The Planar Pin Assignment and Routing Problem (PPARP) is NP-complete

Joel Darnauer

UCSC-CRL-95-41
August 1, 1995

Baskin Center for
Computer Engineering & Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

ABSTRACT

Many package routing problems involve routing one set of pins to another set of pins in a single layer without regard to how the pins in each set are connected. We call this type of problem a *Planar Pin Assignment and Routing Problem (PPARP)*. Unlike general routing problems which correspond to the NP-complete multi-commodity flow problem, the pin distribution problem seems to have a natural formulation as a single-commodity flow problem, which suggests that it can be solved efficiently. In this paper, we show that the problem is surprisingly NP-complete.

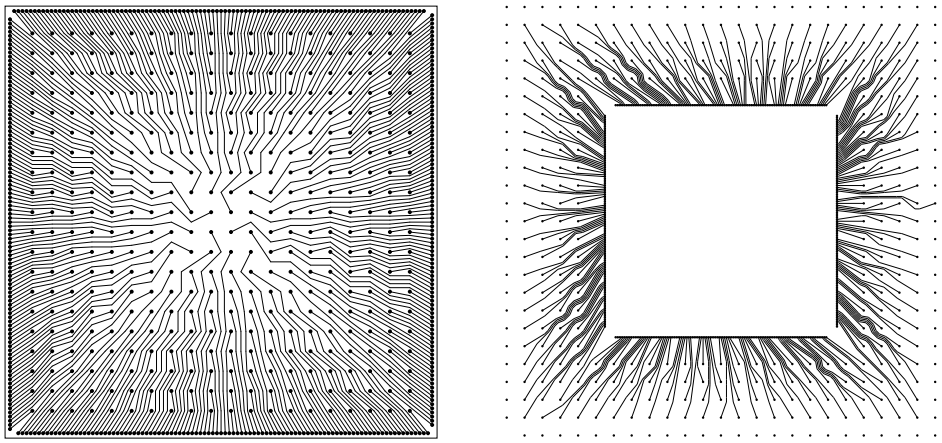

Figure 1.1: Escape routing and PGA routing are examples of PPARP

1 Problem Definition

Package designers often have routing problems where they must connect one set of pads to another set of pads and must come up with a pin assignment and routing. Escape routing and PGA routing (figure 1.1) are instances of this type of problem. This task is repetitious and a good target for automation. Unfortunately, most routing programs cannot handle pin assignment.

Previous work has addressed the topic of generating fan-in and fan-out patterns for arrays of pads and rings of pads[DD94, YD95]. The previous approaches have three main deficiencies:

1. They rely on the symmetry of the problem to generate solutions and cannot cope with missing, skewed, or arbitrarily placed bumps.
2. Even in these very symmetric cases, they fail to intelligently balance the interconnect and can produce designs with cuts that are $\sqrt{2}$ times more dense than the lower bound. It is not known however, whether this bound is attainable.
3. They may not be able to handle the case where the number of pads in the two sets is unequal.
4. They do not take into account the presence of obstacles or the effect of pad shape on the channel size.

We wish to develop an algorithm that solves the general problem without any of these deficiencies. Specifically, we wish to solve the following problem, which we call Detailed Planar Pin Assignment and Routing (D-PPARP).

1.1 Detailed PPARP

Definition 1: *Detailed Planar Pin Assignment and Routing(D-PPARP)*

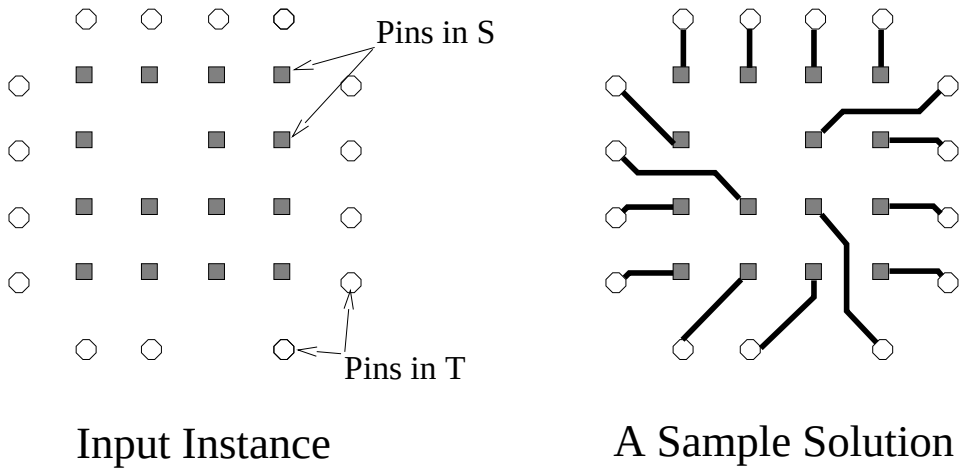

Figure 1.2: An instance of D-PPARP and a possible solution

INPUT:

b: a polygon representing the bounds of the routing area.
S: a set of polygons for one class of pins.
T: a set of polygons for the other class of pins.
U: a set of polygons representing obstacles.
w: a positive integer for the minimum wire width.
s: a positive integer for the minimum wire spacing.
D: a set of feasible wire directions (Manhattan, Octilinear, etc.)

S, T, U are non-intersecting polygons inside G.
Without loss of generality $|S| \leq |T|$.

OUTPUT:

R_{geom}*: a set of line segments representing the routing.*
R_{geom} *must satisfy the design rules* (w, s, D)
and connect each element in S to a unique element of T.

Let the vertices of a polygon g be denoted by $p(g)$ and the vertices of a set of polygons G be denoted by $p(G)$. The coordinate system in which the routing area is defined will be the lattice of integer pairs. The points of this integer lattice will be used to define the vertices and endpoints of polygons and lineal wire segments respectively. Although the endpoints of these lines must be integral, particular interior points on the lines will have continuous coordinates, we observe that this is not a difficulty in most CAD and graphics systems, and from now on, ignore the subtleties of the representation of points and lines. Assuming that each coordinate can be represented with a constant amount of space which is independent of the problem size, we will say that the size of an instance of D-PPARP is $N = p(S) + p(T) + p(U) + p(b)$.

Figure 1.2 shows an instance of PPARP and a possible solution. It should be apparent from the definition that an instance of PPARP can have many solutions, or no solutions. It should also be clear that the pin assignments are implicitly represented in the routing information.

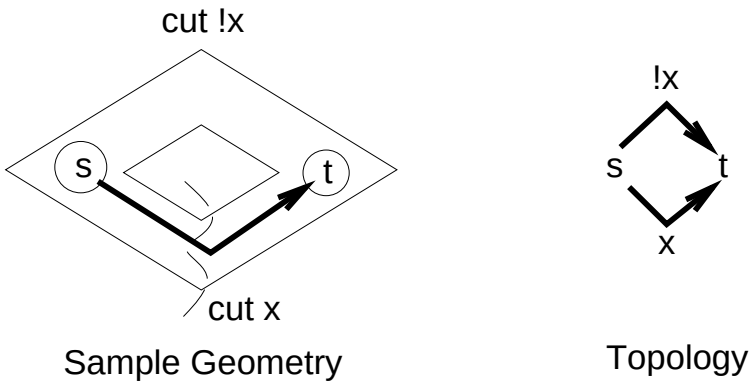

Figure 2.1: 0-1 variables can be represented by flow in selected channels

2 PPARP is NP-complete

The basic problem we will encounter is that the constraints may allow prevent some set of cuts from being satisfied simultaneously. We try the following constructive reduction from SATISFIABILITY using the method of components[GJ79]

Given an instance of satisfiability in the form of a boolean circuit, we can find a PPARP instance such that a feasible topological routing exists if and only if the circuit is satisfiable. The assignment of (0-1) to variables will correspond to the existence of wire within certain channels in the routing. We will position sources and sinks alternately along this path so that existence of flow will correspond to the clockwise or counterclockwise direction of flow as well. (Figure 2.1).

By allowing cycles to touch at certain locations (called "bottlenecks"), we can use the cut capacity constraints to enforce a general rule of the form $x + y < 1$. (Figure 2.2). By using this limiting capability, we can control the direction of flow in some cycles based on the flow in other cycles. Figure 2.3 shows how to construct rudimentary boolean operations using cycles of this type. Clearly we have enough components now to construct any planar boolean circuit. Finally, using the construction in figure 2.4 we can propagate the values of variables across cycles. This allows us to realize even non-planar boolean circuits as a physical flow of wires. If there is a feasible solution to the flow problem for this circuit, then there is a satisfying assignment of inputs for the circuit. Since satisfiability is NP-complete, then so is PPARP.

It is instructive to note that restricting PPARP in some ways has no impact on its NP-completeness. For example, allowing fractional-width wires allows solutions to circuits that would otherwise be infeasible. For example, we could require that flow across cuts always flow in the same source-sink direction, but since it is always possible to arrange for the flow in a bottleneck to always travel in one direction, this does not help. The critical factor seems to be the fact that the two cuts in the bottleneck cross. If this can be eliminated, then the construction fails.

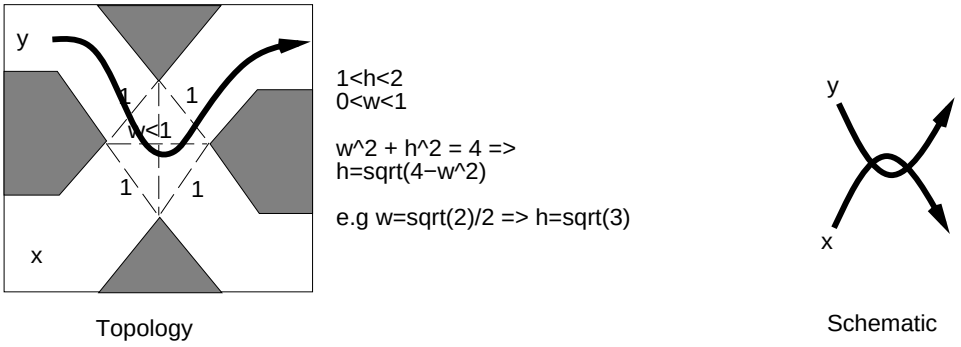

Figure 2.2: Intersecting channels allow constraints to propagate from one variable to another

3 Conclusions

No polynomial algorithm can guarantee finding an optimal solution to PPARP. However, the bidirected flow in the triangulation suggests an efficient heuristic based on min-cost flow, that may suffice for many problem instances.

Acknowlegements

Thanks to Marco Yu and David Staepelaere who read the early drafts of this paper and provided criticism and suggestions. I am also indebted to Phokion Kolaitis, who taught me about linear programming and NP-completeness, and to Martine Schlag, who pointed out the crucial difference between single-commodity and multi-commodity flow.

References

[DD94] Joel Darnauer and Wayne Dai. Fast pad redistribution from periphery io to array io. In *IEEE Multichip Module Conference*, 1994.

[GJ79] Garey and David S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-completeness*. W.H. Freeman, 1979.

[YD95] Man-fai Yu and Wayne Wei-Ming Dai. Single-layer fanout routing and. Technical Report UCSC-CRL-95-18, Computer Engineering, UC Santa Cruz, June 28 1995.

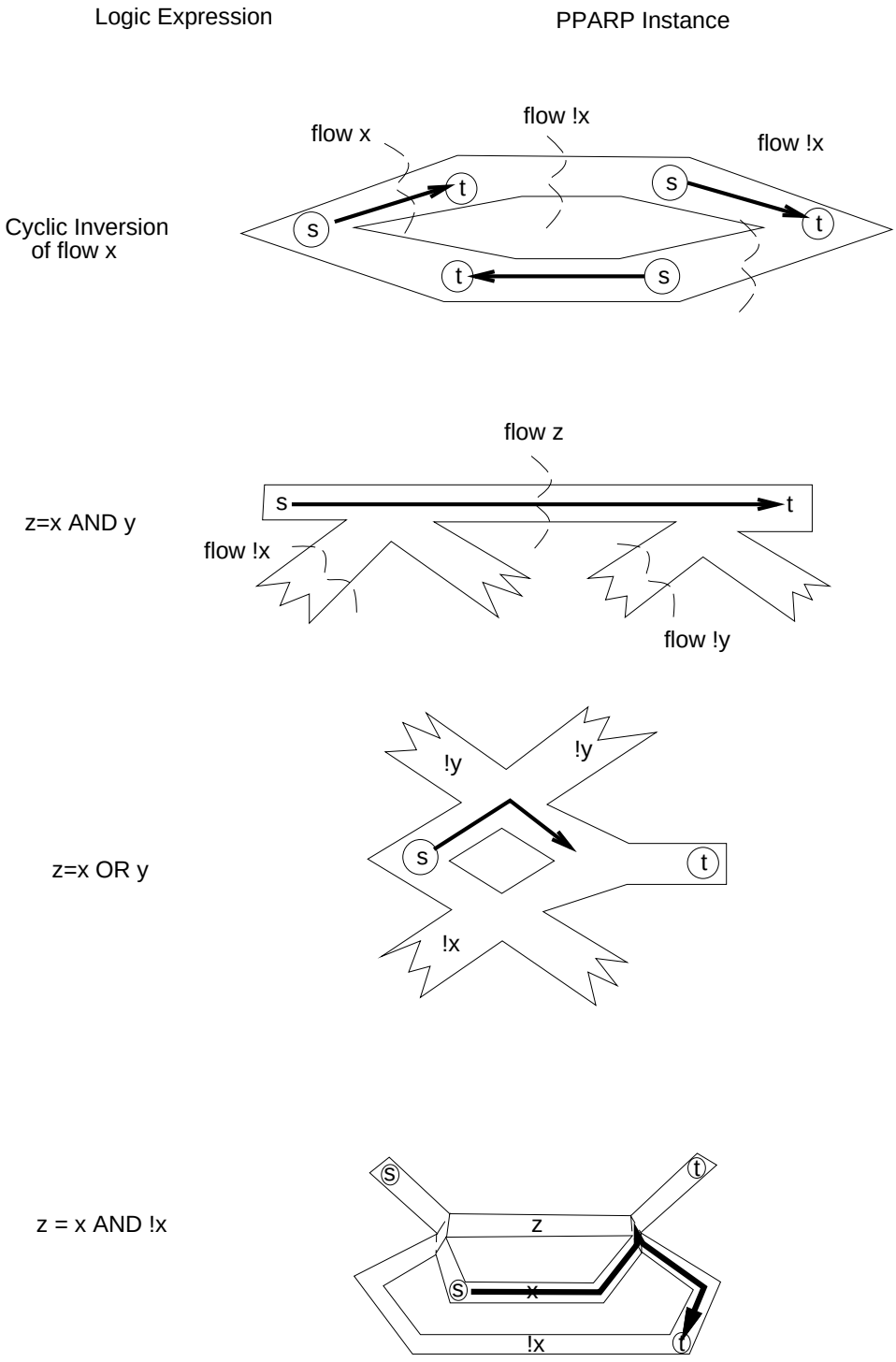

Figure 2.3: Intersections and flows can be used to construct rudimentary gates

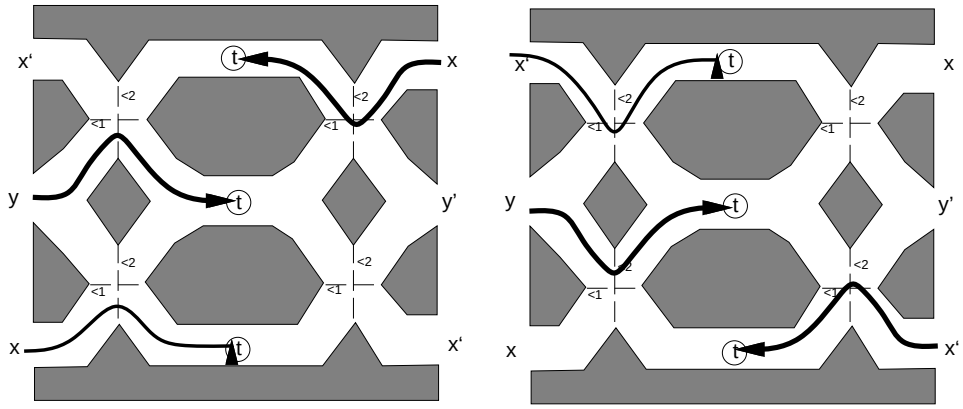

Figure 2.4: A special construction allows us to cross cycles over one another to construct non-planar circuits.