Single-Machine Scheduling Problem by Tabu

Result :

approximate optimal solution (job sequence) : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 20, 19, 18]
fitness function value (total weighted tardiness) : 5292
parameter settings : iteration = 13 ; tabu list size = 3
trial-and-error :
iteration = 16 ; tabu list size = 7
➔ [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 14, 20, 18, 17, 16, 19] , 5516
iteration = 15 ; tabu list size = 6
➔ [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 20, 17, 19] , 5392
iteration = 14 ; tabu list size = 4
➔ [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 20, 18, 19] , 5376
iteration = 13 ; tabu list size = 3
→ [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 20, 19, 18] , 5292

Code :

```
jobs=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
processing_time=[10,10,13,4,9,4,8,15,7,1,9,3,15,9,11,6,5,14,18,3]
due_date=[50,38,49,12,20,105,73,45,6,64,15,6,92,43,78,21,15,50,150,99]
weights=[10,5,1,5,10,1,5,10,5,1,5,10,10,5,1,10,5,5,1,5]


def Find_max(a,b):
    if a >= b:
        return a
    else:
        return b


def P_sum(list2,num):
    P = 0;
    for i in range(num):
        P = P + processing_time[list2[i]-1];

    return P


def Scheduling(list1):
```

```
    T=0
    for i in range(len(jobs)):
        T = T + weights[list1[i]-1]*Find_max((P_sum(list1,i+1)-due_date[list1[i]-1]),0)

    return T


def interchange(listt,i,j):
    e=listt[i]
    listt.pop(i)
    listt.insert(j,e)


def Find_best_index(listt):
    min_T =listt[0]
    min_index = 0
    for i in range(len(listt)-1):
        if listt[i+1]<min_T:
            min_T = listt[i+1]
            min_index = i+1
    return min_index



iteration = 1
solution = []
solution.extend(jobs)
total_weight = Scheduling(solution);

tabu_list=[]
neighborhood = []
pairwise=[]
temp_T=[]
neighbor=[]

while iteration <= 13:    # 13
    neighborhood.clear()
    pairwise.clear()
    temp_T.clear()

    for i in range(len(jobs)-1):
        TABU = False
```

```
        for j in range(len(tabu_list)):
            if solution[i]==tabu_list[j][0] and solution[i+1]==tabu_list[j][1]:
                TABU = True
                break
            if solution[i]==tabu_list[j][1] and solution[i+1]==tabu_list[j][0]:
                TABU = True
                break

        if TABU == True:
            continue
        else:
           pairwise.append((solution[i],solution[i+1]))
           neighbor.clear()
           neighbor.extend(solution)
           interchange(neighbor,i,i+1)
           neighborhood.append(neighbor)
           #print("neighbor : {}".format(neighbor))
           temp_T.append(Scheduling(neighbor))

best_index = Find_best_index(temp_T)
print("best_temp_T : {}".format(temp_T[best_index])) ###

if len(tabu_list)<3:    #5
        tabu_list.append(pairwise[best_index])
else:
        tabu_list.pop(0)
        tabu_list.append(pairwise[best_index])

solution.clear()
solution.extend(neighborhood[best_index])
print("solution : {}".format(solution))

iteration = iteration+1
```