Network Dynamics – Homework 1

16 November 2025

Abstract

In this report we present our solution to Homework 1. This report is the result of active collaboration between: Kevin Elezi (s316685), Marco Conte (s346753), Melika Ghasemi (s309495) and Mohamed Hatem (s346906).

Exercise 1

Consider the network in Figure 1 with link capacities:

$$c_1 = c_2 = c_3 = c_4 = 3, \quad c_5 = 2, \quad c_6 = 1.$$

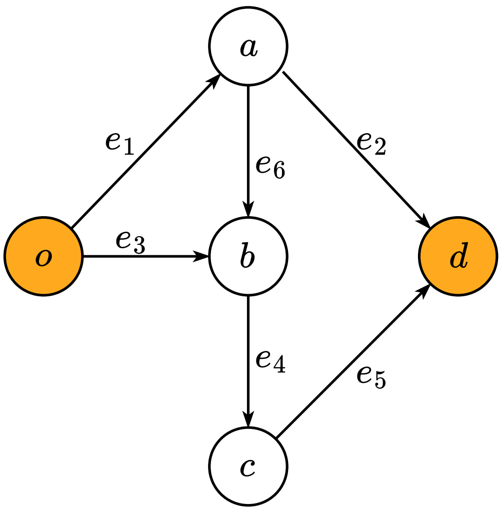

Figure 1: Exercise 1 graph.

Question 1. Compute the capacity of all the cuts and find the minimum capacity to be removed for no feasible flow from o to d to exist.

Question 2. You are given $x > 0$ extra units of capacity ($x \in Z$). How should you distribute them in order to maximize the throughput that can be sent from o to d? Plot the maximum throughput from o to d as a function of $x \geq 0$.

Question 3. You are given the possibility of adding to the network a directed link e_8 with capacity $c_8 = 1$ and $x > 0$ extra units of capacity ($x \in Z$). Where should you add the link and how should you distribute the additional capacity in order to maximize the throughput that can be sent from o to d? Plot the maximum throughput from o to d as a function of $x \geq 0$.

Question 1

We know from theory that the maximum throughput of a network equals the capacity of its minimal o-d cut, i.e.

$$\tau^\star = \min_{U \subset V : o \in U,\, d \notin U} c_U,$$

where $c_U = \sum_{e \in \delta^+(U)} c_e$ denotes the aggregate capacity of all edges leaving U.

Therefore, let's compute all the existing subsets min-cut U and their aggregate capacities:

$$\begin{aligned}
&U_1 = \{o, b, c\}, && c_{U_1} = c_1 + c_5 = 5\\
&U_2 = \{o, a, b, c\}, && c_{U_2} = c_2 + c_5 = 5\\
&U_3 = \{o\}, && c_{U_3} = c_1 + c_3 = 6\\
&U_4 = \{o, b\}, && c_{U_4} = c_1 + c_4 = 6\\
&U_5 = \{o, a, b\}, && c_{U_5} = c_2 + c_4 = 6\\
&U_6 = \{o, a\}, && c_{U_6} = c_2 + c_3 + c_6 = 7\\
&U_7 = \{o, c\}, && c_{U_7} = c_1 + c_3 + c_5 = 8\\
&U_8 = \{o, a, c\}, && c_{U_8} = c_2 + c_3 + c_5 + c_6 = 9
\end{aligned}$$

The **minimum cut capacity** is therefore:

$$c_{U^\star} = \min_i c_{U_i} = 5,$$

which gives the minimal total capacity to remove to make the flow infeasible. This minimum is obtained by the two cuts U_1 and U_2.

Question 2

Since the **maximum throughput** equals the **minimum cut capacity**, increasing throughput requires increasing the capacities of edges in the minimum cuts. The main challenge is deciding how to distribute the extra capacity $x > 0$.

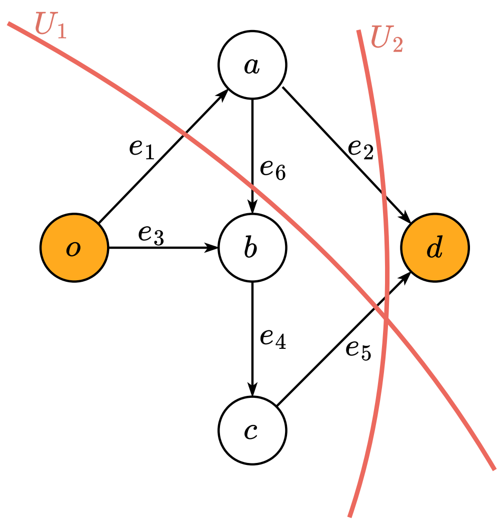

Figure 2: Exercise 1's min-cuts : U_1 and U_2.

Let the original edge capacities be represented by the vector

$$\mathbf{c} = (c_1, c_2, c_3, c_4, c_5, c_6).$$

Consider the minimum cuts obtained in Exercise 1 and observable in Figure 2 :

$$U_1 \text{ with } c_{U_1} = c_1 + c_5, \quad U_2 \text{ with } c_{U_2} = c_2 + c_5.$$

If we increase only c_2 by 1, this can be written as

$$\mathbf{c} + (0, 1, 0, 0, 0, 0),$$

which increases c_{U_2} to 6 but leaves $c_{U_1} = 5$ unchanged.

Similarly, increasing only c_5 by 1 corresponds to

$$\mathbf{c} + (0, 0, 0, 0, 1, 0),$$

which increases both c_{U_1} and c_{U_2} to 6, thereby raising the maximum throughput to 6.

Hence, a good heuristic approach is to distribute extra capacity along the *common edges* of the minimum cuts, as these increments affect the largest number of minimum-cut sets.

Extra capacity distribution (heuristic)

Algorithm 1 Increase maximum throughput with extra capacity

```
Require: Edge capacities $\mathbf{c} = (c_1, \dots, c_6)$, source $o$, sink $d$, extra capacity $x_{\max}$
Ensure: Maximum throughput $\tau^\star_{od}$ for each $x = 0, 1, \dots, x_{\max}$
 1: $\mathbf{c}_{\text{current}} \leftarrow \mathbf{c}$
 2: Initialize MaxFlows $\leftarrow$ [ ]
 3: for $x = 0$ to $x_{\max}$ do
 4:     Compute maximum flow $\tau$ with $\mathbf{c}_{\text{current}}$
 5:     Append $\tau$ to MaxFlows
 6:     Find all minimum cuts $U_i^\star$
 7:     Pick the edge $e^\star$ that appears in the largest number of minimum cuts
 8:     Increase its capacity by 1: $\mathbf{c}_{\text{current}}[e^\star] \leftarrow \mathbf{c}_{\text{current}}[e^\star] + 1$
 9: end for
10: return MaxFlows
```

Explanation: At each step, we find the edge that is common to the most minimum cuts and increase its capacity. This is a simple heuristic to raise the maximum throughput.

With this approach, we obtained the graph presented in Figure 3. As you can notice, there is a recognizable pattern, which is derived from the network design. The first edge selected ($x = 1$) is e_5. For $x \geq 2$, the increased capacities alternate between e_1 and e_3. This is consistent with the theoretical idea that edges belonging to the shortest paths from o to d are more likely to appear as common edges in the minimum cuts. However, this is just a theoretical consideration and may not hold in every network.

Question 3

We consider the augmented network $\mathcal{G}'$ obtained by adding a new directed edge $e_8 = (o, d)$ with initial capacity $c_8 = 1$ and $x > 0$ extra units of capacity ($x \in Z_{\geq 0}$), as shown in Figure 4. Let $\mathcal{C}_{\min}$ denote the set of all minimum cuts of $\mathcal{G}'$ between o and d.

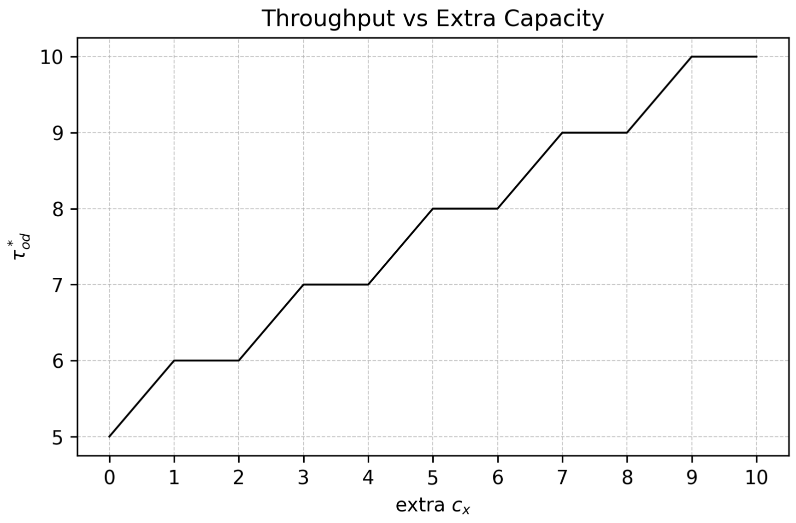

Figure 3: Maximum throughput τ^*_{od} as a function of extra capacity x in $\mathcal{G}$.

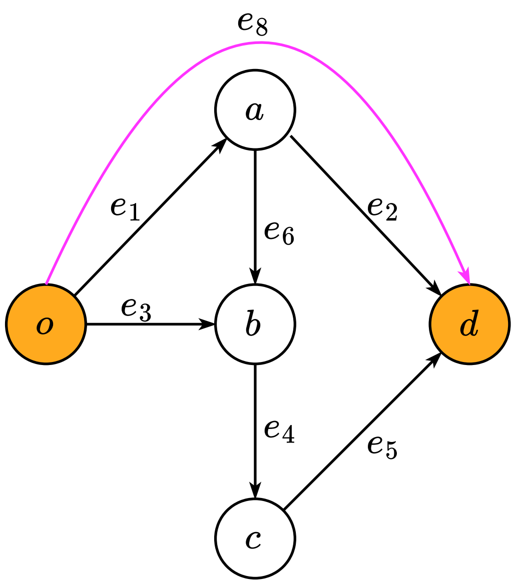

Figure 4: The augmented network $\mathcal{G}'$ with the added edge $e_8 = (o, d)$.

Observation: e_8 belongs to every minimum cut of $\mathcal{G}'$, i.e.

$$e_8 \in C, \quad \forall C \in \mathcal{C}_{\min}.$$

Hence, allocating extra capacity to e_8 directly increases the network's maximum flow. Let $\tau^*_{od}(x)$ denote the maximum flow from o to d in $\mathcal{G}'$ when

x extra units are added to e_8. Then:

$$\tau^*_{od}(x) = \tau^*_{od}(0) + x, \quad x \in Z_{\geq 0}.$$

By creating a direct link from o to d, e_8 bypasses all other bottlenecks. Every additional unit of capacity immediately increments the global maximum flow, producing a linear relation between x and τ^*_{od}, as shown in Figure 5.

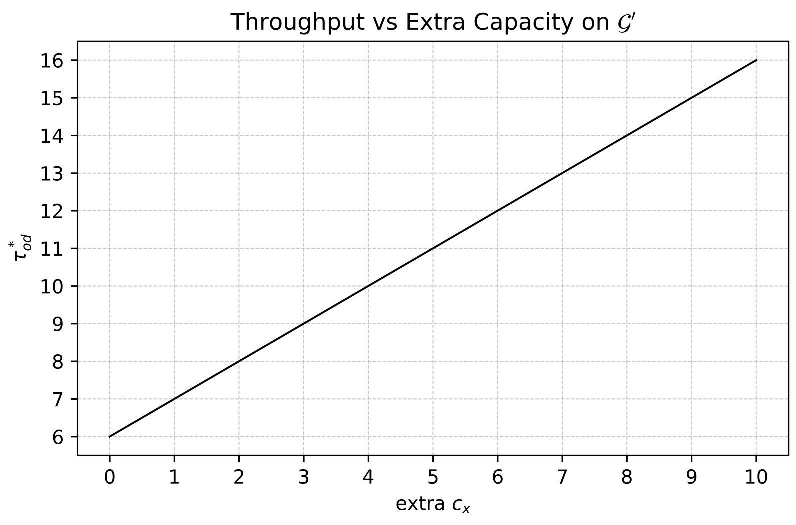

Figure 5: Maximum throughput τ^*_{od} as a function of extra capacity x in $\mathcal{G}'$.

Following the heuristic of distributing extra capacity to edges appearing most frequently in minimum cuts, e_8 is always the dominant edge in $\mathcal{G}'$. Therefore, the optimal strategy is to assign all extra capacity to e_8.

Exercise 2

Consider the simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in Figure 6.

Question 1. Compute the Katz centrality. Compute the Katz centrality vector for the given graph using $\beta = 0.15$ and a uniform intrinsic centrality $\boldsymbol{\mu}$.

Question 2. Distributed PageRank algorithm. Propose and describe a distributed algorithm to compute PageRank centrality with damping factor $\beta = 0.15$ and uniform intrinsic centrality μ.

Question 3. Explain the results. Explain the outcomes obtained in (a) and (b), focusing on the centrality values of nodes n_6 and n_9.

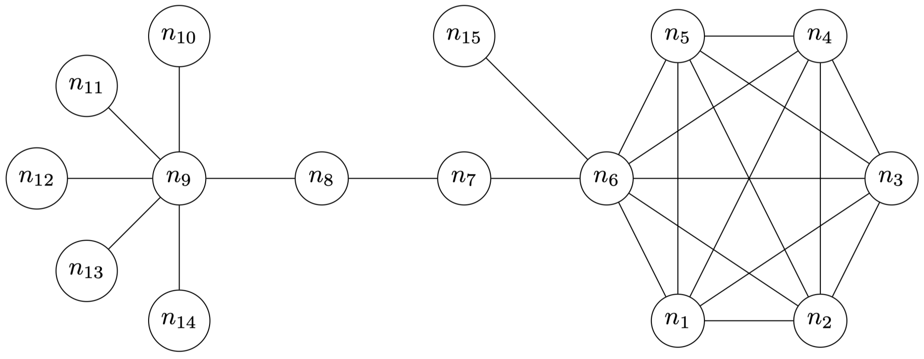

Figure 6: Exercise 2 graph

Question 4. Compute the Page-Rank centrality with uniform intrinsic centrality μ and for every

$$\beta \in \{0,\ 0.15,\ 0.5,\ 0.9,\ 1\}.$$

Is the difference between the centrality of node n_6 and node n_9 monotone in β ? Explain the results, focusing in particular on the extreme values of β .

Question 1

Katz Centrality Computation

We computed the Katz centrality vector for the given graph using $\beta = 0.15$ and a uniform intrinsic centrality vector $\boldsymbol{\mu}$. The computation was performed iteratively until convergence, following the standard recursive formula:

$$\mathbf{z} = \frac{1-\beta}{\lambda_{\max}}\mathbf{W}'\mathbf{z} + \beta\boldsymbol{\mu}$$

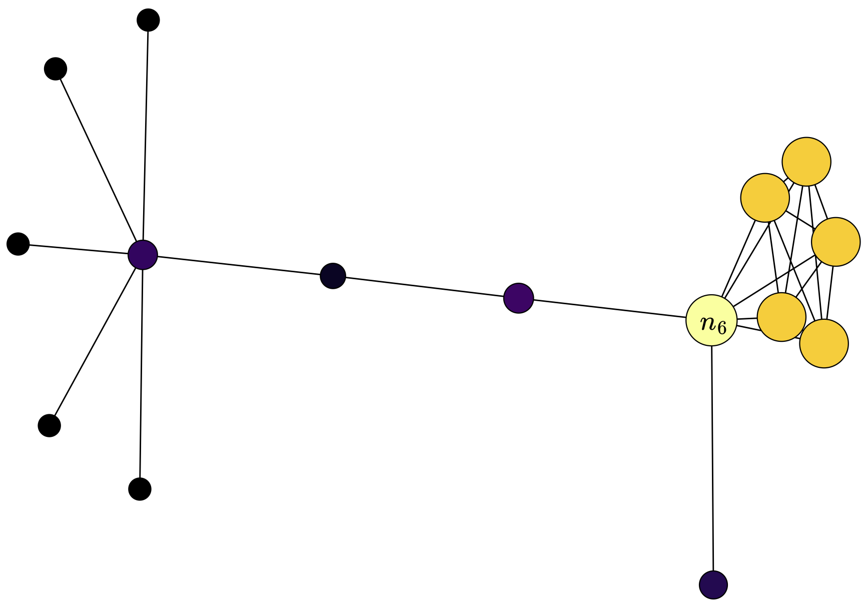

Figure 7: Katz centrality of all nodes as computed iteratively. The hotter the node color, the higher the centrality.

The resulting Katz centrality vector is:

$$\mathbf{z}_{\mathrm{Katz}} = \begin{bmatrix} 0.1181 \\ 0.1181 \\ 0.1181 \\ 0.1181 \\ 0.1181 \\ 0.1301 \\ 0.0442 \\ 0.0317 \\ 0.0428 \\ 0.0243 \\ 0.0243 \\ 0.0243 \\ 0.0243 \\ 0.0243 \\ 0.0389 \end{bmatrix}.$$

Question 2

Distributed PageRank Computation

For the PageRank (Bonacich) centrality with damping factor $\beta = 0.15$ and uniform intrinsic centrality $\boldsymbol{\mu}$, we employed a distributed iterative algorithm analogous to Katz centrality:

$$\mathbf{z} = (1 - \beta)\,\mathbf{P}^\top \mathbf{z} + \beta\,\boldsymbol{\mu}$$

with convergence monitored across all nodes.

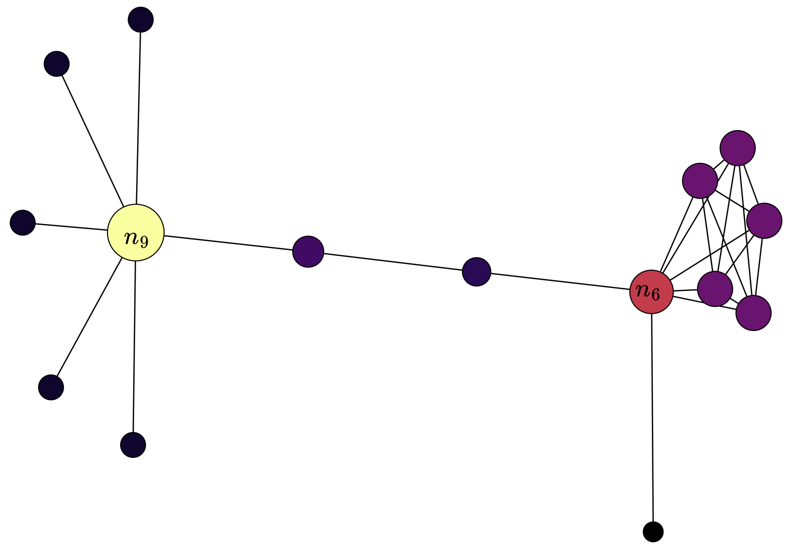

Figure 8: Bonacich centrality of all nodes as computed iteratively. The hotter the node color, the higher the centrality.

The resulting PageRank centrality vector is:

$$\mathbf{z}_{\text{PageRank}} = \begin{bmatrix} 0.0747 \\ 0.0747 \\ 0.0747 \\ 0.0747 \\ 0.0747 \\ 0.1144 \\ 0.0486 \\ 0.0582 \\ 0.1941 \\ 0.0375 \\ 0.0375 \\ 0.0375 \\ 0.0375 \\ 0.0375 \\ 0.0239 \end{bmatrix}.$$

Question 3

Katz Centrality — n_6 vs n_9

In this context, the most interesting nodes to analyze are node n_6 and node n_9, because both act as bridges connecting the main network to two different types of sink subgraphs. The structural difference is that node n_9 connects to a star-like subgraph, whereas node n_6 connects to a fully connected (clique-like) subgraph. This affects how influence propagates and accumulates through each node.

Katz centrality assigns importance to a node not only based on its direct neighbors, but also on how many longer walks can be initiated from it, while attenuating contributions from longer paths. Nodes receive an intrinsic baseline importance and gain additional value if they are connected to other highly influential nodes.

Starting from the classical Katz centrality formulation:

$$\mathbf{z} = \frac{1-\beta}{\lambda_{\max}} \mathbf{W}'\mathbf{z} + \beta\boldsymbol{\mu},$$

where $\boldsymbol{\mu}$ is the vector of intrinsic centralities and $\alpha = \frac{1-\beta}{\lambda_{\max}}$. Repeated substitution yields the infinite series:

$$\mathbf{z} = \beta(\mathbf{I} + \alpha\mathbf{W}' + \alpha^2(\mathbf{W}')^2 + \alpha^3(\mathbf{W}')^3 + \dots)\boldsymbol{\mu}.$$

This formulation shows that centrality depends on the cumulative influence transmitted through walks of different lengths, but decays as powers of α grow. Therefore, most of the centrality comes from the first few reachable neighborhoods.

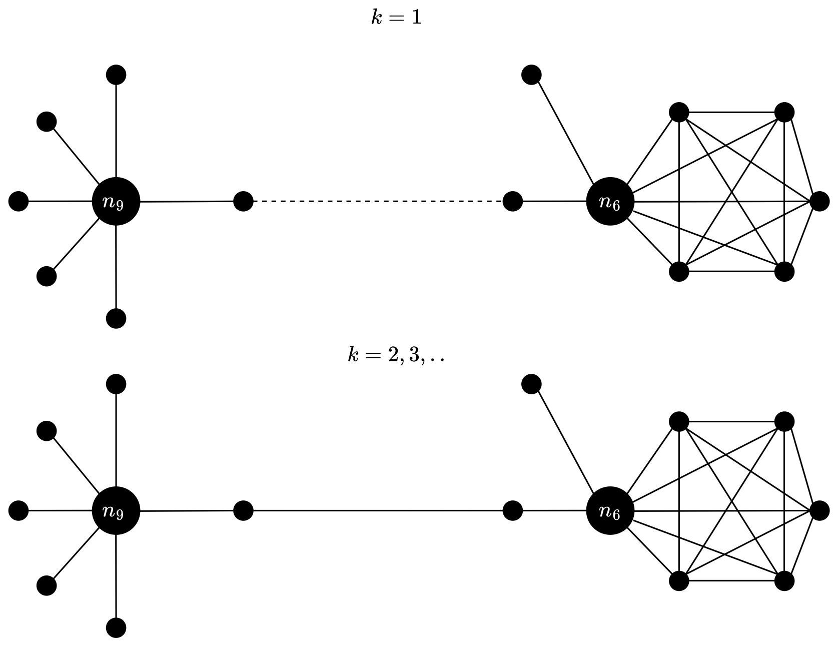

Figure 9: Neighborhoods of nodes n_6 and n_9. Top: nodes reachable in one step ($k = 1$). Bottom: nodes reachable in two or more steps ($k = 2, 3, \dots$). Node n_6 remains more connected, leading to higher Katz centrality.

Figure 9 illustrates the key difference. Although both nodes are gateways to a sink component, node n_6 links to a fully connected subgraph, generating many short reinforcing walks, while node n_9 leads to a star structure that does not mutually reinforce influence. As a result, node n_6 accumulates significantly more centrality than node n_9.

Moreover, because Katz centrality propagates influence through all neighbors, the nodes within the fully connected sink subgraph of node n_6 also achieve very high centrality, simply because they recursively reinforce one another.

Bonacich Centrality — n_6 vs n_9

As in the Katz case, the two most meaningful nodes to analyze are n_6 and n_9, since both serve as structural gateways between the main graph and two different sink subgraphs. However, Bonacich centrality relies on a different propagation mechanism. Instead of using the raw adjacency matrix, it employs the *distribution matrix* $\mathbf{P}$, defined as

$$P_{ji} = \frac{W_{ji}}{\sum_k W_{jk}},$$

which normalizes outgoing influence by the degree of the source node. The resulting centrality vector is computed as

$$\mathbf{z} = (1-\beta)\,\mathbf{P}^{\top}\mathbf{z} + \beta\,\boldsymbol{\mu}$$

The core interpretative principle is that *centrality received from a neighbor is inversely proportional to the number of nodes that the neighbor influences.* If a node has many outgoing connections, it must divide its influence among several destinations, thereby weakening the contribution each target receives. Conversely, if a node has only a single or very few outgoing connections, then it transfers nearly its entire influence to those limited neighbors, making such connections extremely valuable.

This effect becomes crucial when comparing n_6 and n_9. Node n_9 is linked to leaf nodes belonging to a star subgraph, each of which has very few outward connections. From the perspective of Bonacich centrality, this is a highly advantageous configuration: nearly all of the influence of those leaves flows exclusively toward n_9, rather than being dispersed among alternative paths. In contrast, although n_6 connects to a fully connected clique, the nodes in that clique distribute their influence among many neighbors. As a result, the contribution that each clique node transfers back toward n_6 becomes significantly diluted.

Therefore, while n_6 benefits in Katz centrality from the abundance of reinforcing walks emerging within a dense subgraph, Bonacich centrality rewards *exclusive* or *near-exclusive* links, where a neighbor has no alternative recipients for its influence. In essence, Bonacich centrality does not merely value connectivity but values **being the privileged recipient** of influence that is **not shared elsewhere**. For this reason, n_9 ultimately surpasses n_6 under Bonacich centrality.

Question 4

Bonacich centrality is a convex combination of network-based influence and an intrinsic component, with the relative weight controlled by β:

- $\beta = 0$: centrality is purely determined by the network structure.

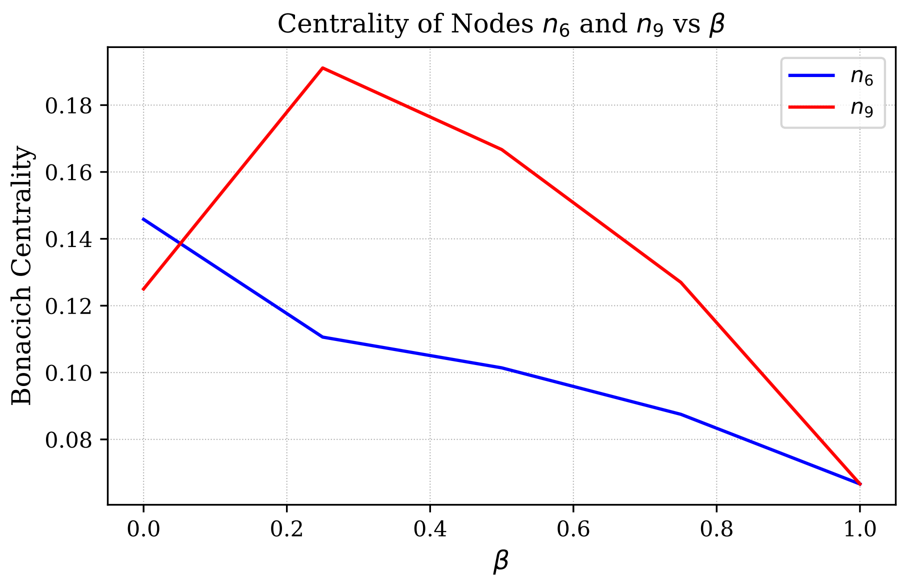

Figure 10: Bonacich centrality of nodes n_6 and n_9 as a function of β.

- $\beta \to 1$: centrality is dominated by the intrinsic vector $\boldsymbol{\mu}$, leading to almost uniform centrality across nodes.

4.1. Network-only regime ($\beta = 0$):

At $\beta = 0$, centrality relies solely on the network structure:

$$\mathbf{z} = \mathbf{P}^\top \mathbf{z}.$$

Node n_6 benefits from being part of a densely connected clique, which amplifies its centrality through recursive propagation, whereas n_9 connects to a star-like structure with limited mutual reinforcement. As illustrated in Figure 10, this topology gives n_6 a higher centrality than n_9.

4.2. Mixed regime ($0 < \beta < 1$):

Increasing β introduces the intrinsic component $\boldsymbol{\mu}$ into the centrality calculation:

$$\mathbf{z}^{(k+1)} = (1 - \beta)\mathbf{P}^\top \mathbf{z}^{(k)} + \beta \boldsymbol{\mu}.$$

Node n_9, acting as the bridge for several leaf nodes, gains relatively more from the intrinsic contribution, whereas n_6's influence is diluted across its densely connected neighbors. Figure 10 shows that this effect produces a crossover, where n_9 surpasses n_6 in centrality for intermediate β values.

4.3. Intrinsic-only regime ($\beta = 1$):

For $\beta = 1$, network effects vanish and centrality is entirely determined by the intrinsic vector:

$$\mathbf{z} = \boldsymbol{\mu}.$$

All nodes attain the same centrality, and Figure 10 illustrates this uniformity, confirming that $z_6 = z_9 = 1/|\mathcal{V}|$.

Exercise 3

We are given a portion of the highway network in Los Angeles, represented as a directed graph shown in Figure 11. Each node corresponds to an intersection between two highways. For each link $e_i \in \{e_1, \ldots, e_{28}\}$, we are given:

- Capacity c_e: maximum admissible traffic flow on link e (vehicles/hour).
- Free-flow travel time l_e: minimum travel time on link e under uncongested conditions.
- Delay (travel time) function $\tau_e(f_e)$: travel delay on link e as a function of the flow f_e.

$$\tau_e(f_e) = \begin{cases} \dfrac{l_e}{1 - f_e/c_e}, & 0 \leq f_e \leq c_e, \\ +\infty, & f_e > c_e. \end{cases}$$

Question 1. Compute the shortest path between nodes 1 and 17. Since the network is initially empty, this corresponds to the path with minimum free-flow travel time.

Question 2. Compute the maximum flow between nodes 1 and 17.

Question 3. Given a flow vector f, compute the external inflow vector ν satisfying the flow conservation constraint:

$$Bf = \nu,$$

where B is the node-arc incidence matrix.

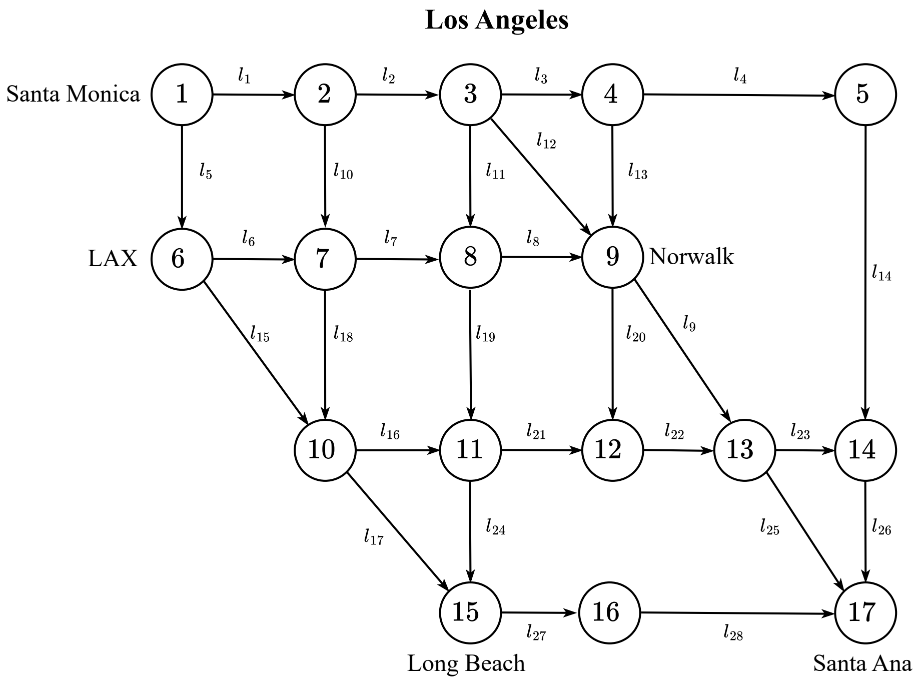

Figure 11: Approximation of a portion of the highway network in Los Angeles.

Question 4. Compute the social optimum flow f^* with respect to the delay function $\tau_e(f_e)$. Minimize the following total cost function:

$$\min_f \sum_{e\in\mathcal{E}} \left(\frac{l_e c_e}{1 - f_e/c_e} - l_e c_e \right)$$

under flow conservation constraints.

Question 5. Compute the Wardrop equilibrium $f^{(0)}$ by minimizing:

$$\min_f \sum_{e\in\mathcal{E}} \int_0^{f_e} \tau_e(s)\, ds.$$

Question 6. Introduce optimal tolls such that the toll on each link e is:

$$\omega_e = f_e^*\, \tau_e'(f_e^*),$$

where f_e^* is the system-optimal flow computed in Question 4. The effective delay becomes $\tau_e(f_e) + \omega_e$. Compute the new Wardrop equilibrium $f^{(\omega)}$ and compare it with f^*. What do you observe?

Question 7. Redefine the system cost as the total additional delay over free flow, given by:

$$\psi_e(f_e) = f_e\big(\tau_e(f_e) - l_e\big).$$

Compute the new system-optimal flow f^* under this cost definition. Construct tolls ω^* such that the Wardrop equilibrium $f(\omega^*)$ coincides with f^*, and verify the result.

Question 1

In principle, the problem can be reformulated as a network flow optimization task. More precisely, we consider the following linear program:

$$\min_f \; l^\top f$$

subject to

$$Bf = \nu, \qquad f \geq 0,$$

where $\nu = (\tau, 0, \ldots, 0, -\tau)^\top$ denotes the exogenous flow vector, and where the network is assumed to operate under uncongested conditions. Once the optimal flow f^* is obtained, the shortest path corresponds to the set of links for which the optimal flow is strictly positive.

Although the formulation above can be solved as a convex optimization problem, we also verified the result using the built-in shortest path functionality available in `networkx`, which relies on Dijkstra's algorithm. In particular, we solved both the linear program and the Dijkstra-based shortest path, and confirmed that the two approaches yield the same optimal route.

The resulting shortest path is illustrated in Figure 12. Note that, in this specific case, the optimal solution also happens to minimize the number of edges along the path, although this property is not guaranteed in general since edge lengths may differ.

Question 2

To compute the maximum flow between the specified origin and destination nodes, one could apply classical algorithms such as the Ford–Fulkerson method. However, `networkx` provides a built-in *maximum flow* solver, which we employed to obtain the result. The maximum flow value between the two terminals is equal to 22 448.

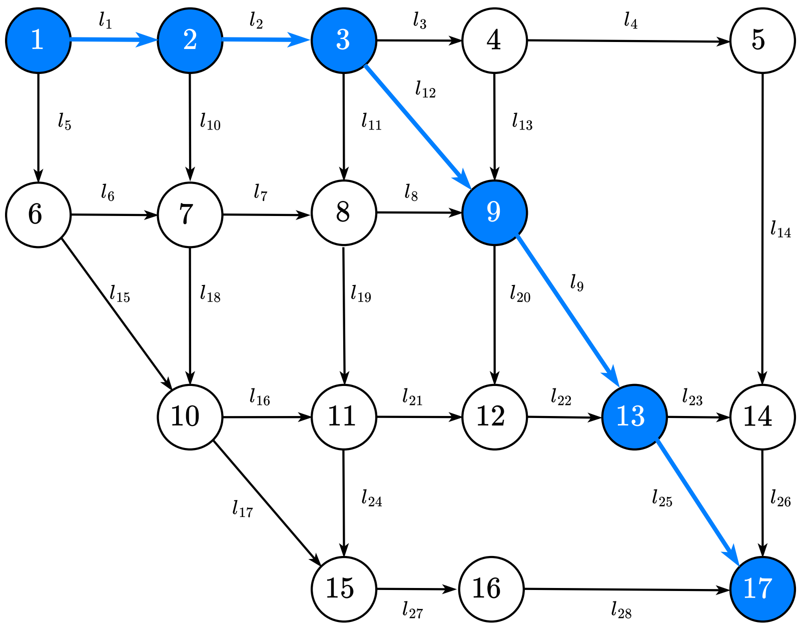

Figure 12: Shortest path connecting Santa Monica to Santa Ana.

Question 3

The external inflow vector can be computed using the node–edge incidence matrix as:

$$\nu = Bf.$$

By solving this linear system with the flow vector $f \in R^{28}$, we obtain the external inflow vector $\nu \in R^{17}$ as:

$$\nu = \begin{bmatrix} 16806 \\ 8570 \\ 19448 \\ 4957 \\ -746 \\ 4768 \\ 413 \\ -2 \\ -5671 \\ 1169 \\ -5 \\ -7131 \\ -380 \\ -7412 \\ -7810 \\ -3430 \\ -23544 \end{bmatrix}.$$

Question 4

To simplify the notation, let us define the feasible flow set as:

$$X := \{f \in R^{\varepsilon} \mid Bf = \nu,\ f \geq 0\}.$$

Using this definition, the problem can be expressed as the following flow optimization:

$$\min_{f \in X} \sum_{e \in \varepsilon} \psi_e(f_e),$$

where $\psi_e(f_e)$ denotes the cost associated with link e.

This formulation corresponds to a classical instance of the System Optimum Traffic Assignment Problem (SO-TAP). In particular, with the delay functions defined previously, the objective can be rewritten as:

$$\min_{f \in X} \sum_{e \in \varepsilon} \frac{f_e l_e}{1 - f_e/c_e}.$$

We solved this convex optimization problem using the `CVXPY` library. The resulting optimal flow $f^* \in R^{28}$ is reported in the accompanying code and is also illustrated in Figure 13.

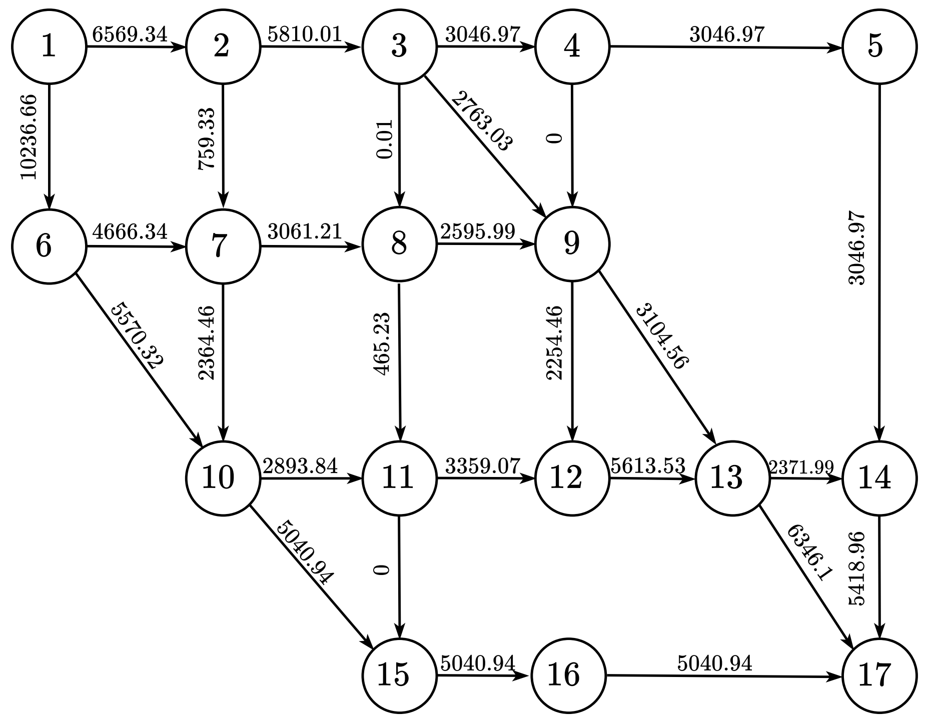

Figure 13: Graph representation of the highway network showing the optimal flows f^* on each edge.

Question 5

In the SO-TAP framework, users do not necessarily follow the system-optimal flow f^*. Instead, each user selects the path that minimizes their own travel delay, i.e., they behave selfishly and choose the path that minimizes the experienced delay.

This concept is formalized by the notion of Wardrop equilibrium, where no individual user can reduce their travel time by unilaterally changing routes.

The flow vector corresponding to the Wardrop equilibrium can be obtained by solving the following optimization problem:

$$\min_{f \in X} \sum_{e \in \varepsilon} l_e c_e \ln(c_e - f_e),$$

where $X = \{f \in R^{\varepsilon} \mid Bf = \nu,\ f \geq 0\}$ is the set of feasible flows.

The resulting Wardrop flow vector $f^{(0)}$ is reported in the accompanying code and is illustrated in Figure 14.

The Wardrop cost, defined as the total travel time under the equilibrium

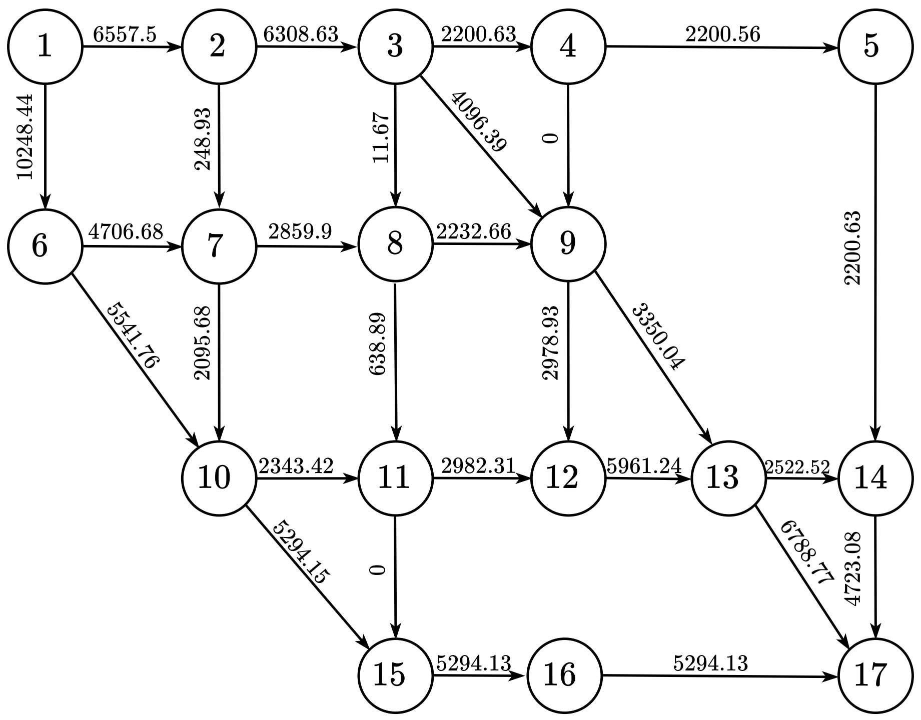

Figure 14: Flow vector at Wardrop equilibrium corresponding to the delay function $\tau_e(f_e) = \frac{l_e}{1-f_e/c_e}$, representing the travel time experienced by users individually.

flow $f^{(0)}$, is given by

$$C_{\text{Wardrop}} = \sum_{e \in E} f_e^{(0)}\, \tau_e\big(f_e^{(0)}\big) \approx 26\,495.34.$$

The Price of Anarchy (PoA), which measures the efficiency loss due to selfish routing compared to the system-optimal flow f^*, is computed as

$$\text{PoA} = \frac{C_{\text{Wardrop}}}{C_{\text{SO}}} \approx 1.0135,$$

where C_{SO} is the total cost at the system optimum.

Question 6

The idea behind introducing tolls is to make users internalize their negative externality. In other words, tolls directly influence selfish user behavior by making each user consider not only their experienced delay but also the cost imposed by the tolls.

With tolls, the optimization problem can be expressed as:

$$\min_{f \in X} \sum_{e \in E} \left[\int_0^{f_e} \tau_e(s)\, ds + \omega_e f_e \right],$$

where $X = \{f \in R^{\varepsilon} \mid Bf = \nu,\ f \geq 0\}$ is the set of feasible flows.

According to Corollary 4.3 of the lecture notes, if the toll on each link is set as

$$\omega_e = f_e^* \, \tau_e'(f_e^*),$$

then the resulting Wardrop equilibrium flow $f(\omega)$ coincides with the system-optimal flow f^*.

As expected, solving this problem using `CVXPY` produces a flow vector identical to the one obtained in Question 4, since the introduction of tolls $\omega_e = f_e^* \, \tau_e'(f_e^*)$ guides the selfish users towards the designed system-optimal flow.

Question 7

Social Optimum f^*

The social optimum flow f^* is obtained by solving the following optimization problem:

$$\min_{f \in X} \sum_{e \in E} \left(\frac{l_e}{1 - f_e/c_e} - l_e \right) f_e,$$

where $X = \{f \in R^{\varepsilon} \mid Bf = \nu,\ f \geq 0\}$ is the set of feasible flows.

Plugging this problem into `CVXPY`, we obtained the numerical solution, a vector in R^{28}, which is reported in the accompanying code and illustrated in Figure 15.

Constructing tolls ω^* to align Wardrop equilibrium with f^*

Without tolls, the Wardrop equilibrium did not coincide with the system-optimal flow, since users only experience the delay $\tau_e(f_e)$ individually.

To guide users toward the social optimum, we define the toll vector ω^* as the difference between the marginal social cost $\psi_e'(f_e)$ and the experienced delay $\tau_e(f_e)$:

$$\omega_e^* = \psi_e'(f_e^*) - \tau_e(f_e^*) = f_e^* \, \tau_e'(f_e^*) - \frac{l_e}{(1 - f_e^*/c_e)^2} + \frac{l_e}{1 - f_e^*/c_e}.$$

Using these tolls, the resulting Wardrop equilibrium $f(\omega^*)$ **coincides exactly with the social optimum f^*, because tolls make users internalize the negative externalities they impose on others.

The Price of Anarchy in this case is

$$\text{PoA} = \frac{C_{\text{Wardrop}}(\omega^*)}{C_{\text{SO}}} \approx 0.99999996,$$

showing that introducing the tolls successfully aligns selfish user behavior with the social optimum.

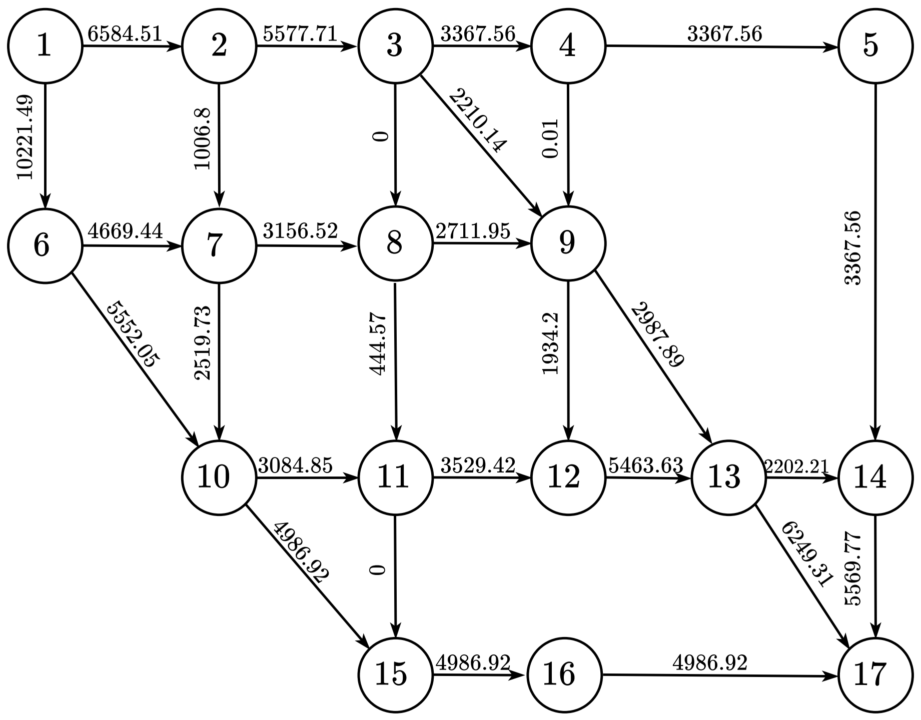

Figure 15: System-optimal flow f^* obtained using the constructed tolls ω^*, showing that the Wardrop equilibrium now coincides with the social optimum.