Lecture 17: Shortest Paths IV - Speeding up Dijkstra

Lecture Overview

- Single-source single-target Dijkstra
- Bidirectional search
- Goal directed search - potentials and landmarks

Readings

Wagner paper on [website] (upto Section 3.2)

DIJKSTRA single-source, single-target

\[
\begin{align*}
\text{Initialize()} \\
Q &\leftarrow V[G] \\
\text{while } Q \neq \phi &\text{ do } u \leftarrow \text{EXTRACT_MIN}(Q) \text{ (stop if } u = t!) \\
&\text{ for each vertex } v \in \text{Adj}[u] \\
&\quad \text{ do RELAX}(u, v, w)
\end{align*}
\]

Observation: If only shortest path from \( s \) to \( t \) is required, stop when \( t \) is removed from \( Q \), i.e., when \( u = t \)
Bi-Directional Search

Note: Speedup techniques covered here do not change worst-case behavior, but reduce the number of visited vertices in practice.

![Bi-directional Search](image1)

Figure 1: Bi-directional Search.

Bi-D Search

Alternate forward search from $s$
backward search from $t$
(follow edges backward)

$d_f(u)$ distances for forward search
$d_b(u)$ distances for backward search

Algorithm terminates when some vertex $w$ has been processed, i.e., deleted from the queue of both searches, $Q_f$ and $Q_b$

![Bi-D Search](image2)

Figure 2: Bi-D Search.
Subtlety: After search terminates, find node \( x \) with minimum value of \( d_f(x) + d_b(x) \). \( x \) may not be the vertex \( w \) that caused termination as in example to the left!
Find shortest path from \( s \) to \( x \) using \( \Pi_f \) and shortest path backwards from \( t \) to \( x \) using \( \Pi_b \).

Note: \( x \) will have been deleted from either \( Q_f \) or \( Q_b \) or both.

Minimum value for \( d_f(x) + d_b(x) \) over all vertices that have been processed in at least one search

\[
d_f(u) + d_b(u) = 3 + 6 = 9
\]
\[ d_f(u') + d_b(u') = 6 + 3 = 9 \]
\[ d_f(w) + d_b(w) = 5 + 5 = 10 \]

**Goal-Directed Search or \( A^\ast \)**
Modify edge weights with potential function over vertices.
\[ \pi(u, v) = w(u, v) - \lambda_t(s) + \lambda_t(t) \]

Search toward target:

![Figure 4: Targeted Search](image)

**Correctness**
\[ \pi(p) = w(p) - \lambda_t(s) + \lambda_t(t) \]
So shortest paths are maintained in modified graph with \( \pi \) weights.

![Figure 5: Modifying Edge Weights.](image)

To apply Dijkstra, we need \( \pi(u, v) \geq 0 \) for all \((u, v)\).
Choose potential function appropriately, to be feasible.

**Landmarks**
Small set of landmarks \( LCV \). For all \( u \in V, l \in L \), pre-compute \( \delta(u, l) \).

Potential \( \lambda^{(l)}_t(u) = \delta(u, l) - \delta(t, l) \) for each \( l \).

CLAIM: \( \lambda^{(l)}_t \) is feasible.
Feasibility

\[ \overline{w}(u, v) = w(u, v) - \lambda_t^{(l)}(u) + \lambda_t^{(l)}(v) \]
\[ = w(u, v) - \delta(u, l) + \delta(t, l) + \delta(v, l) - \delta(t, l) \]
\[ = w(u, v) - \delta(u, l) + \delta(v, l) \geq 0 \text{ by the } \Delta \text{-inequality} \]

\[ \lambda_t(u) = \max_{l \in L} \lambda_t^{(l)}(u) \text{ is also feasible} \]