Single-Source Shortest Paths as a Linear Program

CSCI 4113/6101
Instructor: Norbert Zeh
September 9, 2025

Our primary use of linear programs in this course is as a tool to model combinatorial optimization problems. At first, you may find these two concepts rather incompatible: What do things like shortest paths, minimum spanning trees, vertex covers or independent sets have to do with assigning real values to variables? The trick is to build LPs whose variables are associated with parts of the combinatorial objects we try to model. Most of the problems we try to model are subset selection problems: A vertex cover or independent set is a subset of the vertices of the graph; our goal is to select the right subset. A minimum spanning tree is defined by its set of edges, so our goal is to select the right subset of edges of the graph. We can model these problems by associating a variable with every vertex (for the vertex cover and independent set problems) or with every edge (for the minimum spanning tree problem). We constrain each variable to be between 0 and 1. If a solution assigns a value of 0 to a variable, then the corresponding vertex or edge is not part of the solution; if the variable has value 1, then the vertex or edge *is* part of the solution. We add appropriate constraints to the LP to ensure that the set of vertices or edges defined in this way by a feasible solution is indeed a vertex cover, independent set or minimum spanning tree. You may wonder what happens when a variable has a value strictly between 0 and 1. Is the vertex or edge in the solution or not? That's a great question with deep consequences, which we will explore in the next topic.

In this topic, we will practice modelling combinatorial problems as linear programs using the the single-source shortest paths problem (SSSP) as an example. For this problem, we will associate a variable x_v with every vertex v of the graph. Our goal is to define the right constraints so that in an optimal solution $\hat{x}$ of the LP, $\hat{x}_v$ is the distance from the source vertex s to v.

Let us start by reviewing the necessary graph-theoretic definitions. Given a graph $G = (V, E)$, a **walk** from a vertex s to a vertex t is a sequence of vertices $P = \langle v_0, \ldots, v_k \rangle$ such that $v_0 = s$, $v_k = t$, and there exists an edge $e_i = (v_{i-1}, v_i) \in E$ for all $1 \leq i \leq k$. This sequence P is a **path** if $v_i \neq v_j$, for all $i \in [k]_0$ and $j \in [k]_0 \setminus \{i\}$ (see Fig. 1a). Depending on the context, P may be viewed as the sequence of its vertices, the sequence of its edges $\langle e_1, \ldots, e_k \rangle$ or the alternating sequence of both $\langle v_0, e_1, v_1, \ldots, e_k, v_k \rangle$. A walk $P = \langle v_0, \ldots, v_k \rangle$ is a **cycle** if $v_0 = v_k$ (see Fig. 1b). An **edge-weighted graph** is a pair (G, w) consisting of a graph $G = (V, E)$ and an assignment $w : E \to \mathbb{R}$ of weights to its edges. The **weight** or **length** of a walk P in an edge-weighted graph (G, w) is the total weight of all edges in P: $w(P) = \sum_{e \in P} w_e$ (see Fig. 1a). A **shortest walk** from a vertex $s \in V$ to a vertex $t \in V$ is a walk $\Pi_{G,w}(s, t)$ from s to t in G such that every walk P from s to t in G satisfies $w(P) \geq w(\Pi_{G,w}(s, t))$. The **distance** from s to t is $\text{dist}_{G,w}(s, t) = w(\Pi_{G,w}(s, t))$.

Why did we just define the distance between two vertices as the length of a shortest *walk*, not of a shortest *path*? The condition that a path cannot visit a vertex more than once is difficult to model both

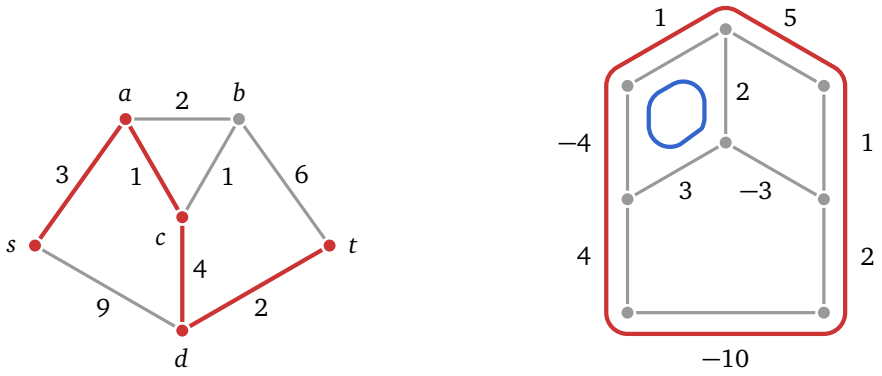

Figure 1: (a) The red path $\langle s, a, c, d, t\rangle$ is a shortest path from s to t, of length 10. $\langle s, d, t\rangle$ is a path from s to t but is not a shortest path because its length is 11. (b) The red and blue paths are cycles. The red one is a negative cycle of weight -1. The blue one is not a negative cycle; its weight is 2.

in combinatorial algorithms for computing shortest paths and in linear programs. It is much easier to look for shortest *walks* between vertices. However, shortest walks in an edge-weighted graph (G, w) are well defined only if there are no negative cycles in (G, w), where a **negative cycle** is a cycle C in G with $w(C) < 0$ (see Fig. 1b). Indeed, if there exists a walk from s to t that includes a vertex in some negative cycle C, then it is possible to generate arbitrarily short walks from s to t by including sufficiently many passes through C in the path. Thus, we will assume that there are no negative cycles in (G, w). Under this assumption, we have

Lemma 1. *If (G, w) is an edge-weighted graph without negative cycles, then there there exists a path from s to t of length $\mathrm{dist}_{G,w}(s, t)$, for any two vertices s and t. This path is a shortest path.*

Proof. It suffices to show that there exists a path P from s to t of length $\mathrm{dist}_{G,w}(s, t)$. This immediately implies that P is a shortest path because every path is a walk, that is, there cannot exist a path from s to t of length less than $\mathrm{dist}_{G,w}(s, t)$.

Consider a shortest walk $\Pi_{G,w}(s, t) = \langle v_0, \ldots, v_k\rangle$ from s to t in (G, w). Assume that $\Pi_{G,w}(s, t)$ contains the fewest vertices among all shortest walks from s to t. If $\Pi_{G,w}(s, t)$ is a path, then it is the path P we are looking for. So assume that $\Pi_{G,w}(s, t)$ is not a path, that is, there exist indices $i, j \in [k]_0$ with $i < j$ and $v_i = v_j$. The walk $C = \langle v_i, \ldots, v_j\rangle$ is a cycle. Since there are no negative cycles in (G, w), its weight is non-negative. Therefore, the walk $Q = \langle v_0, \ldots, v_i, v_{j+1}, \ldots, v_k\rangle$ is a walk from s to t of length $w(Q) = w(\Pi_{G,w}(s, t)) - w(C) \leq \mathrm{dist}_{G,w}(s, t)$, that is, $w(Q)$ is a shortest walk from s to t. However, Q contains fewer vertices than $\Pi_{G,w}(s, t)$, contradicting the choice of $\Pi_{G,w}(s, t)$. Thus, $\Pi_{G,w}(s, t)$ must be a path. □

The single-source shortest paths problem is to compute the distance from some source vertex s to every vertex $v \in V$:

Problem 2 (Single-source shortest paths, SSSP)**.** *Given an edge-weighted graph (G, w) and a source vertex $s \in V$, compute $\mathrm{dist}_{G,w}(s, v)$, for all $v \in V$.*

In general, the goal is to find the actual shortest paths $\Pi_{G,w}(s, v)$, not only the distances $\mathrm{dist}_{G,w}(s, v)$.

Exer. 2 asks you to show that computing the distances is the hard part of the problem.

The LP formulation of the SSSP problem is based on the following property of distances from s:

Observation 3. *For every edge (u,v) in an edge-weighted graph (G,w), we have*

$$\mathrm{dist}_{G,w}(s,v) \le \mathrm{dist}_{G,w}(s,u) + w_{u,v}.$$

Moreover, the absence of negative cycles in (G,w) implies that

Observation 4. $\mathrm{dist}_{G,w}(s,s) = 0$.

Thus, if we want to design an LP with variables x_v, for all $v \in V(G)$, whose optimal solution $\hat{x}$ satisfies $\hat{x}_v = \mathrm{dist}_{G,w}(s,v)$, then we should impose the constraints

$$\begin{aligned} &x_s = 0, \\ &x_v \le x_u + w_{u,v} \quad \forall (u,v) \in E(G). \end{aligned} \tag{1}$$

What's the objective function? To answer this question, it helps to focus on undirected graphs for now and visualize the graph as a model consisting of beads representing vertices and strings between these beads representing the edges. The length of the string between two beads u and v is $w_{u,v}$. Now imagine holding this model by the bead s and another bead v and pulling these two beads apart as far as you can. How far apart can you pull them? Exactly $\mathrm{dist}_{G,w}(s,v)$ because you will end up pulling all strings on the path $\Pi_{G,w}(s,v)$ tight, and then you cannot pull any further. If you simply hold up the entire model by the bead s and let gravity do the pulling for you, then every vertex v will hang exactly $\mathrm{dist}_{G,w}(s,v)$ lower than s. This is illustrated in Fig. 2.

Now it takes just a little extra effort to realize that pulling every vertex as far down below s as far as possible and assigning the resulting distance from s to each vertex to the variable x_v maximizes the sum

$$\sum_{v \in V(G)} x_v.$$

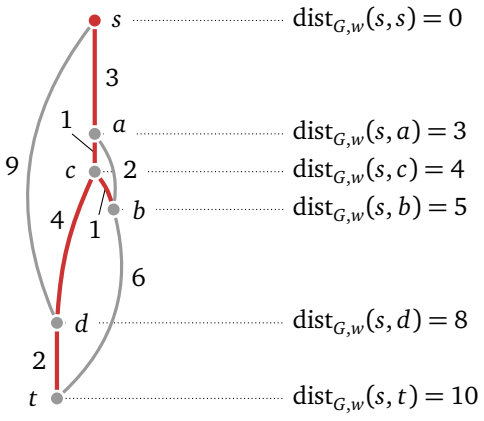

Figure 2: Illustration of the single-source shortest paths problem using the beads-and-string model. Holding the model from the vertex s makes each vertex hang the indicated distance below s.

Thus, we obtain the following LP formulation of the SSSP problem:

$$\begin{aligned} &\text{Maximize} \sum_{v \in V} x_v \\ \text{s.t.} \quad & x_s = 0 \\ & x_v - x_u \le w_{u,v} \quad \forall (u,v) \in E \end{aligned} \tag{2}$$

It may seem strange at first that a *shortest* paths problem is expressed as a maximization LP, but our beads-and-string model explains exactly why this is: the shortest path $\Pi_{G,w}(s,v)$ limits how far we can pull v away from s, so maximizing $\hat{x}_v$ without "overstretching" the strings, as expressed by the constraints that $x_v - x_u \le w_{u,v}$, ensures that $\hat{x}_v = \text{dist}_{G,w}(s,v)$. The next proposition proves this formally:

PROPOSITION 5. *The optimal solution $\hat{x}$ of the LP (2) satisfies $\hat{x}_v = \text{dist}_{G,w}(s,v)$, for all $v \in V$.*

Proof. Consider the subgraph $H = (V, E') \subseteq G$ such that

$$E' = \{(u,v) \in E \mid \hat{x}_v - \hat{x}_u = w_{u,v}\},$$

let $S \subseteq V$ be the set of all vertices reachable from s in H, and let $T = V \setminus S$ (see Fig. 3a). First we prove that $T = \emptyset$, that is, that every vertex is reachable from s in H.

Let

$$C = \{(u,v) \in E \mid u \in S, v \in T\}.$$

Since every edge $(u,v) \in C$ has the property that u is reachable from s in H but v is not, it must satisfy $(u,v) \notin E'$ and, thus, $\hat{x}_v - \hat{x}_u < w_{u,v}$. Therefore,

$$\delta = \min\{w_{u,v} - (\hat{x}_v - \hat{x}_u) \mid (u,v) \in C\} > 0.$$

Now define a new solution $\tilde{x}$ as

$$\tilde{x}_v = \begin{cases} \hat{x}_v & \text{if } v \in S \\ \hat{x}_v + \delta & \text{if } v \in T \end{cases}.$$

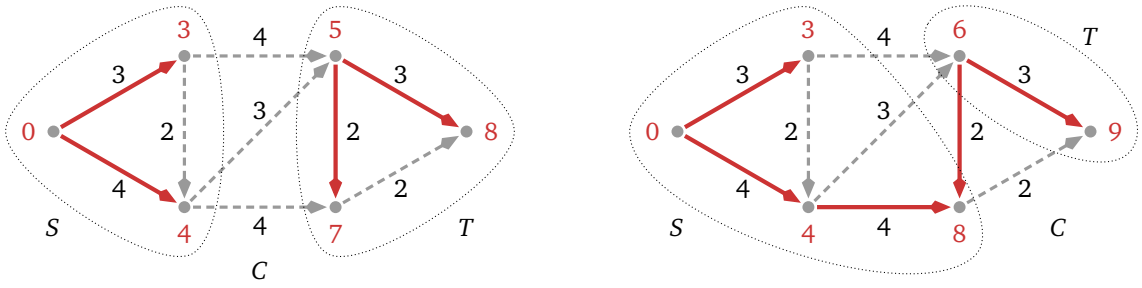

Figure 3: (a) The definition of the sets S and T in the proof of Prop. 5 corresponding to the solution $\hat{x}$ represented by the red vertex labels. Black edge labels are the edge lengths. The graph H contains all tight edges (red). (b) A solution $\tilde{x}$ with higher objective function value. The new set S of reachable vertices is bigger.

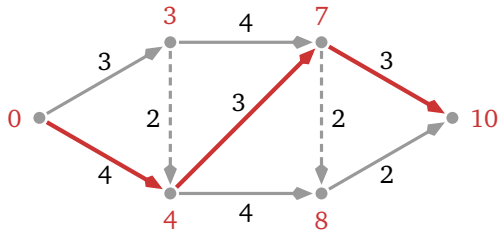

Figure 4: An optimal solution to (2) for the graph in Fig. 3a. Solid edges are tight, so $S = V$. The red path is a shortest path from the leftmost to the rightmost vertex.

This is illustrated in Fig. 3b. Observe that $s \in S$, so $\tilde{x}_s = \hat{x}_s = 0$ because $\hat{x}$ satisfies (2). For every edge $(u,v) \in E$, if $v \in S$, then $\tilde{x}_v = \hat{x}_v$ and $\tilde{x}_u \geq \hat{x}_u$. Thus, $\tilde{x}_v - \tilde{x}_u \leq \hat{x}_v - \hat{x}_u \leq w_{u,v}$. If $u, v \in T$, then $\tilde{x}_v = \hat{x}_v + \delta$ and $\tilde{x}_u = \hat{x}_u + \delta$. Thus, $\tilde{x}_v - \tilde{x}_u = \hat{x}_v - \hat{x}_u \leq w_{u,v}$. Finally, if $u \in S$ and $v \in T$, then $\tilde{x}_v = \hat{x}_v + \delta$ and $\tilde{x}_u \geq \hat{x}_u$. Moreover, $(u,v) \in C$, so $\delta \leq w_{u,v} - (\hat{x}_v - \hat{x}_u)$. Therefore, $\tilde{x}_v - \tilde{x}_u \leq \hat{x}_v - \hat{x}_u + \delta \leq w_{u,v}$. This shows that $\tilde{x}$ is a feasible solution of (2).

Next observe that $\sum_{v \in V} \tilde{x}_v - \sum_{v \in V} \hat{x}_v = \delta |T|$. Since $\tilde{x}$ is a feasible solution and $\hat{x}$ is an optimal solution of (2), we have $\sum_{v \in V} \tilde{x}_v - \sum_{v \in V} \hat{x}_v \leq 0$, that is, $T = \emptyset$ and $S = V$.

The proposition now follows if every vertex $v \in S$ satisfies $\hat{x}_v = \mathrm{dist}_{G,w}(s,v)$. Let $P = \langle v_0, \ldots, v_k \rangle$ be a path from s to v in H (see Fig. 4). Then $\hat{x}_{v_0} = \hat{x}_s = 0$ and, for $1 \leq i \leq k$, $\hat{x}_{v_i} = \hat{x}_{v_{i-1}} + w_{v_{i-1},v_i}$ because P is a path in H. Thus, $\hat{x}_v = \hat{x}_{v_k} = \sum_{i=1}^{k} w_{v_{i-1},v_i} = w(P)$. Since P is also a path from s to v in $G \supseteq H$, this implies that

$$\hat{x}_v \geq \mathrm{dist}_{G,w}(s,v). \tag{3}$$

Conversely, for the shortest path $\Pi_{G,w}(s,v) = \langle u_0, \ldots, u_\ell \rangle$ from s to v in G, $\hat{x}_{u_0} = \hat{x}_s = 0$ and, since $\hat{x}$ is a feasible solution of (2), $\hat{x}_{u_i} \leq \hat{x}_{u_{i-1}} + w_{u_{i-1},u_i}$, for all $1 \leq i \leq \ell$, that is,

$$\hat{x}_v = \hat{x}_{u_\ell} \leq \sum_{i=1}^{\ell} w_{u_{i-1},u_i} = \mathrm{dist}_{G,w}(s,v). \tag{4}$$

Together, (3) and (4) show that $\hat{x}_v = \mathrm{dist}_{G,w}(s,v)$. This finishes the proof. □

Exercises

Exercise 1. *In general, there may be more than one shortest path between two vertices s and t in a connected edge-weighted graph (G,w). Prove that it is possible to choose a particular shortest path $\Pi_{G,w}(s,v)$ for each vertex $v \in V$ such that the union of these shortest paths is a tree. This tree is called a* ***shortest path tree****.*

Exercise 2. *Show that finding the shortest paths from a vertex s to all vertices v in an edge-weighted graph (G,w) is no harder than computing the distances from s to all vertices in G. Specifically, if $d : V \to \mathbb{R}$ is a labelling of the vertices of G such that $d_v = \mathrm{dist}_{G,w}(s,v)$ for all $v \in V$, show how to compute a shortest path tree T with root s as in Exer. 1 in $O(n+m)$ time.*