CLC ___________ Number___________

UDC___________ Available for reference ☐Yes ☐No

Southern University of Science and Technology

Undergraduate Thesis

Thesis Title:	**Reproduction and improvement of parallel exploration strategy based on Negatively Correlated Search**
Student Name:	**Yilai Chen**
Student ID:	**12013025**
Department:	**Department of Computer Science and Engineering**
Program:	**Computer Science**
Thesis Advisor:	**Assistant Professor Peng Yang**

Date: April 11, 2024

COMMITMENT OF HONESTY

1. I solemnly promise that the paper presented comes from my independent research work under my supervisor's supervision. All statistics and images are real and reliable.

2. Except for the annotated reference, the paper contents no other published work or achievement by person or group. All people making important contributions to the study of the paper have been indicated clearly in the paper.

3. I promise that I did not plagiarize other people's research achievement or forge related data in the process of designing topic and research content.

4. If there is violation of any intellectual property right, I will take legal responsibility myself.

Signature:

Date:

Reproduction and improvement of parallel exploration strategy based on Negatively Correlated Search

陈驿来

（计算机科学与工程系　指导教师：杨鹏）

[**ABSTRACT**]: This paper introduces NCNES[1], a more principled version of the negatively correlated search (NCS)[2] algorithm designed for effective exploration in search processes. The idea is given that more diversified populations will be more beneficial to search[3]. Unlike previous NCS approaches driven by intuition, NCNES provides a deeper understanding by framing parallel exploration as a process of seeking probabilistic models leading to high-quality solutions while remaining distinct from previous models. Empirical assessments using reinforcement learning, where exploration is crucial, demonstrate NCNES's advantages, particularly in games with uncertain and delayed rewards.

Subsequently, I apply NCNES to the CEC Benchmark 2017 without using neural networks, employing matrix operations directly to optimize the benchmark's objective function. Furthermore, I integrate NCNES with an NCNES-friendly Cooperative Co-evolution (CC) framework to scale-up NCNES while preserving its effective parallel exploration. This combination enhances the algorithm's performance, showcasing the effectiveness of NCNES in real-world optimization tasks.

In summary, NCNES offers a principled approach to effective exploration in optimization algorithms, with empirical results demonstrating its superiority in handling complex and uncertain search spaces. The integration with the CC framework further enhances its performance, making NCNES a promising choice for a wide range of optimization problems.

[**Key words**]: Evolutionary algorithms (EAs), Exploration, Benchmark

目录

1. Introduction

Evolutionary Algorithms (EAs) are a very important algorithm in fields such as reinforcement learning (RL) and benchmarking (Benchmarking). In these fields, the goal of optimizing search often involves a balance between exploration and exploitation, as well as efficient searches of large-scale search spaces. Evolutionary algorithms are known for their population characteristics and comprehensive exploration of the search space, and have gradually become one of the preferred methods to solve these complex problems.

NCS is the algorithm that originally proposed the concept of negative correlation. It has attracted great interest in the academic community so far[4-7] and has shown high performance in various test scenarios[8-12]. NCS is the algorithm that originally proposed the concept of negative correlation. It has aroused great interest in the academic community so far and has shown high performance in various test scenarios. However, it uses heuristic updates based on intuition, often lacking mathematical rigor. Thus introducing Negatively Correlated Natural Evolution Strategy (NCNES), an evolutionary algorithm (EA) based on Negatively correlated search (NCS) proposed in the paper[2], which uses an iterative method to explore and obtain the optimal solution. The driving idea of this algorithm is that creating some populations with a certain diversity can be more conducive to finding the optimal solution. NCNES will measure the distance between different populations to ensure that the population has a certain diversity. After integrating the evaluation value and diversity distribution of the existing populations, NCNES will generate the probability distribution value of the next generation population and generate the next generation population accordingly. generation population. Based on this, NCNES is characterized by focusing on the diversity of the probability distribution of solutions between two adjacent generations in the solution space, and consciously maintaining divergent global search elements.

Specifically, the Negatively Correlated Natural Evolution Strategy (NCNES) clearly divides the population into multiple sub-populations[13]. The evolution of each sub-population is treated as an independent search process and exploited by traditional evolutionary algorithms. At the same time, these search processes coordinately explore different search spaces by driving their probability distributions to be negatively correlated. The significant improvement of NCNES over NCS is that compared to the former which only proposes the concept of fuzzy negative correlation exploration, the latter models the mathematical process in detail, starting from initialization and population division to the end of each iteration. For the gradient update process, NCNES has formulated rigorous mathematical formula models, and formulated different learning rate formulas according to the model training progress. It has improved the strategy of negative correlation search ideas in all aspects, making it a rigorous algorithm.

In terms of optimization, NCNES performs gradient descent on two models at the same time. The result is a negative correlation natural evolution strategy, which can form parallel exploration search behavior. Different search processes will evolve in parallel to different but promising search space areas. In the later stage of the project, the Cooperative Co-evolution and CC strategy in the paper[14] will be introduced to expand the scale of EA by dividing the decision variables into multiple independent groups and processing smaller sub-problems (i.e., decision variable groups) respectively.

In order to verify the performance of the optimized NCNES, I will use CEC 2017 Benchmark to test and verify it. The CEC2017 test function set is quite widely used. It is somewhat challenging for the algorithm and has high academic recognition. All functions in the test set have been rotated and displaced, which increases the difficulty of algorithm optimization. The function mainly includes 30 test functions: F1 and F3 are uni-modal functions with no local minimum and only global minimum. These uni-modal functions can test the convergence ability of the algorithm; F4-F10 are multi-modal functions with local extreme points. These Multi-modal functions are used to test the ability of the algorithm to jump out of the local optimum; F11-F20 are hybrid functions that contain three or more CEC2017 benchmark functions after rotation/displacement, and each sub-function is given a certain weight; F21-F30 are composed of at least A composite function composed of three hybrid functions or rotational shifts of the CEC 2017 benchmark function. Each sub-function has an additional bias value and a weight. These combined functions further increase the optimization difficulty of the algorithm.

The main content of this article will be a detailed description of the NCNES algorithm, and an introduction to its improvements. Then it will introduce the introduced Cooperative Co-evolution (CC) framework, and finally analyze the results of the algorithm on benchmark tests such as CEC 2017 Benchmark. Its advantages and shortcomings that need improvement.

2. Negatively Correlated Natural Evolution Strategy

The origin of negative correlation search (NCS) is how to rethink the role of populations in search. In the process of teamwork, people generally believe that effective information sharing is needed between populations to avoid duplication of work and improve the efficiency and ability of information search. However, an unresolved problem is what information to share and how to share it. The idea of negative correlation requires individuals in the population to have different search behaviors by imitating the way humans cooperate to avoid repeatedly searching the same area of the search space. Each search behavior defines how offspring are sampled based on their parents, so I use a probability distribution to represent each generation's position in the probability distribution. Use mathematical correlations between distributions to statistically model diversity among populations. Therefore, by explicitly driving multiple probability distributions to be negatively correlated, this algorithm can effectively control the diversity of the next population.[15]

2.1 Introduction of the mathematical model

The basic idea of NCNES is to divide all solutions into lambda sub-populations, and each sub-population uses the same simple Gaussian distribution function for sampling. Following are the reasons why I use Gaussian distribution in this problem:

- Gaussian distribution is one of the most commonly used distributions in search;

- By using Gaussian distribution, $\nabla_{\theta_i} f(\theta_i)$has an analytical closed form, which facilitates efficient calculation

- Bhattacharyya distance can also be analytically calculated based on Gaussian distribution.

The iterative process can be described as targeting the adaptability evaluation value and the diversity evaluation value, and evolving independently through traditional evolutionary algorithms. Therefore, obviously, the new distribution should be able to sample new solutions with high fitness values. Furthermore, the new distributions should have less overlap (correlation) with existing distributions so that they can be used to sample different regions of the solution space.

Table 1 The introduction of the major mathematical denotations

Denotation	Description
λ	The number of sub-populations (search processes).
μ	The number of samplings (solutions) in each sub-population at each iteration.
$p(\theta_i)$	The probabilistic distribution of the ith sub-population with the parameter θ_i.
$f(x)$	The fitness value of a sample x.
$d(p(\theta_i))$	The diversity value between the ith probabilistic distribution and the others.
$\nabla_{\theta i}$	The partial gradient of a function with respect to θ_i.
$\mathcal{J}$	The reformed objective to be optimized in NCNES.
$\mathcal{F}$	The fitness model in the reformed objective.
$\mathcal{D}$	The diversity model in the reformed objective.
φ	The trade-off parameter balancing the fitness and diversity during the search.
$({}_i, \Sigma_i)$	The mean vector and covariance matrix of the ith probabilistic distribution.
F	The Fisher information matrix.

Next, I will introduce the specific process of evolution in a mathematical way. As mentioned before, our solutions are generated by specified probability distribution functions. Here, for the sake of uniformity, we set all distributions to be Gaussian distribution, while the parameters of the distribution, e.g., mean and covariance, can be different. Therefore, in the initial state, when all populations use the same probability density function, we can have the following function

$$\mathcal{J} = \int f(x)p(x|\theta_1)dx \tag{1}$$

We simply seems to only need to maximize this single fitness function. However, since the algorithm needs to explore multiple areas of the solution space in parallel at the same time, we then introduce the second Gaussian distribution function below. It should be noted that NCNES emphasizes the negative correlation of different search directions, so the function should be:

$$\mathcal{J} = \int f(x)p(x|\theta_1)dx + \int f(x)p(x|\theta_2)dx - C(p(\theta_1), p(\theta_2)) - C(p(\theta_2), p(\theta_1)) \tag{2}$$

Among them, $C(p(\theta_i), p(\theta_j))$ represents the correlation value between the i-th distribution and the j-th distribution, the closer the sample distributions are to each other, the greater the correlation value is. Now we generalize the distribution to multiple populations and multiple distributions, and we will get:

$$\mathcal{J} = \sum_{i=1}^{\lambda} \int f(x)p(x|\theta_i)dx + \sum_{i=1}^{\lambda} \sum_{j=1, i\neq j}^{\lambda} (-C(p(\theta_i), p(\theta_j))) \tag{3}$$

Now we have preliminarily obtained the optimization objective, which consists of two addends, which we call $\mathcal{F}$ and $\mathcal{D}$. To put it figuratively, we take the role of $\mathcal{F}$ as to evaluate the expectation of the performance value of the solutions of the current population based on the current distribution function; the role of $\mathcal{D}$ is to construct a distribution model between the current population and other populations. That is to say, for each population, when calculating the $\mathcal{D}$ value, it is necessary to construct its distance from all other $\lambda - 1$ populations and maximize it as much as possible. Therefore, $\mathcal{D}$ can be expressed as:

$$\mathcal{D} = \sum_{i=1}^{\lambda} \sum_{j=1, i\neq j}^{\lambda} (-C(p(\theta_i), p(\theta_j))) = \sum_{i=1}^{\lambda} d(p(\theta_i)), \tag{4}$$

Now, since we have incorporated all the factors to be optimized into the only objective $\mathcal{J}$, our only goal is to maximize the value of $\mathcal{J}$ to obtain the optimal solution to the evolution problem. And because the parallel search process of multiple populations we defined is independent during the optimization process, we can perform partial derivative descent on

$\mathcal{J}$ for each θ_i

$$\nabla_{\theta_i}\mathcal{J} = \nabla_{\theta_i}\mathcal{F} + \nabla_{\theta_i}\mathcal{D} = \nabla_{\theta_i} f(\theta_i) + \nabla_{\theta_i} d(p(\theta_i)), \quad i = 1, \ldots, \lambda \tag{5}$$

2.2 Update iteration process of NCNES model

Here, due to the results of the above inference, in order to obtain the maximized $\mathcal{J}$ value, we need to obtain partial derivatives for the $\mathcal{D}$ value and $\mathcal{F}$ value respectively, and perform gradient optimization on them. In mathematics, that means finding $\nabla_{\theta_i} f(\theta_i)$ and $\nabla_{\theta_i} d(p(\theta_i))$. For $\nabla_{\theta_i} f(\theta_i)$, we can continue to infer:

$$\nabla_{\theta_i} f(\theta_i) = \nabla_{\theta_i} \int f(x) p(x|\theta_i)\, dx \tag{6}$$

$$= \mathbb{E}_{\theta_i}[f(x)\nabla_{\theta_i} \log p(x|\theta_i)] \tag{7}$$

$$\approx \frac{1}{\mu}\sum_{k=1}^{\mu} f(x_k^i)\nabla_{\theta_i} \log p(x_k^i|\theta_i), \tag{8}$$

We can notice that $f(\theta_i)$ can be the expectation of $p(x|\theta_i)$,so we have (6) $f(\theta_i) = \int f(x)p(x|\theta_i)dx$。 Here x means the samples from probability $p(x|\theta_i)$. Next, we use the properties of the expected value to move the gradient ∇_{θ_i} inside the expected value. So we have (6) $\nabla_{\theta_i} f(\theta_i) = \nabla_{\theta_i} \int f(x)p(x|\theta_i), dx$。 We then exploit the properties of the log gradient to multiply the function $f(x)$ in the expected value by the log probability $\log p(x|\theta_i)$. So we have (7) $\nabla_{\theta_i} f(\theta_i) = \mathbb{E}\theta_i[f(x)\nabla\theta_i \log p(x|\theta_i)]$. Finally, we use the sampling method to approximate the expected value. Here we assume that we obtain the sample x_k^i through μ independent sampling, where $k = 1, 2, ..., \mu$. Then, we calculate the function value $f(x_k^i)$ for each sample and multiply it with the gradient of the logarithmic probability $\nabla_{\theta_i} \log p(x_k^i|\theta_i)$, and finally find average. In this way, we get an approximation of the gradient(8): $\nabla_{\theta_i} f(\theta_i) \approx \frac{1}{\mu}\sum_{k=1}^{\mu} f(x_k^i)\nabla_{\theta_i} \log p(x_k^i|\theta_i)$.

Now for calculating $\nabla_{\theta_i} d(p(\theta_i))$, it will be important to define $C(p(\theta_i), p(\theta_j))$. Here we use the Bhattacharyya distance as the negative correlation measure, namely, generally speaking for continuous distribution we have:

$$C(p(\theta_i),(\theta_j)) = -\log\left(\int \sqrt{p(x|\theta_i)p(x|\theta_j)dx}\right) \tag{9}$$

because in our problem it is a discrete distribution so we have :

$$C(p(\theta_i),p(\theta_j)) = -\log\left(\sum_{x\in X} \sqrt{p(x|\theta_i)p(x|\theta_j)}\right) \tag{10}$$

Then we solve for the gradient, for continuous distribution and discrete distribution respectively:

$$\nabla_{\theta_i} d(p(\theta_i)) = \sum_{j=1}^{\lambda} \nabla_{\theta_i} \log\left(\int \sqrt{p(x|\theta_i)p(x|\theta_j)dx}\right) \tag{11}$$

$$\nabla_{\theta_i} d(p(\theta_i)) = \sum_{j=1}^{\lambda} \nabla_{\theta_i} \log\left(\sum_{x\in X} \sqrt{p(x|\theta_i)p(x|\theta_j)}\right) \tag{12}$$

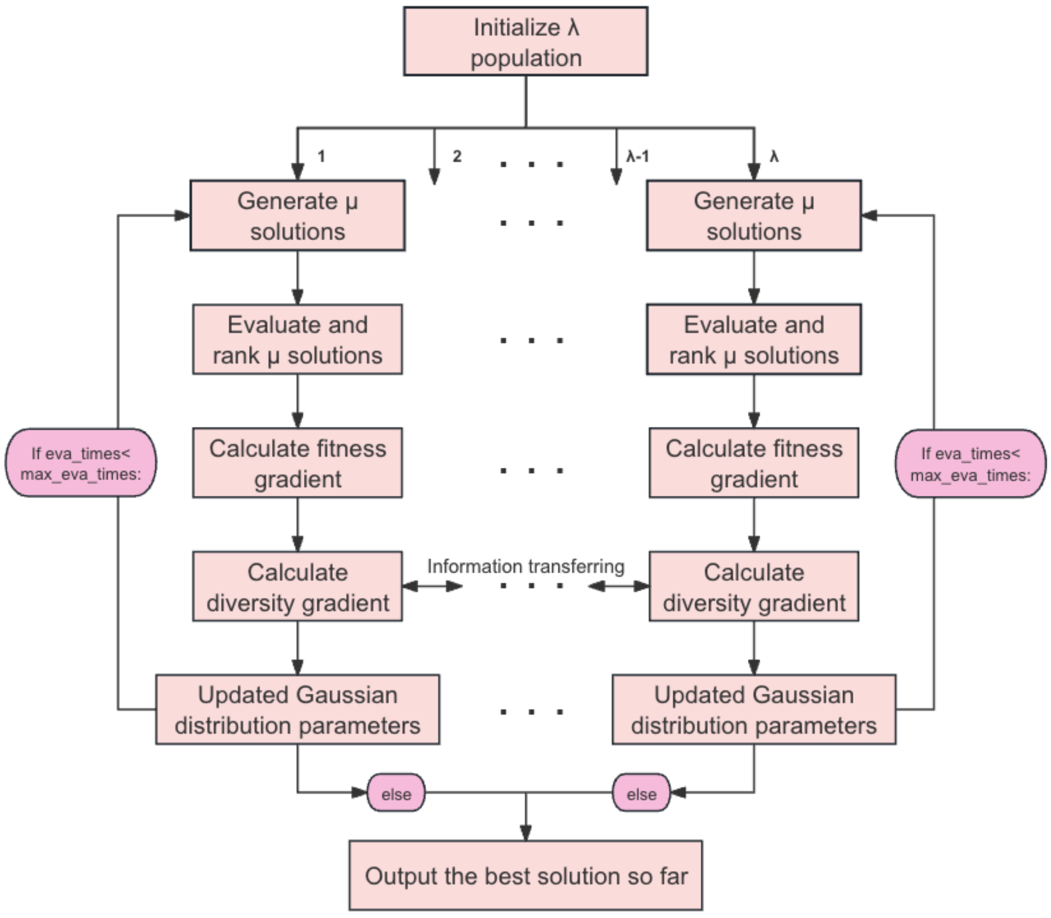
After that, since now we have $\nabla_{\theta_i} d(p(\theta_i))$ and $\nabla_{\theta_i} f(\theta_i)$, its easy to combine them to form $\nabla_{\theta_i}\mathcal{J}$. While at the mean time, it will be necessary to set a φ, acting as the trade-off parameter balancing the fitness and diversity during the search:

$$\nabla_{\theta_i}\mathcal{J} = \nabla_{\theta_i} f(\theta_i) + \varphi \cdot \nabla_{\theta_i} d(p(\theta_i)) \tag{13}$$

Finally, using the $\nabla_{\theta_i}\mathcal{J}$, we update θ_i by:

$$\theta_i = \theta_i + \eta \cdot \nabla_{\theta_i}\mathcal{J} \tag{14}$$

Here in 1, I drew a flow chart to intuitively illustrate the main steps of the algorithm. Specifically, first generate the population, then generate individuals, then evaluate the individuals, and then sort according to the fitness index of the individual, combined with the mean of the Gaussian distribution, to obtain the gradient value of fitness. , and then obtain the diversity value of the population and other populations based on the mean and covariance of the Gaussian distribution, and finally update the population based on the two.



Initialize λ population
1
2
λ-1
λ
Generate μ solutions
Evaluate and rank μ solutions
Calculate fitness gradient
Calculate diversity gradient
Information transferring
Updated Gaussian distribution parameters
If eva_times< max_eva_times:
else
Output the best solution so far

Figure 1 Theoretical algorithm flow chart

2.3 NCNES's specific implementation in Gaussian distribution

In the above, we only briefly elaborated on the main ideas and mathematical feasibility of the negative correlation natural evolution searching algorithm. Since this project uses Gaussian distribution to fit the distribution of each population, and each iteration updates the Gaussian distribution itself, and the individual solutions are generated from the updated Gaussian distribution, so next we will use the specific parameters m and $\sum$ of the Gaussian distribution and brought into the formulas inferred above, respectively, we can get:

$$\begin{aligned} \nabla_{m_i} f(\theta_i) &= \frac{1}{\mu}\sum_{k=1}^{\mu} \Sigma_i^{-1}(x_i^k - m_i)\cdot f(x_i^k), \\ \nabla_{\Sigma_i} f(\theta_i) &= \frac{1}{\mu}\sum_{k=1}^{\mu}\left(\frac{1}{2}\Sigma_i^{-1}(x_i^k - m_i)(x_i^k - m_i)^T\Sigma_i^{-1} - \frac{1}{2}\Sigma_i^{-1}\right)\cdot f(x_i^k), \end{aligned} \tag{15}$$

$$d(p(\theta_i)) = \sum_{j=1}^{\lambda} \frac{1}{8} (m_i - m_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2} \right)^{-1} (m_i - m_j) + \frac{1}{2} \log \left(\frac{|\frac{\Sigma_i + \Sigma_j}{2}|}{\sqrt{|\Sigma_i| \cdot |\Sigma_j|}} \right) \tag{16}$$

$$\nabla_{m_i} d(p(\theta_i)) = \frac{1}{4} \sum_{j=1}^{\lambda} \left(\frac{\Sigma_i + \Sigma_j}{2} \right)^{-1} (m_i - m_j),$$

$$\nabla_{\Sigma i} d(p(\theta_i)) = \tfrac{1}{4} \textstyle\sum_{j=1}^{\lambda} \left(\left(\frac{\Sigma_i + \Sigma_j}{2} \right)^{-1} - \frac{1}{4} \left(\frac{\Sigma_i + \Sigma_j}{2} \right)^{-1} (m_i - m_j)(m_i - m_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2} \right)^{-1} - \Sigma_i^{-1} \right). \tag{17}$$

$$\mathbf{F}_{m_i} = \frac{1}{\mu} \sum_{k=1}^{\mu} \Sigma_i^{-1} (x_i^k - m_i)(x_i^k - m_i)^T \Sigma_i^{-1})$$

$$\mathbf{F}_{\Sigma_i} = \frac{1}{4\mu} \sum_{k=1}^{\mu} \left(\Sigma_i^{-1} (x_i^k - m_i)(x_i^k - m_i)^T \Sigma_i^{-1} - \Sigma_i^{-1} \right) \left(\Sigma_i^{-1} (x_i^k - m_i)(x_i^k - m_i)^T \Sigma_i^{-1} - \Sigma_i^{-1} \right)^T \tag{18}$$

Formulas (15)-(17) describe that when using Gaussian distribution, $\nabla_{\theta_i} f(\theta_i)$ can be formed form $\nabla_{m_i} f(\theta_i)$ and $\nabla_{\Sigma_i} f(\theta_i)$. And $\nabla_{\theta_i} d(p(\theta_i))$ can be formed form $\nabla_{m_i} d(p(\theta_i))$ and $\nabla_{\Sigma i} d(p(\theta_i))$. Here, we need to introduce a Fisher matrix (18) to complete a task similar to normalization, because we notice that $\nabla m_i \mathcal{J} \propto \frac{1}{\Sigma_i}$ and $\nabla_{\Sigma_i} \mathcal{J} \propto \frac{1}{\Sigma_i{}^2}$, we need to use $\mathbf{F}_{m_i}$ and $\mathbf{F}_{\Sigma_i}$ to ensure that the size of each gradient update will not be extremely huge or minuscule, and control it within a reasonable range, but it can also reflect the relative size of the step size. As a result, we have:

$$\begin{aligned} m_i &= m_i + \eta_m \cdot \mathbf{F}_{m_i}^{-1} \cdot \nabla_{m_i} J, \\ \Sigma_i &= \Sigma_i + \eta_\Sigma \cdot \mathbf{F}_{\Sigma_i}^{-1} \cdot \nabla_{\Sigma_i} J, \end{aligned} \tag{19}$$

At the mean time, it is worth mentioning that according to the formula listed previously, when dealing with different problems, the corresponding fitness value will be very different, so it is necessary for us to take measures to unify the range of this value. Our way to do this is to first find the fitness value of all μ solutions in this population, and then rank them according to the fitness value from small to large. $\pi(k)$ represents the ranking of the k-th solution, and then rank them according to the following formula The fitness value of each solution is reassigned.

$$U_i(\pi(k)) = \frac{\max\left(0, \log(\frac{\mu}{2} + 1) - \log(\pi(k))\right)}{\sum_{j=1}^{\mu} \max\left(0, \log(\frac{\mu}{2} + 1) - \log(k)\right)} - \frac{1}{\mu} \tag{20}$$

In addition, for the update of the learning rate, it is how to adjust the step size parameters η_m and η_Σ. Specifically, a strategy is adopted here to dynamically adjust these two parameters so that they gradually decrease as the search proceeds(21). The parameter T_{max} in the formula represents the total time budget of the entire search process, and T_{cur} represents the currently consumed time. e is a natural constant. η_m^{init} and η_Σ^{init} are the initial values of the step size parameter respectively. The formula gradually decreases the two step parameters from the initial value to zero through an exponential decay function.

$$\begin{aligned} \eta_m &\leftarrow \eta_m^{init} \cdot \frac{e - e^{\frac{T_{cur}}{T_{max}}}}{e-1}, \\ \eta_\Sigma &\leftarrow \eta_\Sigma^{init} \cdot \frac{e - e^{\frac{T_{cur}}{T_{max}}}}{e-1} \end{aligned} \tag{21}$$

At the same time, I also added the attenuation of the ϕ value(22). I hope that the model will pay attention to the diversity index in the early stage of search, and only focus on the fitness value in the later stage of model training. This is more in line with the common sense of model search and learning.

$$\varphi \leftarrow \varphi\text{init} \cdot e^{-\frac{e^{T_\text{cur}}}{T_\text{max}} \cdot e^{-1}} \tag{22}$$

2.4 Implementation and detailed Pseudo-code

To summarize, NCNES is an evolutionary algorithm based on multivariate Gaussian distribution; each sub-population is driven by a distribution to drive its iterative optimization; through the proposed diversity model of negative correlation of multivariate Gaussian distribution, a negative correlation is formed between multiple Gaussian distributions. Therefore, it was named ”Negatively Correlated Natural Evolution Strategies” (NCNES). Detailed steps for NCNES are listed in the following Pseudo-code.

Algorithm 1 Pseudo-code of proposed NCNES

```
1:  for $i = 1$ to $\lambda$ do
2:      Initialize a Gaussian distribution for the $i$th Search Process as $N(m_i, \Sigma_i)$
3:  end for
4:  $T_{\text{cur}} = 0$;
5:  while $T_{\text{cur}} < T_{\text{max}}$ do
6:      $\eta_m \leftarrow \eta_m^{\text{init}} \cdot \frac{e - e^{\frac{T_{\text{cur}}}{T_{\text{max}}}}}{e-1}$;
7:      $\eta_\Sigma \leftarrow \eta_\Sigma^{\text{init}} \cdot \frac{e - e^{\frac{T_{\text{cur}}}{T_{\text{max}}}}}{e-1}$;
8:      $\varphi \leftarrow \varphi_{\text{init}} \cdot e^{-\frac{e^{T_{\text{cur}}}}{T_{\text{max}}} \cdot e^{-1}}$
9:      for $i = 1$ to $\lambda$ do
10:         Generate $\mu$ solutions $\mathrm{x}_i^k \leftarrow N(m_i, \Sigma_i), \forall k = 1, \ldots, \mu$
11:         Evaluate the fitness $f(x_i^k), \forall k = 1, \ldots, \mu$;
12:         $T_{\text{cur}} \leftarrow T_{\text{cur}} + \mu$;
13:         Update $x^*$ as the best solution ever found;
14:         Rank the $k$th solution in terms of its fitness $f(x_k)$ as $\pi(k), \forall k = 1, \ldots, \mu$;
15:         Set $U_i(\pi(k)) = \frac{\max\left\{0, \log\left(\frac{\mu}{2}+1\right) - \log(\pi(k))\right\}}{\sum_{k=1}^{\mu} \max\left\{0, \log\left(\frac{\mu}{2}+1\right) - \log(k)\right\}} - \frac{1}{\mu}$;
16:         $\nabla_{m_i} f \leftarrow \frac{1}{\mu} \sum_{k=1}^{\mu} \Sigma_i^{-1}(x_i^k - m_i) \cdot U_i(\pi(k))$;
17:         $\nabla_{\Sigma_i} f \leftarrow \frac{1}{2\mu} \sum_{k=1}^{\mu} \left(\Sigma_i^{-1}(x_i^k - m_i)(x_i^k - m_i)^T \Sigma_i^{-1} - \Sigma_i^{-1}\right) \cdot U_i(\pi(k))$;
18:         $\nabla_{m_i} d \leftarrow \frac{1}{4} \sum_{j=1}^{\lambda} \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} (m_i - m_j)$;
19:         $\nabla_{\Sigma_i} d \leftarrow \frac{1}{4} \sum_{j=1}^{\lambda} \left(\left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} - \frac{1}{4}\left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} (m_i - m_j)(m_i - m_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} - \Sigma_i^{-1}\right)$;
20:         $F_{m_i} \leftarrow \frac{1}{\mu} \sum_{k=1}^{\mu} \Sigma_i^{-1}(x_i^k - m_i)(x_i^k - m_i)^T \Sigma_i^{-1}$;
21:         $F_{\Sigma_i} \leftarrow \frac{1}{4\mu} \sum_{k=1}^{\mu} \left(\Sigma_i^{-1}(x_i^k - m_i)(x_i^k - m_i)^T \Sigma_i^{-1} - \Sigma_i^{-1}\right)\left(\Sigma_i^{-1}(x_i^k - m_i)(x_i^k - m_i)^T \Sigma_i^{-1} - \Sigma_i^{-1}\right)^T$;
22:         $m_i \leftarrow m_i + \eta_m \cdot F_{m_i}^{-1}(\nabla_{m_i} f + \varphi \cdot \nabla_{m_i} d)$;
23:         $\Sigma_i \leftarrow \Sigma_i + \eta_\Sigma \cdot F_{\Sigma_i}^{-1}(\nabla_{\Sigma_i} f + \varphi \cdot \nabla_{\Sigma_i} d)$;
24:     end for
25: end while
```

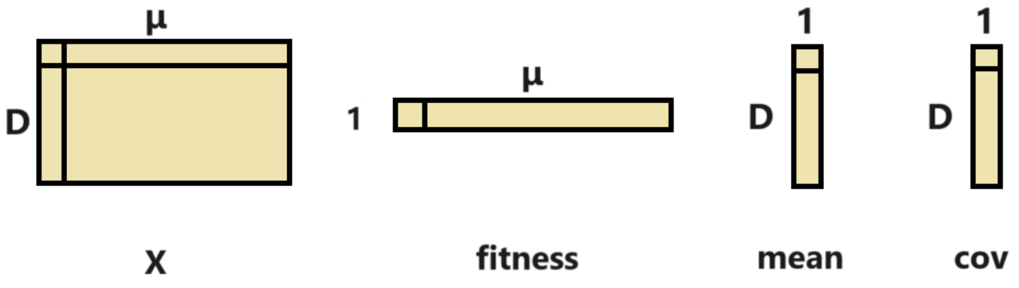


Figure 2 data structure of main variables used in the NCNES, where X refers to a population, fitness, mean, and cov represent the fitness values, mean and covariance of all solutions within the population.

3. Cooperative coevolutionary framework

Evolutionary algorithms work in most cases by random sampling in the search space. A common problem with evolutionary algorithms is that their performance usually decreases dramatically when the search space is significantly expanded. In order to solve this large-scale problem, one of the methods commonly used by people is CC[16], Cooperative coevolution. The main original intention of proposing the CC framework is to follow the divide and conquer strategy, divide a solution into different parts, and perform optimization iterations separately. The purpose is to make the sub-problems independent of each other as much as possible and not affect each other, so as to obtain better solutions to the problem with high efficiency. The specific approach can be summarized as first randomly generating a standard solution, then sequentially supplementing (covering) different low-dimensional partial solutions to the standard solution, and evaluating it using the previous evaluation criteria. Therefore, after adopting the CC strategy, under ideal circumstances, each sub-problem only needs to use evolutionary algorithms to deal with its own part, while also remaining independent of each other and not interfering with each other, so that large-scale problems can be effectively solved.

3.1 Comparison of CC frameworks based on different strategies

When talking about the idea of cooperative coevolution, it is easy to find that a major difficulty is how to evaluate the performance of each sub-problem solution when splitting a solution individual into multiple sub-problem solutions, because the existing evaluation function is based on the complete solutions. So that the current evaluation function cannot make a fair evaluation of a sub-solution. For this problem, traditional CC supplements each partial solution with the same supplementary vector in each iteration, completes it to the dimensions of the original objective function, and then evaluates and optimizes it. This approach is called an individual-centric method. In most evolutionary algorithm environments, this is a good strategy. It can not only greatly reduce the amount of calculation and improve efficiency, but also can omit the storage of the relative position of the solution and the parent solution. Waiting for information, this strategy has been successful in many algorithms. However, in the NCNES problem, due to the particularity of negative correlation search, each solution may be at a different (local) optimal value in the search space. If these partial solutions are supplemented with the same supplementary vector, it may cause the original solution to lose the particularity of its position in the solution space. So correspondingly, Popovici proposed a strategy such as shuffle-and-pair[17]. This strategy theoretically uses different completion vectors to supplement sub-solutions. Therefore, in the context of NCNES, we do not intend to let each the supplementary solutions of the sub-problems to be the same. I use the corresponding different supplementary solutions to supplement the different sub-solutions in turn, and then update the Gaussian probability distribution function of this partial solution according to the gradient update strategy. Of course, there are other ways to complete the solution, or methods that do not require the completion of sub-solutions, and I will not describe them one by one here.[18-20]

Algorithm 2 shows NCNES under the traditional CC framework

Algorithm 2 Classic cooperative coevolution

```
 1: Initialize $\lambda$ solutions $x_i$ randomly, $i = 1, \ldots, \lambda$.
 2: while stop criteria are not met do
 3:     Divide the $D$-dimensional problem into $M$ sub-problems.
 4:     for $j = 1$ to $M$ do
 5:         Evolve all the partial solutions $x_{i,j}$ for one iteration with an EA.
 6:         for $i = 1$ to $\lambda$ do
 7:             Complement each partial solution $x_{i,j}$ with the same complementary vector $v_j$ as
                $[x_{i,j}; v_j]$.
 8:             Evaluate each complemented solution $f[x_{i,j}; v_j]$ as the quality of $x_{i,j}$.
 9:         end for
10:     end for
11:     Output the best complemented solution ever found with respect to $f$.
12: end while
```

3.2 CC framework specific for NCNES

As mentioned above, if the traditional CC framework is used in the NCNES's situation and the same supplementary dimension solution is used for the sub-solutions, the difference in evaluation quality between all partial solutions will become very small, which defeats the original intention of negative correlation search. But our original goal is to keep different solutions as far away from each other as possible, which means that they will have great differences during the search process and may converge to multiple different optimal solutions. Therefore, using the same supplementary vectors in NCNES will reduce its parallel exploration capability and lead to missing multiple optimal solutions. To solve this problem, it is also possible to consider using multiple different supplementary vectors to supplement the partial solution, but this will increase the huge computational cost and thus reduce the value of this approach. Therefore, I proposed a CC framework for NCNES, a new co-evolution method, CC-NCNES, which aims to preserve the parallel exploration performance of negative correlation search (NCNES) as much as possible. The difference from traditional CC is that in CC-NCNES, the partial solution of each sub-problem is supplemented by a different supplementary vector, instead of using the same supplementary vector like traditional CC. In terms of specific implementation, each partial solution will do the following: After removing the values at the same position of the Gaussian distribution mean of the parent of this population, they add it to itself, turning it into a standard-dimensional solution. The mean of the Gaussian distribution of the parent solution represents the current expectation for the best performing solution, so we choose to use the mean to fill in the remainder of the child solution to achieve completeness. Therefore, CC-NCNES actually improves the evolutionary performance under the NCNES framework by sequentially optimizing each sub-problem of each individual. This strategy does not affect the parallel exploration characteristics of different sub-solutions located in different sub-spaces in NCNES, so it naturally avoids the situation in traditional CC that a supplementary vector causes different sub-solutions to become more and more similar.

According to the above CC-NCNES pseudo code, it can be seen that compared to NCNES, the main change is to add a loop for the jth sub-problem. After the sub-problem and the supplementary vector are merged and completed, the NCNES algorithm is performed on it separately. In the process of generating supplementary vector, I use the formula

Algorithm 3 Algorithm 3 CC-NCNES

```
1: Initialize $\lambda$ solutions $x_i$ randomly, $i = 1, \ldots, \lambda$.
2: while stop criteria are not met do
3:     Divide the $D$-dimensional problem into $M$ sub-problems.
4:     for $j = 1$ to $M$ do
5:         for $i = 1$ to $\lambda$ do
6:             Generate an offspring solution $x'_{i,j}$ from the distribution $p_{i,j} \sim N(x_{i,j}, \Sigma_{i,j})$.
7:             Complement $x'_{i,j}$ as $[x'_{i,j}; v_{i,j}]$, where $v_{i,j}$ is the mean of the remaining dimensions
               of the parent
8:             Take $f([x'_{i,j}; v_{i,j}])$ as the new individual and do the NCNES steps to evaluate the
               qualities of $x'_{i,j}$.
9:         end for
10:    end for
11:    Update the distribution function of the dimension corresponding to this round
12: end while
```

$v_{i,j} = x_i / x_i j$, that is to say, the complementary vector for the jth sub-problem is the quotient of x_i for $x_i j$. In this way, the selected supplementary vector is the optimal expectation vector which is currently considered most suitable for the current sub-solution, fully respecting the regional specificity of each solution and retaining the diversity of each solution.

4. Experiments

In order to evaluate and verify the search ability of NCNES, we use CEC2017 benchmarks and CEC2022 benchmarks to test it. Since the main content of this project is to implement NCNES and join the Cooperative Coevolution framework, the main purpose of our design test is to evaluate the performance of NCNES in CEC2017 and CEC2022 respectively, and since CC-NCNES adopts the analysis method based on NCNES, The governance strategy divides each solution individual into multiple smaller sub-solutions, and performs NCNES operations on each sub-solution fragment separately. After the implementation of this strategy, the interdependence of different dimensions in the same solution is theoretically reduced. Therefore, each dimension should be optimized more clearly than before, thereby increasing the convergence efficiency of the algorithm. Therefore, another part of the experimental part is to run the NCNES and CC-NCNES algorithms respectively on different number of dimensions (10, 30), hoping to discover the advantages of CC-NCNES after optimization.

4.1 Comparison on CEC 2017 benchmarks

The CEC 2017 test function set is widely used, it poses certain challenges to algorithms and has high academic recognition. All functions in the test set have been rotated and displaced, which increases the difficulty of algorithm optimization. The function mainly contains 29 test functions, and the specific function description, search range and optimal value are listed as follows:

Table 2 CEC 2017 Benchmark

Serial Number	Function Description	Search Range	Optimum
$F1$	Shifted and Rotated Bent Cigar Function.	[-100, 100]	100
$F3$	Shifted and Rotated Zakharov Function.	[-100, 100]	300
$F4$	Shifted and Rotated Rosenbrock's Function.	[-100, 100]	400
$F5$	Shifted and Rotated Rastrigin's Function.	[-100, 100]	500
$F6$	Shifted and Rotated Expanded Scaffer's F6 Function.	[-100, 100]	600
$F7$	Shifted and Rotated Lunacek Bi-Rastrigin's Function.	[-100, 100]	700
$F8$	Shifted and Rotated Non-Continuous Rastrigin's Function.	[-100, 100]	800
$F9$	Shifted and Rotated Levy Function.	[-100, 100]	900
$F10$	Shifted and Rotated Schwefel's Function.	[-100, 100]	1000
$F11$	Hybrid Function 1.	[-100, 100]	1100
$F12$	Hybrid Function 2.	[-100, 100]	1200
$F13$	Hybrid Function 3.	[-100, 100]	1300
$F14$	Hybrid Function 4.	[-100, 100]	1400
$F15$	Hybrid Function 5.	[-100, 100]	1500
$F16$	Hybrid Function 6.	[-100, 100]	1600
$F17$	Hybrid Function 7.	[-100, 100]	1700
$F18$	Hybrid Function 8.	[-100, 100]	1800
$F19$	Hybrid Function 9.	[-100, 100]	1900
$F20$	Hybrid Function 10.	[-100, 100]	2000
$F21$	Composition Function 1.	[-100, 100]	2100
$F22$	Composition Function 2.	[-100, 100]	2200
$F23$	Composition Function 3.	[-100, 100]	2300
$F24$	Composition Function 4.	[-100, 100]	2400
$F25$	Composition Function 5.	[-100, 100]	2500
$F26$	Composition Function 6.	[-100, 100]	2600
$F27$	Composition Function 7.	[-100, 100]	2700
$F28$	Composition Function 8.	[-100, 100]	2800
$F29$	Composition Function 9.	[-100, 100]	2900
$F30$	Composition Function 10.	[-100, 100]	3000

Among them, F1 and F3 are uni-modal functions with no local minimum and only global minimum. These uni-modal functions can test the convergence ability of the algorithm; F4-

F10 are multi-modal functions with local extreme points. These multi-modal functions are used to test the algorithm's jump out. The ability of local optimization; F11-F20 is a hybrid function that contains three or more CEC 2017 benchmark functions after rotation/displacement, and each sub-function is given a certain weight; F21-F30 is a hybrid function that consists of at least three hybrid functions or CEC 2017 benchmark functions The composite function formed after function rotation shift, each sub-function has an additional bias value and a weight. These combined functions further increase the optimization difficulty of the algorithm. (The F2 function of the original function set was officially deleted because F2 was unstable)

Table 3 NCNES on CEC 2017 Benchmark D = 10

Serial Number	Best	Worst	Mean	variance
$F1$	5.8147E+02	3.0335E+04	8.1647E+03	4.1988E+03
$F2$	**2.0000E+02**	3.8725E+04	**2.0076E+02**	2.2938E+02
$F3$	**3.0848E+02**	3.0922E+02	**3.0862E+02**	5.5929E-02
$F4$	**4.0257E+02**	4.0886E+02	4.0851E+02	5.5075E-00
$F5$	**5.0000E+02**	**5.0000E+02**	**5.0000E+02**	**4.9958E-07**
$F6$	**6.2122E+02**	6.4587E+02	6.2750E+02	3.7227E+00
$F7$	**7.1681E+02**	7.2886E+02	7.2041E+02	4.3002E+00
$F8$	**8.0000E+02**	**8.0000E+02**	**8.0000E+02**	2.5532E-09
$F9$	9.4236E+02	1.9038E+03	1.0064E+03	7.7422E+00
$F10$	**1.0013E+03**	**1.0010E+03**	**1.0072E+03**	**1.2067E+01**
$F11$	4.9412E+04	6.3254E+04	5.2397E+04	1.1421E+03
$F12$	1.4339E+03	9.9384E+03	3.1044E+04	1.4634E+03
$F13$	**1.3309E+03**	7.3298E+03	4.7792E+03	4.7515E+03
$F14$	2.4715E+03	8.2932E+04	7.7572E+04	4.9593E+04
$F15$	**1.5010E+03**	1.5293E+03	**1.5174E+03**	2.5552E+01
$F16$	**1.6047E+03**	**1.6021E+03**	**1.16125E+03**	**4.6921E+00**
$F18$	**1.8204E+03**	1.8657E+04	8.7459E+03	6.5805E+03
$F19$	**1.9376E+03**	1.9855E+03	1.9489E03	8.4205E+00
$F21$	**2.1000E+03**	**2.1000E+03**	**2.1000E+03**	**4.3271E-03**
$F22$	**2.2000E+03**	**2.2000E+03**	**2.2000E+03**	**2.6050E-05**
$F23$	**2.3000E+03**	**2.3000E+03**	**2.3000E+03**	**2.0114E-03**
$F24$	2.8784E+03	2.8801E+03	2.8798E+03	8.1228E-01
$F25$	**2.5000E+03**	**2.5000E+03**	**2.5000E+03**	**2.5767E-03**
$F26$	2.9672E+03	2.9673E+03	2.9672E+03	6.4629E-05
$F27$	**2.7000E+03**	**2.7031E+03**	**2.7014E+03**	**2.5546E+00**
$F28$	3.8025E+03	3.9521E+03	3.8715E+03	7.4365E+03

In the testing phase, we set the total number of evaluations to $10000 * D * M$, this is done to ensure that the number of evaluations for a complete individual is the same after dividing the solution into M sub-solutions. Each time we start training and iterate all populations from initialization until the sum of the number of evaluations of all solutions in the population is $10000 * D * M$, it is a search process. For each test benchmark function, we perform the same search process ten times, and calculate the mean and covariance of the results to reflect the performance of the search algorithm.

It can be seen from the data shown in Table 3 that for most functions of CEC 2017, our test results have reached or are close to the optimal value, and in some problems the mean has almost reached the optimal value, and the variance is very Small, which means that NCNES shows very stable performance on these problems. However, on some issues, the optimization of NCNES seems insufficient. After my observation and speculation, this is mainly because the number of iterations of $10000 * D * M$ is too small for the current update speed, and the function value has been used before convergence has occurred. The number of iterations is over. Of course, this phenomenon also shows that the current optimization efficiency of NCNES is limited. In addition, in some functions, the function values trained by NCNES converge early when they do not reach the optimal value. After analysis, I think this is because the diversity index in the later period accounts for a very low proportion, causing the function to lose a certain degree of global optimality. search ability, thus falling into a local optimum. This problem should be solved by adjusting the trade-off of the fitness value and diversity value, and may require a better dynamic proportion coefficient.

4.2 Comparison on CEC 2022 benchmarks

CEC 2022 is an effective method to evaluate algorithm performance and verify its ability to solve complex Ops. It consists of 12 benchmark functions, including uni-modal, multi-modal, hybrid and composite functions, where F1 is a uni-modal function, F2-F5 are multi-modal functions, and F6-F8 are hybrid functions, which can be uni-modal or multi-modal. , F9-F12 are composite and multi-modal functions.

Similarly, by analyzing the data in Table 5, it can be found that first of all, a considerable number of functions have been optimized to the optimal value, but their worst value and

Table 4 CEC 2022 Benchmark

Serial Number	Function Description	Search Range	Optimum
$F1$	Shifted and Rotated Zakharov Function.	[-100, 100]	300
$F2$	Shifted and Rotated Rosenbrock's Function.	[-100, 100]	400
$F3$	Shifted and Rotated Expanded Scaffer's F6 Function.	[-100, 100]	600
$F4$	Shifted and Rotated Non-Continuous Rastrigin's Function.	[-100, 100]	800
$F5$	Shifted and Rotated Levy Function.	[-100, 100]	900
$F6$	Hybrid Function 1.	[-100, 100]	1800
$F7$	Hybrid Function 2.	[-100, 100]	2000
$F8$	Hybrid Function 3.	[-100, 100]	2200
$F9$	Composition Function 1.	[-100, 100]	2300
$F10$	Composition Function 2.	[-100, 100]	2400
$F11$	Composition Function 3.	[-100, 100]	2600
$F12$	Composition Function 4.	[-100, 100]	2700

Table 5 NCNES on CEC 2022 Benchmark D = 10

Serial Number	Best	Worst	Mean	variance
$F1$	3.2195E+03	5.2663E+04	4.3101E+03	8.7391E+02
$F2$	**4.0701E+02**	**4.0723E+02**	**4.0712E+02**	1.3621E-01
$F3$	**6.0000E+02**	**6.0000E+02**	**6.0000E+02**	**1.1504E-12**
$F4$	**8.1554E+02**	**8.2231E+02**	**8.1983E+02**	**2.7654E-00**
$F5$	**9.0000E+02**	**9.0000E+02**	**9.0000E+02**	**6.6536E-09**
$F6$	3.4482E+04	4.2531E+04	3.8994E+04	3.4928E+03
$F7$	**2.0100E+03**	**2.0431E+03**	**2.0242E+03**	**4.8023E+00**
$F8$	**2.2225E+03**	**2.5674E+03**	**2.2422E+03**	1.1012E+02
$F9$	2.6680E+03	2.6690E+03	2.6684E+03	**2.9461E-01**
$F10$	**2.6221E+03**	**2.8453E+03**	**2.6275E+03**	**4.7241E+00**
$F11$	**2.6005E+03**	2.8632E+03	**2.7815E+03**	1.3976E+02
$F12$	2.8664E+03	2.8665E+03	2.8664E+03	**2.1092E-02**

mean value may not have time to reach the optimal value before the end of the iteration, resulting in a large variance. These All show that the performance of NCNES is not very ideal compared with mainstream optimization algorithms. In addition, notice that in $F9$ and $F12$, the function converges early when it does not reach the optimal solution. The small variance indicates that the function falls into the same local optimum in each training, which means that NCNES is In terms of diversity indicators, the proportion of global search needs to be increased, and ϕ, the dynamic function that balances fitness and diversity, needs to be optimized later.

According to Table 6, I ran CC-NCNES on the CEC2017 data set in D=10. It can be seen that on some issues, CC-NCNES performed equally well, even surpassing the original NCNES. This initially proves the effectiveness of CC-NCNES. However, since most of the

Table 6 CC-NCNES on CEC 2017 Benchmark D = 10

Serial Number	Best	Worst	Mean	variance
$F1$	1.426E+09	3.0335E+10	2.1649E+05	3.1626E+08
$F2$	5.582E+03	9.245E+03	7.243E+03	1.586E+03
$F3$	**4.000E+02**	**5.232E+02**	**4.5964E+02**	3.7310E+01
$F4$	2.5896E+03	3.9654E+03	3.4353E+03	6.7390E+02
$F5$	**5.0000E+02**	**5.0000E+02**	**5.0000E+02**	**3.31666E-04**
$F6$	5.2782E+04	8.3225E+04	7.8248E+04	1.7122E+04
$F7$	**7.5400E+02**	**7.8633E+02**	**7.8611E+02**	1.4695E+01
$F8$	**8.0147E+02**	**8.0322E+02**	**8.0237E+02**	6.6291E-01
$F9$	1.8249E+03	2.3246E+03	2.1038E+03	1.4249E+02
$F10$	1.7447E+04	3.2651E+04	2.8876E+04	6.0654E+03
$F11$	2.5308E+07	1.2351E+08	8.9069E+07	5.3906E+07
$F12$	1.2634E+05	4.1412E+06	2.5849E+06	1.8433E+06
$F13$	3.780E+03	1.2361E+04	8.286E+03	2.5186E+03
$F14$	8.4644E+04	5.3356E+05	3.1680E+05	2.2005E+05
$F15$	7.6730E+03	1.8621E+04	1.3234E+04	3.7039E+03
$F16$	3.2229E+03	5.9566E+03	4.4226E+03	7.2271E+02
$F19$	**2.178E+03**	**2.296E+03**	**2.284E+03**	6.5754E+01
$F20$	2.690E+03	2.812E+03	2.768E+03	4.957E+01
$F21$	**2.224E+03**	**2.227E+03**	**2.226E+03**	**1.3364E-00**
$F22$	3.211E+03	3.523E+03	3.412E+03	1.0582E+02
$F23$	4.771E+03	5.002E+03	4.891E+03	9.8412E+01
$F24$	2.972E+03	3.288E+03	3.014E+03	3.7159E+01
$F25$	3.001E+03	3.028E+03	3.012E+03	7.3329E+00
$F26$	3.007E+03	3.020E+03	3.015E+03	4.7669E+00
$F27$	3.0124E+03	3.2108E+03	3.0145E+03	3.8799E+00

problems are not as good as the previous NCNES, the following reasons are analyzed here: the optimization strategy of CC-NCNES may be more suitable for certain types of data sets but not applicable to other types. Differences in features between datasets may lead to differences in the performance of the algorithm. CC-NCNES may introduce more complexity than the original NCNES, such as introducing more hyperparameters or additional computational steps. This may make the algorithm more difficult to tune and prone to falling into local optimal solutions. At the same time, CC-NCNES may be more suitable for solving high-dimensional problems, so below, I use the same CEC2017 data set with D=30, 30 dimensions and then run NCNES and CC-NCNES respectively, hoping to get a more obvious difference.

4.3 High-dimensional solution effects of NCNES and CC-NCNES on CEC 2017 benchmarks

In this section, I use NCNES and CC-NCNES to run on the 30-dimensional, high-dimensional CEC2017Benchmark. The following is a tabular arrangement of the result data.

Table 7 NCNES on CEC 2017 Benchmark D = 30

Serial Number	Best	Worst	Mean	variance
$F1$	5.8152E+09	8.0213E+09	6.6773E+09	7.5142E+08
$F2$	1.1645E+05	1.6321E+05	1.357E+05	7.7314E+03
$F3$	**9.888E+02**	**1.3425E+03**	**1.2298E+03**	1.5408E+02
$F4$	2.533E+03	4.231E+06	3.179E+04	3.0434E+02
$F5$	**5.0000E+02**	**5.0000E+02**	**5.0000E+02**	**3.9213E-03**
$F6$	1.6989E+05	3.1621E+05	2.2461E+05	2.2426E+05
$F7$	1.1075E+03	1.3211E+03	1.153E+03	2.5382E+02
$F8$	**8.1869E+02**	**8.3263E+02**	**8.2184E+02**	**2.4415E-00**
$F9$	7.477E+04	9.2131E+04	8.3454E+04	3.5473E+02
$F10$	2.0865E+04	3.1298E+05	2.4754E+05	3.1935E+04
$F11$	1.0532E+07	3.8231E+08	1.6550E+07	2.6097E+07
$F12$	1.7975E+07	9.3584E+07	6.189E+07	2.224E+07
$F13$	2.919E+07	8.6466E+07	5.20E+07	1.730E+07
$F14$	8.4644E+04	5.3356E+05	3.1680E+05	2.2005E+05
$F15$	1.4921E+04	2.0137E+04	1.7094E+04	1.533E+03
$F16$	3.240E+05	9.3142E+07	3.881E+07	3.8410E+07
$F17$	1.56E+05	1.1209E+06	2.27E+05	1.051E+05
$F18$	1.9196E+05	5.2389E+05	2.501E+05	4.8926E+04
$F19$	3.150E+03	3.6142E+03	3.338E+03	1.06E+02
$F20$	6.143E+03	6.754E+03	6.515E+03	2.6853E+02
$F21$	**2.401E+03**	2.411E+03	2.410E+03	5.737E-00
$F22$	6.849E+03	1.2385E+04	8.069E+03	6.3581E+02
$F23$	5.172E+03	6.0502E+03	5.552E+03	2.3051E+02
$F24$	3.043E+03	3.213E+03	3.145E+03	4.0913E+01
$F25$	3.419E+03	3.421E+03	3.425E+03	3.8312E+00
$F26$	3.196E+03	3.1542E+03	3.219E+03	1.244E+01
$F27$	3.208E+03	2.226E+03	3.214E+03	4.4224E+00
$F28$	1.1107E+08	3.1289E+09	1.9667E+09	8.8536E+08

Compared with the case of D=10, we observe that CC-NCNES shows superiority on more problems on the data set of D=30. With the same number of evaluations, the CC strategy often produces the same or even better solutions. This shows that CC-NCNES has stronger adaptability and optimization capabilities in higher-dimensional problems. One possible explanation is that in higher-dimensional problems, the search space is larger, and traditional optimization algorithms are often susceptible to the curse of dimensionality, re-

Table 8 CC-NCNES on CEC 2017 Benchmark D = 30

Serial Number	Best	Worst	Mean	variance
$F1$	4.639E+10	7.1856E+10	5.169E+10	4.209E+09
$F2$	**6.789E+04**	**9.866E+04**	**8.333E+04**	9.251E+03
$F3$	5.163E+03	1.211E+04	8.944E+03	1.822E+03
$F4$	4.3246E+04	7.5542E+04	5.305E+04	5.189E+03
$F5$	**5.0002E+02**	**5.0011E+02**	**5.0003E+02**	**3.1910E-03**
$F6$	1.7288E+06	1.9881E+06	1.8324E+06	7.7238E+04
$F7$	**1.330E+03**	1.8666E+03	1.417E+03	4.6203E+01
$F8$	**8.3418E+02**	**8.4001+02**	**8.3942E+02**	3.021E+00
$F9$	7.305E+03	7.813E+03	7.766E+03	2.2005E+02
$F10$	7.957E+05	3.1222E+06	2.274E+06	9.310E+05
$F11$	3.4614E+09	5.2376E+08	4.4774E+09	7.2617E+08
$F12$	2.429E+09	7.5662E+06	5.1387E+09	1.471E+06
$F13$	6.9032E+05	1.8642E+06	1.4286E+06	4.49E+05
$F14$	9.8705E+08	1.5233E+09	1.52128E+09	4.9080E+08
$F15$	4.7308E+07	2.0120E+08	1.9192E+08	1.213E+08
$F19$	**2.143E+03**	**2.451E+03**	**2.265E+03**	7.2452E+01
$F20$	**2.699E+03**	**2.919E+03**	**2.810E+03**	7.0991E+01
$F21$	**2.219E+03**	**2.2375E+03**	**2.2260E+03**	2.464E+00
$F22$	3.296E+04	3.842E+03	3.733E+04	3.4289E+03
$F23$	2.437E+04	2.8324E+03	2.703E+03	1.990E+03
$F24$	**5.339E+03**	**6.5231E+03**	**6.312E+03**	6.2237E+02

sulting in performance degradation. CC-NCNES uses a collaborative strategy to decompose a high-dimensional problem into multiple low-dimensional sub-problems and optimize each sub-problem independently. This method of decomposition and collaboration can make more efficient use of computing resources, reduce the impact of the curse of dimensionality, and achieve better optimization results in higher-dimensional problems. In addition, the CC strategy can effectively share information between different sub-problems, and through appropriate adjustments, it can make the optimization process more stable and robust. This collaborative optimization strategy helps achieve better performance on higher-dimensional problems. Overall, CC-NCNES shows better optimization capabilities on higher-dimensional problems, which proves its effectiveness in dealing with the disaster of dimensionality and complex problems and provides benefits for solving practical high-dimensional problems.

Now，in below, I use Function 9, which have the most obvious advantages, as examples to explain the possible reasons why these functions have advantages under the CC strategy. In Function5, since there are different peaks that are far apart, a wider search space is needed

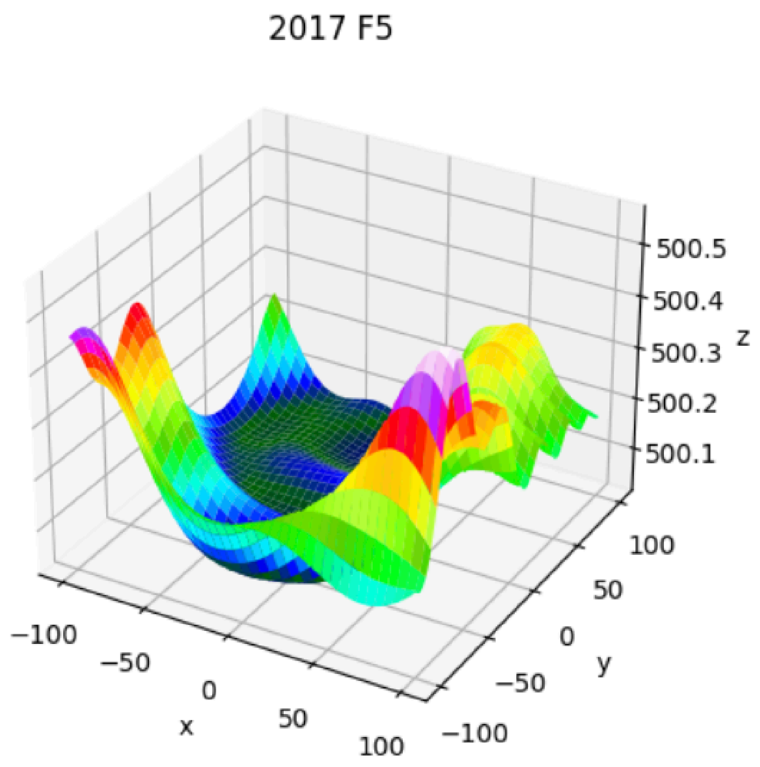


Figure 3 Function 5 in CEC 2017 Test Bench

to cover these peaks. The CC strategy enables extensive exploration at a global scale by decomposing the problem into multiple sub-problems and optimizing each sub-problem independently. This helps to find local optimal solutions located on different mountain peaks. In Function9, there are dense small peaks, and the CC strategy can achieve local utilization by finely adjusting the solution of each sub-problem. By decomposing the search space into multiple subspaces and optimizing for each subspace, the CC strategy can explore local areas more efficiently and find local optimal solutions. The CC strategy can adapt to the complexity of different functions. For multiple distant peaks in Function5, the CC strategy can cope with its complexity by decomposing the problem and searching in parallel. For dense small peaks in Function9, the CC strategy can handle the local details of the function through fine-grained optimization, thereby effectively utilizing search resources. The CC strategy helps maintain solution diversity by applying different search strategies and parameter settings in sub-problems. In Function5 and Function9, this diversity can help the CC strategy better

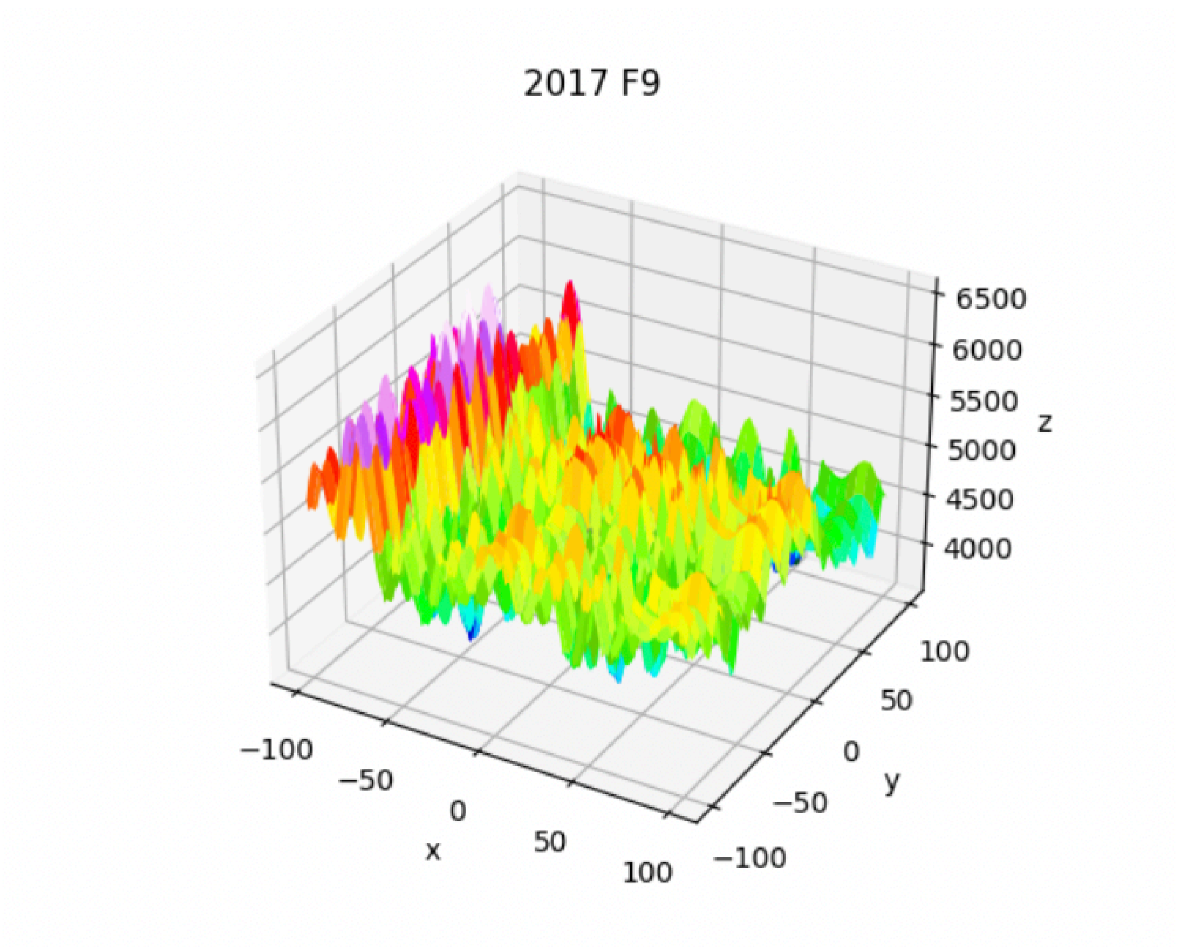


Figure 4 Function 9 in CEC 2017 Test Bench

explore the search space and avoid falling into the local optimal solution. In Function9, due to the presence of dense small peaks, local gradient information may be more important. The CC strategy can more effectively utilize local gradient information by decomposing the problem and performing local optimization on each sub-problem, thereby converging to the local optimal solution faster.

5. Conclusion

This paper summarizes a new mathematical principle called NCNES (Negative Correlation Search Enhanced with Negative Entropy Search), which enables exploration and development by clearly defining and maximizing the diversity model and fitness model of the next generation population. balance. NCNES has a clearer mathematical interpretation than the original NCS and shows how to explore the solution space in parallel and inversely.

In addition, NCNES also successfully solved two technical problems of the original NCS, making the algorithm more robust and reliable.

In order to evaluate the performance of NCNES, a specific instantiation scheme is proposed and called NCNES. NCNES adopted the well-known NES as the search strategy for each subpopulation and was tested on different Benchmarks. The results show that NCNES shows good performance on multiple benchmarks and demonstrates the advantages of the algorithm.

In addition, I also introduced the CC framework into NCNES to further improve the performance of the algorithm in high-dimensional problems. By combining CC with NCNES, the algorithm is better able to adapt to high-dimensional problems and achieve a better balance of exploration and exploitation in a wider search space. This combination makes NCNES show obvious advantages in dealing with high-dimensional problems, thus improving the practicality and scope of application of the algorithm.

参考文献

[1] YANG P, ZHANG H, YU Y, et al. Evolutionary reinforcement learning via cooperative coevolutionary negatively correlated search[J]. Swarm and Evolutionary Computation, 2022, 68: 100974.

[2] TANG K, YANG P, YAO X. Negatively correlated search[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(3): 542-550.

[3] ČREPINŠEK M, LIU S H, MERNIK M. Exploration and exploitation in evolutionary algorithms: A survey[J]. ACM computing surveys (CSUR), 2013, 45(3): 1-33.

[4] QUN N, BING L, KECHENG J, et al. An improved negatively correlated search inspired by Particle Swarm Optimization[C]. in: Journal of Physics: Conference Series: vol. 1267: 1. 2019: 012037.

[5] WANG S, YANG X, CAI Z, et al. An improved firefly algorithm enhanced by negatively correlated search mechanism[C]. in: 2018 IEEE International Conference on Progress in Informatics and Computing (PIC). 2018: 67-72.

[6] CHEN H, PENG Q, LI X, et al. An efficient negative correlation gravitational search algorithm[C]. in: 2018 IEEE International Conference on Progress in Informatics and Computing (PIC). 2018: 73-79.

[7] XU Z, LEI Z, YANG L, et al. Negative correlation learning enhanced search behavior in backtracking search optimization[C]. in: 2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC): vol. 1. 2018: 310-314.

[8] YANG P, TANG K, LOZANO J A, et al. Path planning for single unmanned aerial vehicle by separately evolving waypoints[J]. IEEE Transactions on Robotics, 2015, 31(5): 1130-1146.

[9] LI G, QIAN C, JIANG C, et al. Optimization based Layer-wise Magnitude-based Pruning for DNN Compression.[C]. in: IJCAI: vol. 330. 2018: 2383-2389.

[10] NIU Q, JIANG K, LIU B. A novel binary negatively correlated search for wind farm layout optimization[C]. in: 2019 IEEE Congress on Evolutionary Computation (CEC). 2019: 191-196.

[11] JIAO D, YANG P, FU L, et al. Optimal energy-delay scheduling for energy-harvesting WSNs with interference channel via negatively correlated search[J]. IEEE Internet of Things Journal, 2019, 7(3): 1690-1703.

[12] LIN Y, LIU H, XIE G, et al. Time series forecasting by evolving deep belief network with negative correlation search[C]. in: 2018 Chinese Automation Congress (CAC). 2018: 3839-3843.

[13] WIERSTRA D, SCHAUL T, GLASMACHERS T, et al. Natural evolution strategies[J]. The Journal of Machine Learning Research, 2014, 15(1): 949-980.

[14] YANG P, ZHANG H, YU Y, et al. Evolutionary reinforcement learning via cooperative coevolutionary negatively correlated search[J]. Swarm and Evolutionary Computation, 2022, 68: 100974.

[15] YANG P, YANG Q, TANG K, et al. Parallel exploration via negatively correlated search[J]. Frontiers of Computer Science, 2021, 15: 1-13.

[16] YANG Z, TANG K, YAO X. Large scale evolutionary optimization using cooperative coevolution [J]. Information sciences, 2008, 178(15): 2985-2999.

[17] POPOVICI E, BUCCI A, WIEGAND R P, et al. Coevolutionary Principles.[Z]. 2012.

[18] OMIDVAR M N, LI X, MEI Y, et al. Cooperative co-evolution with differential grouping for large scale optimization[J]. IEEE Transactions on evolutionary computation, 2013, 18(3): 378-393.

[19] YANG P, TANG K, YAO X. A parallel divide-and-conquer-based evolutionary algorithm for large-scale optimization[J]. IEEE Access, 2019, 7: 163105-163118.

[20] YANG P, TANG K, YAO X. Turning high-dimensional optimization into computationally expensive optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 22(1): 143-156.

致谢

在撰写本论文过程中，我要衷心感谢我的导师杨鹏教授。杨老师在我科研的道路上给予了我极大的支持和帮助，不仅开拓了我的学术视野，还指导我探索研究方法，循循善诱地引导我在科研领域取得进步。杨老师对我的关怀和指导，让我深感温暖和鼓舞。

同时，我也要感谢计算机系的其他优秀老师，特别是那些教学序列的老师们。他们严谨负责的教学态度和精湛的教学技能，让我深受启发和感动，时刻提醒我学无止境，不断努力。

在我度过的四年计算机系时光里，我还结识了许多优秀的同学，特别是在创新实践项目中的组员和专业课项目中的队友们。与他们一起度过的快乐与辛苦的时光，让我收获颇丰，让我深刻体会到团队的力量和友谊的可贵。他们的支持和鼓励，是我不断前行的动力和动力。

最后，我要感谢所有在我科研和学习道路上给予我帮助和支持的人，你们的陪伴和鼓励是我前行路上最大的动力。同窗数人，明明稳重，静静开朗. 娜娜温婉，何其有幸，遇见你们，知我年少，予我温暖！