Min-step Path Algorithm

Let G = (V,E) be a digraph (or graph) with n vertices. Suppose that we seek to determine a *minimum length* path from node u to node v, or determine if no path exists. This approach is basically Dijkstra's algorithm — it uses an enhanced version of BFS. Define the following sequence of subsets, $P_k \square V$:

$P_0 = \{u\}$, and

$P_{k+1} = P_k \square \{y \mid x \square P_k \text{ and } (x,y) \square E\}, k \geq 0.$

Formally, we have the

Assertion: P_k is the set of all nodes reachable from u by a path of length k or less.

P_k is clearly a "telescoping" sequence of sets, $P_0 \square P_1 \square P_2 \square \ldots \square V$. Also, if $P_{k+1} = P_k$, then $P_{k+2} = P_k$, $P_{k+3} = P_k$, and all the subsequent sets are the same. Since P_0 has one element, P_1 has at least two elements if it differs from P_0, P_2 at least three if it differs from P_1, etc., and these sets can increase for at most n–1 steps since V has size n. Hence $P_{k+1} = P_k$ for some $k \leq n–1$, say k = M. So the algorithm is to compute the sets P_0, P_1, P_2, … until the *first* P_k where $v \square P_k$, or until k = M. Then if $v \square P_k – P_{k-1}$, k is the length of the shortest path; P_M is all nodes reachable from u so if P_M does not include v, then v is unreachable from u.

Notice that so far this process determines the *length* of a shortest path, not the path itself. But once the sequence of sets leading to v is determined, the path itself can be extracted. In particular, $P_k – P_{k-1}$ is the set of nodes with a shortest path of length k. So if v first appears in P_k, there must be node $x_{k-1} \square P_{k-1} – P_{k-2}$ with edge (x_{k-1},v) (i.e., a path shorter than k–1 to a node with an edge to v gives a path shorter than k to v). Similarly, since x_{k-1} first occurs in P_{k-1}, there must be a node $x_{k-2} \square P_{k-2} – P_{k-3}$ with an edge (x_{k-2},x_{k-1}), etc., and eventually $x_1 \square P_1 – P_0$ with edge (u,x_1). Then the desired path is u, x_1, x_2, … , x_{k-1}, v.

Actually this computation provides more than a shortest path algorithm. Suppose that G is a *graph* and we wish to determine if it is connected. One approach would be to consider every pair of nodes and decide whether or not there is a (shortest) path using the algorithm developed above. But this is far more effort than necessary since we would check for a (shortest) path between each of the n(n-1) pairs of distinct nodes. Actually we only need to do the computation above for some one node, and check if all nodes appear in the reachability set.

Corollary: graph G = (V,E) is connected if and only if $P_M = V$, where the starting node is arbitrary.