, 83, 1-20. https://doi.org/10.1186/1687-1812-2014-83.
41. Donca, I.-C.; Stan, O.P.; Misaros, M.; Gota, D.; Miclea, L. Method for Continuous Integration and Deployment Using a Pipeline Generator for Agile Software Projects. *Sensors* **2022**, *22*, 4637. https://doi.org/10.3390/s22124637.