分类号 ____________　　　　　　　　　　编号____________

U D C____________　　　　　　　　　　密级____________

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目：**Reproduction and improvement of parallel exploration strategy based on Negatively Correlated Search**

基于 NCS 的并行探索策略的复现与改进

姓　　名：	陈驿来
学　　号：	**12013025**
系　　别：	计算机科学与工程系
专　　业：	计算机科学与技术
指导教师：	杨鹏 助理教授

2024 年 4 月 11 日

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

基于 NCS 的并行探索策略的复现与改进

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

诚信承诺书

1. 本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。

2. 除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名:

________ 年____ 月____ 日

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

——基于 NCS 的并行探索策略的复现与改进

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. 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 calculation, 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.

Negatively Correlated Natural Evolution Strategy (NCNES) is an evolutionary algorithm (EA) based on Negatively correlated search (NCS) proposed in the paper [1], 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. 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 [2] 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.

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.

表 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.

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

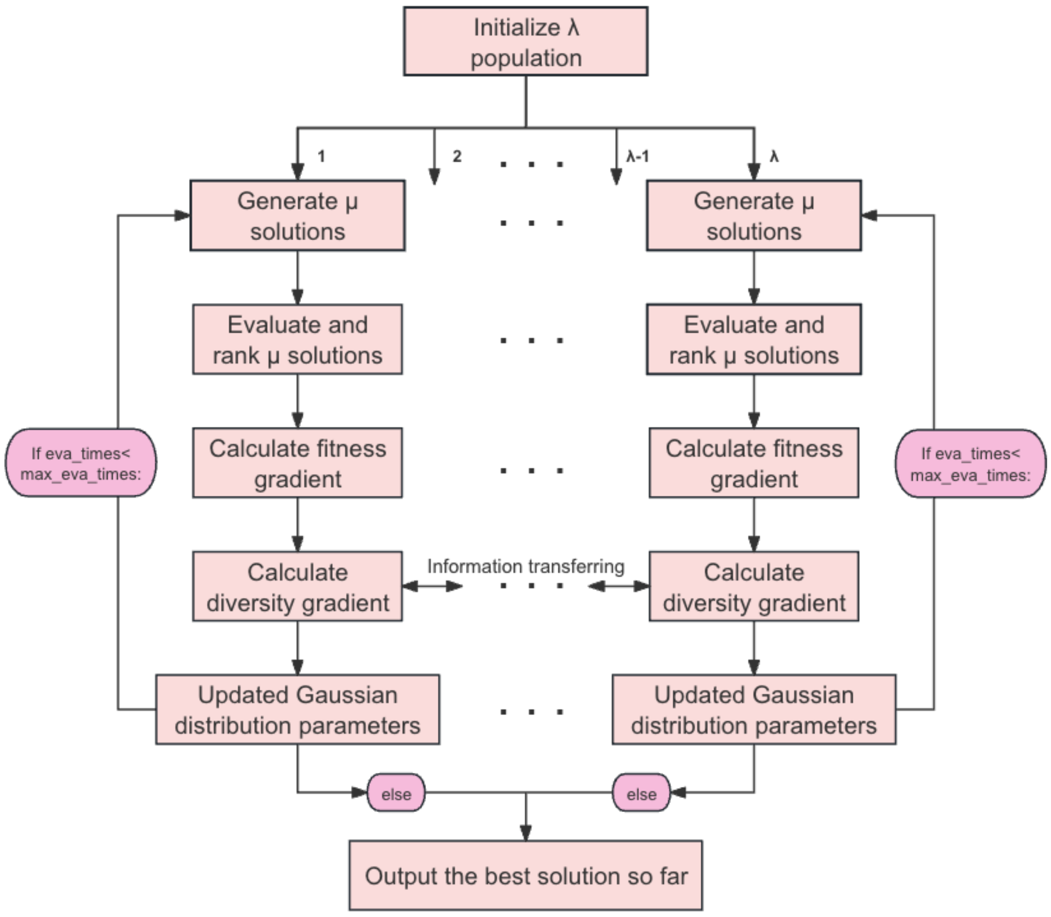
Initialize λ population
1
2
λ-1
λ
Generate μ solutions
Evaluate and rank μ solutions
Calculate fitness gradient
Calculate diversity gradient
Information transferring
Updated Gaussian distribution parameters
If eva_times< max_eva_times:
else
Output the best solution so far

图 1 Theoretical algorithm flow chart

distribution and brought into the formulas inferred above, respectively, we can get:

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

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

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

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

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

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.

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

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.

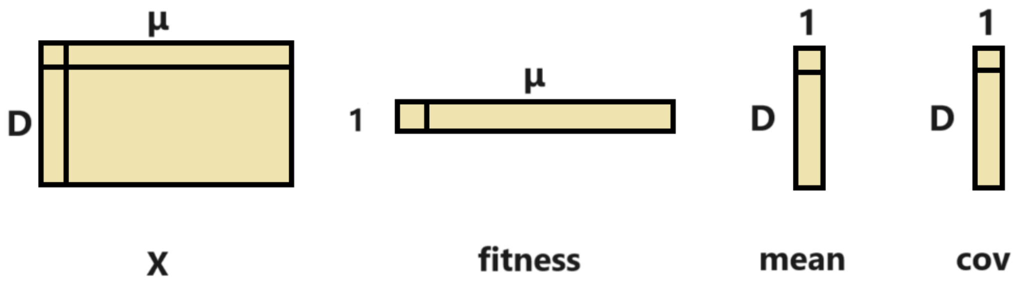

图 2 data structure of main variables used in the NCNES, where X refers to a population, fitness, mean, and cov represent the fitness values, mean and covariance of all solutions within the population.

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 $T_{\text{cur}} < T_{\max}$ do
6:     $\eta_m \leftarrow \eta_m^{\text{init}} \cdot \frac{e - e^{\frac{T_{\text{cur}}}{T_{\max}}}}{e-1}$;
7:     $\eta_\Sigma \leftarrow \eta_\Sigma^{\text{init}} \cdot \frac{e - e^{\frac{T_{\text{cur}}}{T_{\max}}}}{e-1}$;
8:     $\varphi \leftarrow \varphi\text{init} \cdot e^{-\frac{e^{T_{\text{cur}}}}{T_{\max}} \cdot e^{-1}}$
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
```

3. Experiments

In order to evaluate and verify the search ability of NCNES, we use CEC2017 benchmarks and CEC2022 benchmarks to test it.

3.1 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:

表 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)

表 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 $10000 * D$, and 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 $10000 * D$, 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 $10000 * D$ 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.

3.2 Comparison on CEC 2022 benchmarks

表 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

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 mean value may not have time to reach the optimal value before the end of the iteration,

表 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**

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.

4. Conclusion

This paper summarizes a newly proposed mathematical principle of NCS (Negative Correlation Search) and called NCNES, which explicitly models and maximizes the diversity model (for exploration) and fitness model (for exploration) of the next generation population. development). Both models can be maximized by performing gradient descent for each search process. Compared with the original NCS, NCNES has a clearer mathematical explanation, showing how to explore the solution space in parallel and negatively correlated, and how to achieve this. In addition, the new NCS successfully solved two technical problems of the original NCS. To evaluate the performance of the new NCS, the authors propose a concrete instantiation called NCNES. NCNES adopts the well-known NES as the search strategy for each sub-population. NCNES performs well on different Benchmarks, demonstrating the advantages of the algorithm.[3][2][4][1][5][6][7]

参考文献

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

[2] TANG K, YANG P, YAO X. Negatively Correlated Cooperative Search[J/OL]. CoRR, 2015, abs/1504.04914. arXiv: 1504.04914. http://arxiv.org/abs/1504.04914.

[3] ČREPINŠEK M, LIU S H, MERNIK M. Exploration and Exploitation in Evolutionary Algorithms: A Survey[J]. ACM Computing Surveys, 2013, 45: Article 35. DOI: 10.1145/2480741.2480752.

[4] 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. 01. 2018: 310-314. DOI: 10.1109/IHMSC.2018.00078.

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

[6] KAILATH T. The Divergence and Bhattacharyya Distance Measures in Signal Selection[J]. IEEE Transactions on Communication Technology, 1967, 15(1): 52-60. DOI: 10.1109/TCOM.1967.1089532.

[7] LEI Y, YANG P, TANG K, et al. Optimal Stochastic and Online Learning with Individual Iterates [C/OL]. in: WALLACH H, LAROCHELLE H, BEYGELZIMER A, et al. Advances in Neural Information Processing Systems: vol. 32. Curran Associates, Inc., 2019. https://proceedings.neurips.cc/paper_files/paper/2019/file/332647f433a1c10fa2e2ae04abfdf83e-Paper.pdf.