# Report of NCS and OLMP

**11710716 彭鸿鑫**

## Negative Correlated Search(NCS)

### Main idea:

Main idea of NCS is to maintain a solution population , evaluate the distance between solutions to make the search is negatively correlated.

Use Bhattacharyya distance to evaluate the distance:

$$D_B(p_i,p_j) = -ln(\int \sqrt{p_i(x)p_j(x)}dx)$$

$$D_B(p_i,p_j) = -ln(\sum_{x\epsilon X} \sqrt{p_i(x)p_j(x)})$$

where $p_i$and $p_j$ denote the probability density function of two distributions.

In  N search, to choose one solution to keep ,measure the corresponding distributions:

$$Corr(p_i) = man_j\{D_B(p_i,p_j)|j \neq i\}$$

where $p_j$represent the distributions corresponding to other RLSs.

In i search, $x_i$ to represent the exiting solution, then use Gaussian mutation operator generates a new solution $x_i^{'}$

$$x_{id}^{'} = x_{id} + N(0,\sigma_i)$$

where the $N(0,\sigma_i)$ denotes a Gaussian random variable with zero mean and standard deviation $\sigma_i$.

then each $\sigma_i$ is adapted for every $epoch$ iterations according to the 1/5 successful rule.

$$\sigma_i = \begin{cases} \frac{\sigma_i}{r} & if \frac{c}{epoch} < 0.2 \\ \sigma_i * r & if \frac{c}{epoch} > 0.2 \\ \sigma_i & if \frac{c}{epoch} = 0.2 \end{cases}$$

where r is a parameter that suggested to be set beneath 1, and c is the times that a replacement happens during the past epoch iterations.

Larger D() and larger Corr ()are better and how to balance and choose solutions:

$$\begin{cases} discard\ x_i, & if \frac{f(x)}{Corr(p_i^{'})} < \lambda \\ discard\ x_i^{'}, & otherwise \end{cases}$$

where $\lambda\ \epsilon\ (0,+\infty)$ is a parameter to affect the search process and consequently the performance of NCS. Because the $\lambda$ is associate with the time of iteration.

$$\lambda_t = N(1, 0.1 - 0.1 * \frac{t}{T_{max}})$$

where $T_{max}$ is the user-defined total number of iterations for an execution of NCS-C.

there is the pseudo-code of NCS-C:

**Algorithm 1.** NCS-C ($T_{max}$, $\boldsymbol{\sigma}$, $r$, $epoch$, $N$)

Randomly generate an initial population of $N$ solutions.
Evaluate the $N$ solutions with respect to the objective function $f$.
Identify the best solution $\mathbf{x}^*$ in the initial population and store it in *BestFound*.
Set $t \leftarrow 0$
**While** $(t < T_{max})$ **do**
  Set $\lambda_t \leftarrow \mathcal{N}(1, 0.1 - 0.1 * \frac{t}{T_{max}})$.
  **For** $i = 1$ **to** $N$
    Generate a new solution $\mathbf{x}_i'$ by applying Gaussian mutation operator with $\sigma_i$ to $\mathbf{x}_i$.
    Compute $f(\mathbf{x}_i')$, $Corr(p_i)$ and $Corr(p_i')$.
  **EndFor**
  **For** $i = 1$ **to** $N$
    **If** $f(\mathbf{x}_i') < f(\mathbf{x}^*)$
      Update *BestFound* with $\mathbf{x}_i'$.
    **EndIf**
    **If** $\frac{f(\mathbf{x}_i')}{Corr(p_i')} < \lambda_t$
      Update $\mathbf{x}_i$ with $\mathbf{x}_i'$.
    **EndIf**
  **EndFor**
  $t \leftarrow t + 1$
  **If** mod$(t, epoch) = 0$
    **For** $i = 1$ **to** $N$
      Update $\sigma_i$ for each RLS according to the 1/5 successful rule.
    **EndFor**
  **EndIf**
**EndWhile**
**Output** *BestFound*

## Application of NCS:

In my opinion, the NCS is a method to find a optimal solution through searching in different areas. We can use NCS to improve the speed of getting optimal solution, through different search areas. Changing step size after $epoch$ time of iterations can make the final result not just local optimal solutions.

# OLMP

## Main idea:

Main idea of OLMP is optimization based OLMP, to pruned the DNN network. Reducing the size and parameters of each layer.

**Magnitude-based pruning (MP)**: Given a network W and a threshold $\varepsilon$,Magnitude-based pruning indicates:

$$MP(W,\varepsilon) = \{w||w \geq \varepsilon, w\epsilon\}$$

This method prunes the connections whose absolute connection weights are lower than $\varepsilon$.

**Layer-wise magnitude-based pruning (LMP)**: Instead of apply MP on the whole network, LMP applies one each layer separately:

$$LMP(W,\{\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_L\}) = U_{l=1}^{L} MP(W,l,\varepsilon_l),$$

where $MP(W,l,\varepsilon) = \{W_{i,j}^{l}||W_{i,j}^{l}| \geq \varepsilon, 1 \leq i \leq n^{l}, 1 \leq j \leq n^{n+1}\}$.

**Optimization based LMP (OLMP):**

$$\varepsilon^{*} = argmin\ \ |W^{'}|s.t.\ f(W) - f(W^{'}) \leq \delta\ \ \ (\varepsilon\epsilon R^{L}, W^{'} = LMP(W,\varepsilon)),$$

where $\varepsilon = \{\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_L,\}, W^{'}$ is the model pruned by applying LMP with $\varepsilon$ on W.

find the best $\varepsilon^{*}$ in the function of OLMP then use it to LMP, it can best pruned DNN.

## Application:

OLMP is the optimization based LMP, used to pruned DNN network, reducing parameters and wide sizes of each layer. The DNN networks are complex with huge size. After training, the network must be save in application. OLMP can make the application which used DNN network be smaller size, that is user download faster from network.

# Parameter description:

**Role and effect:**

The role of **lambda** is to affect the search process and consequently the performance of NCS.

To balance the distance and correlation of solutions.

The role of **r** is the parameter to multiply the last $\sigma$ which is generate next solution x through Gaussian random variable with N(0,$\sigma$)

The role of **epoch** is the times of iterations to check weather to change the step size, that is the time of iteration between every changing step sizes.

The role of **N** is the number of solutions in population. Also mean the number of search areas.

### F6:

**lambda:** 1.12135537 **r:** 0.900114194 **epoch:** 13 **N:** 1 **result:** 390.00000001150926

**time**: 51.34 **best range**: r 0.900114-0.900115 r 1.01213-1.01214 epoch 13 N 1

### F12:

**lambda:** 1.006726866 **r:** 0.991659726 **epoch:** 9 **N:** 2 **result:** -459.99999999999943

**time**: 33.12 **best range:** lambda 1.006-1.007 r 0.9916-0.9917 epoch 9 N 2

### OLMP:

**lambda:** 1.012135537 **r:** 0.900114194 **epoch:** 20 **N:** 92 **result:** 0.9889914106747684

**time**: 62.54 **best range**: N 92

## Tuning procedure:

I use the random algorism to change the parameter of NCS, and short the range of parameter through better result.

```
            random algorism

                lambda = range of lambda
                r = rang of r
                epoch = range of epoch
                N = range of epoch
                NCS(lambda,r,epoch,N)
                file write the result in new exel
            select the better parameters of better solutions
            change the ranges of parameters
```

In the OLMP I find the result depend on N, and is not relative with other parameters. When N = 92, the result is best. Other parameters can change the time of OLMP.

**Evolutionary Algorism**

**Initial a population** of parameters as solution. Initial parameters from randomly generating.

**Fitness function** because the best solution we have already got in the rank list, so NCS() must be closed to 390. It depend on the accuracy we want.

**Selection Part** is to find weather the result of the new solution closer to the optimal result.

**Crossover Part** is to get the mean value of each parameters from parents parameters.

**Mutation Part** is to find new parameter if it can not get better result in small area.

## Reference:


[1] K. Tang, P. Yang and X. Yao, "Negatively Correlated Search," in IEEE Journal on Selected Areas in Communications, vol. 34, no. 3, pp. 542-550, March 2016.

2 P. Yang， K. Tang, and X. Yao, " Negatively Correlated Search as a Parallel Exploration Search Strategy "in

eprint arXiv:1910.07151, Oct.2019.

[3] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” IEEE Trans. Evol. Comput., vol. 3, no. 2, pp. 82–102, Jul. 1999.