# IMPLICIT ROUTING AND SHORTEST PATH INFORMATION
(Extended Abstract)

Evangelos Kranakis[*][†]
(kranakis@scs.carleton.ca)

Danny Krizanc[*][†]
(krizanc@scs.carleton.ca)

Jorge Urrutia[‡][†]
(jorge@csi.uottawa.ca)

**Abstract**

We study the problem of constructing graphs from shortest path information (complete or partial). Consider graphs with labeled vertices and edges. Given a collection $V$ of vertices and for each $u \in V$ a positive integer $d(u)$, and a family $\mathcal{F}_u = \{F_{u,i} : i < d(u)\}$ of subsets of $V$ construct a graph such that for each $u$ and each link $i$ of $u$, $F_{u,i}$ is the set of nodes having an optimal length path to $u$ passing through link $i$. In the complete information case we show that a shortest path family uniquely determines the graph and conclude the existence of graphs such that any full information shortest path routing scheme requires a total of $\Omega(n^2)$ memory bits. We also study the class of "unique shortest path graphs", i.e. graphs for which all vertices are connected by a unique shortest path.

**1980 Mathematics Subject Classification:** 68Q99
**CR Categories:** C.2.1
**Key Words and Phrases:** Boolean routing, Feasible family, Implicit routing, Realizable family, Shortest paths, Shortest path family, Unique shortest path graps.
**Carleton University, School of Computer Science:** TR-95-08

[*]Carleton University, School of Computer Science, Ottawa, ON, K1S 5B6, Canada.

[‡]University of Ottawa, Department of Computer Science, Ottawa, ON, K1N 9B4, Canada.

[†]Research supported in part by NSERC (National Science and Engineering Research Council of Canada) grant.

## 1 Introduction

Routing messages in a network is one of the most fundamental tasks in distributed computation. A routing scheme consists of a distributed algorithm that directs traffic in the network. Given a cost function on the network links it is important to route messages along shortest paths. This can be accomplished most efficiently by storing at each node of the network a routing table.

In general, routing tables may require a lot of space. To overcome this problem several implicit routing schemes, which use structural information to reduce space, have been proposed. These include: Interval routing [12, 14], Compact path routing [1, 3], Prefix routing [2, 1], Boolean routing [4], etc.

It is clear that in general the total number of bits needed to store all the routing tables is $O(n^2 d)$, where $d$ is the maximal degree of the network.

For any graph $G = (V, E)$ construct the shortest path family $\mathcal{P}(G) = \{\mathcal{P}_u(G) : u \in V\}$ of $G$ as follows: for each link $e$ adjacent to $u$ let $P_{u,e}(G)$ be the set of vertices $v$ of $G$ for which there is a shortest path from $u$ to $v$ that uses link $e$. Now define $\mathcal{P}_u(G) = \{P_{u,i}(G) : i < \deg(u)\}$.

There have been several results in the literature in which it is desired to reconstruct a graph from a given information, like the degree sequence, etc. In this paper we also consider the problem of reconstructing the graph from its shortest path family and study the complexity of implicit routing.

### 1.1 Reconstruction problem

We are given a collection $V$ representing the set of vertices of a graph. In addition for each vertex $u \in V$ we are given a positive integer $d(u)$ (representing the degree of $u$) and a family $\mathcal{F}_u = \{F_{u,i} : i < d(u)\}$ of $d(u)$-nany sets. We call $\mathcal{F} = \{\mathcal{F}_u : u \in V\}$ a path family if the following conditions are met:

- for each $u \in V$, and each $i < d(u)$, $\emptyset \neq F_{u,i} \subseteq V \setminus \{u\}$,
- $\bigcup \mathcal{F}_u = V \setminus \{u\}$, for all $u \in V$.

We are interested in the problem of realizing such families as path families of graphs, i.e. such that $F_{u,i}$ is a nonempty subset of the set of vertices in $v \in V$ for which there is an optimal length path from $u$ to $v$ that uses link $i$.

We call $\mathcal{F}$ a feasible family if there is a graph $G$ such that for all $u \in V$ and each $i < d(u) = \deg_G(u)$, $F_{u,i} \subseteq P_{u,i}(G)$. In this case we say that the graph $G$ is feasible for the family $\mathcal{F}$. We call the feasible family $\mathcal{F}$ realizable by the graph $G$ if in addition for all $u \in V$, $\mathcal{F}_u = \mathcal{P}_u(G)$. In this case we say that the graph $G$ realizes the family $\mathcal{F}$.

### 1.2 Results of the paper

The questions we pose are the following.

**Question 1.1** *Let $\mathcal{F} = \{\mathcal{F}_u : u \in V\}$ be a path family. Can we reconstruct the graph from the shortest path information as given by the family $\mathcal{F}$? I.e.,*

1. *Can we determine if a family is feasible?*
2. *Can we determine if a family is realizable?*

*Moreover, in all cases above give an efficient algorithm.*

In this paper we give a polynomial time algorithm to test realizability, while feasibility is only known to be in $NP$. As a consequence we prove the existence of graphs such that any shortest path routing scheme requires a total of $\Omega(n^2)$ memory bits. We also introduce the class of "unique shortest path graphs".

## 2 Reconstruction from Shortest Paths

In this section we study the reconstruction problem from feasible as well as realizable shortest path families.

### 2.1 Feasible path families

Here is an example of a feasible family.

**Example 2.1** *Let the set of vertices be $V = \{1, 2, \ldots, n\}$ and the family*

$$\mathcal{F}_i = \{F_{i,j} = \{j\} : j \neq i, j < n-1\}.$$

*Then the family is realized by the complete graph $K_n$, i.e. $\mathcal{F} = \mathcal{F}(K_n)$.*

**Example 2.2** *Let the set of vertices be $V = \{1, 2, \ldots, 2n\}$ and the family*

$$\mathcal{F}_i = \{F_{i,j} = \{n+j\} \cup [1, n] \setminus \{i\} : 1 \leq j \leq n\}.$$

*Then the family is realized by the complete bipartite graph $K_{n,n}$, i.e. $\mathcal{F} = \mathcal{F}(K_{n,n})$.*

In general, a feasible family may not determine a unique graph. This was first observed by Peleg and Upfal [10]. In this case there may be more than one (up to isomorphism) graph which is feasible for the family.

**Example 2.3** *Let the set of vertices and the path family be the ones arising from the graphs depicted in Figure 2. Also assume that the graphs $H_3, H_4$ are not isomorphic. It is then easy to see that the resulting graphs have identical feasible path families but are not isomorpic.*

We can strengthen this construction in order to prove the following result.

**Theorem 2.1** *For each $n$, there exist $2^{\Omega(n)}$ non-isomorphic graphs with identical feasible shortest path families.*

Proof (Outline) We use an extension of the idea depicted in the figure of Example 2.3. We substitute recursively the graphs $H_i$ with non-isomorphic graphs having identical shortest path families, and use a simple counting argument. ■

**Example 2.4** *An example of a 3 connected graph is given in Figure 3. The feasible shortest path family is defined as follows:*

$$\begin{array}{ll} \mathcal{F}_a = \{\{b,2\},\{1,3\},\{c,d,4\}\} & \mathcal{F}_b = \{\{1,a\},\{2,4\},\{c,d,3\}\} \\ \mathcal{F}_c = \{\{4,d\},\{1,3\},\{a,b,2\}\} & \mathcal{F}_d = \{\{3,c\},\{2,4\},\{a,b,1\}\} \\ \mathcal{F}_1 = \{\{2,b\},\{a,d\},\{3,4,c\}\} & \mathcal{F}_2 = \{\{1,a\},\{b,c\},\{3,4,d\}\} \\ \mathcal{F}_3 = \{\{4,d\},\{b,c\},\{1,2,a\}\} & \mathcal{F}_4 = \{\{3,c\},\{a,d\},\{1,2,b\}\}. \end{array}$$

**Theorem 2.2** *Every graph $G$ can be embedded in graphs $H, H'$ such that the graphs $H, H'$ have identical feasible shortest path families.*

Proof (Outline) This is constructed by using the idea depicted in Figure 1. The graph $H$ is the sum $G + G$ of the graph $G$ with itself [8]. The graph $H'$ is the same as the graph $H$ except that take an edge $\{u, v\}$ of $G$ and delete the edges connecting: $u$ with its copy and $v$ with its copy. In addition, add the edges connecting $u$ with the copy of $v$ and $v$ with the copy of $u$. ■

Testing feasibility is clearly in $NP$. Nevertheless the following question still remains open.

**Question 2.1** *Is there a polynomial time algorithm to test if a given shortest path family is feasible?*

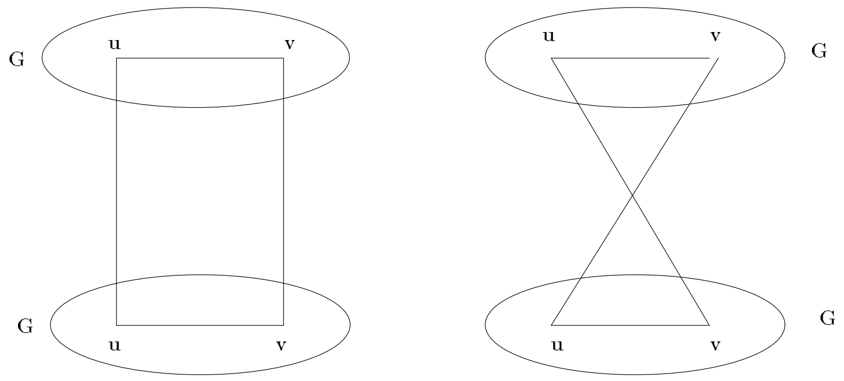

Figure 1: Two graphs with identical shortest path families. The graph $G+G$ and the graph resulting by swapping the vertices $u$ and $v$.

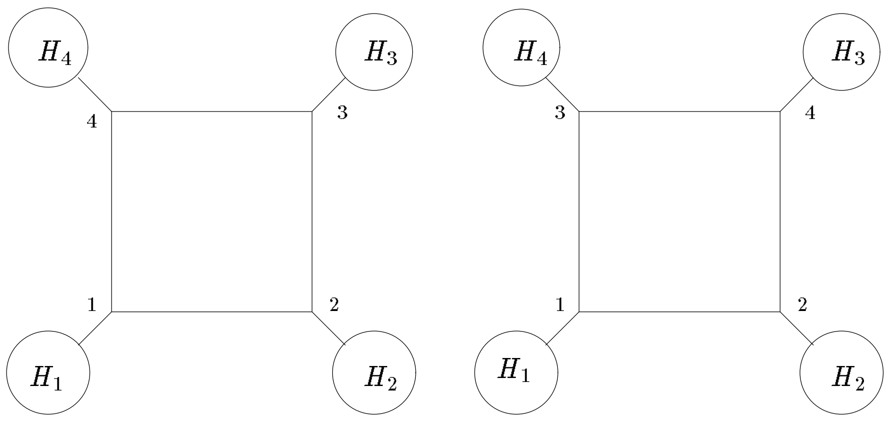

Figure 2: Non-isomorphic graphs with identical shortest path families.

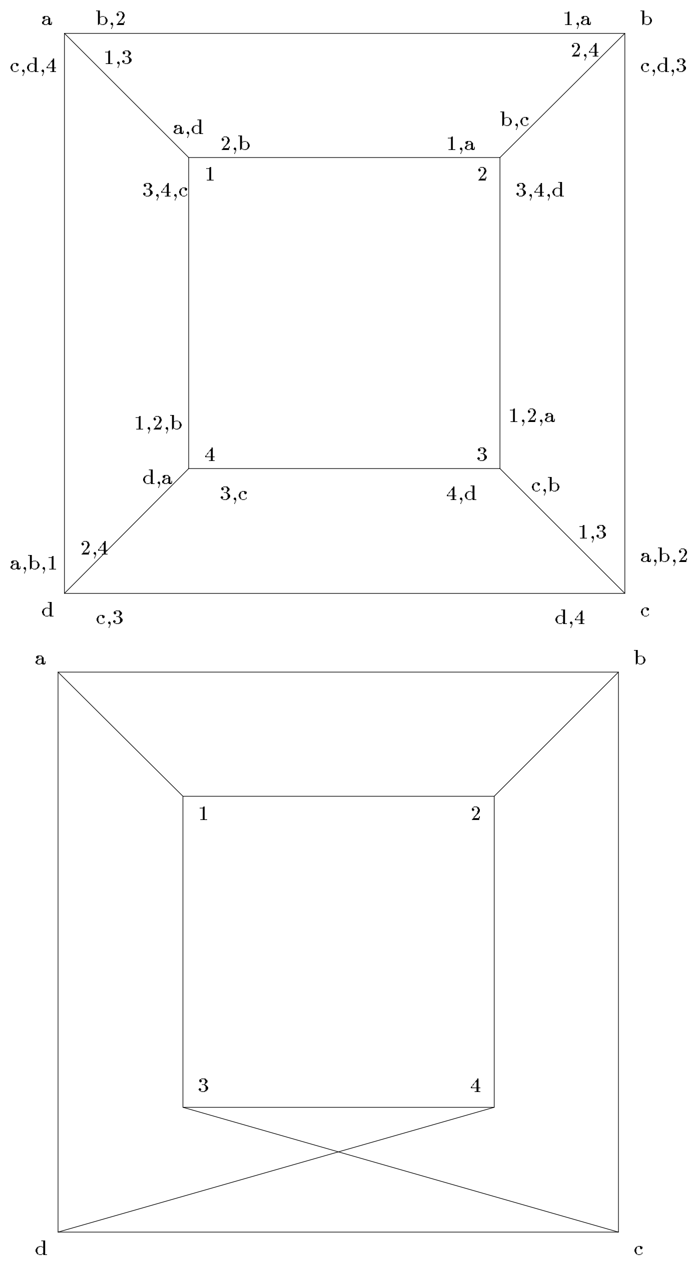

Figure 3: Two non-isomorphic 3-connected graphs with identical shortest path families.

## 2.2 Realizable path families

A shortest path family may or may not supply the complete information on shortest paths. However, it is interesting to determine whether a realizable family determines a unique graph.

**Theorem 2.3** *If $\mathcal{F}$ is a realizable family then there is a unique graph with it as its shortest path family. Moreover, there is a polynomial time algorithm to test if a given family is realizable.*

Proof (Outline) For any two vertices $u, v \in V$ we need to determine whether or not $\{u, v\}$ is an edge. For each vertex $u$ and each $i < \deg(u)$ define the set $U_{u,i}$ as the set of all vertices $x \in F_{u,i}$ such that $x$ does not occur in any of the other sets of the family $\mathcal{F}_u$, i.e.

$$U_{u,i} = F_{u,i} \setminus \{F_{u,j} : j \neq i, j < \deg(u)\}.$$

Now fix a vertex $u \in V$ and consider a link $i < \deg(u)$. We want to determine the vertex $v$ which is adjacent to $u$ along link $i$.

First we observe that $v$ must be an element of the set $U_{u,i}$. Define an ordering on the set $U_{u,i}$ as follows:

> $x <_{u,i} y$ if and only if there is a shortest path from $u$ to $y$ that passes through $x$.

It is clear that $<_{u,i}$ is a partial ordering. Moreover we can compute the minimal elements of this ordering. Then the element $v$ adjacent to $u$ is the $<_{u,i}$-minimal element such that $u$ is in the set $U_{v,j}$ for some $j$. This completes the proof of the theorem. ∎

The problem of constructing graphs having "complex" interval routing schemes was considered in [9, 11]. Theorem 2.3 has an important implication in the context of implicit routings [4]. It implies the (nonconstructive) existence of graphs on $n$ vertices in which every implicit routing scheme which represents all shortest paths (we call such a scheme a full information scheme) in a graph requires a high number of memory bits.

**Theorem 2.4** *There exist graphs such that such that any full information implicit shortest path routing scheme requires a total of $\Omega(n^2)$ memory bits.*

Proof (Outline) Clearly there exist $2^{\Omega(n^2)}$ graphs. Hence there exist graphs whose encoding requires $\Omega(n^2)$ bits. However, by Theorem 2.3 every such graph is uniquely determined by a shortest path family. Therefore there must exist a graph such that any family encoding it requires at least $\Omega(n^2)$ bits. ∎

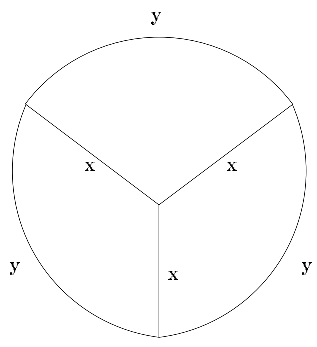

Figure 4: A unique shortest path graph with edge-weights equal to 1. The arcs represent paths with the corresponding number of vertices. It is assumed that $y$ is odd and $2x > y$.

## 3 Unique Shortest Path Graphs

In this section we study the "Unique Shortest Path" (or USP for short) graphs. A graph $G$ is called a USP graph if for any two vertices $u, v$ there is a unique shortest path connecting $u$ to $v$. If we allow weights on the links then by analogy we obtain the class of weighted unique shortest path graphs.

It is clear that USP graphs are realizable only by pairwise disjoint path families. Although it is easy to recognize whether a graph is a USP graph (run a first-depth algorithm from each source vertex labeling the nodes with the distance from the source), no elegant characterization of USP graphs is known.

Examples of such graphs include trees, cliques and odd length cycles. In addition, one can contruct other USP graphs. These include, the Peterson graph, as well as the graphs depicted in Figures 4 and 5. Additional graphs can be constructed as follows. Take any USP graph and "hang" USP graphs on its nodes. Clearly, the resulting graph is a USP graph. It is not known if USP graphs exist that have high implicit routing memory requirements.

If we allow weights on the edges it is possible to find an edge-weight assignment that makes the graph a USP graph. In fact we can prove the following theorem.

**Theorem 3.1** *For any graph $G$ we can assign integer weights (having total sum $O(|E|\delta)$, where $\delta$ be the diameter of the graph) to the edges in such a*

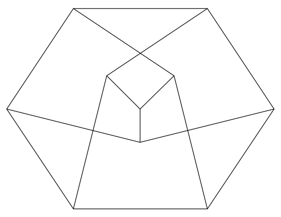

Figure 5: A unique shortest path graph with edge-weights equal to 1.

*way that it becomes a unique shortest path graph.*

Proof (Outline) We label the edges of the graph with weights having total sum at most $|E|(2\delta + 1) = O(|E|\delta)$, where $\delta$ be the diameter of the graph. To determine the weights we argue as follows. Take a spanning tree of the graph. Give the edges of the spanning tree the weight 1 and all other edges the weight $\delta$. It is easy to see from the choice of weights that the shortest path between any two vertices consists of tree edges. This proves the theorem. ■

## 4 Conclusion

In this paper we considered the problem of constructing graphs from shortest path information. We proved the existence of graphs requiring a "high" total of memory bits. It is interesting to note that very little is known on the reconstruction problem from "partial" shortest path information (i.e. feasible families). The class of unique shortest path graphs seems to be interesting in its own right and it would be interesting to know if such graphs exist requiring a "high" total of memory bits.

## References

[1] E. M. Bakker, "Combinatorial Problems in Information Networks and Distributed Datastructuring", *Ph.D. Thesis*, Dept. of Computer Science, Utrecht University, The Netherlands, 1991.

[2] E. M. Bakker, J. van Leeuwen and R. B. Tan, "Prefix Routing Schemes in Dynamic Networks", Tech. Report RUU-CS-90-10, Dept. of Computer Science, Utrecht University, The Netherlands, Mar. 1990.

[3] E. M. Bakker, J. van Leeuwen and R. B. Tan, "Linear Interval Routing Schemes", Tech. Report RUU-CS-91-7, Dept. of Computer Science, Utrecht University, The Netherlands, Feb. 1991.

[4] M. Flammini, G. Gambosi and S. Salamone, "Boolean Routing", in WDAG'93, SVLNCS Vol. 725, pp. 219 - 233, 1993.

[5] G. N. Frederickson and R. Janardan, "Designing Networks with Compact Routing Tables", *Algorithmica*, vol. 3, 1988, pp. 171-190.

[6] G. N. Frederickson and R. Janardan, "Efficient Message Routing in Planar Networks", *SIAM J. Comput.*, vol. 18, no. 4, Aug. 1989, pp. 843-857.

[7] G. N. Frederickson and R. Janardan, "Space-Efficient Message Routing in $c$-Decomposable Networks", *SIAM J. Comput.*, vol. 19, no. 1, Feb. 1990, pp. 164-181.

[8] F. Harary, *Graph Theory*, Addison-Wesley Publishing Co., Reading, MA, 1969.

[9] E. Kranakis, D. Krizanc and S. S. Ravi, "On Multiple Linear Interval Routing Schemes", in proceedings of WG'93 (Workshop on Graph Theoretic Concepts in Computer Science), Vol. 790, Springer Verlag LNCS.

[10] D. Peleg and E. Upfal "A Tradeoff between Space and Efficiency for Routing Tables", in ACM STOC 1988, pages 43 - 52 (also in Journal of ACM, Vol. 36, pages 510 - 530, 1989).

[11] P. Ružička, "On the Efficiency of Interval Routing Algorithms", in *Proceedings of MFCS*, Carlsbad, Chechoslovakia, Aug. - Sep. 1988, Springer Verlag LNCS vol. 324 (Edited by M. P. Chytil, L. Janiga and V. Koubek), pp. 492 - 500, 1988.

[12] N. Santoro and R. Khatib, "Labelling and Implicit Routing in Networks", *The Computer Journal*, vol. 28, no. 1, 1985, pp. 5-8.

[13] J. van Leeuwen and R. B. Tan, "Computer Networks with Compact Routing Tables", in *The Book of L*, Edited by G. Rozenberg and A. Salomaa, Springer-verlag, Berlin 1986, pp 259-273.

[14] J. van Leeuwen and R. B. Tan, "Interval Routing", *The Computer Journal*, vol. 30, no. 4, 1987, pp. 298-307.