Project INF421:
User scheduling in 5G

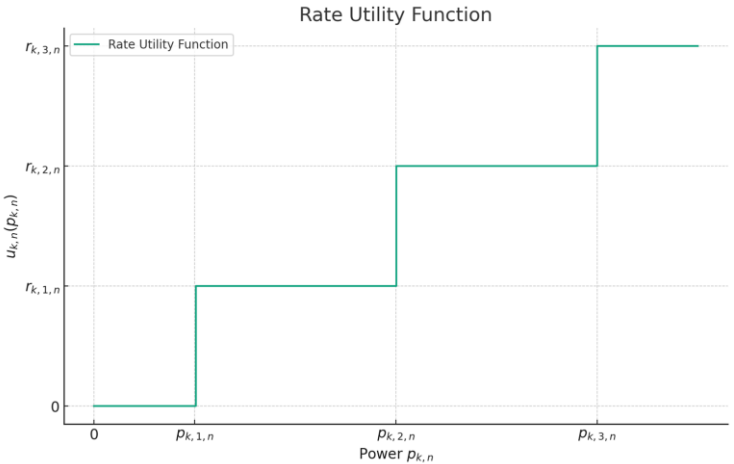
Group 7

Zhicheng HUI

Zhengya NING

Problem formulation

A set $\mathcal{K}$ of K users
A set $\mathcal{N}$ of N channels
The utility functions for each pair of user k and channel n



Problem formulation

$$\text{Maximize} \quad \sum_{k,m,n} x_{k,m,n} \times u_{k,n}(p_{k,m,n})$$

$$\text{Subject to} \quad \sum_{k,m} x_{k,m,n} = 1 \quad \forall n \in \mathcal{N}$$

$$\sum_{k,m,n} x_{k,m,n} \times p_{k,m,n} \leq p$$

$$x_{k,m,n} \in \{0, 1\}.$$

Pre-processing

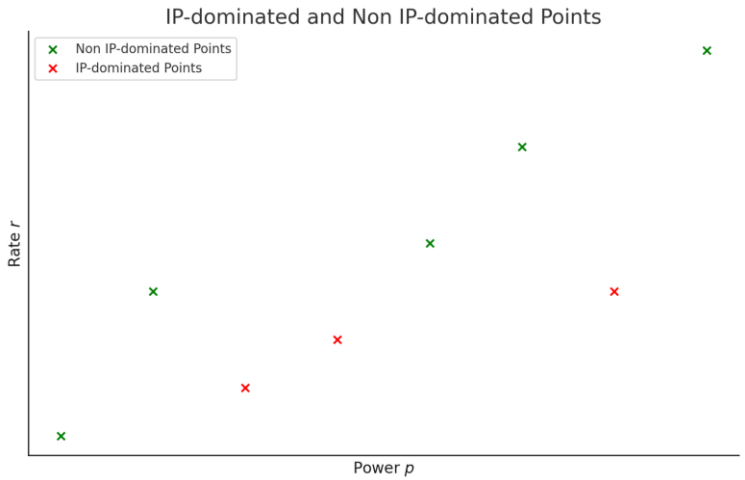
Quick preprocessing

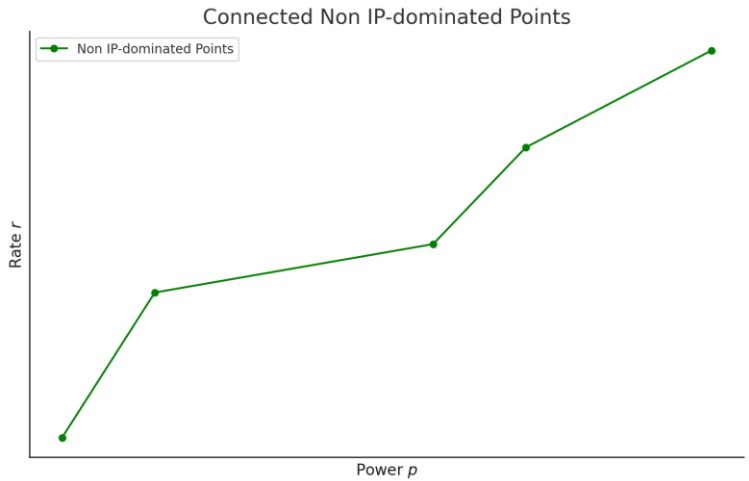
No solution: $\sum_{n} \min_{k,m} p_{k,m,n} \leq p$

Remove this triplet: $p_{k',m',n'} + \sum_{n,n \neq n'} \min_{k,m} p_{k,m,n} \leq p$

Pre-processing

Remove IP-dominated terms





Pre-processing

Remove IP-dominated terms

Algorithm 1: removeInversion(L)

Input: An array of triplets L sorted in ascending order of transmit power
Output: The array L with IP-dominated terms removed

```
lastRate ← 0
foreach (k, m, n) ∈ L do
    if r_{k,m,n} <= lastRate then
        delete (k, m, n) from L
    else
        lastRate ← r_{k,m,n}
    end
end
return L
```

Algorithm 2: removeIPDominated($N, L_1, ..., L_N$)

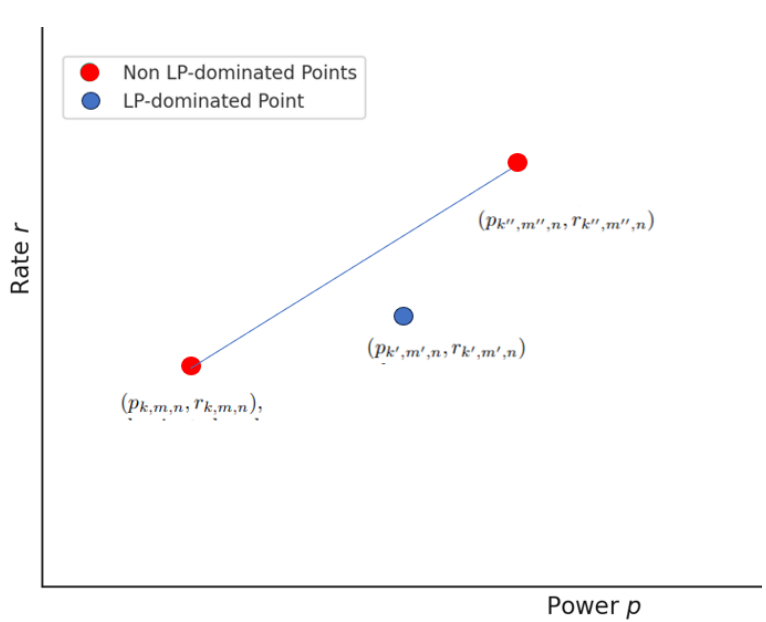
Input: The number of channels N and their triplets in arrays $L_1, ..., L_N$
Output: Arrays of triplets with IP-dominated terms removed

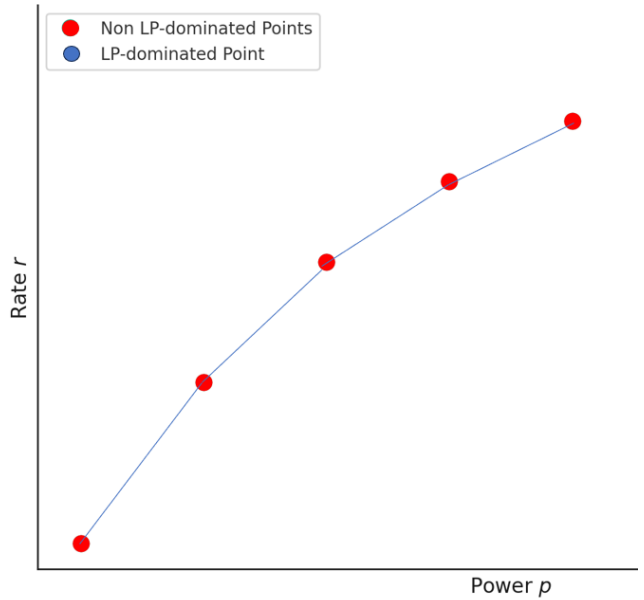
```
for i ← 1 to N do
    Sort the array L_i of triplets (k, m, n) by increasing order of p_{k,m,n}
    L_i ← removeInversion(L_i)
end
return L_1, ..., L_N
```

Complexity: $O(NKM \log(KM))$

Pre-processing

Remove LP-dominated terms





Pre-processing

Remove LP-dominated terms

Algorithm 3: removeIncreasingRatio(L)

Input: An array of triplets L sorted in ascending order of transmit power
Output: The array L with LP-dominated terms removed

```
tempQueue ← EmptyQueue
foreach (k'', m'', n) ∈ L do
    while tempQueue.size() ≥ 2 and (r_{k'',m'',n} − r_{k',m',n}) / (p_{k'',m'',n} − p_{k',m',n}) ≥ (r_{k',m',n} − r_{k,m,n}) / (p_{k',m',n} − p_{k,m,n}) do
        Remove the last triplet from the queue
    end
    Add the current triplet to the queue
end
Remove all the triplets in L
Put the triplets in the queue to L
return L
```

Complexity: $O(NKM \log(KM))$

Pre-processing

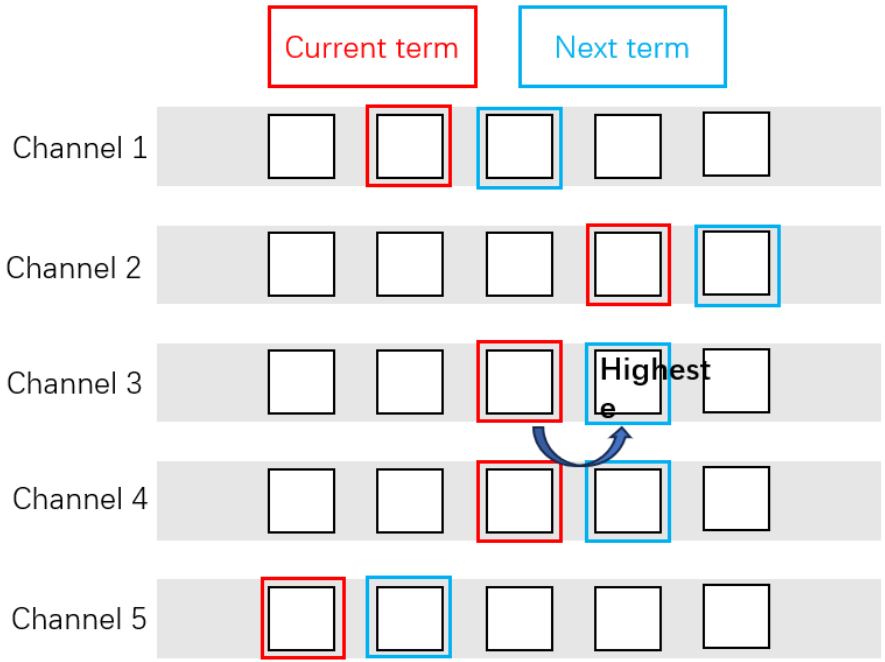
	Original size	Size after step 1	Size after step 2	Size after step 3
test1.txt	24	24	10	5
test2.txt	24	0	0	0
test3.txt	24	24	13	9
test4.txt	614400	614400	14732	4991
text5.txt	2400	1954	301	180

Linear programming and greedy algorithm

Incremental efficiency: $$e_{\ell n} = \frac{r_{\ell,n} - r_{\ell-1,n}}{p_{\ell,n} - p_{\ell-1,n}}$$

Greedy idea: always select the term with highest incremental dfficiency

Linear programming and greedy algorithm



Complexity: $O(N\bar{K}M\log(N))$.

LP Solver

GLPK (GNU Linear Programming Kit)

maximize

$$z = 10x_1 + 6x_2 + 4x_3$$

subject to

$$\begin{aligned} x_1 + \ x_2 + \ x_3 &\leq 100 \\ 10x_1 + 4x_2 + 5x_3 &\leq 600 \\ 2x_1 + 2x_2 + 6x_3 &\leq 300 \end{aligned}$$

where all variables are non-negative

$$x_1 \geq 0, \ x_2 \geq 0, \ x_3 \geq 0$$

maximize

$$z = 10x_1 + 6x_2 + 4x_3$$

subject to

$$\begin{aligned} p &= \ x_1 + \ x_2 + \ x_3 \\ q &= 10x_1 + 4x_2 + 5x_3 \\ r &= \ 2x_1 + 2x_2 + 6x_3 \end{aligned}$$

and bounds of variables

$$\begin{aligned} -\infty < p \leq 100 \quad & 0 \leq x_1 < +\infty \\ -\infty < q \leq 600 \quad & 0 \leq x_2 < +\infty \\ -\infty < r \leq 300 \quad & 0 \leq x_3 < +\infty \end{aligned}$$

LP Solver

$$\text{Maximize} \quad \sum_{k,m,n} x_{k,m,n} \times u_{k,n}(p_{k,m,n})$$

$$\text{Subject to} \quad \sum_{k,m} x_{k,m,n} = 1 \quad \forall n \in \mathcal{N}$$

$$\sum_{k,m,n} x_{k,m,n} \times p_{k,m,n} \leq p$$

$$x_{k,m,n} \in \{0, 1\}.$$

Matrix of size

$$(N^2 + 1)KM$$

For text4 more than 4e8 coefficients

Results

	Optimal rate	Used power	Total power
test1.txt	386.1538	96.0000	100.0000
test2.txt	No solution	No solution	No solution
test3.txt	368.0000	94.0000	100.0000
test4.txt	8543.0973	11644.0000	16000.000000
text5.txt	1390.0000	751.0000	1000.0000

	CPU runtime (ms)	Optimal rate	Used power
Greedy algorithm	29	1390.0000	751.0000
LP solver	1689	1637.0000	1000.0000

Dynamic Programming

Idea: Break the problem down into smaller subproblems

The problem: Calculate the maximum data rate under a total transmit power budget

Tactic: Introduce different stages during the decision

Dynamic Programming

The maximum sum data rate achievable with the first n channels under a budget of p

$$f_{n,p} = \max_{k,m}\{f_{n-1,p-p_{k,m,n}} + r_{k,m,n}\}$$

Time complexity: $O(NpKM)$.

Space complexity: $\Theta(Np) \longrightarrow \Theta(p)$

Dynamic Programming

Another formulation: Given a sum data rate as goal, find the minimum necessary power

$$f_{n,r} = \min_{k,m}\{f_{n-1,r-r_{k,m,n}} + p_{k,m,n}\}$$

Time complexity: $O(NUKM)$

Dynamic Programming

A few implementation detail:

$$f_{n,p} = \max_{k,m}\{f_{n-1,p-p_{k,m,n}} + r_{k,m,n}\} \qquad p < p_{k,m,n} \rightarrow \text{not affordable}$$

$$f_{n,r} = \min_{k,m}\{f_{n-1,r-r_{k,m,n}} + p_{k,m,n}\} \qquad r < r_{k,m,n} \rightarrow \text{mission complete}$$

Branch and Bound

Brute-force : search space of size $(KM)^N$, explosion !

Pruning by estimating the solution upper bound in order to reduce the search space

Branch and Bound

Integer solution of LP ⟶ feasible solution

Risk of degenerate into brute-force if no feasible solution found

Use stack (DFS) to produce an initial solution quickly

Branch and Bound

- Solution of LP smaller or equal than an existing feasible solution ⟶ prune
- Solution of LP is integer ⟶ update solution
- Solution of LP bigger than an existing feasible solution ⟶ continue

Online algorithm

Not always wise to go for the best ratio $\frac{r_{k,m,n}}{p_{k,m,n}}$

Use the expectation as criteria $\mathbb{E}\left(\frac{r_{k,m,n}}{p_{k,m,n}}\right) = \frac{r^{max}}{p^{max}}$

Wait until there's no other choice (last user)

Thank you for your attention!