EE 382N: Distributed Systems **Fall 2017**

Lecture 3: September 7

Lecturer: Vijay Garg *Scribe: Boya Wang, Cameron Chalk*

3.1 Outline

- Dilworth's Theorem
- Chain Clocks
- Verification of Vector Clock

3.2 Dilworth's Theorem

3.2.1 Proof of Dilworth's Theorem

In this section we cover Dilworth's Theorem, which gives the necessary and sufficient number of chains required to construct a chain cover of a poset (a chain cover of a poset is equivalent to a partition of the poset into chains). Another write-up of the proof for further reference can be found in [Gal94]. Formally,

Theorem 3.1 *Given a poset P of width w, w chains are necessary and sufficient to cover P.*

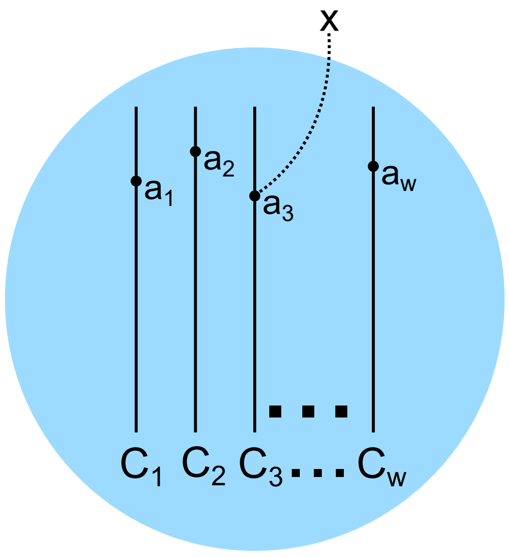

Figure 3.1: The decomposition of P into w chains $\{C_1, C_2, \ldots, C_w\}$, given by the inductive hypothesis. $a_1, a_2, \ldots, a_w$ are the maximum element in each chain that belongs to some antichain.

Proof: First, note that it is trivial to say that w chains are necessary to cover P. Consider the antichain of size w; each element must be in its own chain (since they are incomparable), so at least w chains are needed.

We will show that w chains are sufficient via induction on the size of the poset. The inductive hypothesis is as follows: For a poset of size $n-1$ and width w, w chains are sufficient to cover the poset. For the inductive step, consider the poset P shown abstractly in Figure 3.1, covered by w chains, $\{C_1, C_2, \ldots, C_w\}$. For each chain C_i, consider the maximum element a_i such that each a_i is part of some antichain of size w; call A the set of all such a_i.

Claim: A is an antichain. Towards contradiction, assume that A is not an antichain. Then there exists a pair $a_i, a_j \in A$ such that $a_j < a_i$. We chose a_i from an antichain A_i of size w. This antichain A_i, since it is of size w, must intersect C_j. Consider $a' \in A_i \bigcap C_j$. Since we chose a_j as the maximal element in C_j such that a_j is in an antichain of size w, we know that $a' \leq a_j$. By transitivity, we then know that $a' < a_i$. This is a contradiction since they are from the same antichain A_i.

Now consider adding a new element x to the poset. If $x \leq a$ for each $a \in A$, then x must be in its own chain. Then the width and the number of chains is now $w+1$, which is consistent with the theorem. Otherwise, there exists an a_i such that $a_i < x$ (which means x can go in that chain). Consider $C = x \bigcup \{e \in C_i | e \leq a_i\}$. We know that C is a chain. Consider $Q = P - C$. We know that Q has no antichain of size w, since a_i was the maximal element in the chain which could be in such an antichain, and we also removed all smaller elements from the chain. We know that $A - a_i$ is an antichain of size $w-1$. Since Q is smaller than P, the inductive hypothesis tells us that Q can be covered by $w-1$ chains. To get P, add in the chain C, resulting in a total chain cover size and width of w. ∎

3.2.2 Application of Dilworth's Theorem

Perfect matching problem. Suppose there are n girls (G) and n boys (B). Every girl has a set of boys that she wants to marry. The goal is find a perfect matching between these boys and girls (every girl will match to a boy) (Figure 3.2).

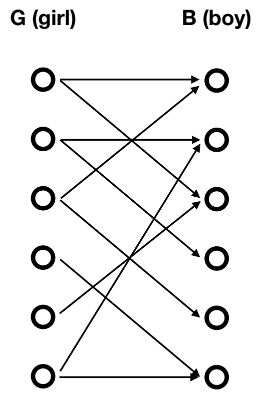

Figure 3.2: A bipartite graph $(G + B, E)$.

To state the problem more formally: (**Hall's Theorem**)
There is a finite bipartite graph $(G + B, E)$ with bipartite sets G and B, where $|G| = |B|$. I is a subset of vertices in G: $I \subseteq G$. $N(I)$ is the subset of vertices in B ($N(I) \subseteq B$) such that every vertice has a matching with the vertices in I: $N(I)$ is defined as $\{b \in B | \exists g \in I; (g, b) \in E\}$.

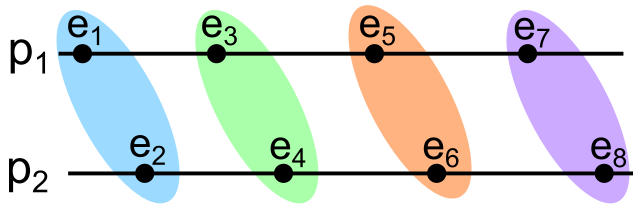

Figure 3.3: Alice's greedy approach: assume each $e_i < e_i + 1$, and Bob gives each e_i sequentially. If Alice is greedy, when e_2 arrives, she may greedily place it in a chain with the previous comparable event, e_1. She may do the same with e_3 and e_4. As more events are included, Alice keeps adding new chains. However, we know this poset can be represented by a vector clock of size 2, since there are only 2 processes.

Theorem 3.2 *Perfect matching in a bipartite graph is possible if and only if $\forall I \subseteq G$, such that $|N(I)| \geq |I|$.*

The necessary part is easy: If the number of vertices in $N(I)$ is smaller than that of I, some vertices in I will not find a match. The sufficient part needs Dilworth's Theorem. Consider the bipartite graph as a poset. If the bipartite can be covered by n chains, then it achieves perfect matching. It is sufficient to show that there is no antichain of size larger than n.

Proof: Case 1 ($|N(I) < |I|$):
If for some I such that $|N(I) < |I|$, clearly there is no perfect matching.
Case 2 ($|N(I) \geq |I|$):
Pick any antichain A. Define I as the vertices both in G and A: $I = A \cap G$. (All the girls are in the antichain.) The number of vertices in A is $|A|$. Since A is an antichain, it does not include the vertices in the chains: $|A| \leq |I| + |B| - |N(I)|$. Since $|N(I) \geq |I|$, $|A| \leq |B| + (|I| - |N(I)|) \leq |B| = n$. Therefore, the number of antichains is no more than n and the poset can be decomposed into at least n chains.

■

3.3 Chain Clock

Recall the vector clocks from the previous class session. We know that the size of the vector required for the vector clock scheme is equivalent to the size of the chain vector for the poset constructed by the scheme. Dilworth's Theorem gives us a tight lower and upper bound on the size of the chain cover of a poset P. However, actually *using* the theorem requires knowledge about the width w of the poset. In other words, the entire poset (i.e., the entire future of events for the vector clock) must be known beforehand to know what the chain cover size will be. We consider this an "offline" solution for our poset chain cover problem. In contrast, we would like an "online" solution, so that we have a small size vector for our vector clock scheme as new events are coming to the system (i.e., we don't know how wide the poset will become).

To approach an online solution, consider the following game: There are two players, Alice and Bob. Bob presents events to Alice. Alice chooses the chain in which the events are placed. Bob wants to maximize the number of chains; Alice wants to minimize the number of chains.

First, let's consider a naive, greedy approach for Alice. In this approach, Alice will place a new event in whichever chain was most recently chosen. We can see the issue with this approach in Figure 3.3. In a system with only two processes, we know can be handled by a vector of size two (the # of processes) using Matter's and Fidge's vector clock algorithm. However, Alice's naive solution in this case may use (asymptotically) as many chains as there are events in the system if Bob gives events whose processes alternate each time.

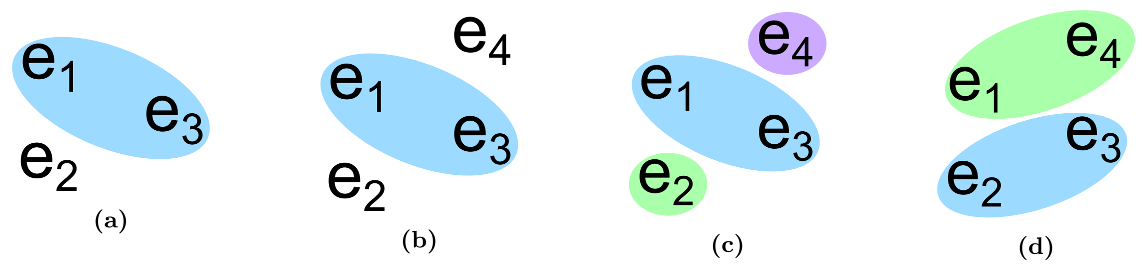

Figure 3.4: Consider the sequence of events given by Bob: (e_1, e_2, e_3, e_4). If $e_1, e_2 < e_3$, then Alice must choose between e_1 and e_2 to form a chain with e_3, as seen in **(a)**. Depending on Alice's choice, Bob can give e_4 such that e_4 requires a new chain; for example, in **(b)**, $e_2 || e_4$, and $e_1 < e_4$, but $e_3 || e_4$. So e_4 cannot be in the chain with e_1, e_3, nor with e_2, so it must use its own chain, shown in **(c)**. If Alice had chosen a different chain than shown in **(a)**, then she could have given a 2-chain cover instead, shown in **(d)**. This type of "forcing" move by Bob can be extrapolated to show that Bob has a strategy forcing Alice to use $\binom{k+1}{2}$ chains.

A better strategy for Alice is: given an event e_i from process i, place e_i into a chain in which the latest event in that chain is also from process i. If this is not possible, make a new chain.

Exercise. Show that this new strategy for Alice guarantees that no more than n chains will be required, where n is the total number of processes.

If you solved the exercise above, then we know that Alice won't do *worse* than Matter's and Fidge's vector clock algorithm from last class. However, Dilworth's tells us that we can do better in cases where the width w of the generated poset is less than n, the number of processes. In systems with many users (such as an internet website), since each user analogizes to a process, the width of the antichain may be much smaller than the number of total processes. So, we'd like a strategy for Alice's game which ensures that we use exactly w chains. However, the following lower bound by Szemérdi tells us that this is not possible:

Claim: If k is the width of the poset, then Bob has a strategy to force Alice to use $\binom{k+1}{2} = \Theta(k^2)$ chains. For the strategy, see Figure 3.4. Assume Alice has two events a and b which are not in any chain. On the third event c, Alice must choose to place c in a chain with either a or b. Without loss of generality, if Alice chooses to place c in a chain with a, Bob can use a new event $d \geq a$ to force a new chain. We skip the details, but you can extrapolate on this strategy to see that the number of chains which Bob can force Alice to use is $\binom{k+1}{2}$.

However, Felsner considered this problem under the following constraint: assume that Bob must give events in an order consistent with the poset order. I.e., if $e \to f$, then Bob must give e to Alice before he gives f. Felsner showed that this lower bound still holds under this constraint (Bob can still force Alice to use $\binom{k+1}{2}$ chains), but also showed the following:

Claim: There exists an algorithm for Alice s.t. she never uses more than $\binom{k+1}{2}$ chains, under the constraint that Bob gives the events in an order consistent with the poset order. Figure 3.5 outlines the data structure used by the algorithm. The following invariant is maintained: in each B_i, the heads (top) of nonempty chains are incomparable. When Bob gives Alice an event x, starting with $i = 1$:

- If the head of any chain in B_i is NOT less than x, move to B_{i+1}.
- If the head of any chain in B_i IS less than x, insert x into that chain.

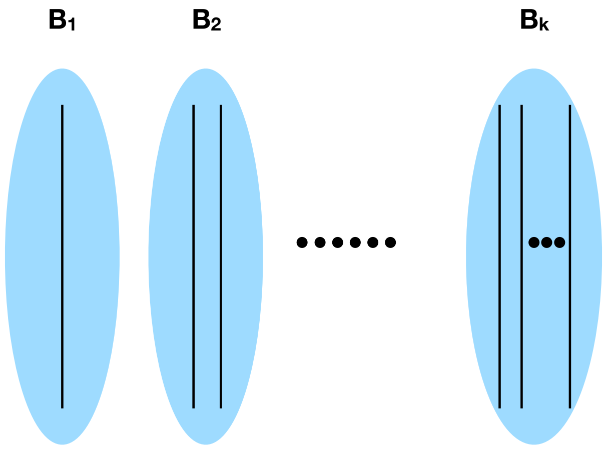

Figure 3.5: The data structure maintained by Alice. Each B_i contains i chains.

- When x is inserted into a chain in B_i, replace the chains in B_i which do not contain x with the chains in B_{i-1}.

Since we already compared the heads of chains in B_{i-1} with x, we know that they are incomparable with x, and thus maintain the invariant (each pair of heads of the new B_i are incomparable). The number of chains is $1+2+3\cdots+k = \binom{k+1}{2}$.

3.3.1 Open question

The above theorem applies when the events come in an order consistent with poset order. If the constraint is removed and we allow events come in any order, current research shows that the number of chains is lower bound by $O(k^2)$ and upper bound by $(5^k - 1)/4$. There is a large gap between these bounds. It is still an open question if the bounds can be made tighter.

3.4 Verification of Vector Clock

See the proofs in Chapter 4 in the textbook: Elements of Distributed Computing by Vijay K. Garg, 2002.

References

[Gal94] F. Galvin, A proof of Dilworth's chain decomposition theorem, *The American Mathematical Monthly* **101** (1994), pp. 352–353.