1.

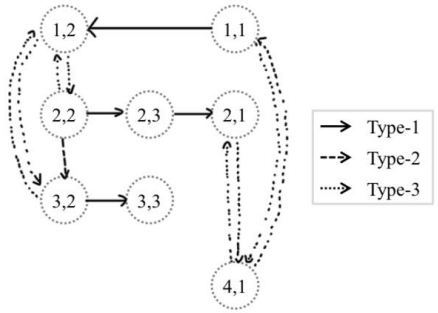
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from PIL import Image
from skimage import measure
import math
import random

count=0
tmax=100000
tnow=tmax
tmin=1
r=0.95
message=17
tao=0.001
vnow=208.55999994277954
vbest=vnow
vbestforever=vnow
C=np.array([0.52,0.6,0.52,0.6,0.52,0.6,0.92,0.52,0.6,0.68,0.52,0.76,0.52,0.52,0.68,0.52,0.52])
T=np.array([50,5,5,5,5,5,10,10,10,10,50,100,100,100,1000,1000,1000])
P=np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16])
B=np.zeros([message])
Q=np.zeros([message])
R=np.zeros([message])
newP=np.zeros([message])

def change(C,T,P,m,n):#找鄰居的解，用兩值互換的方法
    tmp=C[m]
    C[m]=C[n]
    C[n]=tmp
    tmp=T[m]
    T[m]=T[n]
    T[n]=tmp
    tmp=P[m]
    P[m]=P[n]
    P[n]=tmp
    return 0

def summary(R):
    add=0
    for k in range(message):
        add=add+(R[k])
    return add

def objective(C,T,P,tmax,tmin,r,message,tao):#算此順序下的總和
    k=0
    for k in range(message):
        B[k]=C[k]
    i=0
    k=0
    for k in range(message):
        for i in range(k,message,1):
            if C[i]>B[k]:
                B[k]=C[i]
    k=0
    j=0
    flag=1
    tmp=np.zeros([message])
    rhs=np.zeros([message])
    for k in range(message):
        flag=1
        Q[k]=B[k]
        while flag==1:
            j=0
            tmp[k]=0
            for j in range(k):
                tmp[k]=tmp[k]+math.ceil((Q[k]+tao)/T[j])*C[j]
            rhs[k]=B[k]+tmp[k]
            if rhs[k]+C[k]>T[k]:
                R[k]=1500
                break
            elif rhs[k]+C[k]<=T[k]:
                if Q[k]==rhs[k]:
                    R[k]=Q[k]+C[k]
                    flag=0
                elif Q[k]!=rhs[k]:
                    Q[k]=rhs[k]
    u=summary(R)
    return u

print(objective(C,T,P,tmax,tmin,r,message,tao))  #印初始的總和208.56

evetime_value = []
num=0
while True:
    if tnow<=tmin:
        break
    for i in range(15):
        num+=1
        m=random.randint(0, 8)
        n=random.randint(9, 16)
        change(C,T,P,m,n)
        vnow=objective(C,T,P,tmax,tmin,r,message,tao)
        diff=vnow-vbest
        if(vnow<vbestforever):#將出現過最小的總和值存入vbestforever
            vbestforever=vnow
        if diff<=0:
            #count=count+1
            #print(objective(C,T,P,tmax,tmin,r,message,tao),count)
            vbest=vnow
            print(objective(C,T,P,tmax,tmin,r,message,tao),1)
            evetime_value.append(vnow)
        elif diff>0:
            prob =math.exp(-diff/tnow)
            randum=random.uniform(0,1)
            if randum<prob:
                vbest=vnow
                print(objective(C,T,P,tmax,tmin,r,message,tao),2)
                evetime_value.append(vnow)
            elif randum>=prob:
                change(C,T,P,m,n)
                vnow=objective(C,T,P,tmax,tmin,r,message,tao)
    tnow=tnow*r
print(objective(C,T,P,tmax,tmin,r,message,tao),3)
for i in range(17):
    print("message",i,"的priority為:",P[i])
print("最佳的summation of the worst-case response time:",vbestforever)
print(num)
plt.figure(figsize = (10,6))
plt.xlabel("Iteration",fontsize = 15)
plt.ylabel("value",fontsize = 15)
plt.plot(evetime_value,linewidth = 1, label = "smallest value ever", color = 'r')
plt.legend()
plt.show()
```

```
Help  Variable Explorer  Plots  Files

Console 1/A

message 0 的priority為: 5
message 1 的priority為: 7
message 2 的priority為: 3
message 3 的priority為: 1
message 4 的priority為: 4
message 5 的priority為: 2
message 6 的priority為: 8
message 7 的priority為: 10
message 8 的priority為: 6
message 9 的priority為: 9
message 10 的priority為: 16
message 11 的priority為: 13
message 12 的priority為: 12
message 13 的priority為: 14
message 14 的priority為: 0
message 15 的priority為: 15
message 16 的priority為: 11
最佳的summation of the worst-case response time: 204.67999999999998
```

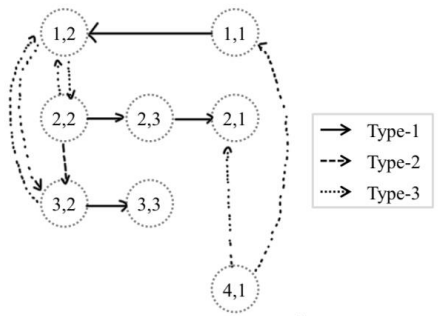
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from PIL import Image
from skimage import measure
import math
import random
count=0
tmax=100000
tnow=tmax
tmin=1
r=0.95
message=17
tao=0.001
vnow=208.55999994277954
vbest=vnow
vbestforever=vnow
C=np.array([0.52,0.6,0.52,0.6,0.52,0.6,0.92,0.52,0.6,
0.68,0.52,0.76,0.52,0.52,0.68,0.52,0.52])
T=np.array([50,5,5,5,5,5,10,10,10,10,50,100,100,100
,1000,1000,1000])
P=np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16])
B=np.zeros([message])
Q=np.zeros([message])
R=np.zeros([message])
newP=np.zeros([message])

def change(C,T,P,m,n):#找鄰居的解，用兩值互換
的方法
    tmp=C[m]
    C[m]=C[n]
    C[n]=tmp
    tmp=T[m]
    T[m]=T[n]
    T[n]=tmp
    tmp=P[m]
    P[m]=P[n]
    P[n]=tmp
    return 0

def summary(R):
    add=0
    for k in range(message):
        add=add+(R[k])
    return add

def objective(C,T,P,tmax,tmin,r,message,tao):#算此
順序下的總和
    k=0
    for k in range(message):
        B[k]=C[k]
    i=0
    k=0
    for k in range(message):
        for i in range(k,message,1):
            if C[i]>B[k]:
                B[k]=C[i]
    k=0
    j=0
    flag=1
    tmp=np.zeros([message])
    rhs=np.zeros([message])
    for k in range(message):
        flag=1
        Q[k]=B[k]
        while flag==1:
            j=0
            tmp[k]=0
            for j in range(k):
tmp[k]=tmp[k]+math.ceil((Q[k]+tao)/T[j])*C[j]
            rhs[k]=B[k]+tmp[k]
            if rhs[k]+C[k]>T[k]:
                R[k]=1500
                break
            elif rhs[k]+C[k]<=T[k]:
                if Q[k]==rhs[k]:
                    R[k]=Q[k]+C[k]
                    flag=0
                elif Q[k]!=rhs[k]:
                    Q[k]=rhs[k]
    u=summary(R)
    return u

print(objective(C,T,P,tmax,tmin,r,message,tao)) #印
初始的總和208.56

evetime_value = []
num=0
while True:
    if tnow<=tmin:
        break
    for i in range(15):
        num+=1
        m=random.randint(0, 8)
        n=random.randint(9, 16)
        change(C,T,P,m,n)
        vnow=objective(C,T,P,tmax,tmin,r,message,tao)
        diff=vnow-vbest
        if(vnow<vbestforever):#將出現過最小的總和
值存入vbestforever
            vbestforever=vnow
        if diff<=0:
            #count=count+1
#print(objective(C,T,P,tmax,tmin,r,message,tao),cou
nt)
            vbest=vnow
print(objective(C,T,P,tmax,tmin,r,message,tao),1)
            evetime_value.append(vnow)
        elif diff>0:
            prob =math.exp(-diff/tnow)
            randum=random.uniform(0,1)
            if randum<prob:
                vbest=vnow
print(objective(C,T,P,tmax,tmin,r,message,tao),2)
                evetime_value.append(vnow)
            elif randum>=prob:
                change(C,T,P,m,n)
vnow=objective(C,T,P,tmax,tmin,r,message,tao)
    tnow=tnow*r
print(objective(C,T,P,tmax,tmin,r,message,tao),3)
for i in range(17):
    print("message",i,"的priority為:",P[i])
print("最佳的summation of the worst-case response
time:",vbestforever)
print(num)
plt.figure(figsize = (10,6))
plt.xlabel("Iteration",fontsize = 15)
plt.ylabel("value",fontsize = 15)
plt.plot(evetime_value,linewidth = 1, label =
"smallest value ever", color = 'r')
plt.legend()
plt.show()
```

2.

1. (8pts) Given the scenario in the figure above, follow the legend and draw the corresponding timing conflict graph.



2. (12pts) Following 1., given that Vehicle 4 enters Conflict Zone 1 before Vehicles 1 and 2, find a DEADLOCK solution which has no cycle in the corresponding timing conflict graph. Follow the legend and draw the corresponding timing conflict graph. Explain why there is a deadlock.



當vehicle 1在zone 1, vehicle 2在zone 3, vehicle 3在zone 2時,
則vehicle 1須等vehicle 3離開zone 2, vehicle 2須等vehicle 1
離開zone 1, vehicle 3須等vehicle 2離開zone 3, 此時發生deadlock.

3.

因為只有一個conflict zone，所以當vehicle進入後便可以直接離開此zone給下一個vehicle使用，故不會出現在resource conflict graph中，同時此例也不會產生resource conflict graph，因此此時當原本的圖沒有cycle，則不會有deadlock，並且依照到達的先後順序通過即可

4.

可以只使用rule1,4，因為rule1是處理同一個vehicle(只包含type1時)，自己進入conflict zone的先後順序，即代表transition發生的先後順序，rule4是處理同一個conflict zone中，不同vehicle進入的先後順序(包含type2、type3)，如此一來，可以涵蓋出rule2,3,5的處理方式，因此好處是可以簡化只考慮2個rule

5.

1.如同大多車廠的看法，即使賓士有推出level3，我也認為level3中目前汽車與人的權責區分太模糊，除非能制定出一套固定的準則來定義，但由於真實情況的變化太多，所以應該很難制定完整的規範，且太過頻繁的控制權切換，也會有較差的使用者體驗，喪失原本自動駕駛的意義。

2.我認為level4要能真正使用需要能接受即使在被允許的環境下，仍有可能完全停止，而由於有特定的環境限制，故我認為已現今的狀況能夠達成，但另一個需要考量的是車廠是否願意為發生的事故負責。

3.我認為level 5是無法達成的，因為當假如路直接斷掉，則車輛必須停止無法使用自動駕駛，或像是當鏡頭可能被外界的東西蓋住，則若沒有人為去除也會無法使用。