人力资源分配问题

问题描述

某项目包含 n 个子任务，记完成第 i 个子任务所需投入的工时为 D_i（人*天），任意时刻投入第i个子任务的人数都不得超过 M_i人，共有劳动力X人。$X, T \in R$.

1. 如何分配可使完成所有任务的总时间T（天）最短，T是多少？

2. 如果一些子任务不能在项目开始时立即开工，而是需要一定的预热时间才能开始投入人力，记第i个任务的预热时间为R_i，如何解决问题1？

Solution for question 1

思路一

尽量使每人每天都有活儿干即可。记t时刻分配到第i个任务的劳动力为$S_i(t)$，约束条件：$\sum_{i=1}^{n} S_i(t) \le X$ 且 $0 \le S_i(t) \le M_i$。令$\int_0^{T_i} S_i(t)dt = D_i$，则 $T = max(T_i)$ (i=1,...,n)

目标： min T

因为$X \in R$，在无M约束条件下，以所有子任务同时完工为目标，可使T最小，最小值为$T = \sum_{i=1}^{n} D_i / X$,显然 $S_i \equiv \frac{D_i}{\sum_{i=1}^{n} Di} X$；在M约束条件下，若$\frac{D_i}{\sum_{i=1}^{n} Di} X > M_i$，取$S_i = M_i$，最终完工时间为$T = max(\frac{D_i}{S_i})$.

```
clear;% This algorithm has O(n) running time
X=randi(30); D=ceil(abs(randn(1,25)*5+5)); M=randi([1,6],1,25);
S=D/sum(D)*X;%若不超过M
S(S>M)=M(S>M); %若超过M
T=max(D./S)
```

```
T = 141
```

```
%用规划求解器验证结果
d=size(D);
[~,~,T2]=fminimax(@(x) D./x,ones(d),ones(d),X,[],[],zeros(d),M)
```

```
Local minimum possible. Constraints satisfied.

fminimax stopped because the size of the current search direction is less than
twice the default value of the step size tolerance and constraints are
satisfied to within the default value of the constraint tolerance.

<stopping criteria details>
T2 = 141.0000
```

Solution for question 1

思路二 A greedy method

example: D=[6,1], M=[3,1], X=3

记t时刻分配到第i个任务的劳动力为$S_i(t)$，$\sum_{i=1}^{n} S_i(t) \le X$且$0 \le S_i(t) \le M_i$，对子任务i=1,...,n， 按升序排列 M_i 及其

相应的子任务i，记排列后的任务为i=1,...,n。排列后，从i=1到i=n依次分配劳力，每次都尽可能多的分配劳力给每个子任务。某子任务完成后，空闲出的劳力继续分配给后续任务。

At time t=0, find the first k s.t $X \le \sum_{i=1}^{k} M_i$, the first k sequences would then run first. for i=1,...,k-1 $S_i = M_i$, $S_k = X - \sum_{i=1}^{k-1} S_i$, at time $t = t + min(\frac{D_1}{S_1}, ..., \frac{D_k}{S_k})$, update as follow:

1. find task j which has $min(\frac{D_1}{S_1}, ..., \frac{D_k}{S_k})$,

2. transfer out S_j workers into S_k first, S_{k+1} second and so on, till the first p (p>=0) s.t. $S_{k+p} = X - \sum_{i=1}^{k+p-1} M_i$,

3. set k=k+p then

Repeat until all tasks are done

Solution for question 2

思路一

记第i个任务的预热时间为R_i，记t时刻分配到第i个任务的劳动力为$S_i(t)$，约束条件：$\sum_{i=1}^{n} S_i(t) \le X$ 且 $0 \le S_i(t) \le M_i * I(t \ge R_i)$。令$\int_0^{T_i} S_i(t)dt = D_i$，则 $T = max(T_i)$ (i=1,...,n)

目标： min T

以上模型可进一步写为如下形式.

Notations

1. [u,~,index]=unique([R,0]), u表示向量[R,0]中的无重复元素的升序排列, index(i)表示[R,0]中的第i个元素在u中的位置索引
2. 记$dt = diff(u) = [dt_1, ..., dt_r]$，则dt为连续时间段t离散化为r段后的表达，例如R=[1,3,3,6]时，u=[0,1,3,6],dt=[1,2,3]
3. 记$S_{n\times(r+1)} = \begin{pmatrix} S_{1,1} & \cdots & S_{1,r} & S_{1,r+1} \\ \vdots & \ddots & \vdots & \vdots \\ S_{n,1} & \cdots & S_{n,r} & S_{n,r+1} \end{pmatrix}$,$S_{i,j}$对应于第i个子任务在第j时间段内的人力分配.

Constraints

1. $\sum_{i=1}^{n} S_{i,j} \le X$ and $0 \le S_{i,j} \le M_i$ for $\forall j = 1, ..., r+1$.
2. $S_{i,j} = 0$ for $\forall j < index(i), i = 1, ..., n$
3. $S_{n\times(r+1)} \cdot \begin{pmatrix} dt \\ \Delta t \end{pmatrix} = \begin{pmatrix} S_{1,1} & \cdots & S_{1,r} & S_{1,r+1} \\ \vdots & \ddots & \vdots & \vdots \\ S_{n,1} & \cdots & S_{n,r} & S_{n,r+1} \end{pmatrix} \cdot \begin{pmatrix} dt_1 \\ \vdots \\ dt_r \\ \Delta t \end{pmatrix} = D^T$

Goal: minimize $sum([dt, \Delta t])$ over $(S, \Delta t)$ $\Leftrightarrow$ minimize Δt, where $\Delta t = max\{(D^T - S(:, 1:end-1) \cdot dt^T)./S(:, end)\}$

the final total time required to finish all the task is then T=$sum([dt, \Delta t])$

```
% The above algorithm is implemented as a function that is called MinTAllocation
```

```
clear;%data initialization
%X=randi(30), D=ceil(abs(randn(1,5)*5+5)), M=randi([1,6],1,5), R=randi([0,3],1,5)
%D = [160 20 10];M = [20 2 20];R = [0 0 2];X = 14;
X=19;D=[1 4 11 6];M=[3 2 3 6]; R=[0 1 3 0];
[S1,T1]=MinTAllocation(X,D,M,R)
```

```
Local minimum possible. Constraints satisfied.

fminimax stopped because the size of the current search direction is less than
twice the default value of the step size tolerance and constraints are
satisfied to within the default value of the constraint tolerance.

<stopping criteria details>
S1 = 4x3 double

    1.0000    1.0000    1.0000
         0    2.0000    1.4946
         0         0    3.0000
    1.2550    1.5100    1.7003

T1 = 6.6667
```

```
%t=timeit(@()MinTAllocation(X,D,M,R));
```

Solution for question 2

思路二

1. 同思路一，仍然把时间分成r段，例如R=[1,3,3,6,0]时，[u,~,index]=unique([R,0]), 返回u=[0,1,3,6], 因此我们把时间分成了r=4段：[0,1], [1,3], [3,6], [6,∞]. dt=diff(u)返回前r-1段时间的时长：dt=[1,2,3].
2. 在第j (j的初始值为1) 段时间区间内，以D_i为权重，将劳力X按加权比分配到该时间段内已经可以开工的子任务中，若按该法分配的劳力超出某子任务的劳力上限，则将超出部分以按D加权的方式分配到其它劳力未满的子任务中，重复此过程，直到所有劳力X都投入使用，或所有可开工的子任务都达到劳力的上限。记此时第i个子任务分配到的劳力为S_i人。计算第j阶段过后各个子任务的剩余工作量$D = (D - S \cdot dt_j) \cdot I(D - S \cdot dt_j >= 0)$. 更新j=j+1
3. 重复步骤2-3直到j=r+1, 此时所有子任务都已经过预热期，按Solution 1 for question 1 思路一 中的办法计算第r+1阶段完成所有剩余任务所需的最短时间Δt ，则总时间为$max(R) + \Delta t$

```
% The above algorithm is implemented as a function that is called MinTAllocation2
t=timeit(@()MinTAllocation2(X,D,M,R));
[S2,T2]=MinTAllocation2(X,D,M,R),disp(['Running Time:',num2str(t),'s'])
```

```
S2 = 4x3 double

     3     0     0
     0     2     0
     0     0     3
     6     0     0

T2 = 6.6667
Running Time:9.3342e-05s
```

Solutions for Question 2

思路三 A greedy method

Carry out the process in Solution for question 1 思路二 for every phase of ready.