CS5811 Simple Temporal Network Example

Consider the following events with associated time intervals:

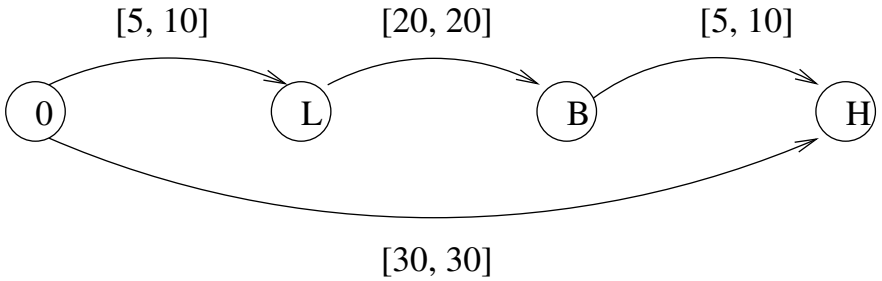
1. I was in Houghton at 8:30.
2. I left home between 8:05 and 8:10.
3. It takes me 20 minutes to drive to the bridge.
4. I waited 5-10 minutes at the bridge.

Given the above constraints, the only scenario that is possible is that I left at 8:05 and waited only 5 minutes at the bridge. In order to implement an algorithm that performs a similar type of reasoning, we first represent in problem as a quantitative constraint graph.

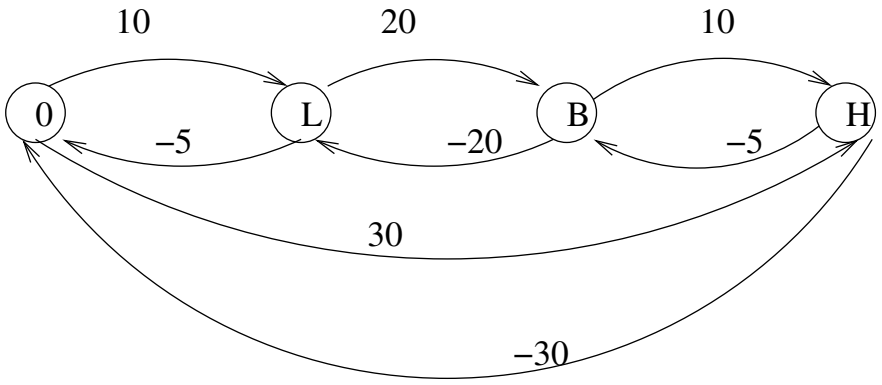
The following is the constraint graph representing the above problem. In this graph, the nodes correspond to events and the arcs show the temporal constraints between two events. The nodes in the example refer to the following:

"0" time zero (8:00 in this example)
"L" time of leaving home.
"B" time of arriving at the bridge.
"H" time of arriving in Houghton.

The first number in each bracket is the lower bound, and the second number is the upper bound for the duration of the event. For example, it took between 5 to 10 minutes after 8:00 to leave home.



We then convert the constraint graph into a distance graph. To understand this conversion, consider a constraint $[a, b]$ betweeen two nodes x_1 and x_2. The constraint specifies that $a \leq x_2 - x_1 \leq b$. Split this into two parts as $a \leq x_2 - x_1$ and $x_2 - x_1 \leq b$. Multiply the first part by -1 to get: $x_1 - x_2 \leq -a$. Using these, we create a graph where an arc with label n from x to y means $x - y \leq n$. In a distance graph, the semantics of the nodes remain the same. Each constraint arc from a to b is replaced with two arcs. The first arc goes from a to b and is labeled with the upper bound. The second arc goes from b to a and is labeled with the negative of the lower bound. This way, when we are going from "left to right" we are adding the upper bonds; and when we are going form "right to left" we are adding the lower bounds.



The above graph represented as a constraint matrix is as follows. We are using "99" as a large number to represent ∞, i.e., no constraints.

		0	1	2	3
		0	L	B	H
0	0	0	10	99	30
1	L	-5	0	20	99
2	B	99	-20	0	10
3	H	-30	99	-5	0

The constraint graph after running Floyd-Warshall's algorithm is:

		0	1	2	3
		0	L	B	H
0	0	0	**5**	25	30
1	L	-5	0	20	25
2	B	-25	-20	0	**5**
3	H	-30	-25	-5	0

Note that the 10s at $(0, L)$ and (B, H) changed to 5s. Also, $(0, 2), (1, 3), (2, 0)$, and $(3, 1)$ changed from 99.

Example shortest path: From $0(0)$ to $1(L)$ the direct link is 10. If we go from 0 to H directly, and then backwards to L, the distance is $30 - 5 - 20 = 5$. This means that the maximum time between 0 and H is 30, but two events between L and H take a minimum of 25. Therefore, the maximum time between 0 and L is 5. A similar computation can be done for the distance from $2(B)$ to $3(H)$.