Material covered:

Deterministic all pairs shortest paths on weighted graphs in linear time. This can be found in section 4.1 of “Stephan Holzer and Roger Wattenhofer. Optimal distributed all pairs shortest paths and applications, Proceedings of the 31st annual ACM Symposium on Principles of Distributed Computing (PODC’12), pages 355-364”. We also covered an animation of this algorithm by Jukka Suomela: http://users.ics.aalto.fi/suomela/apsp/

Optimal deterministic source detection. This can be found in section 4 of “Christoph Lenzen and David Peleg, efficient distributed source detection with limited bandwidth, Proceedings of the 32nd annual ACM Symposium on Principles of Distributed Computing (PODC’13), pages 375-382”. We follow the presentation of Christoph Lenzen, lecture notes 8 “Distance Approximation and Routing” of a class on Theory of Distributed Systems at MPI Saarbruecken: http://resources.mpi-inf.mpg.de/departments/d1/teaching/ws14/ToDS/script/routing.pdf

Fast deterministic computation of small k-dominating sets. This follows the material in section 2 of “Shay Kutten and David Peleg, Fast distributed construction of k-dominating sets and applications, Proceedings of the fourteenth annual ACM symposium on Principles of Distributed Computing (PODC’95), pages 238-251, 1995”.

Deterministic (1+eps)-approximation of the diameter of a unweight graph. This can be found in section 6.2 of “Stephan Holzer and Roger Wattenhofer. Optimal distributed all pairs shortest paths and applications, Proceedings of the 31st annual ACM Symposium on Principles of Distributed Computing (PODC’12), pages 355-364”. However, in this lecture we use the source detection algorithm mentioned above instead of the S-SP algorithm.
LAST LECTURE:

- MIN-CUT

THIS LECTURE:

- DETERMINISTIC, UNWEIGHTED GRAPHS
- ALL-PAIRS SHORTEST PATHS $O(n)$.
- $(S,H,K)$-SOURCE DETECTION IN $H+K+1$.
- $k$-DOMINATING SET OF SIZE $\lceil n^{1/k} \rceil$ IN $O(D)$
- $(1+\epsilon)$-APPROXIMATION OF $D$ IN TIME $O(n/d + D)$

"PIELINING" TECHNIQUES.

DEF: BFS$_v$ DENOTES BFS-TREE / COMPUTATION STARTED IN $v \leftrightarrow v$.

ALGO: (APSP) // ASSUME LEADER $t$ IS GIVEN

compute BFS$_t$-TREE $T$ ROOTED IN $t$;

send a token/pebble $p$ to traverse $T$ IN DFS-WAY;

while $p$ traverses $T$ do

  if $p$ visits a node $v$ not visited before then

    wait one time slot;

    start BFS$_v$-COMPUTATION FROM NODE $v$;

  // computes all distances to $v$.
LEM: IN THE ALGORITHM NO NODE W IS SIMULTANEOUSLY
ACTIVE FOR BOTH BFS_u AND BFS_v.

PROOF: CONSIDER BFS_u / BFS_v STARTED AT TIME t_u > t_v
AT NODES u ? v ? V.

\[ \Rightarrow \forall \text{ P VISITED V AFTER } u \Rightarrow t_v \geq t_u + d(u,v) + 1 \quad (1) \]

AS P TRAVERSES T

P WAITS 1 SLOT

\begin{align*}
\text{CONSIDER } w \neq u, v \text{ INVOLVED IN BFS_u / BFS_v AT TIME } \\
t_u + d(u,w) / t_v + d(v,w) \text{ IF EXECUTED INDEPENDENTLY,} \\
\text{IT IS: } t_v + d(v,w) \geq (t_u + d(u,v) + 1) + d(v,w) \\
\geq t_u + d(u,w) + 1 \\
\neq t_u + d(u,w)
\end{align*}

THM: THE ALGORITHM COMPUTES APSP IN O(n).

PROOF: LEM I IS VALID FOR ANY u ? v ? w ? V.

\[ \Rightarrow \text{ALL MESSAGES CAN BE SENT WITHOUT CONGESTION.} \]

\[ \Rightarrow \text{EACH BFS_v STOPS AFTER D STEPS.} \]

\[ \Rightarrow \text{Runtime P TRAVELS FOR O(n) STEPS} \]

\[ \leq O(n + D) = O(n) \]
REM: • APSP ∈ Ω(m), so this algo is optimal.

• What if we want to compute distances between S and V, S ⊆ V? Can we do O(|S| + d)?

DEF: TOTAL ORDER OF DISTANCE/NODE-PAIRS

\((d_v, v) \prec (d_w, w)\) iff • \(d_v < d_w\), or
\(\text{NO} \quad \text{NO}\) • \(d_v = d_w\) AND \(v \subset w\).

DEF: GIVEN SOURCES \(S \subseteq V\), HOP-BOUND \(H\), SOURCE-BOUND \(V\),
\(L_V := \{ (d(v,s), s) | s \in S \}\)

\(L_V(H) := \{ (d(v,s), s) \in L_V | d(v,s) \leq H \}\)

\(L_V(H,K) := \{ (d(v,s), s) \in L_V(H) | \text{NO} \quad \text{NO}\) \(\exists (d(v,s), s) \in L_V(H) \text{ s.t.} \)
• \(|L_V(H,K)| = \min \{K, |L_V(H)|\}\)

THE SMALLEST \(K\)
ENTRIES OF \(L_V(H)\).

APPLIC: • DISTANCE ORACLES
• COMPACT ROUTING
• STEINER-TREE APPROX.
• (WEIGHTED) APSP APPROX.
ALGO: USE A BFS TREE TO ASSIGN NEW IDS IN $|E|, \ldots, |S|$ TO
NODES IN $S$;  // AGGREGATE + DIVIDE VALUES
FOR $i=1, \ldots, |S|$ DO
NODE WITH NEW ID $i$ STARTS BFS-COMPUTATION.
STOP ATER $H$ ROUNDS.

THM: COMPUTES $(S,H,|S|)$-DETECTION IN $O(D + |S|H)$ TIME.

REM: COMPUTING APSP WITH ALGO I IN $O(n)$ MIGHT
BE FASTER.

ALGO: PIPELINED BELLMAN-FORD:

$Lv := \begin{cases} \emptyset & : v \notin S \\ \{(0,v)\} & : v \in S \end{cases}$  // WILL BE UPDATED, BUT SELPS
ONLY SMALLEST COPY OF $(d_{S}, S)$
PER NODES $S$

FOR $r=1, \ldots, H+k+1$ DO

$L'_{v} := \{a \in L_{v} | a \ NOT SENT BEFORE\}$;

IF $L'_{v} \neq \emptyset$ THEN

$(d_{S}, S) = \min(L'_{v})$;
SEND $(d_{S}+1, S)$ TO NEIGHBORS;

FOR EACH $(d_{S}, S)$ RECEIVED FROM NEIGHBORS DO

$L_{v} := L_{v} \cup \{(d_{S}, S)\}$;

RETURN $L_{v}$;

THM: THE ALGORITHM SOLVES THE $(S, H, U)$-DETECTION
PROBLEM IN TIME $H+k+1$. 
**Proof:** Runtime: \( H + U + 1 \) iterations, each time \( 1 \).

**Correctness:** Few more lemmata.

**Def:** \( L_v^r := \text{Content of } L_v \text{ in round } r \)

**Lem:** \((d(w, w)) \in L_v^r \) implies 1) \( w \in S \), 2) \( d(w) \geq d(v, w) \).

**Proof:** 1) Only these entries with \( w \in S \) are exchanged.
2) \( d(w) \)-value was increased by \( 1 \) in each hop.

**Rem:** 1) \( r' \leq r \implies L_v^r \subseteq L_v^{r'} \) w.r.t. updates \((0, S)\)
2) \( L_v(h, k) \subseteq L_v^r \implies K \) smallest elements of \( L_v^r \) are \( L_v(h, k) \).

**Lem II:** Let \( h \leq H, k \leq K, v \in V \), then:

\[
L_v(h, k) \subseteq \left\{ (d(w, s) + 1, s) \mid (d(w, s), s) \in L_w(h - 1, k) \text{ and } (v, w) \in E \right\}.
\]

Denote the part contributed by \( v \) as \( \text{fixed } \) \( w \in T(v) \) by

\[
= \{ L_w(v, h - 1, k) \}
\]

**Proof:**

\[
d(w, s) \leq L_v(h, k) \leq L_w(h - 1, k)
\]

For any \((d(v, s), s) \in L_v(h, k) \) and \( w \in T(v) \) on a shortest \( s-v \)-path, it is

\[
d(w, s) = d(v, s) - 1 \geq h - 1 \implies (d(w, s), s) \in L_w(h - 1).
\]
ASSUME \((d(w, s), s) \in L_w(h-1, k)\)

\Rightarrow \text{THERE ARE } k \text{ ELEMENTS } \((d(w', s'), s') \in L_w(h-1)\)

\text{s.t. } \((d(w, s'), s') \subseteq (d(w, s^*), s^*)\).

\Rightarrow \((d(v', s'), s') \subseteq (d(v, s), s)\)

AS \(w\) IS ON SHORTEST \((v, s)\)-PATH.

Furthermore \((d(v', s'), s') \in L(h), AS\)

\(d(v', s') \leq d(v, s) \leq h\)

\Rightarrow \text{THERE ARE } k \text{ ELEMENTS IN } L(h) \text{ SMALLER THAN } (d(v', s'))

\(\forall \text{Def. of } L_V(h, k)\).

**LEM:** \text{LET } v \in V, r \in \{0, \ldots, h+k+1\}, h+k \leq r+1, \text{ THEN IT IS}

(1) \(L_V(h, k) \subseteq L_V^r\), AND

(2) \(v\) HAS SENT \(L_w(v, h, k)\) BY END OF ROUND \(r+1\), TO EACH \(w \in T(V)\).

**PROOF:** INDUCTION ON \(r\).

\(r = 0, \text{ i.e. } h+k = 1:\) \text{ CASE } h = 0: \text{ IT IS } L_V(0, k) = \{v \in S : V \subseteq S\}

\text{FOR ALL } k \in \{0, 1\} \text{ AND } = L_V^0.

\text{CASE } h = 1: \Rightarrow k = 0 \Rightarrow L_V(h, 0) = \emptyset \subseteq L_V^0

\text{ THIS PROVES (1), AS } |L_V(h, k)| \leq 1, L_V(h, k)

\text{ WILL BE SENT IN ROUND } r+1, \text{ WHICH PROVES (2).}

\(r \rightarrow r+1, \text{ i.e.}:\) \text{ ASSUME (1) AND (2) FOR } h + k = r+1.

\text{WE PROVE (1) AND (2) FOR } h + k = r+2.

\(\Box\)
CASE \( h = 0 \): 
\[ L_v(0, k) = \{ \emptyset \} : v \in S \] 
for \( k \in \{0, \ldots, t+2\} \) 

\[ \leq L_v^{t+1}. \]

CASE \( h > 0 \): WE SHOW STATEMENT (1):

For \( h' + h' = t+1 \), v's neighbors \( \exists w \in S \) each sent \( L_{v,w}(h', h') \) to v in round \( t+1 \).

\[ \Rightarrow \] v knows \( L_v(h'+1, h') \) by LEM II.

Similarly v received \( L_{v,w}(h'-1, h'+1) \)

\[ \Rightarrow \] v knows \( L_v(h', h'+1) \) by LEM II.

Therefore it is

\[ L_v(h, k) \leq L_v(h'+1, h') U L_v(h', h'+1) \]

as \( h'+h'+1 = h+k \)

\[ \leq \bigcup_{w \in T(v)} (L_{v,w}(h', h') U L_{v,w}(h'-1, h'+1)) \]

\[ \leq L_v^{t+1} \]

\[ \Rightarrow \] STATEMENT (1)

WE NOW SHOW STATEMENT (2):

- After round \( t+1 \), v knows \( L_v^*(h, k) \). This follows from what we just proved.

- By induction hypothesis, Statement (2), \( L_v(h', k') \) is already sent by round \( t+1 = h'+k' \).

Choose \( h' = h \) and \( k' = k^* - 1 \) already sent.

\[ \Rightarrow \] ONLY \( L_v(h, k) \setminus L_v(h, k-1) \) not sent yet.

contains at most 1 element \((k-(k-1))\)
**Theorem:** There is a $k$-dominating set of size $O(n/k)$. On any graph $G$.

**Proof:** Let $T$ be a BFS-tree of $G$. Let $l(v)$ be the level of $v$ in $T$. Define $T_i := \{v \in V \mid l(v) = i\}$ for $i = 0, \ldots, \text{depth}(T)$. Then:

$$D_i := \bigcup_{j=0}^{i} T_i + \lfloor k(i+1) \rfloor \quad \text{for} \quad i = 0, \ldots, k.$$ 

As there are $n$ nodes and $k+1$ sets $D_i$, there is one $D_i$ of size $O(n/k)$. Clearly, $D_i$ is $k$-dominating. 

---

**The Algorithm Makes Sure That**

The smallest element of $L^r_v$ gets sent.

**Rem.** If $L_v(h, k) \neq \emptyset$ and $L_v(h, k-1) \neq \emptyset$, it is exactly $\min(L^r_v)$.

This concludes statement (2) for $r+1$.

**Rem:** The BFS-computations can be derived from the distances and are pipelined via messages $(d(v,s), s)$, while there are further messages $(d_s, s)$ with $d_s > d(v,s)$ sent/discard.

This completes the proof of Thm. 0
The following algorithm computes a $k$-dom set of size $O(n/k)$ in time $O(d)$.

**Algo:**

<table>
<thead>
<tr>
<th>$N(v, i) = 0$; (for all $i \in {0, \ldots, \text{depth}(T)}$)</th>
</tr>
</thead>
</table>

**Participate in computing $T$ and $l(v)$;**

<table>
<thead>
<tr>
<th>$N(v, l(v)) = 1$;</th>
</tr>
</thead>
</table>

**For $i = l(v), \ldots, \text{depth}(T)$ do**

<table>
<thead>
<tr>
<th>Send $N(v, i)$ to parent in $T$;</th>
</tr>
</thead>
</table>

**Receive $N(w, i+1)$ from each child $w$ in $T$;**

<table>
<thead>
<tr>
<th>$N(v, i+1) := \sum_{w \in T(v) \setminus \text{PARENT}} N(w, i+1)$;</th>
</tr>
</thead>
</table>

**Root now computes $|D_{i'}| := \sum_{\gamma = 0}^{\left\lfloor n/k \right\rfloor} N(\text{root}, k\gamma + l)$;**

**Root computes $i' := \arg\min_{i \in \{0, \ldots, k\}} |D_{i'}|$;**

**Return $D_{i'}$;**

---

**Ex:**

From level: 6, 7, 2

↑ Corresponds to one round of sending $N(v, i)$ up the tree.
**THM:** The following algorithm computes a $(1+\varepsilon)$-approximation of the diameter in $O(\frac{n}{\varepsilon} + D)$ rounds.

**ALGO:**
1. Compute $D' := 2 \cdot \text{depth}(T)$; // $T$ is a BFS tree
2. Compute $k := \lceil \frac{\varepsilon D'}{4} \rceil$;
3. Compute $\ell$-dominating set $\text{DOM}$ of size $O(n/k)$;
4. Perform $(\text{DOM}, D', \text{DOM})$-detection;
5. $d_{\text{max}} := \max_{u \in \text{DOM}} d(u,v)$; // Aggregate via BFS-tree
6. Return $(1+\varepsilon) \cdot d_{\text{max}}$;

**RUNTIME:** Computing $D', \text{DOM}, d_{\text{max}}$ take $O(D)$ time each. Performing $(\text{DOM}, D', \text{DOM})$-detection takes $D' + |\text{DOM}| + 1 = O(D + n/k) = O(D + \frac{n}{\varepsilon})$.

**APPROX-FACTOR:** $D'$ is a 2-approximation of $D$.

1. It is $d_{\text{max}} \leq D$ $\Rightarrow$ $(1+\varepsilon) d_{\text{max}} \leq (1+\varepsilon) D$.
2. It is $d_{\text{max}} \geq D - \ell k$, as $\text{DOM}$ is an $\ell$-dominating set. $\Rightarrow$ $(1+\varepsilon) d_{\text{max}} \geq (1+\varepsilon) D \cdot (1 - \frac{\varepsilon D'}{4}) \geq D$.

$(1)+(2)$ yield $D \leq (1+\varepsilon) d_{\text{max}} \leq (1+\varepsilon) D$.

$\Rightarrow$ $(1+\varepsilon) d_{\text{max}}$ is a $(1+\varepsilon)$-approximation of $D$. 

$\blacksquare$