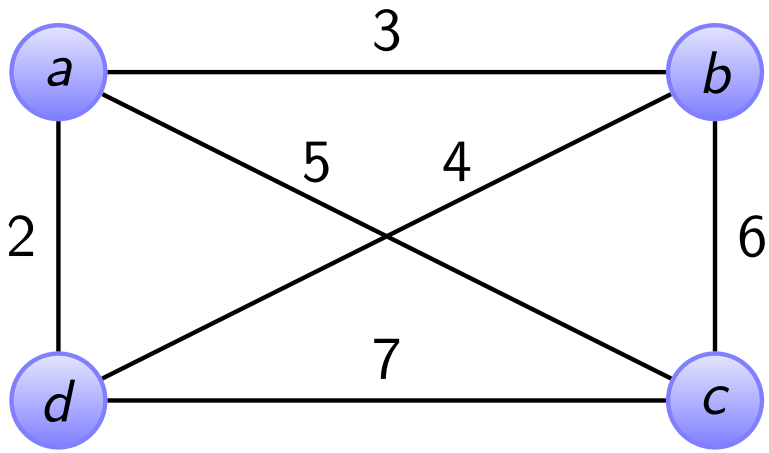
Weighted Graphs

Definition

A **weighted graph**, (V, E, w), is a graph (V, E) together with a weight function $w : E \to \mathbb{R}$. If $e \in E$, then $w(e)$ is the weight of edge e.

For example, the graph at the right is a weighted graph of K_4.



Often weights are associated with a *cost* or a *benefit* incurred when traversing the edge. Many important problems involve finding a trail, path, or circuit with minimum cost.

The Traveling Salesman Problem

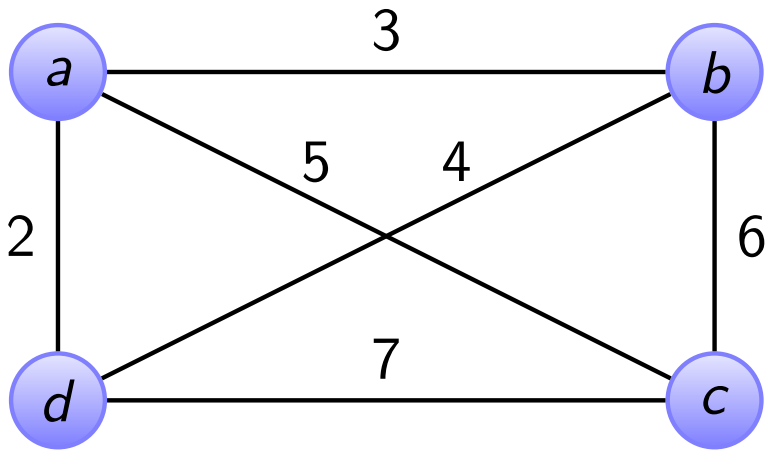
A very important problem is known as the **Traveling Salesman Problem** (TSP). Consider a salesman who must visit n different cities (or offices within a city). There is a cost associated with travel between each location he must visit and he's interested in completing his rounds as efficiently as possible based on some measure (least cost, or perhaps shortest time).

The TSP can be stated as: Given a connected, weighted graph G, find a Hamiltonian circuit in G of minimum total weight.

When the number of vertices in a graph is small, we can find every possible Hamiltonian circuit and just pick one with smallest weight.

The Traveling Salesman Problem

For example, consider all possible cycles (a circuit that only visits each vertex once, except the start/end vertex) in our weighted K_4.



cycle	weight
$a\,b\,c\,d\,a$	$3+6+7+2=18$
$a\,b\,d\,c\,a$	$3+4+7+5=19$
$a\,c\,b\,d\,a$	$5+6+4+2=17$
$a\,c\,d\,b\,a$	$5+7+4+3=19$
$a\,d\,b\,c\,a$	$2+4+6+5=17$
$a\,d\,c\,b\,a$	$2+7+6+3=18$

- There are six cycles – but half of these are reversals of the other half.
- The starting point is arbitrary – a given cycle will have the same cost regardless of starting point.

The Traveling Salesman Problem

Question: How many different Hamiltonian circuits does K_n have?

Answer: Every vertex in K_n is connected to every other vertex.

- There are $P(n, n) = n!$ ways to pick a starting vertex and choose a path that returns to the starting vertex.
- We must divide this by n since our count includes n identical cycles that differ only in starting point.
- We must divide by 2 since the forward and reverse of a cycle should be considered the same.

Thus, there are $\frac{n!}{2n} = \frac{(n-1)!}{2}$ different Hamiltonian circuits in K_n.

This is rather bad news...

The Traveling Salesman Problem

Suppose a computer can generate and check the cost of 10^6 Hamiltonian circuits in a graph every second. (For the sake of simplicity we assume that this time is independent of the number of vertices in the graph).

- When $n = 10$, this computer can solve the TSP in under a second.
- When $n = 12$, however, it takes this computer 20 seconds to solve the TSP.
- When $n = 15$ it will take over 12 hours...
- and when $n = 20$ it would take over 1900 years.

Granted, this is a brute-force approach to solving the TSP. The best known optimal algorithm has an operation count on the order of $n^2 2^n$ (i.e., $O(n^2 2^n)$), which means that when the number of vertices in the graph increases by one, it will take more than twice as long to find an optimal Hamiltonian circuit.

The Traveling Salesman Problem

This situation is bleak, but it is much better if we are willing to accept a *nearly optimal* solution to the TSP.

It is not unreasonable to assume that we are solving the TSP on a complete graph since edges do not necessarily indicate routes between locations but costs associated with traveling from one location to another.

Two simple algorithms we'll consider for the TSP on a complete graph are both **greedy** algorithms; they are multistep algorithms that make optimal choices at each step with the assumption that this will lead to a near optimal overall result.

In practice, these often work well but can produce far from optimal circuits in certain cases.

The Traveling Salesman Problem

Let G be a weighted graph based on K_n.

Vertex Greedy Algorithm (VGA)

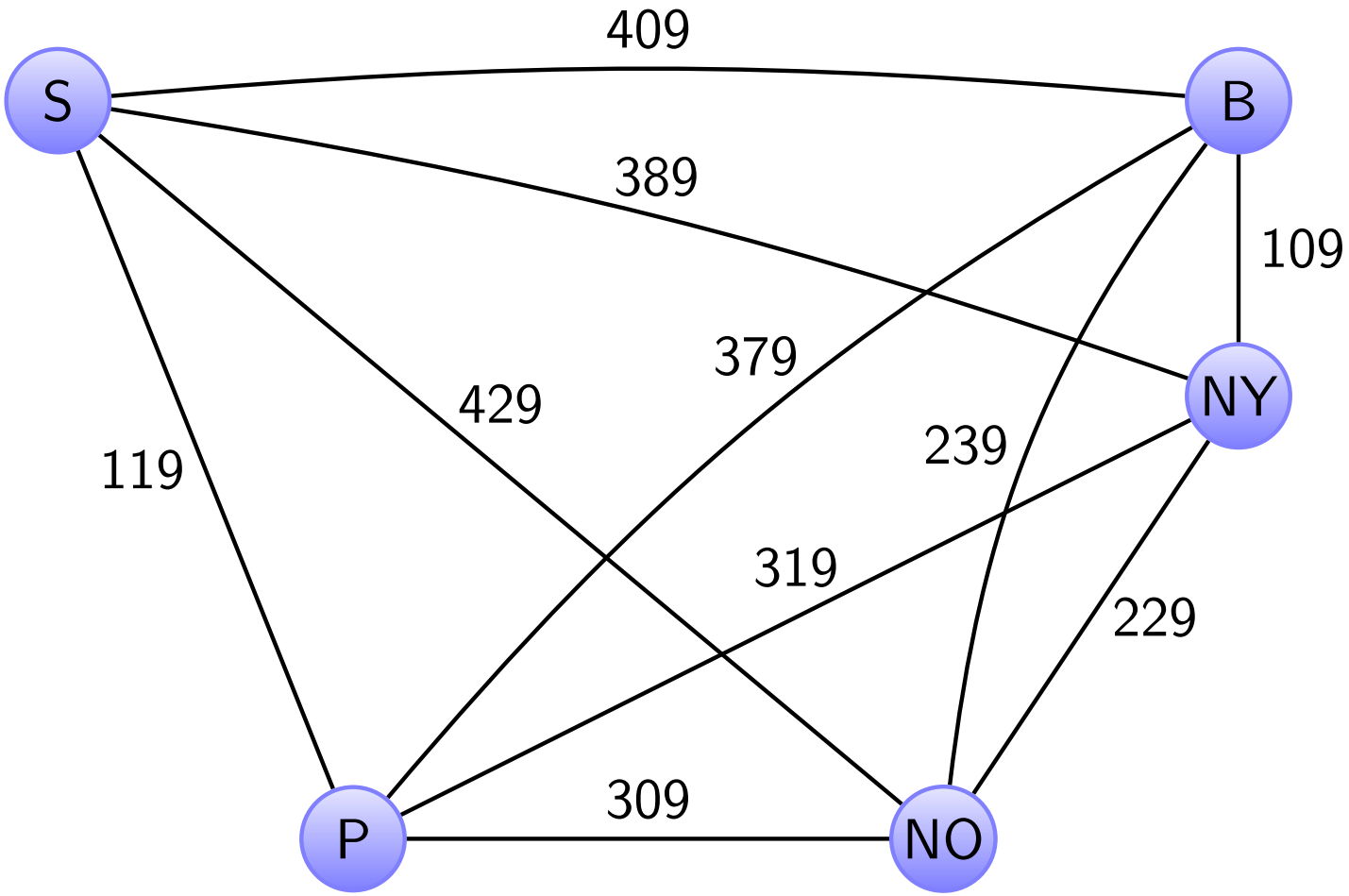
1. Identify starting vertex as v_1 and create set $V = \{v_1\}$.
2. For $i = 2$ to n:
   - Let v_i be an unvisited vertex v for which the edge from v_{i-1} to v has minimum weight.
   - $V = V \cup \{v_i\}$.
3. Set $v_{n+1} = v_1$.

Edge Greedy Algorithm (EGA)

1. Sort edges by weight.
2. Identify edge of minimum weight as e_1 and create set $V = \{v : e_1 \text{ is incident to } v\}$. Initialize $i = 2$.
3. While $|V| < n$:
   - Let e_i be an edge of minimum weight that does not create a cycle of length less than n or create a vertex of degree 3 in V.
   - $V = V \cup \{v : e_i \text{ incident to } v\}$.
   - $i = i + 1$.

The Traveling Salesman Problem

The following graph shows the cost of flying between Seattle (S), Phoenix (P), New Orleans (NO), New York (NY), and Boston (B).



The Traveling Salesman Problem

Example

Find near-optimal Hamiltonian cycles using the VGA and the EGA.

VGA

V	Cost
B	\$0
B, NY	\$109
B, NY, NO	\$338
B, NY, NO, P	\$647
B, NY, NO, P, S	\$766
B, NY, NO, P, S, B	\$1,175

EGA

e_i	Cost
(B,NY)	\$109
(S,P)	\$228
(NY,NO)	\$457
(NO,P)	\$766
(B,S)	\$1,175

In this case both algorithms find the same cycle; this isn't always true.

The Traveling Salesman Problem

- The graph in our last example is K_5, so there are $\frac{4!}{2} = 12$ Hamiltonian cycles. Checking all of them reveals that the optimal cycle is Boston, NY, Seattle, Phoenix, New Orleans, Boston with a cost of \$1,165.
- The ratio of the cost we found to the true optimal cost is

$$\frac{1175}{1165} \approx 1.0086$$

This means that the near-optimal cycle found incurs an extra cost of only 0.86%. In this case we'd probably consider the cycle we found to be acceptable.