UC Irvine
ICS Technical Reports

Title
Shortest paths in orthogonal graphs

Permalink
https://escholarship.org/uc/item/6w02c4bm

Authors
Bhatia, Sandeep
Hirschberg, Daniel
Scherson, Isaac D.

Publication Date
1991-05-01

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Notice: This Material
may be protected
by Copyright Law
(Title 17 U.S.C.)

Z
699
C3
no. 91-45

LIBRARY
University of California
IRVINE

Shortest Paths in Orthogonal Graphs

Sandeep Bhatia[1], Daniel Hirschberg[2] and Isaac D. Scherson[2]

[1]Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

[2]Department of Information and Computer Science
University of California
Irvine, CA 92717

Technical Report #91–45
May 1, 1991

Shortest Paths in Orthogonal Graphs

Sandeep Bhatia[1], Daniel S. Hirschberg[2] and Isaac D. Scherson[2]

[1]Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

[2]Department of Information and Computer Science
University of California
Irvine, CA 92717

Abstract

Orthogonal graphs were introduced as a simple but powerful tool for the description and analysis of a class of interconnection networks. Routing, and hence finding shortest paths between any two nodes of an orthogonal graph, becomes an important problem. It is shown in this paper that routing in this class of graphs reduces to a node covering problem in the bipartite coverage graph of the orthogonal graph. A minimum cover clearly leads to a shortest path. In general, the problem of finding the minimum node cover in a bipartite graph is NP-complete. However, the bipartite coverage graphs corresponding to orthogonal graphs have a regular pattern of edges. This allows the development of a routing algorithm which results in a minimum cover. The procedure executes in polynomial time in the number of bit-nodes of the bipartite graph. It therefore results in a shortest path algorithm whose time complexity is quadratic in the logarithm of the number of nodes in the original orthogonal graph.

Keywords: Orthogonal Graphs, Coverage Graphs, Routing, Minimum Cover, Shortest Path.

1 Introduction

Orthogonal graphs were recently introduced as a unified framework for the description and analysis of a large class of interconnection networks [10, 11, 12]. The graphs are defined by a construction rule based on three parameters, namely the number of nodes (2^m), a *mask length* $n \leq m$, and a set of connectivity modes $Q \subseteq \{0, 1, ..., m-1\}$. Two nodes labeled with m-bit strings are connected if there exist an integer $q \in Q$ such that the node labels differ in at most $m - n$ bits starting at bit position $(q + n) \bmod m$. The resulting structures correspond to spanning bus hypercube-like structures and describe systems like the binary hypercube, Batcher's Multidimensional Access (MDA) memories, spanning bus hypermeshes, most known multistage interconnection networks, and many others. In this paper we address the routing problem in orthogonal graphs and suggest a simple algorithm to find shortest paths. It is clear that, given that orthogonal graphs describe such a large class of systems, the proposed algorithm applies to those systems as well.

It will be shown that the connectivity of an orthogonal graph can be depicted by means of a bipartite graph which consists on one side of nodes representing label bits and, on the other side, cover nodes corresponding to integers in the set Q. A path between any two nodes is an ordered sequence of nodes which differ in at most $m - n$ bits covered by some mode defined in Q. The routing problem reduces to that of finding a cover set of a subset of label bits, those in which source and destination labels differ. However, the general problem of finding a minimum cover in arbitrary bipartite graphs is NP-complete. Fortunately, some properties of the coverage graph of orthogonal graphs allow the development of a simple routing procedure which results in a shortest path. We show here that a minimum solution can be found in at most $O(m^2)$ steps, where m is the logarithm of the number of nodes in the orthogonal graph.

In Section 2, a review of orthogonal graphs is given for completeness and to provide the nomenclature used in the paper. Section 3 describes the algorithm to identify a minimum length path between any two nodes of an orthogonal graph.

2 Review of Orthogonal Graphs

An orthogonal graph is an undirected graph $G(n, m, Q)$ which has 2^m nodes labeled distinctly from Y_m, the set of m-bit binary labels. The edges of graph G are defined by a construction rule parameterized by n, which is an integer less than m, and Q, the *set of modes*, which is a subset of the integers $\{0, ..., m-1\}$. Let y_i denote the i-th bit of a label, $\underline{y} \in Y_m$, and let $\oplus$ denote the exclusive-OR operation.

The edge construction rule is as follows: Two *distinct* nodes labeled $\underline{y}$ and $\underline{y}'$ are connected by an edge if and only if there exists a $q \in Q$ such that

$$\sum_{i=q}^{(q+n) \bmod m} y_i \oplus y_i' = 0$$

The nodes are said to be *orthogonal mode q*. This is denoted symbolically as $\underline{y} \perp_q \underline{y}'$.

The following properties are ascribed to orthogonal graphs:

1. Connectivity: An orthogonal graph $G(n, m, Q)$ is connected if and only if

$$\forall i \in [0, m-1], \exists q \in Q \textit{ such that } |(q+n) \bmod m - i| < (m - n)$$

We say that mode q *covers* bit position i.

2. Disjoint modes: Q is said to be a *disjoint set of modes* if and only if for all $i \in [0, m-1]$ there is a unique mode $q \in Q$ which covers bit position i.

3. Omega graphs: A connected orthogonal graph with a disjoint set of modes is called an *omega graph.* We denote these graphs as $\omega G(n, m)$ because, in order to satisfy connectivity and disjointness, there must be some integer ω such that $m = \omega(m-n)$ and $Q = \{0, (m-n), 2(m-n), ..., (\omega-1)(m-n)\}$. Note that any omega graph with $Q = \{l, l+(m-n), ..., l+(\omega-1)(m-n)\}$ for any integer $0 < l < m-n$ is isomorphic to the graph for $l = 0$. The diameter of an omega graph is the size of Q ($\#Q$).

3 Routing

The routing problem in orthogonal graphs was outlined in [11, 12]. Because of the similarity between this problem and that of Boolean minimization, it was conjectured that finding a shortest path would require an exhaustive search as a solution is not unique. In this paper we use some properties of the bipartite coverage graphs of an orthogonal graph to develop an algorithm which results in a shortest path solution.

As was outlined in [11, 12], to find a path between nodes $\underline{y}$ and $\underline{y}'$, we first compute a routing tag $\underline{t} = \underline{y} \oplus \underline{y}'$. The tag $\underline{t}$ has 1's in all those bit positions in which the nodes labels differ.

We define the *coverage graph* for an orthogonal graph $G(n, m, Q)$ to be a bipartite graph G^* in which the left-hand-side nodes (bit nodes) correspond to label bits and the right-hand-side nodes (q-modes or cover nodes) correspond to modes in Q. A q-mode is adjacent to a bit node if and only if the mode covers that bit position. A connected orthogonal graph will have a coverage graph in which all bit nodes are connected to at least one cover node. For the remainder of our discussion we consider only connected orthogonal graphs.

A set α of cover nodes in the coverage graph corresponds to (generates) a set of paths in the orthogonal graph. If $q \in \alpha$, then a node labeled with a bit string $\underline{y}$ is adjacent to a set of nodes whose labels differ, from $\underline{y}$, in at most $m-n$ bits starting at bit position $(q+n) \bmod m$. Path sequences can be generated as follows. Starting from a given source node in the orthogonal graph, iterate obtaining the label of the next node in a path by inverting the bits of the current node label at bit positions corresponding to a subset of the bit nodes adjacent to one of the cover nodes of α. Note that a regular $(m-n)$-ary tree can be constructed for each node in the orthogonal graph. This tree represents all possible shortest paths available from node $\underline{y}$ to all other nodes.

We define the *routing graph* as the subgraph of the coverage graph obtained by deleting the bit nodes corresponding to the 0-bits in $\underline{t}$ and the edges associated with those deleted nodes. The routing problem is reduced to finding a minimum subset of the q-modes that cover the bit nodes in the routing graph.

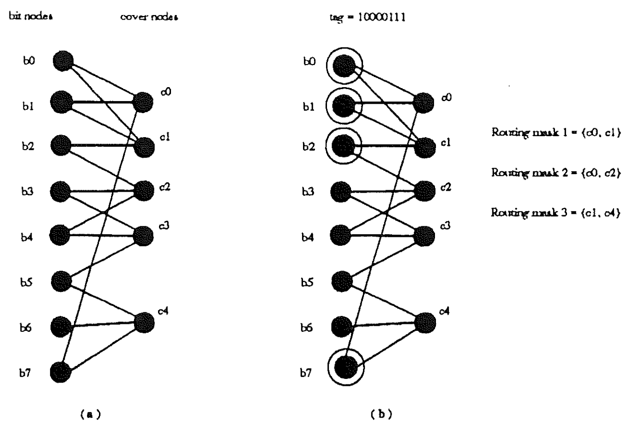

Figure 1: (a) Coverage graph for $G(5, 8, \{0, 2, 3, 5, 6\})$ (b) Routing problem for path from 01101011 to 1110100

In Figure 1, the left bipartite graph is the general converage graph for $G(5, 8, \{0, 2, 3, 5, 6\})$. An example routing problem is given in the right graph of Figure 1. The routing tag, $\underline{t} = \underline{y} \oplus \underline{y}'$, is shown at the top and the circled nodes, corresponding to 1-bit positions in $\underline{t}$, are those that must be covered. A solution minimum covering set of q-modes corresponds to a *routing mask* set which need not be applied in any predefined order. Note that the minimum covering set is not necessarily unique. For this example, the three solution routing mask sets are indicated.

It is worth noting here that if $G(n, m, Q)$ is an ω-graph, the coverage set will contain all cover nodes in the routing graph as bit nodes are uniquely covered. Such a case is trivial and will not be discussed any further.

We define the following notation :

b_i : a bit node in the coverage graph.

c_j : a cover node in the coverage graph.

$adj(\alpha)$: the set of nodes adjacent to node α in the coverage graph. See Figure 2 for examples.

β_i : the set of bit nodes in the routing graph that are adjacent to the i^{th} cover node of the minimum cover. Note that the union of all β_i's gives all 1-bits in the routing tag $\underline{t}$.

Some of the β_i's may have non-empty intersection. Arbitrarily, remove common bit nodes from intersecting β_i's so that they become a disjoint partition of the set of 1-bits in $\underline{t}$.

$\hat{\beta}_i$: a subset of bit nodes in the routing graph that are adjacent to the i^{th} cover node of the minimum cover so that the set of $\hat{\beta}_i$'s is a disjoint partition of the 1-bits in $\underline{t}$.

The set of $\hat{\beta}_i$'s gives the minimum path (indices of the intermediate nodes, if any) for the routing tag $\underline{t}$. Starting from the source node, the index of the next node in a minimum path is obtained by inverting the index at bit positions corresponding to one of the $\hat{\beta}_i$'s. Each of the $\hat{\beta}_i$'s is considered only once. The destination node is reached when all the $\hat{\beta}_i$'s have been

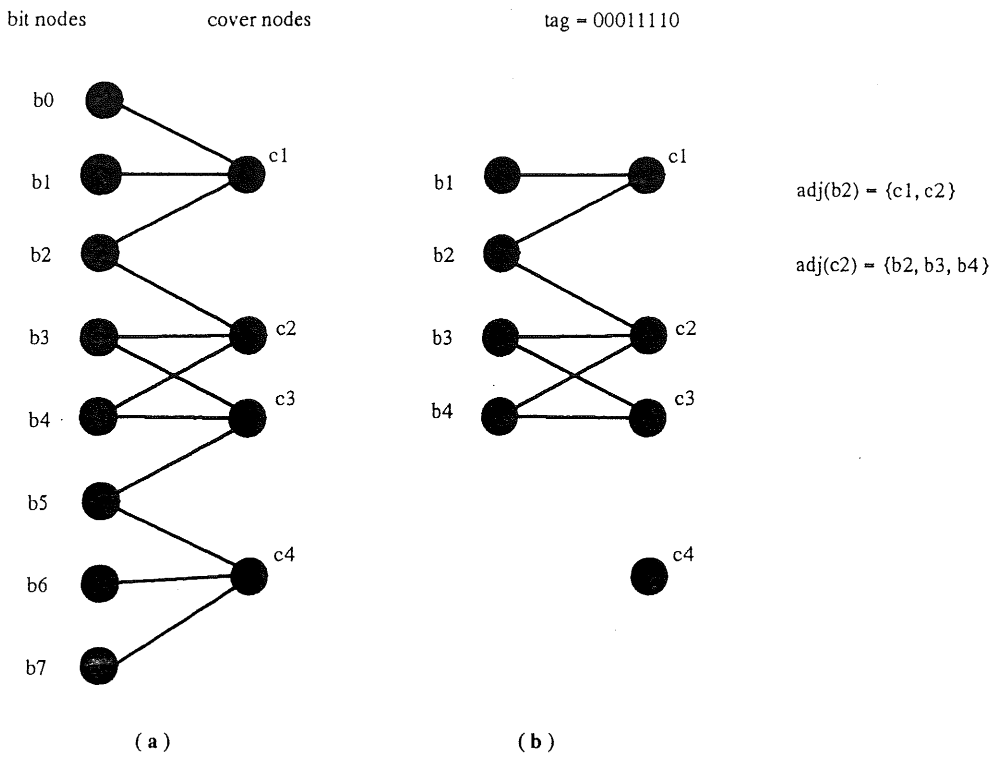

Figure 2: A typical coverage graph (a) Coverage graph for $G(5, 8, \{0, 3, 5, 6\})$, (b) Routing graph for path from 01000111 to 01011001

considered.

For the example of Figure 2, c_1 and c_2 form a minimum cover. The bit nodes adjacent to them are $\beta_1 = \{b_1, b_2\}$ and $\beta_2 = \{b_2, b_3, b_4\}$. Let $\hat{\beta}_1 = \{b_1, b_2\}$ and $\hat{\beta}_2 = \{b_3, b_4\}$. Therefore, a minimum-length path from 01000111 to 01011001 is

$$01000111 - 01000001 - 01011001$$

The order of using the $\hat{\beta}_i$'s is not important, so another minimum path is

$$01000111 - 01011111 - 01011001$$

For each cover node c_j in the coverage graph, $adj(c_j)$ is a contiguous subset of the bit nodes if they are considered as forming a ring $b_0, b_1, ..., b_{m-1}, b_0$. Accordingly, for cover node c_j, we denote $adj(c_j)$ by the pair of indices $firstb(c_j)$ and $lastb(c_j)$. If $firstb(c_j) \leq lastb(c_j)$ then $adj(c_j) = \{b_{firstb(c_j)}, ..., b_{lastb(c_j)}\}$. Otherwise, $adj(c_j) = \{b_{firstb(c_j)}, ..., b_{m-1}, b_0, ..., b_{lastb(c_j)}\}$. It follows that, in the routing graph, $adj(c_j)$ is also a contiguous subset of the bit nodes.

Similarly, for each bit node b_i in the coverage graph, $adj(b_i)$ is a contiguous subset of the cover nodes if they are considered as forming a ring $c_0, c_1, ..., c_{n-1}, c_0$. Accordingly, for bit node b_i, we denote $adj(b_i)$ by the pair of indices $firstc(b_i)$ and $lastc(b_i)$. If $firstc(b_i) \leq lastc(b_i)$

then $adj(b_i) = \{c_{firstc(b_i)}, \ldots, c_{lastc(b_i)}\}$. Otherwise, $adj(b_i) = \{c_{firstc(b_i)}, \ldots, c_{n-1}, c_0, \ldots, c_{lastc(b_i)}\}$. In the routing graph, $adj(b_i)$ is also a contiguous subset of the cover nodes.

The following is an algorithm solving the covering problem :

Let β^* be the set of indices of the 1-bits in $\underline{t}$.
We will find *optT*, the minimum size subset of Q that covers β^*.

```
$optT \leftarrow Q$                initially the smallest subset that works is the entire set

foreach $z \in Q$ do             $z$ is the index of the first cover node to be placed in $T$

  $x \leftarrow z$
  $T \leftarrow \{c_x\}$                 $T$ builds up to a candidate for $optT$
  $S \leftarrow \beta^* - adj(c_x)$        $S$ is the part of $\beta^*$ not yet covered

  while $S$ is non-empty do
    if $\exists i \in S$ s.t. $i > lastb(c_x)$
    then $nextb \leftarrow \min\{i \in S | i > lastb(c_x)\}$
    else $nextb \leftarrow \min\{i \in S\}$
                                 $nextb$ is the (circular) first index $i$ s.t. $b_i$ not yet covered
    $x \leftarrow lastc(b_{nextb})$        $x$ is the (circular) last index $j$ s.t. $c_j$ covers $b_{nextb}$
    $T \leftarrow T \cup \{c_x\}$           add one more cover node to $T$
    $S \leftarrow S - adj(c_x)$         remove the bit nodes in $S$ covered by $c_x$
  endWhile

  if $(|T| < |optT|)$ then $optT \leftarrow T$

endFor
```

The correctness of the algorithm is easily shown. One of the iterations of the **for** loop will initialize the set T to contain a cover node of a minimum cover set T^*. As we next see, this iteration will produce a minimum cover node set T. Each iteration of the **while** loop determines the circular first bit node b_{nextb} not yet covered by T and then covers it with the circular last possible cover node c_x. Any other cover node that could cover b_{nextb} will not cover any bit nodes not covered by c_x or previously chosen cover nodes. Therefore each chosen cover node in T will perform as well its counterpart in T^* and the cardinality of T will be equal to that of T^*.

The preprocessing to calculate *lastb* and *lastc* takes time $O(m)$. Each iteration of the **while** loop takes constant amortized time. Each iteration of the **for** loop takes time $O(m)$. The **for** loop will be iterated $|Q| \leq m$ times. Therefore, the time complexity of the algorithm to determine the minimum size cover node set is $O(m^2)$. From the solution cover set, a shortest path can be produced in time proportional to the length of the path, which is $O(m)$.

We conjecture that it may be possible to calculate a restriction on the choices of which item to be first placed in the minimum cover set. Such a restriction would reduce the complexity of this algorithm correspondingly.

4 Conclusions

In this paper, an algorithm to identify a shortest path in an orthogonal graph is presented. Since orthogonal graphs can be used to describe various interconnection networks, this provides a tool to find a shortest path from any node to any other node in a general interconnection network.

The work can be further expanded to identify several different shortest paths, if more than one shortest path exists, or to find a minimum length path in the presence of a fault in a set of nodes and links in the interconnection network.

References

[1] K. E. Batcher, *The Multidimensional Access Memory in STARAN,* IEEE Transactions on Computers, Vol. C-26, No. 2, February 1977, pp. 172-177.

[2] R. E. Buehrer et al., *The ETH Multiprocessor EMPRESS: A Dynamically Reconfigurable MIMD System*, IEEE Transactions on Computers, Vol. C-31, No. 11, November 1982, pp. 1035-1044.

[3] L. N. Bhuyan and D. P. Agrawal, *Generalized Hypercube and Hyperbus Structures for a Computer Network,* IEEE Transactions on Computers, Vol. C-33, No. 4, April 1984, pp. 323-333.

[4] R. P. Grimaldi, *Discrete and Combinatorial Mathematics. An Applied Introduction,* Reading, Addison-Wesley Publishing Co., Inc., 1989.

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

[6] K. Hwang and D. Kim, *Generalization of Orthogonal Multiprocessor for Massively Parallel Computation,* Proceedings of Frontiers 88, 2nd Symposium on the Frontiers of Massively Parallel Computation.

[7] K. Hwang, P. S. Tseng and D. Kim, *An Orthogonal Multiprocessor for Large-Grain Scientific Computations,* IEEE Transactions on Computers, Vol. C-38, No. 1, January 1989, pp. 47-61.

[8] I. D. Scherson and Y. Ma, *Vector Computations in an Orthogonal Memory Access Multiprocessing System,* Proceedings of the 8th Symposium on Computer Arithmetic, May 1987, pp. 28-37. February 1989, pp.238-249.

[9] I. D. Scherson and Y. Ma, *Analysis and Applications of the Orthogonal Access Multiprocessor,* The Journal of Parallel and Distributed Computing, Vol. 7, No. 2, October 1989, pp.232-255.

[10] I. D. Scherson, *Definition and Analysis of a Class of Spanning Bus Orthogonal Multiprocessing Systems,* Proceedings of the 1990 ACM Computer Science Conference, February 19-22, 1990, Washington DC, pp. 194-200.

[11] I. D. Scherson, *Orthogonal Graphs and the Analysis and Construction of a Class of Multistage Interconnection Networks,* Proceedings of the 1990 International Conference on Parallel Processing, August 1990.

[12] I. D. Scherson, *Orthogonal Graphs for the Construction of a Class of Interconnection Networks,* IEEE Transactions on Parallel and Distributed Systems, in press.

[13] I. D. Scherson, Ashish Mehra and Jennifer Rexford, *Toward Scalable Algorithms for Multidimensional Access Systems* Proceedings of the IEEE 1990 Symposium on the Frontiers of Massively Parallel Computation, Maryland, October 1990.

[14] C. L. Seitz, *The Cosmic Cube,* Communications of the ACM, Vol. 28, No. 1, January 1985, pp. 22-33.

[15] H. J. Siegel, *Interconnection Networks for Large Scale Parallel Processing : Theory and Case Studies,* Reading, Lexington Books, Lexington, Mass. 1984.

[16] H. S. Stone, *High Performance Computer Architecture,* Reading, MA, Addison Wesley, 1987.

[17] A. Tucker, *Applied Combinatorics,* Reading, John Wiley & Sons, Inc., 1980.