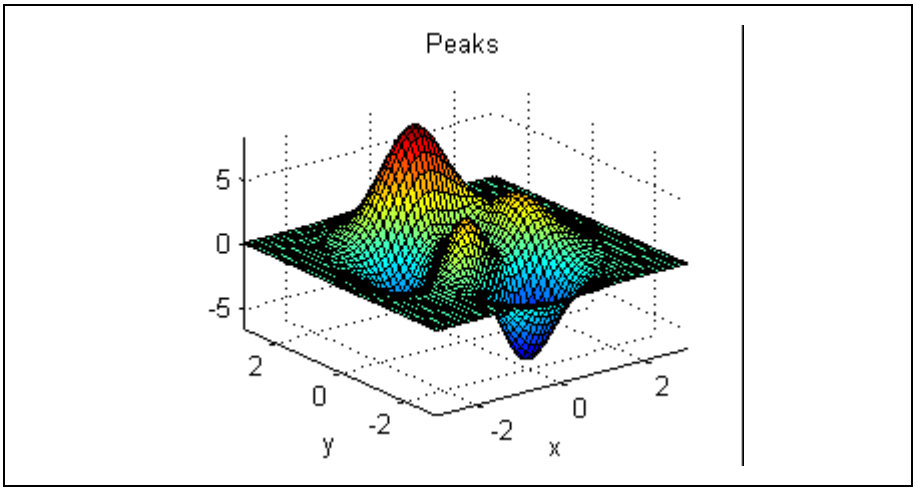
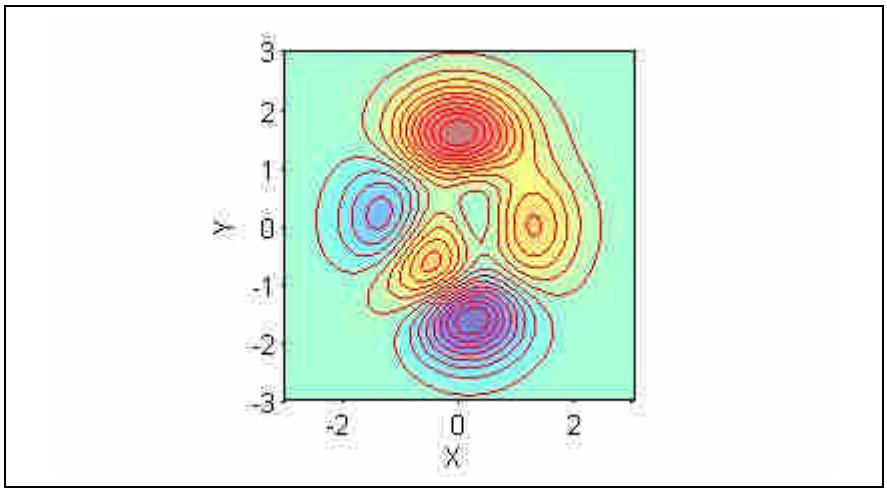
「基因演算法」學習速成

南台科技大學　電機系　趙春棠 講解

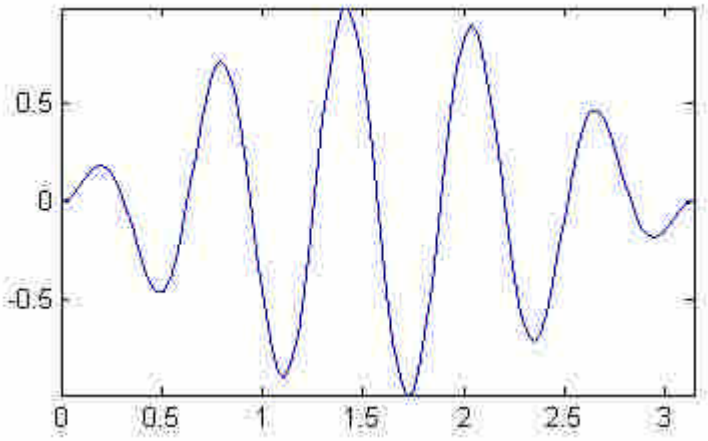
```
% 以下程式作者： 清大 張智星 教授，摘自 “Neuro-Fuzzy and Soft Computing,” J.-S. R. Jang, C.-T. Sun, and E. Mizutani【讀者可自 張教授網站下載該書籍中的所有 Matlab 程式】
% 主程式： go_ga.m
% 這是書中的一個範例，了解每一個程式指令後，大概就對「基因演算法」，就能有一全面性的初步了解了！
clear all;
close all;
clc;
generation_n = 30;  % Number of generations
popuSize = 20;              % Population size
xover_rate = 1.0;    % Crossover rate
mutate_rate = 0.01; % Mutation rate
bit_n = 8;        % Bit number for each input variable
global OPT_METHOD  % optimization method.
OPT_METHOD = 'ga';  % This is used for display in peaksfcn
figure;
blackbg;       % 執行 blackbg.m

obj_fcn = 'peaksfcn';      % Objective function: 'peaksfcn' or 'wavefcn'

if strcmp(obj_fcn, 'peaksfcn'),% objective function is 'peaksfcn'
      var_n = 2;              % Number of input variables
      range = [-3, 3; -3, 3];      % Range of the input variables
      % Plot peaks function
      peaks;          % 輸出如下（這是本例所解的問題：兩個輸入變數的函數，求極值）
```




```
colormap((jet+white)/2);
% Plot contours of peaks function
figure;
blackbg;
[x, y, z] = peaks;
pcolor(x,y,z); shading interp; hold on;
contour(x, y, z, 20, 'r');
hold off; colormap((jet+white)/2);
axis square; xlabel('X'); ylabel('Y');      % 輸出如下
```




```
else                    % objective function is 'wavefcn'
    var_n = 1;          % Number of input variables
    range = [0 pi];     % Range of the input variables
    x = linspace(range(1), range(2));
    y = sin(10*x).*sin(x);
    plot(x, sin(10*x).*sin(x));
    axis([-inf inf -inf inf]);      % 輸出如下（這是另一個求解的問題：單一輸入變數）
```



end

% Initial random population

popu = rand(popuSize, bit_n*var_n) > 0.5; % 本例即 rand(20, 8*2)

某次輸出（主程式 i=1）：popu （20*16）=

1	0	0	0	0	0	1	0	1	1	1	0	0	1	0	0
0	0	0	1	0	1	1	1	0	0	1	0	0	0	1	0
0	1	0	0	1	1	1	0	1	0	1	0	1	0	1	0
						.....									
1	1	0	0	1	1	0	1	1	0	0	0	1	1	1	0

upper = zeros(generation_n, 1); % 本例即 zeros(30, 1)

average = zeros(generation_n, 1); % 本例即 zeros(30, 1)

lower = zeros(generation_n, 1); %% 本例即 zeros(30, 1)

% Main loop of GA

for i = 1:generation_n; % 本例作 30 世代

    % delete unnecessary objects

    delete(findobj(0, 'tag', 'member'));

    delete(findobj(0, 'tag', 'individual'));

    delete(findobj(0, 'tag', 'count'));

    % Evaluate objective function for each individual

    fcn_value = evalpopu(popu, bit_n, range, obj_fcn);

某次輸出：（主程式 i=1） fcn_value （20*1） =

0.3304

0.5162
0.5897
.....
-0.2983

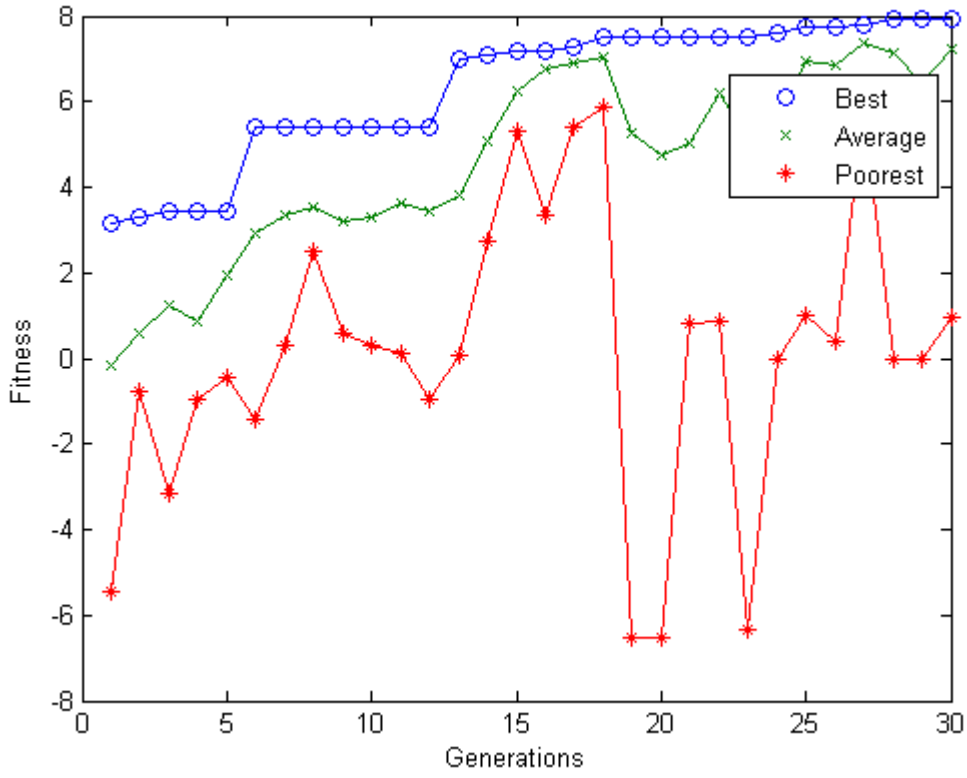
＊ 將以上 popu 二進位碼，計算出對應的數值，代入函數中即得上述結果。

```
%    if (i==1),
%        fprintf('Initial population.\n');
%        fprintf('Press any key to continue...\n');
%        pause;
%    end

    % Fill objective function matrices
    upper(i) = max(fcn_value);      % upper、average、lower 在本例都是 size=30*1 喔！
    average(i) = mean(fcn_value);
    lower(i) = min(fcn_value);

    % display current best
    [best, index] = max(fcn_value);
    fprintf('Generation %i: ', i);
    if strcmp(obj_fcn, 'peaksfcn'),% obj. function is 'peaksfcn'
        fprintf('f(%f, %f)=%f\n', ...
            bit2num(popu(index, 1:bit_n), range(1,:)), ...
            bit2num(popu(index, bit_n+1:2*bit_n), range(2,:)), ...
            best);
```

例： 某次最佳輸出 f(-0.223529, 1.000000)=3.628006 （主程式 i=1）
[best, index] = max(fcn_value); % 其中 best = 3.6280 index = 19 （i=1）
說明： popu(19,:) = 0 1 1 1 0 1 1 0 1 0 1 0 1 0 1 0
bit2num([0 1 1 1 0 1 1 0], [-3, 3])= -0.2235
bit2num([1 0 1 0 1 0 1 0], [-3, 3])= 1

```
    else                    % obj. function is 'wavefcn'
        fprintf('f(%f)=%f\n', bit2num(popu(index, :), range), best);
    end
    % generate next population via selection, crossover and mutation
    popu = nextpopu(popu, fcn_value, xover_rate, mutate_rate);
%    if (i==5),
```

```
%           fprintf('Population after the 5th generation.\n');
%           fprintf('Press any key to continue...\n');
%           pause;
%    end
%    if (i==10),
%           fprintf('Population after the 10th generation.\n');
%           fprintf('Press any key to continue...\n');
%           pause;
%    end
end

figure;
blackbg;
x = (1:generation_n)';
plot(x, upper, 'o', x, average, 'x', x, lower, '*');
hold on;
plot(x, [upper average lower]);
hold off;
legend('Best', 'Average', 'Poorest');
xlabel('Generations'); ylabel('Fitness');
```



Generation 1: f(0.317647, 2.270588)=3.144607
Generation 2: f(1.470588, -0.152941)=3.279510
Generation 3: f(1.376471, -0.152941)=3.456410
……
Generation 30: f(0.058824, 1.494118)=7.941152

【以下說明其他相關副程式】

function blackbg

```
% Change figure background to black
%    Issue this to change the background to black (V4 default)
%    Roger Jang, 981210
tmp = version;
```

註： tmp = ‘7.0.0.19920 (R14)’ % 以 Matlab 第七版爲例 tmp(1)= ‘7’ tmp(2)= ‘.’ tmp(3)= ‘0’ …

```
if str2num(tmp(1))==5, clf; colordef(gcf, 'black'); end
```

註： clf % 清除畫面 colordef(gcf, 'black') %背景設爲黑色

function fitness = evalpopu(population, bit_n, range, fcn)

```
%EVALPOPU    Evaluation of the population's fitness values.
%    population: 0-1 matrix of popu_n by string_leng
%    bit_n: number of bits used to represent an input variable
%    range: range of input variables, a var_b by 2 matrix
%    fcn: objective function (a MATLAB string)
global count
pop_n = size(population, 1);
fitness = zeros(pop_n, 1);
for count = 1:pop_n,
    fitness(count) = evaleach(population(count, :), bit_n, range, fcn);
end
```

function out = evaleach(string, bit_n, range, fcn)

```
% EVALEACH Evaluation of each individual's fitness value.
%    bit_n: number of bits for each input variable
%    string: bit string representation of an individual
%    range: range of input variables, a ver_n by 2 matrix
```

```
%     fcn: objective function (a MATLAB string)

var_n    = length(string)/bit_n;
input = zeros(1, var_n);
for i = 1:var_n,
      input(i) = bit2num(string((i-1)*bit_n+1:i*bit_n), range(i, :));
end
out = feval(fcn, input);

function num = bit2num(bit, range)
% BIT2NUM Conversion from bit string representations to decimal numbers.
%     BIT2NUM(BIT, RANGE) converts a bit string representation BIT ( a 0-1
%     vector) to a decimal number, where RANGE is a two-element vector
%     specifying the range of the converted decimal number.
%
%     For example:
%
%     bit2num([1 1 0 1], [0, 15])
%     bit2num([0 1 1 0 0 0 1], [0, 127])

%     Roger Jang, 12-24-94

integer = polyval(bit, 2);
num = integer*((range(2)-range(1))/(2^length(bit)-1)) + range(1);

function ver = matlabv
% MATLAB major version
tmp = version;
ver = str2num(tmp(1));

function z = peaksfcn(input)
%PEAKSFCN The PEAKS function.
%     PEAKSFCN(INPUT) returns the value of the PEAKS function at the INPUT.
%
%     See also PEAKS.

%     Roger Jang, 12-24-94.
```

```
global OPT_METHOD  % optimization method
global PREV_PT     % previous data point, used by simplex

x = input(1); y = input(2);
% The following function should be the same as the one in PEAKS.M
z =   3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
    - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
        - 1/3*exp(-(x+1).^2 - y.^2);

if matlabv==4,
    property='linestyle';
%elseif matlabv==5,
else
    property='marker';
%else
%   error('Unknown MATLAB version!');
end

% Plotting ...
if strcmp(OPT_METHOD, 'ga'), % plot each member; for GA
    line(x, y, property, 'o', 'markersize', 15, ...
        'clipping', 'off', 'erase', 'xor', 'color', 'w', ...
        'tag', 'member', 'linewidth', 2);
else  % plot input point for simplex method
    line(x, y, property, '.', 'markersize', 10, ...
        'clipping', 'off', 'erase', 'none', 'color', 'k', ...
        'tag', 'member');
    if ~isempty(PREV_PT), % plotting traj
        line([PREV_PT(1) x], [PREV_PT(2) y], 'linewidth', 1, ...
            'clipping', 'off', 'erase', 'none', ...
            'color', 'k', 'tag', 'traj');
    else  % plotting starting point
%       line(x, y, property, 'o', 'markersize', 10, ...
%           'clipping', 'off', 'erase', 'none', ...
%           'color', 'w', 'tag', 'member', 'linewidth', 3);
    end
    PREV_PT = [x y];
end
```

```
drawnow;

function new_popu = nextpopu(popu, fitness, xover_rate, mut_rate)
% generate next population via selection（選擇）, crossover（交配） and mutation（突變）
%  呼叫本副程式例：  popu = nextpopu(popu, fcn_value, xover_rate, mutate_rate);
new_popu = popu;
popu_s = size(popu, 1);
string_leng = size(popu, 2);
```

本例說明：

```
>> size(popu)
   ans =       20       16
>> popu_s = size(popu, 1)
   popu_s = 20
>> string_leng = size(popu, 2)
   string_leng = 16
```

```
% ====== ELITISM: find the best two and keep them
tmp_fitness = fitness;                    % fitness 即是  fcn_value
[junk, index1] = max(tmp_fitness);        % find the best  本例某次： junk = 3.6280 index1 = 19 （主程式 i=1）
tmp_fitness(index1) = min(tmp_fitness);   %本例某次： tmp_fitness(19) =  -3.0034 （主程式 i=1）
[junk, index2] = max(tmp_fitness);        % find the second best  本例某次：  junk = 2.8788   index2 =  3 （主程式 i=1）
new_popu([1 2], :) = popu([index1 index2], :);
```

說明：本例某次 popu([index1 index2], :) 即是 popu([19 3], :)

```
ans = 0   1   1   1   0   1   1   0   1   0   1   0   1   0   1   0
      0   1   1   0   1   1   0   1   1   1   1   0   0   0   0   0
```

以上最好的兩組，將被保留。

```
% rescaling the fitness
fitness = fitness - min(fitness);      % keep it positive  恆正，fitness（20*1 喔！）
total = sum(fitness);   %本例某次：total = 68.7807 （主程式 i=1）
if total == 0,
    fprintf('=== Warning: converge to a single point ===\n');
```

```
    fitness = ones(popu_s, 1)/popu_s;  % sum is 1
else
    fitness = fitness/sum(fitness);        % sum is 1
```

說明：本例某次 （主程式 i=1），after fitness scaling

```
fitness =
    0.0429
    0.0429
    0.0855
    ……..
    0.0385
```

```
end
cum_prob = cumsum(fitness);
```

說明：本例某次 （主程式 i=1）

```
cum_prob =
    0.0429
    0.0858
    0.1713
    …….
    1.0000
```

```
% ====== SELECTION and CROSSOVER
  for i = 2:popu_s/2,                    % 本例 i = 2: 20/2 = 2:10 （共 9 次，每次得兩個基因，共 18 個，加上之前保留最佳的兩個，共 20 個）
    % === Select two parents based on their scaled fitness values
    tmp = find(cum_prob - rand > 0);
```

說明：本例某次 （主程式 i=1 ，本副程式 i=1）

```
tmp =
    16
    17
    18
    19
    20
```

```
    parent1 = popu(tmp(1), :);
```

說明：本例某次 （主程式 i=1 ，本副程式 i=1）

Parent1 = 0 1 0 0 1 0 0 0　　1 1 0 0 0 1 0 1

```
tmp = find(cum_prob - rand > 0);
```

說明：本例某次 （主程式 i=1 ，本副程式 i=1）

tmp （19*1）=

2

3

……..

20

```
parent2 = popu(tmp(1), :);
```

說明：本例某次 （主程式 i=1 ，本副程式 i=1）

Parent2 = 0 0 0 0 1 1 1 0　　1 0 1 0 1 0 1 1

```
% === Do crossover
if rand < xover_rate,                  %  本例  xover_rate = 1.0;
    % Perform crossover operation
    xover_point = ceil(rand*(string_leng-1));     %  本例  string_leng = 16
```

說明：本例某次 （主程式 i=1 ，本副程式 i=1）

```
    >> rand*(string_leng-1)
    ans =    1.7225
    >> ceil(ans)
    ans =    2 （故  xover_point =2）
```

註： CEIL(X) rounds the elements of X to the nearest integers towards infinity.

```
    new_popu(i*2-1, :) = ...
        [parent1(1:xover_point) parent2(xover_point+1:string_leng)];

    new_popu(i*2, :) = ...
        [parent2(1:xover_point) parent1(xover_point+1:string_leng)];
```

說明：本例某次 （主程式 i=1 ，本副程式 i=1），「交配」結果整理（xover_point =2）

Parent1　= **0 1 0 0 1 0 0 0　　1 1 0 0 0 1 0 1**

Parent2　= **0 0 0 0 1 1 1 0　　1 0 1 0 1 0 1 1**

產生子代如下：

new_popu1 = **0 1 0 0 1 1 1 0　　1 0 1 0 1 0 1 1**

new_popu2 = **0 0 0 0 1 0 0 0　　1 1 0 0 0 1 0 1**

```
    end
%   fprintf('xover_point = %d\n', xover_point);
%   disp(parent1);
%   disp(parent2);
%   disp(new_popu(i*2-1, :));
```

```
%    disp(new_popu(i*2, :));
%    keyboard;
end

% ====== MUTATION (elites are not subject to this.)
mask = rand(popu_s, string_leng) < mut_rate;        % 本例 popu_s = 20
string_leng = 16    mutate_rate = 0.01
```

說明：本例某次（主程式 i=1 ，本副程式 i=1）結果

mask（20*16） =

```
    0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
          ...................
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
```

```
new_popu = xor(new_popu, mask);    % 上述 mask 表中，若爲「1」者，表示對應的該染色體會突變（「1」=>「0」,「0」=>「1」）
% restore the elites
new_popu([1 2], :) = popu([index1 index2], :);        % 可見前面所保留最好的基因，並不允許突變
```

■ 後記

晚學執行以上程式後，明瞭了「基因演算法」的整個大概的流程，不過對於 new_popu.m 程式中的「交配」過程，感覺似乎並不是很妥當。以前述兩個母代（如下）交配爲例：

Parent1 = **0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 1**

Parent2 = **0 0 0 0 1 1 1 0 1 0 1 0 1 0 1 1**

其實我們知道在 Parent1 中，前 8 個 bits 「0 1 0 0 1 0 0 0」代表的是一個變數；後 8 個 bits「1 1 0 0 0 1 0 1」代表的是另一個變數，彼此之間理論上應是「相互獨立」的，所以作「交配」時，也應該分開作才合理。於是筆者更動部份程式如下：

new_popu.m（程式修改）

```
…….（略）……..
string_leng = size(popu, 2);
string_leng_hf = string_leng/2;    % 新增 指令
…….（略）……..
% ====== SELECTION and CROSSOVER
for i = 2:popu_s/2,
```

```
    % === Select two parents based on their scaled fitness values
    tmp = find(cum_prob - rand > 0);
    parent1 = popu(tmp(1), :);
    parent1_hl = popu(tmp(1), 1:string_leng_hf); parent1_hr = popu(tmp(1),
string_leng_hf+1:end);   % 新增 指令
    tmp = find(cum_prob - rand > 0);
    parent2_hl = popu(tmp(1), 1:string_leng_hf); parent2_hr = popu(tmp(1),
string_leng_hf+1:end);   % 新增 指令
    parent2 = popu(tmp(1), :);
     % === Do crossover
     if rand < xover_rate,
          % Perform crossover operation
          % xover_point = ceil(rand*(string_leng-1)); 原指令
          xover_point = ceil(rand*(string_leng_hf-1));     % 新指令
          %new_popu(i*2-1, :) = ...     原指令
          %       [parent1(1:xover_point) parent2(xover_point+1:string_leng) ];
          %       原指令
          new_popu(i*2-1, :) = ...
                    [parent1_hl(1:xover_point)
                         parent2_hl(xover_point+1:string_leng_hf)
                         parent1_hr(1:xover_point)
                         parent2_hr(xover_point+1:string_leng_hf)];
          %new_popu(i*2, :) = ...         原指令
          %       [parent2(1:xover_point) parent1(xover_point+1:string_leng)];
          %       原指令
          new_popu(i*2, :) = ...
                    [parent2_hl(1:xover_point)
                         parent1_hl(xover_point+1:string_leng_hf)
                         parent2_hr(1:xover_point)
                         parent1_hr(xover_point+1:string_leng_hf)];
     end
```

說明：本例某次 （主程式 i=1 ，本副程式 i=1），「交配」結果整理（xover_point =2）

Parent1 = **0 1 0 0 1 0 0 0**（hl 部分） **1 1 0 0 0 1 0 1**（hr 部分）

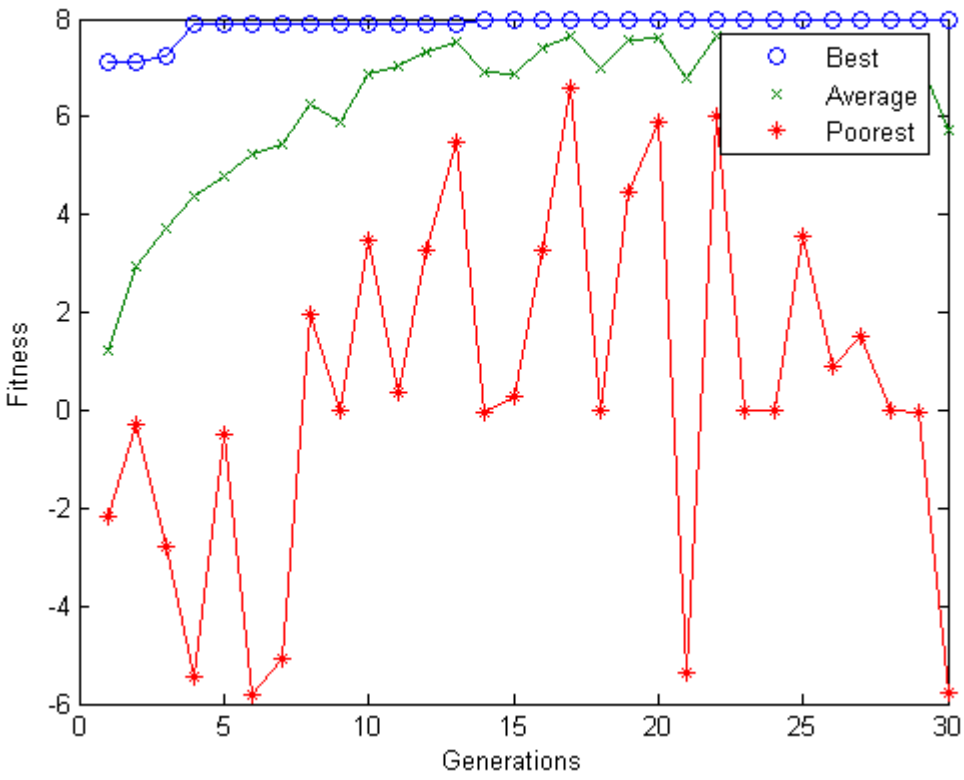
Parent2 = **0 0 0 0 1 1 1 0**（hl 部分） **1 0 1 0 1 0 1 1**（hr 部分）

產生子代如下：

new_popu1 = **0 1 0 0 1 1 1 0** **1 1 1 0 1 0 1 1**

new_popu2 = **0 0 0 0 1 0 0 0** **1 0 0 0 0 1 0 1**

以下是某次執行的結果，我們發現有時效果好許多喔！僅作爲讀者的參考！



此外，無論是原作者的程式，或是晚學修改後的程式，讀者不妨多執行幾次看看，會發現有時候會落入「local」的「極值」喔！可見得這個程式仍然是有問題的。除了可以改變 mutation rate 試試以外，本程式在「交配」時「母代」的選擇上，似乎是完全隨機的，此點似乎可改善，