Introduction to Algorithms

Chap 24-2

Shortest Paths II
• Bellman-Ford algorithm
• Linear programming and difference constraints
• VLSI layout compaction

Hsin-Lung Wu, CSIE, NTPU
Negative-weight cycles

**Recall:** If a graph $G = (V, E)$ contains a negative-weight cycle, then some shortest paths may not exist.

**Example:**

![Diagram of a graph with a negative-weight cycle](image)
**Negative-weight cycles**

**Recall:** If a graph $G = (V, E)$ contains a negative-weight cycle, then some shortest paths may not exist.

**Example:**

Bellman-Ford algorithm: Finds all shortest-path lengths from a source $s \in V$ to all $v \in V$ or determines that a negative-weight cycle exists.
Bellman-Ford algorithm

\[
d[s] \leftarrow 0
\]
\[
\text{for each } v \in V - \{s\} \text{ do } d[v] \leftarrow \infty
\]
\[
\text{for } i \leftarrow 1 \text{ to } |V| - 1 \text{ do for each edge } (u, v) \in E
\]
\[
\text{do if } d[v] > d[u] + w(u, v) \text{ then } d[v] \leftarrow d[u] + w(u, v)
\]
\[
\text{for each edge } (u, v) \in E
\]
\[
\text{do if } d[v] > d[u] + w(u, v) \text{ then report that a negative-weight cycle exists}
\]

At the end, \( d[v] = \delta(s, v) \), if no negative-weight cycles.

Time = \( O(VE) \).
Example of Bellman-Ford

![Graph](image)
Example of Bellman-Ford

Initialization.
Example of Bellman-Ford

Order of edge relaxation.
Example of Bellman-Ford

Graph with weighted edges:

- Source: A
- Sink: E
- Edges: A to B (4), B to C (3), C to D (5), D to E (8), B to E (2)
- Negative weight: on edge A to B (4)

Shortest path from A to E:
- A → B → E
- Total weight: 4 + 2 = 6

No negative cycles detected.
Example of Bellman-Ford
Example of Bellman-Ford
Example of Bellman-Ford
Example of Bellman-Ford
Example of Bellman-Ford
Example of Bellman-Ford
Example of Bellman-Ford
Example of Bellman-Ford

End of pass 1.
Example of Bellman-Ford
Example of Bellman-Ford
Example of Bellman-Ford
Example of Bellman-Ford

The diagram illustrates a network with labeled edges, demonstrating the Bellman-Ford algorithm. The network consists of vertices labeled A, B, C, D, and E, with directed edges connecting them. The weights on the edges are shown as numerical values, and the algorithm is used to find the shortest path from a source vertex to all other vertices in the network.
Example of Bellman-Ford
Example of Bellman-Ford

![Graph Diagram]
Example of Bellman-Ford
Example of Bellman-Ford
Example of Bellman-Ford

End of pass 2 (and 3 and 4).
Correctness

**Theorem.** If $G = (V, E)$ contains no negative-weight cycles, then after the Bellman-Ford algorithm executes, $d[v] = \delta(s, v)$ for all $v \in V$. 
Correctness

Theorem. If $G = (V, E)$ contains no negative-weight cycles, then after the Bellman-Ford algorithm executes, $d[v] = \delta(s, v)$ for all $v \in V$.

Proof. Let $v \in V$ be any vertex, and consider a shortest path $p$ from $s$ to $v$ with the minimum number of edges.

Since $p$ is a shortest path, we have

$$\delta(s, v_i) = \delta(s, v_{i-1}) + w(v_{i-1}, v_i).$$
Initially, $d[v_0] = 0 = \delta(s, v_0)$, and $d[v_0]$ is unchanged by subsequent relaxations (because of the lemma from Lecture 14 that $d[v] \geq \delta(s, v)$).

- After 1 pass through $E$, we have $d[v_1] = \delta(s, v_1)$.
- After 2 passes through $E$, we have $d[v_2] = \delta(s, v_2)$.
  
  …

- After $k$ passes through $E$, we have $d[v_k] = \delta(s, v_k)$.

Since $G$ contains no negative-weight cycles, $p$ is simple. Longest simple path has $\leq |V| - 1$ edges.
Detection of negative-weight cycles

**Corollary.** If a value $d[v]$ fails to converge after $|V| - 1$ passes, there exists a negative-weight cycle in $G$ reachable from $s$.  □
Linear programming

Let $A$ be an $m \times n$ matrix, $b$ be an $m$-vector, and $c$ be an $n$-vector. Find an $n$-vector $x$ that maximizes $c^T x$ subject to $Ax \leq b$, or determine that no such solution exists.
Linear-programming algorithms

Algorithms for the general problem

• Simplex methods — practical, but worst-case exponential time.
• Interior-point methods — polynomial time and competes with simplex.
Linear-programming algorithms

Algorithms for the general problem

- Simplex methods — practical, but worst-case exponential time.
- Interior-point methods — polynomial time and competes with simplex.

**Feasibility problem:** No optimization criterion. Just find $x$ such that $Ax \leq b$.
- In general, just as hard as ordinary LP.
Solving a system of difference constraints

Linear programming where each row of $A$ contains exactly one 1, one $-1$, and the rest 0’s.

**Example:**

$$
\begin{align*}
x_1 - x_2 & \leq 3 \\
x_2 - x_3 & \leq -2 \\
x_1 - x_3 & \leq 2
\end{align*}
\right\}
\begin{align*}
x_j - x_i & \leq w_{ij}
\end{align*}
$$
Solving a system of difference constraints

Linear programming where each row of \( A \) contains exactly one 1, one –1, and the rest 0’s.

Example:

\[
\begin{align*}
  x_1 - x_2 &\leq 3 \\
  x_2 - x_3 &\leq -2 \\
  x_1 - x_3 &\leq 2 \\
\end{align*}
\]

Solution:

\[
\begin{align*}
  x_1 &= 3 \\
  x_2 &= 0 \\
  x_3 &= 2 \\
\end{align*}
\]
Solving a system of difference constraints

Linear programming where each row of $A$ contains exactly one 1, one $-1$, and the rest 0’s.

Example:

\[
\begin{align*}
    x_1 - x_2 &\leq 3 \\
    x_2 - x_3 &\leq -2 \\
    x_1 - x_3 &\leq 2
\end{align*}
\]

Solution:

\[
\begin{align*}
    x_1 &= 3 \\
    x_2 &= 0 \\
    x_3 &= 2
\end{align*}
\]

Constraint graph:

\[
x_j - x_i \leq w_{ij}
\]

(The “$A$” matrix has dimensions $|E| \times |V|$.)
Unsatisfiable constraints

**Theorem.** If the constraint graph contains a negative-weight cycle, then the system of differences is unsatisfiable.
Unsatisfiable constraints

**Theorem.** If the constraint graph contains a negative-weight cycle, then the system of differences is unsatisfiable.

**Proof.** Suppose that the negative-weight cycle is $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k \rightarrow v_1$. Then, we have

\[
\begin{align*}
x_2 - x_1 & \leq w_{12} \\
x_3 - x_2 & \leq w_{23} \\
\vdots & \\
x_k - x_{k-1} & \leq w_{k-1,k} \\
x_1 - x_k & \leq w_{k1}
\end{align*}
\]
Unsatisfiable constraints

**Theorem.** If the constraint graph contains a negative-weight cycle, then the system of differences is unsatisfiable.

**Proof.** Suppose that the negative-weight cycle is $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k \rightarrow v_1$. Then, we have

\[
\begin{align*}
x_2 - x_1 & \leq w_{12} \\
x_3 - x_2 & \leq w_{23} \\
& \vdots \\
x_k - x_{k-1} & \leq w_{k-1,k} \\
x_1 - x_k & \leq w_{k1}
\end{align*}
\]

Therefore, no values for the $x_i$ can satisfy the constraints.

\[
0 \leq \text{weight of cycle} < 0
\]
Satisfying the constraints

Theorem. Suppose no negative-weight cycle exists in the constraint graph. Then, the constraints are satisfiable.
Satisfying the constraints

**Theorem.** Suppose no negative-weight cycle exists in the constraint graph. Then, the constraints are satisfiable.

**Proof.** Add a new vertex $s$ to $V$ with a 0-weight edge to each vertex $v_i \in V$. 
Satisfying the constraints

**Theorem.** Suppose no negative-weight cycle exists in the constraint graph. Then, the constraints are satisfiable.

**Proof.** Add a new vertex $s$ to $V$ with a 0-weight edge to each vertex $v_i \in V$.

**Note:** No negative-weight cycles introduced $\Rightarrow$ shortest paths exist.
Proof (continued)

Claim: The assignment \( x_i = \delta(s, v_i) \) solves the constraints. Consider any constraint \( x_j - x_i \leq w_{ij} \), and consider the shortest paths from \( s \) to \( v_j \) and \( v_i \):

The triangle inequality gives us \( \delta(s, v_j) \leq \delta(s, v_i) + w_{ij} \). Since \( x_i = \delta(s, v_i) \) and \( x_j = \delta(s, v_j) \), the constraint \( x_j - x_i \leq w_{ij} \) is satisfied.
Bellman-Ford and linear programming

**Corollary.** The Bellman-Ford algorithm can solve a system of $m$ difference constraints on $n$ variables in $O(mn)$ time.

Single-source shortest paths is a simple LP problem.

In fact, Bellman-Ford maximizes $x_1 + x_2 + \cdots + x_n$ subject to the constraints $x_j - x_i \leq w_{ij}$ and $x_i \leq 0$ (exercise).

Bellman-Ford also minimizes $\max_i \{x_i\} - \min_i \{x_i\}$ (exercise).
Application to VLSI layout compaction

**Integrated-circuit features:**

**Problem:** Compact (in one dimension) the space between the features of a VLSI layout without bringing any features too close together.
Constraint: \[ x_2 - x_1 \geq d_1 + \lambda \]

Bellman-Ford minimizes \( \max_i \{ x_i \} - \min_i \{ x_i \} \), which compacts the layout in the \( x \)-dimension.