Seeing Around Corners: Fast Orthogonal Connector Routing

Kim Marriott[1], Peter J. Stuckey[2], and Michael Wybrow[1]

[1] Caulfield School of Information Technology,
Monash University, Caulfield, Victoria 3145, Australia,
{Michael.Wybrow,Kim.Marriott}@monash.edu
[2] Department of Computing and Information Systems,
University of Melbourne, Victoria 3010, Australia,
pstuckey@unimelb.edu.au

Abstract. Orthogonal connectors are used in drawings of many types of network diagrams, especially those representing electrical circuits. One approach for routing such connectors has been to compute an orthogonal visibility graph formed by intersecting vertical and horizontal lines projected from the corners of all obstacles and then use an A* search over this graph. However the search can be slow since many routes are in some sense topologically equivalent. We introduce obstacle-hugging routes which provide a canonical representative for a set of topologically equivalent routes. We also introduce a new *1-bend visibility graph* that supports computation of these canonical routes. Essentially this contains a node for each obstacle corner and connector endpoint in the diagram and an edge between two nodes iff they can be connected using an orthogonal connector with one bend. We show that the use of a 1-bend visibility graph significantly improves the speed of orthogonal connector routing.

Keywords: orthogonal routing, visibility graphs, circuit diagrams

1 Introduction

Most interactive diagram editors provide some form of automatic connector routing between shapes whose position is fixed by the user. Usually the editor computes an initial automatic route when the connector is created and updates this each time the connector end-points (or attached shapes) are moved. Orthogonal connectors, which consist of a sequence of horizontal and vertical line segments, are a particularly common kind of connector, used in ER and UML diagrams among others. Wybrow *et al*. [1] gave polynomial time algorithms for automatic object-avoiding orthogonal connector routing which are guaranteed to minimise length and number of bends.

The connector routing algorithm given in Wybrow *et al*. [1] uses a three stage process. The first stage computes an *orthogonal visibility graph* in which edges in the graph represent horizontal or vertical lines of visibility from the corners and connector ports of each obstacle. Connector routes are found using an A* search through the orthogonal visibility graph that finds a route that minimizes bends and overall connector length. Finally, the actual visual route is computed. This step orders and nudges apart

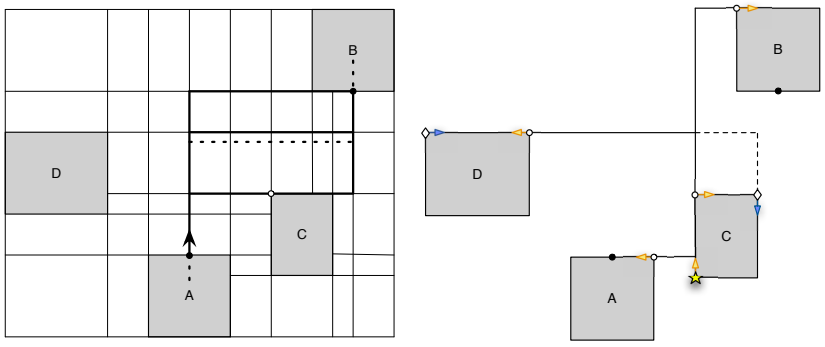

Fig. 1. (a) The orthogonal visibility graph and three topologically equivalent routes of equal cost between objects *A* and *B*. (b) The 1-bend visibility graph showing the visibility of nodes from the bottom left of *C* (star) going upwards to the circled nodes, with arrows indicating directions. The asymmetry is illustrated by the edge from the diamond node of D to the diamond node on *C*.

the connectors in shared segments so as to ensure that unnecessary crossings are not introduced, that crossings occur at the start or end of the shared segment and that connectors where possible run down the center of alleys. Unfortunately, for larger dense graphs the approach can be quite slow, the dominating cost is the time taken to find the optimal route for each connector in the second stage.

One of the main reasons that orthogonal connector routing is slow is that there are many "topologically equivalent" routes of equal cost to each vertex, greatly increasing the search space size. Figure 1 illustrates the problem. However, for our purposes these routes are equivalent as computation of the actual visual route will lead to an identical final layout (using the dashed edge which goes midway between objects *B* and *C*). The main contributions of this paper are to:

- Identify a class of connector route we call *obstacle-hugging* that provide a canonical representative for a set of topologically equivalent routes;
- Present a new kind of visibility graph for computing these routes which we call the *1-bend visibility graph* in which nodes are object vertices and a search direction and there is an edge between two nodes if they can be connected using an orthogonal connector with one bend;
- Provide theoretical and empirical proof that this new approach is significantly faster than the current approach.

Our new approach has similar characteristics to the standard visibility graph used in poly-line connector routing. If we have n objects then the orthogonal visibility graph has $O(n^2)$ nodes, $O(n^2)$ edges and an optimal route can be $O(n^2)$ in length. In contrast the 1-bend visibility graph has $O(n)$ nodes, $O(n^2)$ edges and any optimal (orthogonal) route is $O(n)$ in length. This is similar to poly-line connector routing where the *standard visibility graph* has the same asymptotic sizes [2]. It also bears some similarities to the

rectangularization approach of Miriyala *et al.* [3], though rectangularization is heuristic and so unlike our approach is not guaranteed to find an optimal route.

Orthogonal connector routing has been extensively studied in computational geometry, in part because of its applications to circuit design. Lee *et al.* [4] provides an extensive survey of algorithms for orthogonal connector routing, while Lenguauer [5] provides an introduction to the algorithms used in circuit layout. The 1-bend visibility graph, is as far as we are aware, completely novel.

2 Obstacle Hugging Routes

For simplicity we assume obstacles are rectangles: more complex shapes can be approximated by their bounding rectangle and assume for the purposes of complexity analysis that the number of connector points on each object is a fixed constant.

We are interested in finding a poly-line route of horizontal and vertical segments for each connector. We specify such an orthogonal route as sequence of points $p_1, \ldots, p_m$ where p_1 is the start connector point, p_m the end connector point and $p_2, \ldots, p_{m-1}$ the bend points. Note that orthogonality means that either $x_{j+1} = x_j$ or $y_{j+1} = y_j$ where $p_i = (x_i, y_i)$. We require that the routes are *valid*: they do not pass through objects and only contain right-angle bends, i.e., alternate horizontal and vertical segments.

We wish to find routes that are short and which have few bends: We therefore assume our penalty function $\mathsf{p}(R)$ for measuring the quality of a particular route R is a monotonic function of the length of the path, $||R||$, and the number of bends (or equivalently segments) in R. A route R between two connector points is *optimal* if it is valid and it minimises $\mathsf{p}(R)$.

Given a valid object-avoiding orthogonal route $p_1, \ldots, p_i, p_{i+1}, \ldots, p_m$ where $p_i = (x_i, y_i)$, and $p_{i+1} = (x_{i+1}, y_{i+1})$ and $x_{i+1} = x_i$, that is p_i, p_{i+1} is a vertical edge, we can define a *vertical edge move* creating a new orthogonal route $p_1, \ldots, p'_i, p'_{i+1}, \ldots, p_m$ where $p'_i = (x, y_i)$ and $p'_{i+1} = (x, y_{i-1})$. We can similarly define a *horizontal edge move*. Note that an edge move may make the route invalid, e.g. $p_i = p_{i-1}$. We can restore this by removing unnecessary points and edges, if $p_i = p_{i-1}$ we obtain a new proper orthogonal route $p_1, \ldots, p_{i-2}, p'_{i+1}, \ldots, p_m$.

An edge move is *topology preserving* if the rectangle enclosed by $p_i, p_{i+1}, p'_{i+1}, p'_i$ does not overlap any obstacles. Topology preserving edge moves induce a natural equivalence relationship on the set of optimal routes: Two optimal orthogonal routes R_1 and R_2 from p_1 to p_m are *topologically equivalent* if there is a sequence of topology preserving edge moves that map R_1 to R_2. Because R_1 and R_2 are optimal no edge move will remove points and edges since otherwise the route was not optimal. This means each move is invertible and so this is a true equivalence relationship. Each of the routes shown in Figure 1 are topologically equivalent since they can be mapped to one another by a single horizontal edge move.

One of the main reasons that existing orthogonal connector routing is slow is that the A* algorithm explores a large number of topologically equivalent routes. In order to reduce the search space we want to choose a single canonical route for each equivalence class. Object-hugging routes provide such a canonical representative:

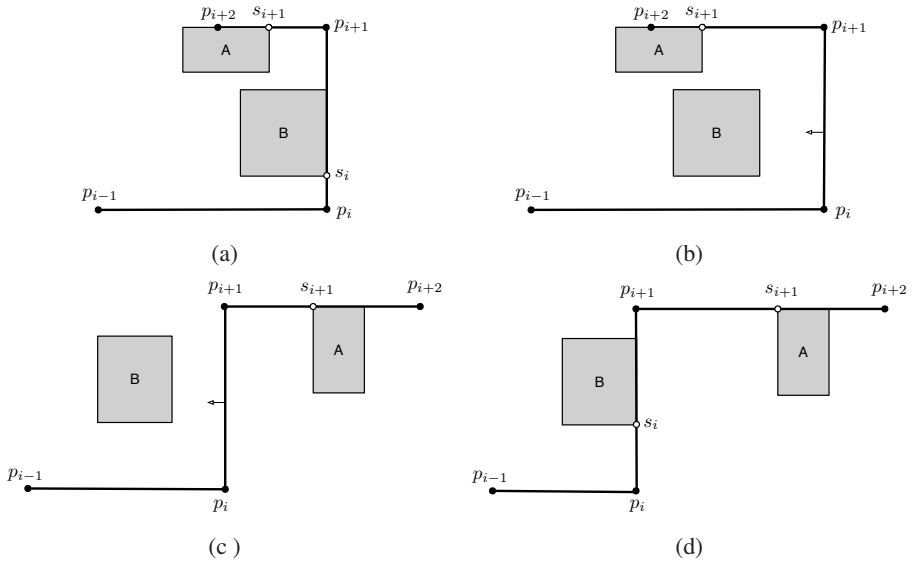

Fig. 2. Various cases in the construction of an optimal obstacle-hugging route.

An orthogonal route $p_1, \ldots, p_m$ is *obstacle-hugging* if each segment $|p_j, p_{j+1}|$ for $j = 2, \ldots, m-2$ intersects the boundary of an object in the direction from p_{j-1} to p_j. This means the path bends around the far side of each object when following the route from its start.

The definition means that all intermediate (i.e., not the first or last) segments $|p_j, p_{j+1}|$ on an obstacle-hugging route have a supporting object o_j s.t. a vertex s_j of o_j lies on the segment. We call s_j the *support vertex* for the segment $|p_j, p_{j+1}|$.

In Figure 1 the bottom route between *A* and *B* is obstacle-hugging since the second segment intersects the top of the shape and the first segment goes towards the top. Neither the middle route or the top route along the bottom of *B* is obstacle-hugging.

Theorem 1. *Given an optimal orthogonal route from $p_1, \ldots, p_m$ there is an optimal orthogonal route from p_1 to p_m which is obstacle-hugging.*

Proof. We show how, given the optimal route $R = p_1, \ldots p_m$, to construct an topologically equivalent route which has the same distance and number of bends and is obstacle-hugging.

The construction is iterative backwards for $i = m-1$ to 2. For each i it computes a support vertex s_i for segment $|p_i, p_{i+1}|$ such that $|s_i, p_{i+1}|$ has a nonempty overlap with the side of some object in the direction from p_{i-1} to p_i and possibly moves p_i and p_{i+1} while still maintaining the same topology and overall cost.

Assume that we have computed support vertex s_{i+1}, we show how to compute s_i. There are two cases to consider: p_{i-1}, p_i, p_{i+1} and p_{i+2} form a U-turn (Figure 2(a)) or p_{i-1}, p_i, p_{i+1} and p_{i+2} form a step (Figure 2(c)).

In the case of the U-turn, we must have an object B whose boundary lies along the segment $|p_i, p_{i+1}|$ as shown in Figure 2(a) because otherwise we could move the segment as shown in Figure 2(b) which would imply that R was not optimal. We therefore set s_i to be the vertex of object B closest to p_{i-1}. Clearly $|s_i, p_{i+1}|$ intersects the side of the object B in direction from p_{i-1} to p_i.

In the case of the step, we move the segment $|p_i, p_{i+1}|$ toward p_{i-1} by decreasing the length of segment $|p_{i-1}, p_i|$ and increasing the length of segment $|p_{i+1}, p_{i+2}|$ until it runs into some object B. This must happen before the length of segment $|p_{i-1}, p_i|$ decreases to 0 because otherwise we could have reduced the cost of R by removing this segment and the two bend points which would imply that R was not optimal. The construction is shown in Figure 2(c) and (d). We again set s_i to be the vertex of object B closest to p_{i-1}. And once more $|s_i, p_{i+1}|$ intersects the side of the object B in direction from p_{i-1} to p_i. □

Theorem 2. *An optimal obstacle-hugging route is* canonical*, that is for each optimal route there is exactly one topologically equivalent optimal obstacle-hugging route.*

Proof. Suppose to the contrary we have two optimal obstacle-hugging routes R and R' from p_1 to p_m which are topologically equivalent. As they are topologically equivalent they must have the same number of segments: let p_i and p'_i be the ith points in route R and R' respectively. We prove by induction that $p_i = p'_i$ for $i = 1, \cdots, m$. As they both start from the same connection point we have that $p_1 = p'_1$. Now assume that $p_k = p'_k$ for $k = 1, \cdots, i$. If $i + 1 = m$ then as R and R' have the same end point so $p_{i+1} = p'_{i+1}$. Otherwise assume that $p_{i+1} \neq p'_{i+1}$. As each route is obstacle-hugging the next segments $|p_{i+1}, p_{i+2}|$ and $|p'_{i+1}, p'_{i+2}|$ in each route must have a support vertices s_{i+1} and s'_{i+1}. But this implies that there is an object between the two routes. Thus there is no sequence of edge moves that can make these two routes the same since they travel on different sides of at least one object and so this contradicts the assumption they are topologically equivalent. □

3 The 1-Bend Visibility Graph

While we could compute object-hugging routes using the orthogonal visibility graph by modifying the A* algorithm to prune non-object-hugging routes we can also compute them using a different kind of visibility graph. This follows from the observation that we can regard the support vertices $s_2, \ldots, s_{m-2}$ for a canonical optimal object-hugging route $p_1, \ldots, p_m$ as "split points" between a sequence of 1-bend visibility edges between object vertices, apart from the first and last segments which go to the start and end connector point respectively. Thus when we search for an optimal route we can search in a 1-bend visibility graph and build the a canonical route as $p_1, s_2, s_3, \ldots, s_{m-2}, p_m$. This ensures we will only consider one possible topologically equivalent route.

The 1-bend visibility graph is a directed graph where nodes are combination of connector points and corners of rectangles. The 1-bend visibility graph can be constructed as follows. Let I be the set of *interesting points* (x, y) in the diagram, i.e., the corners of the rectangular objects and the connector points.

1. Generate the *interesting horizontal segments* $H_I = \{|(l,y),(r,y)|$ where $(x,y) \in I$ and r is the biggest value where no object overlaps $|(x,y),(r,y)|$ and l is the least value where no object overlaps $|(l,y),(x,y)|$.
2. Generate the *interesting vertical segments* $V_I = \{|(x,b),(x,t)|$ where $(x,y) \in I$ and t is the greatest value no object overlaps $|(x,y),(x,t)|$ and b is the least value no object overlaps $|(x,b),(x,y)|$.
3. Compute the 1-bend visibility graph by intersecting all pairs of segments from H_I and V_I. Suppose we have intersecting segments $|(x,b),(x,t)| \in H_i$ and $|(l,y),(r,y)| \in V_i$ we add an edge from each vertex point (x,h) of an object o_1 on the horizontal segment where the segment $|(x,h),(x,y)|$ intersects the side of o_1 in the direction (x,h) to (x,y) to each vertex point (v,y) on an object o_2 in the direction (x,y) to (v,y) where the segment $|(x,y),(v,y)|$ only intersects the object at (v,y).
   Similarly we add an edge from each vertex point (v,y) of an object o_1 on the vertical segment in the direction (v,y) to (x,y) where $|(v,y),(x,y)|$ intersects the side of o_1 to each vertex point (x,h) on an object o_2 in the direction (x,y) to (x,h) where $|(x,y),(x,h)|$ only intersects the object at (x,h).
   The edges to and from connection points with directions are created similarly but there is no requirement for intersection/nointersection with any object.

For example the directed edges from the bottom left corner (star) of C going upwards in the visibility graph of Figure 1(a) are shown in Figure 1(b) as circles with directions given by arrow. There are edges from the connection point on A in the up direction go to each of circled nodes except the one on A itself. Note that the graph is *not* symmetric! The only edge from the top side of shape D to shape C goes from the diamond node in the right direction, to the diamond node on C in the down direction.

Theorem 3. *The orthogonal visibility graph can be constructed in $O(n^2)$ time for a diagram with n objects using the above algorithm. It has $O(n)$ nodes and $O(n^2)$ edges.*

Proof. The interesting horizontal segments can be generated in $O(n \log n)$ time where n is the number of objects in the diagram by using a variant of the line-sweep algorithm from [6, 7]. Similarly for the interesting vertical segments. The last step takes $O(n^2)$ time since there are $O(n)$ interesting horizontal and vertical segments. It follows from the construction that it has $O(n)$ nodes and $O(n^2)$ edges. □

Construction of a path from connector point p_1 to p_m starts from p_1 and constructs a path for each possible (feasible) direction, to reach p_m in any direction. The best path (according to the penalty function p) is returned for final visual route computation.

Consider the construction of the path from the connection point on A leaving vertically to the connection point on B entering vertically. The only nodes adjacent to the initial connection point are the circled nodes of B, C and D. Using A* search and the admissable heuristic described in [1], the node on C is preferable, and then we find an optimal route to the connection point on B (the bottom route in Figure 1). The remaining nodes are fathomed by the heuristic. Effectively we find the route with no search. Contrast this with the usual approach [1] where every node on the three paths in Figure 1(a) needs to be visited as well as others!

Table 1. Evaluation of 1-bend and orthogonal visibility graphs for several real-world diagrams. We give total routing time, construction time and number of edges in visibility graph, and the average time and number of steps in the search of each connector path. Times are in milliseconds.

	1-bend visibility graph					Orthogonal visibility graph				
	Total	Construction		Routing (avg)		Total	Construction		Routing (avg)	
Diagram	Time	Time	$\|E\|$	Time	Steps	Time	Time	$\|E\|$	Time	Steps
v185e225	197	41	35K	0.3	4	216	34	28K	0.4	34
v508e546	867	191	196K	0.5	3	1,964	185	136K	2.6	28
v4330e2755	180,831	14,070	25.9M	42	178	1,829,431	6,741	3.0M	645	14,859

4 Evaluation and Conclusion

Theoretically, searching for optimal obstacle-hugging connector routes over the 1-bend visibility graph should be considerably faster than finding optimal routes in the orthogonal visibility graph. This is because we don't explore topologically equivalent routes, but also since the optimal route is $O(n)$ in length where n is the number of obstacles rather than the $O(n^2)$ length in the orthogonal visibility graph.

We compared the performance of a prototype implementation of the 1-bend visibility graph approach against the orthogonal visibility graph implementation in the **libavoid** C++ routing library. The test machine was a 2012 MacBook Pro with a 2.3 GHz Intel Core i7 processor and 16GB of RAM. As shown in Table 1, while 1-bend visibility graphs are larger and take longer to build, routing over them is significantly faster. For a very large diagram like our final example, use of a 1-bend visibility graph speeds up connector routing by a factor of ten!

We have presented a canonical connector route and a new kind of visibility graph that significantly improves the speed of orthogonal connector routing. We plan to include the new approach in our widely used connector routing library **libavoid**.

Acknowledgments. We acknowledge the support of the ARC through Discovery Project Grants DP0987168 and DP110101390.

References

1. Wybrow, M., Marriott, K., Stuckey, P.J.: Orthogonal connector routing. In: Graph Drawing 2009. Volume 5849 of LNCS., Springer (2010) 219–231
2. Wybrow, M., Marriott, K., Stuckey, P.J.: Incremental connector routing. In: Graph Drawing 2005. Volume 3843 of LNCS., Springer (2006) 446–457
3. Miriyala, K., Hornick, S.W., Tamassia, R.: An incremental approach to aesthetic graph layout. In: Proc. 6th Intl. Workshop on Computer-Aided Software Engineering, IEEE (1993) 297–308
4. Lee, D., Yang, C., Wong, C.: Rectilinear paths among rectilinear obstacles. Discrete Applied Mathematics **70**(3) (1996) 185–216
5. Lengauer, T.: Combinatorial Algorithms for Integrated Circuit Layout. John Wiley & Sons, Inc., New York, NY, USA (1990)
6. Dwyer, T., Marriott, K., Stuckey, P.J.: Fast node overlap removal. In: Graph Drawing 2005. Volume 3843 of LNCS., Springer (2006) 153–164
7. Dwyer, T., Marriott, K., Stuckey, P.J.: Fast node overlap removal—correction. In: Graph Drawing 2006. Volume 4372 of LNCS., Springer (2007) 446–447