第8章　总体布线 (Global Routing)

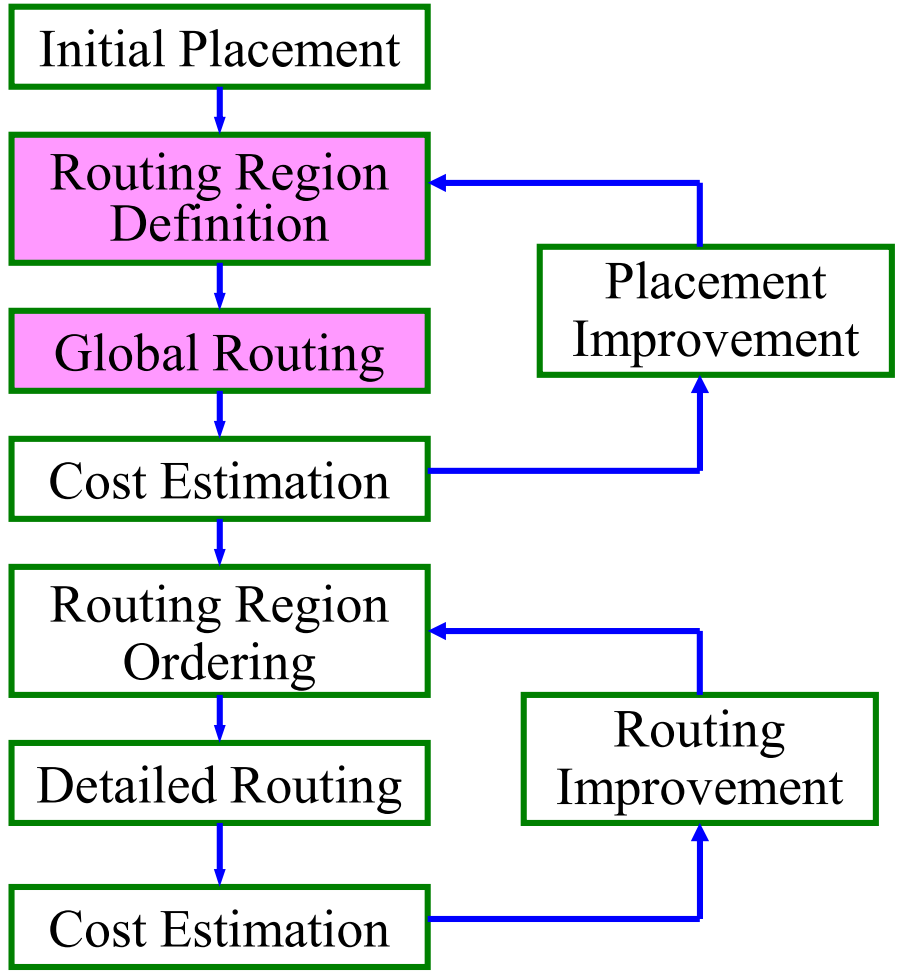
8.4.1 总体布线问题

8.4.2 总体布线算法的分类

8.4.3 总体布线图上的斯坦纳树算法

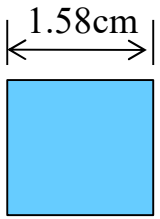
8.4.4 总体布线算法

Placement and Routing Flow Review



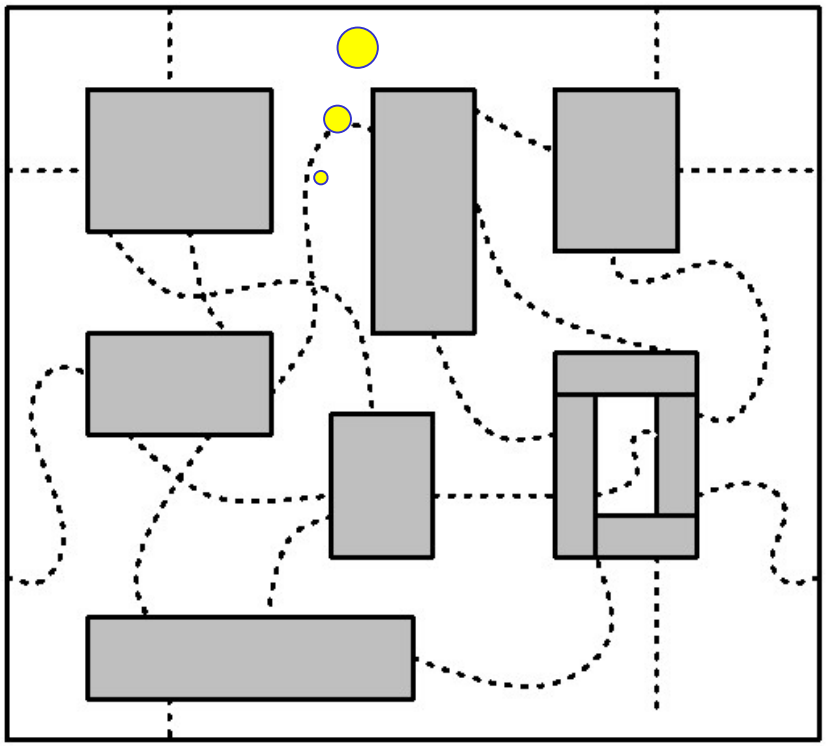
8.4.1 总体布线问题

- 目的:
  - 确定每一条线网的拓扑结构，完成线网走线通道的分配。
- 目标:
  - **保证整个布线设计的布线完成率；**
  - **整个布线设计完成后芯片的面积尽可能小，连线总长度最短；**
  - **关键线网长度最短或芯片时延最小；**



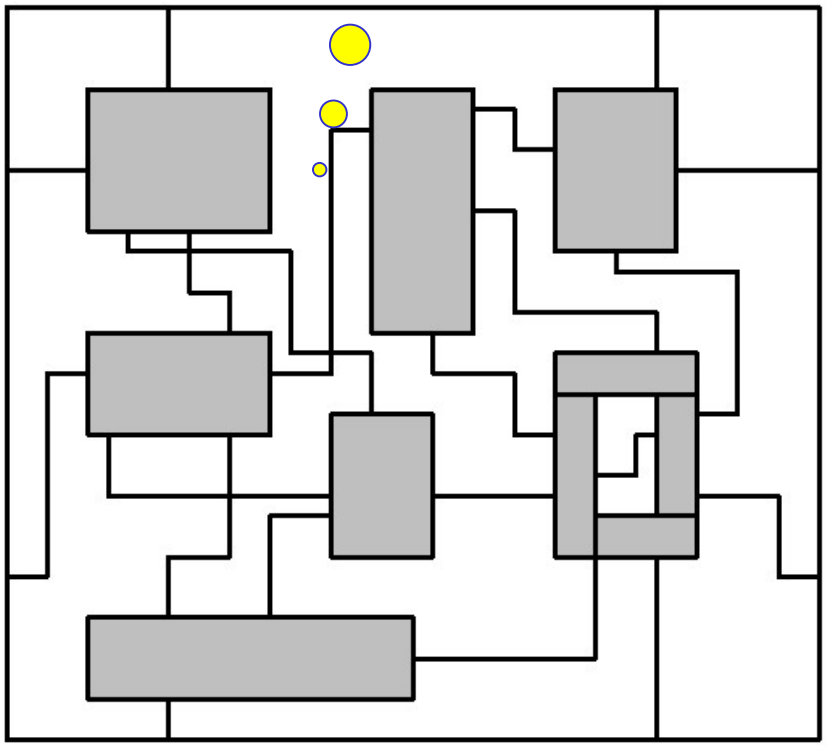
- 面临的挑战:
  - 2.5cm^{2}的芯片70nm工艺下，将达到360,000×360,000轨道数，600×600总体布线区域规模，以及性能需求。

大致路线，
通道级别



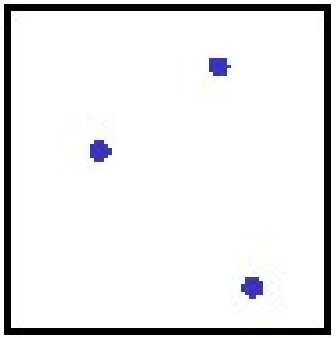
Global Routing
(a)

具体走线，
轨道级别



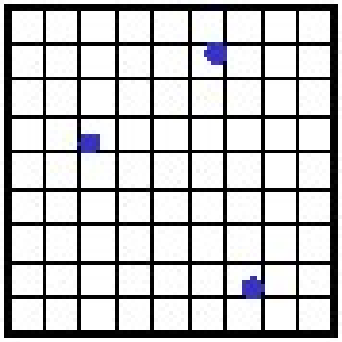
Detailed Routing
(b)

Routing Region



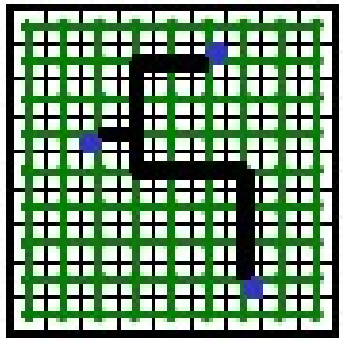
Chip with 1,000,000 nets (showing only three terminals of one net)

Global Routing



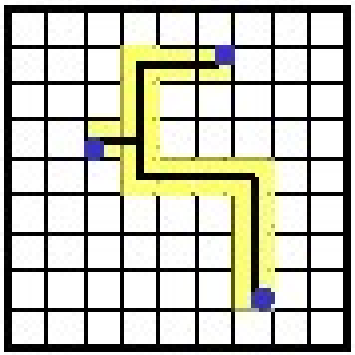
300 × 300 grid graph of "gcells" with capacities

Capacities take pre-routed nets and blockages into account.



Find Steiner trees in grid graph for all nets regarding capacities.

Detailed Routing

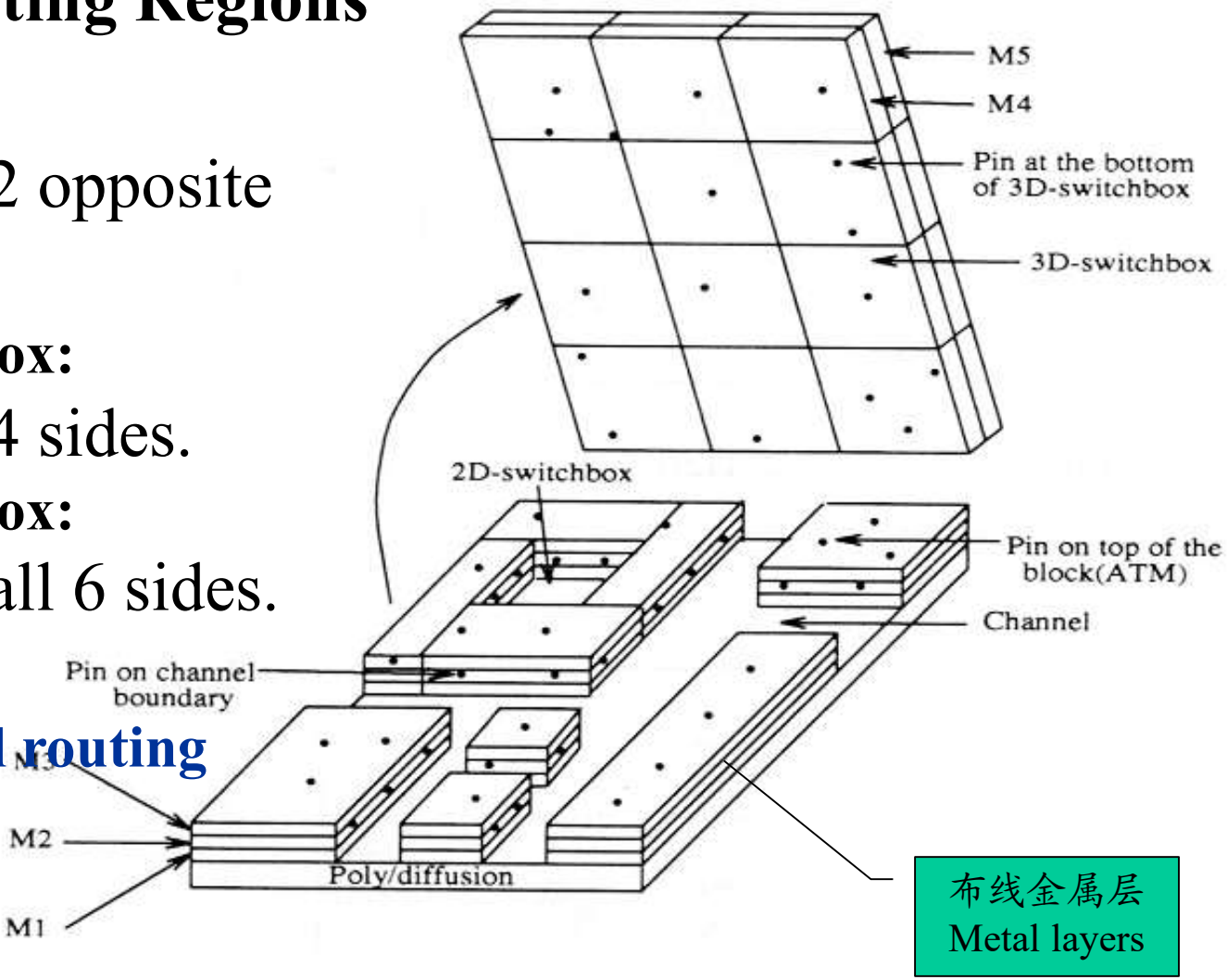


Detailed routing is performed in small "sboxes"

布线问题

3 Types of Routing Regions

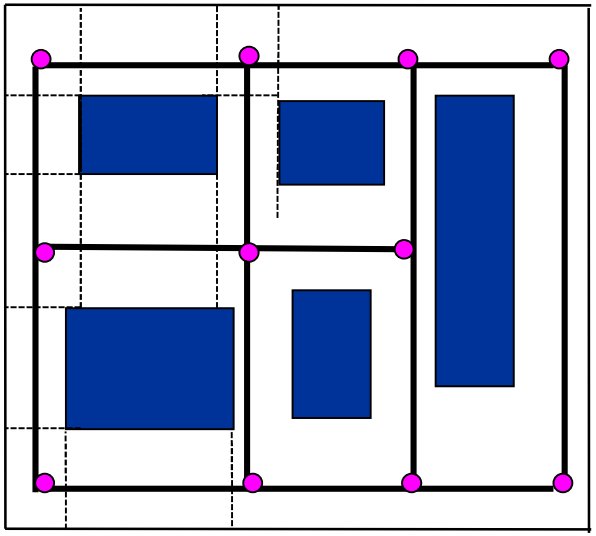
- **Channel:**
  - ➢Pins on 2 opposite sides.
- **2-D Switchbox:**
  - ➢Pins on 4 sides.
- **3-D Switchbox:**
  - ➢Pins on all 6 sides.
- **Over the cell routing**

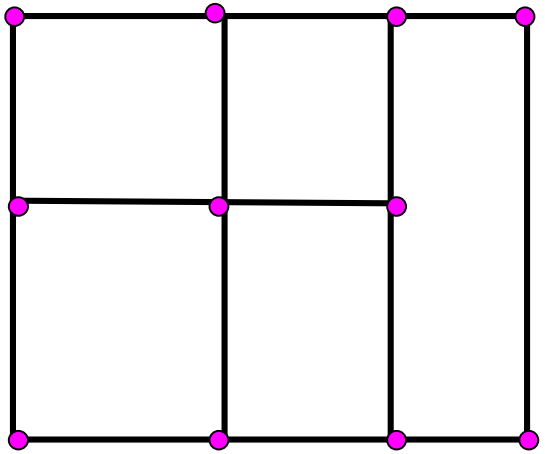


8.4.1.1 总体布线图

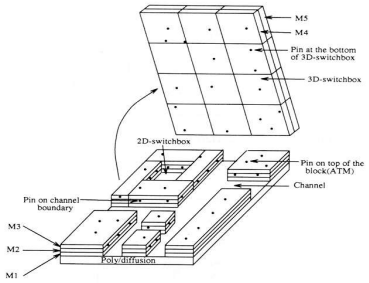
定义6.1：G＝(V, E)是一个布图规划图，其中$\forall v_i \in V$表示版图中的一个通道与另一个通道的相交点；若$\forall v_i$, $v_j \in V$，且$\exists e_{ij} \in E$是图G上对应的一个布线通道段，则称v_i与v_j在布图规划图G 中相邻。

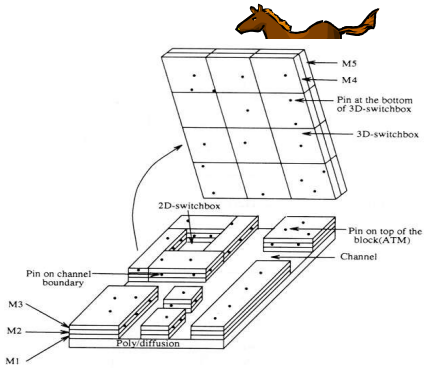
若$e \in E$，假设 c(e)、l(e)分别表示边e相对应的通道容量和长度.





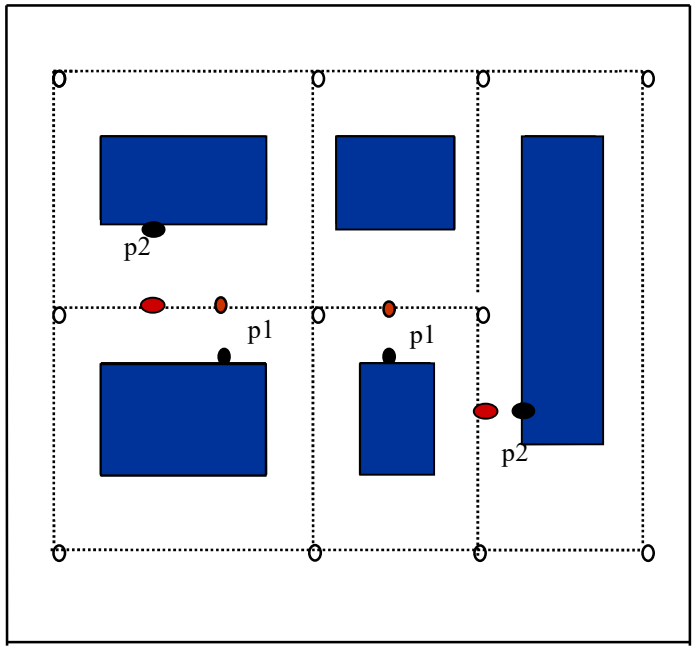
布图规划图



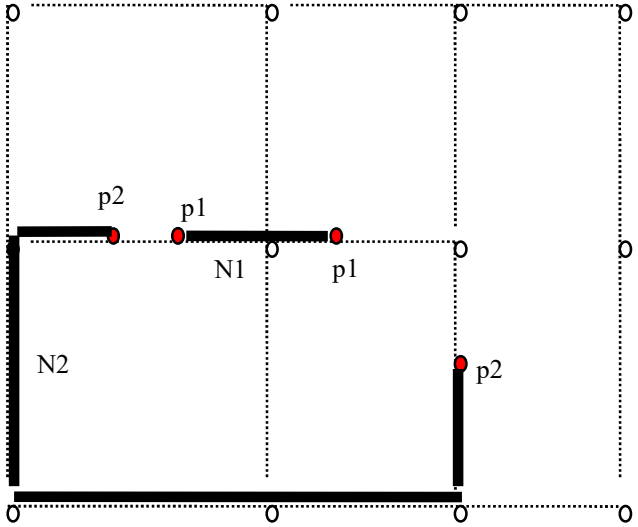


总体布线图：

- ❖ 带有布线需求信息的总体布线图；
- ❖ 需要在总体规划图上，完成线网端点到规划图节点的映射；
- ❖ 必要时增加新的规划图节点，用于表示线网端点。



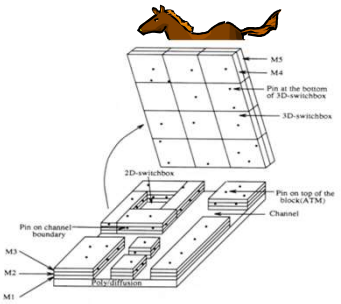
布图规划图

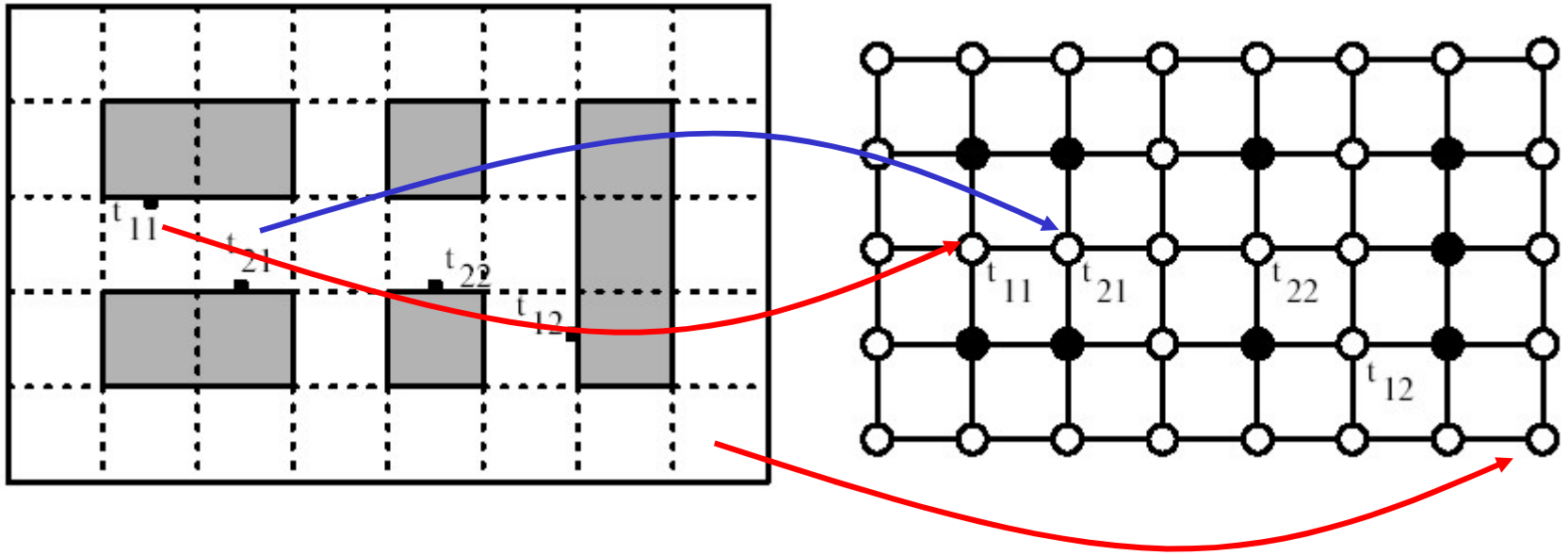


总体布线图建模

Global Routing Graph Model

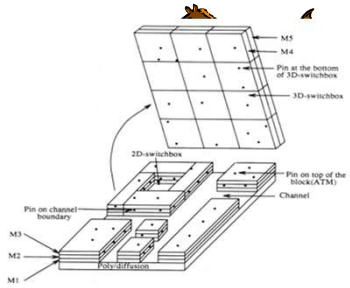
Grid Graph Model



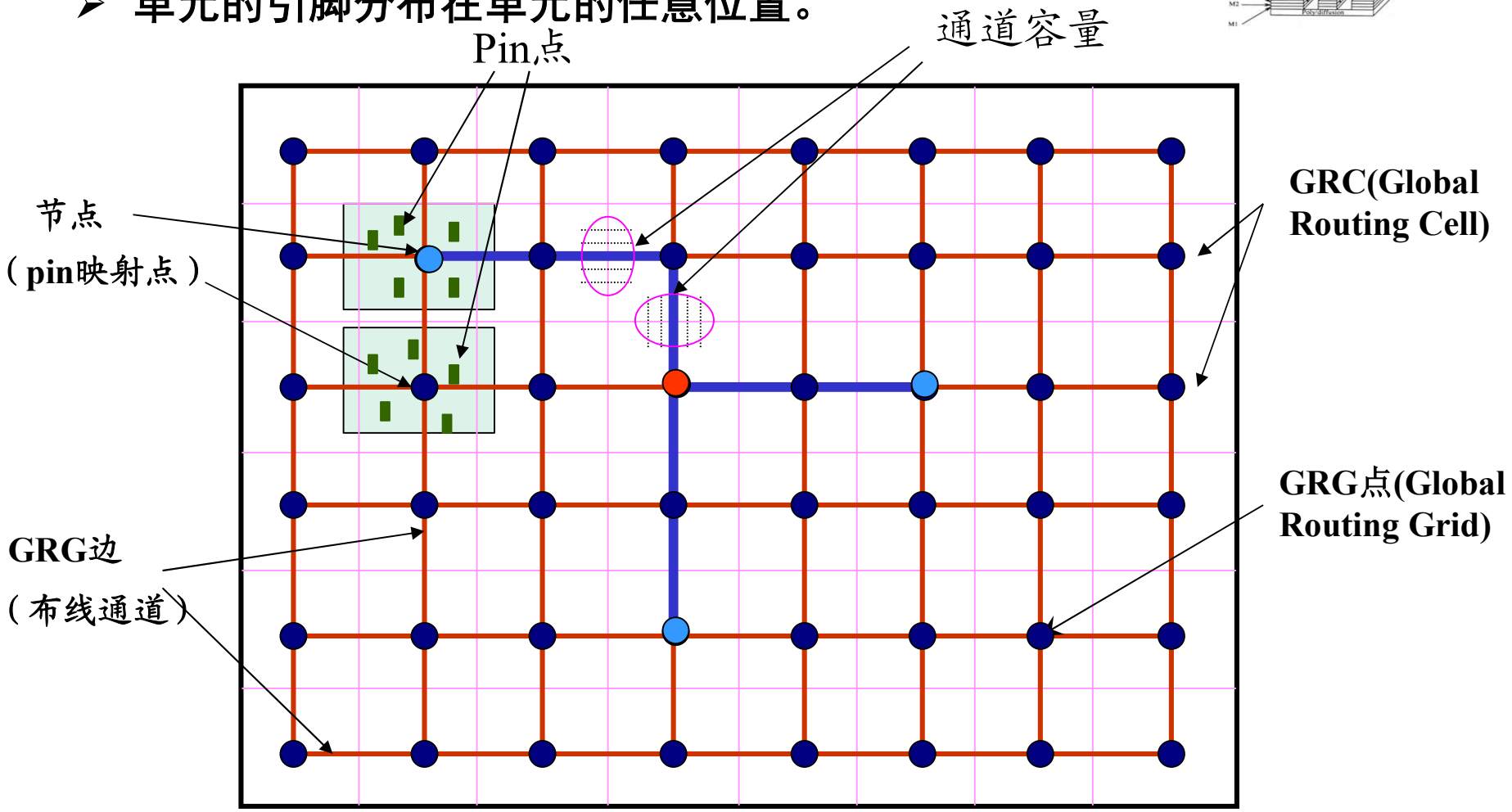


- Each cell is represented by a vertex.
- Two vertices are joined by an edge if the corresponding cells are adjacent to each other.
- The occupied cells are represented as filled circles, whereas the others are as clear circles.

多层布线工艺布图模式:

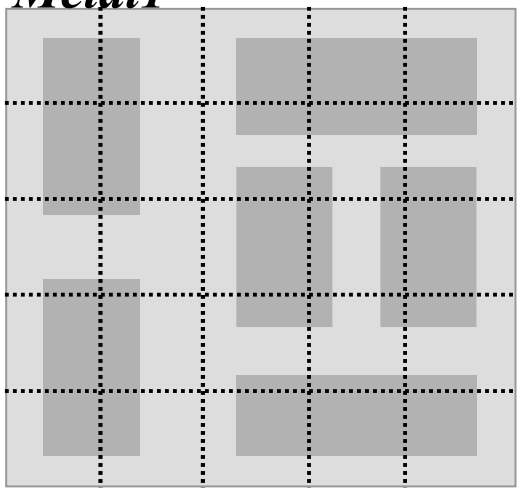


- 单元的引脚分布在单元的任意位置。

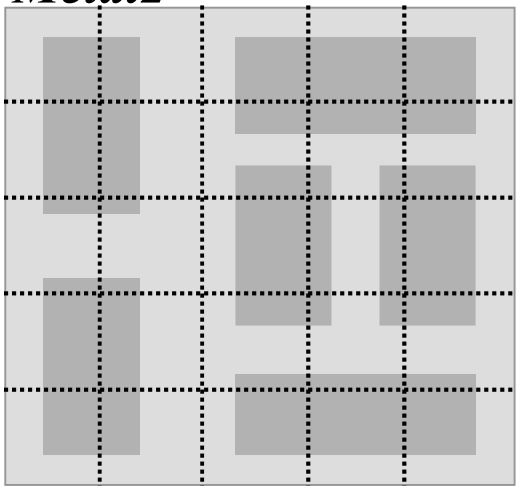


Gcells (Tiles) with macro cell layout

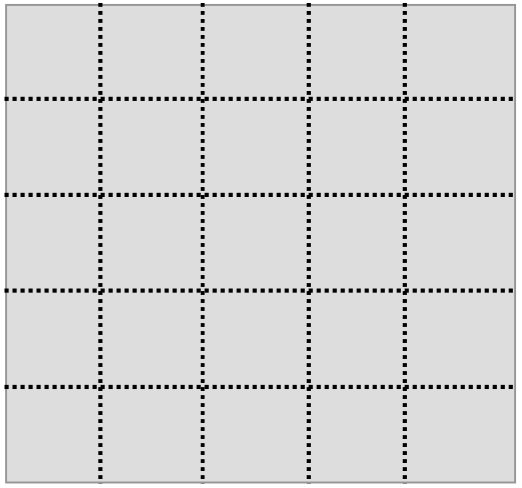
Metal1

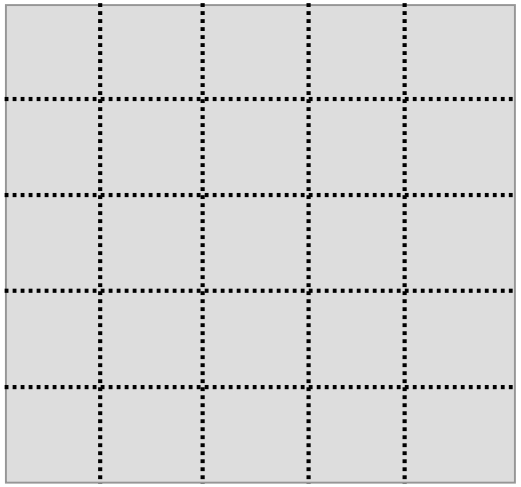


Metal2



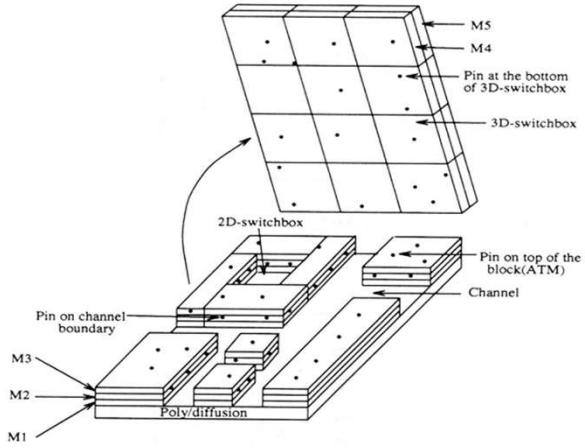
Metal3





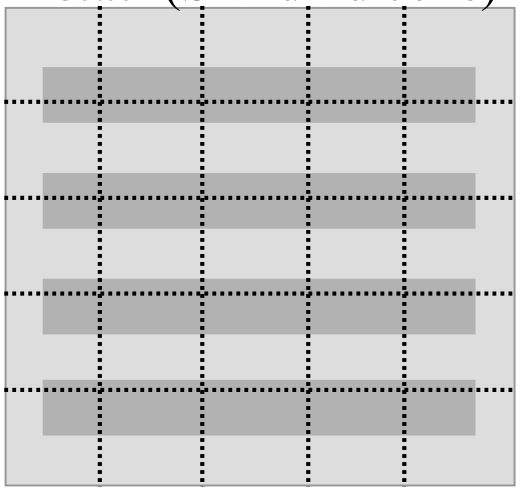
Metal4

etc.

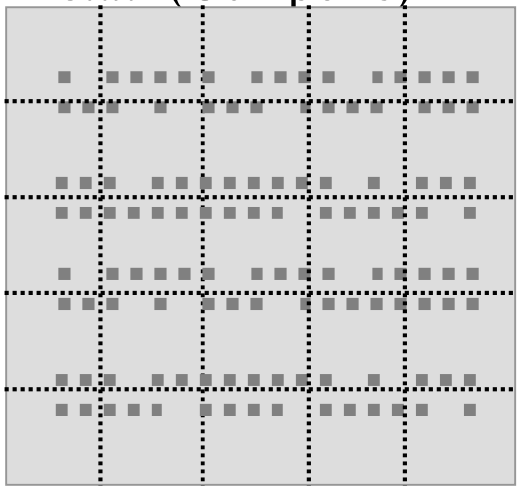


Gcells (Tiles) with standard cells

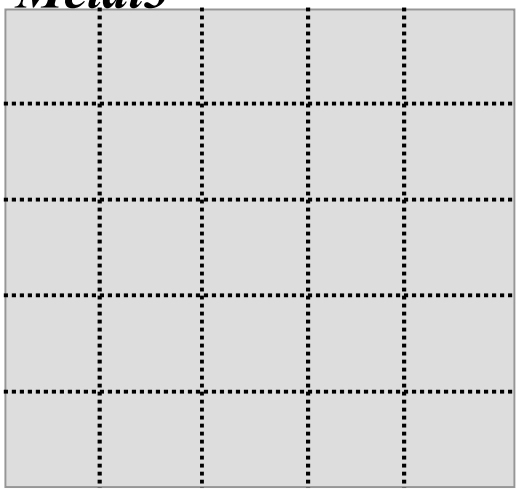
Metal1(Standard cells)

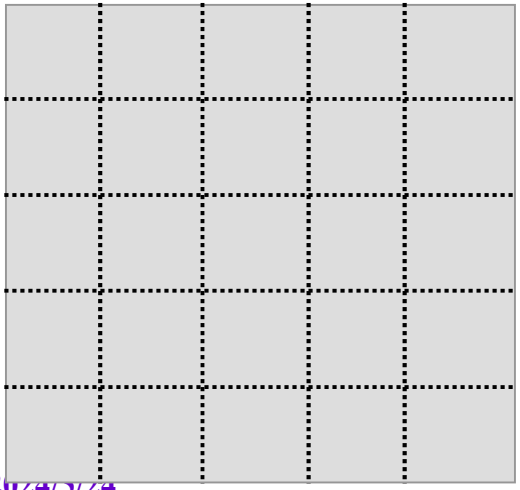


Metal2(Cell ports)



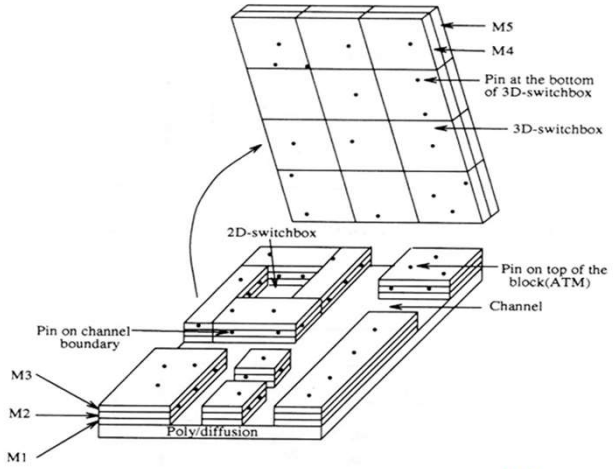
Metal3



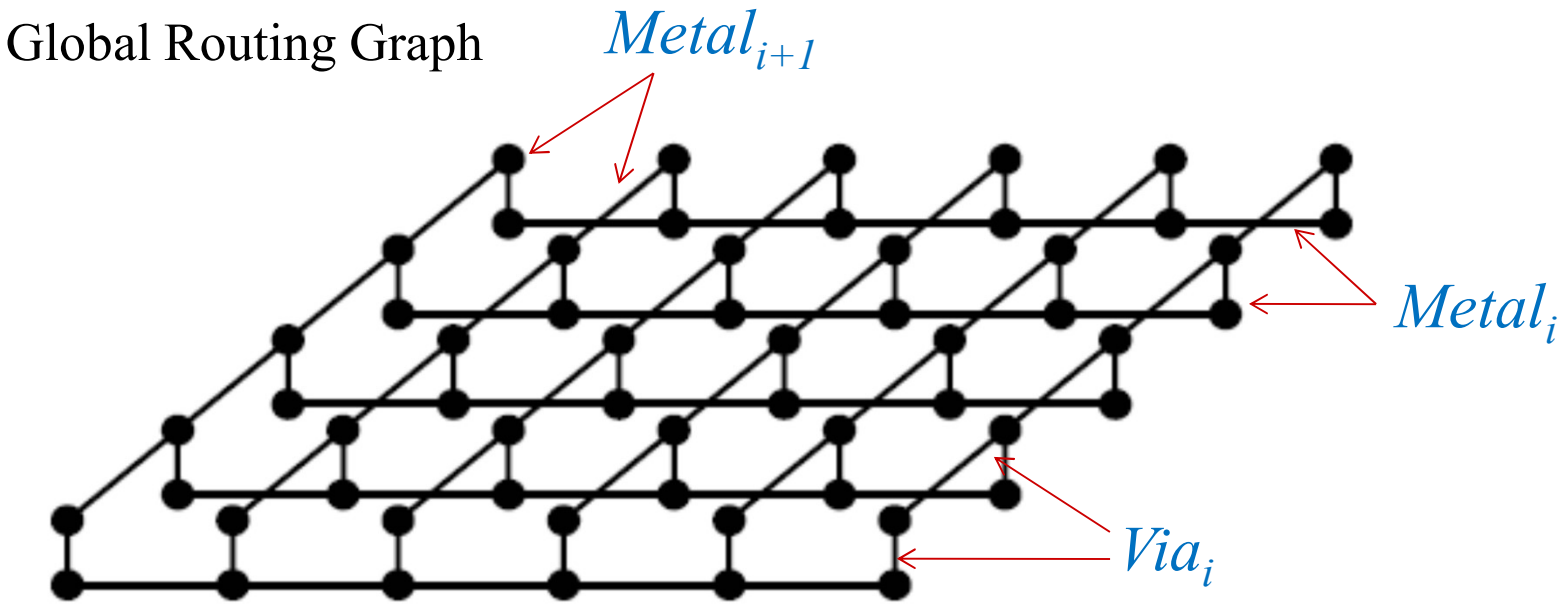


Metal4

etc.



Grid Graph Model – 3D



Planes (= layers) and edges between planes to model via costs

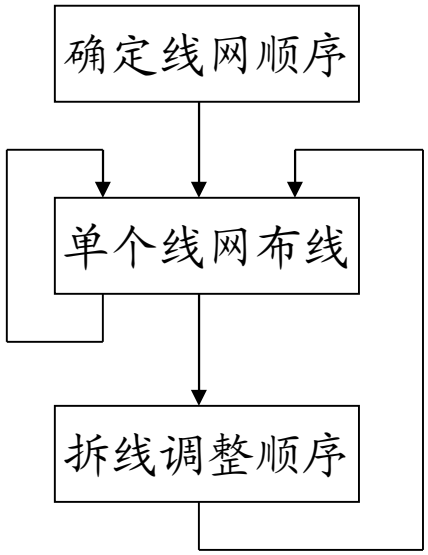
- Sometimes all layers can be flattened into one *H* layer and one *V* layer, but this is less common today

8.4.2 总体布线算法的分类

- 总体布线算法分类:
  - 从布图模式或从对象分类:
    - 门阵列;
    - 标准单元;
    - 积木块BBL方法;
    - 多层布线模式。
  - 从算法优化目标分类:
    - 以总线长最小为目标（芯片面积最小);
    - 以最大布线密度最小为目标（布线均匀);
    - 以性能优化为目标。
  - 从算法本身的性质分类:
    - 串行算法;
    - 并行算法。

8.4.2.1 串行算法

- 布线策略:
  - 按照设定的顺序，对每个线网逐一进行布线，一次只完成一条线网的布线;
  - 在新的布线资源情况下，进行线网布线;
  - 存在布线结果对线网布线次序的依赖性问题（Order Dependent Problem);
  - 新的线网遇到堵塞无法布通时，采用拆线的策略，拆除部分已布线网，调整剩余布线顺序重新布线;
  - 存在布线设计收敛，以及布通率问题。



- 确定线网布线顺序的原则:
  - 关键性高的线网先布;
  - 引脚数少的线网优先于引脚数多的网;
  - 覆盖区域小的线网优先于覆盖区域大的网。

- 串行布线用到的算法:
  - 两端线网算法:
    - 迷宫算法（Maze routing）
    - 线搜索算法（Line-probe）
    - 最短路径算法（Shortest path）
    - 模式布线（Pattern Routing）
  - 多端线网算法:
    - 斯坦纳树算法（Steiner tree）
    - 带有拥挤度分析的最小代价Steiner树的串行布线算法。

8.4.2.2 并行算法

- 解决面临的两个关键问题:
  - 布线通道拥挤的不可预见性;
  - 布线结果对布线顺序的依赖性。

- 常见的并行算法:
  - 层次式整数规划方法;
  - 网络流方法;
  - 层次布线算法。

8.4.3 总体布线图上的斯坦纳树算法

- 总体布线阶段面临的问题是进行线网的互连优化。
  - 最基本的任务是为每一条线网寻求最优化的拓扑树结构，实现线网连线长度最短（或其它目标最优）。
- 两种基本的互连拓扑优化技术
  - 最小生成树（Minimum Spanning Trees，MST）

    Kruskal's algorithms 和 Prim's algorithms

    ——复杂度在$O(nlogn)$
  - 最小斯坦纳树（Steiner Minimal Trees，SMT）
- **采用矩形网格总体布线图上的斯坦纳树生成算法**

定义8.4.5: 一棵斯坦纳树ST中，若ST树中的每条边均是直角矩形边，则此ST被称为**矩形斯坦纳树RST** (Rectilinear Steiner Trees)。

♦VLSI设计中的斯坦纳树问题

- ❖ 给定一条互连线网的端点集合P，在版图上寻找一个斯坦纳点的集合S，使得由节点集合（$P \cup S$）形成的最小生成树的代价最小。

♦VLSI中的斯坦纳树问题同样是NP-hard问题

- ❖ 只能通过寻求近似算法求解斯坦纳树问题。

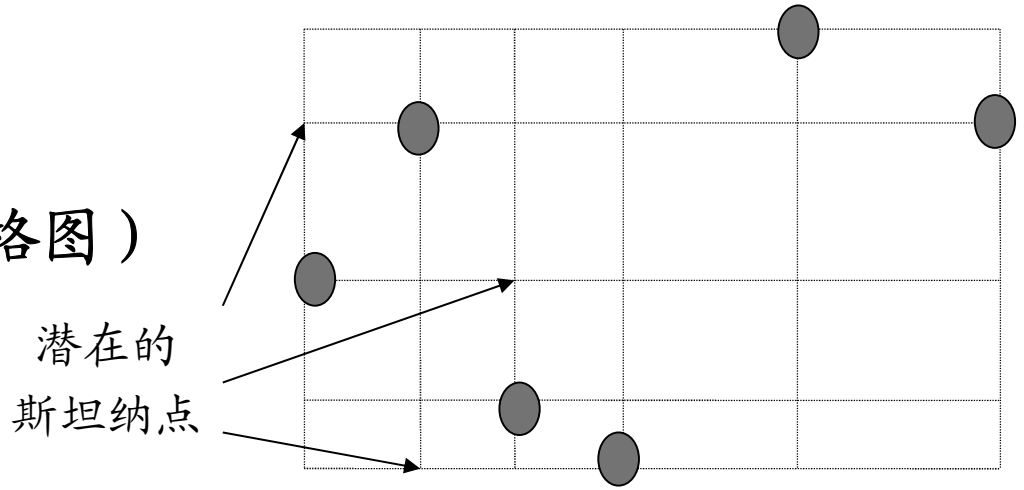
VLSI设计中斯坦纳树的特性（一）

♦Hanan点现象

❖ Hanan于1966年研究发现，以线网的每一个端点为基点，画水平和垂直坐标线，形成的网格称为Hanan网格，网格的交点称为Hanan点。那么，线网的斯坦纳点将会出现在这些Hanan点上。

♦Hanan网格

（也称为基础网格图）



VLSI中斯坦纳树的特性（二）

- **最小生成树MST树与最小斯坦纳树SMT树**
  - Hwang于1976年研究发现，矩形MST树长度与矩形SMT树长度的比值不会超过3/2。

$$\frac{L_{MST}}{L_{RSMT}} \leq \frac{3}{2}$$

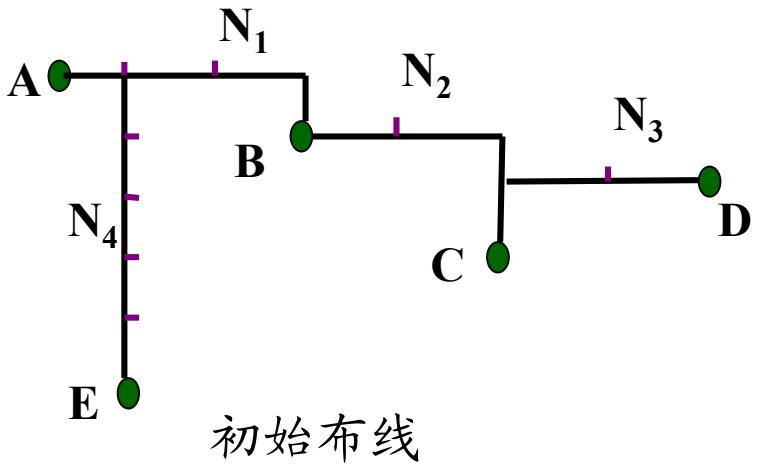
  - 基于这个现象，最小生成树MST成为很多斯坦纳树启发式算法的基础。
  - 可以通过最小生成树的矩形化来获取斯坦纳树的初始构形。

8.4.3.1 基于最短路径树的斯坦纳树算法

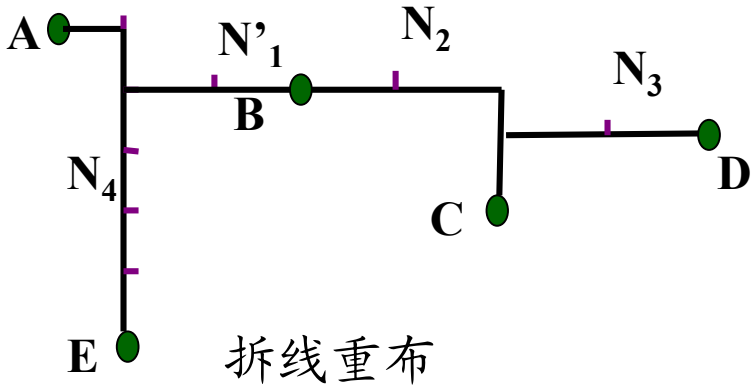
- 算法思想
  - 利用最短路径算法求解两个端点之间的布线路径;
  - 多端线网的布线问题可分解成一系列两端线网的布线问题。这样，多端线网的斯坦纳树问题就变成一系列两点间的最短路径问题。
- 总体布线图上的最短路径问题
  - 在总体布线图G=(V，E，W)上，相邻顶点v_i，v_j的边e_{ij}的长度为w_{ij}，在G图上寻找连接指定两点的一条最短路径。
  - 算法的时间复杂度为$O(N^2)$。
- 求解方法
  - 迷宫算法;
  - 线探索法。

算法示例说明:

- 用改进的两点路径算法;
- 从某个端点出发，每次只连接一个端点;
- 用已经完成的连接路径作为起始源点，向目标点进行扩展;
- 用拆线重布的方法改进连接路径质量。



初始布线

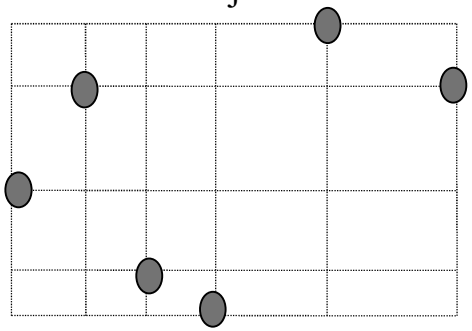


拆线重布

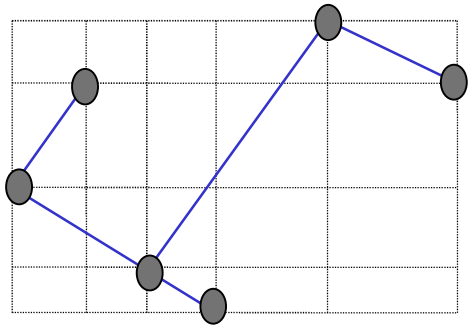
8.4.3.2 基于最小代价生成树的斯坦纳树算法

- 算法思想
  - 利用最小代价生成树算法求解多端点线网的MST树;
  - 对所求出的MST树的每一条边进行矩形化。
- 算法说明

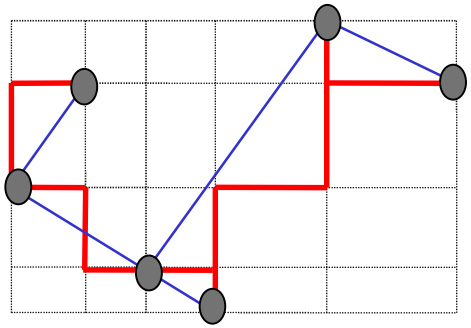
S为待布线网，G(S)为基础网格图，MST是其最小生成树，对每条边的矩形转化产生一棵近似的MRST树。如果树T的一条边e_{ij}是在G(S)上通过v_i,v_j间最短距离的矩形边化，称为**阶梯式边(Staircase)布图**。



待布线网S和G(S)

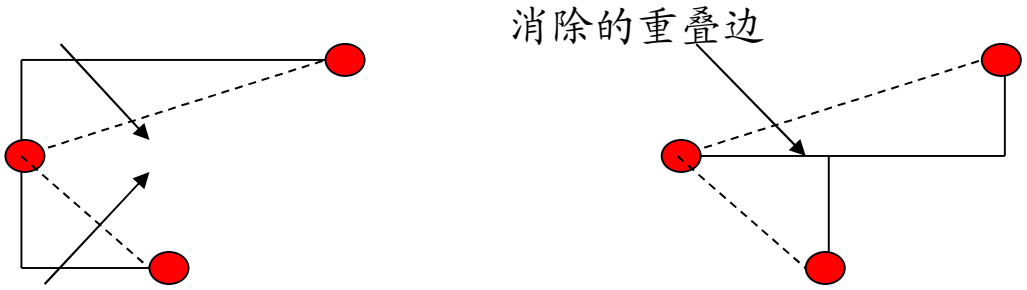


最小生成树MST

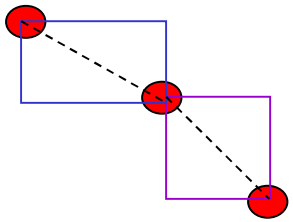


矩形化斯坦纳树

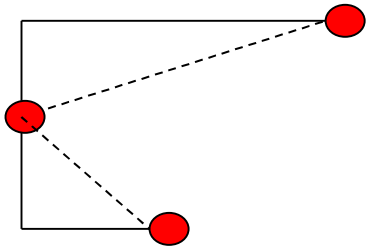
- 边矩形化过程中重叠边消除效果
  - 同一条MST边，存在多种矩形化方式;
  - MST边的矩形化将会产生一定的边重叠;
  - 重叠边的消除将会使得所产生的矩形化斯坦纳树RST的线长减少。
- 利用重叠边合并操作，生成RST树算法思想
  - 求解最小生成树;
  - 不断寻求有最大重叠的MST边矩形化操作。

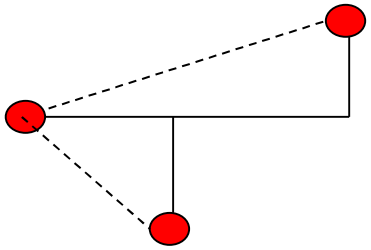


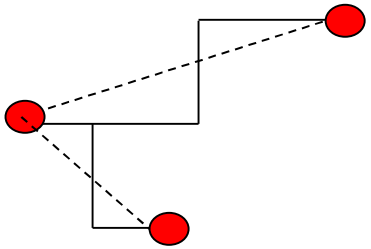
重叠消除时的边形状组合问题:



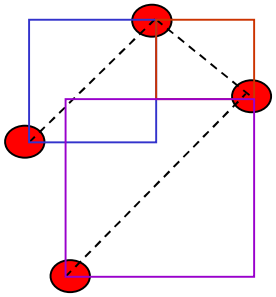
两条边互不相关

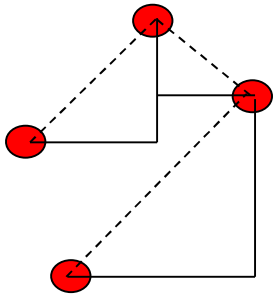


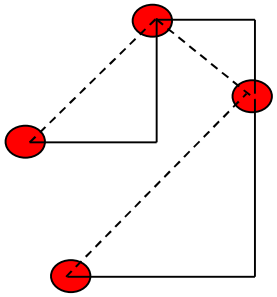


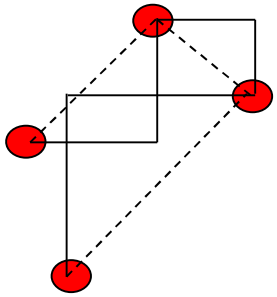


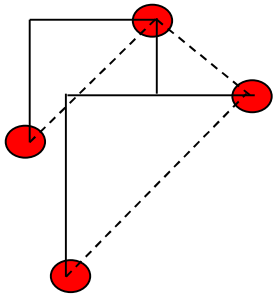
两条边相互关联

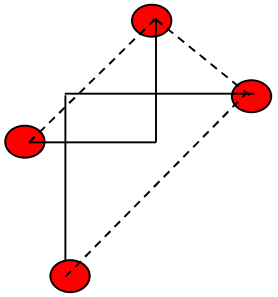












三条边相互关联

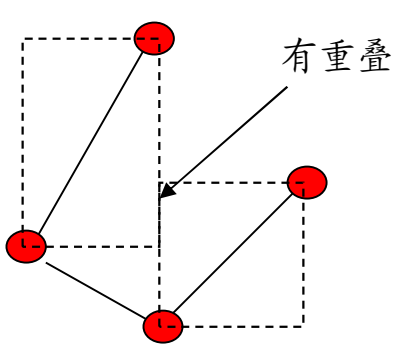
8.4.3.3 基于可分离性的RST树算法

- **Ho于1990年提出通过边的合并产生优化RST树的算法**
  - 当MST具有**可分离性**性质时，可从MST得到一棵优化的S-RST树。

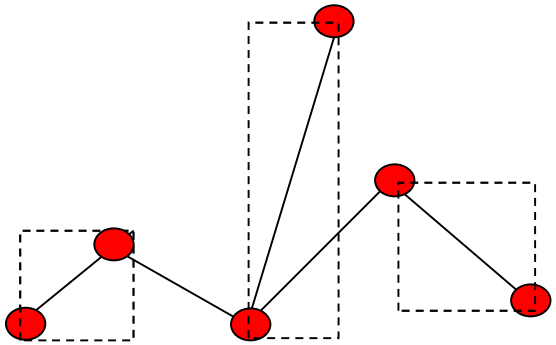
J. M. Ho, G. Vijayan and C. K. Wong, A New Approach to the Rectilinear Steiner Tree Problem, IEEE Trans. on CAD, 1990, 9(2), 185-193

定义8.4.8：若在阶梯式布线中，两条不相邻的边的布线结果不会产生相交或重叠，则称这两条边是**可分离的**。

定义8.4.9：若MST中的所有不相邻的边两两均满足可分离性，则称这棵MST树是**满足可分离性的MST**，记为**SMST**。



不可分离MST树



可分离MST树

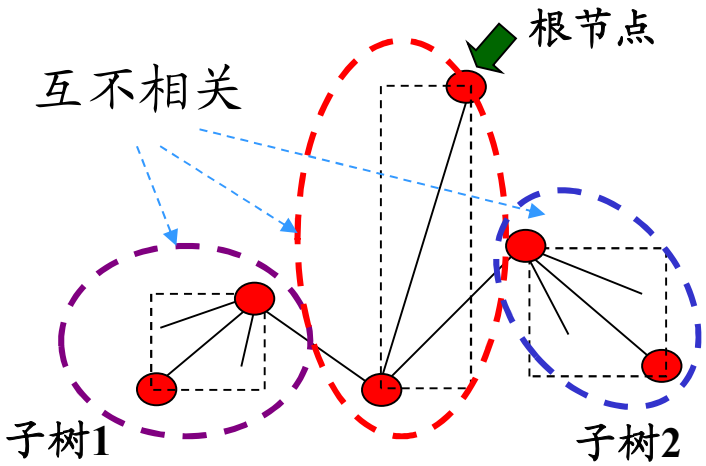
- **满足可分离性MST树是可构造的**
  - 用（ *dist(i, j), -|y(i)-y(j)|, -max(x(i), x(j))* ）作边权;
  - 利用改进的Prim算法可在MST上求得满足可分离性的SMST树;
  - 线长没有变化。
- **RST树构造算法分两步完成:**

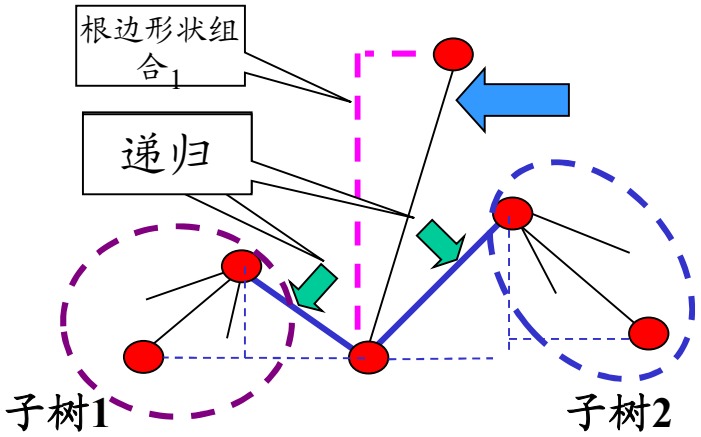
  第一步：针对已给定的一条线网N_p，通过应用改进的Prim算法，在$O(|N_P|^2)$时间复杂度下构造一棵满足可分离性的SMST树;

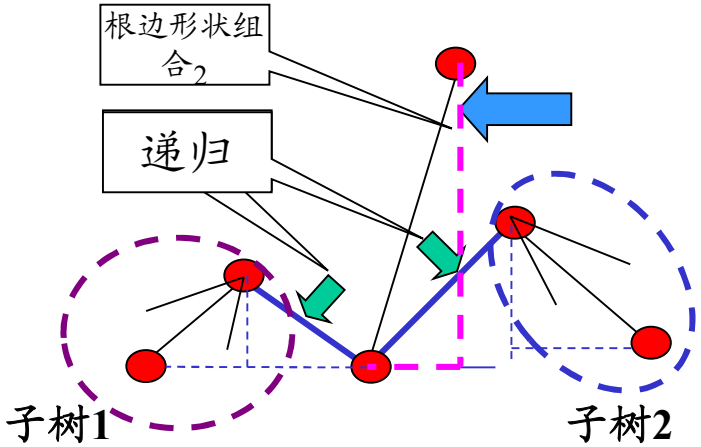
  第二步：在已有的SMST基础上，获取一棵优化的Z-RST树。
- **一棵最优的Z-RST树是一棵最优的S-RST树。**

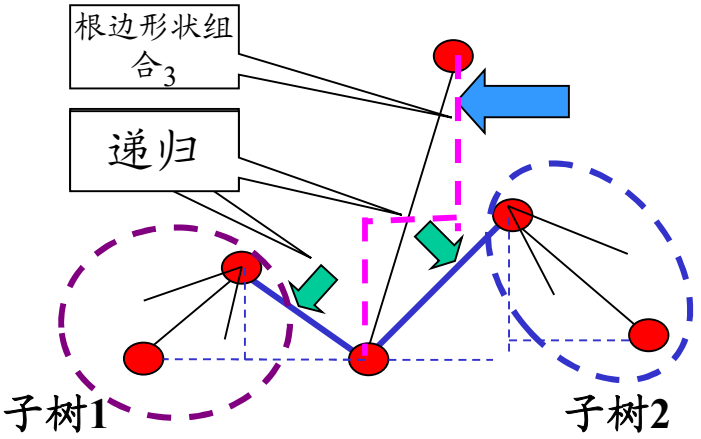
J. M. Ho, G. Vijayan and C. K. Wong, A New Approach to the Rectilinear Steiner Tree Problem, IEEE Transactions on CAD, 1990, Vol. 9 No.2, pp: 185-193

满足可分离性MST树的形状组合与递归









Z-RST树构造算法描述

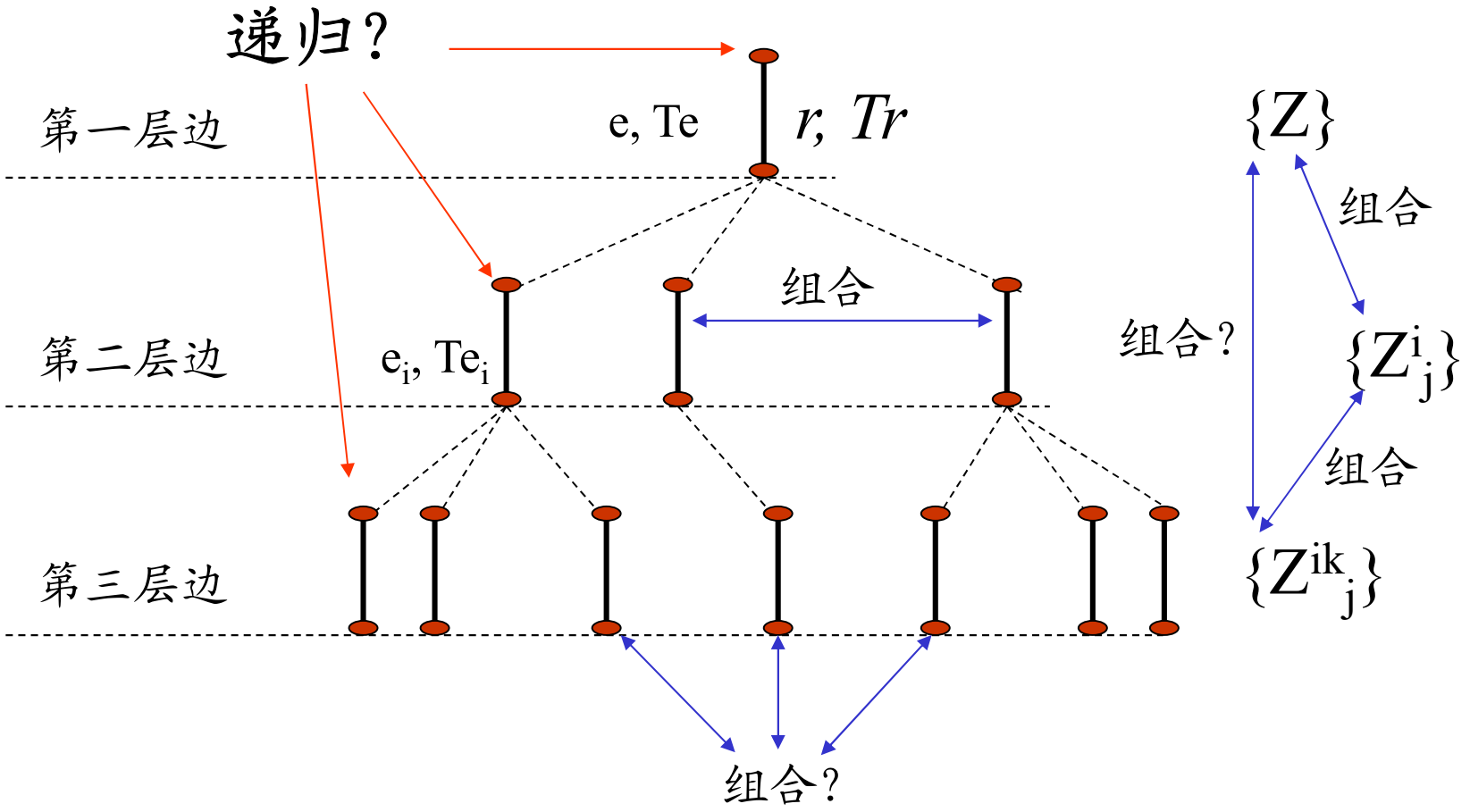
❖ 算法从SMST树的任一条边 r 开始，求解满足可分离性树 Tr 的最优化SMST树 M。

```
Algorithm Z-RST( r, Tr, CostM, M )
  input: r, Tr;                    /* 线网的SMST树Tr，以及指定的起始边r;
  output: CostM, M;                /* 所求的最优SMST树M，及其最小代价CostM;
begin
  CostM=∞;
  for each Z_shaped layout Z of r do  /* 对边r的每一种Z形状循环求解;
     LEAST_COST( Z, Tr, CostTempM, TempM );
                          /* 求解边r当前Z形状下的最优SMST树;
     if CostTempM<CostM then {  /*寻找边r的所有可能Z形状最小解;
              M=TempM;
              CostM= CostTempM;}
  endfor;
end
```

递归求解

```
Function LEAST_COST(Z, Te, CostMz[e], Mz[e])
  input: Z, Te;                    /* 给定起始边的形状Z，以及起始边表示的可分离树Te。
  output CostMz[e], Mz[e];         /* 对应该形状起始边的最佳Z形树Mz[e]，以及代价。
begin
   if CHILD_EDGES_NUM(e)≠0 then { /* 计算每棵子树的最小Z形树
     for each child edge ei of e do      /* 每棵子树对应的边
       for each layout Zij of ei do      /* 每一种Z形状
          LEAST_COST(Zij, Tei, CostMZij[ei], Mzij[ei])  /* 子树递归
       endfor;
     endfor;
     for (each combination of the layouts containing one optimal Z-RST layout for each Tei } do
        merge layouts in the combination with the layout Z of edge e;     /* 所有子树的组合
        calculate the resulting cost of merged layout;  /*与根边的Z形合并
     endfor;
     CostMz[e]=minmum cost among all merged layouts;  /* 取最小的合并结果
     Mz[e]=the layout corresponding to the minimum cost;}
   else (* The bottom edge is reach *){
        Mz[e]= Z;
        CostMz[e]=cost of Z;}
end
```

利用改进的Prim算法求出的SMST树的分层结构

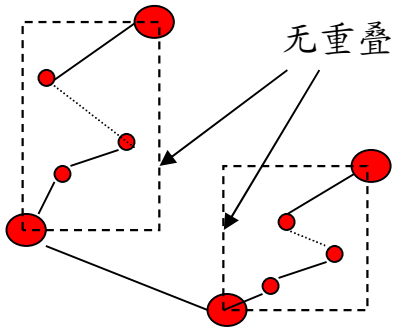


♦ 算法思想回顾

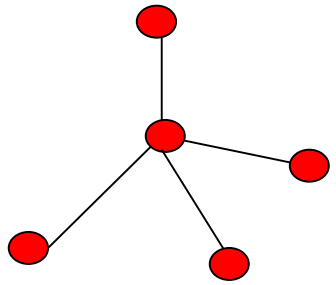
❖利用满足可分离树性质，减小计算规模

- “任意两条不相邻的边，布线结果不会产生相交或重叠”
- RMST树中任意两条边：不相邻，或连接同一节点；
- 布线时，不相邻的边不需要进行形状上的组合优化；
- 减少布线时的组合优化传递现象，只限定在相邻边的组合。

❖利用树的层次结构，采用递归方法实现动态规划求解策略

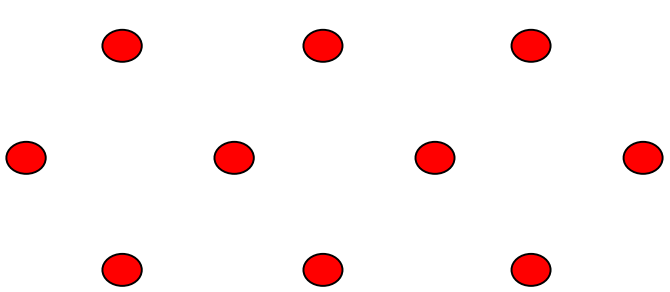


分布不同子树

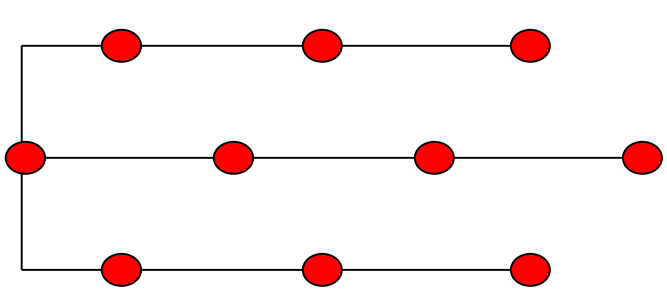


同一节点连接的边

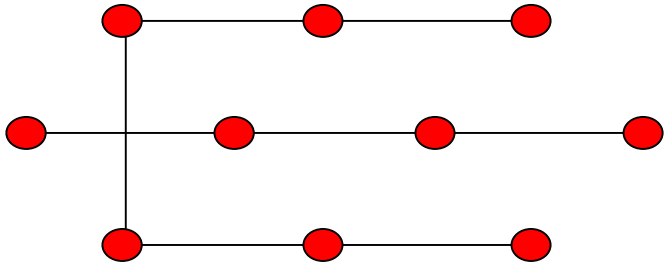
- 算法性能
  - 平均线长改进9%;
  - 存在局部最小解。



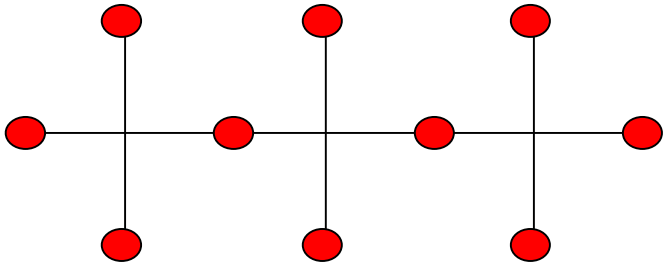
待布树



可分离MST树



局部最佳SMST树

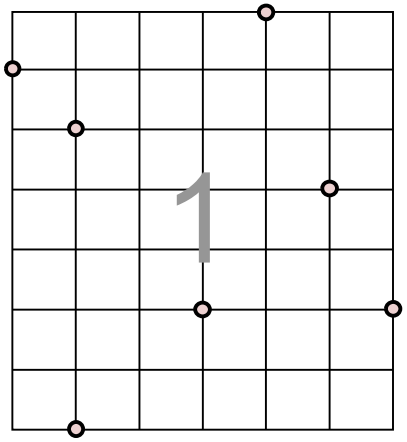


最优SMT树

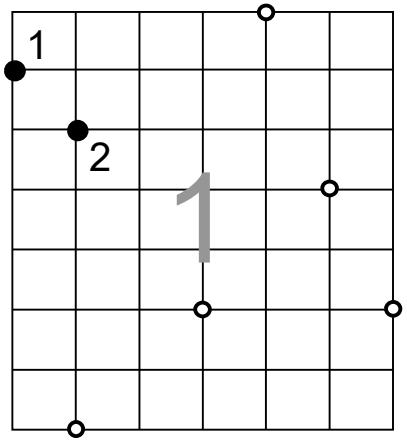
8.4.3.4 A Sequential Steiner Tree Heuristic

1. Find the closest (in terms of rectilinear distance) pin pair, construct their minimum bounding box (MBB)
2. Find the closest point pair (p_{MBB},p_C) between any point p_{MBB} on the MBB and p_C from the set of pins to consider
3. Construct the MBB of p_{MBB} and p_C
4. Add the *L*-shape that p_{MBB} lies on to *T* (deleting the other *L*-shape). If p_{MBB} is a pin, then add any *L*-shape of the MBB to *T*.
5. Goto step 2 until the set of pins to consider is empty

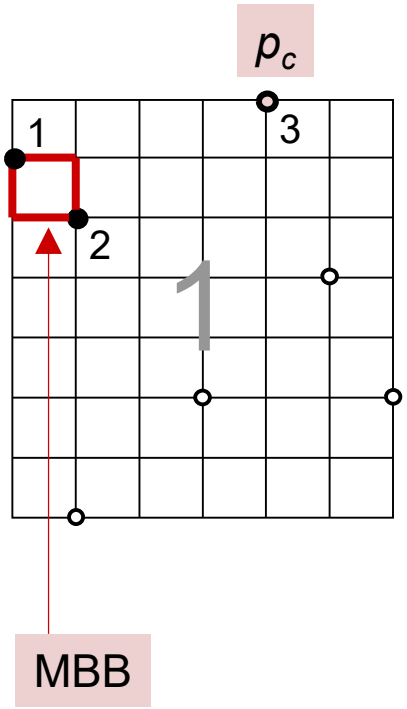
Example Sequential Steiner Tree Heuristic



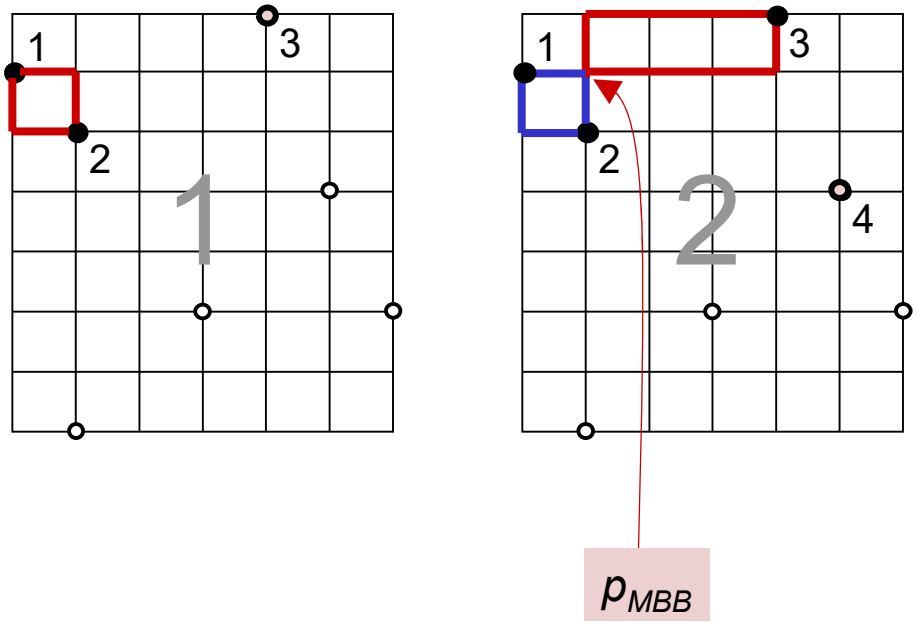
Example Sequential Steiner Tree Heuristic



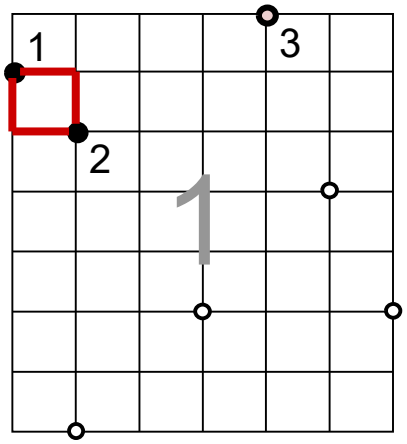
Example Sequential Steiner Tree Heuristic

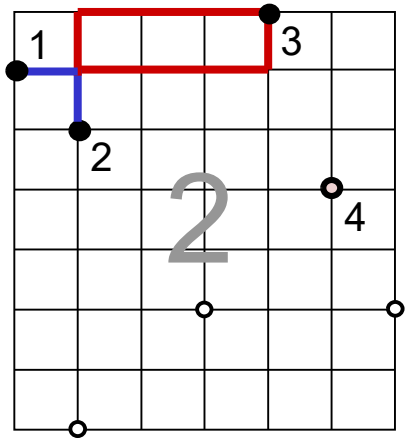


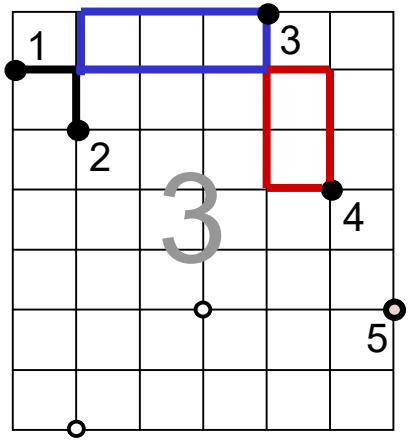
Example Sequential Steiner Tree Heuristic



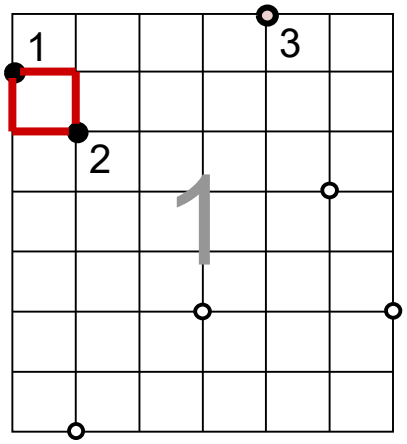
Example Sequential Steiner Tree Heuristic

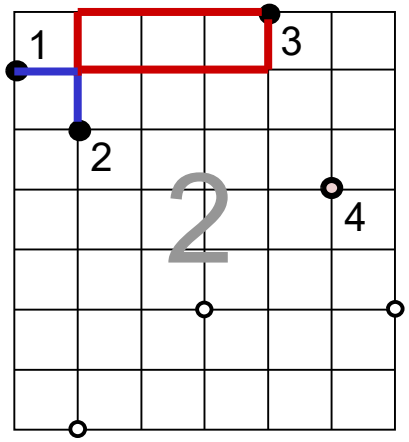


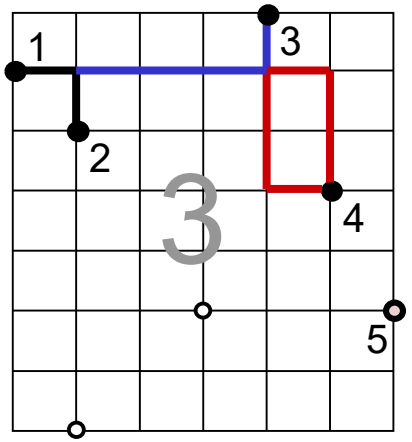


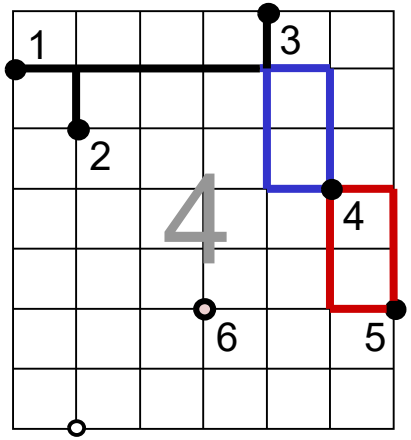


Example Sequential Steiner Tree Heuristic

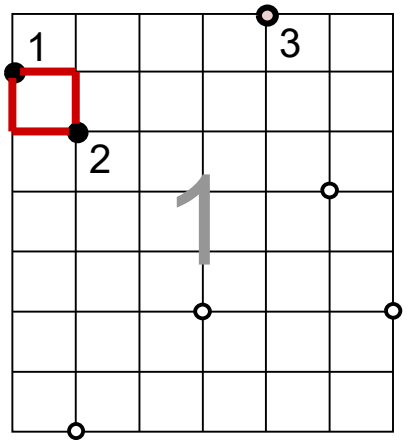


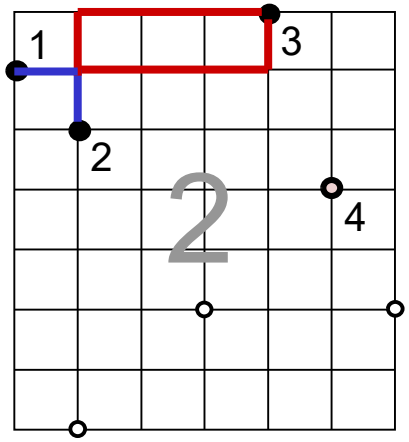


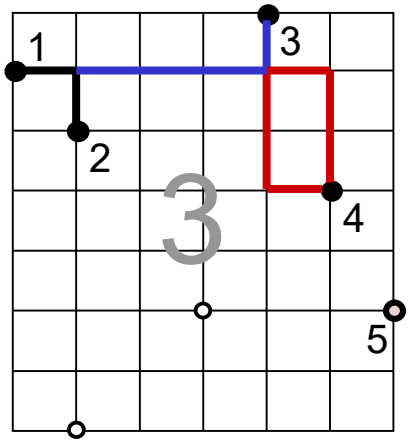


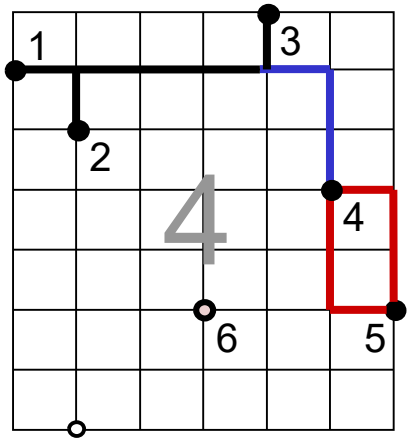


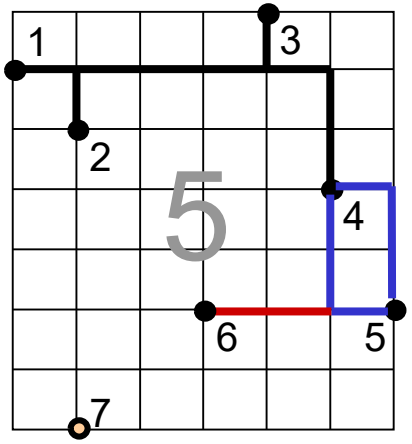
Example Sequential Steiner Tree Heuristic



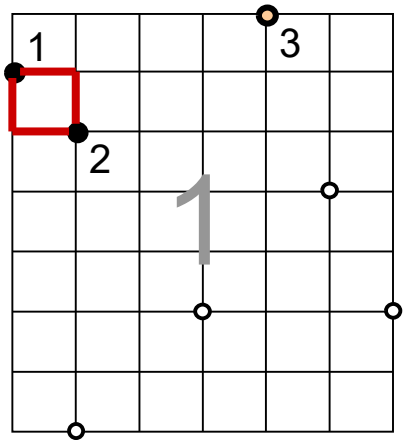


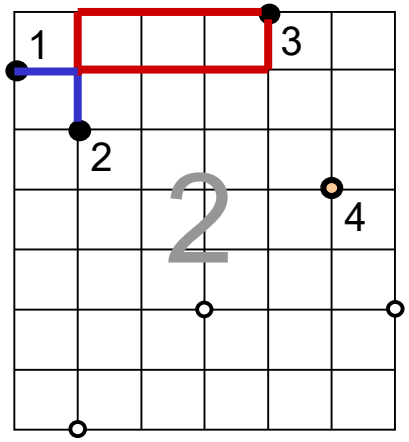


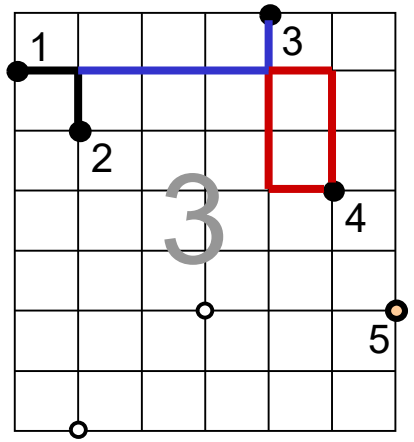


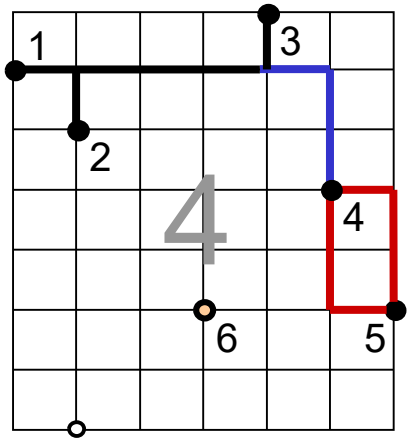


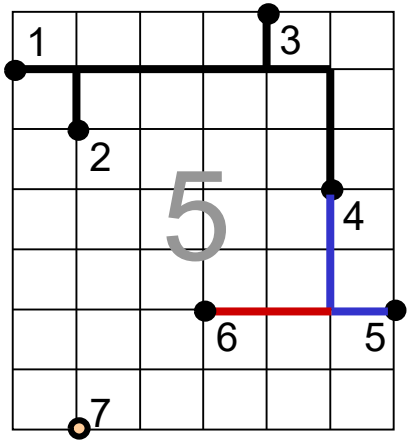
Example Sequential Steiner Tree Heuristic

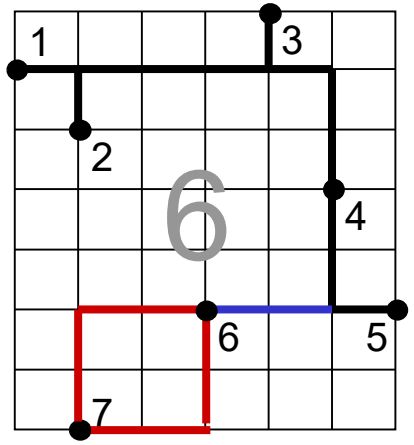




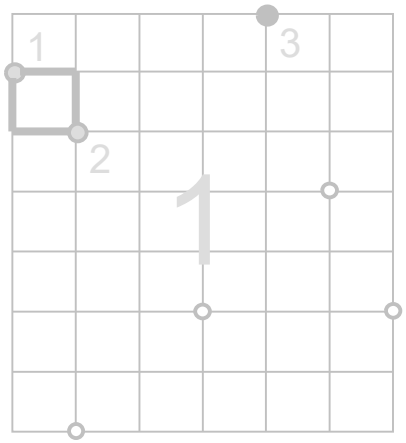


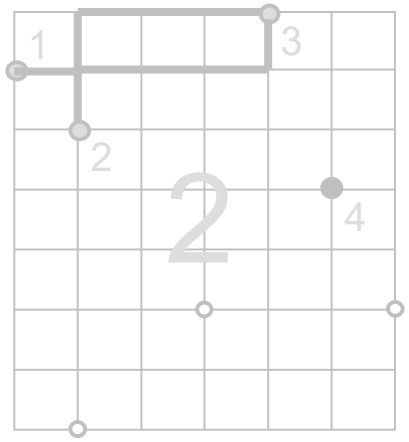


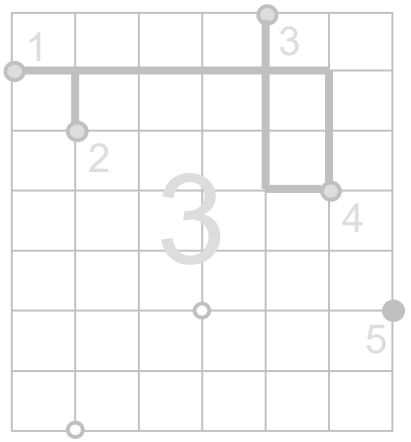


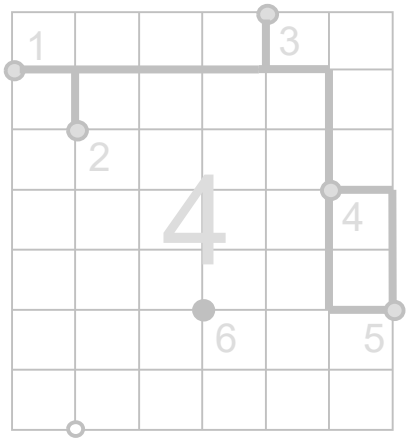


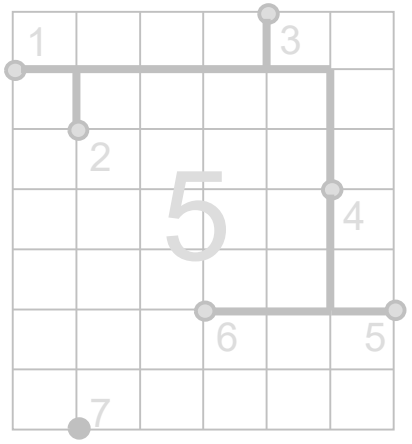
Example Sequential Steiner Tree Heuristic

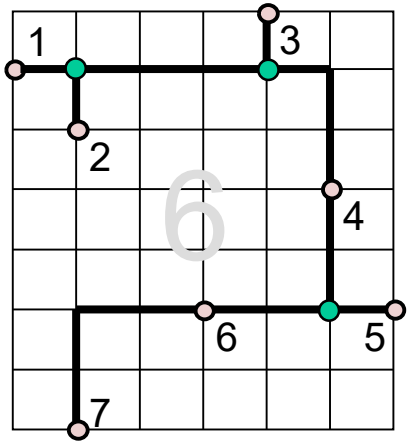








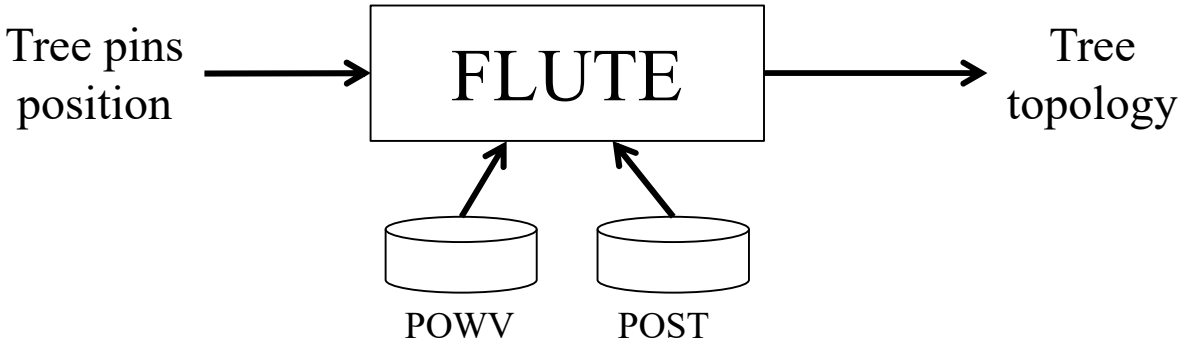




8.4.3.5 FLUTE-斯坦纳树算法

Chris Chu, FLUTE: Fast Lookup Table BasedWirelength Estimation Technique, ICCAD-2004

Chris Chu, YiuChung Wong, Fast and accurate rectilinear steiner minimal tree algorithm for VLSI design, ISPD-2005



Fast and Accurate Rectilinear Steiner Minimal Tree Algorithm for VLSI Design

Chris Chu
Iowa State University

Yiu-Chung Wong
Rio Design Automation

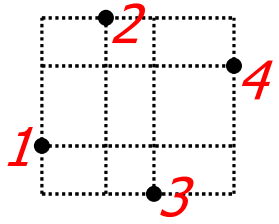
Overview

- A fast and accurate algorithm targeting VLSI applications
- Based on the FLUTE (Fast LookUp Table Estimation) idea [ICCAD-04] with three new contributions
- The new algorithm is still called FLUTE

- Handling of low degree nets is extremely well:
  - Optimal and extremely efficient for nets up to 9 pins
  - Still very accurate for nets up to degree 100
- So FLUTE is especially suitable for VLSI applications:
  - Over all 1.57 million nets in 18 IBM circuits [ISPD 98]
    - More accurate than Batched 1-Steiner heuristic
    - Almost as fast as minimum spanning tree construction

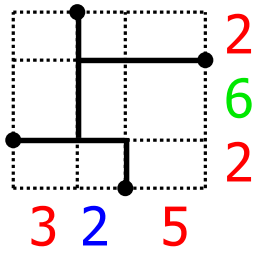
Review of FLUTE

- Lookup Table based approach
- Originally proposed for wirelength estimation
- Given a net:
  1. Find the group index of the net
  2. Get the POWVs from LUT
  3. Find the segment lengths
  4. Find WL for each POWV and return the best



Group index:
3142

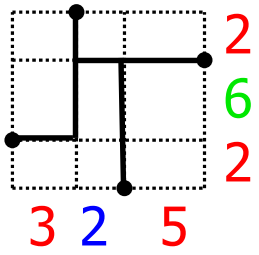
POWVs:
(1,2,1, 1,1,1)



HPWL + 2 = 22

Return

(1,1,1, 1,2,1)

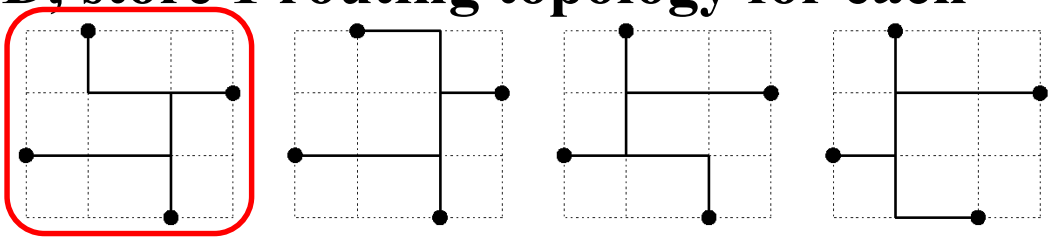


HPWL + 6 = 26

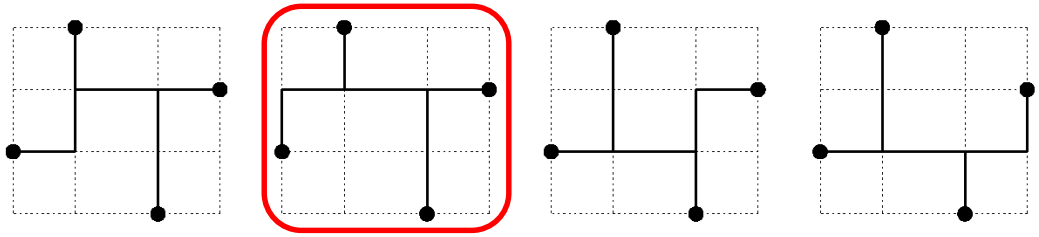
RSMT Construction

- **If degree <= D, store 1 routing topology for each POWV**

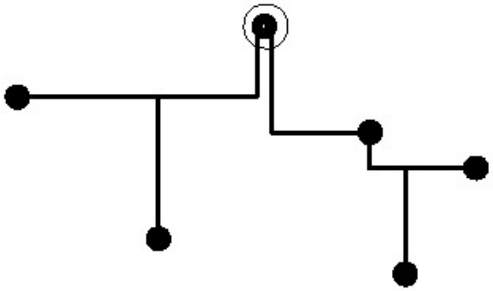
POWV
(1,2,1,1,1,1)

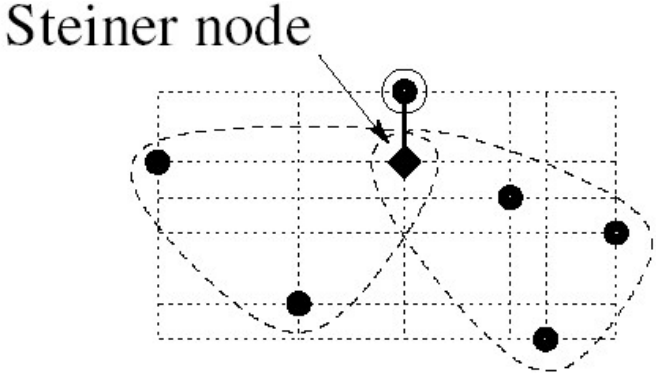


POWV
(1,1,1,1,2,1)



- **If degree > D, Steiner trees of two sub-nets are comb**
  - **Re**





Statistics on POWV Table

- **Boundary compaction technique to build LUT**
  - **Optimal up to degree 9**
  - **Table size for all nets up to degree 9 is 2.75MB**
- **MST-based algorithm to evaluate a net efficiently**
- **Impractical for high-degree nets**

Degree n	# of groups $n!$	# of POWVs in a group			Ave. # of op. per net
		Min.	Ave.	Max.	
2	2	1	1	1	0
3	6	1	1	1	0
4	24	1	1.667	2	1.333
5	120	1	2.467	3	4.267
6	720	1	4.433	8	10.333
7	5040	1	7.932	15	20.025
8	40320	1	15.251	33	38.561
9	362880	1	30.039	79	74.155

Experimental Setup

- **Comparing five techniques:**
  - **RMST – Prim's RMST algorithm**
    - **Prim [BSTJ 57]**
  - **RST-T – Refined Single Trunk Tree**
    - **Chen et al. [SLIP 02]**
  - **SPAN –Spanning graph based algorithm**
    - **Zhou [ISPD 03]**
  - **BI1S -- Batched Iterated 1-Steiner heuristic**
    - **Griffith et al. [TCAD 94]**
  - **FLUTE with D=9 and A=3**

- **18 IBM circuits in the ISPD98 benchmark suite**
- **Placement by FastPlace [ISPD 04]**
- **Optimal solutions by GeoSteiner 3.1 (Warme et al.)**

Benchmark Information

Circuit	# of nets	Ave. degree	Max. degree
ibm01	14111	3.58	42
ibm02	19584	4.15	134
ibm03	27401	3.41	55
ibm04	31970	3.31	46
ibm05	28446	4.44	17
ibm06	34826	3.68	35
ibm07	48117	3.65	25
ibm08	50513	4.06	75
ibm09	60902	3.65	39
ibm10	75196	3.96	41
ibm11	81454	3.45	24
ibm12	77240	4.11	28
ibm13	99666	3.58	24
ibm14	152772	3.58	33
ibm15	186608	3.84	36
ibm16	190048	4.10	40
ibm17	189581	4.54	36
ibm18	201920	4.06	66
All	1570355	3.92	134

Accuracy Comparison

Circuit	Wirelength error (%)				
	RMST	RST-T	SPAN	BI1S	FLUTE
ibm01	4.092	1.942	0.258	0.098	0.076
ibm02	5.849	3.750	0.335	0.117	0.221
ibm03	4.637	1.925	0.267	0.099	0.060
ibm04	4.048	1.270	0.207	0.061	0.050
ibm05	4.489	3.155	0.330	0.111	0.086
ibm06	5.964	2.846	0.378	0.137	0.090
ibm07	4.720	1.693	0.266	0.087	0.042
ibm08	4.784	4.446	0.325	0.119	0.250
ibm09	4.331	1.813	0.236	0.075	0.039
ibm10	4.104	1.787	0.253	0.077	0.052
ibm11	4.018	1.215	0.218	0.062	0.026
ibm12	3.783	1.912	0.246	0.074	0.056
ibm13	4.782	2.001	0.293	0.106	0.049
ibm14	3.908	1.541	0.220	0.069	0.036
ibm15	4.201	1.945	0.265	0.076	0.060
ibm16	4.231	2.426	0.278	0.089	0.061
ibm17	3.905	2.189	0.265	0.080	0.052
ibm18	4.432	3.352	0.298	0.098	0.133
All	4.232	2.263	0.269	0.085	0.070

Runtime Comparison

All experiments are carried out on a 750 MHz Sun Sparc-2 machine

Circuit	Runtime (s)				
	RMST	RST-T	SPAN	BI1S	FLUTE
ibm01	0.03	0.55	3.68	72.48	0.06
ibm02	0.06	0.78	7.17	108.93	0.15
ibm03	0.05	1.05	7.04	140.00	0.11
ibm04	0.06	1.22	7.29	162.15	0.09
ibm05	0.08	1.15	11.95	146.23	0.22
ibm06	0.06	1.38	9.94	176.88	0.16
ibm07	0.09	1.94	13.76	244.50	0.20
ibm08	0.14	2.07	18.58	266.02	0.41
ibm09	0.12	2.44	17.14	308.61	0.23
ibm10	0.16	3.00	26.00	383.44	0.39
ibm11	0.14	3.17	20.08	411.29	0.23
ibm12	0.18	3.07	28.61	394.68	0.44
ibm13	0.19	3.88	28.37	504.71	0.38
ibm14	0.29	6.00	44.21	775.54	0.60
ibm15	0.40	7.37	63.72	949.99	0.95
ibm16	0.43	7.57	77.27	968.36	1.03
ibm17	0.50	7.71	92.85	973.79	1.35
ibm18	0.49	8.03	79.96	1036.36	1.31
All	0.42	7.51	67.1	965.1	1

Normalized →

8.4.4 总体布线算法

❑常用的总体布线方法与策略:

- Negotiation-based GR
- 迭代改善（串行）布线策略和拆线重布方法;
- 基于整数规划的总体布线方法;
- 基于网络流的总体布线方法;
- 层次迭代总体布线方法。

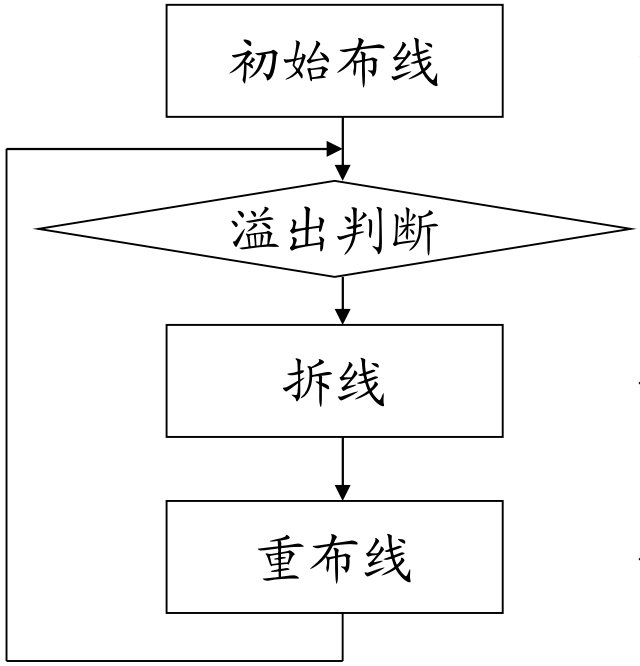
8.4.4.1 迭代改善（串行）布线和拆线重布算法

- **特点**
  - ❖简单、实用;
  - ❖适用于多种布图模式，特别是门阵列布图模式。
- **问题定义**
  - ❖ **布线需求**：通道容量固定，要求确保布通率，实现布线通道走线均匀。
  - ❖ **优化方法**：限制总的溢出量，或是调整最大溢出量。
  - ❖ **定义**：溢出量F_i = 所需资源量N_i － 固有资源S_i;
  - ❖ **总体布线的目标**：寻找一个合理的布线分配，使得

$$\text{Minimize} \sum_{\forall i \in E} F_i \quad ;$$

或者 $\text{Minimize Max}\{F_i\}$。

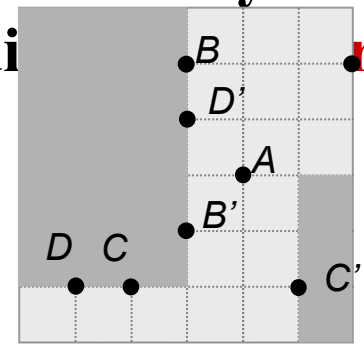
- **布线策略说明**
  - 同传统的串行布线算法在策略上有所不同，主要体现在布线策略对布线顺序的依赖程度上。
- **算法流程**



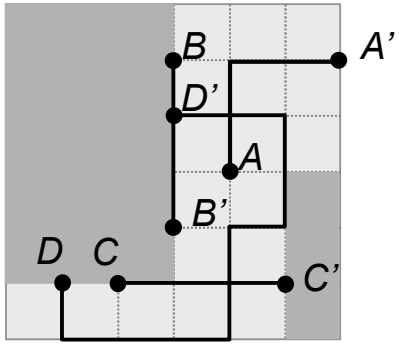
- 完成所有线网的构造；
- 拆除部分溢出区线网；
- 重新构造被拆除的线网。

Rip-Up and Reroute (RRR)

- **Rip-up and reroute (RRR) framework: focuses on hard-to-route nets**
- **Idea: allow temporary violations, so that all nets are routed, but then iteratively remove some nets (rip-up), and route them di route)**

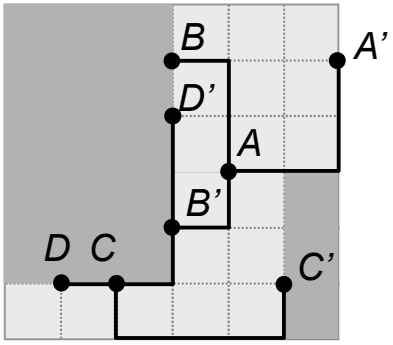


Routing without allowing violations



WL = 21

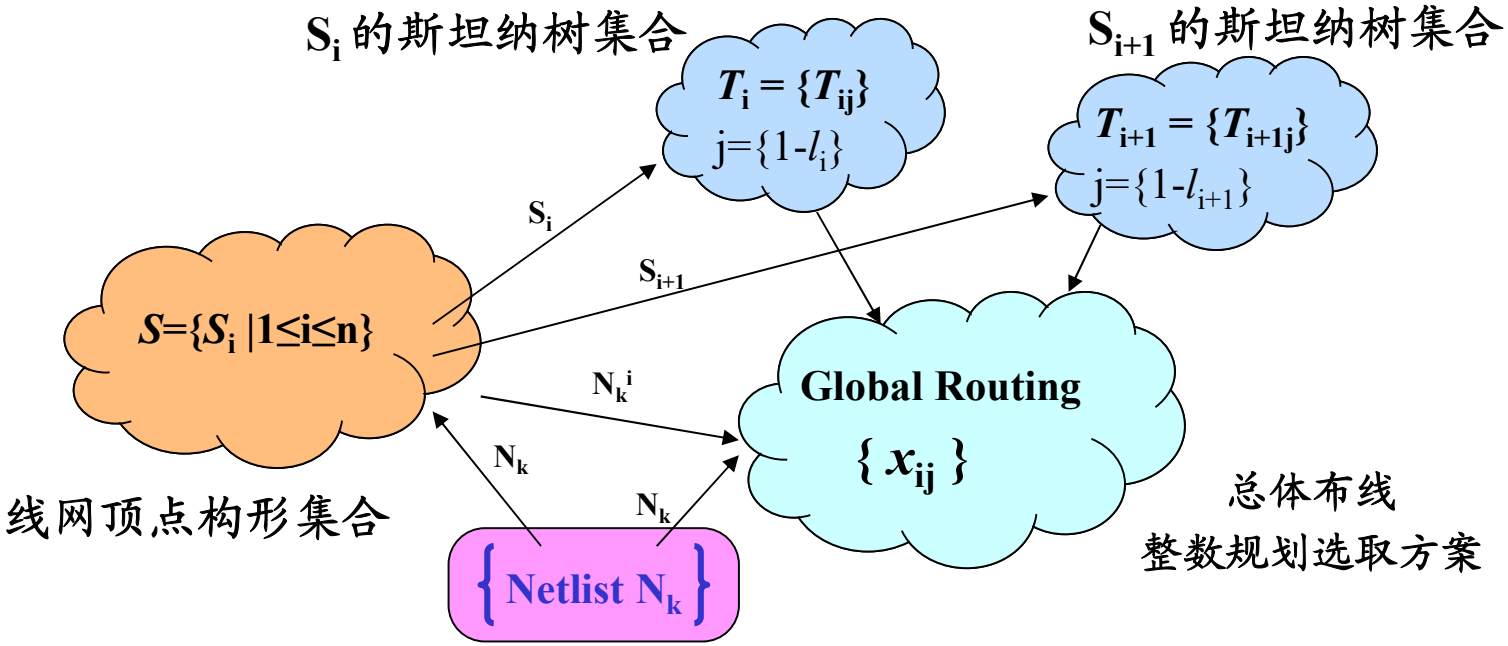
Routing with allowing violations and RRR



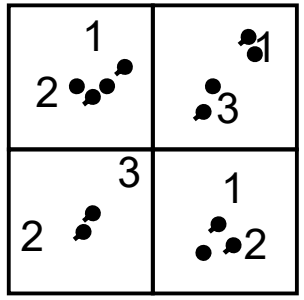
WL = 19

8.4.4.2 基于整数规划的方法

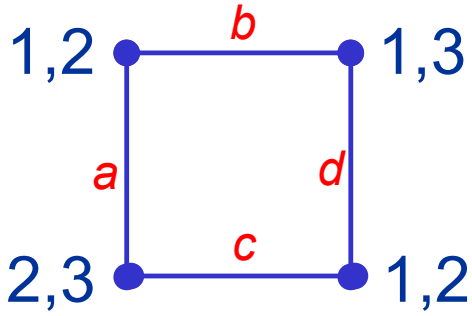
- 算法的出发点
  - 解决同时考虑所有线网的布线问题。
- 算法策略
  - 从线网布线方案选取的角度，把总体布线问题看成是每一条线网从候选斯坦纳树集合中，选取哪一种斯坦纳树构形的问题。将问题转换成整数规划问题。



Concurrent Approach: Example



Netlist:net_1, net_2,net_3



$C_a = C_b = C_c = C_d = 2$

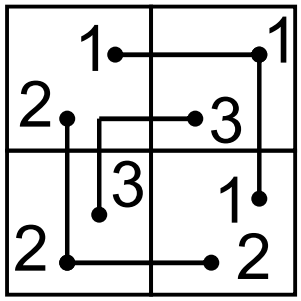
Possible trees:

net_1: 2 x_{11} | 3 x_{12} | 3 x_{13}

net_2: 2 x_{21} | 3 x_{22} | 3 x_{23}

net_3: 2 x_{31} | 2 x_{32}

Solution



$$\text{Min. } 2x_{11}+3x_{12}+3x_{13}+2x_{21}+3x_{22}+3x_{23}+2x_{31}+2x_{32}$$

$$\text{s.t.} \begin{cases} x_{11}+x_{12}+x_{13}=1; & x_{12}+x_{13}+x_{21}+x_{23}+x_{31} \le C_a \\ x_{21}+x_{22}+x_{23}=1; & x_{11}+x_{22}+x_{23}+x_{31} \le C_b \\ x_{31}+x_{32}=1; & x_{12}+x_{22}+x_{23}+x_{32} \le C_c \\ x_{ij}=0 \text{ or } 1 \quad \forall i,j; & x_{11}+x_{12}+x_{22}+x_{23}+x_{32} \le C_d \end{cases}$$

- Consider all the nets simultaneously.
- Formulate as an integer program.
- Given:

Nets	*Set of possible routing trees*
net 1	$T_{11}, T_{12}, \ldots\ldots, T_{1k_1}$
⋮	⋮
net n	$T_{n1}, T_{n2}, \ldots, T_{nk_n}$

L_{ij} = Total wire length of T_{ij}
Ce = Capacity of edge e

- Determine variable x_{ij} s.t. $x_{ij} = 1$ if T_{ij} is used
  $x_{ij} = 0$ otherwise.

基于整数规划的总体布线问题

❖输入:

- 每线网所有/部分候选斯坦纳树构形的集合;
- 布局结果;
- 布线通道容量。

❖目的:

- 在布线通道容量允许的范围内，为每条线网选择一条斯坦树构形，作为该线网的拓扑走线结构。

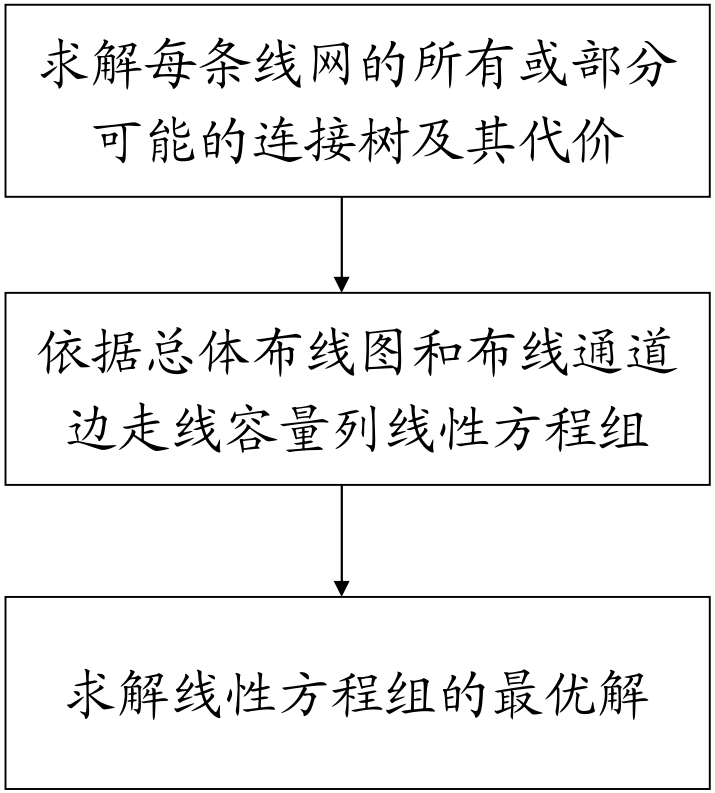
❖目标:

- 最小总线长;
- 布线通道走线均匀。

目标函数： $\min \text{sum of total WL}(net)$

约束函数： $D(g) \leq C(g)$

基于整数规划的总体布线流程



总体布线整数规划问题为：

Minimize $\sum_{i=1}^{n}\sum_{j=1}^{l_i} l(T_{ij}) * x_{ij}$

Subject to: $\sum_{j=1}^{l_i} x_{ij} = n_i, \quad i = 1,\ldots,n$ （completeness constraints）

$\sum_{(ij),e\in T_{ij}} x_{ij} + x_e = c(e), \quad e \in E$ (capacity constraints)

其中：

- $S = \{S_i \mid 1 \le i \le n\}$是在总体布线图中的线网顶点子集的集合，$n_i$表示对应于$S_i$线网类型的线网数；
- $T = \{T_{ij}\}$, $j = 1,2,\ldots,l_i$, $i = 1,2,\ldots,n$, 是相应S_i的斯坦纳树集合，$l(T_{ij})$是斯坦纳树T_{ij}的长度。
- x_e是边e的松弛变量，表示e的冗余容量；
- 整数变量x_{ij}表示用T_{ij}布线的线网数，S_i是一个线网类型，T_{ij}是对应S_i的一种布线结果。

❖ 算法分析

- ➢ 问题的规模
  - ✓ 线网类型——可能的斯坦纳树数量，无法列出线网所有的斯坦纳树。
  - ✓ 总体布线图中边比较多，带来的约束条件也多。
- ➢ 问题的难度

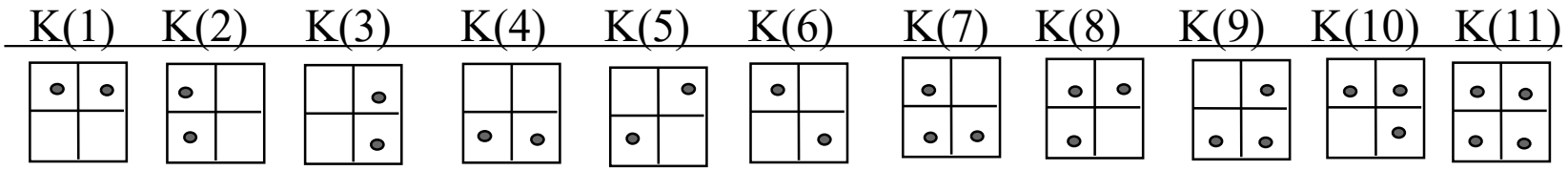
  整数规划问题——NP-hard。

❖ 解决方法

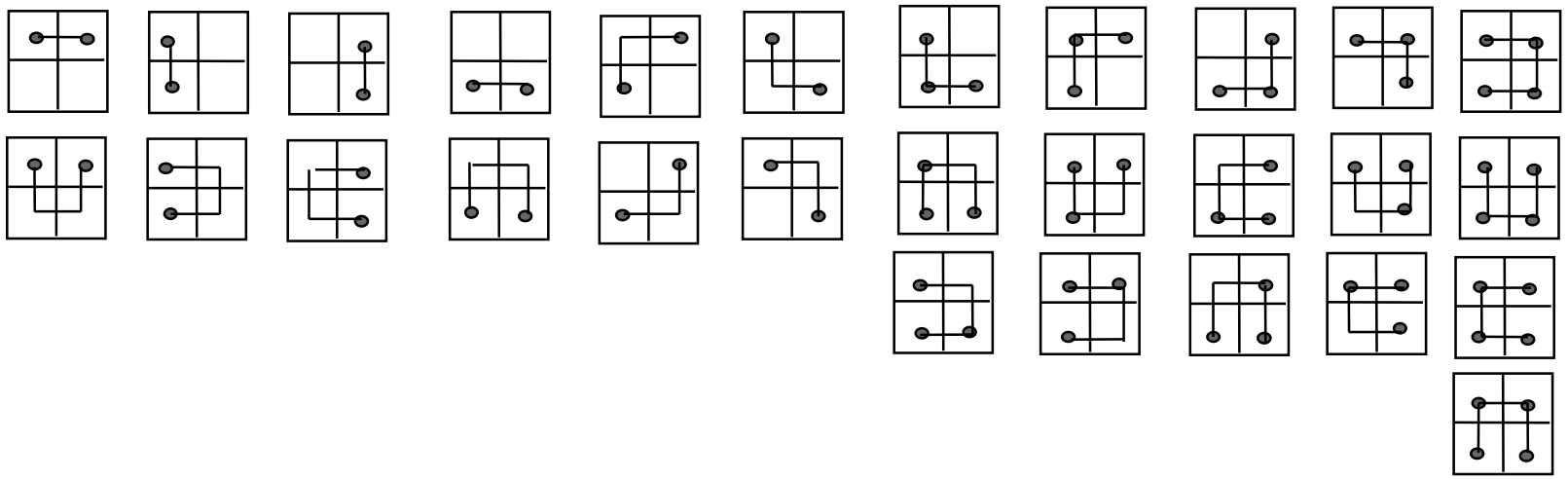
- ➢ 采用层次式的方法将总体布线整数规划问题分解成多个规模足够小并能精确求解的问题。然后，合并这些解，得到总体布线问题的近似解。

1、2×2单元上的布线问题

❖ 2×2单元上的十一种线网类型:



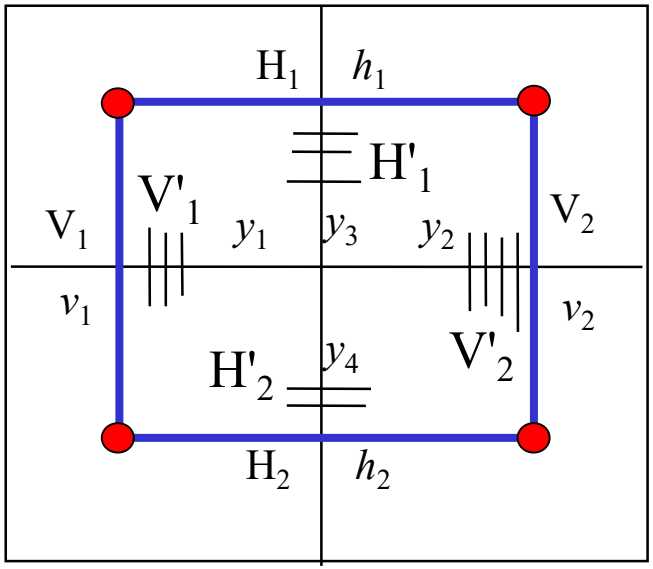
❖ 可能的布线方案



➢ 每一类型有2-4种可能的布线方案

❖ 2×2单元上布线整数规划问题

- 四个内部单元的总体布线图；
- 垂直布线通道V_1和V_2，水平布线通道H_1和H_2；
- 垂直布线通道容量v_1和v_2，水平布线通道h_1和h_2。



❖ 布线需求

- 穿过垂直布线通道线网集合V'_1和V'_2，穿过水平布线通道线网集合H'_1和H'_2；
- 垂直布线通道松弛容量y_1和y_2，水平布线通道y_3和y_4。

❖2×2单元上总体布线整数规划问题形式化描述:

Maximize $\sum_{i=1}^{4} y_i$ 寻找剩余轨道数最大解

Subject to

$\sum_{i,j\in V'_1} x(i,j) + y_1 = v_1$ 布线通道容量约束

$\sum_{i,j\in H'_1} x(i,j) + y_2 = h_1$

$\sum_{i,j\in V'_2} x(i,j) + y_3 = v_2$

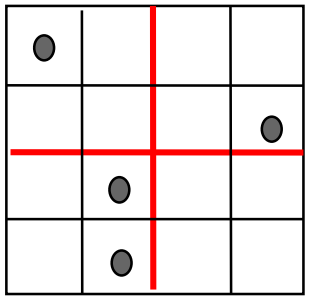
$\sum_{i,j\in H'_2} x(i,j) + y_4 = h_2$

$x(i,j) \geq 0, \quad \forall i,j$

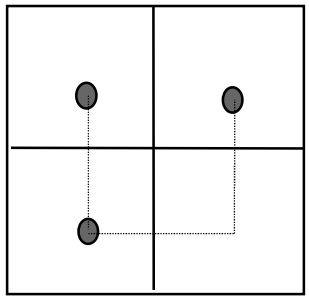
$x(i,j)$，y_1，y_2，y_3，y_4为正整数

2、自顶向下策略

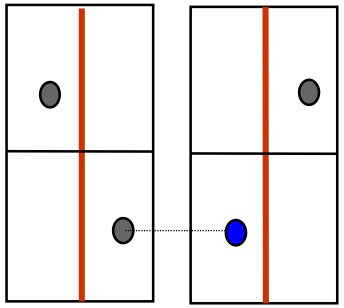
- **算法策略：**自顶向下进行二划分，每一级形成若干2×2单元格局，对它们分别进行整数规划，最后合并分解结果，形成总的布线方案。
- 新的一级划分时，原区域分解带来的边界连接处理，如图c和图f。



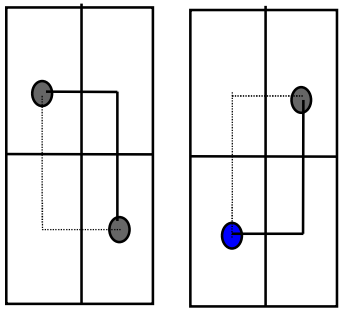
(a) 顶层划分



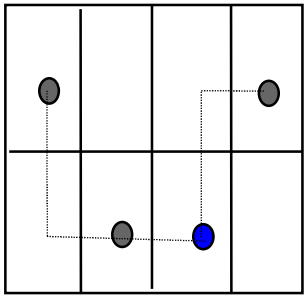
(b) K(8)



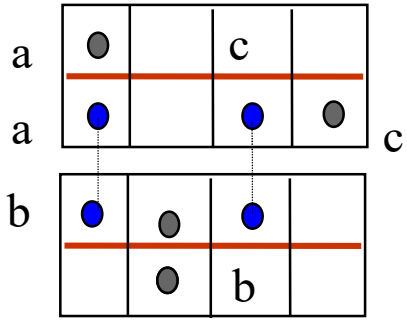
(c) 二级划分



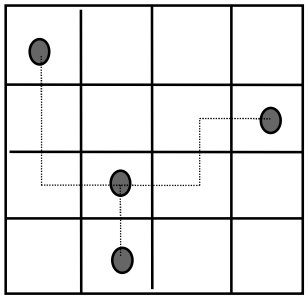
(d)K(6) K(5)



(e) 二级构形



(f) 三级划分

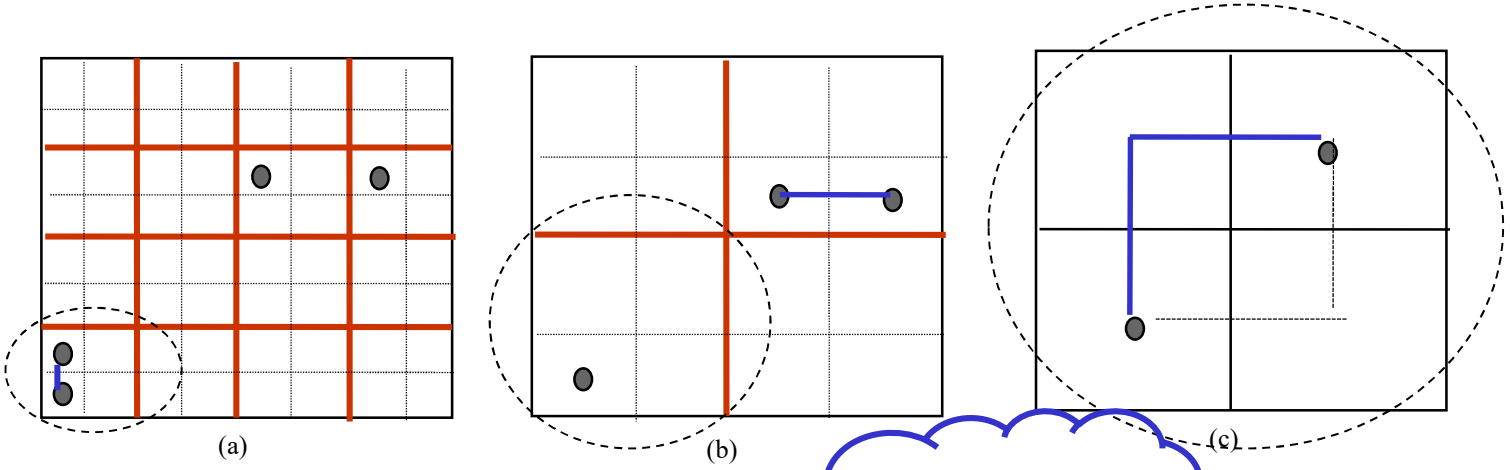


(g) 最终构形

3、自底向上策略

❖ **算法策略：**自底向上进行2×2网格的合并，在逐级合并时，对相应的2×2网格进行整数规划，确定该网格的走线格局。

❖在自底向上的合并过程中，底层网格的线网走线信息被抽象在上层的网格系中，构成新的一级线网需求信息。



8.4.4.3 基于网络流的总体布线算法

目的： 解决总体布线问题中面临的两个主要难题。

- ❖ 布线通道拥挤的不可预见性；
- ❖ 总体布线结果对线网布线顺序的依赖性。

用网络流建模的思想：

- ❖ 总体布线：把有限的布线资源合理地分配给各个线网的布线需求。
- ❖ 借鉴网络流的思想建立总体布线模型：
  - ➢ 两种特殊的节点：source和sink；
  - ➢ 流的需求：总体布线图上商品从source节点流向sink节点；
  - ➢ 边的容量：总体布线图的边允许流过的最大容量；
  - ➢ 最大流建模：不超过边容量情况下流过最大的流量方案；
  - ➢ 最小代价建模：以最小代价完成流需求。

R. C. Carden IV and C. K. Cheng, A Global Router Using an Efficient Approximate Multicommodity Multiterminal Flow Algorithm, Proc. of IEEE/ACM Design Automation Conference, 1991, pp:316-321

Multi-Commodity Flow

- ***Instance***: A directed graph G = (***V***, ***E***), **edge capacities *c(e)*** and **edge cost *l(e)*.** And **demands $d_i(v)$** for vertices $v \in V$ and for k commodities $i = 1,\ldots,k$
- ***Configurations***: all sequences of edge labelings $f_i \rightarrow R+$, $i = 1,\ldots,k$
- ***Solutions***: all sequences of edge labelings that satisfy **Capacity Constraints**, **Flow-Conservation Constraints**.
- ***Minimize***:

$$l(f) = \sum_{e \in E} (l(e) \times \sum_{i=1}^{k} fi(e))$$

8.4.4.4 Modern Global Routing

- General flow for modern global routers, where each router uses a unique set of optimizations: