Code Part SA

### cost.m

function result = cost(x)

result = 0;

leng = length(x);

prod = 1;

for i = 1:leng

if (x(i) < 0 || x(i) > 10)

return;

end

prod = prod \* x(i);

end

if prod < 0.75

return;

end

num1 = 0;

num2 = 2;

denum = 0;

for i = 1:leng

num1 = num1 + (cos(x(i)))^4;

num2 = num2 \* (cos(x(i)))^2;

denum = denum + i \* (x(i))^2;

end

result = -1 \* abs((num1-num2)/sqrt(denum));

### neighbour2D.m

function x = neighbor2D(x)

index = randi(length(x));

newval = x(index) + sqrt(5).\*randn(1,1);

while (newval > 10 || newval < 0)

newval = x(index) + sqrt(5).\*randn(1,1);

end

x(index) = newval;

### SA.m

% Simulated Annealing

function [solution, sbest] = SA(sinitial, Tinitial, alpha, beta, Minitial, maxiter)

M = Minitial;

T = Tinitial;

CurS = sinitial;

BestS = CurS;

CurCost = cost(CurS);

BestCost = CurCost;

solution = zeros(maxiter, 3);

sbest = zeros(maxiter, 20);

Time = 1;

while (Time <= maxiter)

[CurS\_res, CurCost\_res, BestS\_res, BestCost\_res] = Metropolis(CurS, CurCost, BestS, BestCost, T, M);

solution(Time:Time+M-1,1) = Time:Time+M-1;

solution(Time:Time+M-1,2) = CurCost\_res;

solution(Time:Time+M-1,3) = BestCost\_res;

CurS = CurS\_res(M,:);

CurCost = CurCost\_res(M);

BestS = BestS\_res(M,:);

BestCost = BestCost\_res(M);

sbest(Time:Time+M-1,:) = BestS\_res;

Time = Time + M;

T = alpha \* T; % Update T after M iterations

M = beta \* M;

end

% Metropolis

function [CurS\_res, CurCost\_res, BestS\_res, BestCost\_res] = Metropolis(CurS, CurCost, BestS, BestCost, T, M)

i = 1;

CurS\_res = zeros(M,20);

CurCost\_res = zeros(M);

BestS\_res = zeros(M,20);

BestCost\_res = zeros(M);

while (i <= M)

NewS = neighbor2D(CurS);

NewCost = cost(NewS);

DeltaCost = (NewCost - CurCost);

if (DeltaCost < 0)

CurS = NewS;

CurCost = NewCost;

if NewCost < BestCost

BestS = NewS;

BestCost = NewCost;

end

else

if (rand(1) < exp( -1. \* DeltaCost / T))

CurS = NewS;

CurCost = NewCost;

end

end

CurS\_res(i,:) = CurS;

CurCost\_res(i) = CurCost;

BestS\_res(i,:) = BestS;

BestCost\_res(i) = BestCost;

i = i + 1;

end

### main.m

Tinitial = 0.00585;

alpha = 0.99995;

repetition = 20;

r = 10.\*rand(repetition,20);

bestsolution = zeros(10000,1);

fprintf('running SA:\n[');

for i = 1: repetition

[solution, ~] = SA(r(i,:), Tinitial, alpha, 1, 1, 10000);

bestsolution = (bestsolution .\* (i-1) + solution(:, 3)) ./ i;

fprintf('%d ', solution(10000, 3));

end

fprintf(']\n');

plot(1:10000, -bestsolution, 'g:');

Code Part GA

### main.m

% Constants

K = 1;

POP\_SIZE = 50;

ITERATION = 200;

P\_MUTATION = 0.7;

P\_CROSSOVER = 0.5;

V = 5;

res = zeros(1, 20);

result = zeros(ITERATION, 20);

bestsolution = zeros(200,1);

fprintf('running GA:\n[');

for temp = 1:20

% Generate initial population. n=?

InitPop = generatePopulation(POP\_SIZE, 20);

result(:, temp) = GAReal2Q(InitPop, POP\_SIZE, ITERATION, P\_CROSSOVER, P\_MUTATION, V);

bestsolution = (bestsolution .\* (temp - 1) + result(:, temp)) ./ temp;

fprintf('%d ', result(ITERATION, temp));

end

fprintf(']\n');

plot(POP\_SIZE:POP\_SIZE:10000, bestsolution, 'r');

Code Part DDS

### DDS\_inp.txt

% Comment line 1: READ WITH WORD WRAP OFF. Input control file for Matlab DDS ver1.1mp algorithm.

% Comment line 2: Inputs start on line 3. Inputs must be followed by a space or "%" symbol. Some lines can be blank.

bump2 % 3. MATLAB m-file function (without .m extension) OR executable file (as \*.exe) OR batch file (\*.bat)

bump % 4. compact name to append to all algorithm output files

20 % 5. number of optimization trials to run

10000 % 6. maximum number of objective function evaluations per optimization trial

568768723 % 7. random integer to fix Matlab random number generators

787232345 % 8. 2nd random integer to fix Matlab random number generators

0 % 9. Print flag: "0" saves all DDS outputs (max # files) or "1" to save only summary info (min # of files)

% 10. blank if random initial solutions, else Enter filename (exponential or free format, rows are sols, columns are DVs) holding initial soln matrix

% 11. If applicable, enter subdirectory name (or full directory path) containing objective function file (.m, .exe or .bat), else leave blank

% ! next line ! % 12. On NEXT LINE, enter any other comments to save about this run (100 char max):

-1 % 14. MAX problem (enter "-1") or MIN problem (enter "1")

0.2 % 15. r\_val, DDS neighborhood size parameter (0.2 is default and works well). Allowable range is (0.0, 1.0]. If you experiment with r\_val, reduce to <0.2

### Bump2.m

function result = bump2(x)

result = 0;

leng = length(x);

prod = 1;

for i = 1:leng

if (x(i) < 0 || x(i) > 10)

return;

end

prod = prod \* x(i);

end

if prod < 0.75

return;

end

num1 = 0;

num2 = 2;

denum = 0;

for i = 1:leng

num1 = num1 + (cos(x(i)))^4;

num2 = num2 \* (cos(x(i)))^2;

denum = denum + i \* (x(i))^2;

end

result = abs((num1-num2)/sqrt(denum));

Code Part Main

### Q1d.m

%DDS:

DDS = [0.72513 0.70825 0.71831 0.74809 0.71433 0.76385 0.76094 0.69754 0.73678 0.74212 0.75094 0.71568 0.76212 0.75417 0.70109 0.70675 0.73881 0.70731 0.73868 0.75904];

% running GA:

GA = [5.873824e-001 5.753190e-001 6.068312e-001 5.901685e-001 5.844889e-001 5.882655e-001 6.019899e-001 5.771182e-001 5.453153e-001 4.490493e-001 6.121610e-001 6.001846e-001 4.868738e-001 5.687299e-001 5.006061e-001 4.768492e-001 5.565101e-001 6.185262e-001 6.295882e-001 5.898114e-001 ]

% running SA:

SA = [-7.457496e-001 -7.410139e-001 -7.536141e-001 -7.551384e-001 -7.536895e-001 -7.712034e-001 -7.663489e-001 -7.726179e-001 -7.677715e-001 -7.508422e-001 -7.587998e-001 -7.728770e-001 -7.615266e-001 -7.285694e-001 -7.561617e-001 -7.521449e-001 -7.447165e-001 -7.678549e-001 -7.689770e-001 -6.887951e-001 ]

SA = -SA;

% Combining:

res(1,:) = DDS;

res(2,:) = GA;

res(3,:) = SA;

% boxplot

figure

boxplot(res','labels', {'DDS' 'GA' 'SA'});

% ecdf

[fdds,xdds,flo,fup] = ecdf(DDS);

[fga,xga,flo,fup] = ecdf(GA);

[fsa,xsa,flo,fup] = ecdf(SA);

figure

hold on

plot(xdds,fdds, 'r');

plot(xga,fga, 'g');

plot(xsa,fsa, 'b');

ylabel('cumulative probability');

xlabel('objective function value');

title('Empirical CDF for DDS, GA, and SA');

legend('DDS', 'GA', 'SA');

### run\_all.m

figure

hold on

%assume DDS is prerun and preprocessed

load DDS/bump/bump\_AVG.out

plot(1:10000, bump\_AVG(:,2), 'b--');

%run GA

run GA/main.m

%run SA

run SA/main.m

title('average best bump function value vs. function evaluations')

xlabel('function evaluations')

ylabel('average best bump function value')

legend('DDS', 'GA', 'SA')