Routing Algorithms for Interval, Circular-Arc, and Permutation Graphs

Feodor Dragan and Irina Lomonosov

Department of Computer Science
Kent State University
Ohio, USA

Routing Problem

Routing is one of the basic tasks that a distributed network of processors must be able to perform.

A routing scheme is a mechanism that can deliver packets of information (messages) from any node of the network to any other node of the network.

A network can be viewed as a graph, with the nodes representing processors and the edges representing direct connections between processors.
Routing is Distributed Algorithm.

Each processor in the network upon receiving a message has to decide whether the message has already reached its destination, and if not, how to forward it towards the destination.

Measures of Goodness of Routing Scheme

1. **Optimality** – we want to route messages along shortest or near shortest paths.
2. **Compactness** – what memory requirements are to store the necessary routing information at each node.
3. **Efficiency** – how long it takes for a node to decide where to forward a message.

Three Approaches

1. **Complete routing table.** In each node $x$ we specify for each destination $y$ the first edge along some shortest path from $x$ to $y$. (Optimal, efficient, but not compact) – $(0, \Theta(1), n \log d)$. $d$ is a degree of the node $x$.

2. **Interval routing.**
   Edges are labeled with subintervals of $[1..n]$.
   A message is forwarded on the edge labeled with an interval containing the destination label. $(\Omega, O(log d), O(d \log n))$

3. **Labeling schemes** assign two labels per node the *address* and the *local routing table*. Based on these labels and the address of the destination the node decides to which neighbor to forward the message.
Lower Bounds

For every optimal routing strategy and for any $\Delta$ there is a graph of maximum degree $\Delta$ which requires $\Omega(n \log \Delta)$ bit routing tables on $\Theta(n)$ nodes [Gavoille/Perennes’96]. This matches the memory requirements for complete routing tables.

- One way to go is to abandon optimality and route messages along the paths that are ‘close’ to shortest (multiplicative $(\text{dist}(x,y)/\text{dist}(x,y) \leq t)$ or additive $(\text{dist}(x,y) - \text{dist}(x,y) \leq s)$ stretched routing schemes).

Any multiplicative $t$ stretched routing scheme must use
\[
\Omega(\sqrt{n}) \quad \text{bits} \quad \text{for } t < 5 \quad [\text{Thorup/Zwick’01}]
\]
\[
\Omega(n) \quad \text{bits} \quad \text{for } t < 3 \quad [\text{Gavoille/Gengler’01}]
\]
\[
\Omega(n \log n) \quad \text{bits} \quad \text{for } t < 1.4 \quad [\text{Gavoille/Perennes’96}]
\]

- Another way is to consider special classes of graphs.

Previous Results

<table>
<thead>
<tr>
<th>Class</th>
<th>Compactness</th>
<th>Error</th>
<th>Efficiency</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>mesh, hypercube, complete, bipartite, unit interval, interval, tree, ring, unit circular-arc, outerplanar</td>
<td>$O(\Delta \log n)$</td>
<td>0</td>
<td>$O(\log \Delta)$</td>
<td>Fraigniaud/Gavoille’98 Frederickson/Janardan’88 Leeuwen/Tan’86-87 Narayanan/Shende’98 Santoro/Khatib’85</td>
</tr>
</tbody>
</table>

These results are obtained using interval routing scheme. The time to route a message from $x$ to $y$ is $O(\text{dist}(x,y) \times \log \Delta)$
### Previous Results (cont)

<table>
<thead>
<tr>
<th>Class</th>
<th>Compactness</th>
<th>Error</th>
<th>Efficiency</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree</td>
<td>((1+o(1))\log n)</td>
<td>0</td>
<td>(O(1))</td>
<td>Thorup/Zwick’01</td>
</tr>
<tr>
<td>chordal</td>
<td>(O(\log^3 n/\log \log n))</td>
<td>+2</td>
<td>(O(1))</td>
<td>Dourisboure/</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gavoille’02</td>
</tr>
</tbody>
</table>

### Our Results

<table>
<thead>
<tr>
<th>Class</th>
<th>Compactness</th>
<th>Error</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval, circular-arc</td>
<td>(O(d(v) \log n))</td>
<td>0</td>
<td>(O(d) + \log n) (total)</td>
</tr>
<tr>
<td></td>
<td>(O(\Delta \log n))</td>
<td>0</td>
<td>(O(1))</td>
</tr>
<tr>
<td></td>
<td>(O(d(v) \log n))</td>
<td>+1</td>
<td>(O(1))</td>
</tr>
<tr>
<td>proper interval,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>proper circular-arc</td>
<td>(O(\log n))</td>
<td>0</td>
<td>(O(1))</td>
</tr>
<tr>
<td>permutation</td>
<td>(O(\Delta \log n))</td>
<td>+1</td>
<td>(O(1))</td>
</tr>
<tr>
<td>bipartite permutation</td>
<td>(O(\Delta \log n))</td>
<td>0</td>
<td>(O(1))</td>
</tr>
</tbody>
</table>
Our Contribution

1. Our routing scheme for interval graphs is simpler than existing before
easy to extend to circular-arc graphs
more efficient: $O(\text{dist}(x,y) + \log \Lambda)$ vs $O(\text{dist}(x,y)\times \log \Lambda)$

2. We present an optimal scheme for circular-arc graphs. Such scheme was known only for unit circular-arc graphs. No interval routing scheme with one interval per edge exists for circular-arc graphs [Fraigniaud/Gavoille'98].

3. This is the first routing scheme presented for permutation graphs.

4. As a byproduct, we obtain optimal, efficient $O(1)$, and compact $O(\log n)$ scheme for proper circular-arc.

Method

(we describe only one method for interval graphs)

We assign numbers from 1 to $2n$ to endpoints of the line segments. Each node $v$ is represented by a pair of integers $l(v)$ and $r(v)$.

For each node $v$ we define the right maximum neighbor $x(v)$ as an adjacent to $v$ node with the largest right endpoint.

In a similar way we define the left maximum neighbor $y(v)$. 
Method (cont)

For each node $v$ we construct a *list of gates* $Gate_v$ to its neighbors as follows.

We assign a gate number to each point in the open interval $(l(v), r(v))$. These points are right and/or left endpoints of the segments representing the neighbors of $v$.

We say that a node $v$ is *bad* if its segments is a subsegment of another segment. These means that $Gate_v$ does not contain gates to all neighbors.

For each bad node we additionally keep a sorted *list of neighbors* $N(v)$ along with their *gate numbers*.

The size of the local routing table is $O(d(v) \log n)$.

Method (cont)

A pair of integers $\{l(u), r(u)\}$ is considered to be an *address* of a node $u$.

Routing a message from a source node to a destination node $u$ can be done as follows.

If a node $v$ initiates a message ($v$ is a *source node*) then *Message Initiating Algorithm* is used.

If $v$ is an *intermediate node* on a path between a source and a destination then *Message Passing Algorithm* is executed at each node $v$ after receiving a message.
Message Passing Algorithm

**Input:** A current node $v$ and destination node $u$.

**Output:** A neighbor to send the message to.

if $(r(v) < l(u))$ then send the message to $x(v)$ and stop.
if $(r(u) < l(v))$ send the message to $y(v)$ and stop.
if $(l(v) < r(u) < r(v))$ then send the message through $\text{Gate}_v(r(u) - l(v))$ and stop.
if $(l(v) < l(u) < r(v))$ send the message through $\text{Gate}_v(l(u) - l(v))$ and stop.

The complexity is $O(1)$.

---

Message Initiating Algorithm 1

**Input:** A source node $v$ and destination node $u$.

**Output:** A neighbor to send the message to.

execute *Message Passing Algorithm*.

search a sorted list of neighbors $N(v)$ for a gate and send the message through this gate.

**Note:** Searching of $N(v)$ is involved ONLY if a message is initialized by a bad node and $\text{dist}(v, u) = 1$.
Extension to Circular-Arc Graphs

The routing in circular-arc graphs is more complicated problem then routing in interval graphs.

There are two ways to travel between a pair of points on a circle, and only one on a line.

Theorem  For each node $v$ of a circular-arc graph $G$ there exist a node $\varepsilon(v)$ such that for any arc $u \in N(v)$ the following holds. If $r(u) \in [r(v), r(\varepsilon(v))]$ then there exits a shortest path between $v$ and $u$ which goes counterclockwise. Otherwise, there exits a shortest path between $v$ and $u$ which goes clockwise.

We say that $\varepsilon(v)$ is antipodal to $v$.

Example: if $v=\gamma$ then $\varepsilon(\gamma)=g$.

Antipodal nodes for all nodes of $G$ can be found in $O(n^2)$ time.
Permutation Graphs

Our routing scheme for permutation graphs follows from the geometry of the permutation model.

Concluding Remarks

- We presented new routing schemes for interval, circular-arc, and permutation graphs.
- The routing schemes are compact, efficient and optimal for interval and circular-arc graphs and almost optimal for permutation graphs.
- For proper interval and proper circular-arc graphs, we obtained best possible routing schemes.
- In designing our routing schemes we essentially used the intersection models of those families of graphs.
Open Problems

1. Do those three families of graphs admit optimal routing schemes with $O(\log n)$ bit labels and $O(1)$ forwarding protocols? Or size of local routing tables has always to depend on degrees of nodes?

2. Can the presented routing scheme for permutation graphs be improved to route messages via shortest paths only?

3. Which other families of intersection graphs admit efficient routing schemes? One can consider trapezoid graphs, circle graphs, intersection graphs of discs on the plane, and others.