Group 6

Nguyễn Duy Hưng
20194436

Trần Quốc Lập
20194443

Hoàng Thiện Tâm
20194450

MINI-PROJECT 20

Balanced Article Assignment

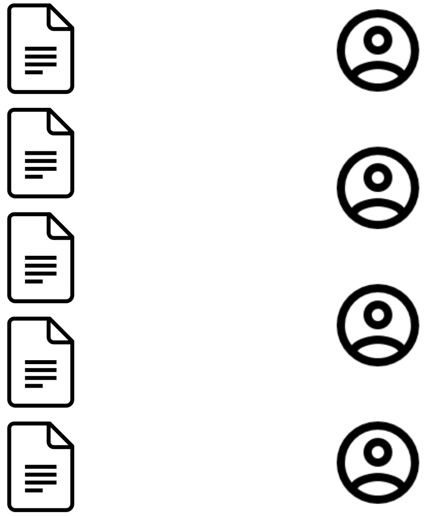
Contents

Problem
Different approaches
Running time

Contents

Problem

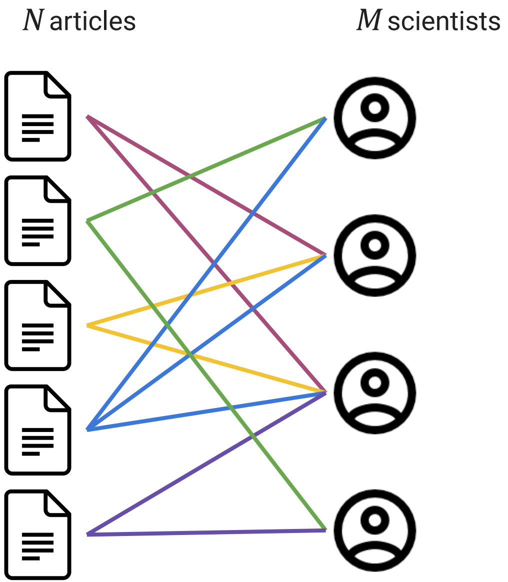
N articles M scientists



DISCARD   NEXT

Contents

Problem

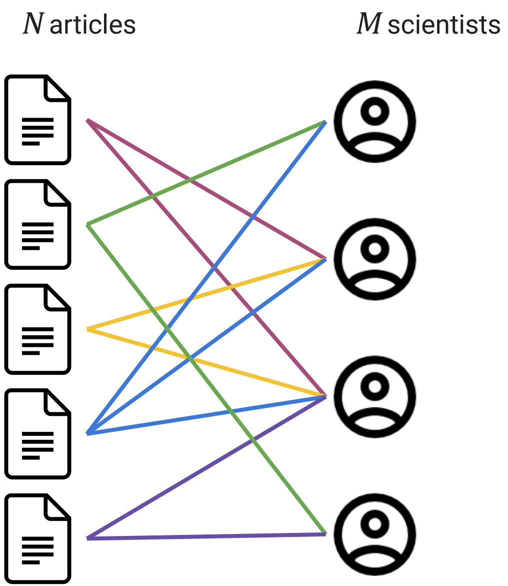


1. Each article **can only be given** to a **certain** list of scientists.

DISCARD NEXT

Contents

Problem

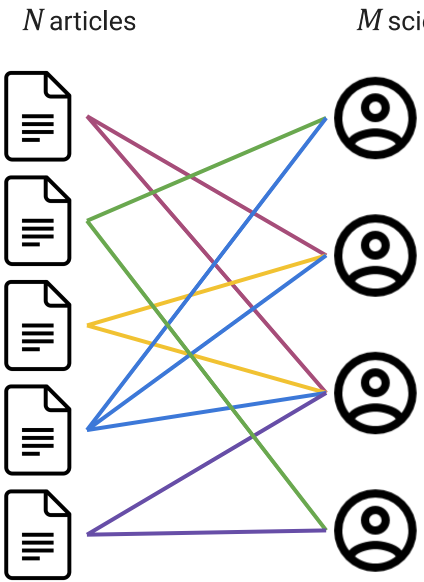


1. Each article **can only be given** to a **certain** list of scientists.
2. Every article must be given to **at least** K scientists

DISCARD NEXT

Contents

Problem



1. Each article **can only be given** to a **certain** list of scientists.

2. Every article must be given to **at least** K scientists

Find a way of assigning:

the ***maximum*** of scientists' workload

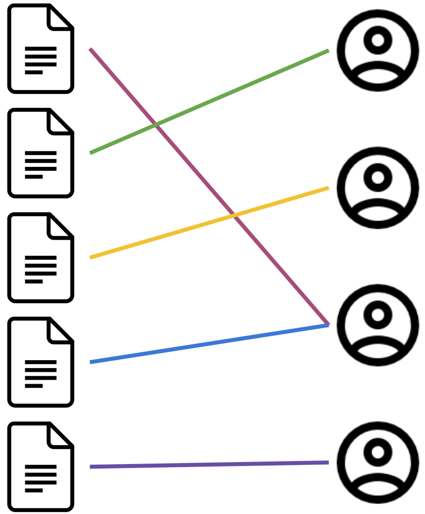
is ***minimized***.

DISCARD NEXT

Contents

Problem

N articles M scientists



1. Each article **can only be given** to a **certain** list of scientists.
2. Every article must be given to **at least** K scientists

Find a way of assigning:

the ***maximum*** of scientists' workload

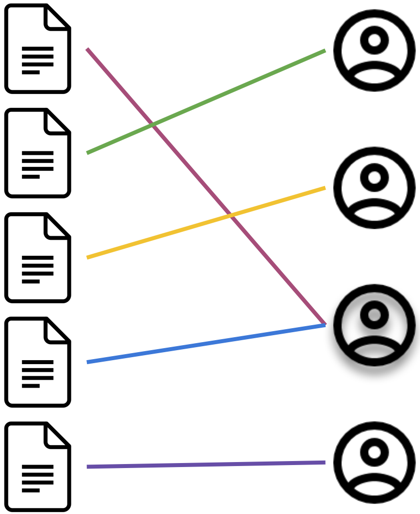
is ***minimized***.

DISCARD NEXT

Contents

Problem

N articles M scientists



1. Each article **can only be given** to a **certain** list of scientists.
2. Every article must be given to **at least** K scientists

Find a way of assigning:

the ***maximum*** of scientists' workload

is ***minimized***.

DISCARD NEXT

Contents

Input

```
 10       7       3
5 1 2 4 6 7
6 1 3 7 2 4 5
4 2 6 7 1
5 4 5 2 1 7
5 3 6 7 1 2
5 4 1 5 3 6
3 1 2 6
4 2 4 7 5
4 5 3 1 4
5 5 4 2 1 3
```

DISCARD NEXT

Mini project 20

Problem

Different approaches

Running time

Contents

← Input

```
    N = 10   M = 7   K = 3
   A[1]  = [1 2 4 6 7]
   A[2]  = [1 3 7 2 4 5]
   A[3]  = [2 6 7 1]
   A[4]  = [4 5 2 1 7]
   A[5]  = [3 6 7 1 2]
A  A[6]  = [4 1 5 3 6]
   A[7]  = [1 2 6]
   A[8]  = [2 4 7 5]
   A[9]  = [5 3 1 4]
   A[10] = [5 4 2 1 3]
```

DISCARD   NEXT

Problem

Different approaches

Running time

Contents

Input

```
    N = 10    M = 7    K = 3
   A[1]  = [1 2 4 6 7]
   A[2]  = [1 3 7 2 4 5]           L = [[1 1 0 1 0 1 1]
   A[3]  = [2 6 7 1]                    [1 1 1 1 1 0 1]
   A[4]  = [4 5 2 1 7]                  [1 1 0 0 0 1 1]
   A[5]  = [3 6 7 1 2]                  [1 1 0 1 1 0 1]
A  A[6]  = [4 1 5 3 6]                  [1 1 1 0 0 1 1]
   A[7]  = [1 2 6]                      [1 0 1 1 1 1 0]
   A[8]  = [2 4 7 5]                    [1 1 0 0 0 1 0]
   A[9]  = [5 3 1 4]                    [0 1 0 1 1 0 1]
   A[10] = [5 4 2 1 3]                  [1 0 1 1 1 0 0]
                                        [1 1 1 1 1 0 0]]
```

DISCARD NEXT

Problem

Different approaches

Running time

Contents

Input

```
      N = 10   M = 7   K = 3
   A[1]  = [1 2 4 6 7]
   A[2]  = [1 3 7 2 4 5]         L = [[1 1 0 1 0 1 1]   X = [[1 1 0 1 0 0 0]
   A[3]  = [2 6 7 1]                  [1 1 1 1 1 0 1]        [1 1 1 0 0 0 0]
   A[4]  = [4 5 2 1 7]                [1 1 0 0 0 1 1]        [1 1 0 0 0 1 0]
   A[5]  = [3 6 7 1 2]                [1 1 0 1 1 0 1]        [1 1 0 1 0 0 0]
A  A[6]  = [4 1 5 3 6]                [1 1 1 0 0 1 1]        [0 0 1 0 0 1 1]
   A[7]  = [1 2 6]                    [1 0 1 1 1 1 0]        [0 0 1 0 1 1 0]
   A[8]  = [2 4 7 5]                  [1 1 0 0 0 1 0]        [1 1 0 0 0 1 0]
   A[9]  = [5 3 1 4]                  [0 1 0 1 1 0 1]        [0 0 1 0 1 0 1]
   A[10] = [5 4 2 1 3]                [1 0 1 1 1 0 0]        [0 0 1 1 1 0 0]
                                      [1 1 1 1 1 0 0]]       [0 0 1 1 1 0 0]]
```

DISCARD NEXT

Contents

Input

```
    N = 10   M = 7   K = 3
   A[1]  = [1 2 4 6 7]
   A[2]  = [1 3 7 2 4 5]
   A[3]  = [2 6 7 1]
   A[4]  = [4 5 2 1 7]
   A[5]  = [3 6 7 1 2]
A  A[6]  = [4 1 5 3 6]
   A[7]  = [1 2 6]
   A[8]  = [2 4 7 5]
   A[9]  = [5 3 1 4]
   A[10] = [5 4 2 1 3]
```

```
L = [[1 1 0 1 0 1 1]
     [1 1 1 1 1 0 1]
     [1 1 0 0 0 1 1]
     [1 1 0 1 1 0 1]
     [1 1 1 0 0 1 1]
     [1 0 1 1 1 1 0]
     [1 1 0 0 0 1 0]
     [0 1 0 1 1 0 1]
     [1 0 1 1 1 0 0]
     [1 1 1 1 1 0 0]]
```

```
X = [[1 1 0 1 0 0 0]
     [1 1 1 0 0 0 0]
     [1 1 0 0 0 1 0]
     [1 1 0 1 0 0 0]
     [0 0 1 0 0 1 1]
     [0 0 1 0 1 1 0]
     [1 1 0 0 0 1 0]
     [0 0 1 0 1 0 1]
     [0 0 1 1 1 0 0]
     [0 0 1 1 1 0 0]]
```

```
optimal_solution = [[1 2 4] [1 2 3] [1 2 6] [1 2 4] [3 6 7]
                    [3 5 6] [1 2 6] [4 5 7] [3 4 5] [3 4 5]]

objective_value = 5

workload = [5, 5, 5, 5, 4, 4, 2]
```

DISCARD NEXT

Problem
Different approaches
Running time

Contents

Problem analysis

Claim

If ∃ at least 1 solution for the assigning ⇒ also ∃ at least 1 optimal solution.

In that optimal solutions, every article has **exactly** K scientists to take.

In a certain way of assigning:

- The number of articles that the scientist `j` takes on: $\sum_{i=1}^{N} x_{ij} l_{ij}$
- The number of scientists that the article `i` is given to: $\sum_{j=1}^{M} x_{ij} l_{ij}$

DISCARD NEXT

Contents

Problem analysis

Claim

If $\exists$ at least 1 solution for the assigning $\Rightarrow$ also $\exists$ at least 1 optimal solution.

In that optimal solutions, every article has **exactly** K scientists to take.

Mathematical model:

- Minimize $$\max_{j=\overline{1,M}} \sum_{i=1}^{N} x_{ij} l_{ij}$$
- Subject to $$\sum_{j=1}^{M} x_{ij} l_{ij} = K \ \forall i = \overline{1, N}$$

DISCARD NEXT

Problem
Different approaches
Running time

Contents

Problem

Different approaches

Running time

Ortools using MIP solver

```
def ortools():
    """OR-tools using Mixed-Integer Programming."""
    global objective_value, optimal_solution

    solver = pywraplp.Solver.CreateSolver('SCIP')
    INF = solver.infinity()

    X = np.array([[None]*M for i in range(N)])
    Y = solver.IntVar(-INF, INF, 'Y')            # maintaining the objective function

    for i in range(N):
        for j in range(M):
            X[i, j] = solver.IntVar(0, int(L[i, j]), 'X[{}, {}]'.format(i, j))

    for i in range(N):
        solver.Add(K == sum(L[i, j]*X[i, j] for j in range(M)))

    for j in range(M):
        solver.Add(Y >= sum(L[i, j]*X[i, j] for i in range(N)))

    solver.Minimize(Y)
    status = solver.Solve()

    if status == pywraplp.Solver.OPTIMAL:
        objective_value = solver.Objective().Value()
        optimal_solution = [[j+1 for j in range(M) if X[i,j].solution_value()] for i in range(N)]
```

Problem

Different approaches

Running time

Contents

Heuristics

Algorithm description

- Consider the article `i` that has not been taken by any scientists yet:
  - Pick out of `A[i]` a list of K scientists who are taking the lightest workload as possible. Assign the article `i` for them.
  - Increase the workload of these scientists by 1.
- Consider the article `i + 1`.
- ...

Pseudo-code

```
Heuristics:
  if FEASIBLE:
     workload[j] <- 0 initialize for each scientist j

     for each article i:
        list <- sorted( [workload[j], j] for j in A[i] )
        optimal_solution[i] <- list[:K]

        for each scientist j in optimal_solution[i]:
           increase workload[j] by 1
     objective_value = max(workload)
```

Problem
Different approaches
Running time

Contents

Backtracking

Algorithm description

- With an article `i`:
    - Consider a K-subset of `A[i]` corresponding to a way of choosing K scientists who can take on the article `i`. There are totally `len(A[i])` C_K subsets of that way.
        - Assigning the article `i` to them by `solution[i]` $\longleftarrow$ that K-subset
        - Increase `workload[j]` by 1.
            - Move to considering the article `i + 1`.
            - ...
                - Ultimately, if `max(workload)` < the smallest `objective_value` that has been recorded so far, we update.
    - Roll back to considering another K-subset of `A[i]`.
- ...

Problem

Different approaches

Running time

Backtracking

Pseudo-code

```
Backtracking:
    initialize workload[j] <- 0 for each scientist j
    if the problem is feasible:
        Try(1, 1)      i.e finding the 1st scientist to take the article 1

Try(i, t):
    for each scientist j in A[i] that hasn't been assigned i yet:
        if workload[j] + 1 < objective_value:
            solution[i][t] <- j          i.e set him as the (t)th scientist to takes the article i
            increase workload[j] by 1
            if i = N and t = K:    then Update
            else if t < K:
                Try(i, t + 1)            i.e find (t + 1)th scientist for the article i
            else if i = N:
                Try(i + 1, 1)            i.e move to the next article
            solution[i][t] <- NULL       i.e omit j in order to try other scientists
            decrease workload[j] by 1
Update:
    if max(workload) < objective_value:
        objective_value <- max(workload)
        optimal_solution <- solution
```

Problem
Different approaches
Running time

Contents

Problem

Different approaches

Running time

Special algorithm

Algorithm description

```
L = [[0 4 5 4 3]
     [2 0 1 1 0]
     [4 1 1 1 1]
     [4 1 1 1 1]
     [3 1 1 1 0]
     [3 1 1 0 1]]
```

Initial state

Ideal:

➔ We aim to minimize the_max_the_article_taken_among_scientists
So, I decided to reduce the max of the article till I can't reduce it any more.
It's not mean that I can reduce the max to any value arbitrarily
I decided to reduce the max to the predecessor(the number that has the value just only less than the max number. Ex:...)
If after reduction, we can't not reach the predecessor, so we find the solution...
Else we continue reduce them :)))

Another case: If we don't have predecessor(ex: a sequence numbers have the same value)
So, we reduce 'slowly': each value reduce only 1
If after reduce, which value doesn't change is our solution

Problem
Different approaches
Running time

Special algorithm

Algorithm description

```
L = [[0 4 5 4 3]
     [2 0 1 1 0]
     [4 1 1 1 1]
     [4 1 1 1 1]
     [3 1 1 1 0]
     [3 1 1 0 1]]
```

Initial state

```
L = [[0 3 4 4 5]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [4 1 1 1 1]]
```

Special algorithm

Algorithm description

```
L = [[0 4 5 4 3]
     [2 0 1 1 0]
     [4 1 1 1 1]
     [4 1 1 1 1]
     [3 1 1 1 0]
     [3 1 1 0 1]]
```

Initial state

```
L = [[0 3 4 4 5]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [4 1 1 1 1]]
```

↓

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [3 1 1 1 0]]
```

Problem

Different approaches

Running time

Special algorithm

Algorithm description

```
L = [[0 4 5 4 3]
     [2 0 1 1 0]
     [4 1 1 1 1]
     [4 1 1 1 1]
     [3 1 1 1 0]
     [3 1 1 0 1]]
```

Initial state

```
L = [[0 3 4 4 5]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [4 1 1 1 1]]
```

↓

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [3 1 1 1 0]]
```

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [3 1 1 1 0]
     [4 1 1 1 1]]
```

Special algorithm

Algorithm description

```
L = [[0 4 5 4 3]
     [2 0 1 1 0]
     [4 1 1 1 1]
     [4 1 1 1 1]
     [3 1 1 1 0]
     [3 1 1 0 1]]
```

Initial state

```
L = [[0 3 4 4 5]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [4 1 1 1 1]]
```

↓

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [3 1 1 1 0]]
```

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [3 1 1 1 0]
     [4 1 1 1 1]]
```

↓

```
L = [[0 3 3 3 3]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [2 1 0 1 0]
     [2 1 1 0 0]]
```

Problem

Different approaches

Running time

Special algorithm

Algorithm description

```
L = [[0 4 5 4 3]
     [2 0 1 1 0]
     [4 1 1 1 1]
     [4 1 1 1 1]
     [3 1 1 1 0]
     [3 1 1 0 1]]
```

Initial state

```
L = [[0 3 4 4 5]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [4 1 1 1 1]]
```

↓

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [3 1 1 1 0]]
```

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [3 1 1 1 0]
     [4 1 1 1 1]]
```

↓

```
L = [[0 3 3 3 3]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [2 1 0 1 0]
     [2 1 1 0 0]]
```

```
L = [[0 3 3 3 3]
     [2 0 0 1 1]
     [2 1 0 1 0]
     [2 1 1 0 0]
     [3 0 1 1 1]
     [3 1 1 0 1]]
```

Special algorithm

Algorithm description

```
L = [[0 4 5 4 3]
     [2 0 1 1 0]
     [4 1 1 1 1]
     [4 1 1 1 1]
     [3 1 1 1 0]
     [3 1 1 0 1]]
```

Initial state

```
L = [[0 3 4 4 5]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [4 1 1 1 1]]
```

↓

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [3 1 1 1 0]]
```

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [3 1 1 1 0]
     [4 1 1 1 1]]
```

↓

```
L = [[0 3 3 3 3]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [2 1 0 1 0]
     [2 1 1 0 0]]
```

```
L = [[0 3 3 3 3]
     [2 0 0 1 1]
     [2 1 0 1 0]
     [2 1 1 0 0]
     [3 0 1 1 1]
     [3 1 1 0 1]]
```

↓

```
L = [[0 3 3 2 2]
     [2 0 0 1 1]
     [2 1 0 1 0]
     [2 1 1 0 0]
     [2 0 1 0 1]
     [2 1 1 0 0]]
```

Problem
Different approaches
Running time

Special algorithm

Algorithm description

```
L = [[0 4 5 4 3]
     [2 0 1 1 0]
     [4 1 1 1 1]
     [4 1 1 1 1]
     [3 1 1 1 0]
     [3 1 1 0 1]]
```

Initial state

```
L = [[0 2 2 3 3]
     [2 1 1 0 0]
     [2 1 0 1 0]
     [2 0 0 1 1]
     [2 0 1 0 1]
     [2 0 0 1 1]]
```

final state

```
L = [[0 3 4 4 5]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [4 1 1 1 1]]
```

↓

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [4 1 1 1 1]
     [3 1 1 1 0]]
```

```
L = [[0 3 4 4 4]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [3 1 1 1 0]
     [4 1 1 1 1]]
```

↓

```
L = [[0 3 3 3 3]
     [2 0 0 1 1]
     [3 0 1 1 1]
     [3 1 1 0 1]
     [2 1 0 1 0]
     [2 1 1 0 0]]
```

```
L = [[0 3 3 3 3]
     [2 0 0 1 1]
     [2 1 0 1 0]
     [2 1 1 0 0]
     [3 0 1 1 1]
     [3 1 1 0 1]]
```

↓

```
L = [[0 3 3 2 2]
     [2 0 0 1 1]
     [2 1 0 1 0]
     [2 1 1 0 0]
     [2 0 1 0 1]
     [2 1 1 0 0]]
```

Experiments

Data	Size	MIP	Backtracking	Special approach	Heuristics		Opt
					Time	Result	
Data 1	5	0.031	0.00016	0.0004	0.00011	3	3
Data 2	10	0.029	0.08074	0.0007	0.00013	5	5
Data 3	20	0.036	_	0.0018	0.00021	9	8
Data 4	50	0.297	_	0.0066	0.00081	22	22
Data 5	100	1.047	_	0.0509	0.00174	44	43
Data 6	500	59.149	_	1.1152	0.01484	129	129

Group 6

Nguyễn Duy Hưng
20194436

Trần Quốc Lập
20194443

Hoàng Thiện Tâm
20194450

Thanks for watching!!!

Love you all! <3 <3 <3