Global Processing via Graph-Theoretic Techniques

- The previous method for edge-linking discussed is based on obtaining a set of edge points through a gradient operation.
- As the gradient is a derivative, the operation is seldom suitable as a preprocessing step in situations characterized by high noise content.
- Here, we discuss a global approach based on representing edge segments in the form of a graph and searching the graph for low-cost paths that correspond to significant edges.
  - This representation provides a rugged approach that performs well in the presence of noise.
  - As might be expected, the procedure is considerably more complicated and requires more processing time than the methods discussed so far.

Basic Graph Theory

- We begin the development with some basic definitions.
- A graph $G = (N, A)$ is a finite, nonempty set of nodes $N$, together with a set $A$ of unordered pairs of distinct elements of $N$.
- Each pair $(n_i, n_j)$ of $A$ is called an arc.
- A graph in which the arcs are directed is called a directed graph.
- If an arc is directed from node $n_i$ to node $n_j$, then $n_j$ is said to be a successor of its parent node $n_i$.
- The process of identifying the successors of a node is called expansion of the node.
Basic Graph Theory (continued)

- The process of identifying the successors of a node is called expansion of the node.
- In each graph we define levels, such that level 0 consists of a single node, called the start node, and the nodes in the last level are called goal nodes.
- A cost, \( c(n_i, n_j) \), can be associated with every arc \( (n_i, n_j) \).
- A sequence of nodes \( n_1, n_2, \ldots, n_k \) with each node \( n_i \) being a successor of node \( n_{i-1} \), is called a path from \( n_1 \) to \( n_k \), and the cost of the path is

\[
c = \sum_{i=2}^{k} c(n_{i-1}, n_i)
\]

Edge Elements

- An edge element is the boundary between two pixels \( p \) and \( q \), such that \( p \) and \( q \) are 4-neighbors.

- In this context, an edge is a sequence of edge elements.
Edge Element Costs

- Each edge element defined by pixels p and q has an associated cost, defined as
  \[ c(p, q) = H - [f(p) - f(q)] \]
- where H is the highest intensity value in the image (7 in this case), f(p) is the intensity value of p, and f(q) is the intensity value of q. As indicated earlier, p and q are 4-neighbors.

Edge Element Cost Graph

- Each node corresponds to an edge element, and an arc exists between two nodes if the two corresponding edge elements taken in succession can be part of an edge.
- The cost of each edge element, computed using the cost equation, is the arc leading into it, and goal nodes are shown as blue rectangles.
- Each path between the start node and a goal node is a possible edge.

Edge Element Cost Graph (continued)

- For simplicity, the edge is assumed to start in the top row and terminate in the last row, so that the first element of an edge can be only
  - \([0,0], (0,1)\) or \([0,1], (0,2)\]
  - and the last element
  - \([2,0], (2,1)\) or \([2,1], (2,2)\)
- The dashed lines represent the minimum-cost path.
- The corresponding edge is

\[
\begin{align*}
0 & \quad 1 & \quad 2 \\
(0) & \quad (1) & \quad (2) \\
\end{align*}
\]
Approximating a Minimum Cost Path

- In general, the problem of finding a minimum-cost path is not trivial in terms of computation.
- Typically, the approach is to sacrifice optimality for the sake of speed, and the following algorithm represents a class of procedures that use heuristics in order to reduce the search effort.
  - Let \( r(n) \) be an estimate of the cost of a minimum-cost path from the start node \( s \) to a goal node, where the path is constrained to go through \( n \).
  - This cost can be expressed as the estimate of the cost of a minimum-cost path from \( s \) to \( n \) plus an estimate of the cost of that path from \( n \) to a goal node; that is,
    \[
    r(n) = g(n) + h(n)
    \]

- Here, \( g(n) \) can be chosen as the lowest cost path from \( s \) to \( n \) found so far, and \( h(n) \) is obtained by using any available heuristic information (such as expanding only certain nodes based on previous costs in getting to that node).
- An algorithm that uses \( r(n) \) as the basis for performing a graph search is as follows.

**Step 1:** Mark the start node OPEN and set \( g(s) = 0 \).

**Step 2:** If no node is OPEN exit with failure; otherwise, continue.

**Step 3:** Mark CLOSED the OPEN node \( n \) whose estimate \( r(n) \) computed from \( r(n) = g(n) + h(n) \) is smallest. (Ties for minimum \( r \) values are resolved arbitrarily, but always in favor of a goal node.)

**Step 4:** If \( n \) is a goal node, exit with the solution path obtained by tracing back through the pointers; otherwise, continue.

**Step 5:** Expand node \( n \), generating all of its successors. (If there are no successors go to step 2.)

**Step 6:** If a successor \( n_i \) is not marked, set \( r(n_i) = g(n) + c(n, n_i) \) mark it OPEN, and direct pointers from it back to \( n \).

**Step 7:** If a successor \( n_i \) is marked CLOSED or OPEN, update its value by letting \( g'(n_i) = \min(g(n_i), g(n) + c(n, n_i)) \). Mark OPEN those CLOSED successors whose \( g' \) values were thus lowered and redirect to \( n \) the pointers from all nodes whose \( g' \) values were lowered. Go to step 2.

**Step 8:** If no heuristic information is available (that is, \( h = 0 \)), the procedure reduces to the uniform-cost algorithm of Dijkstra [1959].

In general, this algorithm does not guarantee a minimum-cost path; its advantage is speed via the use of heuristics.

However, if \( h(n) \) is a lower bound on the cost of the minimal-cost path from node \( n \) to a goal node, the procedure indeed yields an optimal path to a goal (Hart, Nilsson, and Raphael [1968]).