### Code Part

##### cost.m

%calculates cost function

function c = cost(s)

c = (400-(s-21) .\* (s-21)) .\* sin(s .\* pi ./ 6);

##### neighbor.m

%find new neighbor in (-25,25)

function snew = neighbor(s)

%get lowerbound so that it doesn't go below 0

lowbound = s - 25;

if lowbound < 0

lowbound = 0;

end

%get higherbound so that it doesn't go above 500

highbound = s + 25;

if highbound > 500

highbound = 500;

end

%keep generating random number if the number generated equals the current

%number

while (true)

snew = floor(rand\*(highbound - lowbound + 1)) + lowbound;

if snew ~= s

break

end

end

##### RW.m

%random walk

function solution = RW(sinitial, maxiterations)

S = sinitial;

BestS = S;

BestCost = cost(S);

solution = zeros(maxiterations, 5);

i = 0;

%repeat while max iteration is not reached

while (i < maxiterations)

i = i + 1;

%get a neighbor within (-25,25)

S = neighbor(S);

CostS = cost(S);

%update cost if the new cost is better than the best cost

if CostS < BestCost

BestCost = CostS;

BestS = S;

end

%add solution to a matrix for plotting purposes

solution(i, 1) = i;

solution(i, 2) = S;

solution(i, 3) = BestS;

solution(i, 4) = CostS;

solution(i, 5) = BestCost;

end

##### GD.m

% Greed deterministic

function solution = GD(sinitial, maxiterations)

S = sinitial;

CostS = cost(S);

BestS = S;

BestCost = CostS;

solution = zeros(maxiterations, 5);

i = 0;

%repeat until max iterations is reached

while (i < maxiterations)

i = i + 1;

%update cost

tempCost = cost(S);

tempSoln = S;

%search through all neighborhood for best solution

for j = max(S-10, 0) : min(S+10, 500)

%get best cost in the neighborhood

if cost(j) < tempCost

tempCost = cost(j);

tempSoln = j;

end

end

S = tempSoln;

% update best cost if a better solution is found

CostS = cost(S);

if CostS < BestCost

BestCost = CostS;

BestS = S;

end

%store solution into matrix for display purposes

solution(i, 1) = i;

solution(i, 2) = S;

solution(i, 3) = BestS;

solution(i, 4) = CostS;

solution(i, 5) = BestCost;

end

##### GS.m

%Greed stochastic

function solution = GS(sinitial, maxiterations)

S = sinitial;

CostS = cost(S);

BestS = S;

BestCost = CostS;

solution = zeros(maxiterations, 5);

i = 0;

%repeat until max iteration is reached

while (i < maxiterations)

i = i + 1;

oldS = S;

oldCostS = cost(oldS);

%find a random neighbor

S = neighbor(S);

%determine if the neighbor is better than the current solution

CostS = cost(S);

if CostS < BestCost

BestCost = CostS;

BestS = S;

end

%store current iteration into matrix for plot purposes

solution(i, 1) = i;

solution(i, 2) = S;

solution(i, 3) = BestS;

solution(i, 4) = CostS;

solution(i, 5) = BestCost;

if (oldCostS < CostS)

S = oldS;

end

end

##### f.m: (Codes used to generate graphs for part f)

%for generating results

clc

% Part 1)

maxiter = 200;

%generate a random starting point

sinitial = rand(1) \* 501;

%run random walk

result = RW(sinitial, maxiter);

%display the results in plot

figure;

title('RW: scurrent & sbest vs. iteration');

hold on;

plot(result(:,1), result(:,2), 'r');

xlabel('iteration');

ylabel('scurrent');

% sbest

plot(result(:,1), result(:,3), 'g');

xlabel('iteration');

ylabel('s');

legend('scurrent', 'sbest');

% Part 2)

maxiter = 200;

% run random walk 30 times with different starting points

tempresult = zeros(30,maxiter,5);

for k=1:30

sinitial = rand(1) \* 501;

tempresult(k,:,:) = RW(sinitial, maxiter);

end

%find mean of the results

result = mean(tempresult, 1);

%display the results in plot

% scurrent

figure;

hold on;

title('RW: average(COSTcurrent) & average(COSTbest) vs. iterations');

plot(result(1,:,1), result(1,:,4), 'r');

xlabel('iteration');

ylabel('COSTcurrent');

% sbest

plot(result(1,:,1), result(1,:,5), 'g');

xlabel('iteration');

ylabel('Cost');

legend('average(COSTcurrent)', 'average(COSTbest)');

% Part 3)

maxiter = 200;

%run all algorithms 30 times

tempresult1 = zeros(30,maxiter,5);

tempresult2 = zeros(30,maxiter,5);

tempresult3 = zeros(30,maxiter,5);

tempresult4 = zeros(30,maxiter,5);

for k=1:30

sinitial = rand(1) \* 501;

tempresult1(k,:,:) = RW(sinitial, maxiter);

tempresult2(k,:,:) = RS(sinitial, maxiter);

tempresult3(k,:,:) = GD(sinitial, maxiter);

tempresult4(k,:,:) = GS(sinitial, maxiter);

end

%find the mean for all algorithms

result1 = mean(tempresult1, 1);

result2 = mean(tempresult2, 1);

result3 = mean(tempresult3, 1);

result4 = mean(tempresult4, 1);

%display the results in plots

% Multiple plots

figure;

hold on;

title('all algorithms: average(COSTbest) vs. iterations');

plot(result1(1,:,1), result1(1,:,5), 'r');

plot(result2(1,:,1), result2(1,:,5), 'g');

plot(result3(1,:,1), result3(1,:,5), 'b');

plot(result4(1,:,1), result4(1,:,5), 'c');

legend('RW', 'RS', 'GD', 'GS');

xlabel('iteration');

ylabel('COSTbest');

% Part 4)

maxiter = 100;

%run all algorithms 30 times

tempresult1 = zeros(30,maxiter,5);

tempresult2 = zeros(30,maxiter,5);

tempresult3 = zeros(30,maxiter,5);

tempresult4 = zeros(30,maxiter,5);

for k=1:30

sinitial = rand(1) \* 501;

tempresult1(k,:,:) = RW(sinitial, maxiter);

tempresult2(k,:,:) = RS(sinitial, maxiter);

tempresult3(k,:,:) = GD(sinitial, maxiter);

tempresult4(k,:,:) = GS(sinitial, maxiter);

end

%get SBest

Sresult1 = tempresult1(:,maxiter, 3);

Sresult2 = tempresult2(:,maxiter, 3);

Sresult3 = tempresult3(:,maxiter, 3);

Sresult4 = tempresult4(:,maxiter, 3);

%find mean of each algorithm (SBest)

mean1 = mean(Sresult1);

mean2 = mean(Sresult2);

mean3 = mean(Sresult3);

mean4 = mean(Sresult4);

%find standard deviation of each algorithm (Sbest)

stdv1 = std(Sresult1);

stdv2 = std(Sresult2);

stdv3 = std(Sresult3);

stdv4 = std(Sresult4);

%display results in MATLAB output

fprintf('\n\n');

display('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');

display('%%% Average & Std Dev of SBest %%%');

display('% Algorithm average std dev %');

display('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');

fprintf('%% Rand Walk %10.2f %10.2f %%\n', mean1, stdv1);

fprintf('%% Rand Sample %10.2f %10.2f %%\n', mean2, stdv2);

fprintf('%% Greedy Det %10.2f %10.2f %%\n', mean3, stdv3);

fprintf('%% Greedy Stoc %10.2f %10.2f %%\n', mean4, stdv4);

display('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');

%get CostBest

Cresult1 = tempresult1(:,maxiter, 5);

Cresult2 = tempresult2(:,maxiter, 5);

Cresult3 = tempresult3(:,maxiter, 5);

Cresult4 = tempresult4(:,maxiter, 5);

%find mean of each algorithm (CostBest)

mean1 = mean(Cresult1);

mean2 = mean(Cresult2);

mean3 = mean(Cresult3);

mean4 = mean(Cresult4);

%find standard deviation of each algorithm (CostBest)

stdv1 = std(Cresult1);

stdv2 = std(Cresult2);

stdv3 = std(Cresult3);

stdv4 = std(Cresult4);

%display results in MATLAB output

fprintf('\n\n');

display('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');

display('%%% Average & Std Dev of CostBest %%%');

display('% Algorithm average std dev %');

display('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');

fprintf('%% Rand Walk %10.2f %10.2f %%\n', mean1, stdv1);

fprintf('%% Rand Sample %10.2f %10.2f %%\n', mean2, stdv2);

fprintf('%% Greedy Det %10.2f %10.2f %%\n', mean3, stdv3);

fprintf('%% Greedy Stoc %10.2f %10.2f %%\n', mean4, stdv4);

display('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');