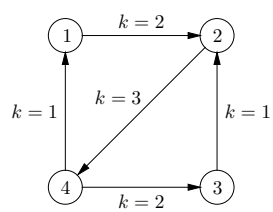
EE263 Homework 2

Fall 2023

2.140. Communication over a wireless network with time-slots. We consider a network with n nodes, labeled $1, \ldots, n$. A directed graph shows which nodes can send messages (directly) to which other nodes; specifically, an edge from node j to node i means that node j can transmit a message directly to node i. Each edge is assigned to one of K *time-slots*, which are labeled $1, \ldots, K$. At time period $t = 1$, only the edges assigned to time-slot 1 can transmit a message; at time period $t = 2$, only the edges assigned to time-slot 2 can transmit a message, and so on. After time period $t = K$, the pattern repeats. At time period $t = K + 1$, the edges assigned to time-slot 1 are again active; at $t = K + 2$, the edges assigned to time-slot 2 are active, etc. This cycle repeats indefinitely: when $t = mK + k$, where m is an integer, and $k \in \{1, \ldots, K\}$, transmissions can occur only over edges assigned to time-slot k. Although it doesn't matter for the problem, we mention some reasons why the possible transmissions are assigned to time-slots. Two possible transmissions are assigned to different time-slots if they would interfere with each other, or if they would violate some limit (such as on the total power available at a node) if the transmissions occurred simultaneously. A message or packet can be sent from one node to another by a sequence of transmissions from node to node. At time period t, the message can be sent across any edge that is active at period t. It is also possible to store a message at a node during any time period, presumably for transmission during a later period. If a message is sent from node j to node i in period t, then in period $t + 1$ the message is at node i, and can be stored there, or transmitted across any edge emanating from node i and active at time period $t + 1$. To make sure the terminology is clear, we consider the very simple example shown below, with $n = 4$ nodes, and $K = 3$ time-slots.



In this example, we can send a message that starts in node 1 to node 3 as follows:

- During period $t = 1$ (time-slot $k = 1$), store it at node 1.
- During period $t = 2$ (time-slot $k = 2$), transmit it to node 2.
- During period $t = 3$ (time-slot $k = 3$), transmit it to node 4.
- During period $t = 4$ (time-slot $k = 1$), store it at node 4.
- During period $t = 5$ (time-slot $k = 2$), transmit it to node 3.

You can check that at each period, the transmission used is active, *i.e.*, assigned to the associated time-slot. The sequence of transmissions (and storing) described above gets the message from node 1 to node 3 in 5 periods. Finally, the problem. We consider a specific network with $n = 20$ nodes, and $K = 3$ time-slots, with edges and time-slot assignments

given in `ts_data.json`. The labeled graph that specifies the possible transmissions and the associated time-slot assignments are given in a matrix $A \in \mathbb{R}^{n \times n}$, as follows:

$$A_{ij} = \begin{cases} k & \text{if transmission from node } j \text{ to node } i \text{ is allowed, and assigned to time-slot } k \\ 0 & \text{if transmission from node } j \text{ to node } i \text{ is never allowed} \\ 0 & i = j. \end{cases}$$

Note that we set $A_{ii} = 0$ for convenience. This choice has no significance; you can store a message at any node in any period. To illustrate this encoding of the graph, consider the simple example described above. For this example, we have

$$A_{\text{example}} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 3 & 0 & 0 \end{bmatrix}.$$

The problems below concern the network described in the data file and *not* the simple example given above.

a) *Minimum-time point-to-point routing.* Find the fastest way to get a message that starts at node 5, to node 18. Give your solution as a prescription ordered in time from $t = 1$ to $t = T$ (the last transmission), as in the example above. At each time period, give the transmission (as in 'transmit from node 7 to node 9') or state that the message is to be stored (as in 'store at node 13'). Be sure that transmissions only occur during the associated time-slots. You only need to give *one* prescription for getting the message from node 5 to node 18 in minimum time.

b) *Minimum time flooding.* In this part of the problem, we assume that once the message reaches a node, a copy is kept there, even when the message is transmitted to another node. Thus, the message is available at the node to be transmitted along any active edge emanating from that node, at any future period. Moreover, we allow multi-cast: if during a time period there are multiple active edges emanating from a node that has (a copy of) the message, then transmission can occur during that time period across *all* (or any subset) of the active edges. In this part of the problem, we are interested in getting a message that starts at a particular node, to all others, and we attach no cost to storage or transmission, so there is no harm is assuming that at each time period, every node that has the message forwards it to all nodes it is able to transmit to. What is the minimum time it takes before all nodes have a message that starts at node 7?

For both parts of the problem, you must give the specific solution, as well as a description of your approach and method.

3.260. Halfspace. Suppose $a, b \in \mathbb{R}^n$ are two given points. Show that the set of points in $\mathbb{R}^n$ that are closer to a than b is a halfspace, *i.e.*:

$$\{x \mid \|x - a\| \leq \|x - b\| \} = \{ x \mid c^\mathsf{T} x \leq d\}$$

for appropriate $c \in \mathbb{R}^n$ and $d \in \mathbb{R}$. Give c and d explicitly, and draw a picture showing a, b, c, and the halfspace.

3.350. Right inverses. This problem concerns the specific matrix

$$A = \begin{bmatrix} -1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

This matrix is full rank (*i.e.*, its rank is 3), so there exists at least one right inverse. In fact, there are many right inverses of A, which opens the possibility that we can seek right inverses that in addition have other properties. For each of the cases below, either find a specific matrix $B \in \mathbb{R}^{5\times 3}$ that satisfies $AB = I$ and the given property, or explain why there is no such B. In cases where there is a right inverse B with the required property, you must briefly explain how you found your B. You must also attach a printout of your Julia script or Jupyter notebook showing the verification that $AB = I$. (We'll be very angry if we have to type in your 5×3 matrix into Julia to check it.) When there is no right inverse with the given property, briefly explain why there is no such B.

a) The second row of B is zero.

b) The nullspace of B has dimension one.

c) The third column of B is zero.

d) The second and third rows of B are the same.

e) B is upper triangular, *i.e.*, $B_{ij} = 0$ for $i > j$.

f) B is lower triangular, *i.e.*, $B_{ij} = 0$ for $i < j$.

3.390. Some true/false questions. Determine if the following statements are true or false. No justification or discussion is needed for your answers. What we mean by "true" is that the statement is true for all values of the matrices and vectors given. You can't assume anything about the dimensions of the matrices (unless it's explicitly stated), but you can assume that the dimensions are such that all expressions make sense. For example, the statement "$A+B = B + A$" is true, because no matter what the dimensions of A and B (which must, however, be the same), and no matter what values A and B have, the statement holds. As another example, the statement $A^2 = A$ is false, because there are (square) matrices for which this doesn't hold. (There are also matrices for which it does hold, *e.g.*, an identity matrix. But that doesn't make the statement true.)

a) If all coefficients (*i.e.*, entries) of the matrix A are positive, then A is full rank.

b) If A and B are onto, then $A + B$ must be onto.

c) If A and B are onto, then so is the matrix $\begin{bmatrix} A & C \\ 0 & B \end{bmatrix}$.

d) If A and B are onto, then so is the matrix $\begin{bmatrix} A \\ B \end{bmatrix}$.

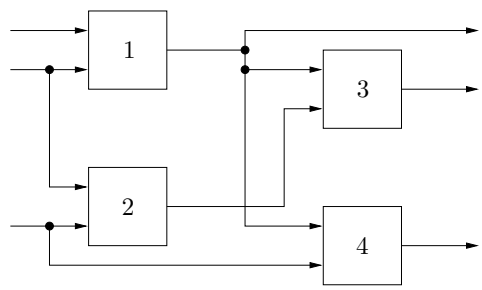
e) If the matrix $\begin{bmatrix} A \\ B \end{bmatrix}$ is onto, then so are the matrices A and B.

f) If A is full rank and skinny, then so is the matrix $\begin{bmatrix} A \\ B \end{bmatrix}$.

3.500. Digital circuit gate sizing. A digital circuit consists of a set of n (logic) gates, interconnected by wires. Each gate has one or more inputs (typically between one and four), and one output, which is connected via the wires to other gate inputs and possibly to some external circuitry. When the output of gate i is connected to an input of gate j, we say that gate i *drives* gate j, or that gate j is in the *fan-out* of gate i. We describe the topology of the circuit by the *fan-out list* for each gate, which tells us which other gates the output of a gate connects to. We denote the fan-out list of gate i as $\mathrm{FO}(i) \subseteq \{1, \ldots, n\}$. We can have $\mathrm{FO}(i) = \emptyset$, which means that the output of gate i does not connect to the inputs of any of the gates $1, \ldots, n$ (presumably the output of gate i connects to some external circuitry). It's common to order the gates in such a way that each gate only drives gates with higher indices, *i.e.*, we have $\mathrm{FO}(i) \subseteq \{i+1, \ldots, n\}$. We'll assume that's the case here. (This means that the gate interconnections form a directed acyclic graph.)

To illustrate the notation, a simple digital circuit with $n = 4$ gates, each with 2 inputs, is shown below. For this circuit we have

$$\mathrm{FO}(1) = \{3, 4\}, \quad \mathrm{FO}(2) = \{3\}, \quad \mathrm{FO}(3) = \emptyset, \quad \mathrm{FO}(4) = \emptyset.$$



The 3 input signals arriving from the left are called *primary inputs*, and the 3 output signals emerging from the right are called *primary outputs* of the circuit. (You don't need to know this, however, to solve this problem.)

Each gate has a (real) *scale factor* or *size* x_i. These scale factors are the design variables in the gate sizing problem. They must satisfy $1 \leq x_i \leq x^{\max}$, where $x^{\max}$ is a given maximum allowed gate scale factor (typically on the order of 100). The total area of the circuit has the form

$$A = \sum_{i=1}^{n} a_i x_i,$$

where a_i are positive constants.

Each gate has an *input capacitance* C_i^{in}, which depends on the scale factor x_i as

$$C_i^{\mathrm{in}} = \alpha_i x_i,$$

where α_i are positive constants.

Each gate has a *delay* d_i, which is given by

$$d_i = \beta_i + \gamma_i C_i^{\mathrm{load}} / x_i,$$

where β_i and γ_i are positive constants, and C_i^{load} is the *load capacitance* of gate i. Note that the gate delay d_i is always larger than β_i, which can be intepreted as the minimum possible delay of gate i, achieved only in the limit as the gate scale factor becomes large.

The load capacitance of gate i is given by

$$C_i^{\text{load}} = C_i^{\text{ext}} + \sum_{j \in \text{FO}(i)} C_j^{\text{in}},$$

where C_i^{ext} is a positive constant that accounts for the capacitance of the interconnect wires and external circuitry.

We will follow a simple design method, which assigns an equal delay T to all gates in the circuit, *i.e.*, we have $d_i = T$, where $T > 0$ is given. For a given value of T, there may or may not exist a feasible design (*i.e.*, a choice of the x_i, with $1 \leq x_i \leq x^{\max}$) that yields $d_i = T$ for $i = 1, \ldots, n$. We can assume, of course, that $T > \max_i \beta_i$, *i.e.*, T is larger than the largest minimum delay of the gates.

Finally, we get to the problem.

a) Explain how to find a design $x^\star \in \mathbb{R}^n$ that minimizes T, subject to a given area constraint $A \leq A^{\max}$. You can assume the fanout lists, and all constants in the problem description are known; your job is to find the scale factors x_i. Be sure to explain how you determine if the design problem is feasible, *i.e.*, whether or not there is an x that gives $d_i = T$, with $1 \leq x_i \leq x^{\max}$, and $A \leq A^{\max}$.

   Your method can involve any of the methods or concepts we have seen so far in the course. It can also involve a simple search procedure, *e.g.*, trying (many) different values of T over a range.

   Note: this problem concerns the general case, and not the simple example shown above.

b) Carry out your method on the particular circuit with data given in the file `gate_sizing_data.json`

   The fan-out lists are given as an $n \times n$ matrix `F`, with i, j entry one if $j \in \text{FO}(i)$, and zero otherwise. In other words, the ith row of `F` gives the fanout of gate i. The jth entry in the ith row is 1 if gate j is in the fan-out of gate i, and 0 otherwise.

Comment. You do not need to know anything about digital circuits; *everything* you need to know is stated above.