## 【HARD LEVEL】

### PROBLEM 02 (30%)

In the field of computer science and math optimization, one of the famous solving strategies is “metaheuristic” which is good at find, generate, or select a sufficient good solution to an optimization problem. Compared to optimization algorithms and iterative methods, metaheuristics do not guarantee that a globally optimal solution can be found on some class of problems. Because the process of finding optimal solution is dependent on the set of random variables generated, many new metaheuristic algorithms are inspired by natural systems. Taking simulated annealing algorithm for example, the way finding optimal solution is inspired by the process of temperature-cooling in metallurgy. In addition, another metaheuristic algorithm is originated by the simplified society model and simulating the unpredictable exercise of flock of birds in a multi-dimensional space. In this problem, we are going to implement this algorithm and solve an optimization problem.

In this algorithm, we have the following variables

| Name | Depiction |
| --- | --- |
| Dim | Dimension of the space which also means the variables’ number in optimization problem |
| Num_b | Number of birds |
| $\mathbf{x}_i$ | Represents the position of bird i and contains Dim numbers. ( $\mathbf{x}_i = [x_{i,0}, x_{i,1}, \ldots\ldots, x_{i,(dim-1)}]$ ) |
| Max_boundary | The maximum value of each dimension. (max = [$x\_max_0$, $x\_max_1$, ……, $x\_max_{(dim-1)}$] |
| Min_boundary | The minimum value of each dimension. (min = [$x\_min_0$, $x\_min_1$, ……, $x\_min_{(dim-1)}$] |
| $f(\mathbf{x}_i)$ | A cost function that returns a score for judging how good the solution $\mathbf{x}_i$ is. |
| Max_cycle | Maximum cycles for iteration |
| $\mathbf{v}_i$ | Represents the velocity of bird i and contains Dim numbers. ( $\mathbf{v}_i = [v_{i,0}, v_{i,1}, \ldots\ldots, v_{i,(dim-1)}]$ ) |
| **k** | A coefficient of velocity maximum |
| $\mathbf{v}^{max}$ | Represents the maximum velocity and contains Dim numbers.<br>( $\mathbf{v}^{max} = [v_0^{max}, v_1^{max}, \ldots\ldots, v_{(dim-1)}^{max}]$ ) |
| w(cycle) | A function of velocity iteration coefficient |
| **C1, C2** | Two acceleration-constant |
| **R1, R2** | Two random number arrays in size of (Max_cycle * Num_b * Dim) |
| $\mathbf{p}_i$ | Represents the best position that bird i has ever been. |
| **g** | Represents the best position that this flock of the birds have ever been. |

- Initialization
  - Initial position: $\mathbf{x}_{i,j}$ = x_min$_j$ + (x_max$_j$ - x_min$_j$) * i / (Num_b – 1)
  - initial velocity: $\mathbf{v}_{i,j} = \mathbf{0}$ $\forall \boldsymbol{i}, \boldsymbol{j}$
- Setting
  - **k** = 0.5
  - $v_i^{max}$ = (x_max$_i$ - x_min$_i$) * **k** / 2, $v_i^{max}$ is a double number.
  - Dim, Num_b, Max_cycle, C1, C2, Max_boundary, Min_boundary: define in setting file (*.set)
  - R1: define in rand1 file (*.rand1)
  - R2: define in rand2 file (*.rand2)
- Iteration update (in the following order)
  - w(cycle) = $w_{max}$ – ($w_{max}$ - $w_{min}$) * cycle / Max_cycle

    cycle $\in$ [0, Max_cycle) , $w_{min}$ = 0.85, $w_{max}$ = 1.15
  - **$\mathbf{v}_{i,j}$ (new) = $\mathbf{v}_{i,j}$ (old)*w(cycle)+C1*($\mathbf{p}_{i,j}$ – $\mathbf{x}_{i,j}$)*R1$_{cycle, i, j}$+C2*($\mathbf{g}_j$ – $\mathbf{x}_{i,j}$)*R2$_{cycle, i, j}$**
  - if $\mathbf{v}_{i,j} > v_j^{max}$ , then $\mathbf{v}_{i,j} = v_j^{max}$.

    if $\mathbf{v}_{i,j} < -v_j^{max}$ , then $\mathbf{v}_{i,j} = -v_j^{max}$
  - $\mathbf{x}_i$ (new) = $\mathbf{x}_i$ (old) + $\mathbf{v}_i$
  - if $\mathbf{x}_{i,j} > x_j^{max}$ , then $\mathbf{x}_{i,j} = x_j^{max}$.

    if $\mathbf{x}_{i,j} < x_j^{min}$ , then $\mathbf{x}_{i,j} = x_j^{min}$
  - update $\mathbf{p}_i$ $\in i$ and update **g**
- after iteration in times of Max_cycle
  - **g** is the optimal solution we can get from this metaheuristic process.

The polynomial regression curve is in the format of ($a_o + a_1*x^1 + \ldots\ldots + a_{(Dim-1)}*x^{(Dim-1)}$). We call this polynomial regression curve R(x) and use the loss function to evaluate the R(x). In this problem, we use following loss function, Loss(R) = $\sum | y - R(x) |$ for all given points in the .data file. When the smaller value Loss (R) is, the better regression curve we get.

In this problem, you need to implement a C/C++ program to find the optimal solution of the polynomial regression curve using the metaheuristic mentioned above. The result of **g** in metaheuristic will represent the optimal solution of coefficients in polynomial regression curve ($a_0$, $a_1$, ……, $a_{(Dim-1)}$) as well.

- Input file
  - .set file: give the following value in each column. Dim(int), Num_b(int), Max_cycle(int), C1(double), C2(double), Max_boundary(Dim integers), Min_boundary(Dim integers).
  - .rand1 file: there are Max_cycle sections and a line contains one "=" between two part. Each part has Num_b columns and Dim double numbers in one line.
  - .rand2 file: there are Max_cycle sections and a line contains one "=" between two part. Each part has Num_b columns and Dim double numbers in one line.
  - .data file: In the first line, there is an integer number (N). Furthermore, there are N lines and two numbers in each line. In each line, the first number is an integer number and it means the point's x value and the second number is a double number and it means the point's y value.
- Output
  - .out: Please show the best result of ($a_0$, $a_1$, ……, $a_{(Dim-1)}$) on the terminal and only one double number in one line.
- Execution command
  - **./hard-2 [.set file name] [.date file name] [.rand1 file name] [.rand2 file name]**

- Note: please use integer and double. Otherwise, there may be some error.

**>vim 1.set**

```
Dim 3
Num_b 24
Max_cycle 243
C1 1.9
C2 1.4
Min_boundary -36 -12 -9
Max_boundary 37 20 6
```

**>vim 1.data**

```
22
24 1522.26
-10 483.133
2 -12.0043
3 -8.25165
-23 1743.26
-25 1926.8
-6 195.565
19 891.778
17 643.802
-4 88.8444
-5 121.279
16 604.861
-16 919.117
15 537.636
7 71.4138
-8 270.499
10 214.958
20 999.539
25 1849.89
0 -4.74096
-3 48.8287
23 1458.3
```

**>./hard-2 1.set 1.data 1.rand1 1.rand2**

```
-2.67489        //a0
-8.48053        //a1
2.9424          //a2
```