Widest shortest-paths

- Metric of the form \((d, b)\), where \(d\) is distance \((\min, +)\) and \(b\) is capacity \((\max, \min)\).
- Metrics are compared lexicographically, with distance considered first.
- Such things are found in the vast literature on Quality-of-Service (QoS) metrics for Internet routing.

\[
wsp = \mathbf{sp} \times \mathbf{bw}
\]
Widest shortest-paths

Weights are globally optimal (we have a semiring)

Widest shortest-path weights computed by Dijkstra and Bellman-Ford

$$R = \begin{bmatrix}
0 & 1 & 2 & 3 & 4 \\
0 & (0, T) & (1, 10) & (3, 10) & (2, 5) & (2, 10) \\
1 & (1, 10) & (0, T) & (2, 100) & (1, 5) & (1, 100) \\
2 & (3, 10) & (2, 100) & (0, T) & (1, 100) & (1, 100) \\
3 & (2, 5) & (1, 5) & (1, 100) & (0, T) & (2, 100) \\
4 & (2, 10) & (1, 100) & (1, 100) & (2, 100) & (0, T)
\end{bmatrix}$$
But what about the paths themselves?

Four optimal paths of weight (3, 10).

\[
P_{\text{optimal}}(0, 2) = \{(0, 1, 2), (0, 1, 4, 2)\}
\]
\[
P_{\text{optimal}}(2, 0) = \{(2, 1, 0), (2, 4, 1, 0)\}
\]

There are standard ways to extend Bellman-Ford and Dijkstra to compute paths (or the associated next hops).

Do these extended algorithms find all optimal paths?

Surprise!

Four optimal paths of weight (3, 10)

\[
P_{\text{optimal}}(0, 2) = \{(0, 1, 2), (0, 1, 4, 2)\}
\]
\[
P_{\text{optimal}}(2, 0) = \{(2, 1, 0), (2, 4, 1, 0)\}
\]

Paths computed by (extended) Dijkstra

\[
P_{\text{Dijkstra}}(0, 2) = \{(0, 1, 2), (0, 1, 4, 2)\}
\]
\[
P_{\text{Dijkstra}}(2, 0) = \{(2, 4, 1, 0)\}
\]

Notice that 0’s paths cannot both be implemented with next-hop forwarding since \(P_{\text{Dijkstra}}(1, 2) = \{(1, 4, 2)\}\).

Paths computed by (extended) distributed Bellman-Ford

\[
P_{\text{Bellman}}(0, 2) = \{(0, 1, 4, 2)\}
\]
\[
P_{\text{Bellman}}(2, 0) = \{(2, 1, 0), (2, 4, 1, 0)\}
\]
Optimal paths from 0 to 2. Computed by Dijkstra but not by Bellman-Ford

Optimal paths from 2 to 1. Computed by Bellman-Ford but not by Dijkstra
Observations

For distributed Bellman-Ford

\[
\text{next-hop-paths}(A) = \text{computed-paths}(A) \subseteq \text{optimal-paths}(A)
\]

For Dijkstra’s algorithm

\[
\text{next-hop-paths}(A) \subseteq \text{computed-paths}(A) \subseteq \text{optimal-paths}(A)
\]

How can we understand this (algebraically)?

The Algorithm to Algebra (A2A) method

\[
\begin{pmatrix}
\text{original metric} \\
\text{complex algorithm}
\end{pmatrix} + \begin{pmatrix}
\text{modified metric} \\
\text{matrix equations (generic algorithm)}
\end{pmatrix}
\]

We can capture path computation with this algebra

\[\text{sp} \times \text{bw} \times \text{seq}(E)\]

But this algebra is not distributive!

\[\neg \text{LC}(\text{sp} \times \text{bw})\]

\[\neg \text{LK}(\text{seq}(E))\]
Towards a non-classical theory of algebraic path finding

We need theory that can accept algebras that violate distributivity.

**Global optimality**

\[ A^*(i, j) = \bigoplus_{p \in P(i, j)} w(p), \]

**Left local optimality (distributed Bellman-Ford)**

\[ L = (A \otimes L) \oplus I. \]

**Right local optimality (Dijkstra’s Algorithm)**

\[ R = (R \otimes A) \oplus I. \]

Embrace the fact that all three notions can be distinct.

---

**Left-Local Optimality**

Say that \( L \) is a left locally-optimal solution when

\[ L = (A \otimes L) \oplus I. \]

That is, for \( i \neq j \) we have

\[ L(i, j) = \bigoplus_{q \in V} A(i, q) \otimes L(q, j) \]

- \( L(i, j) \) is the best possible value given the values \( L(q, j) \), for all out-neighbors \( q \) of source \( i \).
- Rows \( L(i, \_ \) represents out-trees from \( i \) (think Bellman-Ford).
- Columns \( L(\_, i) \) represents in-trees to \( i \).
- Works well with hop-by-hop forwarding from \( i \).
Right-Local Optimality

Say that $R$ is a right locally-optimal solution when

$$R = (R \otimes A) \oplus I.$$ 

That is, for $i \neq j$ we have

$$R(i, j) = \bigoplus_{q \in V} R(i, q) \otimes A(q, j).$$

- $R(i, j)$ is the best possible value given the values $R(q, j)$, for all in-neighbors $q$ of destination $j$.
- Rows $L(i, \_)$ represents out-trees from $i$ (think Dijkstra).
- Columns $L(\_, i)$ represents in-trees to $i$.

With and Without Distributivity

With distributivity

For (bounded) semirings, the three optimality problems are essentially the same — locally optimal solutions are globally optimal solutions.

$$A^* = L = R$$

Without distributivity

It may be that $A^*$, $L$, and $R$ exists but are all distinct.

Back and Forth

$$L = (A \otimes L) \oplus I \iff L^T = (L^T \otimes A^T) \oplus I$$

where $\otimes^T$ is matrix multiplication defined with $a \otimes^T b = b \otimes a$
Dijkstra’s Algorithm

Classical Dijkstra

Given adjacency matrix \( A \) over a selective semiring and source vertex \( i \in V \), Dijkstra’s algorithm will compute \( A^*(i, \_ \_ ) \) such that

\[
A^*(i, j) = \bigoplus_{p \in P(i,j)} w_A(p).
\]

Non-Classical Dijkstra

If we drop assumptions of distributivity, then given adjacency matrix \( A \) and source vertex \( i \in V \), Dijkstra’s algorithm will compute \( R(i, \_ \_ ) \) such that

\[
\forall j \in V : R(i, j) = I(i,j) \bigoplus \bigoplus_{q \in V} R(i, q) \otimes A(q, j).
\]


Dijkstra’s algorithm

**Input**: adjacency matrix \( A \) and source vertex \( i \in V \),

**Output**: the \( i \)-th row of \( R \), \( R(i, \_ \_ ) \).

\[
\begin{align*}
\text{begin} \\
S & \leftarrow \{i\} \\
R(i, \_i) & \leftarrow 1 \\
\text{for each } q \in V - \{i\} : R(i, q) & \leftarrow A(i, q) \\
\text{while } S \neq V \\
\text{begin} \\
\text{find } q \in V - S \text{ such that } R(i, q) \text{ is } \leq_{(s)} \text{-minimal} \\
S & \leftarrow S \cup \{q\} \\
\text{for each } j \in V - S \\
R(i, j) & \leftarrow R(i, j) \bigoplus (R(i, q) \otimes A(q, j)) \\
\text{end} \\
\text{end} \\
\end{align*}
\]
Classical proofs of Dijkstra’s algorithm (for global optimality) assume

## Semiring Axioms

- **AS** \((\oplus)\) :  \(a \oplus (b \oplus c) = (a \oplus b) \oplus c\)
- **CM** \((\oplus)\) :  \(a \oplus b = b \oplus a\)
- **ID** \((\oplus)\) :  \(\emptyset \oplus a = a\)
- **AS** \((\otimes)\) :  \(a \otimes (b \otimes c) = (a \otimes b) \otimes c\)
- **IDL** \((\otimes)\) :  \(\top \otimes a = a\)
- **IDR** \((\otimes)\) :  \(a \otimes \top = a\)
- **ANL** \((\otimes)\) :  \(\emptyset \otimes a = \emptyset\)
- **ANR** \((\otimes)\) :  \(a \otimes \emptyset = \emptyset\)
- **LD** :  \(a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)\)
- **RD** :  \((a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)\)

Note that we can derive right absorption,

- **RA** :  \(a \oplus (a \otimes b) = a\)

and this gives (right) inflationarity,  \(\forall a, b : a \leq a \otimes b\).
What will we assume? Very little!

### Semiring Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{AS}(\oplus) )</td>
<td>( a \oplus (b \oplus c) = (a \oplus b) \oplus c )</td>
</tr>
<tr>
<td>( \text{CM}(\oplus) )</td>
<td>( a \oplus b = b \oplus a )</td>
</tr>
<tr>
<td>( \text{ID}(\oplus) )</td>
<td>( 0 \oplus a = a )</td>
</tr>
<tr>
<td>( \text{ANL}(\oplus) )</td>
<td>( (a \oplus (b \oplus c)) = (a \oplus b) \oplus c )</td>
</tr>
<tr>
<td>( \text{IDL}(\otimes) )</td>
<td>( 1 \otimes a = a )</td>
</tr>
<tr>
<td>( \text{ANL}(\otimes) )</td>
<td>( (a \otimes 1) = 1 )</td>
</tr>
<tr>
<td>( \text{ANR}(\otimes) )</td>
<td>( (a \otimes 0) = 0 )</td>
</tr>
<tr>
<td>( \text{LD} )</td>
<td>( a \oplus (b \oplus c) = (a \oplus b) \oplus c )</td>
</tr>
</tbody>
</table>

### Additional Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{SL}(\oplus) )</td>
<td>( a \oplus b \in {a, b} )</td>
</tr>
<tr>
<td>( \text{ANL}(\otimes) )</td>
<td>( 1 \oplus a = 1 )</td>
</tr>
<tr>
<td>( \text{RA} )</td>
<td>( a \oplus (a \otimes b) = a )</td>
</tr>
</tbody>
</table>

- Note that we can no longer derive \( \text{RA} \), so we must assume it.
- Again, \( \text{RA} \) says that \( a \leq a \otimes b \).
- We don’t use \( \text{SL} \) explicitly in the proofs, but it is implicit in the algorithm’s definition of \( q_k \).
- We do not use \( \text{AS}(\oplus) \) and \( \text{CM}(\oplus) \) explicitly, but these assumptions are implicit in the use of the “big-\( \oplus \)” notation.
Under these weaker assumptions ...

Theorem (Sobrinho/Griffin)

Given adjacency matrix $A$ and source vertex $i \in V$, Dijkstra’s algorithm will compute $R(i, \_)$ such that

$$\forall j \in V : R(i, j) = I(i, j) \oplus \bigoplus_{q \in V} R(i, q) \otimes A(q, j).$$

That is, it computes one row of the solution for the right equation

$$R = RA \oplus I.$$

Dijkstra’s algorithm, annotated version

Subscripts make proofs by induction easier ....

begin

$S_1 \leftarrow \{i\}$

$R_1(i, i) \leftarrow 1$

for each $q \in V - S_1 : R_1(i, q) \leftarrow A(i, q)$

for each $k = 2, 3, \ldots, |V|$

begin

find $q_k \in V - S_{k-1}$ such that $R_{k-1}(i, q_k)$ is $\leq L$ -minimal

$S_k \leftarrow S_{k-1} \cup \{q_k\}$

for each $j \in V - S_k$

$R_k(i, j) \leftarrow R_{k-1}(i, j) \oplus (R_{k-1}(i, q_k) \otimes A(q_k, j))$

end

end
Main Claim, annotated

∀k : 1 ≤ k < |V| → ∀j ∈ S_k : R_k(i, j) = I(i, j) ⊕ \bigoplus_{q \in S_k} R_k(i, q) ⊗ A(q, j)

We will use

Observation 1 (no backtracking):

∀k : 1 ≤ k < |V| → ∀j ∈ S_{k+1} : R_{k+1}(i, j) = R_k(i, j)

Observation 2 (Dijkstra is “greedy”):

∀k : 1 ≤ k < |V| → ∀q ∈ S_k : ∀w ∈ V − S_k : R_k(i, q) ≤ R_k(i, w)

Observation 3 (Accurate estimates):

∀k : 1 ≤ k < |V| → ∀w ∈ V − S_k : R_k(i, w) = \bigoplus_{q \in S_k} R_k(i, q) \otimes A(q, w)

Observation 1

∀k : 1 ≤ k < |V| → ∀j ∈ S_{k+1} : R_{k+1}(i, j) = R_k(i, j)

Proof: This is easy to see by inspection of the algorithm. Once a node is put into S its weight never changes again.
The algorithm is “greedy”

Observation 2

\[ \forall k : 1 \leq k \leq |V| \implies \forall q \in S_k : \forall w \in V - S_k : R_k(i, q) \leq R_k(i, w) \]

By induction.
Base: Since \( S_1 = \{i\} \) and \( R_1(i, i) = \top \), we need to show that

\[ \top \leq A(i, w) \equiv \top = \top \oplus A(i, w). \]

This follows from \( \mathbb{ANL}(\oplus) \).

Induction: Assume \( \forall q \in S_k : \forall w \in V - S_k : R_k(i, q) \leq R_k(i, w) \) and

show \( \forall q \in S_{k+1} : \forall w \in V - S_{k+1} : R_{k+1}(i, q) \leq R_{k+1}(i, w) \).

Since \( S_{k+1} = S_k \cup \{q_{k+1}\} \), this means showing

1. \( \forall q \in S_k : \forall w \in V - S_{k+1} : R_{k+1}(i, q) \leq R_{k+1}(i, w) \)
2. \( \forall w \in V - S_{k+1} : R_{k+1}(i, q_{k+1}) \leq R_{k+1}(i, w) \)

By Observation 1, showing (1) is the same as

\[ \forall q \in S_k : \forall w \in V - S_{k+1} : R_k(i, q) \leq R_{k+1}(i, w) \]

which expands to (by definition of \( R_{k+1}(i, w) \))

\[ \forall q \in S_k : \forall w \in V - S_{k+1} : R_k(i, q) \leq R_k(i, w) \oplus (R_k(i, q_{k+1}) \otimes A(q_{k+1}, w)) \]

But \( R_k(i, q) \leq R_k(i, w) \) by the induction hypothesis, and

\( R_k(i, q) \leq (R_k(i, q_{k+1}) \otimes A(q_{k+1}, w)) \) by the induction hypothesis and \( \mathbb{RA} \).

Since \( a \leq_L b \wedge a \leq_L c \implies a \leq_L (b \oplus c) \), we are done.
By Observation 1, showing (2) is the same as showing

$$\forall w \in V - S_{k+1} : R_k(i, q_{k+1}) \leq R_{k+1}(i, w)$$

which expands to

$$\forall w \in V - S_{k+1} : R_k(i, q_{k+1}) \leq R_k(i, w) \oplus (R_k(i, q_{k+1}) \otimes A(q_{k+1}, w))$$

But $R_k(i, q_{k+1}) \leq R_k(i, w)$ since $q_{k+1}$ was chosen to be minimal, and

$R_k(i, q_{k+1}) \leq (R_k(i, q_{k+1}) \otimes A(q_{k+1}, w))$ by RA.

Since $a \leq_L b \wedge a \leq_L c \implies a \leq_L (b \circ c)$, we are done.

---

**Observation 3**

**Observation 3**

$$\forall k : 1 \leq k \leq |V| \implies \forall w \in V - S_k : R_k(i, w) = \bigoplus_{q \in S_k} R_k(i, q) \otimes A(q, w)$$

Proof: By induction:

Base : easy, since

$$\bigoplus_{q \in S_1} R_1(i, q) \otimes A(q, w) = \bar{1} \otimes A(i, w) = A(i, w) = R_1(i, w)$$

Induction step. Assume

$$\forall w \in V - S_k : R_k(i, w) = \bigoplus_{q \in S_k} R_k(i, q) \otimes A(q, w)$$

and show

$$\forall w \in V - S_{k+1} : R_{k+1}(i, w) = \bigoplus_{q \in S_{k+1}} R_{k+1}(i, q) \otimes A(q, w)$$
By Observation 1, and a bit of rewriting, this means we must show
\[ \forall w \in V - S_{k+1} : R_{k+1}(i, w) = R_{k}(i, q_{k+1}) \otimes A(q_{k+1}, w) \bigoplus_{q \in S_k} R_{k}(i, q) \otimes A(q) \]

Using the induction hypothesis, this becomes
\[ \forall w \in V - S_{k+1} : R_{k+1}(i, w) = R_{k}(i, q_{k+1}) \otimes A(q_{k+1}, w) \bigoplus R_{k}(i, w) \]

But this is exactly how \( R_{k+1}(i, w) \) is computed in the algorithm.

---

**Proof of Main Claim**

**Main Claim**

\[ \forall k : 1 \leq k \leq |V| \implies \forall j \in S_k : R_k(i, j) = I(i, j) \bigoplus_{q \in S_k} R_k(i, q) \otimes A(q, j) \]

**Proof:** By induction on \( k \).
Base case: \( S_1 = \{i\} \) and the claim is easy.
Induction: Assume that
\[ \forall j \in S_k : R_k(i, j) = I(i, j) \bigoplus_{q \in S_k} R_k(i, q) \otimes A(q, j) \]

We must show that
\[ \forall j \in S_{k+1} : R_{k+1}(i, j) = I(i, j) \bigoplus_{q \in S_{k+1}} R_{k+1}(i, q) \otimes A(q, j) \]
Since $S_{k+1} = S_k \cup \{q_k+1\}$, this means we must show

1. $\forall j \in S_k: R_{k+1}(i, j) = I(i, j) \oplus \bigoplus_{q \in S_{k+1}} R_{k+1}(i, q) \otimes A(q, j)$
2. $R_{k+1}(i, q_{k+1}) = I(i, q_{k+1}) \oplus \bigoplus_{q \in S_{k+1}} R_{k+1}(i, q) \otimes A(q, q_{k+1})$

By use Observation 1, showing (1) is the same as showing

$$\forall j \in S_k: R_k(i, j) = I(i, j) \oplus \bigoplus_{q \in S_{k+1}} R_k(i, q) \otimes A(q, j),$$

which is equivalent to

$$\forall j \in S_k: R_k(i, j) = I(i, j) \oplus (R_k(i, q_{k+1}) \otimes A(q_{k+1}, j) \oplus \bigoplus_{q \in S_k} R_k(i, q) \otimes A(q, j))$$

By the induction hypothesis, this is equivalent to

$$\forall j \in S_k: R_k(i, j) = R_k(i, j) \oplus (R_k(i, q_{k+1}) \otimes A(q_{k+1}, j)),$$

Put another way,

$$\forall j \in S_k: R_k(i, j) \leq R_k(i, q_{k+1}) \otimes A(q_{k+1}, j)$$

By observation 2 we know $R_k(i, j) \leq R_k(i, q_{k+1})$, and so

$$R_k(i, j) \leq R_k(i, q_{k+1}) \leq R_k(i, q_{k+1}) \otimes A(q_{k+1}, j)$$

by RA.
To show (2), we use Observation 1 and \( I(i, q_{k+1}) = \vec{0} \) to obtain

\[
R_k(i, q_{k+1}) = \bigoplus_{q \in S_{k+1}} R_k(i, q) \otimes A(q, q_{k+1})
\]

which, since \( A(q_{k+1}, q_{k+1}) = \vec{0} \), is the same as

\[
R_k(i, q_{k+1}) = \bigoplus_{q \in S_k} R_k(i, q) \otimes A(q, q_{k+1})
\]

This then follows directly from Observation 3.

---

**Finding Left Local Solutions?**

\[
L = (A \otimes L) \oplus I \iff L^T = (L^T \otimes T A^T) \oplus I
\]

\[
R^T = (A^T \otimes R^T) \oplus I \iff R = (R \otimes A) \oplus I
\]

where

\[
a \otimes^T b = b \otimes a
\]

Replace \( RA \) with \( LA \),

\[
LA : \forall a, b : a \preceq b \otimes a
\]