Purdue University
Purdue e-Pubs

ECE Technical Reports | Electrical and Computer Engineering

1-5-1995

A Difficult Channel Routing Generator

Heng Yi Chao
Purdue University School of Electrical Engineering

Mary P. Harper
Purdue University School of Electrical Engineering

Follow this and additional works at: http://docs.lib.purdue.edu/ecetr

Chao, Heng Yi and Harper, Mary P., "A Difficult Channel Routing Generator" (1995). *ECE Technical Reports.* Paper 108.
http://docs.lib.purdue.edu/ecetr/108

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

A Difficult Channel Routing Generator

Heng-Yi Chao
Mary P. Harper

TR-EE 95-1
JANUARY *1995*

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907-1285

A Difficult Channel Routing Generator

Heng-Yi Chao and Mary P. Harper
School of Electrical Engineering
1285 Electrical Engineering Building
Purdue University
West Lafayette, IN 47907-1285
hengyi@ecn.purdue.edu, harper@ecn.purdue.edu

January 5, 1995

Abstract

For channel routing problems, Deutsch's examples were used extensively as benchmarks for testing new algorithms. However, it is also extremely important to test the performance of channel routing algorithms on a wider variety of difficult examples. In this paper, we present a random channel routing generator which can generate difficult channel routing instances of arbitrary size.

Keywords: **VLSI, CAD,** channel routing, lower bound, **DAG,** transitive reduction.

1 Introduction

Channel routing plays a central role in the physical design of VLSI chips. To meet the increasing demands of functionality, the number of transistors on a chip today has increased considerably. For example, a new MPEG2 decoder chip consists of 700,000 transistors on an area of 87.23 mm^2. Most layout systems begin with placing modules on a chip, and then wiring together terminals that should be electrically connected on different modules. An efficient approach for solving the wiring problem is to heuristically partition the chip into a set of rectangular channels, and then route each channel separately. This effectively divides a difficult problem into smaller subproblems that can be conquered more easily. Because of its importance in layout automation, the channel routing problem (CRP) has been studied extensively [4, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19].

For channel routing problems, Deutsch's examples [4] were used extensively as benchmarks for evaluating the performance of new algorithms, especially the so-called Deutsch's Difficult Example (DDE) [4, 12, 19]. In this paper, we develop a random channel routing generator which can generate difficult channel routing instances of arbitrary size. This research is motivated by the following facts:

- Because the benchmarks represent an extremely small subset of real problems, they may not represent the complexity that exists in the majority of today's and the future's designs.
- The number of transistors on a chip has increased considerably. Testing on the traditional benchmarks may not be sufficient for evaluating the performance of channel routing algorithms.
- It is possible to design an algorithm that works well for known benchmarks, but not other examples. As our channel routing generator can randomly generate difficult instances of arbitrary size, it will fully test these algorithms.

  Many examples discussed in the literature are not available for testing.

In [2], we have used this random generator to generate some difficult instances to test our new lower bound alogrithm for channel routing. These examples have revealed the importaiice of considering the interaction of constraints when determining a tighter lower bound for channel routing.

In Section 2, we introduce the model as well as the major constraints on a CRP. Then we present the difficult CRP generator in Section 3 and discuss some random examples in Section 4.

2 The Restricted Manhattan Model

We consider the two-layer *restricted* Manhattan model [10, 12,171. Although the three-layer process is available, the two-layer model is still attractive for the following reasons:

- The yield is higher for the two-layer process.
- a The three-layer process is much more expensive than the two-layer process.
- If the product is time critical on the market, the two-layer model provides a faster way of bringing the product to the market.

A two-layer *channel* is a gridded rectangular area on a chip consisting of a metal layer running *horizontally* and a polysilicon layer running *vertically* (or vice versa). A wire in the horizontal layer is called a *track* and a wire in the vertical layer is called a *column*. There are fixed terminals on the top and bottom sides, and floating terminals on the left and right sides of the channel. Each set of points that need to be electrically connected is called a *net*. A net can connect terminals from the top and bottom of the channel and can exit the channel at the left and right sides. Wires of a net on different layers are connected by *vias*. An example of a CRP is shown in Figure 1.

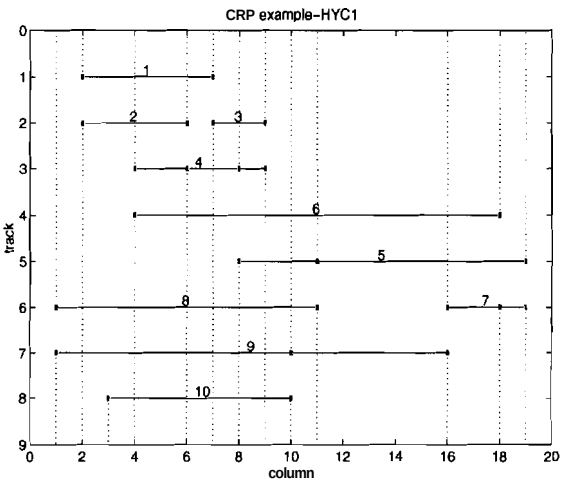

Figure 1: A CRP example HYC1, where the small boxes represent the vias.

The objective of a channel routing problem is to assign a set of nets to a rninimum number of tracks such that no nets overlap on any layer. Let S^* denote the optimal solution (minimum number of tracks required). Let l_i be the leftmost and r_i be the rightmost column of net i. A net i is said to span the c-th column if $l_i \leq c \leq$ r;. The interval $[l_i$, r;] is called the *span* of net i and is denoted as $span_i$. In this paper, our objective is to generate CRPs that can be routed without

doglegs, that is, the horizontal segment of a net cannot split. This wiring style has the advantage that the number of vias is minimal [11, 12, 17].

2.1 Horizontal Constraints and Vertical Constraints

There are two major constraints on a CRP, the horizontal constraints and the vertical constraints.

The constraints that two nets cannot overlap on the horizontal layer are called the horizontal constraints. There is a horizontal constraint between net i and net j if and only if their spans overlap. Let d; be the number of nets that span the i-th column, $d_{max} := \max\{d_i : \text{i is a column}\}$ is called the density of the CRP. Clearly, d_{max} is a lower bound on S* because nets spanning the same column cannot be assigned to the same track.

The constraints that two nets cannot overlap on the vertical layer are called the vertical constraints. If net i connects to the c-th column in the top row and net j connects to the c-th column in the bottom row, $i \neq j$, then net i must be assigned to a track higher than net j. In this case, we say that there is a vertical constraint from i to j, denoted $i \prec j$. Vertical constraints define a partial ordering between nets. Vertical constraints are often represented by a directed graph, the vertical constraint graph (VCG), where vertices represent the nets and arcs represent the vertical constraints. In this paper, vertical constraints are also represented by a bit matrix vc such that $vc(i, j) = 1$ if and only if there is a vertical constraint from i to j.

Note that vertical constraints are transitive, i.e., if $i \prec j$ and $j \prec k$ then $i \prec k$. Hence, if there is a path from i to j in the VCG, then i must be assigned to a track higher than j. For a CRP to be routed without doglegs, the VCG must be acyclic. The length of a path in the VCG is the number of vertices on the path. Nets on the same path cannot be assigned to the same track. Let v_{max} be the length of a longest path in the VCG. Clearly, v_{max} is a lower bound on S*.

Note that an initial vertical constraint implies a horizontal constraint, that is, if there is a vertical constraint from i to j, then there is a horizontal constraint between i and j. For all examples but DDE in [4], $d_{max} \geq v_{max}$. However, these lower bounds (d_{max} and v_{max}) may not constitute a tight lower bound on S* because of the interaction of constraints [2]. This fact was not reflected by the known benchmarks. To evaluate the performance of algorithms in channel routing, it is important to test them on examples with varying constraints.

2.2 Representation

Because the examples in [4] were shown as physical layouts, it is very difficult to recognize the connections in the networks. Hence, different channel specifications may be used even though the same problem is referenced. We prefer a more formal description of the channel specification, which has been used in [12, 16]. For example, Deutsch's difficult example obtained from [8, 12] is shown in Table 1. There are two rows, the *top* row and the *bottom* row, with a net number in each column indicating that the net is connected to that column. A 0 in a column indicates that there is no net connected to the column in that row. We include two extra rows, the *left* and *right* rows, in which the numbers represent the nets that exit to the left and right side of the channel, respectively.

Table 1: Deutsch's difficult problem.

problem		column														
DDP	top	2	4	6	8	10	11	13	3	9	16	5	17	11	5	14
		14	7	12	17	19	1	20	21	23	24	0	16	10	3	11
		25	0	26	11	26	11	0	27	28	11	3	9	16	30	27
		5	31	1	5	1	20	32	23	24	0	9	1	20	29	23
		24	0	3	8	30	38	28	19	6	40	27	35	41	42	6
		19	34	43	30	8	31	43	39	46	36	46	47	48	31	0
		24	23	45	20	1	51	0	40	39	40	39	0	8	30	50
		54	0	0	55	49	19	6	0	47	42	47	42	0	53	58
		6	19	49	50	30	8	60	62	59	54	55	54	56	63	55
		65	0	66	68	66	68	0	60	68	0	46	44	46	44	0
		69	0	55	58	55	58	0	64	71	0	72	63	72	63	0
		57	62	54	70	67	55	61	63	68						
	bottom	3	5	7	9	5	12	14	15	7	12	14	7	4	13	8
		6	15	18	14	8	6	11	22	21	0	18	16	18	16	0
		8	6	26	11	0	24	23	25	20	1	29	0	22	3	22
		3	0	0	9	2	9	2	0	32	23	33	19	6	8	30
		27	34	35	36	37	39	31	39	35	38	31	8	30	37	41
		19	6	44	45	0	33	31	33	31	0	27	35	36	48	49
		31	39	46	47	50	52	20	53	24	0	47	39	0	24	51
		20	52	20	52	23	8	30	50	56	0	0	57	49	19	6
		6	19	49	59	0	0	61	50	30	8	55	0	24	64	20
		52	0	67	68	63	55	24	52	20	69	24	0	46	62	63
		68	0	24	65	20	52	0	70	60	62	54	63	0	24	71
		20	52	67	0	0	0	0	0	0						
	left	1														

3 The Difficult CRP Generator

Lee and Leong [10], and Wang and Lee [17] have tested their algorithmson some randomly generated CRPs in addition to Deutsch's examples. However, d_{max} seems to be a dominant lower bound for all of these random examples.

If we randomly assign a net to each column, it is very likely that the VCG would not be acyclic with the spans of many nets overlapping (this probably is the main reason why the CRP generators in [10, 17] cannot generate "good" (difficult) CRP examples). In this section, we present a difficult CRP generator which is able to randomly generate difficult CRP examples of arbitrary size (number

```
Algorithm MakeVC (n,p)
  1. ∀i,j vc(i,j) := 0
  2. for i = 1 to n − 1
  3.     for j = i + 1 to n
  4.         if random() < p then
  5.             vc(i,j) := 1
  6. end end
```

Figure 2: MakeVC, an algorithm to randomly generate the vertical constraints, where n is the number of nets and p is the probability for an arc to occur in the VCG.

of nets). Without loss of generality, we generate CRPs in which each net only has terminals from either the top or bottom sides. By doing this, we reduce the influence of the horizontal constraints.

To be a valid channel specification that can be routed without doglegs, the VCG must be acyclic. An algorithm, **MakeVC**(n, p), that randomly generates the vertical constraints (represented by vc) for a CRP is shown in Figure 2, where **random** is a random number generator which generates a real number in $(0, 1)$, n is the number of nets, and $0 < \mathrm{p} < 1$ is the probability that an arc occurs in the VCG. Obviously, the VCG represented by vc is acyclic since each arc is directed from i to j only if $\mathrm{i} < \mathrm{j}$.

Given a vc generated by MakeVC and cmax (the maximal number of columns allowed), **MakeCol** (shown in Figure 3) assigns columns for each net. We say that a column c is *free* if there is no nets assigned to that column, i.e., $top(c) = 0$ and $bottom(c) = 0$. Note that cmax may need to be increased when there are no free columns.

Initially all nets are colored **white;** a net is colored **gray** the first time that it is connected to a terminal; a net is colored **black** when it is connected to more than one terminal. Note that an initial vertical constraint implies a horizontal constraint. To reduce the dominance of the horizontal constraints on the CRP, we want to minimize the span of each net when we assign columns to it. **PickCol**(a, b) randomly picks a free column in the interval [a,b]. If there is no free column available in the interval, 0 is returned. If there is an arc (i,j) in the VCG, a column is picked depending on the color of i:

- If $color(i)$ is white or gray, we try to pick a column in $[1, \text{cmax}]$. If **PickCol** returns 0, cmax is increased by one, and c is set to cmax.
- If $color(i)$ is black, there are two cases: if span; and $span_j$ overlap, we find the interval $[\text{L}, R]$ as the intersection of span; and $span_j$; otherwise, we find the interval $[L, \text{R}]$ as the interval between span; and $span_j$. Then we try to pick a free column from $[\text{L}, \text{R}]$ or $[1, \text{L}]$ or $[\text{R}, \text{cmax}]$ in that order.

If **PickCol** successfully returns a free column c, the program goes to Step 24 which assigns i to $top(c)$ and assigns j to $bottom(c)$. Then the color of j and span; and $span_j$ must be modified properly.

Algorithm CRP-generator, shown in Figure 4, randomly generates difficult CRP examples of arbitrary size. It first calls **MakeVC** to generate the vertical constraints, computes the transitive reduction [1, 3, 5] vc^- of vc (again to reduce horizontal constraints), and then calls **MakeCol** to assign columns for each net. Finally, it assigns columns to the nets which have no vertical constraints or have only a single vertical constraint with another net in order that each net is connected to at least two terminals. By adjusting the parameters n, p and cmax, various difficult CRPs can be generated.

4 Experimental Results

We have used CRP-generator to create several benchmark examples, which are listed in Tables 3 and 4. The traditional lower bounds d_{max} and v_{max}, and optimal two-layer dogleg-free solutions (if known) for these examples are listed in Table 2. For instance, an optimal routing of HYC1 is shown in Figure 1. For HYC2, HYC3, HYC5, HYC6, and HYC7, $d_{max} > v_{max}$; for HYC4 and HYC8, $v_{max} > d_{max}$. For each of these examples, the optimal solution differs significantly from the traditional lower bounds, which indicates that is is very important to consider the interaction of the horizontal constraints and vertical constraints for channel routing problems.

5 Conclusion

We have presented a random CRP generator which can generate difficult CRP instances of arbitrary size. This algorithm generates CRP instances that are guaranteed to be routed without doglegs and should prove to be a useful tool for testing the performance of new algorithms for channel routing.

```
Algorithm MakeCol (vc,cmax)
1.  for i = 1 to n − 1
2.  for j = i + 1 to n
3.      if vc(i, j) = 1 then
4.          switch color(i)
5.          case white:
6.              c := PickCol(1,cmax)
7.              if c = 0 then cmax := cmax + 1 and c := cmax
8.              color(i) := gray
9.          case gray:
10.             c := PickCol(1,cmax)
11.             if c = 0 then cmax := cmax + 1 and c := cmax
12.             color(i) := black
13.         case black:
14.             if span_i and span_j overlap then [L, R] := span_i ∩ span_j
15.             else [L, R] := the interval between span_i and span_j
16.             c := PickCol(L, R)
17.             if c > 0 then goto Step 24
18.             c := PickCol(1, L)
19.             if c > 0 then goto Step 24
20.             c := PickCol(R, cmax)
21.             if c > 0 then goto Step 24
22.             cmax := cmax + 1 and c := cmax
23.         end
24.         top(c) := i, bottom(c) := j
25.         if color(j) = white then color(j) := gray
26.         else color(j) := black
27.         modify span_i and spanj
28. end end
```

Figure 3: MakeCol, an algorithm to assign columns for each net.

```
Algorithm CRP-generator (n, p, cmax)
call MakeVC(n, p)
compute the transitive reduction vc⁻ of vc
call MakeCol(vc⁻, cmax)
i := 1
while i ≤ n
    if color(i) = black then i := i + 1
    if color(i) = gray then
        pick a free column c that is closest to span_i
        if c = 0 then
            cmax := cmax + 1 and c := cmax
        color(i) := black
    if color(i) = white then
        c := PickCol(1, cmax)
        if c = 0 then
            cmax := cmax + 1 and c := cmax
        color(i) := gray
    if random() > 0.5 then top(c) := i
    else bottom(c) := i
    modify span_i
end
```

Figure 4: A random difficult CRP generator, where n is the number of nets, ***p*** is the probability for an arc to occur in the VCG, and ***cmax*** is the maximal number of columns allowed.

Table 2: Lower bounds and optimal solutions for some benchmark CRPs.

problem	#nets	#cols	d_{max}	v_{max}	optimal
DDE	72	174	19	23	28
HYC1	10	19	7	7	8
HYC2	10	20	8	7	9
HYC3	15	24	11	8	15
HYC4	50	89	20	22	?
HYC5	50	149	35	32	45
HYC6	70	115	50	42	66
HYC7	70	144	39	19	?
HYC8	70	104	21	43	62

As the **CRP-generator** can generate problems of arbitrary size, problems become intractable quickly as n increases. Hence, exhaustive search methods [9, 10, 17] may not be feasible. For some examples generated by this **CRP-generator**, the optimal solution differs significantly from the traditional lower bounds. It reveals the importance of considering the interaction of constraints when developing efficient algorithms for channel routing.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Company, San Francisco, CA, 1976.

[2] H. Y. Chao. *Finding a Tighter Lower Bound for Optimization Problems with Capacity and Precedence Constraints.* PhD thesis, Purdue University, 1994.

[3] H. Y. Chao and M. P. Harper. Minimizing redundant dependencies and interprocessor synchronizations. In *Proceedings of the Sixth IASTED/ISMM International Conference on Parallel and Distributed Computing and Systems,* pages 294–298, October 1994.

[4] D. N. Deutsch. A dogleg channel router. In *Proceedings of the 13th Design Automation Conference,* pages 425–433, 1976.

[5] D. Gries, A. J. Martin, Jan L. A van de Snepscheut, and J. T. Udding. An algorithm for transitive reduction of an acyclic graph. *Science of Computer Programming,* 12:151–155, 1989.

[6] S. Haruyama, D. F. Wong, and D. S. Fussell. Topological channel routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 11(10):1177–1197, October 1992.

[7] T. T. Ho. A density-based greedy router. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(7):973–981, *1993*.

[8] T. T. Ho, S. S. Iyengar, and S. Q. Zheng. A general greedy channel routing algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(2):204–211, February *1991*.

[9] B. W. Kernighan, D. G. Schweikert, and G. Persky. **An** optimum channel-routing algorithm for polycell layouts of integrated circuits. In *Proceedings of the 10th Design Automation Workshop*, pages *50–59, 1973*.

[10] K. K. Lee and H. W. Leong. **An** improved lower bound for channel routing problems. *IEEE International Symposium on Circuits and Systems*, 4:11–14, June *1991*.

[11] Z. M. Lin. **An** efficient optimum channel routing algorithm. In *IEEE Proceedings of the SOUTHEAST-CON '91*, volume *2*, pages *1179–1183, 1991*.

[12] T. Ohtsuki. *Layout Design and Verification*, volume *4*. Elsevier Science Publishers B.V., Amsterdam, *1986*.

[13] R. K. Pal, S. P. Pal., and A. Pal. A new lower bound for channel routing. In *Proceedings of the IEEE Region 10 Conference on Computer, Communication, Control and Power Engineering*, pages 507–510, October 1993.

[14] J. Reed, A. Sangiovanni-Vincentelli, and M. Santomauro. A new symbolic channel router: YACR2. *IEEE Transactions on Computer-Aided Design*, CAD-4(3):208–219, July *1985*.

[15] R. L. Rivest and C. M. Fiduccia. A greedy channel router. In *19th Design Automation Conference*, 1981.

[16] S. Soh, S. Rai, and R. Mehrotra. **An** efficient approach for channel routing in VLSI. *Computers Electrical Engineering*, 17(2):105–112, *1991*.

[17] J. S. Wang and R. C. T. Lee. An efficient channel routing algorithm to yield an optimal solution. *IEEE Transactions on Computers*, 39(7):957–962, July *1990*.

[18] U. Yoeli. A robust channel router. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(2):212–219, February *1991*.

[19] T. Yoshimura and E. S. Kuh. Efficient algorithms for channel routing. *IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems*, CAD-1:25–35, *1982*.

Table 3: Random CRP examples - set I.

problem		column
HYC1	top	8 1 0 4 0 2 1 4 3 9 5 0 0 0 0 7 0 6 5
	bottom	9 2 1 0 6 0 4 3 5 4 1 0 8 0 0 0 0 9 0 7 7
HYC2	top	6 8 0 3 0 3 9 0 3 0 1 2 7 4 0 5 0 2 0 1
	bottom	9 9 1 0 5 0 6 1 0 0 7 0 4 3 9 8 0 8 0 4 0 3
HYC3	top	l o 7 6 1 1 2 2 3 7 4 1 3 6 8 5 5 9 11 12 8 13 13 0 0
	bottom	11 9 7 3 5 4 6 8 1 1 7 2 6 1 0 1 0 8 6 12 12 13 9 14 15 14 15
HYC4	top	1 4 4 2 3 2 4 1 6 3 2 0 4 1 2 1 5 1 4 15 9 10 9 9 9 14 11 15 12 10 23 13 17 13 16 18 18 21 24 8 5 12 20 19 19 22 16 21 11 7 23 18 23 8 24 25 26 26 27 28 29 30 31 31 32 33 33 34 35 35 36 37 37 37 38 38 38 39 39 39 39 40 40 41 41 41 42 43 44 44 46 47 0
	bottom	3 9 14 4 4 5 18 2 8 10 22 17 15 19 32 20 32 12 29 27 25 28 16 18 17 14 32 17 19 14 17 22 27 22 25 11 6 23 24 32 21 29 18 24 13 10 30 26 27 12 30 28 31 32 31 31 31 31 33 34 39 35 40 35 36 38 37 39 42 43 41 44 48 44 45 46 48 41 46 43 47 49 46 50 47 49 49 50 45
HYC5	top	3 2 8 3 0 3 6 6 0 4 5 0 0 0 0 0 0 0 0 12 5 0 8 45 1 0 4 10 18 9 43 34 40 16 0 0 48 45 49 20 0 0 43 0 0 38 0 22 34 0 0 0 0 2 1 4 0 0 0 0 0 2 3 0 0 5 0 3 7 4 4 0 0 9 0 0 0 3 3 0 0 0 0 0 1 5 0 0 0 0 0 0 0 0 0 0 3 2 7 0 0 11 25 19 15 0 41 24 17 0 0 3 40 30 7 0 0 1 0 4 1 0 0 1 0 4 3 9 3 5 0 0 2 6 0 0 0 41 0 21 0 0 0 29 46 46 0 31 0 14 27 0 0 13 40 0 42 0 24 0 0 0 0 39
	bottom	0 3 2 3 1 3 8 7 0 4 8 0 0 0 0 0 0 0 0 14 8 0 9 47 2 0 8 12 19 12 44 35 42 19 0 0 50 49 50 21 0 0 49 0 0 39 0 23 37 0 0 0 0 6 1 7 0 0 0 0 0 2 4 0 0 6 0 3 8 4 6 0 0 1 1 0 0 0 3 4 0 0 0 0 0 1 6 0 0 0 0 0 0 0 0 0 0 0 3 ~ 3 9 0 0 13 29 20 18 0 43 28 19 0 0 5 44 32 10 0 0 1 1 4 5 0 0 4 0 7 4 1 3 6 0 0 2 7 0 0 0 42 0 22 0 0 0 30 47 48 0 3.3 0 15 29 0 0 1 4 4 5 0 4 6 0 2 5 0 0 0 0 4 0

Table 4: Random CRP examples - set II.

problem		column														
HYC6	top	5	6	5	12	4	19	17	18	25	27	31	31	32	28	36
		3	4	2	10	32	38	28	34	34	24	22	13	42	40	21
		24	24	32	46	1	58	41	16	50	59	14	47	45	43	57
		20	56	45	62	63	38	55	46	8	39	64	1	8	30	3
		39	11	6	44	47	30	58	48	29	63	60	28	35	60	15
		19	55	25	48	33	52	33	6	7	43	0	9	42	23	30
		35	0	69	27	37	2	35	26	51	37	34	54	67	49	59
		68	0	50	0	53	61	66	70	0	62					
	bottom	8	10	7	13	6	21	18	19	29	35	38	37	38	33	41
		5	5	4	11	36	41	32	39	38	28	23	14	44	42	22
		26	25	40	50	4	61	42	17	51	60	15	54	47	44	59
		24	58	46	65	65	43	57	49	10	43	66	3	12	33	9
		40	13	12	45	50	32	60	54	30	64	62	31	43	63	16
		20	56	27	50	36	55	34	7	13	48	0	13	48	24	31
		37	0	70	30	40	9	36	30	53	41	37	55	68	54	61
		69	0	52	0	55	69	67	0	0	64					
HYC7	top	26	21	24	7	18	27	22	30	34	34	17	28	9	13	28
		33	20	15	31	32	10	4	26	6	23	13	3	10	31	29
		9	25	18	28	12	15	27	37	7	20	33	10	21	20	38
		11	8	20	16	16	19	23	14	17	34	10	46	34	16	47
		26	24	38	11	37	33	37	17	42	4	7	6	12	29	14
		39	1	41	42	44	45	24	42	41	43	36	36	35	43	44
		39	45	12	5	5	5	43	39	2	40	45	46	46	47	48
		48	49	49	50	50	51	52	52	52	53	54	54	55	56	56
		56	56	57	57	58	58	59	59	60	60	60	60	61	61	61
		63	63	64	64	66	67	0	0	0						
	bottom	30	27	29	14	40	40	28	31	48	42	31	39	12	27	36
		41	22	20	34	35	17	12	39	15	28	16	5	16	53	31
		10	31	22	32	15	18	39	53	19	30	39	23	23	25	50
		15	9	33	20	18	21	24	23	32	44	15	50	38	29	58
		32	26	49	19	43	38	42	27	51	16	8	26	26	37	15
		50	13	44	45	63	49	25	47	46	44	42	41	37	46	47
		53	55	27	12	17	11	45	44	16	43	54	48	51	60	49
		60	58	61	52	60	69	56	58	63	57	59	70	69	59	62
		66	67	60	61	59	67	64	65	64	65	66	70	62	63	67
		64	66	68	69	69	68	1	2	3						
HYC8	top	26	4	15	9	30	10	9	7	20	26	12	7	9	19	16
		24	15	14	24	30	11	13	6	27	2	10	3	18	25	3
		1	17	5	27	22	29	2	31	32	28	35	8	21	23	5
		32	33	29	31	28	34	36	37	38	39	40	41	42	42	43
		43	44	45	45	46	46	47	48	48	49	50	50	50	51	51
		52	52	52	53	53	54	54	55	55	55	56	57	57	57	58
		59	60	61	62	62	63	64	65	66	67	68	69	0	70	
	bottom	30	5	18	11	35	12	13	8	21	29	14	10	12	22	17
		26	16	15	25	31	15	14	7	30	4	13	7	19	27	4
		3	19	10	28	23	31	6	33	37	35	37	9	22	24	9
		34	34	32	37	29	36	38	38	39	40	41	42	43	44	45
		47	45	46	48	49	50	48	50	52	53	51	53	54	55	56
		54	55	57	55	56	59	60	58	59	60	57	58	59	60	61
		61	61	62	63	66	64	65	68	67	68	69	70	1	0	