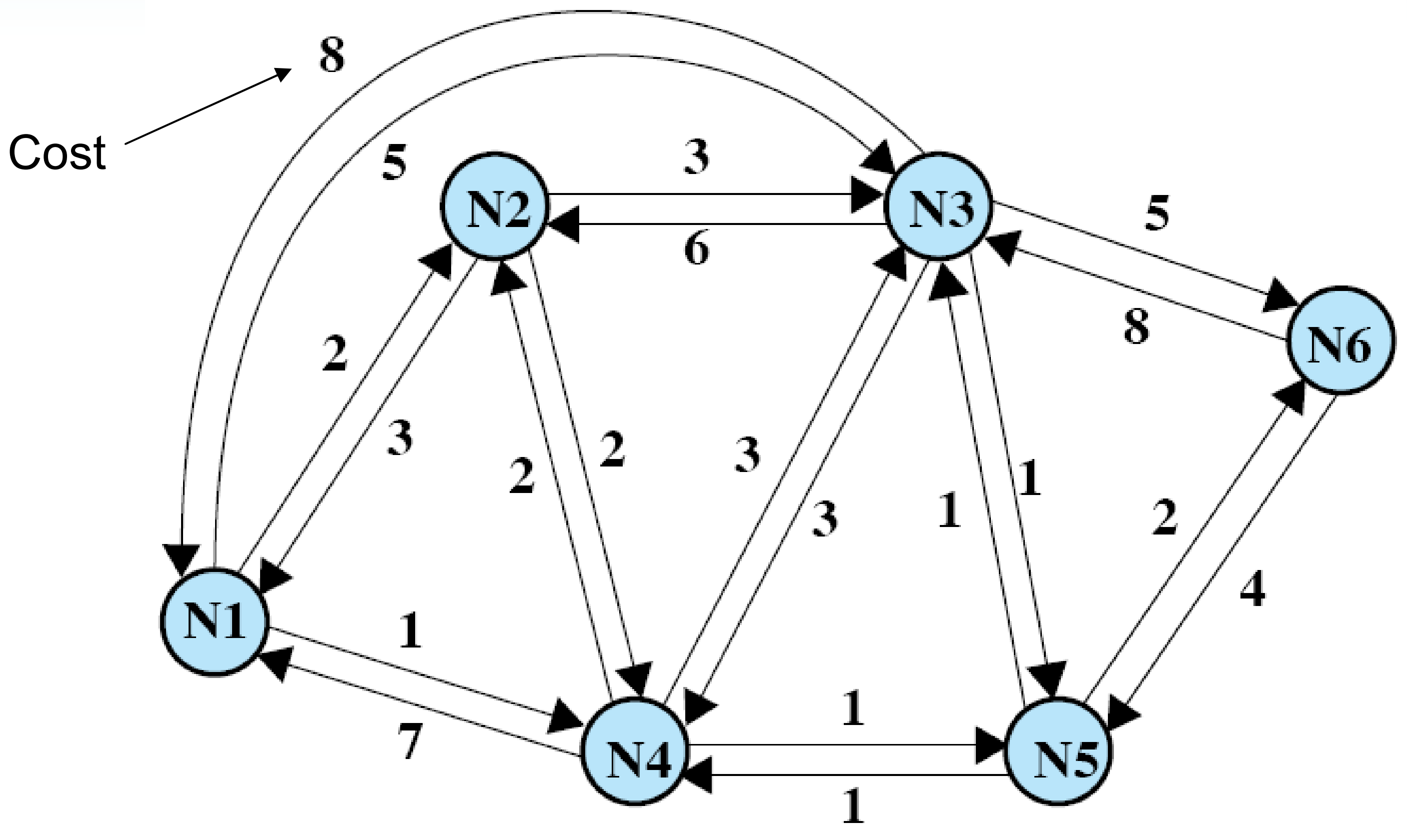
Routing (Extra Material)

Dr Steve Gordon

ICT, SIIT

Example Network Configuration



Link: a direct connection between two nodes (such as via cable or wireless). E.g. link fro N1 to N2
Path (or route): a track or way between two nodes, via one or more links. E.g. a path from N1 to N6 is N1 – N2 – N3 – N6
Hop: traverse a link. E.g. there are 3 hops between N1 and N6 on path N1 – N2 – N3 – N6

Dijkstra’s Algorithm Example

Iter.	T	L(2)	Path	L(3)	Path	L(4)	Path	L(5)	Path	L(6)	Path
1	{1}	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	{1,4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
3	{1, 2, 4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
4	{1, 2, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
5	{1, 2, 3, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
6	{1, 2, 3, 4, 5, 6}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

This table is an alternative way to view the algorithm. In the first iteration, the source (N1) only knows of the cost to its direct neighbours (N2, N3 and N4). The least-cost, as well as the path with least-cost, are listed for each of the five other nodes (note the cost is ∞ for N5 and N6).

The algorithm then selects the node with least-cost (N4, which has cost of 1) and adds it to T. Then the second iteration is executed, where the least-cost path from N1 to each other node is updated. But know we also know the cost from N4 to its neighbours (including N5). So after the second iteration, we have a least-cost and path for N5 (the path is 1 to 4 to 5, which has a cost of 2).

Now we add N2 to T (or it could have been N5, since both have a least-cost of 2), and proceed with iteration 3.

The algorithm finishes when T contains all nodes: we have the least-cost paths from N1 to all other nodes.

Least Cost Paths

	1	2	3	4	5	6
1	-	1-2 (2)	1-4-5-3 (3)	1-4 (1)	1-4-5 (2)	1-4-5-6 (4)
2	2-1 (2)	-	2-3	2-4	2-4-5	2-4-5-6
3	3-5-4-2-1	3-5-4-2	-	3-5-4	3-5	3-5-6
4	4-2-1	4-2	4-5-3	-		
5					-	
6						-

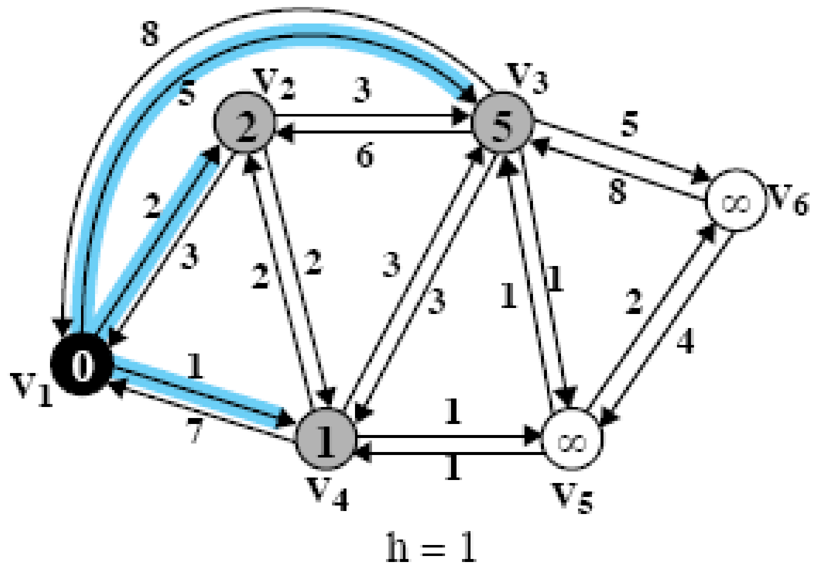
Bellman-Ford Algorithm

- Overview:
  - Find shortest paths from given node, subject to the constraint that paths contain at most one link
  - Then find the shortest paths with a constraint of paths of at most two links
  - And so on …
- Define:
  - s = source node
  - $w(i, j)$ = link cost from node i to node j
  - $w(i, i) = 0$
  - $w(i, j) = \infty$ if the two nodes are not directly connected
  - $w(i, j) \geq 0$ if the two nodes are directly connected
  - h = maximum number of links in path at current stage of the algorithm
  - $L_h(n)$ = cost of least-cost path from s to n under constraint of no more than h links

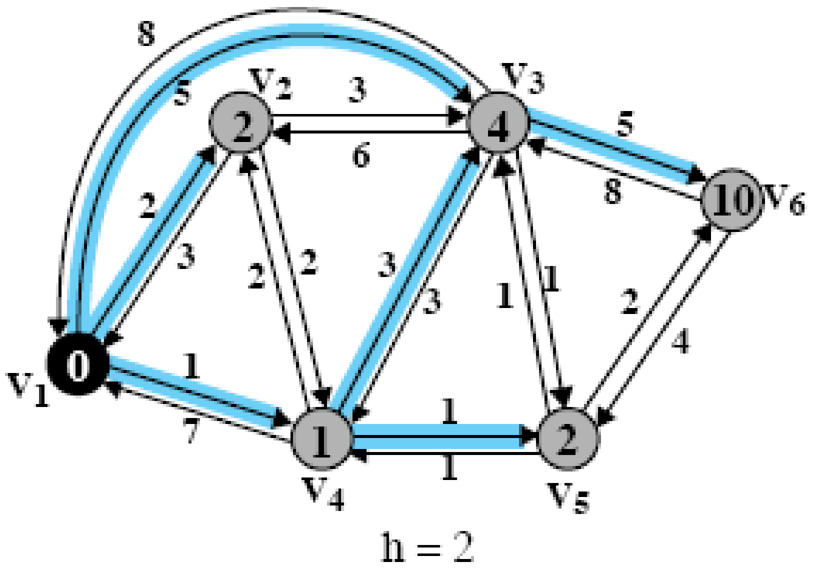
Bellman-Ford Algorithm

- Step 1 [Initialization]
  - $L_0(n) = \infty$, for all $n \neq s$
  - $L_h(s) = 0$, for all h
- Step 2 [Update]
  - For each successive $h \geq 0$
    - For each $n \neq s$, compute: $L_{h+1}(n)=\min_j[L_h(j)+w(j,n)]$
  - Connect *n* with predecessor node *j* that gives minimum
  - Eliminate any connection of *n* with different predecessor node formed during an earlier iteration
  - Path from *s* to *n* terminates with link from *j* to *n*

Bellman-Ford Algorithm Example



Step 1

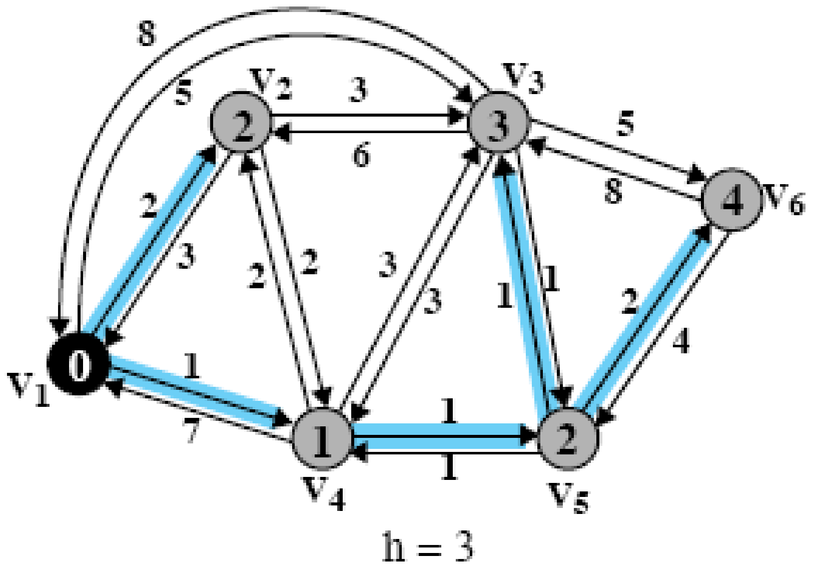


Step 2

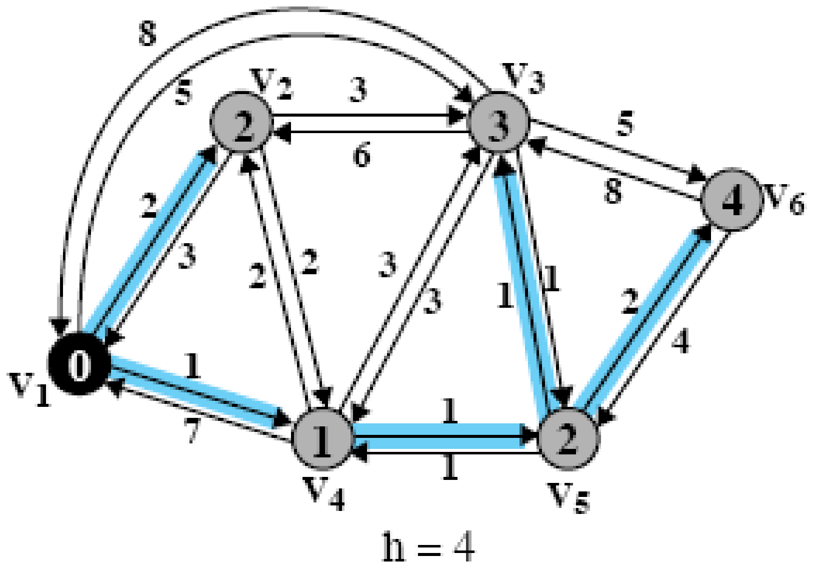
For all the paths that are 1 hop, find the shortest path to other nodes. In this example, only nodes V2, V3 and V4 can be reached in 1 hop.

For all the paths that are 2 hops, find the shortest path to other nodes.

Bellman-Ford Algorithm Example



Step 3



Step 4

Bellman-Ford Algorithm Example

h	$L_h(2)$	Path	$L_h(3)$	Path	$L_h(4)$	Path	$L_h(5)$	Path	$L_h(6)$	Path
0	∞	-	∞	-	∞	-	∞	-	∞	-
1	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	2	1-2	4	1-4-3	1	1-4	2	1-4-5	10	1-3-6
3	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
4	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

Comparison

- Results from two algorithms agree
  - They both find the same least-cost paths
- Bellman-Ford Algorithm
  - Calculation for node *n* needs link cost to neighbouring nodes plus total cost to each neighbour from *s*
  - Each node can maintain set of costs and paths for every other node
  - Can exchange information with direct neighbours
  - Can update costs and paths based on information from neighbours and knowledge of link costs
- Dijkstra’s Algorithm
  - Each node needs complete topology
  - Must know link costs of all links in network
  - Must exchange information with all other nodes
- Evaluation depends on processing time of algorithms, amount of information they exchange, and implementation details
  - We will see later how the algorithms are used in the Internet