The Bellman-Ford Algorithm

Andreas Klappenecker
Single Source Shortest Path Problem

Given a graph $G=(V,E)$, a weight function $w: E \rightarrow \mathbb{R}$, and a source node $s$, find the shortest path from $s$ to $v$ for every $v$ in $V$.

- We allow negative edge weights.
- $G$ is not allowed to contain cycles of negative total weight.
- Dijkstra’s algorithm cannot be used, as weights must be nonnegative.
Bellman-Ford SSPP Algorithm

Input: directed or undirected graph $G = (V,E,w)$
for all $v$ in $V$ {
    $d[v] = \text{infinity}; \text{parent}[v] = \text{nil}$;
}
$d[s] = 0; \text{parent}[s] = s$;
for $i := 1$ to $|V| - 1$ { // ensure that information on distance from $s$ propagates
    for each $(u,v)$ in $E$ { // relax all edges
        if ($d[u] + w(u,v) < d[v]$) then {
            $d[v] := d[u] + w(u,v); \text{parent}[v] := u;$
        }
    }
}
Running Time: $O(VE)$

Input: directed or undirected graph $G = (V,E,w)$

for all $v$ in $V$

\[ d[v] = \text{infinity}; \text{parent}[v] = \text{nil}; \]

\}

d[s] = 0; parent[s] = s;

for $i := 1$ to $|V| - 1$

\{

for each $(u,v)$ in $E$ // relax all edges

\[ \text{if} (d[u] + w(u,v) < d[v]) \text{ then} \]

\[ d[v] := d[u] + w(u,v); \text{parent}[v] := u; \]

\}

Init: $O(V)$

Nested loops:

$O(V)O(E) = O(VE)$
Bellman-Ford Example

Let's process edges in the order (c,b),(a,b),(c,a),(s,a),(s,c)

<table>
<thead>
<tr>
<th>Node</th>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>s</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>∞</td>
</tr>
<tr>
<td>b</td>
<td>∞</td>
</tr>
<tr>
<td>c</td>
<td>∞</td>
</tr>
</tbody>
</table>
Information Propagation

Consider a graph on n+1 vertices:

\[ s \rightarrow a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_{n-1} \rightarrow a_n \]

where each edge has weight 1.

Choose edges from right to left. Then node \( a_i \) has correct distance estimate after \( i^{th} \) iteration.

<table>
<thead>
<tr>
<th>Node</th>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>( a_1 )</td>
<td>( \infty ) 1 1 1 ...</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>( \infty ) 2 2 2 ...</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>( \infty ) ( \infty ) 3 3 ...</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>( \infty ) ( \infty ) ( \infty ) 4 ...</td>
</tr>
</tbody>
</table>
Correctness

Fact 1: The distance estimate $d[v]$ never underestimates the actual shortest path distance from $s$ to $v$.

Fact 2: If there is a shortest path from $s$ to $v$ containing at most $i$ edges, then after iteration $i$ of the outer for loop:

$$d[v] \leq \text{the actual shortest path distance from } s \text{ to } v.$$
Theorem: Suppose that G is a weighted graph without negative weight cycles and let s denote the source node. Then Bellman-Ford correctly calculates the shortest path distances from s.

Proof: Every shortest path has at most |V| - 1 edges. By Fact 1 and 2, the distance estimate d[v] is equal to the shortest path length after |V|-1 iterations.
Variations

One can stop the algorithm if an iteration does not modify distance estimates. This is beneficial if shortest paths are likely to be less than $|V|-1$.

One can detect negative weight cycles by checking whether distance estimates can be reduced after $|V|-1$ iterations.
The Boost Graph Library

The BGL contains generic implementations of all the graph algorithms that we have discussed:

- Breadth-First-Search
- Depth-First-Search
- Kruskal’s MST algorithm
- Strongly Connected Components
- Dijkstra’s SSSP algorithm
- Bellman-Ford SSSP algorithm

I recommend that you gain experience with this useful library. Recommended reading: The Boost Graph Library by J.G. Siek, L.-Q. Lee, and A. Lumsdaine, Addison-Wesley, 2002.