Front. Comput. Sci., 2021, 15(5): 155333
https://doi.org/10.1007/s11704-020-0431-0

RESEARCH ARTICLE

Parallel exploration via negatively correlated search

Peng YANG, Qi YANG, Ke TANG ((✉)), Xin YAO

Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation,
Department of Computer Science and Engineering,
Southern University of Science and Technology, Shenzhen 518055, China.

Abstract Effective exploration is key to a successful search process. The recently proposed negatively correlated search (NCS) tries to achieve this by coordinated parallel exploration, where a set of search processes are driven to be negatively correlated so that different promising areas of the search space can be visited simultaneously. Despite successful applications of NCS, the negatively correlated search behaviors were mostly devised by intuition, while deeper (e.g., mathematical) understanding is missing. In this paper, a more principled NCS, namely NCNES, is presented, showing that the parallel exploration is equivalent to a process of seeking probabilistic models that both lead to solutions of high quality and are distant from previous obtained probabilistic models. Reinforcement learning, for which exploration is of particular importance, are considered for empirical assessment. The proposed NCNES is applied to directly train a deep convolution network with 1.7 million connection weights for playing Atari games. Empirical results show that the significant advantages of NCNES, especially on games with uncertain and delayed rewards, can be highly owed to the effective parallel exploration ability.

Keywords evolutionary computation, reinforcement learning, exploration

1 Introduction

Negatively correlated search (NCS) [1] is a recently proposed evolutionary algorithm (EA) [2] of iteratively searching for optimal solutions. Driven by that a properly diversified population can be more beneficial to search [3], NCS explicitly asks different subsets of the population to periodically share their probabilistic distributions so that they can cooperatively model and control the diversity of the whole population. As the probabilistic distribution actually determines how the new solutions will be sampled, NCS is featured in explicitly modeling the diversity of the next population at the current iteration. On this basis, NCS is capable of capturing the on-going interactions between successive iterations and effectively controlling the diversity of the next population, distinguishing itself from traditional EAs who only measure the diversity of sampled population [3].

Specifically, NCS explicitly divides the population into multiple sub-populations. The evolution of each sub-population is regarded as a separate search process and is conducted by a traditional EA for exploitation. Meanwhile, the search processes are coordinated to explore different search space by driving their probabilistic distributions to be negatively correlated. As a result, NCS has shown to perform a parallel exploration search behavior that multiple search processes are guided to search different promising areas of the search space simultaneously (see Fig.2 in [1] for illustration). Although the basic idea of NCS has attracted increasing research interests [4–7] and has shown very promising performance in various real-world problems [7–12], the original instantiation of NCS [1] was mostly devised by intuition, lacking the mathematical explanations of why the negatively correlated search processes can lead to a parallel exploration and the guidance of how to optimally obtain the negatively correlated search processes.

In this paper, a mathematically principled NCS framework is proposed to address this issue. The new NCS explicitly regards the exploration and exploitation as two objectives of the general search procedure, and works by mathematically modeling and maximizing both a diversity model (for exploration) and a fitness model (for exploitation) of the next population. The diversity model measures the total negative correlations of the probabilistic distributions between pairwise search processes, and the fitness model describes the total expectation of the solution qualities that can be sampled under the probabilistic distributions. In other words, these two models respectively represent how different and how good the new solutions can be generated. By maximizing the diversity model, the search processes tend to be more negatively correlated as the overlaps among probabilistic distributions are getting smaller. By maximizing the fitness model, the expectation of solution qualities that can be sampled by the search processes is improved.

In practice, by employing the natural evolution strategy [13] to evolve each search process, both the diversity model and the fitness model can be optimally maximized via partially gradient descending with respect to each search process. That is, each search process can independently maximize the negative correlation to the others and the expectation of sampling better solutions. On this basis, by gradient descending the two models at the same time, the resultant Negatively Correlated Natural Evolution Strategy (NCNES) is able to form a parallel exploration

Received August 27, 2020; accepted December 11, 2020

E-mail: tangk3@sustech.edu.cn

search behavior that different search processes will in parallel evolve to distinct yet promising areas of the search space.

To verify the effectiveness of NCNES, the reinforcement learning problem is considered for empirical studies, as it is widely acknowledged that the exploration ability has great impacts on the performance of a reinforcement learner [14]. Three popular Atari games [15] covering shooting and obstacles avoidance tasks are selected as the test instances. To play the Atari games, NCNES is required to directly train a deep convolution network with 1.7 million connection weights for optimizing the policy, which imposes great challenges to NCNES as the search space is both large-scale and highly multimodal. Even worse, the environmental rewards are highly uncertain and heavily delayed, making the training further difficult without the help of traditional back-propagation. Empirical results have successfully shown that, NCNES can achieve significantly more scores than the state-of-the-art algorithms (including both EA-based and gradient-based solutions).Furthermore, due to the parallel exploration search behavior, it has shown that NCNES can facilitate the search more computationally efficiently with parallel computing resources.

The rest of this paper are organized as follows. In Section 2, the new mathematically principled NCS is presented in detail, and the weakness of the original NCS that was designed by intuition is also discussed. An instantiation of the new NCS framework, i.e., NCNES, is described in Section 3. In Section 4, the effectiveness of NCNES is verified on three reinforcement learning problems by playing Atari. The conclusions are given in the Section 5.

2 NCS for coordinated parallel exploration

NCS stems from re-thinking of how does population facilitate the search? Although it has been widely acknowledged that effective information sharing among population is the key to successful cooperative search, an open question remains what information to share and how [16]. By mimicking the cooperation in human, NCS asks the individuals in a population to have different search behaviors, so as to avoid repetitively searching a same region of the search space. Similar idea has also been adopted for ensemble learning [17]. Each search behavior is defined as how the offspring will be sampled based on their parents, and usually can be represented as a probabilistic distribution. The mathematical correlation among distributions is utilized to statistically model the diversity among the population. As a result, by explicitly driving multiple probabilistic distributions to be negatively correlated, NCS suffices to control the diversity of the next population.

By implementing the above idea, it is necessary to instantiate a way for modeling the diversity and balancing it with exploitation. In the original NCS, such steps are mainly motivated by intuition, lacking mathematical explanations for in-depth analysis and shown to be sub-optimal. In this section, we first provide the mathematical model that describes the idea of re-designing NCS for parallel exploration; then the new NCS framework is born from the mathematical model by adopting the Gaussian distribution for generating new solutions. It is clearly seen that the new framework is better motivated from mathematical analysis and its advantages over the original NCS are also discussed in detail.

Table 1 The summarization of the major mathematical denotations

Denotation	Description
λ	The number of sub-populations (search processes) at each iteration.
μ	The number of samplings (solutions) in each sub-population at each iteration.
$p(\boldsymbol{\theta}_i)$	The probabilistic distribution of the ith sub-population with the parameter $\boldsymbol{\theta}_i$.
$f(\boldsymbol{x})$	The fitness value of a sample $\boldsymbol{x}$.
$d(p(\boldsymbol{\theta}_i))$	The diversity value between the ith probabilistic distribution and the others.
∇_{θ_i}	The partial gradient of a function with respect to $\boldsymbol{\theta}_i$.
$\mathcal{J}$	The reformed objective to be optimized in NCS.
$\mathcal{F}$	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.
$(\boldsymbol{m}_i, \boldsymbol{\Sigma}_i)$	The mean vector and covariance matrix of the ith probabilistic distribution.
$\mathbf{F}$	The fisher information matrix.

Before that, the major mathematical denotations of the NCS models are summarized in Table 1 for quick reference.

2.1 The mathematical model of NCS

Basically, the idea of NCS requires the population being exclusively grouped into λ sub-populations, each of which is then evolved as a separate search process by a traditional EA, preferably those who sample solutions from an explicit probabilistic distribution [18]. To re-design NCS, let us start a thought game from what kind of probabilistic distribution can facilitate the search better by covering promising areas of the search space and generating new solutions therein.

It is usually straightforward to build a simple well-defined distribution like Gaussian distribution and Cauchy distribution [18]. Unfortunately, such distribution maybe incapable of capturing the complex problem characteristics like the multimodality [19]. Usually, it is non-trivial to properly set up one complicated distribution. Similar to a Gaussian Mixture Model [20], we can have multiple simple distributions instead of one complicated distribution. Another advantage of constructing multiple distributions is that we can explicitly sample different solutions therefrom for the purpose of finding multiple optima [21]. Then this problem can be turned into how to add new simple distributions to the first simple distribution. Clearly, the new distributions should be able to sample new solutions with high fitness values. Moreover, the new distributions should have fewer overlaps (correlations) with existing ones, so that they can be used to sample different regions of the solution space.

For clarity, let us construct the multi-distribution model from scratch. If we initially have one distribution $p(\boldsymbol{\theta}_i)$[1)], there is no worry of overlap. Thus it is only required to sample solutions with higher enough fitness values. Mathematically, this objec-

[1)] $\boldsymbol{\theta}$ denotes the parameters of the distribution of the ith search process. For simplicity, in this paper, we assume all the distributions are with the same type, e.g., Gaussian distribution, while the parameters of the distribution, e.g., mean and covariance, can be different.

[2)] Without loss of generality, the maximization problem is taken for example in this paper.

tive $\mathcal{J}$ (to be maximized) can be modeled as the expectation of fitness values[2)] of the solutions $\boldsymbol{x}$ sampled from $p(\boldsymbol{\theta}_i)$ [13], shown as Eq. (1).

$$\mathcal{J} = \int f(\boldsymbol{x}) p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_1\right) \mathrm{d}\boldsymbol{x}. \tag{1}$$

If we want to add a new distribution $p(\boldsymbol{\theta}_2)$ to $p(\boldsymbol{\theta}_1)$, it has to minimize the correlation between them, as well as maximizing the expected fitness values of both $p(\boldsymbol{\theta}_1)$ and $p(\boldsymbol{\theta}_2)$. For that purpose, the following Eq.(2) should be maximized.

$$\begin{aligned}\mathcal{J} = &\int f(\boldsymbol{x}) p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_1\right) \mathrm{d}\boldsymbol{x} + \int f(\boldsymbol{x}) p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_2\right) \mathrm{d}\boldsymbol{x} + \\ &\left(-C\left(p\left(\boldsymbol{\theta}_1\right), p\left(\boldsymbol{\theta}_2\right)\right) - C\left(p\left(\boldsymbol{\theta}_2\right), p\left(\boldsymbol{\theta}_1\right)\right)\right),\end{aligned} \tag{2}$$

where $C\left(p\left(\boldsymbol{\theta}_i\right), p\left(\boldsymbol{\theta}_j\right)\right)$ means the correlation between the ith and the jth distributions. Now suppose λ distributions are considered, Eq.(2) can be readily extended to Eq.(3).

$$\begin{aligned}\mathcal{J} = &\sum_{i=1}^{\lambda} \int f(\boldsymbol{x}) p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_i\right) \mathrm{d}\boldsymbol{x} \\ &+ \sum_{i=1}^{\lambda} \sum_{j=1, i \neq j}^{\lambda} \left(-C\left(p\left(\boldsymbol{\theta}_i\right), p\left(\boldsymbol{\theta}_j\right)\right)\right).\end{aligned} \tag{3}$$

By maximizing the first additive term, it says that all the distributions should be able to sample solutions with high fitness values. And by maximizing the second additive term, it means that all the distributions should be mutually negatively correlated, by which the overlaps among λ distributions can be maximized. Given that the distributions reflect how new solutions are generated, the first additive term is able to give an expectation of how good the next population might be, and the second additive term is thus capable of modeling the diversity of the next population. On this basis, the diversity model D for all λ distributions is defined as Eq.(4).

$$\mathcal{D} = \sum_{i=1}^{\lambda} \sum_{j=1, i \neq j}^{\lambda} -C\left(p\left(\boldsymbol{\theta}_i\right), p\left(\boldsymbol{\theta}_j\right)\right) = \sum_{i=1}^{\lambda} d\left(p\left(\boldsymbol{\theta}_i\right)\right), \tag{4}$$

where $d\left(p\left(\boldsymbol{\theta}_i\right)\right) = \sum_{j=1, i \neq j}^{\lambda} -C\left(p\left(\boldsymbol{\theta}_i\right), p\left(\boldsymbol{\theta}_j\right)\right)$ is the derived diversity component for the ith search process. By further denoting the first additive term as $\mathcal{F}$ and its ith component as $f\left(\boldsymbol{\theta}_i\right) = \int f(\boldsymbol{x}) p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_i\right) \mathrm{d}\boldsymbol{x}$, Eq.(3) can be re-written as Eq.(5) for clarity.

$$\mathcal{J} = \mathcal{F} + \mathcal{D} = \sum_{i=1}^{\lambda} f\left(\boldsymbol{\theta}_i\right) + \sum_{i=1}^{\lambda} d\left(p\left(\boldsymbol{\theta}_i\right)\right). \tag{5}$$

Thus, the mathematical explanation of NCS can be expressed as maximizing the general objective $\mathcal{J}$, which turns into the maximization of both the diversity model $\mathcal{D}$ for exploration and the fitness model $\mathcal{F}$ for exploitation. It is highly desired that $\mathcal{J}$ can be maximized in parallel to eliminate the interdependencies among search processes and to enjoy the computational acceleration. Since the distribution of a search process is independent from each other by definition, one way to achieve the parallel maximization of $\mathcal{J}$ is to apply the partial gradient descent to $\mathcal{J}$ with respect to each $\boldsymbol{\theta}_i$. The gradient of Eq.(5) can be calculated as Eq.(6).

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_i} \mathcal{J} &= \nabla_{\boldsymbol{\theta}_i} \mathcal{F} + \nabla_{\boldsymbol{\theta}_i} \mathcal{D} \\ &= \nabla_{\boldsymbol{\theta}_i} f\left(\boldsymbol{\theta}_i\right) + \nabla_{\boldsymbol{\theta}_i} d\left(p\left(\boldsymbol{\theta}_i\right)\right), \quad i = 1, \ldots, \lambda.\end{aligned} \tag{6}$$

Clearly, by applying the gradient descent to $\mathcal{J}$, both the diversity model $\mathcal{D}$ and the fitness model $\mathcal{F}$ of each search process can be independently maximized to enable NCS a parallel exploration search behavior, where each search process is highly likely to evolve to an unexplored promising area of the search space, respectively.

2.2 The new NCS framework

To implement Eq.(6), it is required to know how to calculate $\nabla_{\boldsymbol{\theta}_i} f\left(\boldsymbol{\theta}_i\right)$ and $\nabla_{\boldsymbol{\theta}_i} d\left(p\left(\boldsymbol{\theta}_i\right)\right)$, and how to update $\boldsymbol{\theta}_i$ based on them.

For $\nabla_{\boldsymbol{\theta}_i} f\left(\boldsymbol{\theta}_i\right)$, the work in [13] has derived the following formulation (Eq.(7)) that can be directly employed.

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_i} f\left(\boldsymbol{\theta}_i\right) &= \nabla_{\boldsymbol{\theta}_i} \int f(\boldsymbol{x}) p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_i\right) \mathrm{d}\boldsymbol{x} \\ &= \mathbb{E}_{\boldsymbol{\theta}_i}\left[f(\boldsymbol{x}) \nabla_{\boldsymbol{\theta}_i} \log p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_i\right)\right] \\ &\approx \frac{1}{\mu} \sum_{k=1}^{\mu} f\left(\boldsymbol{x}_i^k\right) \nabla_{\boldsymbol{\theta}_i} \log p\left(\boldsymbol{x}_i^k \mid \boldsymbol{\theta}_i\right),\end{aligned} \tag{7}$$

where $\boldsymbol{x}_i^k$ indicates the kth solution in the ith sub-population and μ is the number of the solutions in the ith sub-population. For more details, please refer to [13].

To calculate $\nabla_{\boldsymbol{\theta}_i} d\left(p\left(\boldsymbol{\theta}_i\right)\right)$ by Eq.(4), a correlation measurement $C\left(p\left(\boldsymbol{\theta}_i\right), p\left(\boldsymbol{\theta}_j\right)\right)$ should be specified for the pair of $p(\boldsymbol{\theta}_1)$ and $p(\boldsymbol{\theta}_2)$. Following the original NCS, let the Bhattacharyya distance [22] be the negative correlation measurement, i.e.,

$$C\left(p\left(\boldsymbol{\theta}_i\right), p\left(\boldsymbol{\theta}_j\right)\right) = -\log\left(\int \sqrt{p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_i\right) p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_j\right)} \mathrm{d}\boldsymbol{x}\right),$$

for continuous distributions and

$$C\left(p\left(\boldsymbol{\theta}_i\right), p\left(\boldsymbol{\theta}_j\right)\right) = -\log\left(\sum_{\boldsymbol{x} \in \mathbf{X}} \sqrt{p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_i\right) p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_j\right)}\right)$$

for discrete distributions, respectively. Then $\nabla_{\boldsymbol{\theta}_i} d\left(p\left(\boldsymbol{\theta}_i\right)\right)$ can be given as Eq.(8).

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_i} d\left(p\left(\boldsymbol{\theta}_i\right)\right) &= \sum_{j=1}^{\lambda} \nabla_{\boldsymbol{\theta}_i} \log\left(\int \sqrt{p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_i\right) p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_j\right)} \mathrm{d}\boldsymbol{x}\right) \\ \nabla_{\boldsymbol{\theta}_i} d\left(p\left(\boldsymbol{\theta}_i\right)\right) &= \sum_{j=1}^{\lambda} \nabla_{\boldsymbol{\theta}_i} \log\left(\sum_{\boldsymbol{x} \in \boldsymbol{X}} \sqrt{p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_i\right) p\left(\boldsymbol{x} \mid \boldsymbol{\theta}_j\right)}\right).\end{aligned} \tag{8}$$

After obtaining $\nabla_{\boldsymbol{\theta}_i} f\left(\boldsymbol{\theta}_i\right)$ and $\nabla_{\boldsymbol{\theta}_i} d\left(p\left(\boldsymbol{\theta}_i\right)\right)$, it is straightforward to obtain $\nabla_{\boldsymbol{\theta}_i} \mathcal{J}$ by Eq.(6). Alternatively, a parameter φ can be introduced to trade-off $\nabla_{\boldsymbol{\theta}_i} f\left(\boldsymbol{\theta}_i\right)$ and $\nabla_{\boldsymbol{\theta}_i} d\left(p(\boldsymbol{\theta}_i)\right)$ for a subtle balance between exploitation and exploration, using Eq.(9).

$$\nabla_{\boldsymbol{\theta}_i} \mathcal{J} = \nabla_{\boldsymbol{\theta}_i} f\left(\boldsymbol{\theta}_i\right) + \varphi \cdot \nabla_{\boldsymbol{\theta}_i} d\left(p\left(\boldsymbol{\theta}_i\right)\right). \tag{9}$$

Similar to standard gradient descent methods [23], the objective function $\mathcal{J}$ can be maximized by optimizing the distribution parameters with Eq.(10).

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_i + \eta \cdot \nabla_{\boldsymbol{\theta}_i} \mathcal{J}, \tag{10}$$

Algorithm 1 The new NCS framework

Input: $f, d, \lambda, \mu, \eta, \varphi$;
1: Initialize λ search processes defined by probabilistic model $p(\boldsymbol{\theta}_i), i = 1, ...\lambda$;
2: **while** stopping-criteria not met: **do**
3: **for** each ith search process **do**;
4: Generate μ solutions according to $p(\boldsymbol{\theta}_i)$;
5: Evaluate the fitness of all μ generated solutions;
6: Update $\mathbf{x}^*$ as the best solution ever found;
7: Calculate the gradient of fitness as $\nabla_{\theta_i} f(\boldsymbol{\theta}_i)$;
8: Calculate the gradient of diversity as $\nabla_{\theta_i} d(p(\boldsymbol{\theta}_i))$;
8: $\nabla_{\theta_i} \mathcal{J} \leftarrow \nabla_{\theta_i} f(\boldsymbol{\theta}_i) + \varphi \cdot \nabla_{\theta_i} d(p(\boldsymbol{\theta}_i))$;
10: $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \eta \cdot \nabla_{\theta_i} \mathcal{J}$;
Output: $\boldsymbol{x}^*, f(\boldsymbol{x}^*)$;

where η is a step-size parameter for the gradient descending.

Based on the discussions above, the new NCS framework is listed in Algorithm 1 and described as follows. At the beginning stage, λ probabilistic distributions are initialized to form a set of parallel search processes. For each iteration, the following steps are executed in parallel: 1) each ith search process first generates μ candidate solutions according to its probabilistic distribution $p(\boldsymbol{\theta}_i)$ at step 4; 2) the fitness values of all μ newly generated solutions are evaluated with respect to the fitness function f at step 5; 3) the gradient of the fitness model locally approximated by the ith sub-population, i.e., $\nabla_{\theta_i} f(\boldsymbol{\theta}_i)$, is calculated according to Eq.(7) at step 7; the gradient of the diversity model with respect to the ith sub-population, i.e., $\nabla_{\theta_i} d(p(\boldsymbol{\theta}_i))$, is calculated according to Eq.(8) at step 8; then the gradient of the general objective function, i.e., $\nabla_{\theta_i} \mathcal{J}$, can be accumulated based on Eq.(9) at step 9; the general objective function $\mathcal{J}$ is thus maximized by using gradient descent method (see Eq.(10)), as shown in step 10. Finally, the best ever-found solution $\boldsymbol{x}^*$ that is iteratively recorded (see step 6) will be output as the result of NCS before its halting (see step 11).

2.3 The advantages of the new NCS

In the original NCS, there is no concept of both diversity model and fitness model. But if we look at the original NCS from this perspective, it can be found that the original NCS did not measure the expectation of qualities of unsampled solutions as the fitness model. Instead, to improve the solution qualities, it heuristically compared the fitness values of two sampled solutions for survival. This means that the original NCS cannot utilize the gradient descent method for maximizing the fitness model. Similarly, the diversity model was also maximized by such heuristic comparisons, leaving two technical issues for the original NCS, except for the unclear mathematical explanations.

To be specific, the original diversity model is basically a decentralized model. That is, the diversity of each search process was modeled individually and maximized separately. Comparatively, the new diversity model can be viewed as a centralized model since all correlations between pairwise search processes are counted together. The original diversity model of the ith search process, denoted as $\bar{d}(p(\boldsymbol{\theta}_i))$, was defined as the minimum of the negative correlation between its distribution $p(\boldsymbol{\theta}_i)$ and the distributions of the other search processes, shown as Eq.(11),

$$\bar{d}(p(\boldsymbol{\theta}_i)) = \min_j \left\{ -C\left(p(\boldsymbol{\theta}_i), p\left(\boldsymbol{\theta}_j\right)\right) \mid j \neq i \right\}, \quad \forall i, j = 1, \ldots, \lambda. \tag{11}$$

To maximize each $\bar{d}(p(\boldsymbol{\theta}_i))$ of the ith search process, the original NCS works by comparing the diversity of the current distribution, i.e., the parent distribution $p(\boldsymbol{\theta}_i)$ estimated from the parent sub-population, and the offspring distribution $p\left(\boldsymbol{\theta}'_i\right)$ estimated from the offspring sub-population, and then selecting the larger one to update the distribution $p(\boldsymbol{\theta}_i)$ for the next iteration. In order to obtain good balance between exploration and exploitation, the fitness values are also considered during the maximization of diversity. Let $\boldsymbol{x}_i$ be the parents in the ith search process, and $\boldsymbol{x}'_i$ be their offspring. Then the heuristic comparison goes as Eq.(12),

$$\begin{cases} \text{discard } \mathbf{x}_i \text{ and } \boldsymbol{\theta}_i, & \text{if } f(\mathbf{x}_i) + \varphi \cdot \bar{d}(p(\boldsymbol{\theta}_i)) < \\ & \quad f\left(\mathbf{x}'_i\right) + \varphi \cdot \bar{d}\left(p\left(\boldsymbol{\theta}'_i\right)\right), \\ \text{discard } \mathbf{x}'_i \text{ and } \boldsymbol{\theta}'_i, & \text{otherwise}, \end{cases} \tag{12}$$

where $\varphi \in (0, +\infty)$ is a trade-off parameter, and $f(\mathbf{x}_i)$ are the fitness values of $\mathbf{x}_i$. For more details of the original NCS, please refer to [1].

It can be clearly seen from Eq. (12) that the maximization of both the diversity and the fitness highly depends on the samplings of the candidate solutions (note that the distribution parameters $\boldsymbol{\theta}$ here are also directly estimated from the sampled solutions). However, existing sampling techniques in EAs are usually randomized and thus may involve significant noise, which may mislead the maximization of both the diversity and the fitness. Another issue is that, the above heuristic comparison suffers from the interdependencies among search processes. Specifically, by substituting Eq.(11) to Eq.(12), it can be seen that the heuristic comparison in the ith search process explicitly requires the parent distribution $p\left(\boldsymbol{\theta}_j\right)$ from all the other jth search processes to decide its own parent sub-population and parent distribution at the next iteration, while the heuristic comparisons in other sub-populations also require doing so. Consequently, the heuristic comparison in one search process will be interdependent from that in the others, since the parent distributions of different search processes have to be decided in sequential. Due to the above-mentioned two issues, the diversity and the fitness of each sub-population may not be maximized in parallel, possibly making the parallel exploration of NCS less effective.

Comparatively, in the new NCS, it is no longer needed to compete the exact values of the fitness and diversity pairwise between the parent and offspring sub-populations for survival, as the gradient descent mathematically provides the optimal direction for maximizing both the fitness models and diversity models. On this basis, the random noise of samplings and the interdependencies among sub-populations introduced by the original heuristic comparisons are avoided. As a result, the proposed new NCS framework has successfully addressed the two technical issues of the original NCS, and brings a much clearer explanation to the idea of NCS.

$$\nabla_{m_i} f(\boldsymbol{\theta}_i) = \frac{1}{\mu}\sum_{k=1}^{\mu} \Sigma_i^{-1}\left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)\cdot f\left(\boldsymbol{x}_i^k\right),$$

$$\nabla_{\Sigma_i} f(\boldsymbol{\theta}_i) = \frac{1}{\mu}\sum_{k=1}^{\mu} \left(\frac{1}{2}\Sigma_i^{-1}\left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)\left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)^{\mathrm{T}}\Sigma_i^{-1} - \frac{1}{2}\Sigma_i^{-1}\right)\cdot f\left(\boldsymbol{x}_i^k\right), \tag{13}$$

$$d(p(\boldsymbol{\theta}_i)) = \sum_{j=1}^{\lambda}\frac{1}{8}\left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)^{\mathrm{T}}\left(\frac{\Sigma_i+\Sigma_j}{2}\right)^{-1}\left(\boldsymbol{m}_i - \boldsymbol{m}_j\right) + \frac{1}{2}\log\left(\frac{\left|\frac{\Sigma_i+\Sigma_j}{2}\right|}{\sqrt{|\Sigma_i|\cdot|\Sigma_j|}}\right), \tag{14}$$

$$\nabla_{\boldsymbol{m}_i} d(p(\boldsymbol{\theta}_i)) = \frac{1}{4}\sum_{j=1}^{\lambda}\left(\frac{\Sigma_i+\Sigma_j}{2}\right)^{-1}\left(\boldsymbol{m}_i - \boldsymbol{m}_j\right),$$

$$\nabla_{\Sigma_i} d(p(\boldsymbol{\theta}_i)) = \frac{1}{4}\sum_{j=1}^{\lambda}\left(\left(\frac{\Sigma_i+\Sigma_j}{2}\right)^{-1} - \frac{1}{4}\left(\frac{\Sigma_i+\Sigma_j}{2}\right)^{-1}\left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)\cdot\left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)^{\mathrm{T}}\left(\frac{\Sigma_i+\Sigma_j}{2}\right)^{-1} - \Sigma_i^{-1}\right), \tag{15}$$

$$\mathbf{F}_{\boldsymbol{m}_i} = \frac{1}{\mu}\sum_{k=1}^{\mu}\Sigma_i^{-1}\left(\mathbf{x}_i^k - \boldsymbol{m}_i\right)\left(\mathbf{x}_i^k - \boldsymbol{m}_i\right)^{\mathrm{T}}\Sigma_i^{-1},$$

$$\mathbf{F}_{\Sigma_i} = \frac{1}{4\mu}\sum_{k=1}^{\mu}(\Sigma_i^{-1}\left(\mathbf{x}_i^k - \boldsymbol{m}_i\right)\left(\mathbf{x}_i^k - \boldsymbol{m}_i\right)^{\mathrm{T}}\Sigma_i^{-1} - \Sigma_i^{-1})(\Sigma_i^{-1}\left(\mathbf{x}_i^k - \boldsymbol{m}_i\right)\left(\mathbf{x}_i^k - \boldsymbol{m}_i\right)^{\mathrm{T}}\Sigma_i^{-1} - \Sigma_i^{-1})^{\mathrm{T}}. \tag{16}$$

3 Negatively correlated natural evolution strategies

To instantiate the new NCS framework, the type of probabilistic distribution $p(\boldsymbol{\theta}_i)$ should be specified. In this paper, the Gaussian distribution is employed, i.e., $p(\boldsymbol{\theta}_i)) = \mathcal{N}(\boldsymbol{m}_i, \Sigma_i)$. The underlying reason is three-folds: 1) the Gaussian distribution is the most commonly used distribution in search [18]; 2) by using the Gaussian distribution, $\nabla_{\boldsymbol{\theta}_i} f(\boldsymbol{\theta}_i)$ has an analytic closed form for efficient computation [13]; 3) the Bhattacharyya distance is also analytic based on the Gaussian distribution [1].

By using the Gaussian distribution, $\nabla_{\boldsymbol{\theta}_i} f(\boldsymbol{\theta}_i)$ can be further represented by $\nabla_{\boldsymbol{m}_i} f(\boldsymbol{\theta}_i)$ and $\nabla_{\Sigma_i} f(\boldsymbol{\theta}_i)$, as proposed in NES [5]. Similarly, by using the Gaussian distribution, $\nabla_{\boldsymbol{\theta}_i} d(p(\boldsymbol{\theta}_i))$ can be further represented by $\nabla_{\boldsymbol{m}_i} d(p(\boldsymbol{\theta}_i))$ and $\nabla_{\boldsymbol{\Sigma}_i} d(p(\boldsymbol{\theta}_i))$. Given $C(p(\boldsymbol{\theta}_i), p(\boldsymbol{\theta}_j))$ for Gaussian distribution [1], $d(p(\boldsymbol{\theta}_i)$ can be analytically obtained as Eq.(14), $\nabla_{\boldsymbol{m}_i} d(p(\boldsymbol{\theta}_i))$ and $\nabla_{\boldsymbol{\Sigma}_i} d(p(\boldsymbol{\theta}_i))$ can be derived as Eq.(15).

Thus, $\nabla_{\boldsymbol{m}_i}\mathcal{J}$ and $\nabla_{\boldsymbol{\Sigma}_i}\mathcal{J}$ can be readily obtained by substituting Eqs.(13) and (15) into Eq.(9). Nevertheless, [13] notices that if the above $\nabla_{\boldsymbol{m}_i}\mathcal{J}$ and $\nabla_{\boldsymbol{\Sigma}_i}\mathcal{J}$ are used as the gradients for , there is a serious issue for directly updating $\boldsymbol{m}_i$ and Σ_i with respect to Eq.(10). To be specific, it can be observed that $\nabla_{\boldsymbol{m}_i}\mathcal{J} \propto \frac{1}{\Sigma_i}$ and $\nabla_{\Sigma_i}\mathcal{J} \propto \frac{1}{\Sigma_i^2}$, which means that a large Σ_i can make the learning steps of $\boldsymbol{m}_i$ and Σ_i insignificant, while a small Σ_i can result in a significant update of $\boldsymbol{m}_i$ and Σ_i. This can lead to an unstable search and thus become impossible to precisely locate the optimum [13]. To address this issue, [13] derives the Fisher information matrix $\mathbf{F}$ from the natural gradient of a population. Here we extend it to the multi-population cases where each pair of $\mathbf{F}_{\boldsymbol{m}_i}$ and $\mathbf{F}_{\boldsymbol{\Sigma}_i}$ is respectively assigned for a sub-population, shown as Eq.(16).

With the Fisher information matrix, $\boldsymbol{m}_i$ and Σ_i are updated using Eq.(17).

$$\begin{aligned}\boldsymbol{m}_i &= \boldsymbol{m}_i + \eta_m\cdot\mathbf{F}_{\boldsymbol{m}_i}^{-1}\cdot\nabla_{\boldsymbol{m}_i}\mathcal{J},\\ \boldsymbol{\Sigma}_i &= \boldsymbol{\Sigma}_i + \eta_\Sigma\cdot\mathbf{F}_{\Sigma_i}^{-1}\cdot\nabla_{\Sigma_i}\mathcal{J},\end{aligned} \tag{17}$$

where $\eta_{\boldsymbol{m}}$ and η_Σ are step-size parameters for updating $\boldsymbol{m}_i$ and Σ_i, respectively. Intuitively, since $\mathbf{F}_{\boldsymbol{m}_i}^{-1} \propto \Sigma_i^2$ and $\mathbf{F}_{\Sigma_i}^{-1} \propto \Sigma_i^4$, it turns out that $\mathbf{F}_{\boldsymbol{m}_i}^{-1}\cdot\nabla_{\boldsymbol{m}_i}\mathcal{J} \propto \Sigma_i$ and $\mathbf{F}_{\Sigma_i}^{-1}\cdot\nabla_{\Sigma_i}\mathcal{J} \propto \Sigma_i^2$ will no longer oscillate the search.

Notice that, the above equations are computationally intensive. Specially, the inversion of the Fisher matrix subjects to the computational complexity of $O(D^6)$ if the full covariance matrix are considered [13], where D indicates the dimensionality of the search space. To alleviate the computational costs, we simply restrict the covariance matrix and the Fisher matrix for each distribution to be diagonals. This implies that the interdependencies among decision variables are omitted. Although it may make the algorithm less robust to non-separable problems, it suffices to significantly reduce the computational complexity to $O(D)$, as well as to improve the scalability of the algorithm [24].

Another technique adopted from [13] is the normalization of the fitness values. This is motivated by the difficulty of setting a proper trade-off parameter φ for aggregating $\nabla_{\boldsymbol{\theta}_i} f(\boldsymbol{\theta}_i)$ and $\nabla_{\boldsymbol{\theta}_i} d(p(\boldsymbol{\theta}_i)$, as different problems may have quite varied scales of fitness values. For that purpose, the utility function in [13] is employed in this paper to reshape the fitness values in each sub-population. Specifically, for each sub-population, all μ solutions are first ranked based on their fitness values, where $\pi(k)$ indicates the rank of the kth solution. Then the utility function for each ith sub-population, denoted as U_i, is carried out to reshape the fitness of each kth solution according to Eq.(18). After that, the utility of each solution is used by replacing the term of $f(\boldsymbol{x}_i^k)$ in Eq.(13).

$$U_i(\pi(k)) = \frac{\max\left(0, \log\left(\frac{\mu}{2}+1\right) - \log(\pi(k))\right)}{\sum_{j=1}^{\mu}\max\left(0, \log\left(\frac{\mu}{2}+1\right) - \log(k)\right)} - \frac{1}{\mu}. \tag{18}$$

The step-size parameters η_m and η_Σ can be either tuned offline or adjusted during the search. In this paper, the following strategy is used to adjust these two parameters at each iteration.

Algorithm 2 The proposed NCNES

Input: $f, d, \lambda, \mu, \eta_m^{\text{init}}, \eta_\Sigma^{\text{init}}, \varphi, T_{\max}$

1: **for** $i = 1$ to λ **do**
2: Initialize a Gaussian distribution for the ith Search Process as $\mathcal{N}(\boldsymbol{m}_i, \Sigma_i)$;
3: $T_{cur} = 0$;
4: **while** $T_{cur} < T_{max}$ **do**
5: $\eta_m \leftarrow \eta_m^{init} \cdot \frac{e - e^{\frac{T_{cur}}{T_{max}}}}{e-1}$;
6: $\eta_\Sigma \leftarrow \eta_\Sigma^{init} \cdot \frac{e - e^{\frac{T_{cur}}{T_{max}}}}{e-1}$;
7: **for** $i = 1$ to λ **do**
8: Generate μ solutions $\boldsymbol{x}_i^k \leftarrow \mathcal{N}(\boldsymbol{m}_i, \Sigma_i), \forall k = 1, ..., \mu$;
9: Evaluate the fitness $f\left(\boldsymbol{x}_i^k\right), \forall k = 1, ..., \mu$;
10: $T_{max} \leftarrow T_{cur} + \mu$;
11: Update $\boldsymbol{x}^*$ as the best solution ever found;
12: Rank the kth solution in terms of its fitness $f(\boldsymbol{x}^k)$ as $\pi(k)$, $\forall k = 1, \ldots, \mu$;
13: Set $U_i(\pi(k)) = \frac{\max\left(0, \log\left(\frac{\mu}{2}+1\right) - \log(\pi(k))\right)}{\sum_{j=1}^{\mu} \max\left(0, \log\left(\frac{\mu}{2}+1\right) - \log(k)\right)} - \frac{1}{\mu}, \forall k = 1, ..., \mu$;
14: $\nabla_{\boldsymbol{m}_i} f \leftarrow \frac{1}{\mu} \sum_{k=1}^{\mu} \Sigma_i^{-1} \left(\boldsymbol{m}_i^k - \boldsymbol{m}_i\right) \cdot U_i(\pi(k))$
15: $\nabla_{\Sigma_i} f \leftarrow \frac{1}{2\mu} \sum_{k=1}^{\mu} \left(\Sigma_i^{-1} \left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)\left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)^{\mathrm{T}} \Sigma_i^{-1} - \Sigma_i^{-1}\right) \cdot U_i(\pi(k))$
16: $\nabla_{\boldsymbol{m}_i} d \leftarrow \frac{1}{4} \sum_{j=1}^{\lambda} \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} \left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)$
17: $\nabla_{\boldsymbol{m}_i} d \leftarrow \frac{1}{4} \sum_{j=1}^{\lambda} \left(\left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} - \frac{1}{4}\left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} \left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)\left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)^{\mathrm{T}} \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} - \Sigma_i^{-1}\right)$
18: $\mathbf{F}_{\boldsymbol{m}_i} \leftarrow \frac{1}{\mu} \sum_{k=1}^{\mu} \Sigma_i^{-1} \left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)\left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)^{\mathrm{T}} \Sigma_i^{-1}$
19: $\mathbf{F}_{\Sigma_i} \leftarrow \frac{1}{4\mu} \sum_{k=1}^{\mu} (\Sigma_i^{-1} \left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)\left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)^{\mathrm{T}} \Sigma_i^{-1} - \Sigma_i^{-1})(\Sigma_i^{-1} \left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)\left(\boldsymbol{x}_i^k - \boldsymbol{m}_i\right)^{\mathrm{T}} \Sigma_i^{-1} - \Sigma_i^{-1})^{\mathrm{T}}$
20: $\boldsymbol{m}_i \leftarrow \boldsymbol{m}_i + \eta_m \cdot \mathbf{F}_{\boldsymbol{m}_i}^{-1} \left(\nabla_{\boldsymbol{m}_i} f + \varphi \cdot \nabla_{\boldsymbol{m}_i} d\right)$
21: $\boldsymbol{\Sigma}_i \leftarrow \boldsymbol{\Sigma}_i + \eta_\Sigma \cdot \mathbf{F}_{\Sigma_i}^{-1} \left(\nabla_{\Sigma_i} f + \varphi \cdot \nabla_{\Sigma_i} d\right)$

Output: $\boldsymbol{x}^*, f(\boldsymbol{x}^*)$

$$\eta_m \leftarrow \eta_m^{init} \cdot \frac{e - e^{\frac{T_{cur}}{T_{max}}}}{e-1},$$
$$\eta_\Sigma \leftarrow \eta_\Sigma^{init} \cdot \frac{e - e^{\frac{T_{cur}}{T_{max}}}}{e-1}, \tag{19}$$

where T_{max} is the total time budget for the whole search and T_{cur} is the consumed budget up to now. e is the natural constant. $\eta_{\boldsymbol{m}}^{init}$ and η_Σ^{init} are the initialized values for both step-size parameters, respectively. With Eq.(19), these two step-sizes will decrease over iterations from the initialized values to zero.

So far, all the details have been presented to instantiate an NCS algorithm. To summarize, the proposed algorithm is a multi-Gaussian distribution based EA; Each distribution drives the evolution of one sub-population with the well-established NES; Multiple Gaussian distributions are driven to be negatively correlated by the proposed diversity model. In this regard, the proposed algorithm can also be regarded as a new variant of NES that has the ability of parallel exploration. Thus, it is named Negatively Correlated Natural Evolution Strategies (NCNES) for intuition. The detailed steps of NCNES is listed in Algorithm 2 for reference.

4 NCNES for reinforcement learning

To verify the effectiveness of the proposed new NCS framework as well as NCNES, the reinforcement learning (RL) problem is adopted as the test problem. The considerations are twofolds:1) as a widely-existed multi-step decision making problem, the solution procedure of RL can be naturally formulated as an optimization problem that is large-scale, multi-modal, and uncertain. Such features make RL a more realistic and non-trivial testing environment than the commonly used standard benchmarks for verifying the exploration ability of EAs [14]. Furthermore, as RL does not necessarily touch the domain-specific knowledge, the empirical results maybe more representative than testing on the concrete real-world problems. 2) EAs have been shown to be promising solutions to RL problems as the population-based nature of EAs not only provides the urgent exploration ability to RL [15], but also provides other merits such as parallel acceleration [25–27], noisy-resistance [28, 29], and compatibility of training non-differentiable policies (e.g., trees [30]). Also notice that the canonical NES has been successfully applied to playing Atari games [15]. On this basis, this paper empirically studies the NCNES-based solution for RL problems by playing Atari games.

For the purpose of performance assessment, the empirical studies will uncover three-fold advantages about how effectively the new NCS framework facilitates the search, how well the proposed new diversity model contributes to NCNES, and how well NCNES behaves on reinforcement learning problems.

4.1 Reinforcement learning

RL learns to make Markov decisions so that the long-term rewards can be maximized. In RL, the policy can be iteratively learnt only by interacting with the environment. At each time step, the agent picks an action according to the policy and the observed state of environment, leading to a transition from origin state to the next state, then receives a reward as the feedback to update the policy. The above steps keep going until terminated.

To maximize the expected cumulative discounted reward in

long term, numerous RL methods have been developed in the last decades. Among them, the model-based methods [31–33] try to first learn an accurate environment model, and then get the most beneficial policy by looking up the environment model. Though these methods are mathematically sound, they can hardly scale-up well to the real-world cases, leading the environment model to be either inaccurate or insufficient. The value function based methods [34, 35] then try to learn the value function which refers to the expectation of the accumulated rewards in the future. In recent years, the policy search based methods that adopt the deep neural networks as the policy model have drawn most research attentions due to their powerful performance [15, 35]. The key problem for this type of methods turns into how to train the parameters of the deep network in the RL settings, which faces three major difficulties. First, the search space of training the deep neural networks is highly large-scale and multi-modal; second, due to the Markov decision process nature of RL, the policy learning process is non-differentiable unless some derivable functions are specially designed (e.g., the critic function in A3C [35]); last, the delayed rewards may involve considerable noise. NES is a suitable method for directly training the continuous weights of the neural policy model due to its derivate-free, robust and parallel features. Empirical studies on a set of Atari games have verified the advantages of NES over several state-of-the-art methods [15]. For more details of RL methods, please refer to [36, 37].

It is widely discussed that the exploration ability is a key for RL [38], which allows an agent to improve its current knowledge about the environment, hopefully maximizing the long-term benefit. Among them, Epsilon-Greedy [39] is a simple yet effective method to balance exploration and exploitation in a random way, where epsilon refers to the probability of choosing the exploration strategy during the training. Gibbs sampling [40] studies to model the importance of the action space and sample new action therefrom, trying to recover new promising areas of the action space so as to improve the exploration. Parameter/action space noise methods [41] inject randomized noise directly into the parameters of the policy, making the policy with more diverse behaviors and thus consistent exploration. Curiosity-driven exploration methods [42] formulate the error of a policy as an external objective of curiosity to enable the exploration, which shows to be effective in the environment with sparse rewards. The novelty search method [43, 44] from the derivate-free optimization community explicitly build a new objective to model the novelty to drive the exploration of the parameter space of the policy, while the original objective is abandoned. Though those methods have improved the exploration of RL to some extent, they did not express the proposed parallel exploration ability, where a set of candidate policies are trained to be different yet effective at the same time.

4.2 NCNES for playing Atari

Atari 2600 is a set of video games that have been popular for over 40 years. Atari games successfully cover different types of difficult tasks, such as obstacle avoidance (e.g., Freeway and Enduro), shooting (Beamrider) and other types. The player can do various actions in each game so as to maximize the cumulative reward solely by interacting with the game environment. Due to these features, Atari games have been widely used as the RL simulation platform for empirical studies.

The flowchart of applying NCNES to play Atari games can be seen in Fig.1 for illustration. Basically, the agent aims to learn the policy by iteratively imposing actions to the Atari environment and getting states and rewards therefrom. The policy is modeled as a deep convolutional network for the purpose of conveniently and effectively processing the high-dimensional raw pixel data that is directly received from the video games. NCNES is applied to optimize the connection weights of the policy network without back-propagation. The network architecture of the agent consists of three convolution layers and two full connection layers (see Table 2), as suggested by [34].

More specifically, each individual solution is represented as a vector of all the connection weights of the policy model. Accordingly, the distributions of NCNES search processes are estimated based on those high-dimensional solutions. The training phase is divided into multiple epochs. At each epoch, the agent starts from the beginning of the game and takes a sequence of actions from the policy model to react to the environment, so as to gain as many as possible scores until game overs. After a game (i.e., an epoch) has been finished, the reward will be returned back to the agent as well as NCNES. Then NCNES takes the reward of each epoch as the fitness value of each

Table 2 The network architecture of the DQN based agent

	Input	Output	Kernel Size	Stride	Filters	Activation
Conv1	4x84x84	32x20x20	8x8	4	32	ReLU
Conv2	32x20x20	64x9x9	4x4	2	64	ReLU
Conv3	64x9x9	64x7x7	3x3	1	64	ReLU
Fc1	64x7x7	512	-	-	-	ReLU
Fc2	512	Actions	-	-	-	-

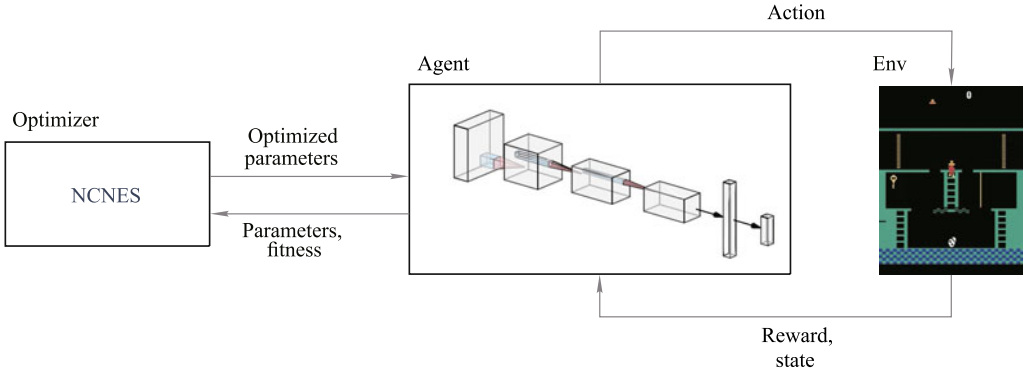

Fig. 1 The flowchart of NCNES based solution for playing Atari

iteration to optimize the connection weights (generating a population of new policy models for the next epoch) in a parallel exploration way, i.e., together with diversity among different search processes. When the training budget runs out, the final policy model will be output for further usages.

From the perspective of optimization, the above problem-solving procedure suffers from three kinds of difficulties.

- First, the search space is extremely large-scale. The deep architecture of the policy results in huge numbers of connection weights to be optimized, where NCNES needs to solve 1.7 million dimensional real-valued optimization problems.
- Second, the search space is highly multi-modal due to the complex architecture of the deep neural networks and the non-uniform distribution of the rewards.
- Third, the feedback is quite uncertain. On one hand, the reward is heavily delayed as the agent can only get the total reward from the environment after the game playing is ended, which makes it very difficult to evaluate the subtle action at each timestep of an epoch. On the other, the total reward involves considerable noise introduced by the randomized Atari games settings, which makes it even harder to evaluate the policy.

Due to the large-scale, uncertain, and multi-modal nature, the optimization problem is non-trivial at all.

4.3 Experiment setup

Three Atari games are selected for the empirical studies, i.e., Freeway, Enduro, and Beamrider. The screenshots of these three games are shown in Fig.2 In freeway, the pedestrian is controlled by three actions (up, down and wait), aiming at avoiding dangerous collisions when goes across a ten-lane highway with large traffic volume, and scores every time it succeeds to reach the other side. The player in Enduro maneuvers a race car to avoid other racers and achieves higher mileage in an endurance race last for several days (counted in the game). The decreased visibility in night or severe weather, and the increased car speed as well as the frequency have posted great challenges. Beamrider is a horizontal scrolling short-range shooter targeted at shooting off destroyable coming enemies with a limited supply of torpedoes and escaping from other undefeatable enemies.

Three RL methods are selected for comparisons, denoted as A3C [35], CES [15] and NCS-C [1], respectively. All those methods are incorporated into the policy search based RL framework for training the same deep neural network as NCNES does, i.e., optimizing the connection weights. Among them, A3C is a state-of-the-art gradient-based method that trains the network with the traditional back-propagation. The other two algorithms are EA-based optimization method. CES is the canonical NES that has been successfully applied to play the Atari games [15]. NCS-C is the instantiation of the original NCS framework. Both the well-established A3C and CES can be used to demonstrate the effectiveness of NCNES on playing Atari games. CES can also be used to assess how parallel exploration can facilitate the search, as NCNES can be simply viewed as a new variant NES with parallel exploration ability. NCS-C is used to show the advantages of the proposed new NCS framework over the original NCS.

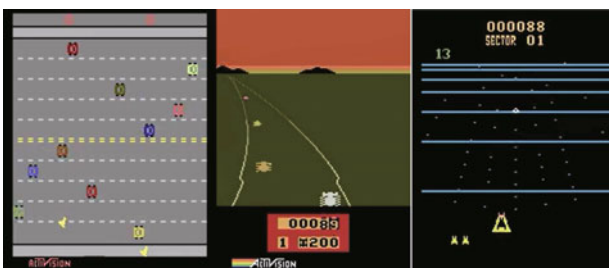

Fig. 2 The screenshots of these three games, i.e., the Freeway, the Enduro and the Beamrider from the left to the right of this figure

For all the comparisons, each algorithm terminates the training phase in a game when the total time budget runs out, and the final solution (policy network) will be returned for testing. The quality of the final solution is measured with the testing score, i.e., averaged score of 30 repeated run in one game-playing without the time limitations. Considering that the environment of a game-playing is randomly initialized, each game-playing will be repeated for three times, i.e., there will be three testing scores for each algorithm on each game. The total time budget is set as the total game frames that each algorithm is allowed to consume for training. For three EA-based methods, the total game frames are set to 100 million. For A3C, as it works quite differently with back-propagation, it is unfair to set the same total game frames with the EA-based ones. In this regard, we counted the game frames consumed by both well-established CES and A3C on the same hardware conditions and in the same game with the same given computational run time. It has been found that the ratio of the consumed game frames between them is about 2.5. As a result, the total game frames are set to 40 million for A3C for fairness. To discretize the games for agents actions execution and states acquiring, the skipping frame is set to 4. That is, for each training phase, the agent is allowed to take 25 million actions for EA-based methods and 10 million actions for gradient-based method.

As both CES and A3C have been successfully applied to play Atari games, we directly borrow the hyperparameters settings from the corresponding papers [15, 35]. The hyperparameters of both NCS-C and NCNES are given as follows. For NCS-C, the number of search processes is set to 8, the sigma is initialized to 0.01, the learning rate of the sigma and the learning epoch are set the same with its original paper [1]. To reduce the noise of the environment, each solution will be re-evaluated for 10 times at each epoch of the training phase, and the averaged score will be returned to NCS-C as the fitness for the solution. For NCNES, the hyperparameters are listed in Table 3 for brevity.

4.4 Results and analysis

Performance analysis on game scoring Three repeated testing scores of each algorithm on three games are shown in Table 4. It can be clearly seen that, NCNES can outperform all the compared algorithms on the tested three games, which successfully verifies the effectiveness of NCNES on reinforcement learning problems. By comparing NCNES with CES, it suffices to show that the parallel exploration can facilitate the search much better as NCNES gains averagely twice scores over CES.

Table 3 The hyperparameter settings of NCNES

Parameter	Value	Remark
λ	5	The number of sub-populations
μ	15	The individuals in each sub-population
φ	0.0001	The trade-off parameter for balancing the exploration and exploitation, set based on the statistically approximated ratio between the scales of fitness gradients and the diversity gradients.
η_m^{init}	0.5	The initial learning rate of mean vectors
η_n^{init}	0.1	The initial learning rate of covariance matrix
t	Randomly pick from [1,2,3,4,5]	The re-evaluation times of each solution to reduce the environmental uncertainty

Table 4 The averaged testing scores of four algorithms on three Atari games

Game		CES	A3C	NCS-C	NCNES
GameFrame		100M	40M	100M	100M
Freeway	Run 1	15.9	0.0	7.0	22.7
	Run 2	12.7	0.0	9.4	21.1
	Run 3	14.1	0.0	3.7	22.1
	Average	14.2	0.0	6.7	**22.0**
Beamrider	Run 1	401.0	908.0	602.0	856.8
	Run 2	508.2	490.2	686.0	620.4
	Run 3	414.1	336.0	482.0	719.3
	Average	441.1	646.7	590.0	**732.2**
Enduro	Run 1	6.2	0.0	6.0	29.8
	Run 2	7.0	0.0	12.8	8.7
	Run 3	8.1	0.0	6.4	11.5
	Average	7.1	0.0	8.4	**17.9**

By comparing NCNES with NCS-C, it can be seen that NCNES gains around three times scores over NCS-C on Freeway, and NCNES also shows significant advantages on other two games. This verifies the effectiveness of the mathematical NCS model. A3C performs less robust than the other three algorithms as its final policy model fails to gain any scores in two games. This maybe because the population-based search can reduce the uncertainty of the algorithms themselves, by 1) frequently sampling from a small region of the search space, which plays the role of re-evaluations to some extent; 2) only requiring the relative order of solutions to determine the search direction, which is less sensitive to the evaluation noise.

Performance analysis on convergence speed To study from the optimization perspective, the score curves of four algorithms on three games are depicted in Fig.3. To depict the curves, at the end of each epoch of the training phase, the current best policy model in terms of the training scores, will be additionally tested for 30 times. And the averaged testing score will be recorded for the purpose of depicting the score curve. Note that, this testing time will not be counted into the total game frames budget, as this score will not be used for helping training. Then the testing score is depicted epoch-by-epoch to form the score curve. Generally, the score curve of an algorithm can express the convergence speed of the optimization algorithm. It can be seen that, NCNES (the red curve) can usually search a very good policy model in very short timesteps. This means that even with a much smaller time budget, NCNES can still outperform the others. For NES and NCS-C, the score curves increase much slower along with the timesteps. This verifies that the new NCS framework can facilitate the

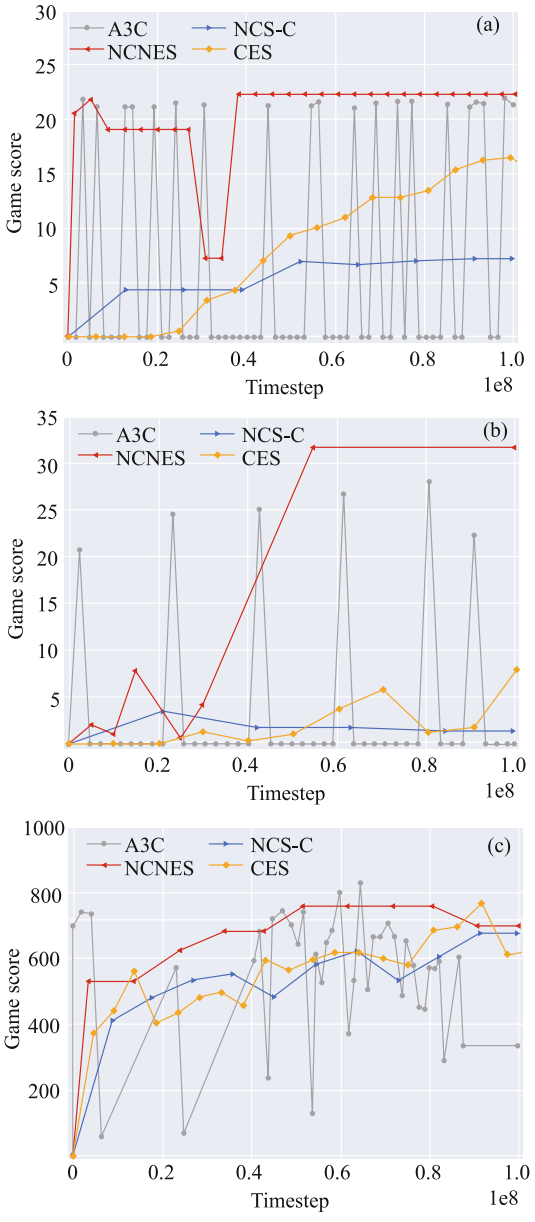

Fig. 3 The score curves of four algorithms on three games, respectively. (a) Freeway; (b) Enduro; (c) BeamRider

search more effectively. Although A3C can occasionally gain high scores, it is very unstable as the score curves oscillate heavily, which even returns very bad policy models (i.e., the averaged score is 0.0 for two games) as the final output. This might be that A3C is less resistant to the environmental noise.

Performance analysis on policy behaviors It is expected that the parallel exploration search behavior of NCNES can help emerge some novel yet useful behaviors that traditional policies are less likely to express. For BeamRider, the agent trained by NCNES prefers staying in the left side of the available area and gains as many as 996 scores in a single testing play (see Fig.4). The motivation behind this trick can be explained as that staying in the left side can prevent at most 50% enemy attacks, and thus is beneficial to longer survival. For Enduro, the agent prefers driving in the middle of the racing track when the weather is good so as to preserve the maximal freedom to move to both sides (see the leftmost figure in Fig.5).

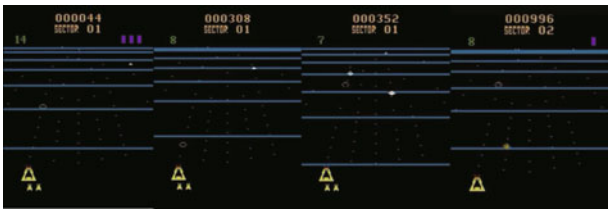

Fig. 4 Tricks learned in BeamRider: the agent prefers to stay in the left side of the available area

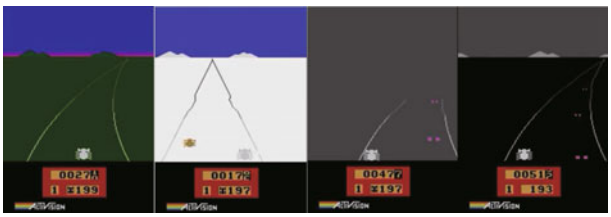

Fig. 5 Tricks learned in Enduro: the agent prefers driving in the middle of the racing track when the weather is good so as to preserve the maximal freedom to move to both sides (see the leftmost figure). When the visibility decreases as it is snowy, foggy, or night (see the other three figures), the agent prefers driving at one side of the racing track for safety

When the visibility decreases as it is snowy, foggy, and night, the agent prefers driving at one side of the racing track for safety, similar to human behaviors (see the other three figures in Fig.5).

Performance analysis on parallel acceleration Then, we show how NCNES can utilize the parallel computing resources. To be specific, three kinds of NCNES are implemented. For the first kind (see Fig.6(a)), NCNES is run on one computing unit in a serial manner. For the second kind (see Fig.6(b)), NCNES is implemented in an island-model based architecture, i.e., five search processes are run on five fixed computing cores respectively during the whole optimization process; At each iteration, information transferring among computing units only happens when the diversity gradients are calculated (also see Algorithm 1, step 10). For the third kind (see Fig.6(c)), NCNES is implemented in a hybrid architecture; Specifically, five search processes are run in an island-model manner with 5 groups of computing units, each group is organized in a master-slave model that consists of 15 computing cores for the fitness evaluations of 15 individuals of a search process, respectively. All three implementations of NCNES are independently run on the same workstation (Intel(R) Xeon(R) CPU E5-2699A v4 @ 2.40GHz) with 44 cores (88 threads)[3)].

The above three implementations of NCNES are simulated on three games with 100M training frames., where three independent runs are conducted for each game. The runtime results are listed in Table 5. It can be seen that, both island-model and hybrid-model can effectively utilize the parallel computing resources for acceleration. That is, by running on a common lab-level workstation, the computing runtime can be largely reduced from around 120 hours (by the serial model) to as short as 2 hours (by the hybrid model). Furthermore, given that the population size of NCNES (including both λ and μ) can be

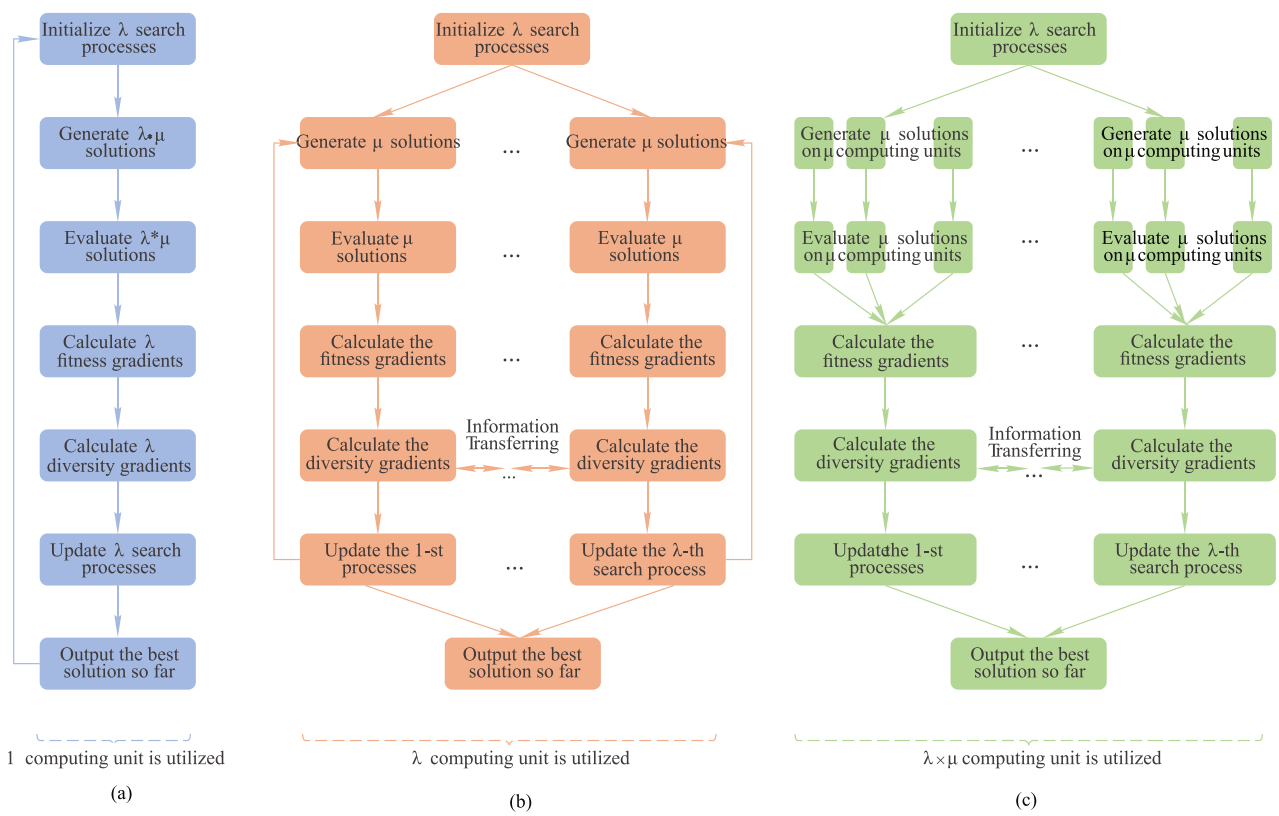

Fig. 6 The flowcharts of three kinds of implementations for NCNES. (a) The serial model of NCNES; (b) the island-model of NCNES; (c) the hybrid-model of NCNES

[3)] The tasks of calculating the diversity gradients (as well as fitness gradients calculation and search processes updating) are always fixed to different physical cores, otherwise the memory sharing mechanism may influence the information transferring efficiency.

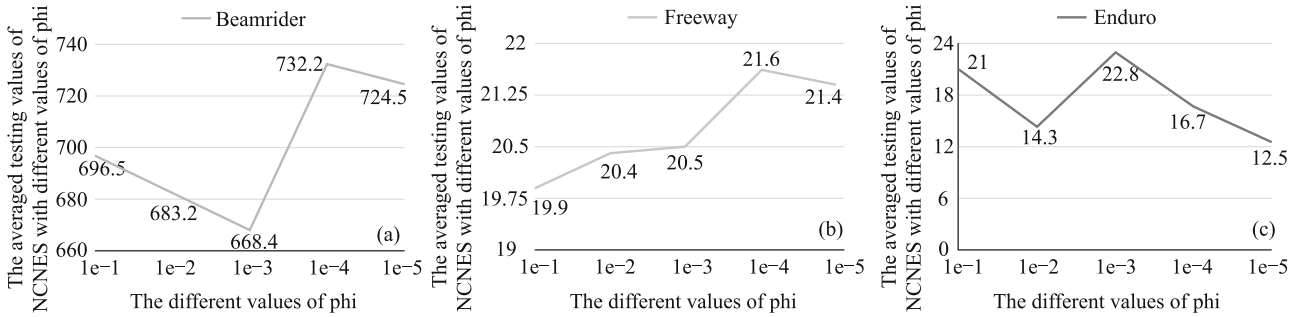

Fig. 7 The sensitivity analysis of φ on three games. (a) Beamrider; (b) Freeway; (c) Enduro

Table 5 The runtime of three implementations of NCNES

Computing model	Game	Serial model	Island-model	Hybrid-model
Computing Units(i.e., m)		1	5	75
Runtime(hours)	Freeway	122.6 ± 0.5	31.2 ± 0.2	2.28 ± 0.0
	BeamRider	116.0 ± 18.8	58.8 ± 22.2	19.48 ± 4.3
	Enduro	119.6 ± 0.7	30.4 ± 0.1	2.16 ± 0.0
Speedup Ratio $\frac{t_{\text{serial}}}{t_{\text{parallel-}m}}$	Freeway	-	0.78 ± 0.01	0.72 ± 0.00
	BeamRider	-	0.43 ± 0.20	0.09 ± 0.03
	Enduro	-	0.79 ± 0.01	0.74 ± 0.02

easily enlarged to enhance the parallel exploration ability, it would be interesting to assess how NCNES can be speedup with large-scale computing resources. This can be measured with the speedup ratio. Theoretically, the speedup ratio[4] $r \in [0, 1]$ says that given m computing units, the parallel implementation can reduce the runtime for $m \times r$ times. The speedup ratios of both island-model and hybrid-model on Freeway and Enduro are very promising, i.e., stably above 0.72.

On the other hand, the speedup ratio on Beamrider is unsatisfactory, i.e., 0.43 for island-model and 0.09 for hybrid model. Actually, this is mostly caused by the blocking synchronization used for transferring the distribution parameters while calculating the diversity gradients. Specifically, at each iteration of NCNES, the solutions are re-evaluated by playing multiple times of the game. As the durations of each game playing can be various for different solutions (e.g., from minutes to hours for Beamrider), each search process may reach the information transferring step at quite different timesteps. Unfortunately, the blocking synchronization would calculate the diversity gradients unless all the distribution parameters are received by each search process. Fortunately, this waiting time can be greatly eliminated by employing the non-blocking asynchronization for approximate information transferring, since the distribution parameters to be transferred has already been obtained at the previous iteration (see Algorithm 1, step 12) and can be transferred at any time afterwards. The price to pay would be the accuracy of the information transferring. To summarize, due to the parallel exploration feature, NCNES is able to be effectively accelerated by parallel computing resources if the computational loads can be well balanced.

Performance analysis on parameter sensitivity Lastly, we present the sensitivity analysis on the most important and featured parameter φ, which trades-off the update of the fitnes model and the diversity model for balancing the exploration and exploitation of NCNES. For this purpose, NCNES is run on three games with five different values of φ, i.e., 1e-1, 1e-2, 1e-3, 1e-4, and 1e-5. The averaged final testing results in depicted in 7. It can be seen that, the performance trends of NCNES with different values of φ vary on different games, e.g., NCNES performs better on Freeway if the value of φ is smaller and NCNES performs better on Enduro if the value of φ is larger. However, such differences are not very significant, i.e., most of them are within 10% of the averaged performance. Thus, though the performance of NCNES depends on the choice of φ, a simple grid search can be enough to help decide a satisfactory φ for NCNES.

5 Conclusions

In this paper, we propose a new mathematically principled NCS framework. The new NCS works by explicitly modeling and maximizing the diversity model (for exploration) and the fitness model (for exploitation) of the next population. Both models can be maximized through gradient descending with respect to each search process. Comparing to the original NCS, the new NCS has clearer mathematical explanations of why the negatively correlated search processes can lead to a parallel exploration search behavior and how to optimally realize it. Besides, the new NCS has also successfully addressed two technical issues of the original NCS. To assess the performance of the new NCS, an instantiation called NCNES is presented. NCNES adopts the well-established NES as the search strategy of each sub-population. NCNES is then applied to solve RL problems for playing Atari games. Specially, NCNES is used to directly train a set of 1.7 million connection weights of the deep policy model under various uncertainties. Empirical studies have shown that, on three typical Atari games, NCNES is able to significantly outperform the state-of-the-arts methods (including both EA-based solution and gradient-based solution). By pairwise comparisons, it also verifies that the proposed new NCS model is better than the original NCS for the purpose of parallel exploration, and the parallel exploration ability can facilitate the search performance as well as the computational efficiency of the new NCS.

[4] The speedup ratio is measured as the ratio of the accumulated runtime on all computing units between the serial implementation and the parallel implementation. Suppose the serial model uses 1 computing unit and its runtime is denoted as , and the parallel model uses computing units and the runtime is denoted as , then the speedup ratio is calculated as . The theoretical speedup ratio varies from 0.0 to 1.0, where indicates the optimal linear speedup. Though some techniques like memory sharing can practically boost over 1.0, they are avoided in this work as the footnote 3 mentioned.

Acknowledgements This work was supported by the Natural Science Foundation of China (Grant Nos. 61806090 and 61672478), Guangdong Provincial Key Laboratory (2020B121201001), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (2017ZT07X386), the Science and Technology Commission of Shanghai Municipality (19511120600), and Shenzhen Science and Technology Program (KQTD2016112514355531).

References

1. Tang K, Yang P, Yao X. Negatively correlated search. IEEE Journal on Selected Areas in Communications, 2016, 34(3): 542–550
2. Back T. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, 1996
3. Črepinšek M, Liu S H, Mernik M. Exploration and exploitation in evolutionary algorithms: a survey. ACM Computing Surveys (CSUR), 2013, 45(3): 1–33
4. Niu Q, Liu B, Jiang K, Wang H. An improved negatively correlated search inspired by Particle Swarm Optimization. Journal of Physics: Conference Series, IOP Publishing, 2019, 1267(1): 12–37
5. Wang S, Yang X, Cai Z, Zou L, Gao S. An improved firefly algorithm enhanced by negatively correlated search mechanism. In: Proceedings of IEEE International Conference on Progress in Informatics and Computing. 2018, 67–72
6. Chen H, Peng Q, Li X, Todo Y, Gao S. An efficient negative correlation gravitational search algorithm. In: Proceedings of IEEE International Conference on Progress in Informatics and Computing (PIC). 2018, 73–79
7. Xu Z, Lei Z, Yang L, Li X, Gao S. Negative correlation learning enhanced search behavior in backtracking search optimization. In: Proceedings of the 10th International Conference on Intelligent Human-machine Systems and Cybernetics. 2018, 310–314
8. Yang P, Tang K, Lozano J A, Cao X. Path planning for single unmanned aerial vehicle by separately evolving waypoints. IEEE Transactions on Robotics, 2015, 31(5): 1130–1146
9. Li G, Qian C, Jiang C, Lu X, Tang K. Optimization based layer-wise magnitude-based pruning for DNN compression. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence. 2018, 2383–2389
10. Niu Q, Jiang K, Liu B. A novel binary negatively correlated search for wind farm layout optimization. In: Proceedings of IEEE Congress on Evolutionary Computation. 2019, 191–196
11. Jiao D, Yang P, Fu L, Ke L, Tang K. Optimal energy-delay scheduling for energy-harvesting WSNs with interference channel via negatively correlated Search. IEEE Internet of Things Journal, 2020, 7(3): 1690–1703
12. Lin Y, Liu H, Xie G, Zhang Y. Time series forecasting by evolving deep belief network with negative correlation search. In: Proceedings of Chinese Automation Congress. 2018, 3839–3843
13. Wierstra D, Schaul T, Glasmachers T, Sun Y, Peters J, Schmidhuber J. Natural evolution strategies. The Journal of Machine Learning Research, 2014, 15(1): 949–980
14. Zhang L, Tang K, Yao X. Explicit planning for efficient exploration in reinforcement learning. Advances in Neural Information Processing Systems, 2019, 32: 7488–7497
15. Chrabaszcz P, Loshchilov I, Hutter F. Back to basics: benchmarking canonical evolution strategies for playing atari. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence. 2018, 1419–1426
16. He J, Yao X. From an individual to a population: an analysis of the first hitting time of population-based evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 2002, 6(5): 495–511
17. Liu Y, Yao X. Ensemble learning via negative correlation. Neural Networks, 1999, 12(10): 1399–1404
18. Yao X, Liu Y, Lin G. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation, 1999, 3(2): 82–102
19. Yang P, Tang K, Lu X. Improving estimation of distribution algorithm on multimodal problems by detecting promising areas. IEEE Transactions on Cybernetics, 2015, 45(8): 1438–1449
20. Reynolds D A. Gaussian mixture models. Encyclopedia of Biometrics, 2009, 741: 659–663
21. Schütze O, Coello C A C, Tantar A A, Tantar E, Bouvry P. EVOLVE—a Bridge Between Probability, Set Oriented Numerics and Evolutionary Computation. Springer Berlin Heidelberg, 2013
22. Kailath T. The divergence and bhattacharyya distance measures in signal selection. IEEE Transactions on Communication Technology, 1967, 15(1): 52–60
23. Lei Y, Yang P, Tang K, Zhou D X. Optimal stochastic and online learning with individual iterates. In: Proceedings of the 33rd Conference on Neural Information Processing Systems. 2019, 5392–5402
24. Yang P, Tang K, Yao X. Turning high-dimensional optimization into computationally expensive optimization. IEEE Transactions on Evolutionary Computation, 2018, 22(1): 143–156
25. Yang P, Tang K, Yao X. A parallel divide-and-conquer-based evolutionary algorithm for large-scale optimization. IEEE Access, 2019, 7: 163105–163118
26. Qian C. Distributed pareto optimization for large-scale noisy subset selection. IEEE Transactions on Evolutionary Computation, 2020, 24(4): 694707
27. Hou N, He F, Zhou Y, Chen Y. An efficient GPU-based parallel tabu search algorithm for hardware/software co-design. Frontiers of Computer Science, 2020, 14(5): 145316
28. Qian C, Yu Y, Tang K, Jin Y, Yao X, Zhou Z H. On the effectiveness of sampling for evolutionary optimization in noisy environments. Parallel Problem Solving from Nature PPSN XIII, 2014, 8672: 302–311
29. Qian C, Yu Y, Zhou Z H. Analyzing evolutionary optimization in noisy environments. Evolutionary Computation, 2018, 26(1): 141
30. Zhou Z H, Feng J. Deep forest: towards an alternative to deep neural networks. In: Proceedings of International Joint Conference on Artificial Intelligence. 2017, 3553–3559
31. Oh J, Singh S, Lee H. Value prediction network. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 6118–6128
32. Ha D, Schmidhuber J. Recurrent world models facilitate policy evolution. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. 2018, 2450–2462
33. Zhong S, Liu Q, Zhang Z, Fu Q. Efficient reinforcement learning in continuous state and action spaces with Dyna and policy approximation. Frontiers of Computer Science, 2019, 13(1): 106126
34. Mnih V, Kavukcuoglu K, Silver D, Rusu A A, Veness J, Bellemare M G, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529–533
35. Mnih V, Badia A P, Mirza M, Graves A, Lillicrap T, Harley T, et al.

Asynchronous methods for deep reinforcement learning. In: Proceedings of International Conference on Machine Learning, 2016, 1928–1937
36. Arulkumaran K, Deisenroth M P, Brundage M, Bharath A A. Deep reinforcement learning: a brief survey. IEEE Signal Processing Magazine, 2017, 34(6): 26–38
37. Qian H, Yu Y. Derivative-free reinforcement learning: a review. Frontiers of Computer Science. 2020, DOI:10.1007/s11704-020-0241-4
38. Tang H, Houthooft R, Foote D, Stooke A, Chen X, Duan Y, Schulman J, DeTurck F, Abbeel P. Exploration: a study of count-based exploration for deep reinforcement learning. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 2753–2762
39. Raykar V, Agrawal P. Sequential crowdsourced labeling as an epsilon-greedy exploration in a markov decision process. In: Proceedings of the 7th International Conference on Artificial Intelligence and Statistics. 2014, 832–840
40. Andrieu C, Freitas N D, Doucet A, Jordan M. An introduction to MCMC for machine learning. Machine Learning, 2003, 50(1–2): 5–43
41. Plappert M, Houthooft R, Dhariwal P, Sidor S, Chen R, Chen X, Asfour T, Abbeel P, Andrychowicz M. Parameter space noise for exploration. In: Proceedings of International Conference on Machine learning. 2018
42. Pathak D, Agrawal P, Efros A A, Darrell T. Curiosity-driven exploration by self-supervised prediction. In: Proceedings of International Conference on Machine Learning. 2017, 2778–2787
43. Lehman J, Stanley K O. Abandoning objectives: evolution through the search for novelty alone. Evolutionary Computation, 2011, 19(2): 189223
44. Conti E, Madhavan V, Such F P, Lehman J, Stanley K, Clune J. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. 2018, 5027–5038

Peng Yang received the BEng degree and PhD degree in computer science and technology from the University of Science and Technology of China (USTC), China in 2012 and 2017, respectively. From 2018, He has been working as a Research Assistant Professor at the Department of Computer Science and Engineering, Southern University of Science and Technology, China. His research interests include large-scale distributed evolutionary computation and its applications. He has been served as a regular reviewer of several world-class journals and a program committee member of a set of top international conferences.

Qi Yang received the BEng degree in Automation from the Huazhong University of Science and Technology, China in 2019. She is currently pursuing Master degree from Southern University of Science and Technology, China. Her research interests include evolutionary computation, reinforcement learning and their applications.

Ke Tang received the BEng degree from the Huazhong University of Science and Technology, China in 2002, and the PhD degree from Nanyang Technological University, Singapore in 2007. From 2007 to 2017, he was with the School of Computer Science and Technology, University of Science and Technology of China, China, first as an Associate Professor from 2007 to 2011 and later as a Professor from 2011 to 2017. He is currently a Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, China. He has over 9100 Google Scholar citations with an H-index of 45. He has published over 70 journal papers and more than 80 conference papers. His current research interests include evolutionary computation, machine learning, and their applications. Prof. Tang was a recipient of the Royal Society Newton Advanced Fellowship in 2015 and the 2018 IEEE Computational Intelligence Society Outstanding Early Career Award. He is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and served as a member on the editorial boards for a few other journals.

Xin Yao is a Chair (Professor) of Computer Science and the Director of the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), University of Birmingham, U.K. He has published 200+ refereed international journal papers. His research interests include evolutionary computation and ensemble learning. He was the President (201442015) of the IEEE Computational Intelligence Society(CIS). He was the Editor-in-Chief (200332008) of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He was the recipient of the 2001 IEEE Donald G. Fink Prize Paper Award, the 2010 and 2015 IEEE Transactions on Evolutionary Computation Outstanding Paper Awards, 2010 BT Gordon Radley Award for Best Author of Innovation (Finalist), the 2011 IEEE Transactions on Neural Networks Outstanding Paper Award, and many other best paper awards. He was also the recipient of the Prestigious Royal Society Wolfson Research Merit Award in 2012 and the IEEE CIS Evolutionary Computation Pioneer Award in 2013.