

Lecture 14

Graphs with cycles

In the presentation up until this point, we have exclusively considered directed acyclic graphs. All the code construction algorithms that we have considered have implicitly used the assumption of acyclicity. For example, in the Linear-Information-Flow (LIF) algorithm, the assignment of a code vector for a given edge can be done because we assume that the assignment for edges that feed into it has already been done. In acyclic graphs, a numbering of the edges can be done so that only lower-numbered edges feed into higher-numbered edges. The code vectors can then be assigned according to this order. However, in cyclic graphs, such an ordering is impossible. Also, the algebraic approach that we presented earlier needs to be modified by introducing a delay parameter. There are also some technicalities that need to be taken care of (please see the paper by Koetter and Medárd for the details). Here, we present an alternate approach, which was demonstrated in the original paper of Ahlswede et al. The basic idea is to consider the original cyclic graph over a long time period and construct an appropriate acyclic graph over which our previous approaches can be used. The maximum-flow over the acyclic graph is very close to the maximum-flow over the original graph when we normalize by the time parameter. Basically, this yields a network coding solution where the linear combinations that the nodes have to perform are time-varying.

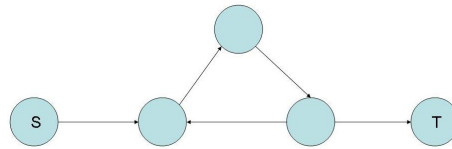


Figure 1: An example of a simple graph with cycles.

Assume we have a network with the directed cyclic graph $G = (v, e)$ with source S and terminal T_1, T_2, \dots, T_i with $\maxflow(s, T_i) \geq h$. From this we can construct a directed acyclic graph $G^* = (V^*, E^*)$ in the following manner.

- (1) The set of nodes $V^* = \bigcup_{\lambda=0}^{\Lambda} V^{(\lambda)}$ where λ is a time parameter such that $0 \leq \lambda \leq \Lambda$, where each V^λ is a copy of V at time λ .
- (2) The set of edges, denoted E^* has three types of edges.
 - (a) $(s^{(0)}, s^{(\lambda)}), 1 \leq \lambda \leq \Lambda$
 - (b) $(T_i^{(\lambda)}, T_i^{(\Lambda)}), 1 \leq \lambda \leq \Lambda - 1$
 - (c) $(i^{(\lambda)}, j^{(\lambda+1)}), (i, j) \in E, 0 \leq \lambda \leq \Lambda - 1$

We denote $s^* = s^{(0)}$ and $t_i^* = t_i^{(\Lambda)}$.

The capacity assignment for the edges in G^* is as follows.

$$C_{u,v}^* = \begin{cases} C_{i,j} & \text{if } (u,v) = (i^{(\lambda)}, j^{(\lambda+1)}) \text{ and } (i,j) \in E, 1 \leq \lambda \leq \Lambda - 1. \\ \infty & \text{otherwise} \end{cases}$$

On inspection, we realize that G^* is acyclic since the edges only go from nodes in lower layers to nodes in higher layers.

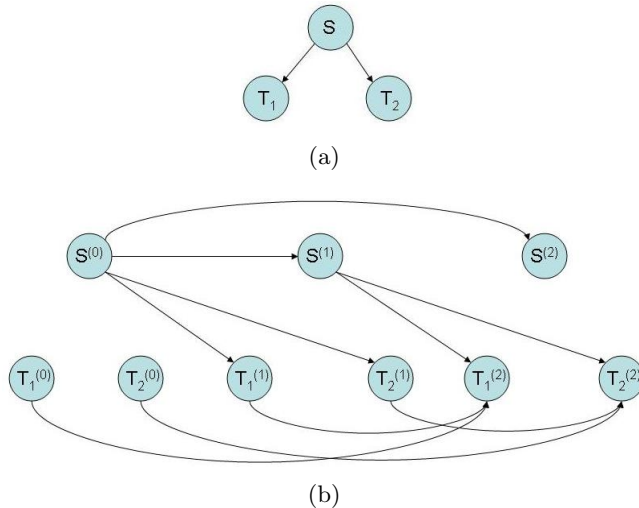


Figure 2: (a) A simple multicast network. (b) Example of the graph G^* for the simple network. In this case, $\Lambda = 2$. The max-flow from $S^{(0)}$ to $T_1^{(2)}$ and $S^{(0)}$ to $T_2^{(2)}$ is two.

A simple example is shown in Fig. 2(b). Of course, in this example G itself is acyclic, however it serves to demonstrate the point.

Suppose that the max-flow from source S to T_l in G is h . It can be shown that the max-flow from s^* to T_l^* in G^* is $(\Lambda - c_1)h$, where c_1 is a constant independent of Λ and h (we shall show this in the next lecture). Note that Λ has the interpretation of being a time parameter. Therefore the normalized (w.r.t. time) max-flow from s^* to T_l^* is $\frac{(\Lambda - c_1)}{\Lambda}h$. It is easy to see that this quantity can be made arbitrarily close to h if Λ is chosen large enough.

Once the directed acyclic graph G^* has been constructed we simply apply the approaches developed earlier for multicast network code design to G^* . Note that in G^* there are copies of the vertices in G corresponding to each time instant. So the local coding vectors for the nodes $v_1^{(i)}$, $i = 0, \dots, \Lambda$ in G^* equivalently indicate the local coding vectors that node $v_1 \in G$ needs to use at each time instant $\lambda \in \{0, \dots, \Lambda\}$ i.e. in the cyclic case the coding solution may be time-varying.

A natural question to ask is whether cycles are essential in maintaining the max-flow to each of the terminals from the source i.e. if G is a cyclic graph, is there an acyclic subgraph of G that preserves the max-flow to each of the terminals. We now show that in general this is not true by presenting an example of a multicast network that demonstrates that, in general, cycles may be essential for the maximum flow from a source to a terminal to be high enough. Consider the graph in Fig. 3 with source node s and terminals t_0 and t_1 . It has a directed cycle formed by nodes u_0, u_1 and u_2 and the max-flow from s to each terminal is three. Notice that to break the cycle,

either (u_0, u_1) , (u_1, u_2) , or (u_2, u_0) need to be removed. However, removing (u_0, u_1) or (u_1, u_2) will cause a drop in max-flow to t_0 . Likewise, removing (u_2, u_0) will cause a drop in max-flow to t_1 . Therefore, in general, the cycle cannot be broken without incurring a penalty on the transmission rate to the terminals. The approach presented before tries to break the cycle over a long period of time so that the penalty can be made arbitrarily small.

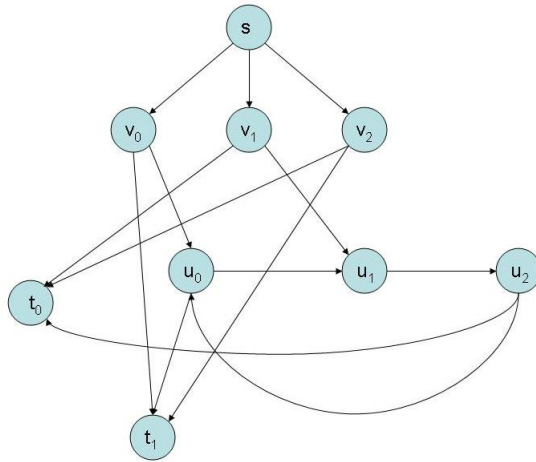


Figure 3: Example of a cyclic graph where the cycle cannot be broken without incurring a penalty in the max-flow to at least one terminal.

$$\begin{aligned} \text{Paths from } s \text{ to } t_0 : & \begin{cases} s \rightarrow v_1 \rightarrow t_0 \\ s \rightarrow v_2 \rightarrow t_0 \\ s \rightarrow v_0 \rightarrow u_0 \rightarrow u_1 \rightarrow u_2 \rightarrow t_0 \end{cases} \\ \text{Paths from } s \text{ to } t_1 : & \begin{cases} s \rightarrow v_0 \rightarrow t_1 \\ s \rightarrow v_2 \rightarrow t_1 \\ s \rightarrow v_1 \rightarrow u_1 \rightarrow u_2 \rightarrow u_0 \rightarrow t_1 \end{cases} \end{aligned}$$