1997

A\ ✓

Floorplan Equipartitioning using Staircase Channel Minimizing the Crossing nets

A dissertation submitted
in partial fulfilment of the requirements for the
M.Tech. (computer science) degree
of the
Indian Statistical Institute

By
Nirmalya Chattopadhyay

Under The Supervision of
Dr. Subhas C. Nandy

INDIAN STATISTICAL INSTITUTE
203, Barrackpore Trunk Road,
Calcutta- 700035

Acknowledgements

My sinceremost thanks are due to Dr. Subhas C. Nandy and Dr.(Mrs) Sushmita S. Kolay, without whom this work would never have been a reality.
I would like to thank all the staff here who provided me with encouragement in work, and made the course work interesting.
I would also like to thank all my classmates for their wonderful company making the two years enjoyable.
A special word of thanks to Miss Rekha Menon, Miss Padma Mahalingam, Mr. Manas Ranjan Jagdev, Mr. Monoj Barua, Mr.Debasis Sarkar for their help in completing my project work.
Finally, I would also to take this opportunity to gratefully acknowledge the help provided to me by Electronics unit and the computing facilities provided to me by Computer and Statistical Service Center in I.S.I.

August 2, 1997 Nirmalya Chattopadhyay

Abstract

This dissertation identifies a new problem of partitioning of VLSI floorplan called Balanced monotone staircase partitioning. In a VLSI floorplan, the isothetic rectangular circuit modules are placed on a 2D floorplan and the net attached to each module is given. The objective of global routing is to connect the terminals attached to different modules that belongs to the same net. In this framework the global routing problem is mapped into a series of hierarchical staircase channel routing. So our objective is to divide the floor into two almost equal halves by a monotone staircase channel and simultenously minimize the number of crossing nets. We have defined a mixed optimization problem where the objective function is a convex combination of the following two ratios:
A ratio = difference in area of two partitions/ Total-area.

Nratio= number of crossing net/Total-net.

The problem of partitioning the floor into almost two equal halves is shown to be NP-COMPLETE in [2]. Here we present a heuristic algorithm using circuit partitioning algorithm given by Fiduccia-Mattheyses[6]. Finally we have suggested a new approach to solve the problem.

Certificate of Approval

This is to certify that the thesis titled **Floorplan Equipartitioning Using Staircase Channel Minimizing The Crossing nets.** submitted by *Mr. Nirmalya Chattopadhyay*, towards partial fulfilment of the requirements for the degree of M.Tech. in Computer Science at the Indian Statistical Institute, Calcutta, embodies the work done under our supervision.

[illegible]

(Subhas C. Nandy)
Computer and Statistical Service Center
Indian Statistical Institute,
Calcutta - 700 035

Contents

Chapter 1

Introduction :

The global routing is an important problem in VLSI design which comes after the placement phase of design cycle. In the placement phase, the position of the different circuit modules on the floor are determined. A netlist is attached to each circuit component. Space not occupied by the cells (circuit modules) are used as routing region. In the routing phase the objective is to connect the terminals attached to different modules which belong to the same net. The most common approach of global routing is the sequencial routing[3] i,e to route one net at a time and to choose the shortest path whenever possible. A good heuristic for the problem of finding the shortest connection for an n-pin net using rectlinear Stiner tree is available in[4]. Both the algorithm assumes an underlinw grid structure which causus them to be inefficient with respect to the space complexity for the large problems. Hightower[5] reduces the memory requirement by using the line segment instead of grid nodes in the search. In the high performance circuit,the opti- mization function is often the minimization of congestion in the routing tracks with 100to achieve the goal.

In VLSI design, the position and aspect ratio of circuit modules becomes fixed after completion of floorplaning and placement phase. Let B=$\{b_1, b_2, \ldots, b_n\}$. be a set of n blocks distributed on a rectangular floorplan. Thus a floorplan is a rectangular dissection of the bounding rectangle with isothetic cutsize. Each rectangle bi is attached with a set of nets and cutlines denote the the routing space. We assume that cutlines are meets at T junctions only.

Slicible and non slicible floorplan

A floorplan is said to be slicible if it is obtained recursively by partitioning a rectangle into two parts either by a horizontal line or by a vertical line, otherwise it is nonslicible. For slicible floorplan hierarchical partitioning through the cutlines may be used to facilitate global routing. For nonslicible floorplan such a partitioning does not exist. Furthermore directed cycles appear in their channel digraphs, so it causes infeasible routing order. But if staircase channel is used then the problem of finding suitable routing order can be solved easily by finding suitable staircase channel hierarchically.

Multiterminal net problem

In multiterminal net problem a net is attached to more than two cells. problem is to minimize the number of nets crossing the cut. One algorithm for this problem is given in [2]. We have implemented the algorithm and tested it by several exam- ples and got the correct result in each case.

Balanced monotone staircase partitioning

The problem of dividing the floorplan by a monotone staircase channel into two partition of almost equal size with minimum number of crossing nets is called balnced monotone staircase partitioning. Here the objective is

to solve a mixed optimization problem where the objective function is a convex combination of the following two ratios:
A-ratio= difference in area of two partitions/Total-area. N-ratio= number of crossing net/ Total-net. The problem is shown to be NP-COMPLETE in [2].

In this dissertation we have considered the balanced monotone staircase problem. We took the min-cut partition obtained by the algorithm given in [2] as our initial solution and try to improve balance by swaping block (which is chosen using FM heuristic) from heavier partition to the lighter partition keeping the staircase monotone and cutsize as minimum as possible. Finally we have suggested anoter approach where cell is chosen according to size difference between the partitions due to its transfer.

Our thesis is organized as follows. In chapter 2 we discussed some general facts which will give us some idea for choosing heuristic. In chaoter 3 we presented our algorithm which we have implemented. In chapter 4 we have discussed our new approach which is yet to be implemented to judge its performance. Our plan is to implement the second heuristic to compare between the two heuristics proposed in this thesis.

Chapter 2

Problem Formulation

2.1 Objective

Consider a floorplan consisting of a set of cells $\{c_1, c_2, \ldots, c_n\}$. We shall denote the cell that appears at the top-left corner as the source and that appears at the bottom right corner as sink block. The partition containing the source (sink) is called S-partition (T-partition). Our objective is to get an almost balanced monotone staircase partition such that the cutsize is as minimum as possible. The objective function to be optimized is mentioned in the introduction.

Precisely our objective is as follows:

- To maintain monotonicity of staircase partition.
- To obtain area difference between S-partition and T-partition as minimum as possible.
- Cutsize should not be much different from min-cut.

transferred group of c_i and c_is denoted by TG(i). So TG(i)= $\{c_i\} \cup$ FOLLOW(i). c_i is called base-cell of TG(i) or base-cell in short.

Now consider the following theorem.

Theorem: Take a monotone staircase partitioning. Suppose c_i is base cell. Then partition is still monotone after moving c_i iff all c_j belongs to FOLLOW(i) are also transferred with c_i;

proof:

Suppose c_i is transferred with all c_j s.t c_j belongs to FOLLOW(i); To show that obtained staircase is monotone. Suppose not. Then there exist a directed edge in G from T to S. Since initially staircase was monotone. So there does not exist any directed edge in G from T to S at that time. This implies either this edge is from one of the transferred cell to a cell c_j of S (if initially c_i is in S) or this edge is to one of the transferred cell from a cell c_j of T (if initially c_i is in T). In both the cases it is clear that c_j is not transferred from its initial position with c_i. Now we see that if c_i is initially in S (T) then there exist a dipath in G from (to) c_i to (from) c_j. So in both the cases c_j belongs to FOLLOW(i). But we have just seen that c_j is not transferred, which is a contradiction.

Conversely, suppose the obtained cut is monotone.we have to show c_i is transferred with all its follower cells. Suppose there exist c_j belongs to FOLLOW(i) which is not transferred. If c_i belongs to S initially then there exist a dipath from c_i to c_j. Otherwise (if c_i belongs to T initially) there exist a dipath from c_j to c_i. In both cases there exist a directed edge from T to S in G. Which ymplies that the cut is non monotone. Which is a contradiction. ♣

So starting with min-cut partition at each pass we wi|l choose an appropriate c_i s.t movement of TG(i) will improve balance and simultenously keeps the cutsize as minimum as possible. Note that c_i is the base cell of TG(i), so from now we call selected cell as base cell. by above theorem monotonicity will be preserved at each pass. So balanced partition will be monotone.

In order to maintain the restriction on the cutsize we have to find a suitable base cell. Needless to mention that there is always a possibility of reducing the cutsize by transfer of cells from top-left corner or bottom right corner. But such a choice of base cell will transfer a huge number of cells to the complementary partition which will again destroies the balance.

The above facts implies that if we give more importance to obtain balance (i,e (2)) than to reduce cutsize then we have to select those cells which has minimum number of follower cells. So it is obvious that we prefer the cells at the boundary of the partition as base cells and develop algorithm based on it. But before presenting an algorithm let us formalize the problem and try to see how the cutsize is affected by the transfer of a base cell.

In the next section we give some formal definitions,and from those we present some observations. We use some notations to make the treatment easier.

2.2 Notations

- N=Set of all nets.

For a net n $\in$ N

- DP(n) => { set of all cells c_i in partition P s.t n $\in$ c_i }.

- D(n) => { set of all cells c_i s.t n $\in$ c_i }.

- : α_n, β_n w.r.t a c_i.

If $c_i \in$ P we denote α_n= |DP(n)| (= number of elements of DP(n)). and β_n =|DQ(n)| where Q is the complementary partition of P.

2.3 Change Of Cutsize

Let us look at the change of cutsize due to transfer of the base cell with its follower cells. We first try to compute this change. For this purpose we have to define some notions which are as follows:

For each c_i we define Gain(i) as the reduction of cutsize due to change of c_i to its complementary partition. i,e Gain(i)=Old-cutsize-New-cutsize, where Old-cutsize is the cutsize before transferring c_i(and its follower cells) and New-cutsize is the cutsize after transferring c_i(and its follower cells).

Definitions

- A net is said to be totally covered by c_i in partition P if $c_i \in$ P and DP(n)$\subset$ TG(i).

- A net is said to be partially covered by c_i in partition P if $c_i \in$ P DP(n)$\cap$[TG(i)]$\neq$ NULL and $\exists$ cell c_j containing n s.t $c_j \in$ P and $c_j \notin$ {$c_i$$\cup$FOLLOW(i)}.

  From now onwards we will not mention about partition P further; We call n totally covered by c_i or n partially covered by c_i.

- FULL(i)={set of all nets which are totally covered by c_i}.

- PART(i)={set of all nets which are partially covered by c_i}.

- POSITIVITY[i]= #{n: n $\in$ FULL(i) and β_n>0}.

- NEGATIVITY[i]= #{n: n $\in$ PART(i) and β_n=0}.

Example: Consider Fig 1. Here 1,2,3,4 are cells and a,b,c,d,f are nets.

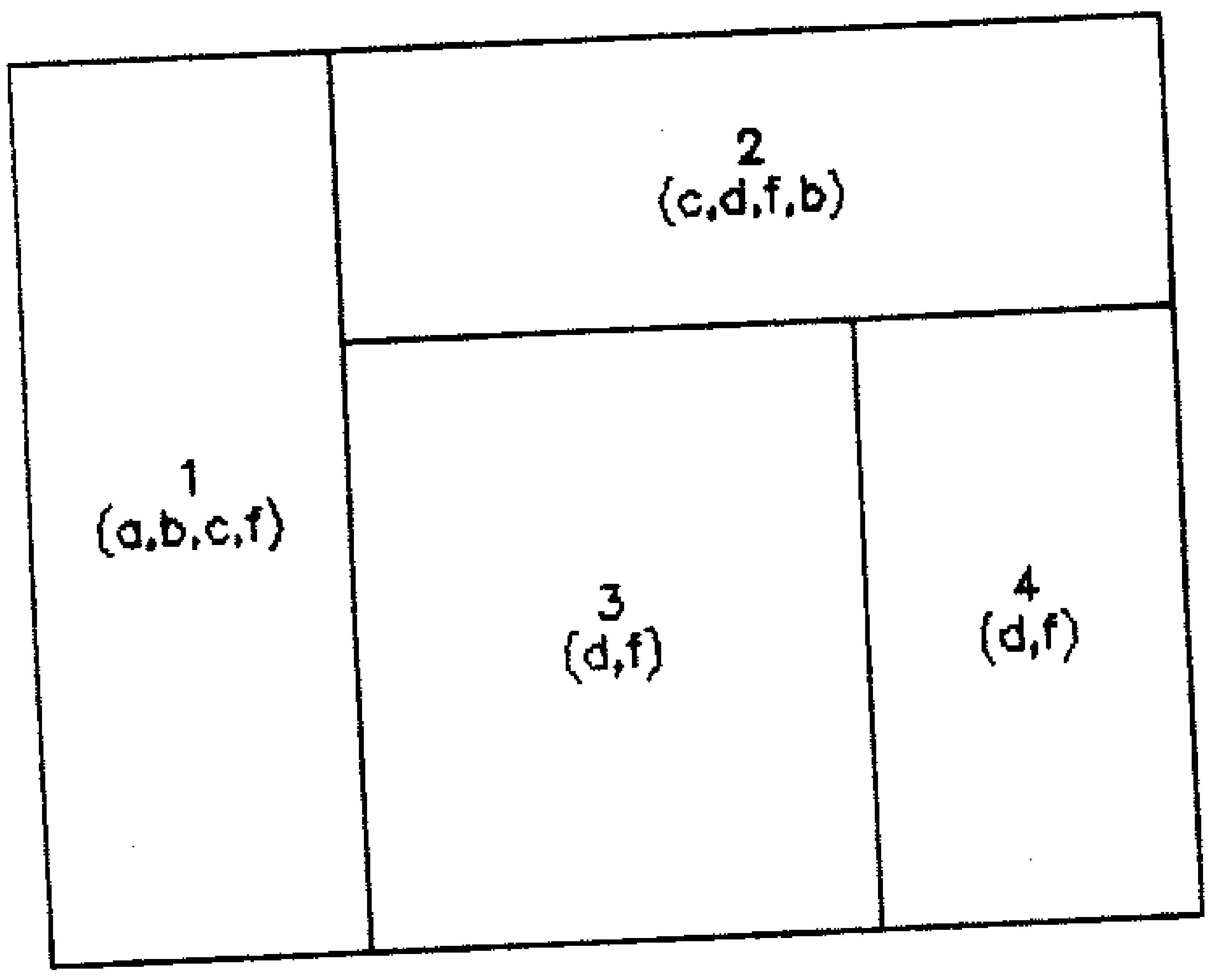


Figure no. 1

Let us consider cell 2.

We get FOLLOW(2)={3, 4}.

FULL(2)= {d}. PART(2)= {b, c, f}.

Lemma: Gain(i) = POSITIVITY[i] - NEGATIVITY[i].

proof:

Since if the net n $\in$FULL(i) then after transferring c_i to its complementary partition $\nexists$ any cell in previous partition of c_i which contains n. again β_n>0 => n crossed the cut before transferring c_i. That means cutsize is reduced by 1 for n; Hence total reduction of cutsize for such type of net is POSITIVITY[i].

Again consider n s.t n $\in$PART(i) and β_n=0; Since β_n=0,before transferring c_i n has no effect on the cutsize. Now since n $\in$PART(i).So $\exists$ at least one cell containing n which has not transferred with c_i; This implies cutsize will be increased by 1 for such n.

Hence total increament of cutsize for such type of net is NEGATIVITY[i].

So New-cutsize = Old-cutsize - POSITIVITY[i] + NEGATIVITY[i].

=> POSITIVITY[i] - NEGATIVITY[i] = Old-cutsize - New-cutsize = Gain(i)(by definition) ♣

Now the status of the nets which are neither on c_i nor on any follow of

c_i. will remain unaffected by the movement of c_i; we call all such nets as impassive nets of c_i. More formally,

IMPASSIVE(i)= set of all nets n s.t DP(n) $\cap$ TG(i) =NULL where $c_i \in$ P.

Result: N=FULL(i)$\cup$PART(i)$\cup$IMPASSIVE(i) (1)

again they are mutually exclusive. i,e

FULL(i)$\cap$PART(i)=NULL; (2)
FULL(i)$\cap$IMPASSIVE(i)=NULL; (3)
IMPASSIVE(i)$\cap$PART(i)=NULL; (4)

proof: To show (1).

Let n$\notin$IMPASSIVE(i). Then D(n)$\cap$[TG(i)]$\neq$NULL.

Put X=D(n)$\cap$[TG(i)]. Now for each $c_j \in D(n)\cap$[TG(i)], $c_j \in$ same partition as c_i; [since $c_j \in$TG(i)]

call this partition P.

i,e DP(n)$\cap$[TG(i)]$\neq$NULL. (A)

If DP(n) $\subseteq$ TG(i) then n $\in$ FULL(i). Otherwise $\exists c_k \in$ DP(n) s.t $c_k \notin$TG(i) (B)

By (A) &(B) n $\in$ PART(i);

Hence if n$\notin$IMPASSIVE(i), n $\in$ PART(i)$\cup$FOLLOW(i). =>N-IMPASSIVE(i) $\subseteq$ PART(i)$\cup$FULL(i).

conversely let n $\in$ PART(i)$\cup$FULL(i). Then DP(n)$\cap$[TG(i)]$\neq$NULL. =>n

$\notin$ IMPASSIVE(i). =>n $\in$ N-IMPASSIVE(i). So (1) is proved.

(2),(3)&(4) are obvious. ♣

So we see that for a fixed cell c_i a net n is in one of the three status. i,e either it is totally covered by c_i or it is partially covered by c_i or it is impassive of c_i. Note that this distribution of the set of nets N is different for different cell.

Example: Consider the floorplan in the picture Here a,b,c,d belongs to both side of the cut. So cutsize=4;
Let us compute the Gain of the cell 6.

FOLLOW(6)={4,3,8}.
FULL(6) = {c}.
PART(6) = {a,b,d,e,f}.
IMPASSIVE(6)= ϕ.
Again, T(a)=2, T(b)=0,T(c)=4,T(d)=2, T(e)=3,T(f)=0.
So POSITIVITY[i]=1, NEGATIVITY[i]=2.
=> Gain(6)= -1.

A Method of computing Gain(i)

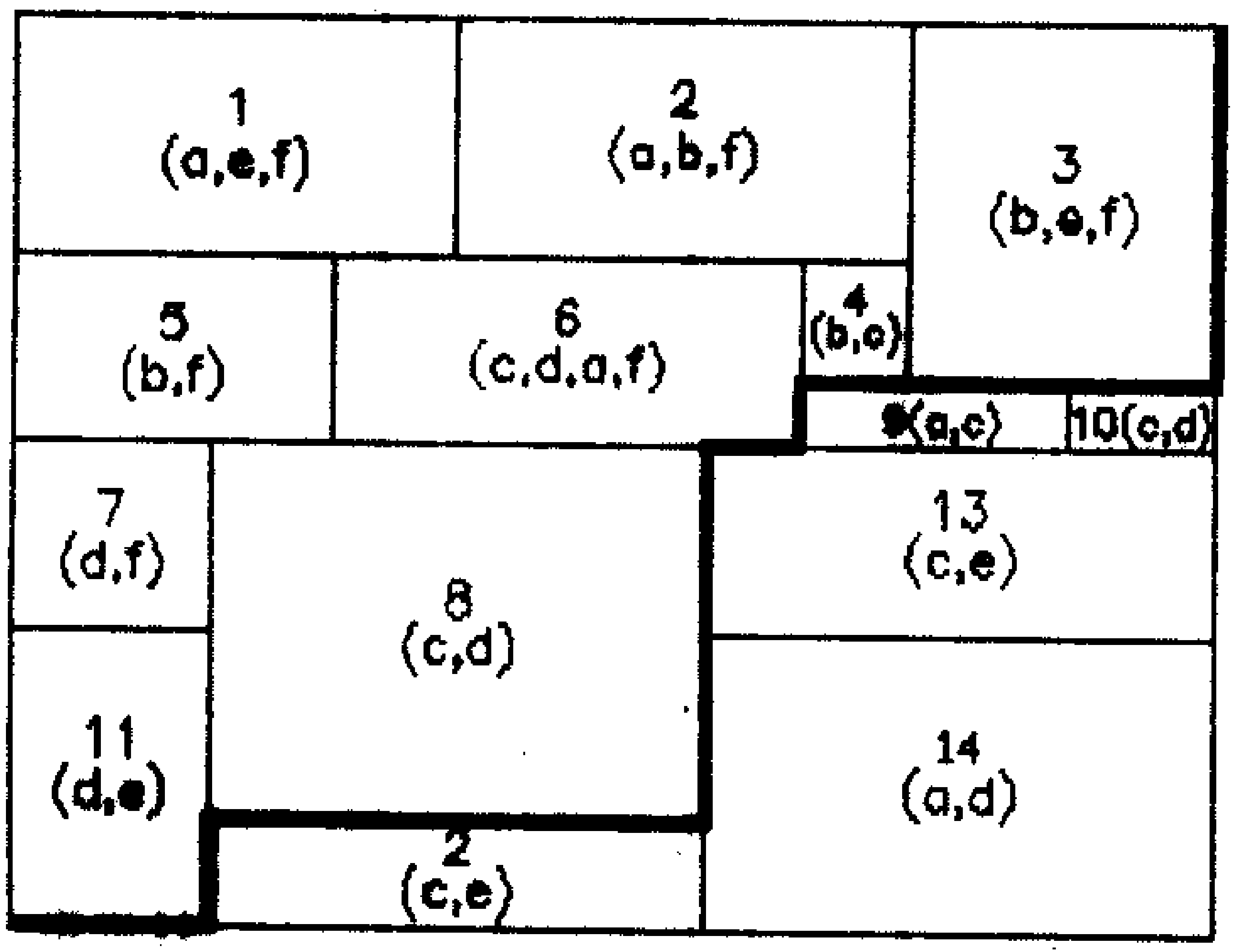
In order to compute Gain(i) for a cell c_i let us define a matrix namely NET-CELL matrix.

NET-CELL matrix for partition S(denoted by NET-CELL(S)).

Let |S|= Number of cells in partition S.

|N|= Total number of nets.

For each cell $c_j \in$ S and for each net n $\in$ N form the ordered pair (j,n);



1 (a,e,f)
2 (a,b,f)
3 (b,e,f)
4 (b,c)
5 (b,f)
6 (c,d,a,f)
9 (a,c)
10 (c,d)
7 (d,f)
8 (c,d)
13 (c,e)
11 (d,e)
14 (a,d)
2 (c,e)

Picture 2

Thus we have total |S|.|N| such ordered pairs.

Now consider the matrix

NET-CELL(S)=((a[i][k])) of order |S|x|S|.|N| as follows

The columns of the matrix corresponds to the cells in S

The rows will correspond to the ordered pairs (j,n) in their lexieographical order such that $c_j \in$ S and n $\in$ N.

Now if the ordered pairs starting with j appears first in p-th row will end at (p+|N|-1)-th row. In this case call p as START(j) and

define RANGE(j)= set of all integers from START(j) to START(j)+|N|-1.

Now for $c_k \in$ RANGE(j) define the element a[i][k] as follows:

a[i][k]=(x,y) where
x=1 if either i=j or $\exists$ a dipath from c_j to c_i(i,e $c_i \in$ FOLLOW(j));
=0 otherwise.
y=1 if n $\in c_i$.
=0 otherwise.
[NOTE THAT a[i][k]'s are from the field (C,+,.)
C is the set of all complex numbers]

similarly constru[illegible] [illegible]L(T)=[a[i][k]] of order |T|x|T|.|N|

where for k $\in$ RANGE(j)

a[i][k]=(x,y) where
x=1 if either i=j or $\exists$ a dipath from c_i to c_j(i,e c_i is in FOLLOW(j)); =0 otherwise.
y=1 if n $\in$ c_i.
=0 otherwise.

OBSERVATIONS:

Consider NET-CELL(S).

Let elements are generally denoted as (x,y); Consider a particuler row(let it corresponds to (j,n));

• Let (p,q) be the sum of all ordered pairs for which x=1.(note that p$\geq$1 as x=1 for i=j). If q=0 then it implies the cell c_j and all the cell which has a dipath from c_j does not contain the net n.That means neither cell c_j nor any of its follower contains net n.=> n $\in$ IMPASSIVE(j).

• If $\exists$ two elements (1,1) and (0,1) in the row (j,n) then it implies that $\exists$ a follower cell of c_j(or cell c_j itself) which contains n and $\exists$ another cell which belongs to the same partition as c_j(which is of course S here note that we consider NET-CELL(S) here) which contains n and is not a follower cell of c_j. Which implies that n $\in$ PART(j).

• If $\exists$ at least one (1,1) and $\nexists$ any (0,1) then it implies that $\exists$ a follower cell of c_j which contains n and $\nexists$ any cell in S partition which is not a follower cell of c_j containing n; i,e n $\in$ FULL(j).

similar results holds for NET-CELL(T). We will write formal algorithm for computing these latter. Now we concentrate on some more observations which will give us some better insight that finally helps to choose better heuristic.

SOME RESULTS AND DECISIONS:

Result1: For min-cut partitioning Gain(i)$\leq$0 $\forall$ i.

proof: If there is no c_i s.t Gain(i)>0. Hence cutsize can be improved further by changing c_i to its complementary partition which contradicts the fact that the cut is a min-cut.

Result2: If $c_j \in$ FOLLOW(i) then FOLLOW(j) $\subseteq$ FOLLOW(i).

proof: case1: $c_i \in$ S. Then $c_j \in$ and $\exists$ a dipath from j to c_k. Take $c_k \in$ FOLLOW(j). Then $c_k \in$ S and $\exists$ a dipath from c_j to c_k. => $\exists$ a dipath from c_i to c_k. So by definition $c_k \in$ FOLLOW(i). Hence the result.

case2: $c_i \in$ T. Then $c_j \in$ T and $\exists$ a dipath from c_j to c_i.

Take $c_k \in$ FOLLOW(j). Then $c_k \in$ S and $\exists$ a dipath from c_k to j. => $\exists$ a dipath from c_k to c_i. So by definition $c_k \in$ FOLLOW(i). Hence the result.

Result3: If $c_j \in$ FOLLOW(i) then FULL(j) $\subseteq$ FULL(i).

proof: Take n $\in$ FULL(j). Then DP(n) $\subseteq$ {cell(j)}$\cup$FOLLOW(j). Since $c_j \in$ FOLLOW(i). So by (R2) FOLLOW(j) $\subseteq$ FOLLOW(i). => DP(n) $\subseteq$ FOLLOW(i) $\subseteq$ TG(i)=FULL(i). => n $\in$ FULL(i) (proved).

Result4: If $c_j \in$ FOLLOW(i) then POSITIVITY[j]$\leq$POSITIVITY[i];

proof: Note that FULL(j)$\cap$\{n: β_n>0\} $\subseteq$ FULL(i)$\cap$\{n:β_n>0\}. [Since by Result3 FULL(j) $\subseteq$ FULL(i) and c_j is in the same partition as c_i, so β_n is same for both of them.] => #\{n: n $\in$ FULL(j) and β_n>0\} $\leq$ #\{n:n $\in$

FULL(i) and $\beta_n>0$}. =>POSITIVITY[j] $\leq$ POSITIVITY[i].

REMARK:

Note that same thing we cannot say about NEGATIVITY[j] if $c_j \in$ FOLLOW(i), because there may exist n $\in$ PART(i) s.t n $\in$ IMPASSIVE(j). So PART(j) may not be a subset of PART(i).

Result5: If PART(i)=NULL then Gain(i)$\geq$Gain(j).

proof: Since PART(i)=NULL. => NEGATIVITY[i]=0; => Gain(i)=POSITIVITY[i] $\geq$POSITIVITY[j] $\geq$ Gain(j) $\forall c_j \in$ FOLLOW(i).

REMARK:

Result shows that if PART(i)=NULL then the chance of c_i for going TOP of the BUCKET is greater than the chance of c_j for going TOP $\forall c_j \in$ FOLLOW(i) if BUCKETs are arranged according to the gain.

Result6: If $c_i \in$ S, n $\in$ source and n $\in$ TG(i) then PART(i) $\neq$NULL. [similar result holds for sink also]

proof: Since source $\notin$FOLLOW(i) for any i s.t $c_i \in$ S;

Now DS(n)$\cap$[TG(i)] $\neq \phi$. => n $\in$ PART(i).

REMARK:

Note that if c_i -> Top-left corner then FOLLOW(i)->{set of all cells in S partition}. => PART(i)-> ϕ. So by Result5 Gain(i) is probably bigger than all of its follows.. But note that transfer of cells nearer to Top-left corner causes the transfer of a huge number of cells(which are all follower cells of c_i) with it. Hence balance will be destroyed. It implies that it may not be possible to transfer a cell which is most suitable for cutsize.

Result7: If $c_i \in P$ then IMPASSIVE(i) $\supseteq$ {n: |DP(n)|=0}. proof: Obvious. Since $\nexists$ any cell in P which contains n. Hence n is neither on c_i nor on any follower cell of c_i. => n $\in$ IMPASSIVE(i).

REMARK: If $c_i \in P$ and IMPASSIVE(i)=ϕ, then by above result {n: P(n)=0}=ϕ. So n $\in$P $\forall$ n $\in$ N. It implies if P and complement of P both contains an IMPASSIVE cell then all nets passes through the cut.(i,e most harmful situation.).

Result 8 : If source(sink) does not contain all the nets then min-cut < |N|.

proof: Suppose source(or sink) does not contain all the nets. Consider the partition S={source} (or T={sink}). Since $\nexists$ any cell whose corresponding node has a dipath to(from) source(sink) in G. => $\nexists$ a directed edge from T to S in G. => resulting cut is monotone. Again on this cut no. of crossing net= no. of nets on source(sink)=c (say). Since the cut is monotone => min-cut $\leq$ c < |N|.

Example: Consider the floorplan in Fig. 3. Here source={1}. Sink ={9}. Note that none of them contains all the nets. So min-cut cannot contain all the nets. Moreover min-cut $\leq$ 3.

Result 9 : If min-cut < |N| then the corresponding partitions, S and T cannot contain IMPASSIVE cells both.

proof: Obvious. Otherwise each net will cross the cut=> min-cut=|N|. Which is a contradiction.

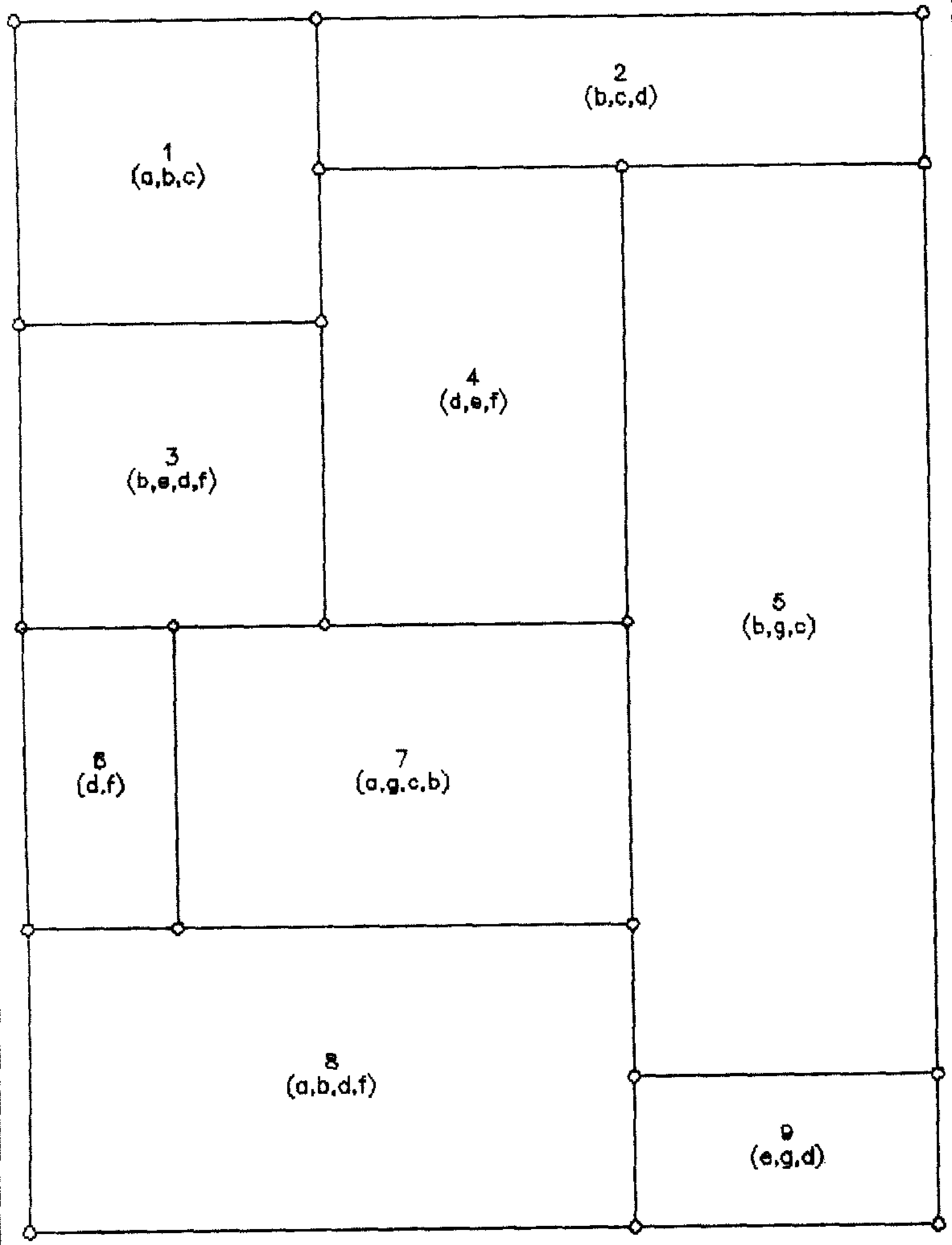


Figure 3

Result10: Let IMPASSIVE(i)=ϕ. Then $c_i \in$ FOLLOW(j) => Gain[j]$\geq$Gain[i]. (Note that IMPASSIVE(j)=ϕ also)

proof: Since IMPASSIVE(i) =ϕ.

=> N=FULL(i)$\cup$PART(i). (1). Since $c_i \in$ FOLLOW(j). So if n $\in$ PART(j) then n$\notin$FULL(i). => n $\in$ PART(i) (by (1)). => PART(j) $\subseteq$ PART(i). => NEGATIVITY[j] $\leq$ NEGATIVITY[i]. Again by Result4 POSITIVITY[i]$\leq$POSITIVITY[j So Gain[j]= POSITIVITY[j]-NEGATIVITY[j] $\geq$ POSITIVITY[i]-NEGATIVITY[i] =Gain[i].

REMARK:

Note that if IMPASSIVE(i)=ϕ then each net passes either through it or one of its follower cells. Hence if $c_i \in$ FOLLOW(j) then IMPASSIVE(j)=ϕ also.If $c_i \in$ FOLLOW(j) call c_j is a ANCESTER of c_i. Now if IMPASSIVE(i)=ϕ then as we go through ANCESTOR to ANCESTORS then gradually Gain increases. Now notice that the cell i for which IMPASSIVE(i)=ϕ has a high possibility that it has a large number of follows. So it is not surprizing that the cells which are most fit for reducing cutsize disturbs balance very often.

Result12: Let n: |DP(n)|=0=ϕ where P is S or T. Then $\exists$ i $\in$ P s.t IMPASSIVE(i)=ϕ.

proof: Since n: |DP(n)|=0=ϕ. So all net belongs to P. W.l.g let P=S. Since each cell of S is a follower cell of source. So each net belongs to either source or a follower cell of source. => IMPASSIVE(source)=ϕ.

Result13: If $c_i \in$ P and IMPASSIVE(i)=ϕ and $c_j \in \overline{P}$ Gain(j) $\geq$ 0.

proof: Since IMPASSIVE(i)=ϕ. => n: |DP(n)=0; So for any $c_j \in \overline{P}$ NEGATIVITY[i]=0; => Gain(i) $\geq$ 0;

For next part since n: |DP(n)| $\geq$0=N.

NOW Gain(j) =|n: n $\in$ FULL(j) and |DP(n)| >0| $\forall$ j $\in \overline{P}$.

=> |n: n $\in$ FULL(j)|=|FULL(j)|.

Result14: For min-cut partitioning if $\exists$ $c_i \in$ S(T) s.t IMPASSIVE(i)=ϕ, then $\forall c_j \in$ T(S) except c_j=sink(source) Gain(j)=0.

proof: W.l.g let $c_i \in$ S s.t IMPASSIVE(i)=ϕ.

=> $\forall c_j \in T \quad Gain(j) \geq 0$; (1)

Now for min-cut partition $\forall c_k \notin$ {source,sink}
Gain(k) $\leq$ 0;
=> $\forall c_j \in$ T-{sink} Gain(j) $\leq$0; (2)

by (1) and (2) result follows.

Lemma: For min-cut partitioning suppose $c_i \in$ S s.t IMPASSIVE(i)=ϕ. Then all nets passing through in cut belongs to sink.(similar result holds for source also).

proof: Take $n_1 \in$ sink. Then it must crosses through the cut otherwise $n_1 \in$ {n:|DS(n)|=0}. Which is a contradiction(note that {n: |DS(n)|=0}=ϕ}).

So |{n:n $\in$ sink}| $\leq$ cut-size. Again cut is min-cut, so cut-size $\leq$ |{n: n $\in$ sink}|. => cutsize= |{n:n $\in$ sink}|. Now {n:n $\in$ sink} $\subseteq$ {n: n crosses the cut } and both has the same cardinality. So the result follows. ♣

/* COMPUTATITION OF S(n) */

Let element of NET-CELL(S) is generally denoted as (dflag,nflag).

Consider row (i,n) for some i. ADD all nflag s.t nflag $\neq$0. This is the number of cells in S which contains n, i,e S(n).

Take $c_i \in$ S.

```
/* COMPUTATION OF Gain(i),|FULL(i)|,|PART(i)|,|IMPASSIVE(i)| */

Gain(i)=0; /* |FULL(i)|=FULL. |PART(i)|=PART,|IMPASSIVE(i)|=IMPASS
*/

FULL=0;

PART=0;

IMPASS=0;

For each net n
{
Consider (i,n) in row in NET-CELL(S).

pflag=FALSE;

fflag=FALSE;

for all element (dflag,nflag)


if((dflag,nflag)=(1,1))
fflag=TRUE;
else if((dflag,nflag)=(0,1))


if(fflag)
pflag=TRUE;
PART++;
if($\beta_n$==0)
Gain(i)- ;
break;
//end if
} //end for
if(fflag)
{
if(!pflag)
{
FULL++;
if($\beta_n$>0)
Gain(i)++;
  } else
{
if(!pflag)
IMPASS++;
}
} //end for
```

Chapter 3

Main algorithm :

In this chapter we will present our algorithm. We have seen in chapter2 that in each transfer a group of cell will move(except a few cases where base cell has no follower cells). Now we see the technique of cell selection in detail. First let us describe the data structure.

3.1 Data structure:

We mainly followed the data structure given in FM algorithm. i,e here corresponding to S and T partition we maintain two array of doubly linked lists call them SBUCKET[0, ..., 2pmax] and TBUCKET[0, ..., 2pmax] respectively where pmax= MAX{Total-net-#{nets$\in$source}, Total-net-#{nets$\in$sink}}.

Put Pos(i)=Gain(i)+pmax. [note that Pos(i) $\geq 0 \ \forall$ i].

Now consider SBUCKET only (similar for TBUCKET). We put a c_i in j-th entry of SBUCKET if $c_i \in$ S and Pos(i)=j.

W.l.g let Size(S) > Size(T) after min-cut partitioning.

Now from Result5 of chapter 2 we see that that the cell at the top left corner has a higher possibility to go to the top of the SBUCKET. Hence possibility of getting a boundary cell at the top of the bucket is too low(note that here boundary cannot be nearer to the top left corner as Size(S) > Size(T)).
So let us define TOP1 as,

TOP1=MAX{Pos(i): $c_i \in$ S and c_i is a boundary cell}. (TOP2 for TBUCKET).

(Note that although we are interested for boundary cells only yet we maintain all the cells in the bucket, as just after transferring base cell boundary cells are changed).

We also maintain a link list called FREE CELL LIST where we put the selected cell(i,e base cell) and all it's follower cells(and lock them all).

3.2 Method of Selection:

Since we have to select a cell from heavier partition and move it to the lighter partition with all it's follower cells. We will proceed as follows. Consider the bucket corresponding to the heavier partition. W.l.g let it is SBUCKET. Among the boundary cells in TOP1 take that cell whose directed area(the sum of area of the cell and it's follower cells) is minimum. If it doesn't destroy the balance(whose possibility is high because very few cells are transferred with a boundary cell) then select it as a base cell otherwise no base cell can be selected and pass is terminated. Update the boundary cells with TOP1 and TOP2.After selecting a cell we lock it and all the cells which has a dipath from selected cell. We then put the selected cell and all of its follower cell.

We want to minimize the objective function namely which is, alpha(Aratio)+(1-alpha)(Nratio) where Aratio= difference in area of two partitions/ Total-area and

Nratio= number of crossing net/Total-net.

First we establish the initial balance from min-cut partitioning(if possible). At that time we choose a cell and swap it to its complementary partition untill all cells are locked. After establishing initial balance we will try to improve it. At each pass the objective function will be computed and will be compaired with previously computed value. If new objective function is less than old then this value is stored otherwise if it is bigger then old value is stored and old partition is maintained. Finally if both are same (i,e no change during the pass) then algorithm will be terminated. Details as follows.

3.3 Algorithm:

INPUT: A set of cells with their geometric positions in the floorplan and associated net list. OUTPUT: A set of cells which are in S partition.

(1) Find min-cut paritioning using algorithm given in [2].

(2) Compute Gain of each cell and initialize buckets.

(3) / ESTABLISH INITIAL BALANCE /

```
W=Size(S)+Size(T);
flag=FALSE.
while(!flag)
{
if((Size(S)<W/2-smax)||(Size(S)>W/2+smax))
{ j=Selected-cell();
Find-boundary(j); // this actually changes the boundary status of the cells due to transfer of j
if(j≠0)
{
Lock(j) //locks $c_j$ and all its follower cells
Pick(j) //takes out $c_j$ and it's follower cells from bucket and puts into free cell list
Modify-gain();
Change-status();// changes status flag of all transferred partitions
} else
break;
} // end if else
flag=TRUE;
} //end while
```

if((Size(S)<W/2-smax)||(Size(S)>W/2+smax)) then initial balance is not

```
established. So balanced partition is not possible. else
{
Prev-objective=Comp-obj(); /computes objective function /
Compute-gain();
Set-bucket();
}
(5) / IMPROVE BALANCE /
flag=FALSE;
MIN=Prev-objective;
while(!flag)
{
Unlock all the cell.
Move-possible=TRUE;

while((UNLOCK())&&(Move-possible))
{
j=Selected-cell();
Find-boundary(j);
if(j==0)
Move-possible=FALSE;
else
{
Lock(j) // locks $c_j$ and qll its follower cells
Pick(j) // takes out $c_j$ and it's follower cells from bucket and puts into free
cell list /
Modify-gain();
Store-status(); // shaves the status of each partition
Change-status(); // changes the status of each partition }
if(Prev-objective > Comp-obj())
{
Prev-objective = Comp-obj();
} else
{
Release();
Back-status();
} } // end else } // end while
if(MIN==Prev-objective)
{
flag=TRUE;
/ OUTPUT HERE /
}
```

```
else
{
MIN=Prev-objective;
Compute-gain();
Set-bucket();
} // end else
} // end while
```

Chapter 4

Another Approach

4.1 Introduction

Let us look at the problem in another way. Previously we have chosen that boundary cell as base-cell which has maximum gain among all the boundary cells. But see that in this way much less number of cells are transferred in each pass, so number of iteration increases(although possibility of rejection of a pass decreases.). Also since cells were arranged in buckets according to their gain, so it was not very clear that which cell would decrease the size difference between S and T greatly.
keeping this idea in mind we have arranged our buckets according to the size difference between S and T due to transfer a cell. Details will be described latter. What is important thing here we will get directly the cell which is most fit to decrease the size difference between S and T. Again notice that previously we have always transferred the cells from heavier side to lighter side. What would be happened if we remove this restriction ? Then there was a chance to transfer cell in lighter side to heavier side, which is too

much harmful for establishing balance. Also notice that we have started from min-cut partition. So there was a possibility to reach in a condition where cutsize and balance both are warser than min-cut partiton (cutsize will be warser than min-cut in any case of course.). That means we might be loser in all respect! So this restriction was necessary there. But note that without this restriction we may also reach to a balanced partition whose balance is not as good as before(with restriction) but cutsize will be improved.

In this approach we will omit this restriction. However the buckets are arranged in such a way that will ensure that balance will never be warser than min-cut balance.

4.2 Theme of the Algorithm

Min-cut-difference: The area difference obtained by min-cut partitioning.

Directed-area: The sum of the area of a c_i and follower cells of c_i is called the directed area of c_i; It is denoted by dr-area[i].

More formally, dr-area[i]= $\sum_{c_j \in \{c_i\} \cup FOLLOW(i)} area[j]$.
Note that dr-area[i] is actually the amount of area transferred by c_i from one partition to the other.

Size: Let M be the set of all partition P s.t either source $\in$ P or sink $\in$ **P** but not both of them. Define a function Size from M to Z^+ (the set of all +ive integers) by Size(P) $= \sum_{c_j \in P} area[j]$. If P contains the source then we denote it by S oterwise by T.

Size difference due to c_i: After transferring c_i to its complementary partition the absolute Size difference between two partition is called the Size difference due to c_i. It is denoted by diff(S,T,i).

If c_i is initially in S block then after transferring c_i from S to T

Size(S) -> Size(S)-dr-area[i];
Size(T) -> Size(T)+dr-area[i];

So diff(S,T,i)=abs((Size(S)-dr-area[i])-(Size(T)+dr-area[i]))
=abs(Size(S)-Size(T)-2(dr-area[i])).
If c_i is initially in T block then after transferring c_i from T to S
Size(S) -> Size(S)+dr-area[i];
Size(T) -> Size(T)-dr-area[i];
So diff(S,T,i)=abs((Size(S)+dr-area[i])-(Size(T)-dr-area[i]))
=abs(Size(S)-Size(T)+2(dr-area[i])).

Active Cell : The c_i for which diff(S,T,i)<= min-cut-difference is called an Active Cell. We are interested for Active Cell's only.

ACT=set of all Active Cell.
dmax=max{diff(S,T,i);i $\in$ ACT};
Note that dmax <= min-cut-difference.
Let us consider two buckets for two blocks(S and T); The index runs from 0 to dmax. Call the buckets as SBUCKET[0, ..., dmax] and TBUCKET[0, ..., dmax], where j-th entry contains a link list of free cells such that if c_i

is in the j-th entry then diff(S,T,i)=j;

If $\exists$ at least one c_i s.t diff(S,T,i)=j we call j is non empty.

Let SMDIF=min{j of SBUCKET such that j is non empty}.

TMDIF=min{j of TBUCKET such that j is non empty }.

Consider c_i from SBUCKET s.t diff(S,T,i)=SMDIF and Gain(i) is maximum among all other cell of SMDIF-th entry.

Consider c_j from TBUCKET s.t diff(S,T,j)=TMDIF and Gain(j) is maximum among all other cell of TMDIF-th entry.

If (Gain(i) > Gain(j))

choose(c_i).

Else If (Gain(i) < Gain(j))

choose(c_j).

Else choose c_i (c_j) if diff(S,T,i) (diff(S,T,j)) is minimum.

We call selected cell as base cell.

$Byconstructionofbucketsitisclearthatdifferencebetween Size(S) and Size(T) willneverb$

RESULTS :

(R1) If Size(S) != Size(T) after min-cut partitioning then at least one bucket is empty initially.

Proof: Case1: Suppose Size(S) > Size(T). Since initial partition = min-cut partition. So initially min-area-difference=Size(S)-Size(T).

Now for $c_i \in$ T we get diff(S,T,i)= |Size(S)-Size(T)+2(dr-area[i])| = Size(S)-Size(T)+2(dr-area[i]) (since Size(S)-Size(T) >0). > Size(S)-Size(T) (since dr-area[i]>0) = min-area-difference >= dmax. So $\nexists c_i \in$ T so that diff(S,T,i) <= dmax, hence none of them will go to the TBUCKET.

Case2: Suppose Size(S) $<$ Size(T). Initially min-area-difference=Size(T)-Size(S).

Now for $c_i \in$ S we get diff(S,T,i)= |Size(S)-Size(T)-2(dr-area[i])| = Size(T)-Size(S)+2(dr-area[i]) (since Size(T)-Size(S) >0). > Size(T)-Size(S) (since dr-area[i]>0) = min-area-difference >= dmax. So $\nexists c_i \in$ S so that diff(S,T,i) <= dmax, hence none of them will go to the SBUCKET.

Remark : Above observation shows the fact that initially cell will be chosen from more weighted partition.

If c_i is transferred from S(T) to T(S) then all the T(S) cells to(from) which $\exists$ a dipath from(to) c_i are now locked by c_i (actually c_i is now added to the FOLLOW set of all these T(S) cells).

Let INDEP(i)= c_j; $c_j \in$ complementary block of c_i and not locked by c_i.

Note that if c_i is transferred from T(S) to S(T) and $c_k \in$ INDEP(i) then c_k $\in$ S(T).

(R2) If c_i is the base cell and $c_k \in$ INDEP(i) then $c_k \in INDEP(j) \forall c_j \in$ FOLLOW(i).

Proof: Suppose not. If c_i is transferred from T(S) to S(T) then since $c_k \in$ INDEP(i) so $c_k \in$ S(T). By our assumption $\exists$ $c_j \in FOLLOW(i) s.t c_k \notin$ INDEP(j). So c_k is locked by c_j. Hence $\exists$ a dipath from(to) c_k to(from) c_j in DAG. Now $c_j \in$ FOLLOW(i), so $\exists$ a dipath from(to) j to(from) c_i in DAG.=> $\exists$ a dipath from(to) c_k to(from) c_i. => $c_k \notin$ INDEP(i). which is a contradiction.

REMARK: Note that if c_i is transferred from one position to the other and

$c_k \notin$INDEP(i) then c_i becomes a foller of cell c_k; So we get the following result.

(R3) If c_i is the base cell and $c_k \in$ complementary block of c_i then dr-area is unchanged iff $c_k \in$ INDEP(i).

Proof: SUPPOSE $c_k \in$ INDEP(i), so $c_k \in$ INDEP(j) $\forall c_j \in$ FOLLOW(i) (by (R1)). Hence no transferred cell will be a follower cell of cell c_k.=>dr-area[k] is unchanged.

conversely, suppose dr-area[k] is unchanged. Hence no transferred cell will be a follower cell of cell(k). => c_i is not a follower of c_k. Hence $c_k \in$ INDEP(i).

So we see that if $\exists$ some transferred cell c_j (from partition P say) s.t $c_k \in$ complement of P and $c_k \notin$ INDEP(j) then dr-area[k] will be changed. But we don't care these cells now because they are locked currently.

Let c_i is the base cell. Call IB(i)= INDEP(i)$\cup\{c_j : c_j \in previouspartitionofc_iandc_j$ is free$\}$.

we now concentrate only on the cells $c_k \in$ IB(i).

UPDATION OF diff(S,T,j)

We now compute the change of diff(S,T,j) for all free cell c_j due to transfer of a c_i.

W.l.g let c_i is transferred from S to T. Then,

Size(S) -> Size(S)-dr-area[i].

Size(T) -> Size-T+ dr-area[i].

Let $c_j \in S\cap$IB(i) [then $c_j \notin$ FOLLOW(i)]. Now if $c_k \in$ FOLLOW(i)$\cap$FOLLOW(j) then c_k will be transferred with c_i. So dr-area[j] will be changed.

Hence new dr-area[j]= dr-area[j]-$\sum_{c_k \in FOLLOW(i) \cap FOLLOW(j)} area[k]$=CH (say).

So diff(S,T,j)=|(Size(S)-dr-area[i])-CH-(Size(T)+dr-area[i])+CH| =|Size(S)-Size(T)-2(dr-area[i])-2CH|

Now for $c_j \in$ T and $c_j \in$ INDEP(i) then dr-area[j] is unchanged. So diff(S,T,j)= |Size(S)-Size(T)-2dr-area[i]+2dr-area[j]|.

After updating diff(S,T,j) we allow only those c_j's for which diff(S,T,j)<=dmax, and set them into appropriate position in bucket.

References:

[1] Subhasis Majumder, Subhas C. Nandy, and Bhargab B. Bhattacharaya, Routing-Driven Hierarchical Floorplan Partitioning Using Staircase Channels

[2] Subhasis Majumder, Routing Driven Floorplan Partitioning Using Staircase Channels.

[3] J.Soukup, Fast maze router, Proc. Design Automation Conference,pp. 100-101, 1978.

[4] F.K. Hwang, An O(nlogn) algorithm for suboptimal rectlinear steiner trees, IEEE Trans. on Circuits and Systems, vol CAS-26, pp. 75-78, 1978.

[5] D.W.Hightower, A solution to the line routing problem on a continuous plane, Proc, 6th Design Automation Workshop, 1969.

[6] C.M.Fiduccia, and R.M. Mattheyes, A linear-time heuristic for improving network partitions, proceedings of the 19-th Design Automation Conference, pp. 175-181, 1982.