% Course: Nonlinear Optimization. %

% FALL.2018. Dr. Cheng. %

% Assignment: (3) %

% Date:(2018.9.26) %

% By: 卢博 %

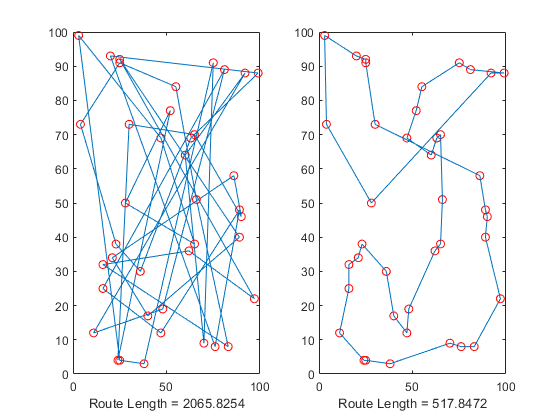
% ID NUMBER: 11849159 %

% LAB:(3.1) %

% Description: This code uses the nor\_path and min\_path function to find a shorter way.

1. Travelling Salesman Problem

clear;clc  
dots = [randi([0 100], 40, 1), randi([0 100], 40, 1)];  
subplot(1,2,1)  
nor\_path(dots) % 普通连接  
subplot(1,2,2)  
min\_path(dots) % 优化连接



画出数据点 1

连接点 1

最后一根线 2

function nor\_path(dots)

## 画出数据点

axis([0 100 0 100])  
n = length(dots); % n个点  
plot(dots(:,1), dots(:,2), 'bo');  
hold on  
distance = 0; % 总距离此时为0

## 连接点

for i = 1:n-1 % 起点到终点画n-1条线  
 x1 = dots(i,1);  
 y1 = dots(i,2);  
 x2 = dots(i+1,1);  
 y2 = dots(i+1,2);  
 distance = distance + norm([x1 y1] - [x2 y2]);  
 plot(x1,y1,'ro');  
 plot(x2,y2,'ro');  
 line([x1 x2],[y1 y2]); % 打点划线  
end

## 最后一根线

x0 = dots(1,1);  
y0 = dots(1,2);  
x40 = dots(40,1);  
y40 = dots(40,2);  
line([x0 x40],[y0 y40]) % 连接终点和起始点  
xlabel(['Route Length = ',num2str(distance)]) % 显示最后总距离

end

画出数据点 1

计算不同点之间距离 并用排列组合的方式组成新矩阵 1

对每个点找到距离最近的点 连接 并且去掉这些数据 同时将"头"移动给下一个点 1

最后一根线 3

function min\_path(dots)

## 画出数据点

axis([0 100 0 100])  
n = length(dots); % n个点  
plot(dots(:,1), dots(:,2), 'bo');  
hold on

## 计算不同点之间距离 并用排列组合的方式组成新矩阵

distance = 0; % 总距离此时为0  
D = pdist(dots); % 计算不同点之间距离  
dots\_D = [nchoosek(1:n,2) D']; % 排列组合得到新矩阵 ex:[1 25 3.677] 第一个点和第25个点之间距离为3.677

## 对每个点找到距离最近的点 连接 并且去掉这些数据 同时将"头"移动给下一个点

temp = 1; % "头"设置为第一个点x1 = dots(1,1) y1 = dots(1,2)  
for i = 1:n-1 % 起点到终点画n-1条线  
 index = find(dots\_D(:,1) == temp | dots\_D(:,2) == temp);  
 temp\_dots\_D = dots\_D(index,:); % 找到所有的距"头"点的距离 包括它到别的点和别的点到它  
 mind = min(temp\_dots\_D(:,3)); % 在这些距离中找到最小值  
 id = find(temp\_dots\_D(:,3)==mind); % 得到最小值的引索 即其他点到"头"点最小距离的组合方式 的引索  
 k = id(1); % 有可能有多个点到"头"点距离都最小 只取一个  
 x1 = dots(temp\_dots\_D(k,1),1);  
 y1 = dots(temp\_dots\_D(k,1),2);  
 x2 = dots(temp\_dots\_D(k,2),1);  
 y2 = dots(temp\_dots\_D(k,2),2);  
 %{  
 这里k是行引索 temp\_dots\_D(k,:)得到的是[1 20 3.02]  
 即第一个点到第二十个点的距离为3.02 其中3.02是所有其他点到第一个点最小的距离  
 %}  
 distance = distance + norm([x1 y1] - [x2 y2]);  
 plot(x1,y1,'ro');  
 plot(x2,y2,'ro');  
 line([x1 x2],[y1 y2]); % 打点划线  
 if temp\_dots\_D(k,1) == temp  
 temp = temp\_dots\_D(k,2);  
 elseif temp\_dots\_D(k,2) == temp  
 temp = temp\_dots\_D(k,1);  
 end  
 %{  
 这个判断语句是为了传递"头"点,比如[1 20 3.02]  
 如果现在的,已经被计算了的"头"点是第一个点,则"头"点传为第二十个点  
 如果"头"点是第二十个点,则"头"点传为第一个点.  
 %}  
 dots\_D(index,:) = []; % 清楚已经被计算了的"头"点所有距离信息，不论是它到别的点还是别的点到它  
end

## 最后一根线

x0 = dots(1,1);  
y0 = dots(1,2);  
x40 = dots(temp,1);  
y40 = dots(temp,2);  
line([x0 x40],[y0 y40]) % 连接终点和起始点  
xlabel(['Route Length = ',num2str(distance)]) % 显示最后总距离

end

[*Published with MATLAB® R2018a*](https://www.mathworks.com/products/matlab)

% Course: Nonlinear Optimization. %

% FALL.2018. Dr. Cheng. %

% Assignment: (3) %

% Date:(2018.9.26) %

% By: 卢博 %

% ID NUMBER: 11849159 %

% LAB:(3.2) %

% Description: This code uses the fit, opfit and con\_size function to find a better way to pack

1. Bin Packing Problem

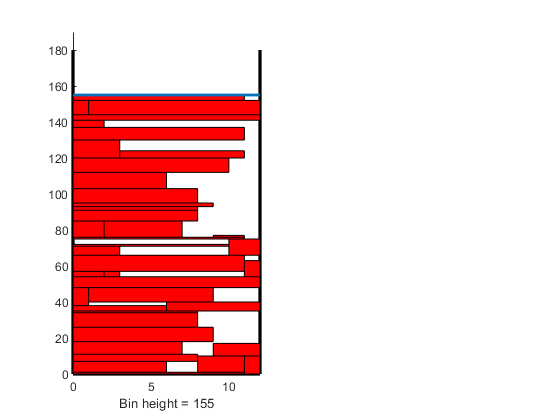
优化前 1

优化后 2

clear;clc  
W = 12;  
rects = [randi([1 12], 40, 1), randi([1 10], 40, 1)]; % 随机生成矩形  
figure

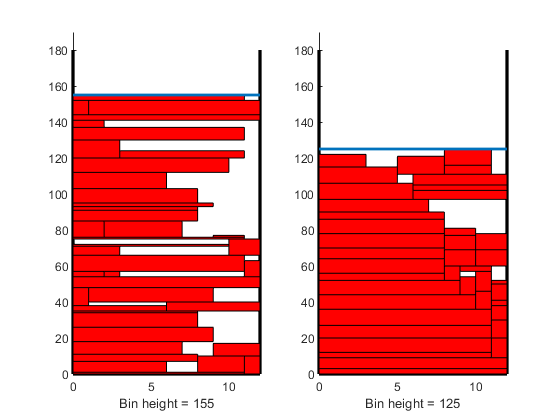
## 优化前

subplot(1,2,1)  
low\_level = zeros(1,12); % 初始化水平高度  
axis([0 12 0 190])  
line([0 12],[0 0],'color','black','linewidth',2.5)  
line([0 0],[0 180],'color','black','linewidth',2.5)  
line([12 12],[0 180],'color','black','linewidth',2.5) % 画出边框  
hold on  
  
for i = 1:length(rects) % 每一行表示一个矩形  
 out\_level = fit(low\_level,rects(i,:));  
 low\_level = out\_level;  
end  
line([0 12], [max(low\_level), max(low\_level)], 'LineWidth', 2)  
hold off  
xlabel(['Bin height = ',num2str(max(low\_level))]) % 显示最后水平高度的最大值



## 优化后

subplot(1,2,2)  
low\_level = zeros(1,12); % 初始化水平高度  
axis([0 12 0 190])  
line([0 12],[0 0],'color','black','linewidth',2.5)  
line([0 0],[0 180],'color','black','linewidth',2.5)  
line([12 12],[0 180],'color','black','linewidth',2.5) % 画出边框  
hold on  
  
low\_level = opfit(low\_level,rects);  
line([0 12], [max(low\_level), max(low\_level)], 'LineWidth', 2)  
hold off  
xlabel(['Bin height = ',num2str(max(low\_level))]) % 显示最后水平高度的最大值



当连续最小高度的长 大于等于 即将进入的矩形的水平长度 1

当连续最小高度的长 小于 即将进入的矩形的水平长度 2

% 优化前的放置矩形的函数  
function [out\_level,out\_index,temp] = fit(low\_level,rect)

x = rect(1);  
y = rect(2);  
A = low\_level;  
index = find(A == min(A));  
[max\_len, max\_index] = con\_size(index); % 找到此时水平高度里 最长的连续最小高度 的长度和引索  
new\_index = index(max\_index(1):max\_index(1)+max\_len-1); % max\_index最长的连续最小高度可能有多个，取其中一个

## 当连续最小高度的长 大于等于 即将进入的矩形的水平长度

if max\_len >= x

out\_index = new\_index(1):new\_index(1)+x-1; % 只输出 进入的矩形的长度对应的引索  
 temp = A(out\_index);  
 for k = 1:x  
 temp(k) = temp(k) + y; % 提高水平高度  
 end  
 X = [new\_index(1)-1 x+new\_index(1)-1 x+new\_index(1)-1 new\_index(1)-1];  
 Y = [min(A) min(A) y+min(A) y+min(A)];  
 fill(X,Y,'r'); % 填充矩形为红色  
 A(out\_index) = temp;

## 当连续最小高度的长 小于 即将进入的矩形的水平长度

else  
 temp\_A = A(index); % 找到所有的最小高度  
 A(index) = 1000; % 把这些高度赋值为1000  
 A(index) = min(A); % 再次找最小高度，这样就找到了原本高度里第二小的值，并将原来的最小高度赋值为第二小的高度  
 low\_level = A; % 用新的水平高度再次尝试  
 [A,out\_in,tmp] = fit(low\_level,rect);  
 out\_index = out\_in; % 当结果可以输出时把每次矩形长度对应的引索记录下来  
 temp = tmp;  
 A(index) = temp\_A; % 将原本的最小高度复原，但是会吧填充了矩形的区域也复原  
 A(out\_in) = tmp; % 因此这一步就是把矩形区域对应的高度提高到至应高度  
end  
out\_level = A;

end

筛选 1

排序 1

放入矩形 2

% 优化后的放置矩形的函数  
function out\_level = opfit(low\_level,rects)

roy = [];  
idx = [];

## 筛选

矩形中高度小于2的旋转，然后把矩形中长度小于2的挑出来

for i = 1:length(rects)  
 if rects(i,2) <= 2  
 temp = rects(i,1);  
 rects(i,1) = rects(i,2);  
 rects(i,2) = temp;  
 end  
 if rects(i,1) <= 2  
 roy = [roy;rects(i,:)];  
 idx = [idx i];  
 end  
end

## 排序

rects(idx,:) = []; % 把长度小于2的矩形去掉然后排倒序  
a = rects;  
b=a(:,1);  
[c,pos]=sort(b);%pos为排序后的下标,c为第一行的排序结果;  
p = [c,pos];  
X=eye(length(p));  
X=rot90(X);  
p=X\*p;  
a(:,2)=a(p(:,2),2);%第二行按照第一行排序的下标对应  
a(:,1)=p(:,1);  
  
aa = roy; % 长度小于2的矩形排正序  
b=aa(:,1);  
[c,pos]=sort(b);%pos为排序后的下标,c为第一行的排序结果;  
aa(:,2)=aa(pos,2);%第二行按照第一行排序的下标对应  
aa(:,1)=c;

## 放入矩形

rects = [a(1:round(length(a)/2),:);aa;a(round(length(a)/2)+1:end,:)]; % 把长度小于2的矩形放在倒序的矩形中间  
for i = 1:length(rects)  
 out\_level = fit(low\_level,rects(i,:));  
 low\_level = out\_level;  
end

end

函数con\_size()

function [len,index] = con\_size(x)  
% 这里的x是引索,比如找到一个数组里面最小值的引索  
% x = find(A == min(A));  
% x = [1 2 3 6 7 8 9 10 22 35]  
% con\_size(x)来判断哪些是连续的，并且得到最长的连续引索[6 7 8 9 10]  
d = diff(x(:))~=1; % 得到每两个数之间是否连续的逻辑数组  
k = find([true;d;true]); % 在d前后补上1  
r = [k(1:end-1) diff(k)]; % 得到每个引索对应连续个数  
len = max(r(:,2)); % 最大连续长度  
idx = find(r(:,2) == len);  
index = r(idx,1);  
end

[*Published with MATLAB® R2018a*](https://www.mathworks.com/products/matlab)