Streamable Graphs

(Please note that this is a preliminary draft, and mistakes are possible.)

1 Preliminaries

▶ **Definition 1.** *Let* $G = (V, E)$ *with* $|V| = n$*. Let* $\sigma : V \mapsto [n]$ *be an ordering of the vertex of* G*.*

For each subset $U \subset V$*, we use* $\sigma|_U$ *to denote the sub-ordering of* σ *induced by* U*. A set* $U \subset V$ *is streamed in* σ *if*

$$\left((\sigma|_U)^{-1}(1), (\sigma|_U)^{-1}(2), \ldots, (\sigma|_U)^{-1}(|U|)\right) \tag{1}$$

is a path in G*.*

▶ **Definition 2.** *Let* $N(v)$ *denote the set of neighbors of* v *in* G*, and*

$$N^+_\sigma(v) := \{u \in N(v) : \sigma(u) > \sigma(v)\}. \tag{2}$$

We will refer to $N^+_\sigma(v)$ *as the succeeding neighbors of* v*.*

▶ **Definition 3.** *A graph* G *is streamable if there exists an ordering* $\sigma : V \mapsto [n]$ *such that for all* $v \in V$*,* $N^+_\sigma(v)$ *is streamed in* σ*. We call such ordering a streaming ordering of* G*.*

For example, all chordal graphs are streamable, as witnessed by any perfect elimination ordering.While C_4 is not streamable, W_4 is. Thus, being streamable is not hereditary.

2 The NP-completeness of recognition

▶ **Theorem 4.** *Recognition of streamable graphs is* NP*-complete.*

Clearly, the recognition of streamable graphs is in NP, as an ordering σ of the vertex set of the graph is a certificate. So we now only need to show the NP-hardness by reducing the following betweenness problem into the recognition of streamable graphs.

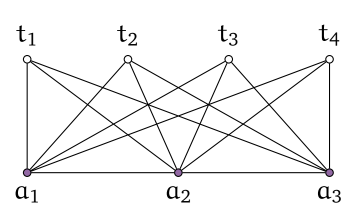
▶ **Definition 5 (Betweeness[1]).** *Given a finite set* S *and a set of ordered triples* $R \subset S \times S \times S$*, the betweenness problem on* (R, S) *asks to determine whether there exists a total ordering of S such that for every triple* (x, y, z) *in* R*, either* $x < y < z$ *or* $z < y < x$*.*

▶ **Theorem 6 ([1]).** *The betweenness problem is* NP*-complete.*

Now we introduce the following lemma that helps us to establish connection between the two problems:

▶ **Lemma 7.** *In a graph* $G = (V, E)$*, for any independent set* I *of size* k*, if there exists a set of vertices* $A \subset V$ *with* $N(v) = A$ *for all* $v \in I$ *and* $|A| \leqslant k - 1$*, then in any streaming ordering* σ *of* G*,* A *is streamed in* σ*.*

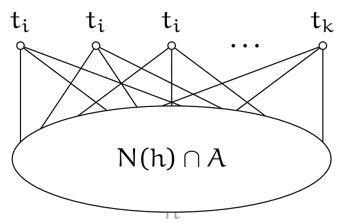
This lemma is somewhat ambiguous, and we refer to the following figure for an illustration:



For the graph above, Lemma 7 tells us $\{a_1, a_2, a_3\}$ must be streamd in σ. In this figure, this means either $\sigma(a_1) < \sigma(a_2) < \sigma(a_3)$ or $\sigma(a_3) < \sigma(a_2) < \sigma(a_1)$ since there are only two possible Hamiltonian paths in the subgraph induced by $\{a_1, a_2, a_3\}$. Now we proof Lemma 7.

Proof of Lemma 7. Given a streaming ordering σ, we let $h := \sigma^{-1}(1)$. We remark that $N(h) = N^+_\sigma(h)$, i.e., all neighbours of h are succeeding neighbours of it since it is the first one in the ordering, and thus $N(h)$ should be streamed in σ.

We now show $h \in I$ by contradiction. Assuming $h \in A$, h could have (open) neighbourhood of the following form:

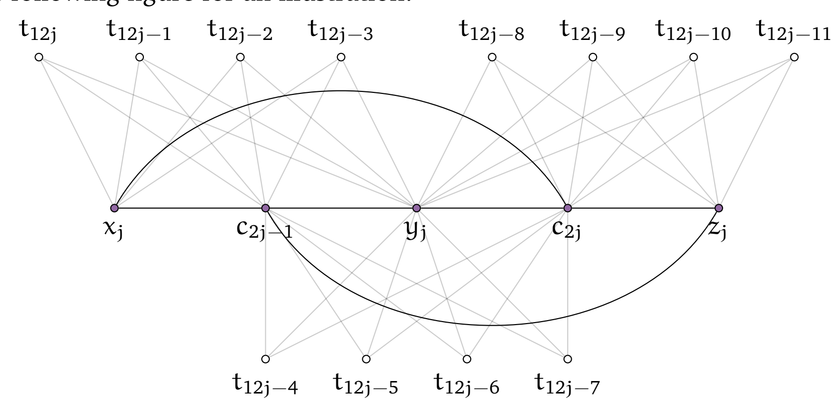


However, the neighbourhood is not possible to be streamed in any ordering as there is no Hamiltonian path, because in a Hamiltonian path, the $|N(h) \cap A|$ bottom vertices can only allow $|N(h) \cap A|+1 \leqslant |A| \leqslant k-1$ alternations between the two rows, while there are k vertices in the top row, making it impossible to for them to fit in any Hamiltonian path. Therefore, $h \in T$, and $N(h) = A$ should be streamed in σ. ◀

Now we describe the construction of graph $G_{R,S}$ given a betweenness problem (R, S) with $R := \{(x_j, y_j, z_j) : j \in [|R|], x_j, y_j, z_j \in S\}$.

1. Create the following sets of vertices:
   a. V_S of $|S|$ vertices, each of which corresponds to an element in S;
   b. V_C of $2|R|+|S|$ vertices and denote the vertices by $V_C := \{c_i\}_{i\in[2|R|+|S|]}$;
   c. V_T of $12|R|$ vertices and denote the vertices by $V_T := \{t_i\}_{i\in[9|R|]}$.
2. Complete (V_S, V_C) to a complete bipartite graph.
3. Complete V_C to a clique, excluding the following edges: $\{(c_{2j-1}, c_{2j}) : j \in [|R|]\}$.
4. $\forall j \in [|R|]$, connect
   a. each vertex in $\{t_{12j-w}\}_{w\in[0,3]}$ to $\{x_j, c_{2j-1}, y_j\}$,
   b. each vertex in $\{t_{12j-w}\}_{w\in[4,7]}$ to $\{c_{2j-1}, y_j, c_{2j}\}$,
   c. each vertex in $\{t_{12j-w}\}_{w\in[8,11]}$ to $\{y_j, c_{2j}, z_j\}$.

   See the following figure for an illustration:



.

Proof of Theorem 4. The betweenness problem (R, S) has a satisfying solution if and only if $G_{R,S}$ is streamable.

Suppose $G_{R,S}$ is a streamable graph, and there is a streaming ordering $\sigma : V(G_{R,S}) \mapsto [|V(G_{R,S})|]$ of $G_{R,S}$. We assert that the sub-ordering of σ induced by V_S is a solution to the betweenness problem (R, S). That is because due to Lemma 7, $\forall j \in [|R|]$,

1. $\{t_{12j-w}\}_{w\in[0,3]}$ enforces $\{x_j, c_{2j-1}, y_j\}$ to be streamed in σ,
2. $\{t_{12j-w}\}_{w\in[4,7]}$ enforces $\{c_{2j-1}, y_j, c_{2j}\}$ to be streamed in σ,
3. $\{t_{12j-w}\}_{w\in[8,11]}$ enforces $\{y_j, c_{2j}, z_j\}$ to be streamed in σ.

In either of the three cases above, the three vertices induce a path and there is only two valid Hamiltonian paths in the induced subgraph, so the three constraints are equivalent to the following:

1. $\sigma(x_j) < \sigma(c_{2j-1}) < \sigma(y_j)$ or $\sigma(y_j) < \sigma(c_{2j-1}) < \sigma(x_j)$,
2. $\sigma(c_{2j-1}) < \sigma(y_j) < \sigma(c_{2j})$ or $\sigma(c_{2j}) < \sigma(y_j) < \sigma(c_{2j-1})$,
3. $\sigma(y_j) < \sigma(c_{2j}) < \sigma(z_j)$ or $\sigma(z_j) < \sigma(c_{2j}) < \sigma(y_j)$.

And it is easy to see that to satisfy the three constraints above, we must have $\sigma(x_j) < \sigma(c_{2j-1}) < \sigma(y_j) < \sigma(c_{2j}) < \sigma(z_j)$ or $\sigma(z_j) < \sigma(c_{2j}) < \sigma(y_j) < \sigma(c_{2j-1}) < \sigma(x_j)$, and in either case y_j lies between x_j and z_j. Therefore, the sub-ordering induced by V_S is a solution to the betweenness problem (R, S).

Now suppose S has an ordering σ that satisfies the betweenness problem, we can construct a streaming ordering of $V(G_{R,S})$. In the following construction, we will use π to denote the constructed sequence of vertices, which will finally grow to a permutation of $V(G_{R,S})$ and the streaming ordering will be π^{-1}. Initially, π is empty.

1. Append all the V_T vertices in arbitrary order to π.
2. Append all the V_S vertices in the order of σ to π.
3. $\forall s \in S$, insert $c_{2|S|+\sigma(s)}$ before s (σ can be replaced with any function that maps S to $[|S|]$).
4. $\forall j \in [|R|]$, find x_j, y_j, z_j in π. If they are in the order of x_j, y_j, z_j, we insert c_{2j-1} after x_j and c_{2j} after y_j; otherwise, insert c_{2j} after z_j and c_{2j-1} after y_j.

Now we verify that for all vertices, the succeeding neighbours are streamed in $\tau := \pi^{-1}$.

1. V_T: $\forall j \in [|R|], w \in [0, 11]$, $N^+_\tau(t_{12j-w}) = N(t_{12j-w})$, and the path is completed by the insertion of c_{2j-1}, c_{2j}.
2. V_S: $s \in V_S$, $N^+_\tau(s)$ is a subset of V_C, which almost induce a clique. We claim, in the sub-ordering induced by $N^+_\tau(s)$, there will not be a pair of c vertices that are adjacent while disconnected, namely, $\nexists\, u, v \in V_C, (v, u) \notin E \wedge \tau|_{N^+_\tau(c)}(u) = \tau|_{N^+_\tau(c)}(v) + 1$. This is because, $\forall j \in [|R|]$, there will be a y_j between (c_{2j-1}, c_{2j}), so $c_{2|S|+\sigma(y_j)}$ will also lie between these two. Moreover, $c_{2|S|+\sigma(y_j)} \in N^+_\tau(s)$, so the claim is true and $N^+_\tau(s)$ must be streamed in σ.
3. V_C: $\forall c \in V_C$, $N^+_\tau(c)$ is the union of a subset of V_C and a subset of V_S. It is streamed because there are no two vertices in V_S that are adjacent in the sub-ordering induced by $N^+_\tau(c)$, namely, $\nexists\, u, v \in V_S, \tau|_{N^+_\tau(c)}(u) = \tau|_{N^+_\tau(c)}(v) + 1$. The argument is similar to the one for V_S.

◀

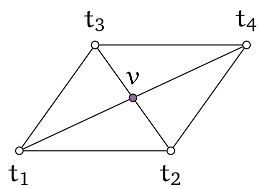
3 Search-to-decision reduction

▶ **Theorem 8.** *Suppose we have an oracle $\mathcal{O}_s$ that determines whether a graph G is streamable. Then, for a streamable graph G, there exists a polynomial-time algorithm that produces a streaming ordering of G using a polynomial number of queries to $\mathcal{O}_s$.*

The idea is to reconstruct a streaming ordering step-by-step, from the last vertex to the first. Specifically, we will use the oracle to find a vertex v such that there is a streaming ordering that ends with v, and then find a vertex w such that there is a streaming ordering that ends with (w, v), and repeat the process until the ordering is complete.

Before we proceed, we need the following lemma:

▶ **Lemma 9.** *In any streaming ordering* $\sigma : \{v\} \cup \{t_b\}_{b \in [4]} \mapsto [5]$ *of the following graph*



,

we must have $\sigma^{-1}(5) \in \{t_b\}_{b\in[4]}$*, i.e., the ordering must not end with* v.

Proof. Suppose there is a streaming ordering σ with $\sigma^{-1}(5) = v$, which means σ starts with a prefix which is a permutation of $\{t_b\}_{b\in[4]}$. Since the σ is a streaming ordering, the prefix should be a streaming ordering for the subgraph induced by $\{t_b\}_{b\in[4]}$, but that induced subgraph is a cycle and no way an streamable graph. Thus we have a contradiction. ◀

We will make use of the lemma above to achieve the following: given a sequence of vertices $w_1, w_2, \dots, w_\ell$ determine if there is a streaming ordering σ with the sequence as a suffix, i.e., $\sigma(w_i) = n - \ell + i$ for all $i \in [\ell]$. Now we elaborate our construction.

Given G and $w = (w_1, w_2, \dots, w_\ell)$, we create $G' := c(G, w)$ as follows:

1. Let $G' := G$.
2. $\forall i \in [\ell]$, create 4 new vertices $t_{i,1}, t_{i,2}, t_{i,3}, t_{i,4}$, and add edges $\{(w_i, t_{i,b}) : b \in [4]\} \cup \{(t_{i,b}, t_{i,b \bmod 4+1}) : b \in [4]\}$ to G'.
3. $\forall i, j \in [\ell]$ with $i < j$, if $(w_i, w_j) \in E$, then add edge $(t_{i,1}, w_j)$ to G'.

Then we have the following lemma:

▶ **Lemma 10.** *In a streaming order* σ' *for* G', $\forall 1 \leqslant i < j \leqslant \ell, (w_i, w_j) \in E$*, we have* $\sigma'(w_i) < \sigma'(t_{i,1}) < \sigma'(w_j)$.

Proof. We mention the following remark:

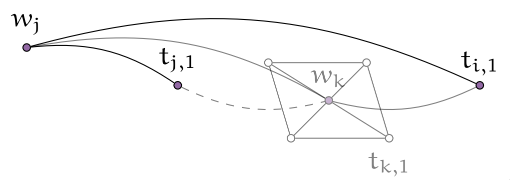
▶ Remark 11. $\forall i \in [\ell]$, if $\exists v \in V'$ with $(w_i, v) \in E'$ and $\sigma'(w_i) < \sigma'(v)$, then

$$\max(\sigma'(t_{i,b}) : b \in \{2,3,4\}) < \sigma'(u_i) < \sigma'(t_{i,1}). \tag{3}$$

Now we first prove the second inequality by contradiction. Namely, we prove

$$\forall 1 \leqslant i < j \leqslant \ell \text{ with } (w_i, w_j) \in E, \sigma'(t_{i,1}) < \sigma'(w_j). \tag{4}$$

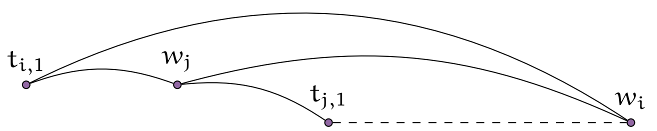
Suppose $\exists i, j, i < j$ with $(w_i, w_j) \in E, \sigma'(t_{i,1}) > \sigma'(w_j)$, we find such a pair (i, j) with the largest $\sigma'(w_j)$. Recall that $(t_{i,1}, w_j) \in E'$ since $i < j$. Then by Remark 11, we have $\sigma'(w_j) < \min\{\sigma'(t_{j,1}), \sigma'(t_{i,1})\}$. Without loss of generality, we assume $\sigma'(t_{j,1}) < \sigma'(t_{i,1})$. Then for the succeeding neighbors of w_j to be streamed, there must be a path of neighbors of w_j from $t_{j,1}$ to $t_{i,1}$. We consider the second last vertex on this path, and denote it by w_k, looking as follows:



Note that since $(w_k, t_{i,1}) \in E'$, we know $i < k$. Still, due to Remark 11, we have $\sigma'(w_k) < \min\{\sigma'(t_{k,1}), \sigma'(t_{i,1})\}$, giving us another pair (i, k) with $\sigma'(w_k) > \sigma'(t_{j,1}) > \sigma'(w_j)$, contradicting the assumption of (i, j) being the pair with the largest $\sigma'(w_j)$. Therefore, there will not be such a pair (i, j) and Equation (4) must hold. Now it remains to show

$$\forall 1 \leqslant i < j \leqslant \ell \text{ with } (w_i, w_j) \in E, \sigma'(w_i) < \sigma'(w_j). \tag{5}$$

It is easy to see that Equations (4) and (5) together with Remark 11 prove the lemma. Suppose $\exists i, j, i < j$ with $(w_i, w_j) \in E$ and $\sigma'(w_i) > \sigma'(w_j)$, then by Remark 11, $\sigma'(w_j) < \sigma'(t_{j,1})$. Similarly, for the succeeding neighbors of w_j to be streamed, there must exist a path of neighbors of w_j from $t_{j,1}$ to w_i. However, by Equation (4), $\sigma'(t_{i,1}) < \sigma'(w_j)$ since $i < j$, then by Remark 11 w_i has no succeeding neighbors. This means it can only appear on the right end of the aforementioned path, giving us $\sigma'(t_{j,1}) < \sigma'(w_i)$, as shown below:



Now we select a pair (i, j) that $(w_i, w_j) \in E$ and $\sigma'(w_i) > \sigma'(w_j)$ with the smallest $\sigma'(w_i) - \sigma'(w_j)$, and look at the second last vertex on this path. There are two different cases:

1. $t_{k,1}$ for some k: then since $(t_{k,1}, w_i) \in E'$, we know $k < i < j$, giving us a pair (k, j) violating (4), thus impossible.
2. w_k for some k: then we know $j < k$ since otherwise (k, j) would be a pair with smaller $\sigma'(w_k) - \sigma'(w_j)$. However, this gives us $i < j < k$, meaning that (i, k) is a pair with smaller $\sigma'(w_k) - \sigma'(w_i)$, contradicting the assumption.

Thus we finalize the proof of (5), together with (4) we complete the proof of Lemma 10. ◀

▶ **Lemma 12.** *In a streaming order σ' for G', $\forall v \in V \setminus w, i \in [\ell]$ with $(v, w_i) \in E$, we have $\sigma'(v) < \sigma'(w_i)$.*

Proof. Suppose there is such (v, i) with $\sigma'(w_i) < \sigma'(v)$, we then find a pair with the smallest $\sigma'(v) - \sigma'(w_i)$. In this case, if we look at the path $N^+_{\sigma'}(w_i)$, it is easy to see that all $N^+_{\sigma'}(w_i) \cap w$ will appear after v in this path. However, by Remark 11, $t_{i,1} \in N^+_{\sigma'}(w_i)$, and by Lemma 10, $t_{i,1}$ appear before all $N^+_{\sigma'}(w_i) \cap w$. In this case, there cannot be a path between $t_{i,1}$ and v, and thus there will be no such (v, i). ◀

The lemma above says that all w vertices should appear after $V \setminus w$ vertices (if they are connected). Lemma 10 and Lemma 12 together give us enough structural information to determine if there is a streaming ordering with w as a suffix. Finally, we have the following lemma:

▶ **Lemma 13.** *Given G, w, there is a streaming ordering σ of G with w as a suffix if and only if G' is streamable.*

Proof. Suppose G' is streamable with a streaming ordering σ', and we use $\sigma_{V\setminus w} := \sigma'|_{V\setminus w}$ to denote the restriction of σ' to $V \setminus w$. Then a streaming ordering σ of G with w as a suffix can be defined as follows:

$$\sigma^{-1} = \left(\sigma^{-1}_{V\setminus w}(1), \sigma^{-1}_{V\setminus w}(2), \dots, \sigma^{-1}_{V\setminus w}(n-k), w_1, w_2, \dots, w_k\right). \tag{6}$$

To see σ is a streaming ordering, we consider all vertices in V and argue their succeeding neighbors are streamed in σ. For $u \in V$, we denote the succeeding neighbors of u in (G, σ) by $N^+_\sigma(u)$ and the succeeding neighbors of u in (G', σ') by $N^{+'}_{\sigma'}(u)$. We first mention the following corollary which is a consequence of Lemma 10 and Lemma 12.

▶ **Corollary 14.** $\forall (u, v) \in E' \cap V \times V$, *if* $\sigma'(u) < \sigma'(v)$, *then* $\sigma(u) < \sigma(v)$.

Then we consider the following two cases:

1. $\forall v \in V \setminus w$: $N^{+'}_{\sigma'}(v) = N^+_\sigma(v)$. Then by Corollary 14, if the first set is streamed in σ', then the second set is streamed in σ.
2. $\forall i \in [\ell], w_i$: $N^{+'}_{\sigma'}(w_i) = N^+_\sigma(w_i) \cup \{t_{i,1}\}$. Moreover, by Lemma 10, $\sigma'(t_{i,1}) < \min_{u \in N^+_\sigma(w_i)} \sigma'(u)$, meaning that $N^+_\sigma(w_i)$ can be streamed in σ' without $t_{i,1}$, thus by Corollary 14 $N^+_\sigma(w_i)$ is streamed in σ.

Conversely, suppose there is a streaming ordering σ of G with w as a suffix, then there is a streaming ordering σ' of G' defined as follows:

$$\begin{aligned}\sigma'^{-1} = (&\sigma^{-1}(1), \sigma^{-1}(2), \dots, \sigma^{-1}(n-k),\\ &t_{1,2}, t_{1,3}, t_{1,4}, w_1, t_{1,1},\\ &t_{2,2}, t_{2,3}, t_{2,4}, w_2, t_{2,1},\\ &\dots,\\ &t_{k,2}, t_{k,3}, t_{k,4}, w_k, t_{k,1}),\end{aligned} \tag{7}$$

and the lemma is proved. ◀

Based on Lemma 13, we can give the algorithm to find a streaming ordering of G in Algorithm 1. Clearly, the algorithm terminates in polynomial time and makes $O(n^2)$ queries to $\mathcal{O}_s$.

References

1 J Opatrny. Total Ordering Problem. *SIAM Journal on Computing*, 8(1):111–114, 1979. doi:10.1137/0208008.

```
Function CHECK-SUFFIX(G, w):
    G′ ← c(G, w);
    return O_s(G′);
Function STREAMING-ORDERING(G):
    if not O_s(G) then
        return ∅;
    end
    W ← [];
    for i ← 1, 2, ..., n do
        for u ∈ V(G) \ w do
            if CHECK-SUFFIX(G, [u] + w) then
                W ← [u] + w;
                break;
            end
        end
    end
    return W^{-1};
```

Algorithm 1 Finding a streaming ordering of G.