Single-Source Shortest Paths II

Chapter 24

Lecture Overview

- Dijkstra's Algorithm
- Linear Programming Introduction
- Difference Constraint Systems
- A Triangle Inequality for Shortest Paths

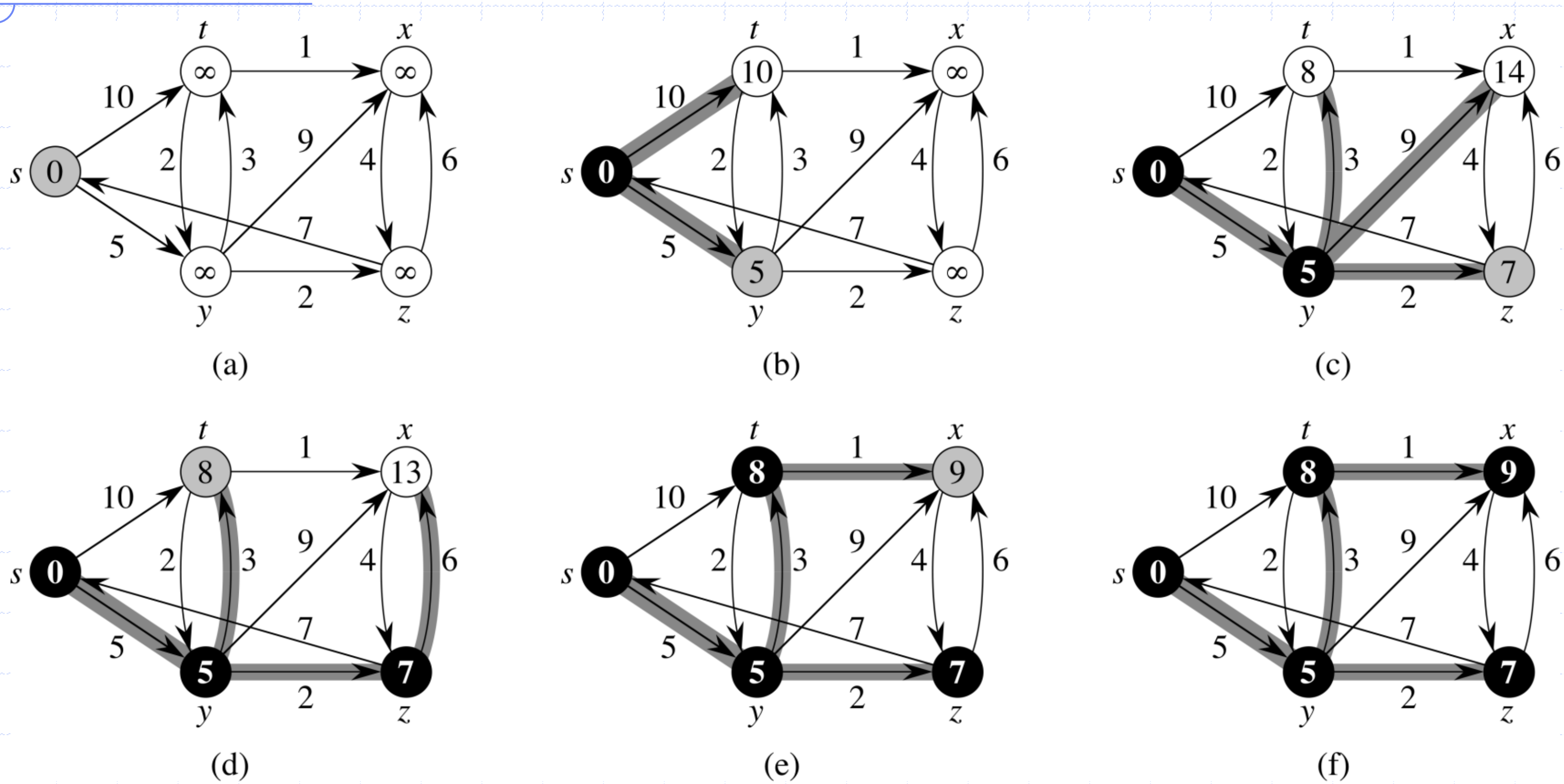
Dijkstra's Algorithm

Dijkstra's algorithm is an algorithm for SSSP in graphs with positive edge weights only. It maintains a set S of vertices whose shortest-path weights from s have been determined.

```
DIJKSTRA(G, s)

INITIALIZE-SINGLE-SOURCE(G, s)
S = Ø
Q = G.vertices                  // key is vertex.d
while !Q.IS-EMPTY()
    u = Q.EXTRACT-MIN()
    S = S + u
    for each vertex v in Adj[u]
        RELAX(u, v)             // and if v.d changes,
                                //     Q.DECREASE-KEY(v)
```

Dijkstra's Algorithm Example



(a) (b) (c) (d) (e) (f)

Dijkstra's Algorithm Analysis

DIJKSTRA(G, s)

INITIALIZE-SINGLE-SOURCE(G, s)	O(V)		
S = Ø	O(1)		
Q = G.vertices	O(V) heapify		
while !Q.IS-EMPTY()	O(V) iterations		
u = Q.EXTRACT-MIN()		O(log V)	O(V log V)
S = S + u		O(1)	
for each vertex v in Adj[u]	O(E) total iterations		
RELAX(u, v)		O(1)	
if(v.d decreased)		O(1)	~~O(E log V)~~
Q.DECREASE-KEY(v)	O(1) amort.	~~O(log V)~~	O(E)

~~O((E+V) log V)~~

using Fibonacci Heap to have a better amortized DECREASE-KEY

O(E + V log V)

Dijkstra's Algorithm Correctness

Dijkstra's algorithm is a greedy algorithm; it always chooses the lightest edge leading out of S.

Theorem. When Dijkstra's algorithm is run on a weighted, directed graph $G = (V, E)$ with nonnegative edge weights w and a source vertex s, it will terminate with $u.d = \delta(s, u)$ for every vertex u in V.

Proof. Use the following invariant:

At the start of each iteration of the **while** loop of lines 4-8, $v.d = \delta(s, v)$ for all vertices v in S.

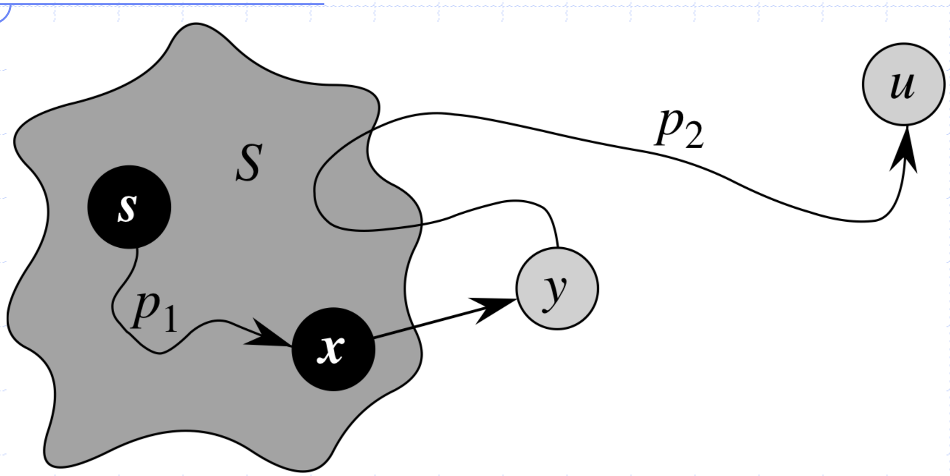
Dijkstra's Algorithm Correctness

At the start of each iteration of the **while** loop of lines 4-8, $v.d = \delta(s, v)$ for all vertices v in S.

Initialization. At the start of the first iteration of the while loop, S is empty, so the invariant holds.

Maintenance. For a contradiction, assume u is the first vertex for which $u.d \neq \delta(s, u)$ when it is added to S. The vertex $u \neq s$, for s is added in the first iteration at a time when $s.d = 0$. There must be a path from s to u else $u.d = \infty = \delta(s, u)$. Let p be a shortest path from s to u. Let y be the first vertex of $V - S$ along p, and x its predecessor.

Dijkstra's Algorithm Correctness



The path p can be broken into p_1 from s to x, the edge xy, and p_2 from y to u. (p_1 and/or p_2 may be empty).

The subpath from s to y must be a shortest path by optimal substructure. Thus, when u is added, y.d = δ(s, y). If y = u, this contradicts our choice of u. If y ≠ u, then δ(s, u) > δ(s, y), but then u.d ≥ δ(s, u) > δ(s, y) = y.d, meaning y would've been added to S first, a contradiction. ■

Dijkstra's Algorithm Correctness

Corollary. When Dijkstra's algorithm is run on a weighted, directed graph G = (V, E) with nonnegative edge weights w and a source vertex s, it will terminate with the predecessor subgraph being a shortest paths tree rooted at s.

Linear Programming with Difference Constraints

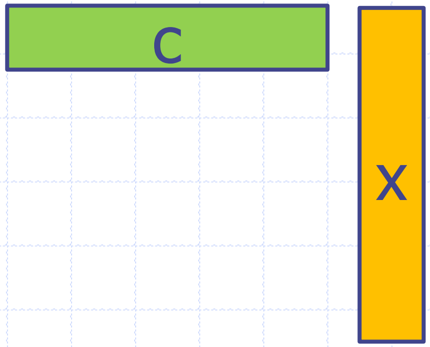
Later in the course we will study the full linear programming problem, where we wish to optimize a linear function that is subject to a set of linear constraints.

Here we study a special type of linear program, one where the constraints are known as difference constraints.

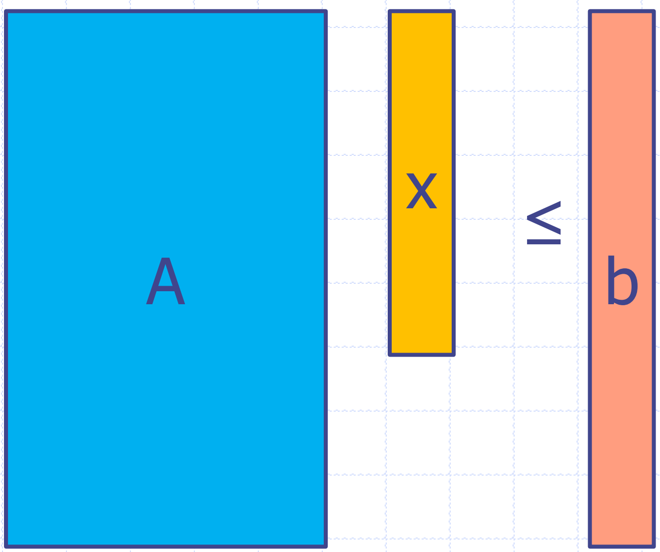
In the general problem, we are given an $m \times n$ matrix A, an m-vector b, and an n-vector c. We wish to find a vector x of n elements which maximizes the objective function $c \cdot x$ subject to the m constraints given by $Ax \leq b$.

Linear Programming

max.



subject to:



max. $2x_1 + 3x_2 - 5x_3$

subject to:

$$x_1 + 2x_2 \le 29$$
$$3x_1 - 4x_3 \le 12$$
$$-x_1 + x_2 + 2x_3 \le -8$$
$$4x_1 - x_2 + 3x_3 \le 0$$

Linear Programming

max. $\begin{bmatrix} 2 & 3 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

subject to:

$$\begin{bmatrix} 1 & 2 & 0 \\ 3 & 0 & -4 \\ -1 & 1 & 2 \\ 4 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 29 \\ 12 \\ -8 \\ 0 \end{bmatrix}$$

max. $2x_1 + 3x_2 - 5x_3$

subject to:

$$\begin{aligned} x_1 + 2x_2 \quad & \leq 29 \\ 3x_1 \quad - 4x_3 & \leq 12 \\ -x_1 + x_2 + 2x_3 & \leq -8 \\ 4x_1 - x_2 + 3x_3 & \leq 0 \end{aligned}$$

Difference Constraints

In a difference constraint, the corresponding row of A has one entry that is a 1, one that is a -1, and the rest of the entries 0. A system of difference constraints has all rows of A being difference constraints.

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leq \begin{bmatrix} 3 \\ 7 \\ 5 \\ -2 \end{bmatrix}$$

$$x_1 - x_2 \leq 3$$

$$x_2 - x_4 \leq 7$$

$$x_3 - x_1 \leq 5$$

$$x_1 - x_4 \leq -2$$

Difference Constraints

$x_1 - x_2 \le 3$
$x_2 - x_4 \le 7$
$x_3 - x_1 \le 5$
$x_1 - x_4 \le -2$

This system has a solution
$x = (x_1, x_2, x_3, x_4) = (4, 2, 0, 6)$
and another solution
$x = (8, 6, 4, 10)$

Note that $(8, 6, 4, 10) - (4, 2, 0, 6) = (4, 4, 4, 4)$.
This can be generalized.

Lemma. Let $x = (x_1, x_2, \ldots, x_n)$ be a solution to a system $Ax \le b$ of difference constraints, and let d be any constant. Then $x + d = (x_1+d, x_2+d, \ldots, x_n+d)$ is also a solution to $Ax \le b$.

Difference Constraints

Proof. For any i and j, $(x_i+d) - (x_j+d) = x_i - x_j$. Thus x+d satisfies any difference constraint that x does. ■

Difference constraints arise in many applications. One of the most common is when the unknowns x_i are times of certain events. Then a difference constraint of $x_i - x_j \leq t$ means that event i must occur within time t of event j. To say that event i must occur at least time t after event j, we use $x_i - x_j \geq t$, or, to keep all constraints as less-than-or-equal constraints, $x_j - x_i \leq -t$.

Constraint Graphs

Given a system of difference constraints, we can make a digraph for it by making a vertex for each unknown (x_i) and an edge for each constraint. For technical reasons, we have an additional vertex x_0 with an edge from x_0 to every other vertex.

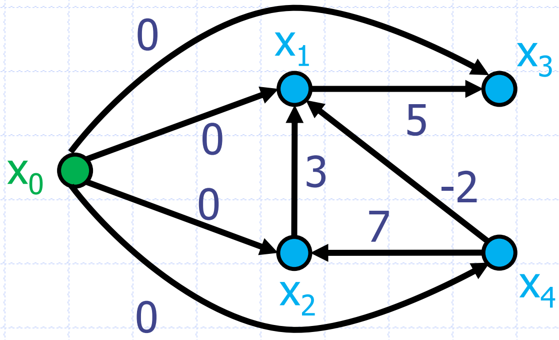
The Constraint Graph $G = (V, E)$ of a difference system $Ax \leq b$ with n unknowns and m constraints is:

$V = \{v_0, v_1, \ldots v_n\}$
$E = \{(v_i, v_j) \mid x_j - x_i \leq b_k \text{ is a constraint}\} \cup$
$\{(v_0, v_1), (v_0, v_2), \ldots (v_0, v_n)\}$

$w(v_i, v_j) = b_k$ when $x_j - x_i \leq b_k$ is a constraint
$w(v_0, v_i) = 0$ otherwise

Constraint Graphs

$x_1 - x_2 \le 3$
$x_2 - x_4 \le 7$
$x_3 - x_1 \le 5$
$x_1 - x_4 \le -2$



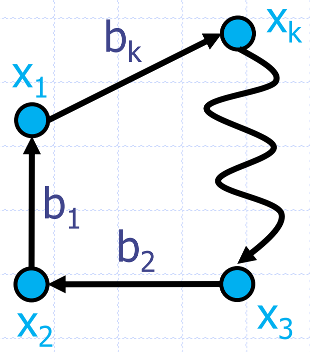
We can find a solution to the system of difference constraints by finding shortest-path weights in the constraint graph:

Theorem: Given a system $Ax \le b$ of difference constraints, let $G = (V, E)$ be the corresponding constraint graph. If G contains no negative-weight cycles, then $x = (\delta(v_0, v_1), \delta(v_0, v_2), \ldots, \delta(v_0, v_n))$ is a feasible solution for the system. If G contains a negative-weight cycle, there is no feasible solution for the system.

Constraint Graphs

Proof. Suppose G has a negative-weight cycle, and without loss of generality assume the constraints in the cycle are

$x_1 - x_2 \le b_1,$
$x_2 - x_3 \le b_2,$
…
$x_{k-1} - x_k \le b_{k-1},$ and
$x_k - x_1 \le b_k$



$$0 \le \sum_{i=1}^{k} b_i < 0$$

The constraints are contradictory. There is no solution.

Constraint Graphs

Proof. Now suppose G has no negative-weight cycles. Consider any edge (v_i, v_j) in E. By the triangle inequality,

$$\delta(v_0, v_j) \le \delta(v_0, v_i) + w(v_i, v_j) \quad \text{or}$$

$$\delta(v_0, v_j) - \delta(v_0, v_i) \le w(v_i, v_j)$$

$$x_j \quad - \quad x_i \qquad \le b_k$$

Or, the constraint k corresponding to that edge is satisfied.

■

Constraint Graphs

The previous theorem tells us that Bellman-Ford can be used to solve systems of difference constraints. Such a system with n unknowns and m constraints gives us a graph with n+1 vertices and n + m edges. Thus Bellman-Ford runs in time $O((n+1)(n+m)) = O(n^2 + nm)$.

This running time can be improved to $O(nm)$ by noting that the edges from v_0 need to be relaxed only in the first round of Bellman-Ford, as no edges go to v_0.

Triangle Inequality

Lemma. Let $G = (V, E)$ be a weighted digraph with weights w and source vertex s. Then, for all edges (u, v) in E, we have $\delta(s, v) \leq \delta(s, u) + w(u, v)$.

Proof. Suppose there is a path from s to u. Then, let p be a shortest path from s to u, and p' be p followed by (u, v). p' is a path from s to v of length $\delta(s, u) + w(u, v)$, so the shortest path from s to v has length no larger than that.

If there is not a path from s to u, then $\delta(s, u) = \infty$. Regardless of whether there is a path from s to v, $\delta(s, v) \leq \delta(s, u)$. ■

Other Useful Lemmas

Other lemmas that were used in the full proofs of the theorems of this chapter are contained in section 24.5. They are mostly simple and intuitive. Please read them!