PURDUE UNIVERSITY

STyGIANet

Solving Throughput of a Topology with Gurobi

CS 422: Computer Networks
Fall 2026

Vamsi Addanki
vaddank@purdue.edu

Gurobi

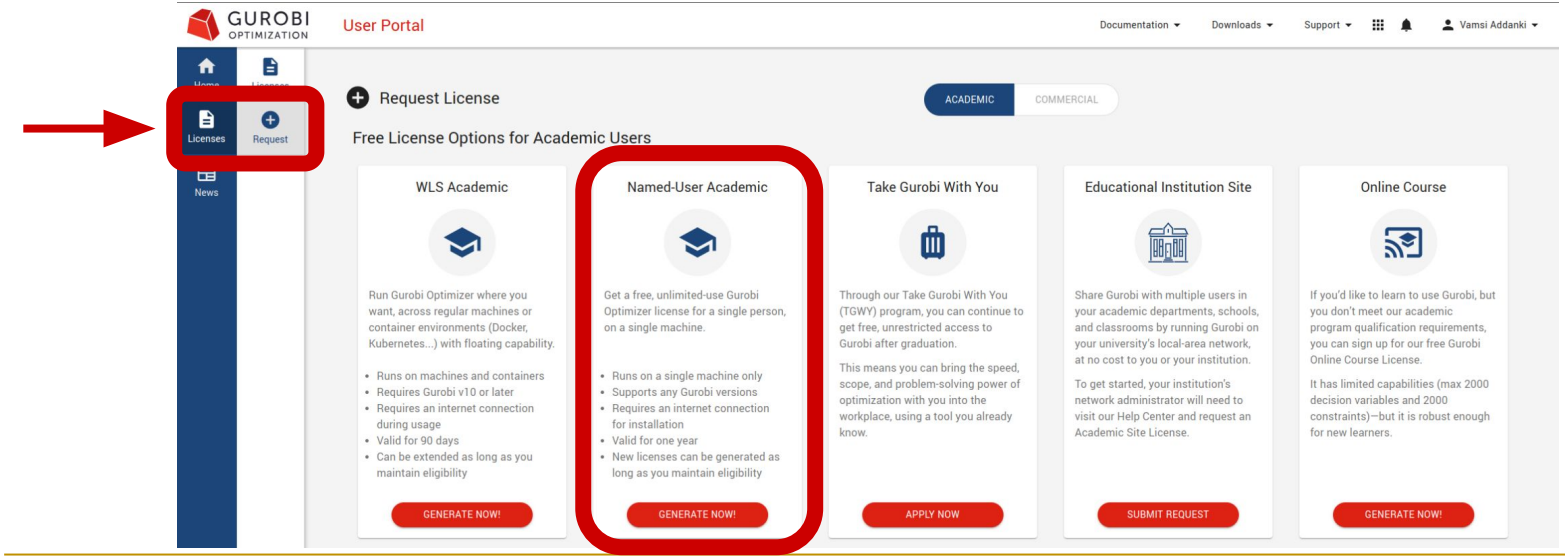
- Express your problem as a mathematical model and Gurobi will solve it for you!
  - LP, ILP, MILP…
- Widely used both in academia and industry
  - Network flow, transportation, traffic engineering, topology engineering, resource allocation, capacity planning…
- Resources:
  - Website: https://www.gurobi.com/
  - Documentation: https://www.gurobi.com/documentation/11.0/refman/index.html

Installing Gurobi

- We will use the python interface of Gurobi in this course
  - C, C++, Java…. Other interfaces also exist if you prefer alternative languages
- Using pip:
  - `python -m pip install gurobipy`
  - This version requires separate installation for license tools
  - https://support.gurobi.com/hc/en-us/articles/360059842732-How-do-I-set-up-a-license-without-installing-the-full-Gurobi-package
- Using conda (recommended):
  - `conda install -c gurobi gurobi`
  - This version supports academic license

Obtaining Gurobi Academic License

- Create an account: https://portal.gurobi.com/
- Request for an academic license

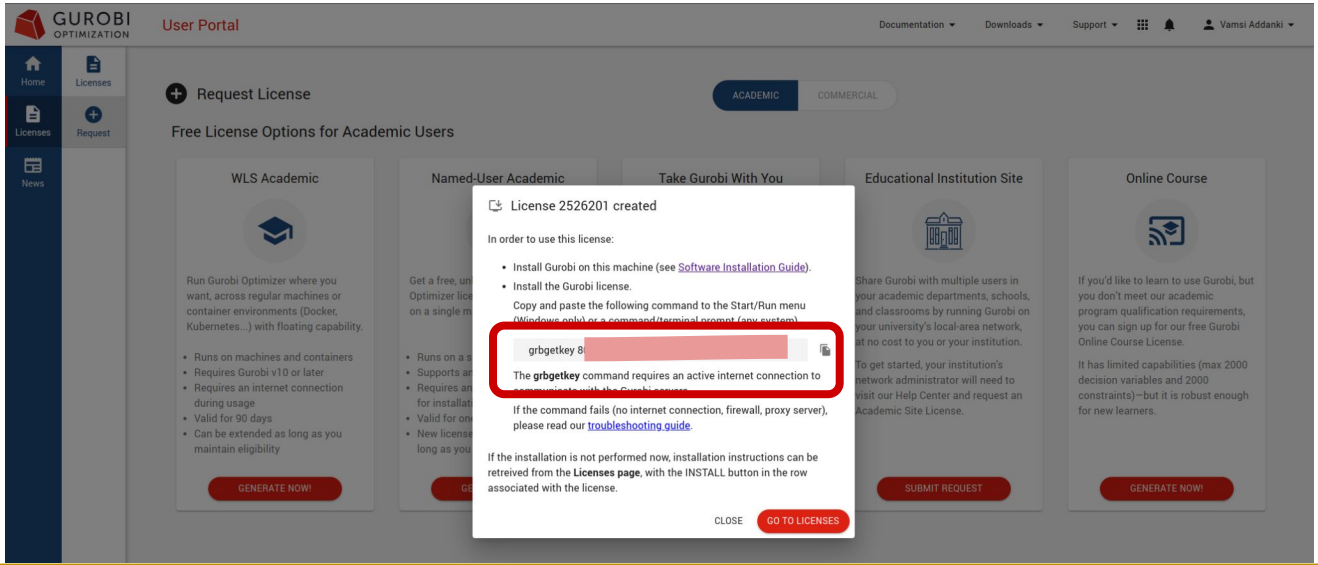


PURDUE UNIVERSITY

STyGIANet

Obtaining Gurobi Academic License

- run `grbgetkey <value>` command in your terminal



PURDUE UNIVERSITY

STyGIANet

Recall: Throughput Maximization Problem (Concurrent flow)

$$maximize \qquad \theta$$

$$\sum_{p \in P} F(p) \cdot \mathbb{I}(e \in p) \leq c(e) \qquad \forall e \in E$$

$$\sum_{p \in P_{s,d}} F(p) \geq \theta \cdot m_{s,d} \qquad \forall s \in S,\, d \in D$$

Recall: Throughput Maximization Problem

$$maximize \quad \theta$$

Throughput (Concurrent flow)

$$\sum_{p \in P} F(p) \cdot \mathbb{I}(e \in p) \leq c(e) \qquad \forall e \in E$$

$$\sum_{p \in P_{s,d}} F(p) \geq \theta \cdot m_{s,d} \qquad \forall s \in S,\ d \in D$$

Recall: Throughput Maximization Problem

$$maximize \quad \theta$$

$$\sum_{p \in P} F(p) \cdot \mathbb{I}(e \in p) \leq c(e) \qquad \forall e \in E$$

$$\sum_{p \in P_{s,d}} F(p) \geq \theta \cdot m_{s,d} \qquad \forall s \in S,\ d \in D$$

Sum of all flows across all paths that contain edge *e*

Recall: Throughput Maximization Problem

$$maximize \quad \theta$$

$$\sum_{p \in P} F(p) \cdot \mathbb{I}(e \in p) \leq c(e) \qquad \forall e \in E$$

$$\sum_{p \in P_{s,d}} F(p) \geq \theta \cdot m_{s,d} \qquad \forall s \in S,\ d \in D$$

Capacity of edge *e*

Recall: Throughput Maximization Problem

$$maximize \quad \theta$$

$$\sum_{p \in P} F(p) \cdot \mathbb{I}(e \in p) \leq c(e) \qquad \forall e \in E$$

$$\sum_{p \in P_{s,d}} F(p) \geq \theta \cdot m_{s,d} \qquad \forall s \in S,\ d \in D$$

Constraint applies to all edges in the topology

Recall: Throughput Maximization Problem

$$maximize \quad \theta$$

$$\sum_{p \in P} F(p) \cdot \mathbb{I}(e \in p) \leq c(e) \qquad \forall e \in E$$

$$\sum_{p \in P_{s,d}} F(p) \geq \theta \cdot m_{s,d} \qquad \forall s \in S,\ d \in D$$

Total flow across all *s*,*d* paths i.e., total flow from *s* to *d*

PURDUE UNIVERSITY

STyGIANet

Recall: Throughput Maximization Problem

$$maximize \quad \theta$$

$$\sum_{p \in P} F(p) \cdot \mathbb{I}(e \in p) \leq c(e) \qquad \forall e \in E$$

$$\sum_{p \in P_{s,d}} F(p) \geq \theta \cdot m_{s,d} \qquad \forall s \in S,\ d \in D$$

scaled down traffic demand between *s,d*

Solving Throughput Maximization Problem with Gurobi

- Import libraries

```
import gurobipy as gp

from gurobipy import GRB

import networkx as nx

import numpy as np
```

Solving Throughput Maximization Problem with Gurobi

- Example topology: Random regular graph with 8 servers, each with 4 links

```
N = 8  # number of nodes
d = 4   # degree
G = nx.random_regular_graph(d, N)
```

Solving Throughput Maximization Problem with Gurobi

- Alternatively, you could create your own topologies as follows

```
G = nx.DiGraph() # Directed graph object

for i in range(N):

    G.add_node(i) # add each node to the graph object

# Depending on the topology you would like to construct, add
edges as follows

G.add_edge(1,2) # this adds an edge from 1 to 2 to the graph.
```

Solving Throughput Maximization Problem with Gurobi

- Define capacities for each link

```
# Create a dictionary to store capacity variables
capacity = {}
for i in range(N):
    for j in range(N):
            if (i,j) in G.edges:
                capacity[(i,j)]=1
            else:
                capacity[(i,j)]=0
```

Solving Throughput Maximization Problem with Gurobi

- Specify the demand matrix

```
# Create All-to-All uniform demand matrix
demand = np.zeros((N,N))
for i in range(N):
    for j in range(N):
        if i!=j:
            demand[i][j] = d/(N-1)
```

Solving Throughput Maximization Problem with Gurobi

- From here on, the formulation remains the same for any topology
- Our LP formulation relies on the set of all paths in the given topology

```
all_paths = {}

for u in G.nodes():

    for v in G.nodes():

        if u!=v:

            all_paths[u, v] = [list(zip(path, path[1:])) for
path in nx.all_simple_paths(G, source=u, target=v)]
```

Solving Throughput Maximization Problem with Gurobi

- From here on, the formulation remains the same for any topology
- Our LP formulation relies on the set of all paths in the given topology

```
all_paths = {}

for u in G.nodes():
    for v in G.nodes():
        if u!=v:
            all_paths[u, v] = [list(zip(path, path[1:])) for
path in nx.all_simple_paths(G, source=u, target=v)]
```

Computes the set of all paths from source u to destination v



PURDUE UNIVERSITY

STyGIANet

Solving Throughput Maximization Problem with Gurobi

- Initialize the model

```
model = gp.Model("throughput")
```

Solving Throughput Maximization Problem with Gurobi

- Add flow variables for each path

```
flow_vars = {}

for (s, d), paths in all_paths.items():

    for p in range(len(paths)):

        flow_vars[s, d, p] =
model.addVar(vtype=GRB.CONTINUOUS, name=f"flow_{s}_{d}_{p}",
lb=0)
```

Solving Throughput Maximization Problem with Gurobi

- Add flow variables for each path

```
flow_vars = {}
for (s, d), paths in all_paths.items():
    for p in range(len(paths)):
        flow_vars[s, d, p] =
model.addVar(vtype=GRB.CONTINUOUS, name=f"flow_{s}_{d}_{p}",
lb=0)
```

Solving Throughput Maximization Problem with Gurobi

- Add flow variables for each path

```
flow_vars = {}
for (s, d), paths in all_paths.items():
    for p in range(len(paths)):
        flow_vars[s, d, p] =
model.addVar(vtype=GRB.CONTINUOUS, name=f"flow_{s}_{d}_{p}",
lb=0
```

Solving Throughput Maximization Problem with Gurobi

- Add throughput variable

```
throughput = model.addVar(vtype=GRB.CONTINUOUS,
name='throughput', lb=0, ub=1)
```

Solving Throughput Maximization Problem with Gurobi

- Capacity constraints for each edge

```
for u, v in G.edges():

    model.addConstr(gp.quicksum(flow_vars[i, j, p] for (i,
j), paths in all_paths.items() for p in range(len(paths)) if
(u, v) in paths[p]) <= capacity[(u,v)],name=f"cap_{u}_{v}")
```

Solving Throughput Maximization Problem with Gurobi

- Capacity constraints for each edge

```
for u, v in G.edges():
    model.addConstr(gp.quicksum(flow_vars[i, j, p] for (i, j), paths in all_paths.items() for p in range(len(paths)) if (u, v) in paths[p]) <= capacity[(u,v)],name=f"cap_{u}_{v}")
```



As the name suggests, addConstr adds the specified constraint to the model

Solving Throughput Maximization Problem with Gurobi

- Capacity constraints for each edge

```
for u, v in G.edges():
    model.addConstr(gp.quicksum(flow_vars[i, j, p] for (i,
j), paths in all_paths.items() for p in range(len(paths)) if
(u, v) in paths[p]) <= capacity[(u,v)],name=f"cap_{u}_{v}")
```

An optimized version of “sum” to add gurobi variables

PURDUE UNIVERSITY

STyGIANet

Solving Throughput Maximization Problem with Gurobi

- Capacity constraints for each edge

```
for u, v in G.edges():
    model.addConstr(gp.quicksum(flow_vars[i, j, p] for (i, j), paths in all_paths.items() for p in range(len(paths)) if (u, v) in paths[p]) <= capacity[(u,v)],name=f"cap_{u}_{v}")
```



Total amount of flow on each edge must be less than its capacity

Solving Throughput Maximization Problem with Gurobi

- Demand constraints for each source-destination pair

Total s-d flow over the set of all s-d paths must be greater than or equal to the scaled demand.

```
for u in G.nodes():
    for v in G.nodes():
            model.addConstr(
                gp.quicksum(flow_vars[u,v,p] for p in
range(len(all_paths[u,v]))) >= throughput*demand[u][v],
                name=f"demand_{u}_{v}")
```

Solving Throughput Maximization Problem with Gurobi

- Set the objective: maximize throughput

```
model.setObjective(throughput, GRB.MAXIMIZE)
```

Solving Throughput Maximization Problem with Gurobi

- That's all Folks!

```
model.optimize()

print("Throughput for the given topology and the given
demand matrix is:", throughput.X)
```

Solving Throughput Maximization Problem with Gurobi

- That's all Folks!

```
model.optimize()

print("Throughput for the given topology and the given
demand matrix is:", throughput.X)
```

Access the variable value after the optimization

Solving Throughput Maximization Problem with Gurobi

- Have fun :)

Warmup: What is the throughput of a 4-regular random graph topology with 8 nodes for uniform demand?

Real-world network problem: Write an LP to solve for the throughput of a k-ary fattree topology for any given demand matrix?

Hint: Be careful! Fattree has different types of nodes: servers generate (and consume) flow. Switches only acts as transit but do not generate (or consume) flow.