Optimized swarm Intelligence methods and applications for Large-scale data

Fangqi Nie

Abstract

In the face of large-scale problems, traditional single-individual or centralized approaches to solving them are no longer sufficient, and researchers have found that groups (societies) exhibit far more complex intelligent behavior than individuals. Therefore, swarm intelligence methods are proposed to bring together swarm intelligence to collaboratively solve large-scale complex problems. First, three evolutionary frameworks for swarm intelligence based on probability distributions are proposed, in which optimization of hierarchical learning swarm intelligence evolution is proposed, and distributed collaborative swarm intelligence evolution optimization is proposed to reduce the time complexity, increase the search diversity, and continuously optimize. Then the application of the corresponding algorithms is proposed, and finally my experience.

Keywords Ant Colony Algorithm, Particle Swarm Algorithm, Niche, Probability-based Evolutionary Algorithm, DEGLSO

1 Introduction

Optimization problem is to find the optimal solution among a given objective function, a given number of solutions, and adjustable parameters, and to find the most suitable parameter value under certain constraints. They are widely found in many fields such as signal processing, image processing, and task assignment. It is an applied technique based on mathematics for solving various optimization problems. It is widely used in many of the above-mentioned fields and has also generated significant economic benefits. The optimization method has proven to improve system efficiency, reduce energy consumption, and use resources rationally, and this effect becomes more obvious as the size of the processing object increases.

It is not difficult to know that in real life, many complex combinatorial optimization problems constantly arise in various academic disciplines. In the face of these large optimization problems, traditional optimization methods transform the multi-objective optimization problem into a single-objective optimization problem by certain artificial approach, and then solve the transformed single-objective optimization problem. The commonly used methods are: objective weighting method, constraint method and objective programming method. However, the traditional optimization algorithms can only find the local optimal solution of the optimization problem, and the result of the solution strongly depends on the initial value. For example, many engineering optimization problems often require finding optimal or near optimal solutions in a complex and large search space. Given the many characteristics of practical engineering problems, such as complexity, nonlinearity, constraint and difficulty in modeling, the search for efficient optimization algorithms has become one of the main research contents of artificial intelligence.

Inspired by human intelligence, the social nature of biological groups, or the laws of natural phenomena, swarm intelligence methods have been proposed: methods that bring together swarm intelligence to collaboratively solve large-scale complex problems. For example, multi-optimized single-solution algorithms, mainly including: genetic algorithms that imitate the evolutionary mechanism of organisms in nature; Ant Colony Optimization that simulate the collective path-finding behavior of ants; Particle Swarm Optimization that simulate the behavior of flocks of birds and schools of fish, and so on. These algorithms have in common that they are developed by simulating or revealing certain phenomena and processes in nature or the intelligent behavior of groups of organisms. In the field of optimization they are called intelligent optimization algorithms, which are simple, general and easy to process in parallel. However, traditional intelligent optimization algorithms suffer from the problem of falling into a local optimum solution so it is proposed to modify the parameters or use adaptive modification of the relevant operators to improve the scalability of the algorithm. While facing the multi-solution multi-optimization problem, it is proposed that the small habitat strategy combined with the evolutionary algorithm based on probability distribution improves the exploration and exploitation ability of the group intelligent collaborative search through segmentation domination and hierarchical learning, but the traditional algorithm, may be poor in performance, so the distributed elite learning strategy is proposed, which reduces the transmission and time complexity, but still maintains the group global search and local exploitation capabilities.

This paper is divided into six areas: (I) Introduction (II) Research background and traditional genetic algorithms (III) Predictability of swarm intelligence: a framework for the evolution of swarm intelligence based on probability distribution (V) Scalable swarm intelligence: distributed collaborative swarm intelligence evolutionary optimization (VI) swarm intelligence optimization applications (VII) Summary

2 Research Background and Traditional Genetic Algorithms

Reasons for research on population intelligence methods and applications for large-scale optimization. First, the national demand, in the "new generation of artificial intelligence development plan" clearly pointed out swarm intelligence methods research as one of the five major development directions. Second, it is a disciplinary frontier, an important method for solving complex large-scale problems that cannot be solved by single-individual or centralized methods, with several branches selected as ESI research frontiers.

2.1 The Challenge of Large-Scale Problems: Effectiveness and Efficiency

2.1.1 Dramatic increase in local optimal

Global optimum means that a particular decision or choice is the best among all decisions or choices. The local optimum, on the other hand, is different from the global optimum in that it is not required to be the best among all decisions. It means that a particular decision or choice is the best among a portion of the decisions or choices. It is defined by a mathematical formula as follows.

$$f(x_0) = minf(x), x_0 \in D$$
$$f(x_0) = maxf(x), x_0 \in D$$

However, the problem of "premature convergence" is magnified when the number of local optimization regions increases sharply. What is premature convergence? Premature convergence is a phenomenon that cannot be ignored in genetic algorithms, mainly when all individuals in the population tend to the same expression and terminate their evolution, and eventually, the algorithm does not give an excellent solution.

2.1.2 Exponential expansion of the solution space

As the size of the group increases, there is an urgent need to improve search diversity.

2.1.3 Increased Time Complexity

Severe reduction in computational efficiency.

2.2 Traditional Genetic Algorith

Back up to another subsection, still within the Discussion section so that we can continue to inform and amaze.

biological terms

Genotype: the internal expression of the chromosome of a sex trait.

Phenotype: the external expression of a chromosomally determined trait or, alternatively, the external expression of an individual formed according to the genotype.

Evolution: the gradual adaptation of a population to its living environment and the continuous improvement of its quality. The evolution of organisms takes place in the form of populations.

Fitness: a measure of how well a species is adapted to the environment in which it lives.

Selection: Selecting a number of individuals from a population with a certain probability. In general, selection is a process of meritocracy based on fitness.

Reproduction: when a cell divides, the genetic material, DNA, is transferred to the newly created cell by replication, and the new cell inherits the genes of the old cell.

Crossover: DNA is cut at an identical position on two chromosomes and the two strings before and after are crossed and combined to form two new chromosomes. Also called genetic recombination or hybridization.

Mutation: some replication errors may (with a small probability) occur during replication, and mutations produce new chromosomes that express new traits.

Coding: genetic information in DNA is arranged in a certain pattern on a long strand. Genetic coding can be seen as a mapping from expression to genotype.

Decoding (decoding): mapping of genotypes to expressions.

Individual: an entity whose chromosomes with characteristics.

Population: the set of individuals, and the number of individuals within that set is called the population.

Simple process of algorithm implementation

1. Establish the mapping between expression and genotype <-> digital encoding scheme

   Encode as a binary string

2. the first kangaroos are scattered randomly on the mountain range <-> initialize a population with random numbers and the individuals of the population are these digitized codes

3. get the kangaroo location <-> decoding process

4. the higher the kangaroo climbs, the more popular it is and the higher the fitness <-> do a fitness assessment for each individual using the fitness function

   Measure: the altitude of the kangaroo's location. The altitude of the kangaroo can be used directly as their fitness score. The fitness function can return the function value directly.

5. Every so often, shoot some kangaroos on the mountain that are at a lower elevation, so that the overall number of kangaroos is equal <-> select on merit with the selection function

   choice function

   Roulette wheel selection: A playback random sampling method in which the probability of each individual entering the next generation is equal to the ratio of its fitness value to the fitness value of the individuals in the whole population.

   Random competitive selection: Each time a pair of individuals is selected according to the roulette wheel, and then these two individuals are allowed to compete, and the one with higher fitness is selected, repeatedly, until the selection is full.

   Optimal preservation strategy: The individual with the highest fitness in the current population does not participate in the crossover and variation operations, but uses it to replace the individual with the lowest fitness in the current generation population after the crossover and variation operations.

   Randomized league selection: The individual with the highest fitness among several individuals is selected at a time to be inherited into the next generation population.

   Exclusion selection: the newly generated offspring will replace or exclude similar individuals of the old sire, increasing the diversity of the population.

6. Individual kangaroo genetic recombination or mutation <-> generating good offspring

   Gene recombination/crossover

   The process of gene exchange of binary codes is very similar to the process of homologous chromosome association as taught in high school biology. A number of codes located at the same position are randomly swapped to produce a new individual. In the case of a gene containing multiple floating point codes, the floating point number can also be used as the base unit to randomly generate a value between the parents gene code value as the child gene code value.

Crossover operator for binary-encoded individuals or floating-point-encoded individuals.

One-point Crossover: refers to setting only one random crossover point in the coding string of an individual, and then exchanging part of the chromosomes of two paired individuals with each other at that point.

Two-point Crossover and Multi-point Crossover

- Two-point crossover: Two crossover points are randomly set in the coding strings of individuals, and then some genes are exchanged.
- Multi-point Crossover : Multiple crossover points are set

Uniform Crossover: Genes at each locus of two paired individuals are exchanged with the same crossover probability, resulting in two new individuals.

Arithmetic Crossover: Two new individuals are created by the linear combination of two individuals. The object of this operation is usually an individual represented by a floating-point code.

Genetic variation

Simple Mutation: The mutation operation is performed on the value of one or more randomly specified motifs in an individual coding string with a mutation probability.

Uniform Mutation: Replacing the original gene values at each locus in an individual coding string with a random number that fits into a range of uniform distribution with a small probability. (Especially suitable for the initial stage of the algorithm)

Boundary Mutation: Randomly replaces the original gene value by one of the two corresponding boundary gene values in the locus. This is particularly suitable for problems where the optimal point is at or near the boundary of the feasible solution.

Non-uniform mutation: A random perturbation of the original gene values, and the result of the perturbation is used as the new gene value after the mutation. After the mutation operation is performed with the same probability for each locus, it is equivalent to a slight change of the entire solution vector in the solution space.

Gaussian approximate variation: the variation operation is performed by replacing the original gene value with a random number from a normal distribution with sign mean P and variance P^2.

Basic Process

This is just a brief look at the most basic genetic algorithms, and what follows is a description of several commonly used probability distribution-based frameworks for swarm intelligence evolution.

3 Population intelligence predictable: a framework for evolution of swarm intelligence based on probability distribution

3.1 Single-solution evolutionary algorithm for multi-peaked problems

Approaching the optimal solution of the problem by iterative evolution of the population

Algorithm 1 The Basic Framework of the Evolutionary Algorithm

```
Input: population size: NP, maximum number of iterations: g_max;
Output: global optimal solution
1: initialize the population and evaluate each individual fitness value;
2: generation = 0;
3: while generation=g_max do
4:     obtain the offspring individuals based on the parent individuals by manipulating the operator, etc;
5:     evaluate the fitness values of the offspring individuals;
6:     eliminate the poor parent individuals with the child individuals and update the global optimal solution Gbest;
7:     generation++;
8: end while
```

No evolutionary algorithm uses a different evolutionary algorithm uses a different evolutionary mechanism and evolutionary operator, next I will introduce in detail the ant colony algorithm and particle swarm optimization algorithm.

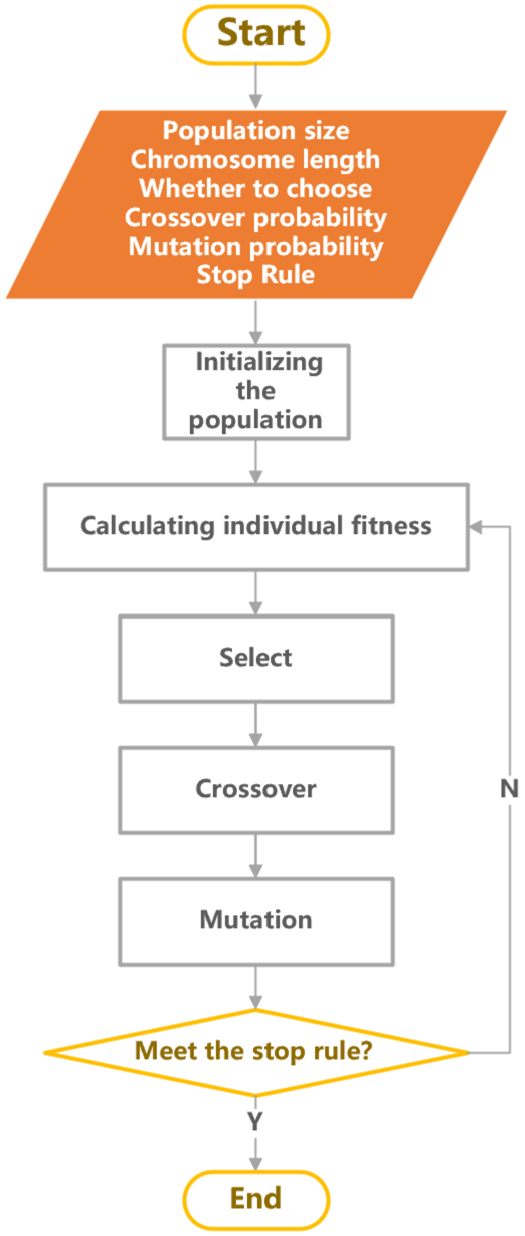

Figure 1: Genetic Algorithm

3.2 Description of the basic principles of the ant colony algorithm

Ant colony algorithm is a classical probabilistic algorithm used to find the optimal path in 1992, which was proposed by Italian scholars Dorigo M et al. based on the foraging behavior of ants in nature. When ants search for food, they will leave unique identification biometric information ------ pheromone for communication among their peers, similar to human language communication. The ants will also choose their next step according to the pheromone concentration left by their peers in multiple paths, generally following the path with higher "pheromone" concentration, and each ant will also leave pheromone on the path it is walking. As the pheromone concentration increases, ants are more likely to choose certain paths, thus forming a positive feedback mechanism.

There are now also monitored experiments on path selection by pheromone concentration in certain species of ants. For example, the double bridge experiment designed by Deneubourg and colleagues was completed. It was obtained that ants use information about changes in their surroundings to indirectly transmit information as they move, thus adjusting their own group behavior.

Important principles of ant colony algorithm implementation

1. Obstacle blocking: When an obstacle blocks the movement direction of an ant, the ant will randomly choose another direction (except with pheromone guidance)
2. Dispersing pheromones: Each ant emits the most pheromones at the beginning when it is looking for food, and the pheromones it emits decrease as it walks more distances.
3. Range: Ants had a limited observation range
4. Movement rules: Ants also choose their next steps based on the pheromone concentration left by their companions in multiple paths, generally following the path with higher "pheromone" concentration. Avoid circling in place.
5. Foraging rule: If ants find a food source within their sensory range, they go straight to it instead of taking the path with more pheromones,

and each ant is allowed a small probability of making a mistake and not going towards the path with the most pheromones to break the local optimum.

Traveling salesman problem A , who travels salesman from his home to a given n cities on business, wants to find a shortest path to each city and visit each city only once.

Ant colony algorithm to solve the TSP problem Basic Process

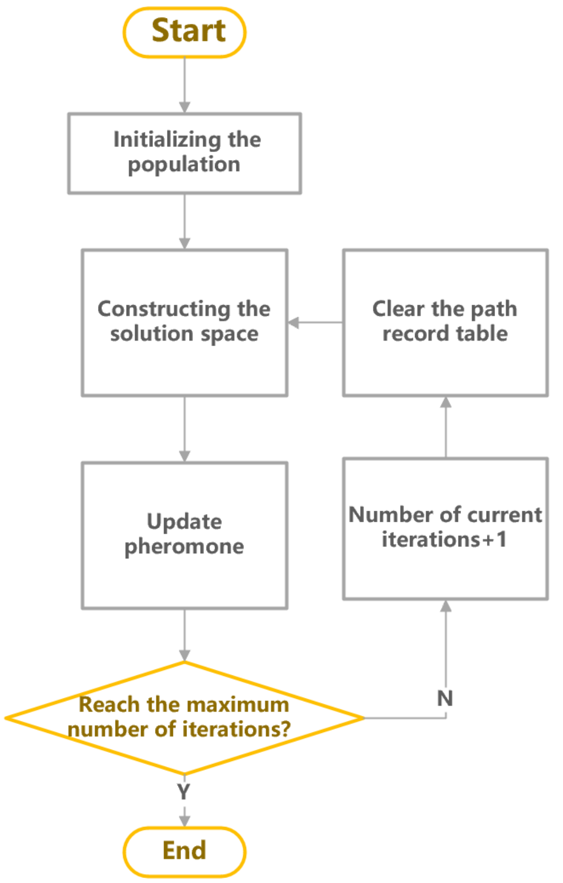
Start
Initializing the population
Constructing the solution space
Clear the path record table
Update pheromone
Number of current iterations+1
Reach the maximum number of iterations?
N
Y
End

Figure 2: AC

See Appendix I for detailed codes.

3.3 Description of the basic principles of the particle swarm algorithm

Particle swarm optimization (PSO) was proposed by Kennedy and Eberhart in 1995. The algorithm is a modification of Hepper's model for simulating a flock of birds (flock of fish) so that the particles can fly to the solution space and land at the best solution, resulting in a particle swarm optimization algorithm. The basic core is to use the sharing of information among the individuals in the flock so that the motion of the whole flock evolves from disorder to order in the problem solution space, thus obtaining the optimal solution to the problem.

Imagine a scenario where a flock of birds is foraging and there is a cornfield in the distance, all birds do not know where the cornfield is but know how far their current position is from the cornfield. The best strategy to find the cornfield, and the simplest and most effective one, is to search the area around the nearest flock of birds, and PSO is an optimization model inspired by this flock foraging behavior.

In PSO, the solution of each optimization problem is a bird in the search space, called a "particle", and the optimal solution of the problem corresponds to the cornfield that the flock is looking for. Each particle has a position vector (the position of the particle in the solution space) and a velocity vector (which determines the direction and speed of the next flight), and can calculate the adaptation value of its current position according to the objective function, which can be interpreted as the distance to the "corn field". At each iteration, the particle learns from its historical position and also from the "experience" of the best particle in the population to determine how to adjust and change the flight direction and speed at the next iteration. After each iteration, the entire population of particles eventually converges to the optimal solution.

Basic framework

PSO's inertia weighting model

Velocity vector

$$V_i^j \to \omega V_i^j + c_1 r_1^j (PBest_i^j - X_i^j) + c_2 r_2^j (GBest^j - X_i^j) \qquad (1)$$

$$j = 1, 2, ..., n$$

$$w \in [0,1],\ r_1, r_2 \in [0,1],\ c_1, c_2 = c$$

$$w\ is\ the\ PSO\ inertia\ weight\ \ w = 0.9w;$$

$$c_1, c_2 \; are \; the \; learning \; factors;$$

$$r_1, r_2 \; between \; [0,1] \; random \; probability \; values.$$

Position vector

$$X_i^j = X_i^j + V_i^j \qquad (2)$$

Steps

1. Initialize the population, calculate the individual fitness value, select the local optimal position vector Pbest of the individual and the global optimal position vector Gbest of the population
2. Set the number of iterations g_max, and make the current iteration g=1
3. Update the individual velocity according to Equation (1)
4. Update the position vector of each individual according to Equation (2)
5. Update the local position vector and global position vector: update the Pbest of each individual and Gbest of the population
6. When the number of iterations all reach g_max, output Gbest if it is satisfied; otherwise continue to iterate

Perception of inertia weights

The influence of the past motion state of the particle on the current behavior is similar to the inertia mentioned in physics. If w«1, the previous motion state can rarely affect the current behavior and the particle's velocity will change quickly; on the contrary, with larger w, although there will be a large search space, it is difficult for the particle to change its motion direction and converge to a better position. A higher w setting promotes global search, and a lower w setting promotes fast local search.

Traveling salesman problem

See Appendix I for detailed codes.

Traditionally, multi-peaked optimization problems generally refer to single-solution problems, and such problems generally hope to find the global optimal

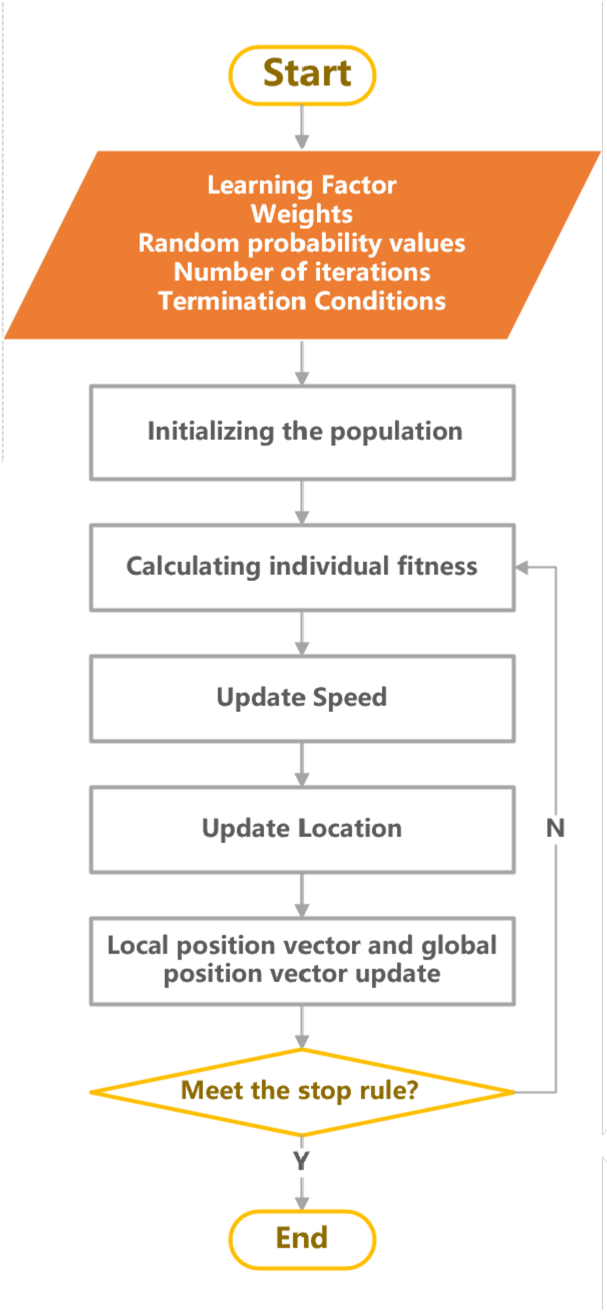
Start
Learning Factor
Weights
Random probability values
Number of iterations
Termination Conditions
Initializing the population
Calculating individual fitness
Update Speed
Update Location
Local position vector and global position vector update
Meet the stop rule?
N
Y
End

Figure 3: PSO

solution in the solution space. Due to the large number of local optimal solutions of the problem, the algorithm is prone to fall into local optimal solutions and premature convergence, and to overcome such drawbacks, it is necessary to maintain high search diversity to help jump out of the local optimal region, and also to save the number of adaptation value evaluations and improve the efficiency of the algorithm search.

Scholars have studied and proposed some improvements from three main perspectives

1. adjusting the parameters of evolutionary algorithms. For example, the performance of genetic algorithms is affected by crossover and variance probabilities, and particle swarm optimization algorithms are affected by inertia weights and learning factors. Also, the choice of these control parameters varies for different types of optimization problems and at different stages of evolution. Scholars then raise the question of how to dynamically and adaptively select appropriate control parameters for the algorithm. Currently, there are two types: 1. evolutionary state-aware adaptive control parameter methods, which analyze the adaptation values and distribution characteristics of the population, perceive the evolutionary state of the algorithm in the evolutionary process, and finally set different parameter adjustment strategies according to the characteristics of the evolutionary state in which the algorithm is located 2. parameter self-adjustment strategies, which adjust the parameters according to the adaptation values

2. operator of evolutionary algorithm, Kennedy et al. proposed learning from the historical optimal solution lbest of some neighborhood of an individual instead of gbest; Cheng et al. proposed a particle swarm algorithm with competitive learning for large-scale optimization problems. Chen Weineng et al. proposed a dominant learning mechanism based on chunking, drawing on the competitive strategy, and introduced the PSO algorithm, forming the PSO algorithm based on the chunking dominant learning mechanism

3. hybrid evolutionary algorithm, which improves the performance of the algorithm by combining multiple evolutionary algorithms.

3.4 Multi-solution optimization algorithms for multi-peaked problems

There may be more than one acceptable approximate global optimal solution in practical applications, then the algorithm needs to find these solutions as much as possible and provide better decision support, compared to a single solution, it needs to have a higher search diversity capability, the current mainstream strategies are: Niche strategy, adaptive evolution strategy based on probability distribution. The Niche strategies are described in detail below.

Niche : is a survival environment in a specific environment where organisms, in their evolutionary process, generally always live together with their own identical species and reproduce together. A simple description of the technique is to divide each generation of individuals into several classes, and select a number of individuals with greater adaptability in each class as the best representatives of a class to form a group, and then in the population, and between different populations, cross mutate to produce a new generation of individual groups. This Niche genetic algorithm can better maintain the diversity of solutions, and at the same time has a high global optimization-seeking ability and convergence speed, which is especially suitable for optimization problems with complex multi-peaked functions. The Niche techniques include pre-selection-based Niche habitat implementation method, exclusion model-based small habitat implementation method, and shared function-based small habitat implementation method.

Min Zheng and Junbo Gao had proposed an improved genetic algorithm for Niche. The algorithm introduces a pre-selection mechanism in the Niche genetic algorithm based on the elimination similarity mechanism, and improves the adaptive crossover probability operator and variation probability operator to dynamically adjust the crossover probability and variation probability size of individuals according to the size of the population fitness value.

Simple flow chart

Detailed derivation process

1. Initialize the population to determine the individual coding length: GL*VN + VN, M: number of populations, GL: individual coding

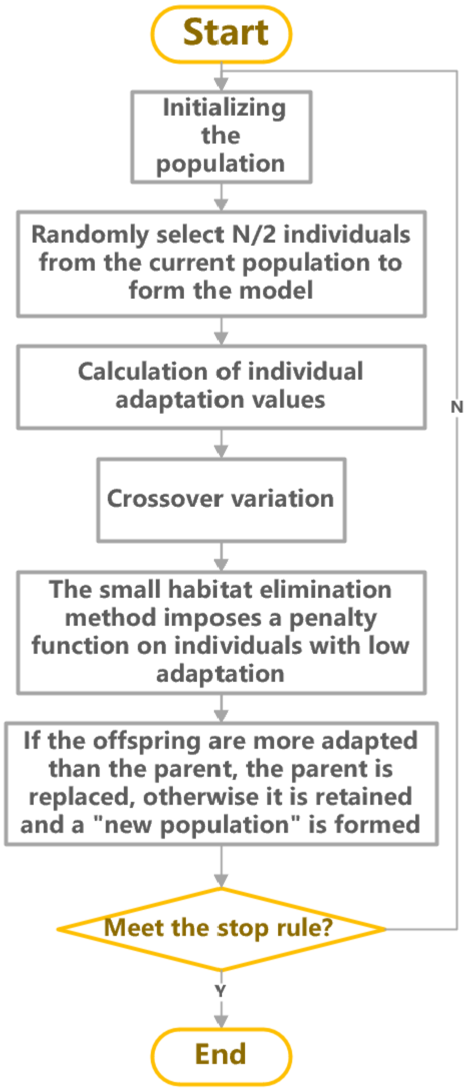
Start
Initializing the population
Randomly select N/2 individuals from the current population to form the model
Calculation of individual adaptation values
Crossover variation
The small habitat elimination method imposes a penalty function on individuals with low adaptation
If the offspring are more adapted than the parent, the parent is replaced, otherwise it is retained and a "new population" is formed
Meet the stop rule?
N
Y
End

Figure 4: Swarm Algorithm

length, VN is the number of independent variables

2. Construct the fitness function:

$$f(x_1,x_2) = \begin{cases} 1 + 0.01 * f_s(x_1,x_2) & f_s(x_1,x_2) < 0 \\ 1 & f_s(x_1,x_2) \geq 0 \end{cases} \quad (1)$$

$$f(x_1,x_2) : Individual\ adaptability$$

$$f_s(x_1,x_2) : The\ value\ of\ the\ function$$

$$Shubert\ corresponding\ to\ the$$

$$function\ on\ the\ x_1, x_2\ coordinates.$$

3. Selection: Pre-selection mechanism, only when the newly generated offspring individuals are more adaptive than the parent individuals, the generated offspring individuals can replace the parent individuals, and then can be inherited to the next generation. Among them, the offspring individuals are structurally similar to the parent individuals, so the replacement of structurally similar individuals can still effectively maintain species diversity.

4. Crossover: If the populations are crossed over with equal probability, it does not guarantee the degree of population optimization. In the process of evolution, the crossover operator size is adjusted according to the fitness value. For example, the crossover probability is higher for individuals with small fitness. Later in the evolutionary process, the individuals are all close to the average fitness level, and a smaller crossover probability at this point is beneficial to retain the optimal individuals for the next generation.

$$P_c(k) = \begin{cases} 0.8 - 0.3(1 - \frac{f(k)}{f_{avg}})^2 & f(k) \leq f_{avg} \\ 0.8 & f(k) > f_{avg} \end{cases} \quad (2)$$

$$p_c(k) : The\ kth\ individual\ crossover\ probability.$$

$$f_{avg} : Average\ adaptation\ of\ contemporary$$

$$populations$$

$$f(k) : The\ kth\ individual\ adaptation.$$

5. Mutation: In the process of biological variation and evolution, when the fitness is significantly lower or higher than the average, the probability is that it has mutated. In order to maintain

the overall optimization of the population, individuals below the average have a higher probability of mutation, while those above the average are likely to evolve slowly with a small probability.

$$P_m(k) = \begin{cases} p_m - 0.05(f(k) - f_{avg}) & f(k) \le f_{avg} \\ p_m(k) = p_m & f(k) > f_{avg} \end{cases} \quad (3)$$

$$p_m(k) : Probability\ of\ the\ kth\ individual\ variation.$$

6. Small habitat elimination method: in each generation of the population, first compare the distance between two two individuals, within a predetermined range L, in comparing the adaptation between the two, apply a stronger penalty function to the lower adaptation, reduce the adaptation value, after the treatment of individuals in the subsequent evolution is more likely to be eliminated. Haiming Distance between the i and j:

$$\|X_i - X_j\| = \sqrt{\sum_{k=1}^{M}(X_{ik} - X_{jk})^2} \quad (4)$$

$$i = 1, 2, ..., M + M - 1$$

$$j = i + 1, ..., M + N$$

$$M : Chromosome\ length$$

7. Applied to the Shubert function test.

$$minf_s(x_1, x_2) = \left\{ \sum_{i=1}^{5} icos\left[(i+1)x_i + i\right] \right\} \quad (5)$$

$$x \left\{ \sum_{i=1}^{5} icos[(i+1)x_2 + i] \right\}$$

$$-10 \le x_i \le 10,\ i = 1, 2$$

4 Group intelligence can be guided: hierarchical learning group intelligence evolutionary optimization

Hierarchical learning and its advantages.

Each dominant individual can be learned: a large number of role models are set in the population, maintaining good search diversity.

The higher the level of individuals, the higher the probability of being learned, the higher level particles concentrate on developing the solution space, the bottom level particles concentrate on exploring the solution space, and rely on population self-organization to achieve equilibrium.

Extension 1: **Segmented dominance learning**

Based on the competition mechanism, the population is divided into a better set of individuals RG and a worse set of individuals RP.

Segment each worse particle dimension and learn each segment to a better particle in RG Main value: Improving exploration and exploitation of population intelligence for collaborative search through hierarchical learning and segmentation domination.

Extension 2: **Adaptive Dominance Learning**

Randomly select two individuals, if an individual is dominated by these two selected individuals, then learn from these two dominated individuals, otherwise go directly to the next generation, with the same time complexity as traditional PSO and half the space complexity of traditional algorithms.

Main value: reduce the space complexity of the method while ensuring exploration and exploitation capability.

5 Population Intelligence Scalable: Distributed Collaborative Population Intelligence Evolutionary Optimization

EDA is a class of evolutionary algorithms that has received a lot of attention in recent years. The algorithm evaluates the information of population evolution probability distribution based on the information of individuals in the current population and gives the sampling of this probability distribution information to generate children individuals, and the algorithm framework is as follows:

Algorithm 2 EDA

```
Input: population size NP, number of individuals M for distribution evaluation, maximum number of iterations g_max;
Output: global optimal solution.
1: initialize the population and evaluate the fitness value of each individual;
2: generation = 0;
3: while generation=g_max do
4:     select M individuals from the population, and evaluate the population distribution using these individuals;
5:     generate new individuals based on the evaluated distribution model by sampling and evaluating the fitness values of the new individuals;
6:     update the global optimal solution by selecting a new population to be generated;
7:     generation++;
8: end while
```

Even though probability-based evolutionary algorithms have advantages in maintaining search diversity, however, evolutionary algorithms based on probability distributions have still not been studied and applied in problems such as multi-solution optimization where search diversity is highly required. Thus, it has been proposed that by combining the small-habitat strategy with the probability distribution-based evolutionary algorithm, a new and effective way to solve complex multi-peak multi-solution optimization problems is provided.

Algorithmic framework

The idea of probability distribution-based evolutionary algorithm for multi-solution optimization is to combine the probability distribution evolution with the small habitat strategy to establish the probability distribution estimation for the region of high fitness value in the whole space s search, so as to further improve the search diversity of the algorithm and to guarantee the solution accuracy in combination with local search. 1.population initialization and evaluation of the fitness value of each individual.
2.Niche partitioning, using the Niche strategy to partition the population into several microhabitats.
3.probability distribution estimation, where the Niche are independent of each other and the probability distribution within each Niche is independent of each other. Based on the subpopulations corresponding to each Niche, the evolutionary distribution within each Niche is assessed. So that multiple probability distribution models are included within the whole population, corresponding to different regions of the solution space.
4.Each Niche merges the parent, and child individuals to obtain a new population using a nearest neighbor based elite selection strategy.
5.Local search, in order to improve the solution accuracy, the best individuals within each Niche are adaptively searched locally in the neighborhood by Gaussian distribution, thus improving the solution quality.
2-6 Iterate iteratively until the algorithm reaches the termination condition. The algorithm comes from Probability Distribution Based Evolutionary Computer Optimization.

Algorithm 3 Niche

```
Input: population size NP, number of groups N, number of local search points K, maximum number of iterations g_max
Output: Whole population
 1: generation = 0;
 2: while generation<=g_max do
 3:     Dividing the population into N groups, each group contains M = NP/N individuals;
 4:     for all $i \in N$ do
 5:         Evaluate the probability distribution of group i;
 6:         $\mu = \frac{\sum_{j=1}^{M} x_j}{M}$, $\delta = \sqrt{\frac{\sum_{j=1}^{M}(x_j-\mu)^2}{M-1}}$;
 7:
 8:         if ( thenrand(0,1)<0.5)
 9:             Cauchy($\mu, \delta$)
10:         else
11:             Gaussian($\mu, \delta$)
12:         end if
13:         for all $j \in M$ do
14:             Obtain the closest individual to the new individual within the group and replace if better
15:         end for
16:     end for
17:     Draw the best individuals from each group, put them into the seed set and calculate the probability of performing a local search for each seed
18:     for all $i \in N$ do
19:         if $rand(0,1) < P_{r_i}$ then
20:             for all $j \in K$ do
21:                 $Gaossian(s_i, 1.0E-4)$ If the individual is better, replace;
22:             end for
23:         end if
24:     end for
25:     generation++;
26: end while
```

Distributed Collaborative Intelligent Evolutionary Optimization

Distributed Elite Learning: DEGLSO

Model with central nodes

The role of the central node is to maintain the elite set

The role of an individual is to learn from and evolve the elites retained by the individual

Communication occurs when and only when individuals fail to discover a more optimal solution

Key values: low communication overhead, high parallelism granularity, and still maintain the global exploration and local development capabilities of the group

6 Group intelligence optimization application

6.1 Application: Large-scale water supply network optimization

Objective: Given a water supply pipeline network, select the appropriate parameters (material, radius) for each section of pipeline and find the minimum cost design solution that can meet the water supply demand

Challenges

Large scale of pipe network

It is not possible to evaluate the design solution with a precise expression of the objective function, and a simulation tool is needed for evaluation (EPANET)

Task partitioning

Determine the initial split based on the flow direction of any initial solution

Different solutions correspond to different flow directions, posing new challenges to evolution.

7 Summary

The paper took more than a month to write, in fact, strictly speaking this is a study note rather than an academic paper, all the content is based on the PowerPoint provided by Ms. Xiaomin Hu, I have almost no understanding of population intelligence, in which all the proper nouns appear, I do not understand, so find literature is also based on these proper nouns as a guide, gradually understand one by one to break.

So why did I choose the direction of genetic algorithm and population intelligence? Before that I had already had a deep understanding of computer vision, whether from the official website or from some other professionals' comments, I was fascinated by the related technologies, applied in animation or visualized in front of a game fan's eyes. I can imagine me applying certain technologies to realize AI combined with audio games or other kinds of games, so that players no longer feel that they are actually separated from the game by a screen, but can actually touch and have thoughts. Let the game has a soul. But after all, the game is virtual. To improve the material life of people does not seem to help much, of course, this is only on behalf of personal remarks.

Especially the current disaster we are facing - the covid-19, I fully feel the importance of science and technology, the important role of science and technology in fighting the epidemic and maintaining the safety of everyone's life. "The most powerful weapon for mankind to compete with diseases is science and technology, and mankind cannot overcome catastrophes and pandemics without scientific development and technological innovation." For example, I studied group intelligence this time, multi-solution multi-optimization problem, how to effectively allocate resources? In particular, some complex system decision-making tasks in open environments. For example, based on group research to get vaccines, group development to get software, how to make the efficient completion of the task in a short period of time, in this information environment of everything connected, group intelligence is increasingly useful and affects our lives all the time. He also went from a closed to open and competitive, now life, future life, optimize the distribution, optimize the decision, make people's life better and more convenient. This is the reason why I chose to study computer in the first place, right?

In returning to this thesis, I roughly learn the genetic algorithm, ant colony algorithm, particle swarm optimization algorithm and multi-solution multi-optimization method combining small habitat strategy with probability-based evolutionary algorithm, and know that the group intelligence prediction gets to find the most suitable solution in a certain range of solution space, and also know that the intelligence of a group is always limited, so I propose to explore the group intelligence collaborative optimization in distributed environment We propose to explore collaborative group intelligence optimization in a distributed environment to improve the efficiency and scalability of collaboration. However, the traditional master-slave distributed model, which is a simple population update followed by individual assignment, adaptation value calculation, etc., results in large communication volume, long communication waiting time, low granularity of distributed computation, and poor scalability. But in fact, there are many less necessary calculations. Scholars then proposed that distributed elite learning, where communication relationships occur only when individuals find a more optimal solution, or when individuals fail to evolve over time, reduces communication overhead and does not have a large impact on the population's global exploration and local exploitation capabilities.

The whole research process I found fascinating, not enough search power to enhance, not enough efficiency to find ways to optimize, for the ultimate better optimization of resources or to be useful in any scenario that needs to be allocated in life. It's fascinating, but I also see the challenge in it. Let's end with one of my favorite classic lines:i went to the woods because i wanted to live deliberately ...i wanted to live deep and suck out all the marrow of life!
to put to rout all that was not life...
and not when i came to die, discover that i had not lived...

I will encounter many difficult problems in the process of learning, I hope I can think more, stand on the shoulders of giants, and learn without end. Believe in yourself and keep going.

8 References

[1] Eberhart R C and Kennedy J. A new optimizer using particle swarm theory. 1995.

[2] Shi Y and Eberhart R C. A modified particle optimizer. 1998.

[3] Kennedy J. Particle swarm optimization. 2010.

[4] Chenbo Zeng Beijing University of Chemical Technology Research on population intelligence improvement algorithms and applications for multimodal optimization problems 2020.

[5] Lu Qing, Liang Changyong, Yang Shanlin Institute of Computer Network Systems, Hefei University of Technology Adaptive small habitat genetic algorithm for multimodal function optimization

[6] Zheng Min, Gao Junbo A small habitat genetic algorithm for multimodal optimization 2014.

[7] Gan J, Warwick K. A genetic algorithm with dynamic Nich niche clustering for multi modal function optimization. Proc. of the IEEE Conference on Evolutionary Computation. Piscataway, USA. 2001. 215–222

[8] Zhao Ya-Qin, Zhou Xian-Zhong. A new method of Chinese text clustering based on small habitat genetic algorithm. Computer Engineering,2006,32(6):206-208.

[9] Zhu Xiaorong,Zhang Xinghua. Global optimization of multi-peak functions based on small habitat genetic algorithm. Journal of Nanjing University of Technology,2006,28(3):39-43.

[10] Zhou M, Sun Shudong. Principles and applications of genetic algorithms. Beijing: National Defense Industry Press, 1999.

[11] Huang Q., Chen X. X.. The improvement of genetic algorithm for small habitats. Journal of Beijing University of Technology, 2004,24(8):675-678.

[12] Zheng S.R.,Lai J.M.,Liu G.L.,Tang G. An improved real number coding hybrid legacy algorithm. Computer Applications,2006,26(8):1959-1962.

[13] Zhang ZZ, Zhang QY. A small-habitat genetic algorithm for exact optimization. Computer Applications,2005,25(8):1903-1905.

A Appendix

A.1 Code I

```
import random
import numpy as np
import math

# Initialize the variable parameters location=
    30 * randn(40, 2)
# Number of ants
num_ant=200
# Number of cities
num_city=30
# Pheromone impact factor
alpha=1
# Desired impact factor
beta=1
# Volatility of pheromone
info=0.1
# Constant
Q=1
count_iter = 0
iter_max = 500
#dis_new=1000

# symmetric matrix, distance between
def distance_p2p_mat():
   dis_mat=[]
   for i in range(num_city):
      dis_mat_each=[]
      for j in range(num_city):
         dis=math.sqrt(pow(location[i][0]-location[j][0],
         \2)+pow(location[i][1]-location[j][1],2))
         dis_mat_each.append(dis)
      dis_mat.append(dis_mat_each)
   # print(dis_mat)
   return dis_mat

# Calculate the distance corresponding to all
    paths
def cal_newpath(dis_mat,path_new):
   dis_list=[]
   for each in path_new:
      dis=0
      for j in range(num_city-1):
         dis=dis_mat[each[j]][each[j+1]]+dis
      dis=dis_mat[each[num_city-1]][each[0]]+dis
      dis_list.append(dis)
   return dis_list


# Point-to-point distance matrix
dis_list=distance_p2p_mat()
# Convert to matrix
dis_mat=np.array(dis_list)
# Expectation Matrix
# Diagonal array is added because the divisor
    cannot be 0
e_mat_init=1.0/(dis_mat+np.diag([10000]*num_city))
diag=np.diag([1.0/10000]*num_city)
# Or make the diagonal element 0
e_mat=e_mat_init-diag
# Initialize the pheromone concentration of
    each edge, all 1 matrix
pheromone_mat=np.ones((num_city,num_city))
# Initialize each ant path, all starting from
    city 0
path_mat=np.zeros((num_ant,num_city)).astype(int)


# while dis_new>400:
while count_iter < iter_max:
   for ant in range(num_ant):
      # All from 0 cities
      visit=0
      # Cities not visited
      unvisit_list=list(range(1,30))
      for j in range(1,num_city):
      # Roulette method to select the next city
      trans_list=[]
      tran_sum=0
      trans=0
      for k in range(len(unvisit_list)):
         trans+=np.power(pheromone_mat[visit]
         \[unvisit_list[k]],alpha)
         \*np.power(e_mat[visit]
         \[unvisit_list[k]],beta)
         trans_list.append(trans)
         tran_sum =trans
      # Generate random numbers
      rand=random.uniform(0,tran_sum)

      for t in range(len(trans_list)):
         if(rand <= trans_list[t]):
            visit_next=unvisit_list[t]
            break
         else:
            continue
      # Fill the path matrix
      path_mat[ant,j]=visit_next
      # Update
      unvisit_list.remove(visit_next)
      visit=visit_next

   # After filling the path matrix of all
       ants, count the total distance
   # of each ant
   dis_allant_list=cal_newpath(dis_mat,path_mat)

   # Update shortest distance and shortest
       path for each iteration
   if count_iter == 0:
      dis_new=min(dis_allant_list)
      path_new=path_mat[dis_allant_list.index(dis_new)].copy()
   else:
    if min(dis_allant_list) < dis_new:
      dis_new=min(dis_allant_list)
      path_new=
      path_mat[dis_allant_list.index(dis_new)].copy()

  # Update the pheromone matrix
  pheromone_change=np.zeros((num_city,num_city))
  for i in range(num_ant):
    for j in range(num_city-1):
      pheromone_change[path_mat[i,j]]
      [path_mat[i,j+1]] +=
          Q/dis_mat[path_mat[i,j]][path_mat[i,j+1]]
    pheromone_change[path_mat[i,
    num_city-1]][path_mat[i,0]] +=
        Q/dis_mat[path_mat[i,
    num_city-1]][path_mat[i,0]]
  pheromone_mat=
  (1-info)*pheromone_mat+pheromone_change
  # Iteration count + 1, go to next iteration
  count_iter += 1

print("Shortest distance:",dis_new)
print("Shortest distance:",path_new)
```

B Appendix

B.1 Code II

```
import numpy as np
import matplotlib.pyplot as plt
import math

def getweight():
  # inertia weight
  w = 1
  return w

def getlearningrate():
  # The learning factor, also known as the
      acceleration constant
  c = (0.4961,1.4961)
  return c

def getmaxgen():
  # Maximum number of iterations
  g_max = 300
  return g_max

def getsizepop():
  # population size
  sizepop = 50
  return sizepop

def getrangepop():
  # Range limits for the particle's position,
      same limits for x and y  # directions

  rangepop = (-2*math.pi , 2*math.pi)
  #rangepop = (-2,2)
  return rangepop

def getrangespeed():
  # The speed range limit of the particle
  rangespeed = (-0.5,0.5)
  return rangespeed

def func(x):
  # x input particle position
  # y particle fitness value
  if (x[0]==0)&(x[1]==0):
    y =  np.exp((np.cos(2*np.pi*x[0])
    \+np.cos(2*np.pi*x[1]))/2)-2.71289
  else:
    y = np.sin(np.sqrt(x[0]**2+x[1]**2))
    /np.sqrt(x[0]**2+x[1]**2)
    \+np.exp((np.cos(2*np.pi*x[0])
    \+np.cos(2*np.pi*x[1]))/2)-2.71289
  return y

# Initialize the particle position, velocity,
    and adaptation values
def initpopvfit(sizepop):
  pop = np.zeros((sizepop,2))
  v = np.zeros((sizepop,2))
  fitness = np.zeros(sizepop)

  for i in range(sizepop):
    pop[i] =
        [(np.random.rand()-0.5)*rangepop[0]*2,
    \(np.random.rand()-0.5)*rangepop[1]*2]
    v[i] =
        [(np.random.rand()-0.5)*rangepop[0]*2,
    \(np.random.rand()-0.5)*rangepop[1]*2]
    fitness[i] = func(pop[i])
  return pop,v,fitness

def getinitbest(fitness,pop):
  # The population optimal particle positions
      and their fitness values
  gbestpop,gbestfitness =
      pop[fitness.argmax()].copy(),fitness.max()
  # individual optimal particle position and
      its fitness value, use copy()
  # so that changes to pop do not affect
      pbestpop, pbestfitness similar
  pbestpop,pbestfitness =
      pop.copy(),fitness.copy()

  return
      gbestpop,gbestfitness,pbestpop,pbestfitness

# weights
```

```
w = getweight()
# learning factor
c = getlearningrate()
# maximum number of iterations
g_max = getmaxgen()
# population size
sizepop = getsizepop()
# Location range limits
rangepop = getrangepop()
# Speed range limits
rangespeed = getrangespeed()
# Initialize particle position, velocity,
    adaptation values
pop,v,fitness = initpopvfit(sizepop)
# Optimal particle position and fitness values
gbestpop,gbestfitness,pbestpop,pbestfitness =
    getinitbest(fitness,pop)
# Result
result = np.zeros(g_max)
for i in range(g_max):
   t=0.5
   # update speed
   for j in range(sizepop):
      v[j] +=
          c[0]*np.random.rand()*(pbestpop[j]-pop[j])
      \+c[1]*np.random.rand()*(gbestpop-pop[j])
      v[v<rangespeed[0]] = rangespeed[0]
      v[v>rangespeed[1]] = rangespeed[1]

   # update location
   for j in range(sizepop):
      #pop[j] += 0.5*v[j]
      pop[j] = t*(0.5*v[j])+(1-t)*pop[j]
      pop[pop<rangepop[0]] = rangepop[0]
      pop[pop>rangepop[1]] = rangepop[1]

   #update fitness
   for j in range(sizepop):
      fitness[j] = func(pop[j])

   for j in range(sizepop):
      if fitness[j] > pbestfitness[j]:
         pbestfitness[j] = fitness[j]
         pbestpop[j] = pop[j].copy()

   if pbestfitness.max() > gbestfitness :
      gbestfitness = pbestfitness.max()
      gbestpop =
          pop[pbestfitness.argmax()].copy()

   result[i] = gbestfitness

plt.plot(result)
plt.show()
```