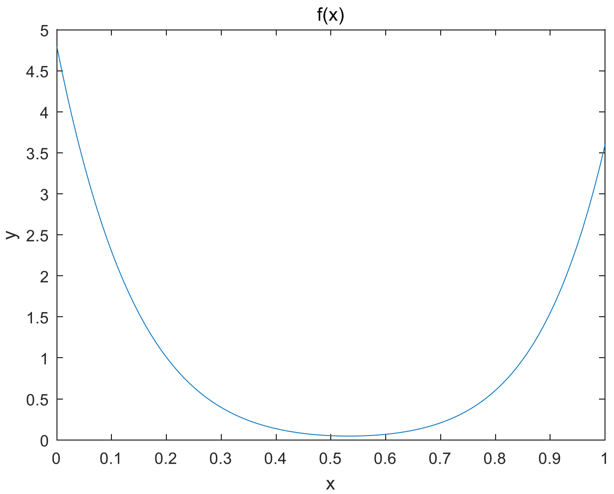
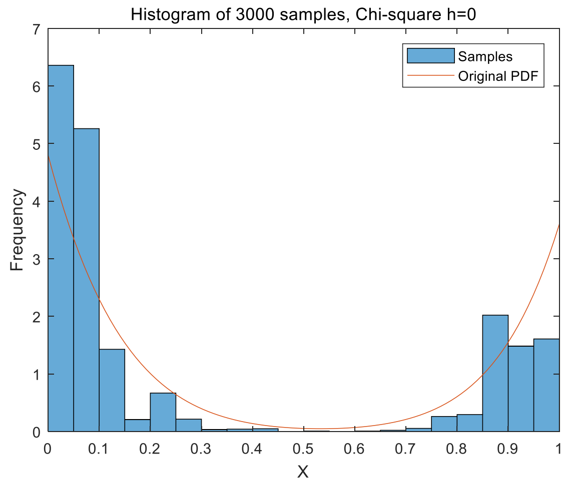
EE511 Project#5 Optimization & Sampling via MCMC

This project is implemented using MATLAB. Codes are attached in the end of this document and also available at https://github.com/uscwy/ee511project5/blob/master/project5.m

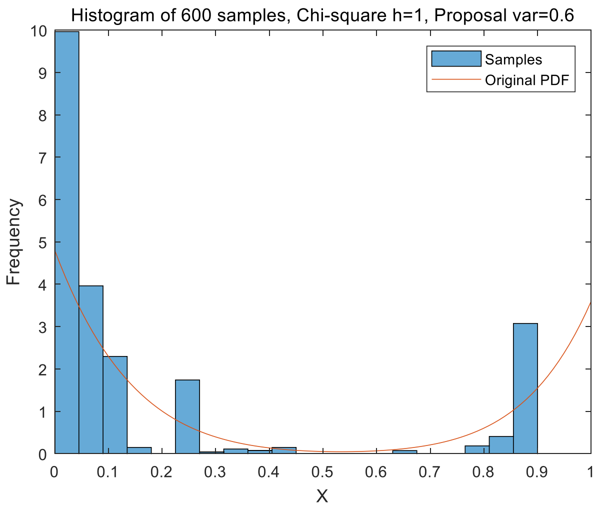
[MCMC for Sampling]

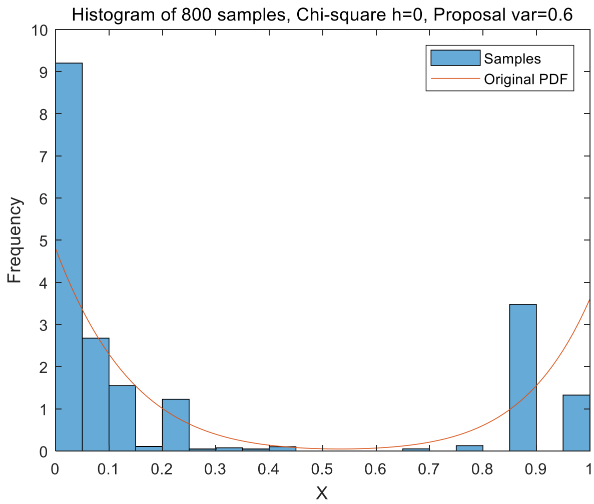
The first plot shows density of X. The histogram shows distribution of 3000 samples generating by Metropolis–Hastings algorithm using proposal density of normal with variance 0.6. Initial point is randomly selected.





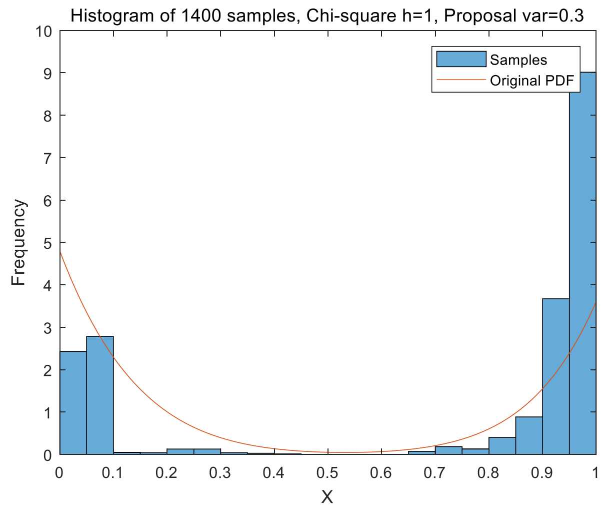
Applying Chi-Square goodness test, the result is h=0, p=0.180292 which show the samples conform to original density approximately. By comparing the chi-square test of 600 samples and 800 samples, we can see that the algorithm converges to its equilibrium distribution after generating 800 samples.

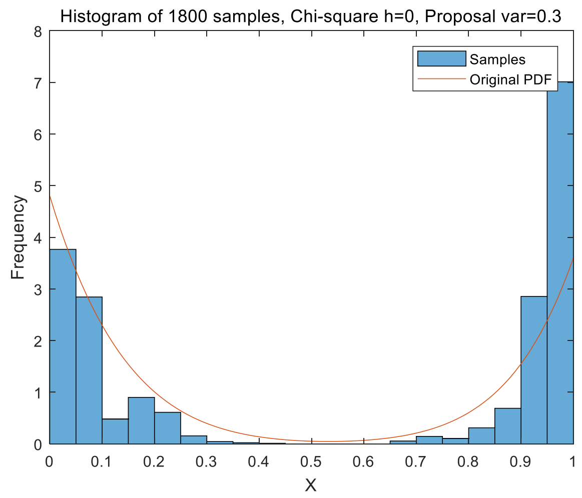




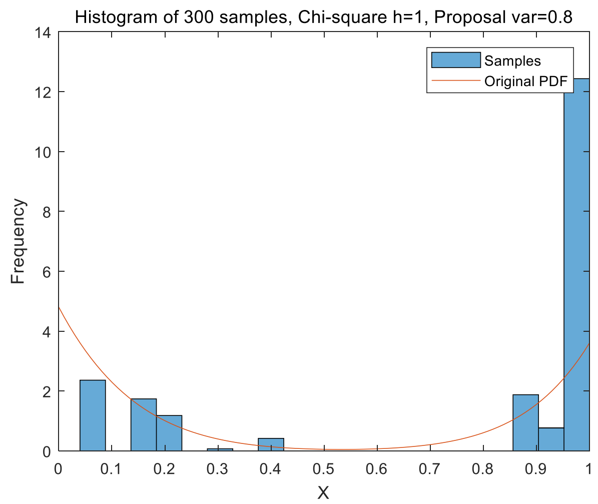
The following shows the sample paths of different proposal pdfs (normal with different variances).

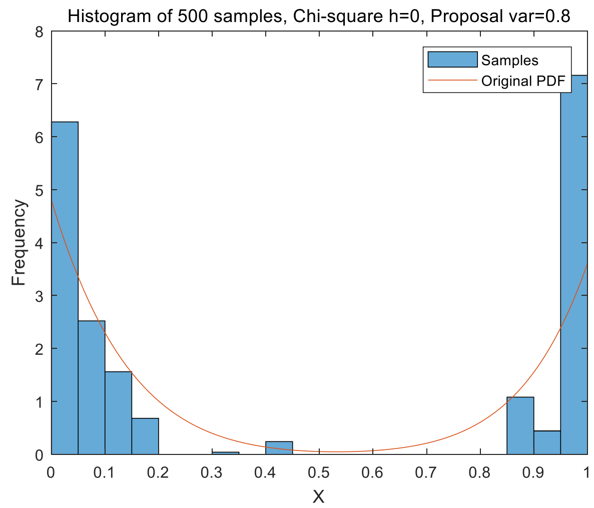
Proposal pdf = normal with variance 0.3, converges after 1800 samples





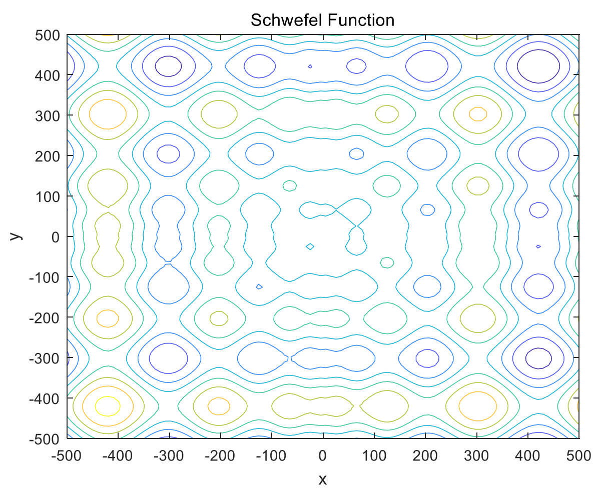
Proposal pdf = normal with variance 0.8, converges after 500 samples.

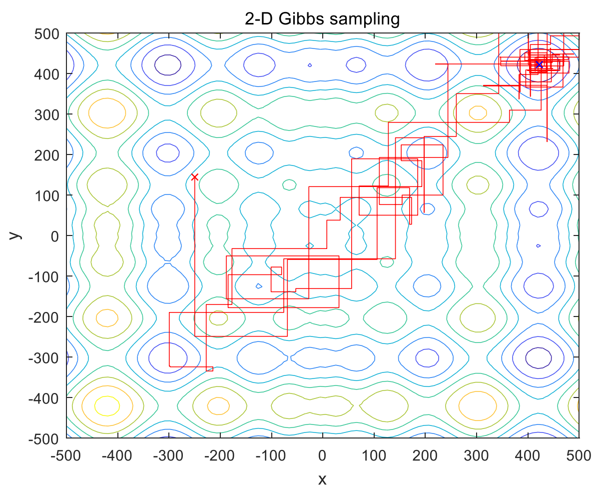




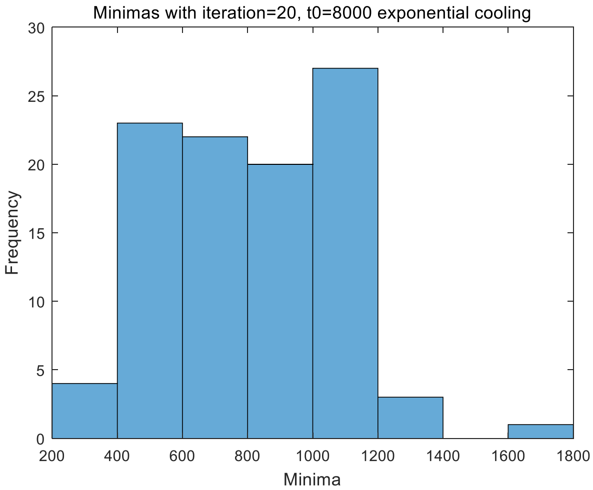
By comparing these plots, we can find with high-variance, the algorithm can converge to equilibrium distribution more quickly; with low-variance, it need more samples (iterations) to achieve equilibrium distribution. E.g. in the case of 0.3, the algorithm need about 1800 iterations to converge. In the case of 0.8, only 500 samples can achieve equilibrium distribution.

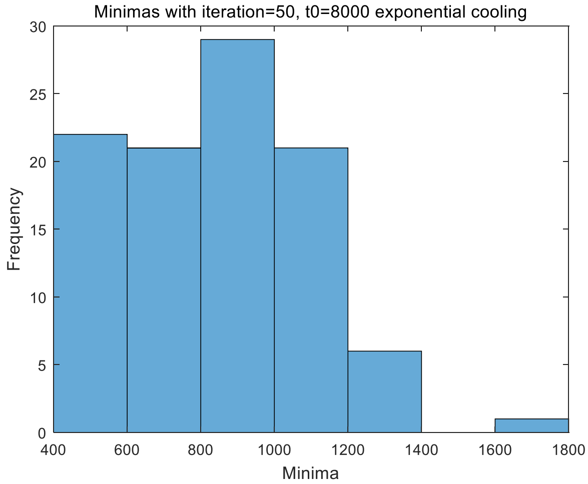
[MCMC for Optimization]

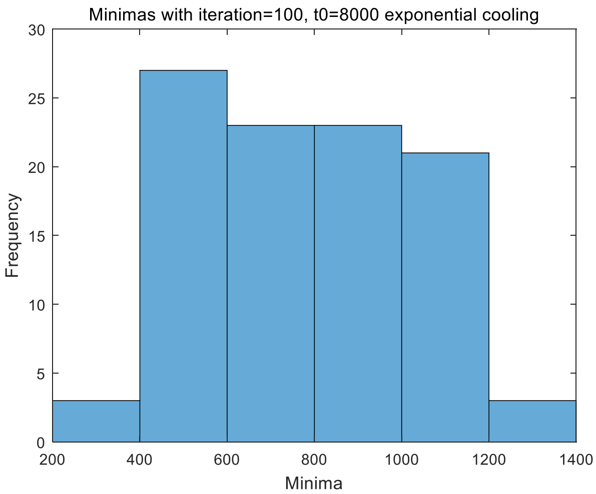


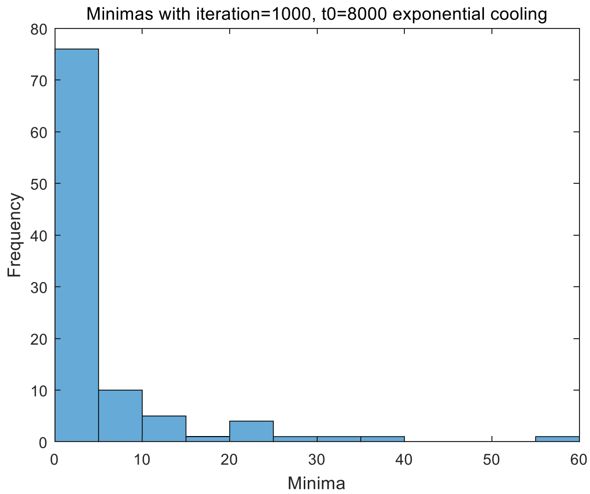


Exponential cooling schedule with initial temperature 8000, iterations = {20, 50, 100, 100} respectively.

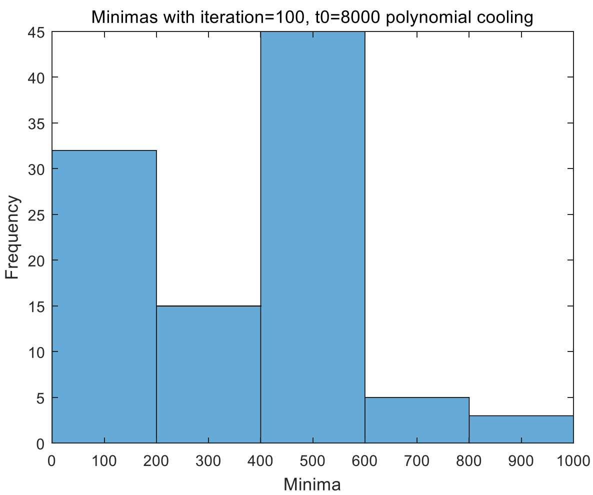


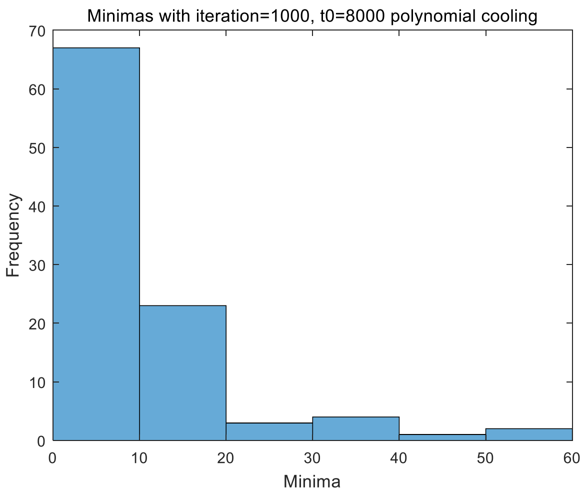






Polynomial cooling schedule with initial temperature 8000, iterations = {100, 1000} respectively.





The best sample paths (below 3D histogram) are achieved by using **logarithmic cooling schedule**.

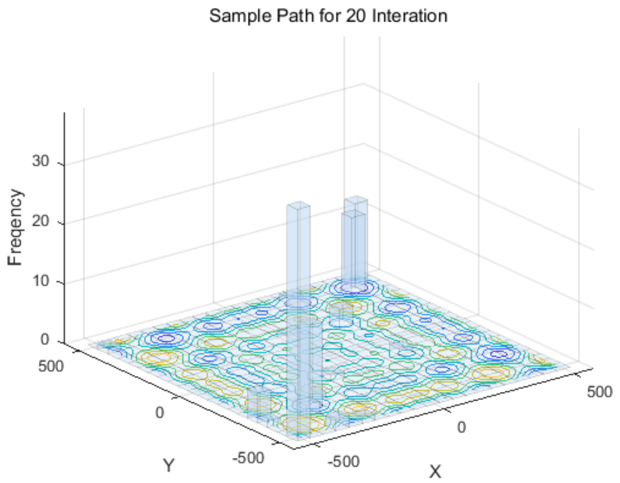
Parameter of Algorithm

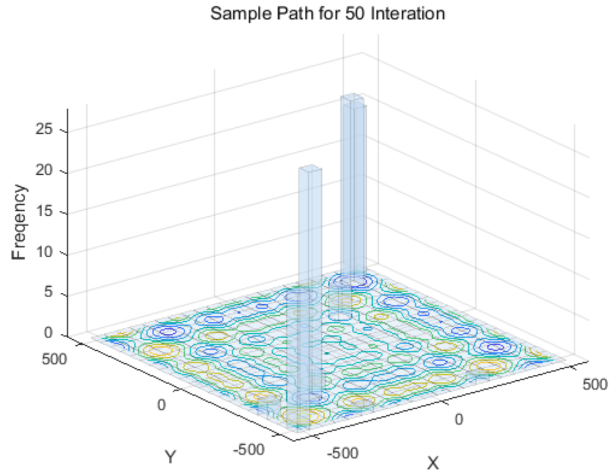
Initial temperature (T_0): 8000

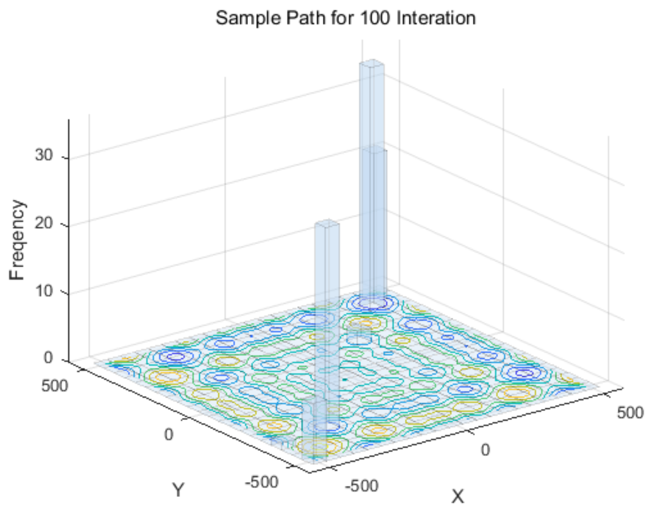
Proposal pdf: normal with variance 400

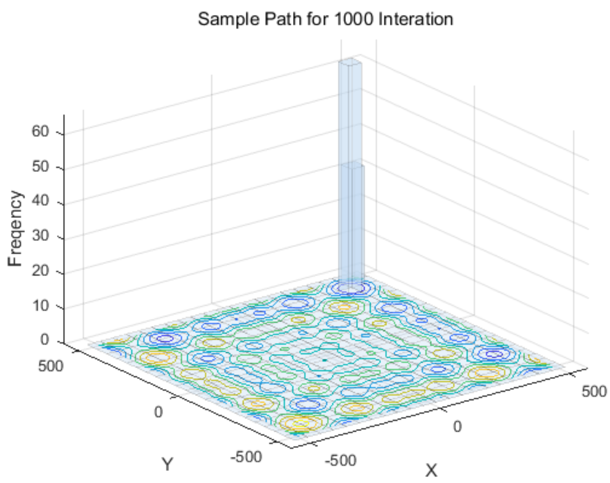
Cooling schedule: $T_k = T_0/(1+80*\log(k))$

Iterations: 20 50 100 1000



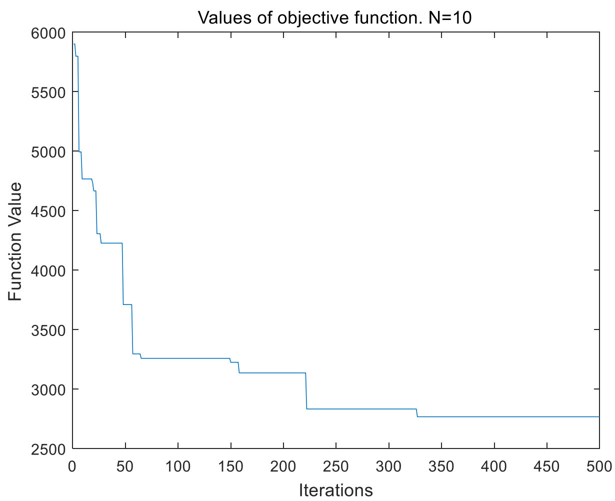


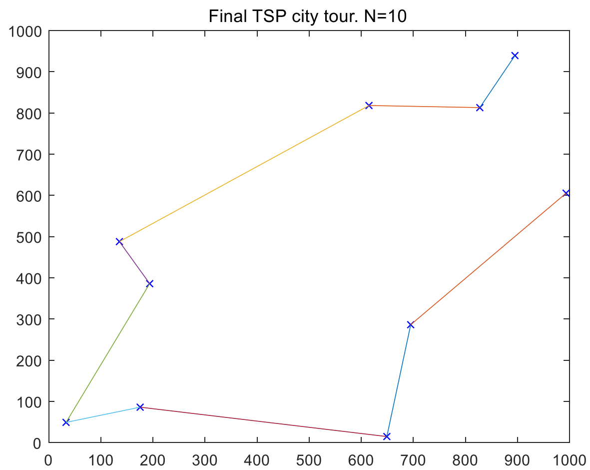




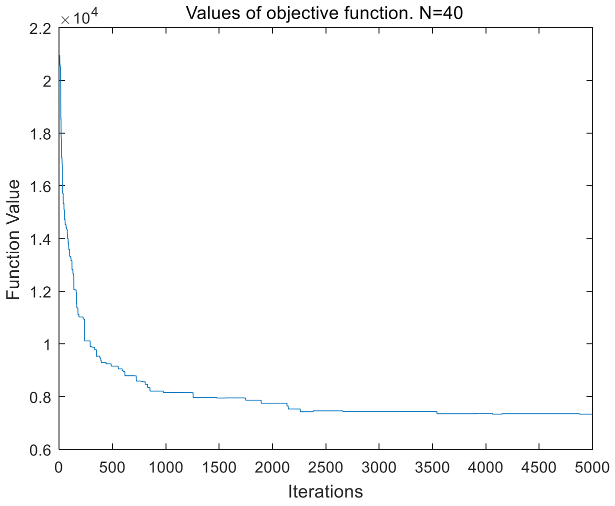
[Optimal Paths]

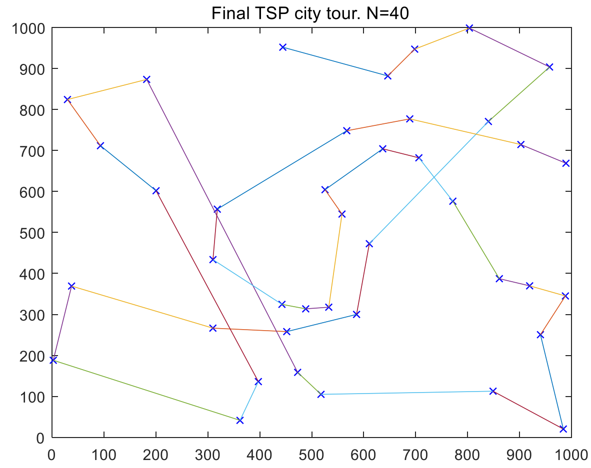
For N=10, the below plot shows the values of objective function in each step and the final TSP city tour. My TSP algorithm uses logarithmic cooling scheme with initial temperature 6000. We can see the value of objective function converges to its minima after 300 iterations.



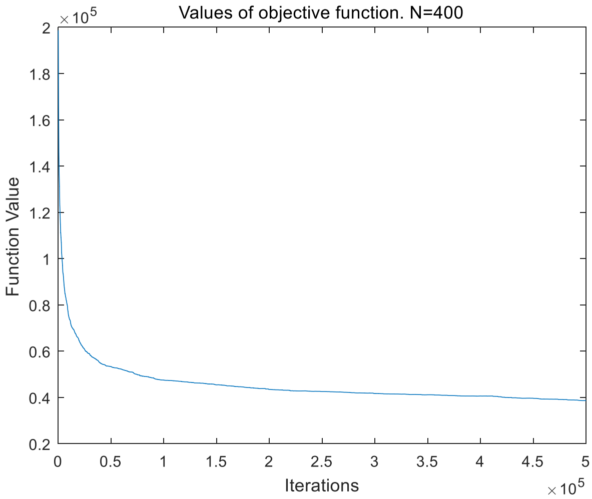


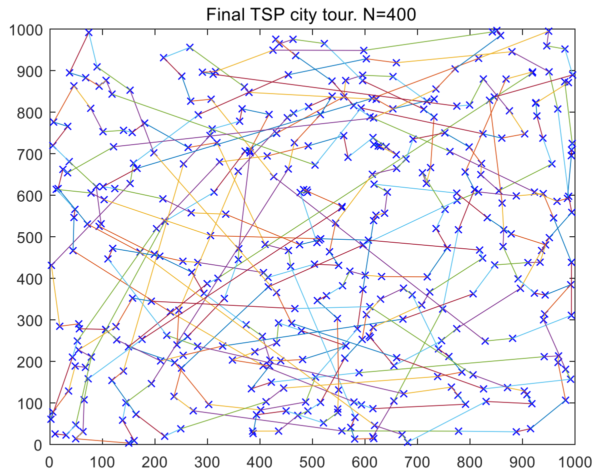
For N=40, under the same cooling scheme, the speed of converge are much slower than N=10.





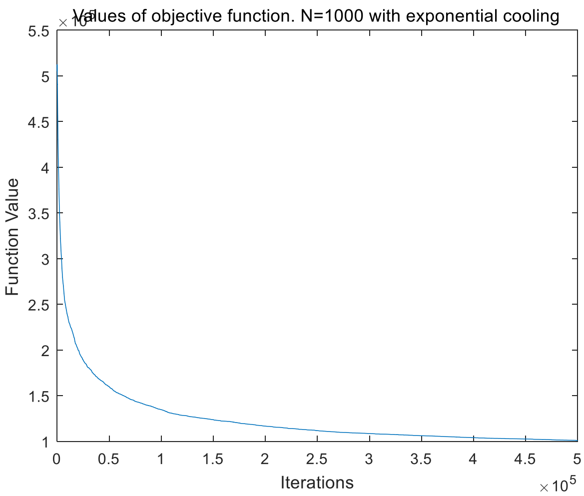
For N=400, speed of converge is very slow. Even after 500000 iterations, it seems not converge to a steady value.

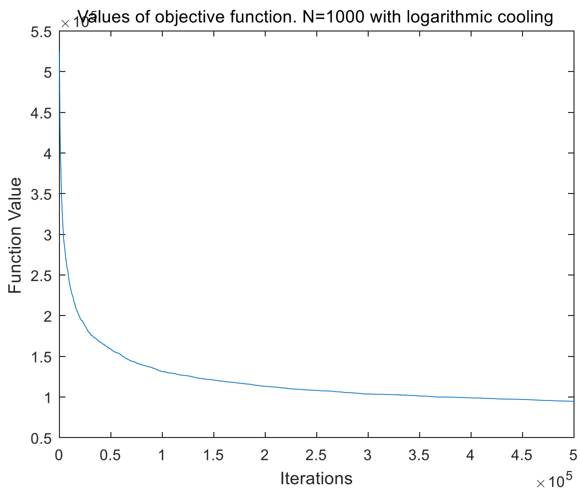




Compare of different cooling schedule

For N=1000, keeping iteration = 500000 unchanged, I tried both exponential (left plot) and logarithmic (right plot) cooling schedule. By comparing the plot, we can see speed of convergence of logarithmic cooling is faster. During the same number of iterations, logarithmic can find smaller value which is less than $1*10^5$. The explanation I given is that exponential cooling is too fast decrease to 0 so that the algorithm has less possibility to explore more new permutations.





```matlab
%EE511 Project 5
%Author: Yong Wang <yongw@usc.edu>
x0=linspace(0,1);
fx = 0.6.*(1-x0).^7./beta(1,8) + 0.4.*x0.^8./beta(9,1);
figure;
plot(x0,fx);
rng(1);
k=800;
x=zeros(k,1);
v=.6;
x(1)=rand; %the intial point
for t=1:k-1
    %generate from normal(xt,1)
    xp = x(t) + v*randn;
    if xp < 0 || (xp > 1)
        fp = 0;
    else
        fp = 0.6.*(1-xp).^7./beta(1,8) + 0.4.*xp.^8./beta(9,1);
    end
    f = 0.6.*(1-x(t)).^7./beta(1,8) + 0.4.*x(t).^8./beta(9,1);
    a = fp*normpdf(x(t),xp)/f*normpdf(xp,x(t));
    if a > 1
        a = 1;
    end
    if a >= rand
        %accept
        x(t+1) = xp;
    else
        %reject
        x(t+1) = x(t);
    end
end
figure;
nbin=20;
h = histogram(x, nbin, 'Normalization','pdf');
bins = h.BinEdges(:,1:nbin);
vals = h.Values;
E = 0.6.*(1-bins).^7./beta(1,8) + 0.4.*bins.^8./beta(9,1);
hold on;
plot(x0,fx);
%chi-square test
[h2,p,st] = chi2gof(bins,'Edges',h.BinEdges,'Frequency',vals,'Expected',E);
fprintf('h=%f  p=%f \n', h2, p);
chi = sum((vals-E).^2./E);
xlabel('X');
ylabel('Frequency');
title(sprintf('Histogram of %d samples, Chi-square h=%d, Proposal var=%.1f',k,h2,v));
legend('Samples', 'Original PDF');
```

```
%% Schwefel Function
x0=linspace(-500,500);
y0=linspace(-500,500);
[X,Y] = meshgrid(x0,y0);
Z = 418.9829*2 - X.*sin(sqrt(abs(X))) - Y.*sin(sqrt(abs(Y)));
contour(x0,y0,Z);
xlabel('x');
ylabel('y');
title('Sample Path');

iter = 1000;
minima=zeros(100,1);
location=zeros(100,2);
t0=8000;
for j=1:100
    x=zeros(iter+1,2);
    %initial point
    x(1,:) = -500+1000*rand(1,2);
    for i=1:iter
        %cooling schedule
        t = t0*0.99^(i-1);  %exponential
        t = t0/i; %polynomial
        %t = t0/(1+80*log(i)); %logarithmic
        v = 400;
        xp = x(i,:);
        if(mod(i,2) == 0)
            %generate new x
            xp(1) = xp(2)+v*randn;
        else
            %generate new y
            xp(2) = xp(1)+v*randn;
        end
        if(abs(xp(1))>500)
            xp(1)=sign(xp(1))*500;
        end
        if(abs(xp(2))>500)
            xp(2)=sign(xp(2))*500;
        end
        z=418.9829*2 - x(i,1)*sin(sqrt(abs(x(i,1)))) - x(i,2)*sin(sqrt(abs(x(i,2))));
        zp=418.9829*2 - xp(1)*sin(sqrt(abs(xp(1)))) - xp(2)*sin(sqrt(abs(xp(2))));

        a = exp(-(zp-z)/t);
        if  zp < z ||  rand < a
            x(i+1,:)=xp;
            m=zp;
        else
            x(i+1,:)=x(i);
            m=z;
        end
```

```matlab
    end
    minima(j) = m;
    location(j,:) = x(i,:);
end
figure;
hist3(location,'Ctrs',{-500:50:500 -500:50:500}, 'FaceAlpha',.35,'EdgeAlpha',.15);
hold on;
contour(x0,y0,Z);
title(sprintf('Sample Path for %d Interation',iter));
zlabel('Freqency');
xlabel('X');
ylabel('Y');

%plot(x(:,1),x(:,2),'r');
%plot(x(iter+1,1),x(iter+1,2),'bx');
figure;
histogram(minima);
xlabel('Minima');
ylabel('Frequency');
title(sprintf('Minimas with iteration=%d, t0=%d polynomial cooling',iter,t0));




%% [Optimal Paths]
ncity=1000;
N=[1000*rand(ncity,1) 1000*rand(ncity,1)];
oriN=N;
t0=6000;iter=500000;

%initial order
order=randperm(ncity);
oldN(1:ncity,:) = N([order],:);
D=zeros(1,iter);
for i=1:iter
    %cooling schedule
    t = t0*0.99^(i-1);  %exponential
    %t = t0/i; %polynomial
    t = t0/(1+80*log(i)); %logarithmic
    %randomly pick up and swap two cities
    idx = randi(ncity, 1, 2);
    newN = oldN;
    newN(idx(1),:) = oldN(idx(2),:);
    newN(idx(2),:) = oldN(idx(1),:);

    tsp=sum(sqrt(sum((diff(oldN).^2),2)));
    tsp_new=sum(sqrt(sum((diff(newN).^2),2)));
    a = exp(-(tsp_new-tsp)/t);
    if  tsp_new < tsp ||  rand < a
```

```
        oldN=newN;
        D(i)=tsp_new;
    else
        D(i)=tsp;
    end
end

figure;
plot(N(:,1),N(:,2),'bx');
hold on;
for i=1:ncity-1
    plot(oldN(i:i+1,1),oldN(i:i+1,2));
end
axis([0 1000 0 1000]);
title(sprintf('Final TSP city tour. N=%d',ncity));

hold off;
figure;
plot([1:iter],D);
title(sprintf('Values of objective function. N=%d with logarithmic cooling',ncity));
xlabel('Iterations');
ylabel('Function Value');
```