Network System Capstone
@CS.NYCU

2025.04.10: Lab3

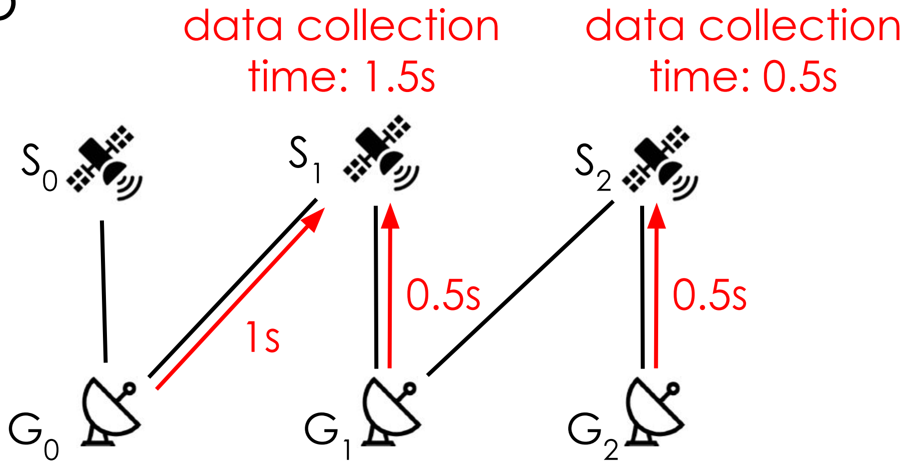
Instructor: Kate Ching-Ju Lin (林靖茹)

Agenda

- **Problem definition**
- Optimization model
- I/O
- Tasks
- Report & Result
- Submission

Problem Definition

- In this lab, we are going to use **OR-Tools** to solve a network optimization problem
- **Satellite–ground station association**
  - Each station wants to upload a data unit
  - Each satellite can receive data from one station at a time
  - Decision: Which satellite each station should connect to

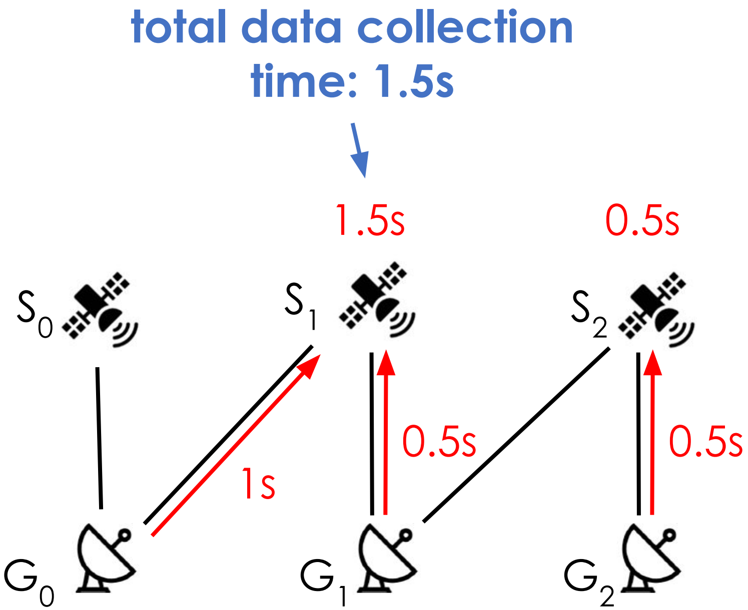


Input

- Network topology: G(S,V,E)
  - S: set of satellites
  - V: set of ground stations
- Parameters (derive the data rate of each link)
  - Tx power of ground stations
  - Mean noise power
  - Frequency
  - Bandwidth
  - Threshold of Rx power

Objective

- Minimize the total data collection time (i.e., job completion time)
  - Minimize the maximum data collection time among all satellites



Output

- Data rate of each link
- Satellite-ground station association results
- Collection time of each satellite
- Job completion time

Agenda

- Problem definition
- **Optimization model**
- I/O
- Tasks
- Report & Result
- Submission

Model (1/2)

- **Input:**
  - $G(\mathcal{V}, \mathcal{S}, \mathcal{E})$: bipartite graph consisting of set of satellites $\mathcal{S}$ and set of ground station $\mathcal{V}$
  - $w_{v,s}$: weight of link $(v, s) \in \mathcal{E}$

- **Variable:**
  - $X_{v,s} \in \{0, 1\}$: binary indicator deciding whether station *v* associates with satellite *s*

Model (2/2)

- **Integer Linear Programming (ILP)**

$$\min T$$

subject to

$$\sum_{s \in \mathcal{S}} X_{v,s} = 1, \forall v \in \mathcal{V}$$

$$\sum_{v \in \mathcal{V}} w_{v,s} X_{v,s} \leq T, \forall s \in \mathcal{S}$$

Agenda

- Problem definition
- Optimization model
- **I/O**
- Tasks
- Report & Result
- Submission

Input File (1/2)

- A plain text file: network.pos
- Line 1: tx power(dBm), noise(dBm), frequency(Hz), bandwidth(Hz) and rx power threshold(dBm)
- Line 2: *#ground station* and *#satellite*
- Each of next *#ground_station* lines: ground station ID, its coordinates (x, y, z)
- Each of next *#satellite* lines: satellite ID, its coordinates (x, y, z)

Input File (2/2)

- Format:

```
txpower noise frequency bandwidth rxpower_threshold
#ground_station #satellite

ground_station_id x y z
ground_station_id x y z
....

satellite_id x y z
satellite_id x y z
....
```

Output File (1/2)

- A plain text file: network.ortools.out
- Line 1: total transmission time(second)
- Each of next #*ground stations* lines: ground station ID and its associated satellite ID
  - List ground station IDs in ascending order
  - Make sure each ground station connects to exactly one satellite
- Each of next #*satellite* line: satellite ID and its data collection time(second)
  - List satellite IDs in ascending order

Output File (2/2)

- Format

```
Maximum_transmission_time

ground_station_id satellite_id
ground_station_id satellite_id
....

satellite_id data_collection_time
satellite_id data_collection_time
....
```

Agenda

- Problem definition
- Optimization model
- I/O
- **Tasks**
- Report & Result
- Submission

Task Overview

network.pos

matlab

network.graph
(TODO1: generate this file)

OR-Tools

Greedy

network.ortools.out
(TODO2: generate this file)

network.greedy.out
(TODO3: generate this file)

Task 1: Calculate Data Rate (1/2)

- Calculate the data rate(kbps) of link (v,s) from ground station v to satellite s based on network.pos
  - Reuse the code of Lab1 and Lab2
  - Tx power, noise, frequency and bandwidth are used to calculate the data rate
  - Link exists if rx power >= rx power threshold
- Generate network.graph
  - This file is the input of OR-Tools program

Task 1: Calculate Data Rate (2/2)

- network.graph format
  - Line 1: *#ground station*, *#satellite*, *#link*
  - Each of next #*link* lines: link data rate

```
#ground_station #satellite #link

ground_station_id satellite_id data_rate
ground_station_id satellite_id data_rate
```

- We also provide a simple input (sample.graph), which you can use to test OR-tools program
  - You can generate simple testing data by yourself

Task 2: OR-Tools Program (1/2)

- Write your OR-Tools program
  - Read the input file network.graph
  - Calculate the weight of link
    - Weight: transmission time of one data unit
    - Assume one data unit is 1000kb
  - Write the code based on the ILP model
  - Output the results to network.ortools.out
  - Check network.ortools.out format on page 13

Task 2: OR-Tools Program (2/2)

- Run OR-Tools program
  - Modify cmake_or-tools/BasicExample/CMakeLists.txt

```
add_executable(${PROJECT_NAME} "")
target_sources(${PROJECT_NAME} PRIVATE "src/basic_example.cc")
target_include_directories(${PROJECT_NAME} PRIVATE ${PROJECT_SOUR
target_compile_features(${PROJECT_NAME} PUBLIC cxx_std_11)
```

  - Build & run

```
$ cd cmake_or-tools
$ cmake --build build -v
$ ./build/bin/BasicExample
```

Task 3: Comparison

- Compare OPT(OR-Tools) with the result of Greedy
  - Each ground station selects the satellite with the **highest data rate**
  - Input and output formats are the same as those used in OR-tools
  - Rename the output file as network.greedy.out
  - Compare Greedy and OPT in the report

Agenda

- Problem definition
- Optimization model
- I/O
- Tasks
- **Report & Result**
- Submission

Report & Result

- Name as report.pdf
- Explain how you implement your lab step by step for each commit version
- Compare OR-Tools and Greedy
  - Compare network.greedy.out and network.ortools.out
  - Comparing the execution time of the two algorithms
  - Compare the advantages and disadvantages of the two algorithms

Agenda

- Problem definition
- Optimization model
- I/O
- Tasks
- Report & Result
- **Submission**

Download Lab3 Repository

- Please install OR-Tools first
- Download Github repository and files

```
$ cd cmake_or-tools/BasicExample/src
$ git init
$ git remote add origin
git@github.com:NYCU-NETCAP2025/lab3-<GITHUB_ID>.git
$ git pull origin main
$ git branch -M main
```

Submission

- Add your own studentID to studentID.txt (same as lab1)
- File structure: (push these files to github)

```
└── src
    ├── basic_example.cc
    ├── bf.m
    ├── lab3_bipartite.cc
    ├── lab3_greedy.cc
    ├── lab3_ortools.cc
    ├── network.graph
    ├── network.greedy.out
    ├── network.ortools.out
    ├── network.pos
    ├── report.pdf
    ├── sample.graph
    └── studentID.txt
```

Due

- **Apr. 24 (Thu.) 23:59, 2025**
- Don’t need to submit to E3
- Commit **your flies** to your Github repository
  - Should have at least **3 commits** (Initial, work in progress, final)
  - One version should be at least **1 day** after another
- **Notice: You will get penalty with wrong file structure and naming**

Grading Policy

- Grade
  - Code correctness – 20%
  - Report – 50%
  - Result – 30%
- Late Policy
  - (Your score) * 0.8^{D}, where D is the number of days overdue
- Cheating Policy
  - Academic integrity: Homework must be your own – cheaters share the score
  - Both the cheaters and the students who aided the cheater equally share the score