Network Dynamics and Learning
Homework I

Patrik Pavan
345890

November 16, 2025

1 Exercise 1: Network Flow Analysis

1.1 Part (a): Cut Capacity and Minimum Cut

All possible cuts:

Table 1: All possible s-t cuts

Cut	U	U^C	Capacity
1	$\{o\}$	$\{a, b, c, d\}$	6
2	$\{o, a\}$	$\{b, c, d\}$	6
3	$\{o, b\}$	$\{a, c, d\}$	6
4	$\{o, a, b\}$	$\{c, d\}$	5
5	$\{o, c\}$	$\{a, b, d\}$	9
6	$\{o, a, c\}$	$\{b, d\}$	5
7	$\{o, b, c\}$	$\{a, d\}$	5
8	$\{o, a, b, c\}$	$\{d\}$	5

Minimum cut capacity: 5

There are multiple minimum cuts with capacity 5 (cuts 4, 6, 7, and 8). The minimum capacity to be removed for no feasible flow from o to d to exist is 5 units.

1.2 Part (b): Optimal Distribution of Extra Capacity

Strategy: The optimal approach is to iteratively identify the minimum cut and add capacity to the bottleneck edges (edges with smallest capacity in the current minimum cut). This ensures that each additional unit of capacity maximally increases the network throughput.

Algorithm:

1. Find the current minimum cut
2. Identify edges crossing the cut
3. Add 1 unit of capacity to the edge with smallest current capacity

4. Repeat until all x units are distributed

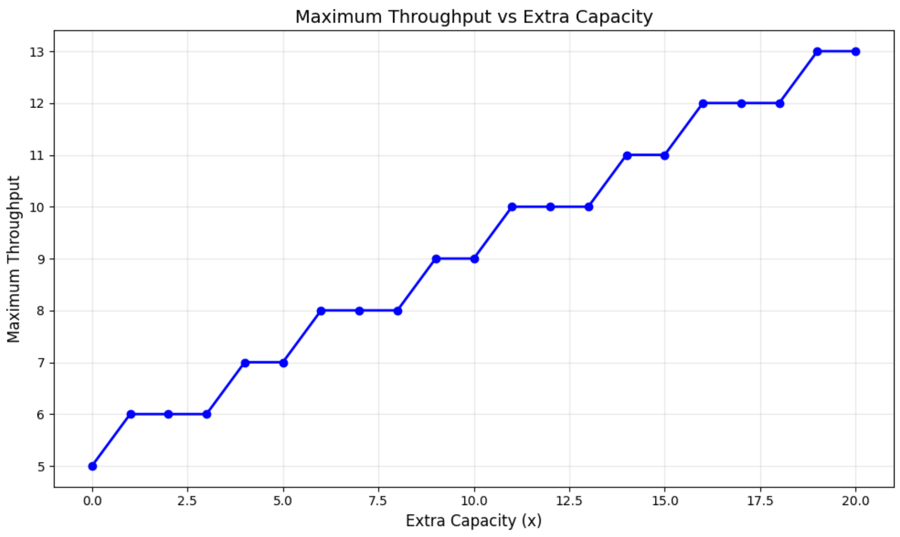

Figure 1: Maximum throughput as a function of extra capacity x.

The plot shows that the throughput increases monotonically with x, but not linearly. Plateaus occur when multiple units must be added before the next bottleneck is resolved.

1.3 Part (c): Adding a New Link with Extra Capacity

Optimal link placement: We add a direct link $e_8 : o \to d$ with initial capacity $c_8 = 1$. This link is optimal because it creates an independent path from source to destination, bypassing intermediate bottlenecks and immediately increasing the baseline throughput.

Strategy for capacity distribution: We use the same greedy algorithm as Part (b), but now the new link e_8 is included in the candidate set. The algorithm iteratively finds the minimum cut and adds capacity to the bottleneck edge (including e_8 if it's the bottleneck).

Results:

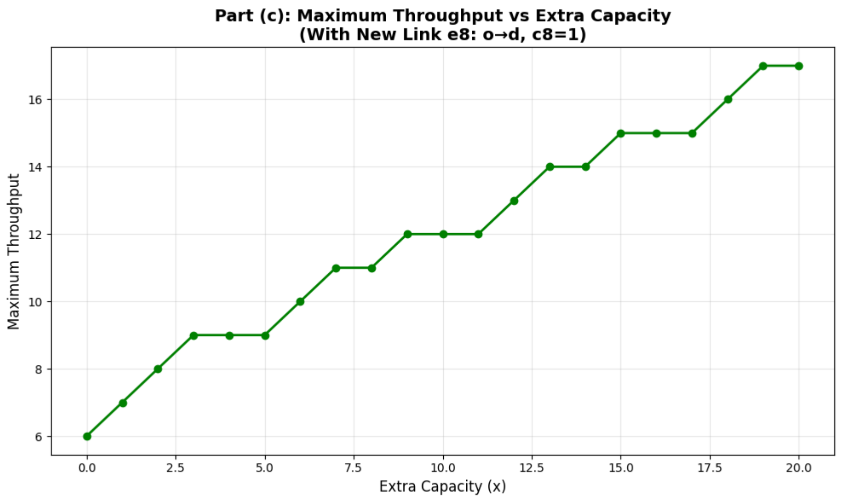

Figure 2: Maximum throughput with new link as a function of extra capacity x

2 Exercise 2: Centrality Measures

Consider the simple graph $G = (V, E)$ in Figure 2 from the homework description.

2.1 Part (a): Katz Centrality

With $\beta = 0.15$ and uniform intrinsic centrality μ:

Node	Katz Centrality
1	0.11813975
2	0.11813975
3	0.11813975
4	0.11813975
5	0.11813975
6	0.13008034
7	0.04424717
8	0.03171920
9	0.04281112
10	0.02430259
11	0.02430259
12	0.02430259
13	0.02430259
14	0.02430259
15	0.03893046

Table 2: Katz centrality values for all nodes with $\beta = 0.15$ and uniform μ.

2.2 Part (b): Distributed PageRank Algorithm

Iterative formula: $z(t+1) = (1-\beta)P'z(t) + \beta\mu$

where $P = D^{-1}W$ is the row-stochastic matrix, D is the diagonal degree matrix, $\beta = 0.15$, and $\mu = \frac{1}{n}\mathbf{1}$ is uniform.

Distributed Algorithm:

1. **Initialize:** Each node i sets $z_i(0) = \frac{1}{n}$
2. **Iterate until convergence:**
   - Each node i sends $\frac{z_i(t)}{d_i}$ to all out-neighbors (where d_i is node i's out-degree)
   - Each node i receives messages from in-neighbors and updates locally:

$$z_i(t+1) = (1-\beta)\sum_{j\to i}\frac{z_j(t)}{d_j} + \beta\cdot\frac{1}{n}$$

3. **Stop** when $\|z(t+1) - z(t)\| < \epsilon$

2.3 Part (c): Analysis of Nodes n6 and n9

Observed values:

- **Node n6**: Katz = 0.130 (rank 1), PageRank = 0.114 (rank 2)
- **Node n9**: Katz = 0.043 (rank 8), PageRank = 0.194 (rank 1)

The rankings are **completely reversed** between the two measures.

Node n6 (bridge to K5 clique): Ranks high in Katz because it connects to well-connected nodes (K5 clique) and gets their full centrality value. Ranks lower in PageRank because each K5 node has 6 connections, so n6 only gets $\frac{1}{6}$ of their PageRank—the rest stays inside the clique.

Node n9 (star hub with 5 leaves): Ranks low in Katz because its leaf neighbors are far from the center and have low centrality. Ranks highest in PageRank because each leaf only connects to n9, so they give 100% of their PageRank back to n9.

Key difference: Katz counts all neighbor centrality equally. PageRank splits a node's centrality among all its connections, so getting centrality from a node with few connections is worth more.

2.4 Part (d): PageRank for Different β Values

Table 3: PageRank centrality for different β values

β	r_{n_6}	r_{n_9}
0.00	0.1459	0.1162
0.25	0.1106	0.1911
0.50	0.1014	0.1667
0.75	0.0875	0.1270
1.00	0.0667	0.0667

Monotonicity analysis: The difference $r_{n_6} - r_{n_9}$ is **not monotone** in β. At $\beta = 0$, n6 has slightly higher centrality (+0.030). The difference becomes negative (n9 wins) for $\beta \in [0.1, 0.75]$ with maximum separation at $\beta \approx 0.25$ (difference = -0.081). At $\beta = 1$, both nodes have equal centrality. This non-monotone behavior occurs because two competing effects operate at different β ranges: dense subgraph preference (favors n6 at low β) versus star amplification (favors n9 at medium β).

Extreme values interpretation:

- $\beta = 0$: PageRank becomes the stationary distribution of the random walk on the graph with no teleportation. Node n6 wins slightly because it connects to the dense K5 clique, which has more internal circulation. However, the difference is small because both structures have reasonable walk properties.
- $\beta = 1$: PageRank equals the uniform intrinsic centrality $\mu = \frac{1}{n}$. The network structure becomes completely irrelevant—all nodes have identical centrality (0.0667 each).

3 Exercise 3: Traffic Network Optimization

3.1 Part (a): Shortest Path

Shortest path: $1 \to 2 \to 3 \to 9 \to 13 \to 17$
Total travel time: 0.5598 time units

3.2 Part (b): Maximum Flow

Maximum flow: 22,448

3.3 Part (c): Flow Conservation

The exogenous flow vector ν is:

$$\begin{aligned}\nu = [&16806,\ 8570,\ 19448,\ 4957,\ -746,\ 4768,\ 413,\ -2,\\ &-5671,\ 1169,\ -5,\ -7131,\ -380,\ -7412,\ -7810,\\ &-3430,\ -23544]^T\end{aligned}$$

3.4 Part (d): Social Optimum

Results:
Optimal total cost: 26,142.67
The optimal flow vector $f^* \in \mathbb{R}^{28}$ is:

$$\begin{aligned}f^* = (&6569.07, 5809.95, 3046.99, 3046.99, 10236.93, 4666.55, 3061.21,\\ &2596.03, 3104.55, 759.12, 0.01, 2762.96, 0.00, 3046.99,\\ &5570.38, 2893.88, 5040.96, 2364.46, 465.19, 2254.44,\\ &3359.07, 5613.50, 2371.92, 0.00, 6346.13, 5418.91,\\ &5040.96, 5040.96)^T\end{aligned}$$

3.5 Part (e): Wardrop Equilibrium

Results:
Total cost at Wardrop equilibrium: 15,731.95
The Wardrop equilibrium flow vector $f^{(0)} \in \mathbb{R}^{28}$ is:

$$\begin{aligned}f^{(0)} = (&6557.54, 6308.56, 2200.67, 2200.67, 10248.46, 4706.68, 2859.96,\\ &2232.26, 3349.96, 248.97, 11.69, 4096.21, 0.00, 2200.67,\\ &5541.78, 2343.35, 5294.13, 2095.69, 639.38, 2978.52,\\ &2982.73, 5961.25, 2522.41, 0.00, 6788.80, 4723.07,\\ &5294.13, 5294.13)^T\end{aligned}$$

3.6 Part (f): Wardrop Equilibrium with Marginal Cost Tolls

Results:

The Wardrop equilibrium flow vector with tolls $f^{(\omega)} \in \mathbb{R}^{28}$ is:

$$\begin{aligned} f^{(\omega)} = (&6570.06, 5810.19, 3047.00, 3047.00, 10235.94, 4666.15, 3061.51, \\ &2595.79, 3104.40, 759.87, 0.00, 2763.18, 0.00, 3047.00, \\ &5569.79, 2893.37, 5040.93, 2364.51, 465.72, 2254.57, \\ &3359.09, 5613.66, 2372.10, 0.00, 6345.96, 5419.11, \\ &5040.93, 5040.93)^T \end{aligned}$$

Observation: The Wardrop equilibrium with marginal cost tolls $f^{(\omega)}$ nearly coincides with the social optimum f^*. The maximum absolute difference is less than 1 vehicle per link. This demonstrates that marginal cost pricing aligns selfish routing behavior with social welfare.

3.7 Part (g): Alternative Cost Function

System optimum f^*:

The optimal flow vector $f^* \in \mathbb{R}^{28}$ is:

$$\begin{aligned} f^* = (&6584.43, 5577.92, 3367.42, 3367.42, 10221.57, 4669.47, 3156.42, \\ &2711.80, 2987.95, 1006.55, 0.00, 2210.48, 0.00, 3367.42, \\ &5552.09, 3084.73, 4986.91, 2519.54, 444.68, 1934.55, \\ &3529.36, 5463.88, 2202.34, 0.00, 6249.23, 5569.77, \\ &4986.91, 4986.91)^T \end{aligned}$$

Optimal cost: 15,350.35

Constructed tolls ω^*:

To align the Wardrop equilibrium with this system optimum: $\omega_e^* = f_e^* \tau_e'(f_e^*) - l_e$

Verification: Computing $f(\omega^*)$ yields maximum difference $\|f(\omega^*) - f^*\|_\infty = 0.24$, confirming the tolls successfully induce the system optimum.

A Python Code

https://github.com/ZeiX-P/network_dynamics_homeworks