Evolutionary Strategies

Dr. Rajib Kumar Bhattacharjya
Department of Civil Engineering
IIT Guwahati
Email: rkbc@iitg.ernet.in

6 November 2015

Evolutionary Strategies

- ES use real parameter value
- ES does not use crossover operator
- It is just like a real coded genetic algorithms with selection and mutation operators only

Two members ES: (1+1) ES

- In each iteration one parent is used to create one offspring by using Gaussian mutation operator

Two members ES: (1+1) ES

- Step1: Choose a initial solution x and a mutation strength σ
- Step2: Create a mutate solution

$$y = x + N(0, \sigma)$$

- Step 3: If $f(y) < f(x)$, replace x with y
- Step4: If termination criteria is satisfied, stop, else go to step 2

Two members ES: (1+1) ES

- Strength of the algorithm is the proper value of σ

- Rechenberg postulate
  - The ratio of successful mutations to all the mutations should be 1/5. If this ratio is greater than 1/5, increase mutation strength. If it is less than 1/5, decrease the mutation strength.

Two members ES: (1+1) ES

- A mutation is defined as successful if the mutated offspring is better than the parent solution.
- If P_s is the ratio of successful mutation over *n* trial, Schwefel (1981) suggested a factor $C_d = 0.817$ in the following σ update rule

$$\sigma^{t+1} = \begin{cases} C_d \sigma^t & if\ P_s < 1/5 \\ \dfrac{1}{C_d} \sigma^t & if\ P_s < 1/5 \\ \sigma^t & if\ P_s = 1/5 \end{cases}$$

Matlab code


```
sigma = 1;
x0 = [1 1];
[n m] = size(x0);
for j=1:1000
for i =1:m
    f0 = objfunc(x0);
    x1 = x0;
    x1(i) =x0(i)*randn(1)*sigma;
    f1 = objfunc(x1);
    if (f1<f0)
        x0 = x1;
    end
end
end
disp(['Optimal solution X=    ', num2str(x0)]);
```

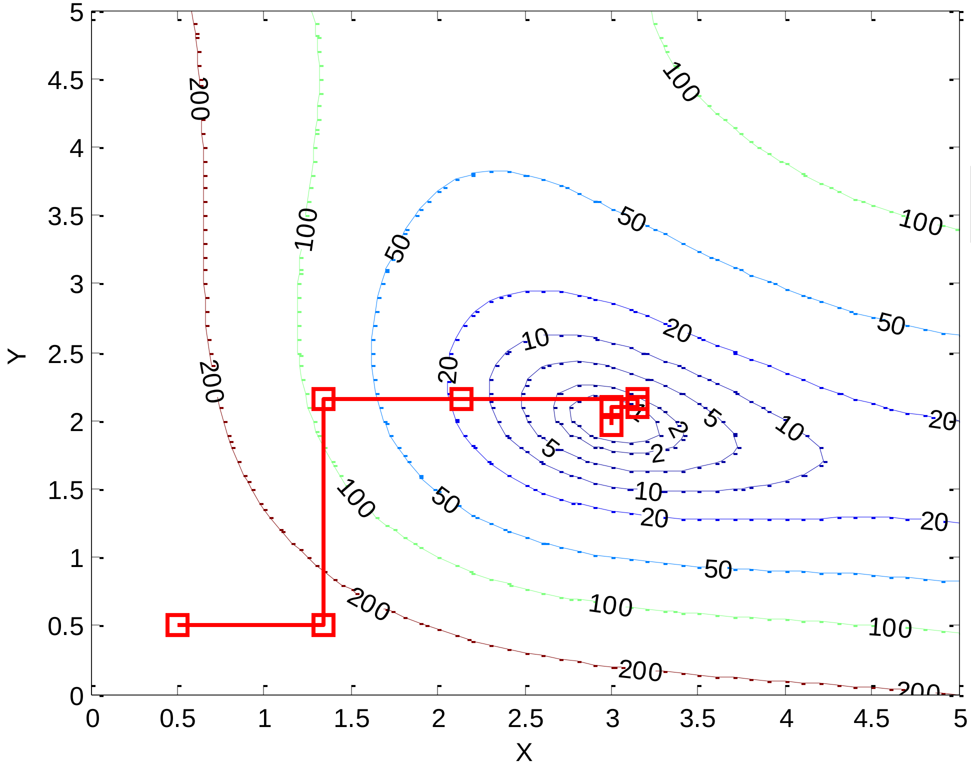
```
function [f] = objfunc( x )

f=(x(1)^2+x(2)-11)^2+(x(1)+x(2)^2-7)^2;
end
```

```matlab
% This programme will implement 1+1 ES
bx = [0 5]; % Upper bound
by = [0 5]; % Lower bound
plotfunction(bx,by) % Ploting the function between upper bound and lower
bound defined above
hold on;
x0 = [0.5 0.5]; % Starting point or initial solution
sigma = 5; % Define sigma value
imax = 3000; % maximum iteration
k =0; % An counter
success =0; % Success counter
[n m] = size(x0);
x11 =x0; % x11 will store solution of all the iteration
for j=1:imax % The program will terminate after 3000 iteration
    k=k+1;
for i =1:m
    f0 = objfunc(x0); % objjunc will calculate the objective function value
    x1 = x0;
    x1(i) =x0(i)*randn(1)*sigma; % Will generate a new solution
    f1 = objfunc(x1);
    if (f1<f0)
        x0 = x1;
        success = success+1;
    end
    x11 = [x11; x0];
end
% Updating sigma value as per Rechenberg postulate after every 20 iterations
if(k==20)
    if(success/k>1/5)
        sigma = sigma/0.817;
    else
        sigma = sigma*0.817;
    end
    k=0;
    success =0;
end
end
plot(x11(:,1), x11(:,2),'-rs','linewidth',2,'MarkerSize',10); % plot the
solution
disp(['Optimal solution X=   ', num2str(x0)]);
disp(['Optimal function value f=   ', num2str(f0)]);
```

Some results of 1+1 ES

$$Minimize\ f = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$



Optimal Solution is
X*= [3.00 1.99]

Objective function value f
= 0.0031007

Multimember ES

$(\mu+\lambda)$ ES

Step1: Choose an initial population of μ solutions and mutation strength σ

Step2: Create λ mutated solution

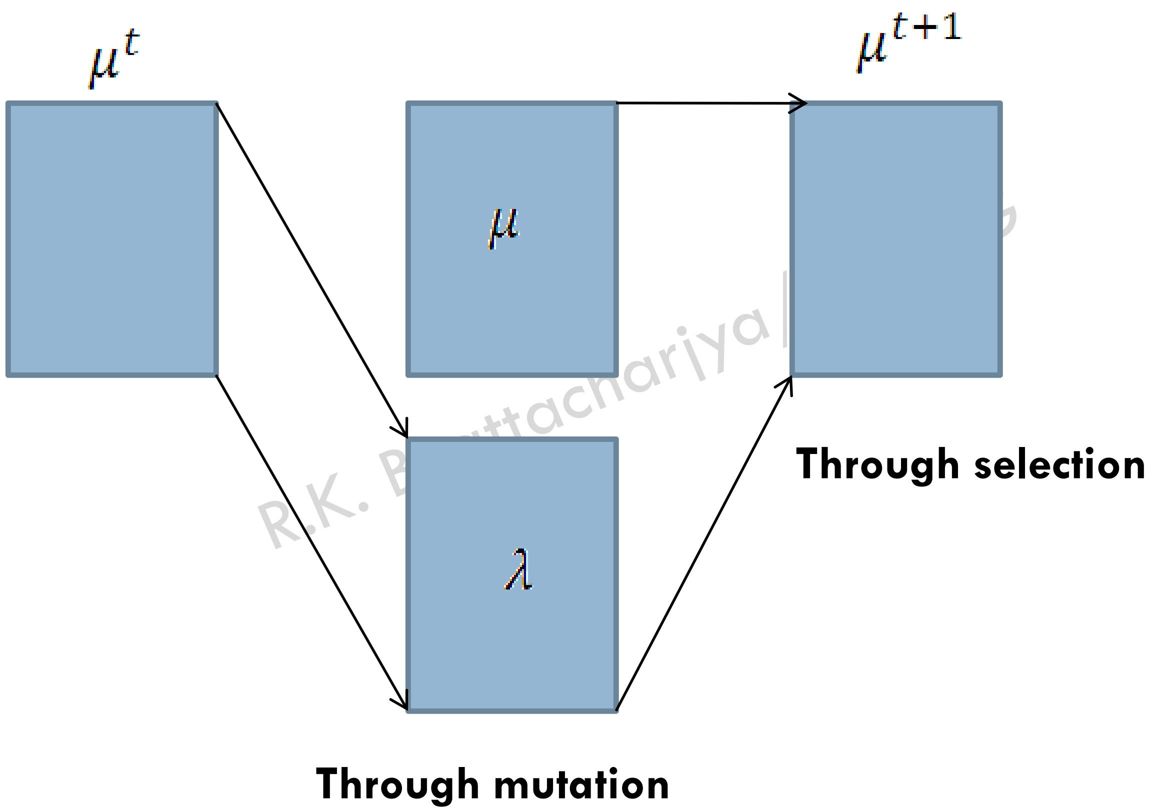
$$y^i = x^i + N(0,\sigma)$$

Step3: Combine x and y, and choose the best solutions μ

Step4: Terminate? Else go to step 2

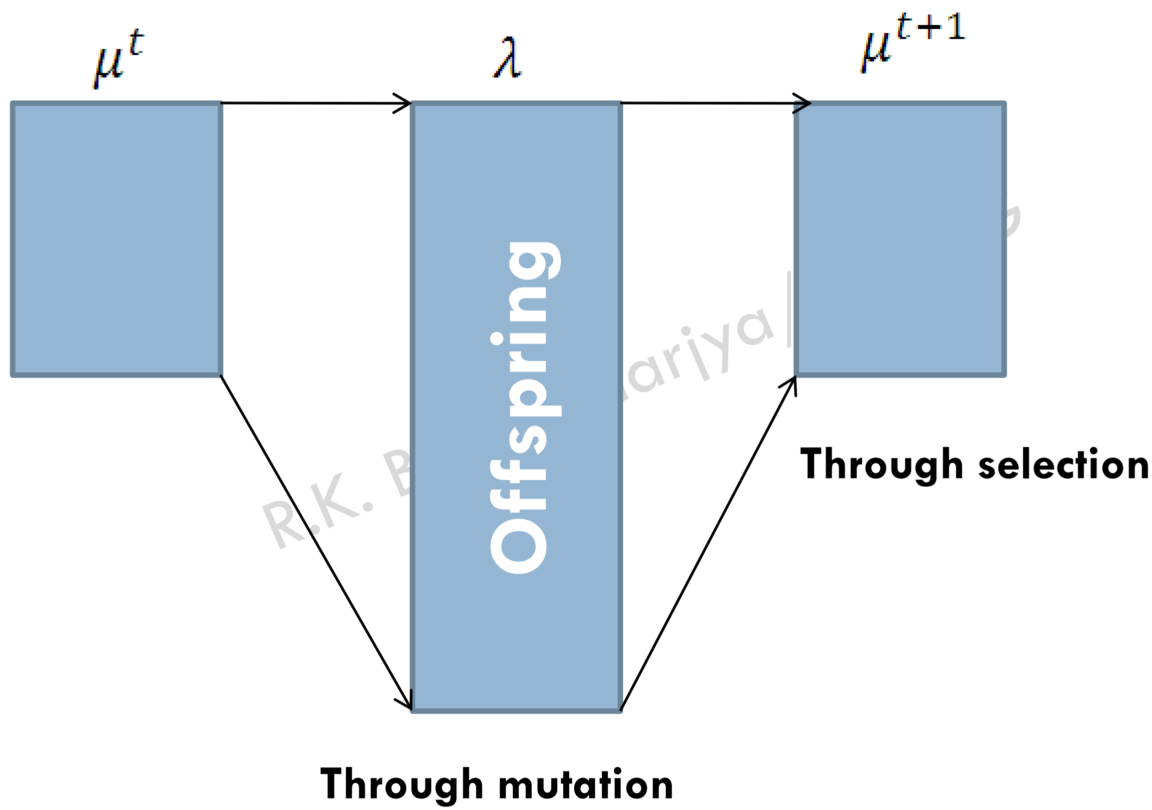
Multimember ES

$(\mu + \lambda)$ ES



Multimember ES

(μ, λ) ES



THANKS