课程设计报告

班级：计研 5

陈松　005636

魏洪川 005627

张轶谦 005624

一．问题简介：

图 G 的一个 drawing 定义为将 G 的每一个顶点分别对应为平面上不同的点，图 G 的每条边(u,v)分别对应为一条 Jordan 曲线，该曲线的端点是 u,v。构造一幅图的 drawing 的问题在一些领域有重要的应用价值，如信息可视化，超大规模集成电路布图。依照不同的应用问题，对 drawing 的构造也有不同要求，如要求顶点都在整数格点上、有最小面积、最小总边长等等，本文中，我们主要研究如何计算获得平面图的 orthogonal representation，并计算出具有最小面积的 orthogonal drawing 的问题，即要求 drawing 的每条边均由一些水平和垂直的线段构成。

1.采用随机算法生成双连通最大度数不超过 4 的平面图，并给出其平面表示。

2.采用网络流技术计算平面图的正交表示

3.采用网络流技术计算边长实现平面图的正交画法。

二．基本概念

关键字：drawing，planar，orthogonal，representation

1．一个图 G 的 drawing 将 G 的每一个顶点对应为平面上的另一点，每一条边(u，v)对应为一条以 u 和 v 为端点的若当曲线。如果一个 drawing 的每条边都是由水平和垂直线段组成，则称其为 orthogonal drawing。如果一个 drawing 任何两条边除了公共顶点外均不相交，则称其为 planar，一个具有 planar drawing 的图称为 planar graph。

2．一个 planar graph 如果每个顶点最多与四条边相连，则称其为 4-planar。只有 4-planar 才有正交表示。

3．一个 planar graph 的两个 planar drawing 如果在每个对应顶点处的边均有相同的顺时针排列次序，则称这两个 drawing 等价（equivalent）。orthogonal 也有类似的定义。

4．所有等价的 orthogonal drawing 构成了一个 orthogonal representation。本文研究的是将平面图的顶点均安排在整点的基础上，实现平面图的 orthogonal drawing。

三．平面图的生成

3.1 算法思想

首先随机生成三个不共线的点，构成一三角形，作为初始的 graph，然后随机的连续调用 InsertVertex 和 InsertEdge 函数，对于前者，随机的选取已经存在的一条边，在其上增加一个顶点，对于后者，随机的选取一个顶点 u，再随机选取一个与 u 有公共 face 的顶点 v，连接 u 和 v，作为新的 edge，这里需要注意随机选择 u 和 v 后要判断从其上已经引出几条边，若已经达到四条，则应重新选取，这样一直执行下去，就可以得到一个 biconnected 4-planar graph。

执行 InsertVertex 函数时比较简单，只需任意选取一条边，在其中点加入一新顶点即可；

执行 InsertEdge 函数时需注意以下几个问题：

1．一个顶点如果已经引出了四条边，则不应再连接新边；

2．共边的两个顶点不可再连接新边；

3．由于一条直线上可能存在多条边，因而经常会出现两个顶点虽然不共边但却共线的情况，这时它们之间也不可连接新边；

依照以上原则，对所有 face 中的每一顶点对按某种顺序进行遍历，寻找可以互相连接新边的顶点对，若遍历所有 face 后仍无法找到可以满足要求的顶点对，则本次插入失败，转而调用 InsertVertex 函数。

3.2 算法实现

{randomly generate a biconnected 4-planar graph}

输入：顶点数 n，边数 m

输出：具有 n 个顶点，m 条边的最大度数为 4 的平面图，当顶点数与边数无法同时满足时（一般是由于顶点数太少），增加顶点数，保证边数足够。

四．平面图的正交画法算法

4.0 正交画法基本算法思想

利用网络流的技术，实现图的正交画法。首先构造一个可以用来计算平面图的正交表示的网络。该网络中包含两种结点，一类对应图中的顶点，另一类对应图中的 face；该网络中包含两类边，一类边从对应顶点的结点到对应 face 的结点，这类边的流量表示对应该 face 的顶点角的大小；另一类边对应 face 的结点到对应 face 的结点，这一类边的流量表示两个 face 的公共部分包含的 bend 的数目。该网络的最小代价流对应具有最少 bend 的正交表示。通过解决网络最大流问题得到最小代价流问题的初始解，然后利用最小代价流算法得到最小代价流，即可求取相应的正交表示；对获得的正交表示进行矩形剖分，并再次利用网络流工具计算边长，最后计算点的坐标得出图的一种正交画法。

算法基本步骤：

输入：双连通的最大度数为 4 的平面图及其平面表示 P。

输出：平面表示 P 的正交表示 H 的画法。

Step1.Construct Network N(P).

Step2.Find a flow x in N(P), of value z(P).

Step3.Compute H from the optimal flow x.

4.1 最大网络流问题简介.

Definition. **Network**：A Network is an edge-capacitated directed graph, with two distinguished verteces called the source and the sink.

在表示网络的图中不允许存在（u, u）类型的边，也就是说从一个顶点出发的边不会回到该顶点。有向边的尾记为 Init(e)，有向边的头记为 Term(e)。

在表示网络的图中每一条边都有一个最大流量（capacity），例如边 e 的流量记为 cap(e)。

在表示网络的图中存在两个特殊的顶点。一个是 s 被称为 source，另一个为 t，被称为 sink。

Definition. A **flow** in a Network **X** is a funtion f that assigns to each e of the network a real number f(e), in such a way that

(1) For each edge e we have 0<= f(e)<=cap(e) and

(2) For each vertex v other than the source and the sink, it is true that

$$\sum_{Init(e)=v} f(e) = \sum_{Term(e)=v} f(e)$$

如果用 Q 表示流出 source 的 flow 值，那么流入 sink 的 flow 值的大小也为 Q。Q 称为整个网络的流量。

最大网络流问题：给定一个网络 **X**，计算最大可能的流量，并找出具有最大流量值的 flow（即给出每一条边的流量）。

4.2.最小代价流问题简介（Minimum cost flow）.

G=(N,A) 是一个有向图,N 为顶点集，A 为边集．与 A 中每一条边相关联的有一个 cost c_{ij} 和一个 capacity u_{ij}.与 N 中每一个顶点相关联的有一个 b(i),表示这条边的净流入量（b(i)>0）或净的流出量（b(i)＜0）.

最小代价流问题定义如下：

$$\text{Minimize } Z(x)=\sum_{(i,j)\in A} c_{ij} * x_{ij}$$

满足如下条件：

$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b(i)$对 N 中的每一个顶点成立，且对 A 中的每一条边有 $0 <= x_{ij} <= u_{ij}$.

我们假设：每一条边的下界为零．所有数据均为整数，并且所有顶点的净流入量之和等于净流出量之和．即：$\sum_{i\in N} b(i)=0$.本论文的问题满足以上假设。

最小代价流问题的初始可行解可以通过求解最大流问题获得．方法如下：

引入两个新顶点 source 顶点记为 s 和 sink 顶点记为 t．对 N 中的每一个具有净流出量的顶点 i，增加一条具有最大流量为 b(i) 的边（s, i）, 对 N 中的每一个具有净流入量的顶点 i，增加一条具有最大流量为|b(i)| 的边（i, t）.然后求解一个从 s 到 t 的最大流问题．即可得最小代价流问题的一个可行解.

4. 3. 正交表示的数据结构.

正交表示描述了图的一类正交画法的形状，并不考虑边的长度，也就是具有相同拓扑结构的一类正交画法。本文档的描述中采用如下描述形式。

图 G 的平面表示以 face 的边的列表形式给出，记为 $P(f_1)$,$P(f_2)$,…等。

对应的正交表示也以 face 的形式给出，记为 $H(f_1)$,$H(f_2)$,…。H(f)的每一个元素有(e, s, a)的格式：

（1）e 是图 G 的边；

（2）s 是二进制串；描述边的形状，第 k 位表示沿顺时针方向遍历 face 的边时在边 e 内所遇到的第 k 个 bend。‘0’和‘1’分别表示 90 度和 270 度的角度。

（3）a 是整数 90，180，270，360 之一。表示在 f 内由边 e 和下一条边（顺时针遍历）所成的角度。

正交表示有如下性质：

性质一：（参考文献[4]）

对每一个元素 r,

ROTATION(r)=ZEROS(s[r])-ONES(s[r])+(2-a[r]/90) ， ZEROS(s) 和 ONES(s)分别代表串 s 中‘1’的数目和‘0’的数目。

性质二：（参考文献[4]）

$\sum_{r\in H(f)}$ROTATION(r)=-4 ， 如 果 f 是 外 部 face ， 否 则 $\sum_{r\in H(f)}$ROTATION(r)=4。

性质三：（参考文献[6]）

$\sum_{r\in H(f)}$（ZEROS(s[r])-ONES(s[r])＋a[r]/90）＝2|H(f)|－4，如果 f 是内部 face，否则$\sum_{r\in H(f)}$（ZEROS(s[r])-ONES(s[r])＋a[r]/90）＝2|H(f)|＋4

如果 r 是 H(f)的一个元素，r 的三个数据域分别表示为 e[r], s[r], a[r]。

4. 4. 网络模型的构造.

N(P)＝（ U, A, s, t, b, c）.

有限顶点集 U;

两个特殊顶点, s 被称为 source 结点，t 被称为 sink 结点；

有向边集 A；

每一条边有一个最大流量 b(e),e∈A；

每一条有向边有一个 cost 值 c(e), e∈A,表示单位流量的代价.

G 表示最大度数为 4 的图，P={P(f):f∈F}为图 G 的一个平面表示,F 为图 G 的 face 集合．与图 G(V, E)关联的网络 N(P)＝（ U, A, s, t, b, c）的定义如下：

U= V′∪F∪{s, t}．V′表示 V 中度数小于等于 3 的顶点集，F 表示对应于 face 的顶点.

$A=A_V \cup A_F \cup A_S \cup A_T$

1．A_V包含（v, f）类型的边，其中 v∈V′，f∈F,并且 v 是 P(f)中某一条边的顶点，定义如下：

E(v, f)={e ∈P(f):边 e 和在 P(f)中的下一条边有公共顶点 v}，

边的单位代价为 1．

2．A_F包含两种类型的边（f, g）和（g, f）,对每一个由两个不同 face 组成的无序对｛f, g｝.

E(f, g)={ e ∈E:e 同时出现在 P(f)和 P(g)中．}

边的单位代价为 0，最大流量无上届。

3．A_S包含：

(a). (s, f)类型的边，其中　f 是 internal face　且｜P(f)｜<＝3；

(b). (s, v), 其中 v∈ V′.

A_S中的边的单位代价为 0，最大流量为：

b(s, f)=4 - |P(f)|, b(s, v)=4 – deg(v).

4．A_T包含(f, t)类型的边，其中|P(f)|>=5.

A_T中的边单位代价为 0 ，最大流量定义入下：

b(f, t)=|P(f)|+4 如果 f 是外部 face. 否则 b(f, t)=|P(f)|- 4。

对以上网络的定义有如下解释：

（1） 网络中的每一个单位流量代表 90 度的角度：对 A_V 中的边，流量 x(v, f)表示在 P(f)内有公共顶点 v 的两条边形成的位于 f 内的角度，计算为（x(v, f)+1）*90; A_F 中的边（f, g）表示沿两个 face 的公共部分在 f 内形成的角度为 90 度的 bend 的数目。

（2） 以上定义可保证图 G 的正交表示中每一个顶点周围的角度之和为 360 度；且每一个 face 都是垂直线段和水平线段构成的多边形。

（3） 整个网络的总代价的大小即为图 G 的正交表示中的 bend 的数目。

下面说明以上构造符合最大流问题的定义。

对对应 V′的结点：

$$\sum_f(x(v, f)- x(s, v)= \sum_f\sum_{r\in R(v,f)}(a[r]/90-1)-(4-deg(v))$$

$$=360/90-deg(v)-4+deg(v)=0.$$

其中 R(v,f)={r∈H(f):e[r]和顺时针遍历 face 的下一条边有公共顶点 v.}

对应 F 的结点，由正交表示的性质可得：

$$\sum_u x(f, u)- \sum_w x(w, f)$$

$$=\sum_g x(f, g)- \sum_g x(g, f)- \sum_v x(v, f)+ |P(f)|\pm 4$$

$$=\sum_{r\in H(f)}(ZEROS(s[r])- ONES(s[r])- a[r]/90+ 1)+ |P(f)| \pm 4=0.$$

如上所定义的 N(P)可以通过求取最大流的方式，获得一个可行解，该可行解一定满足如下条件：

每一条边（s, u）一定达到最大流量。

每一条边（u, t）一定达到最大流量。

所求的最大流量为：$Z(p)=\sum_u b(s, u)= \sum_w b(w, t)$。（参考文献[2].p169.）

因为由欧拉公式可得：

$$\sum_w b(w, t) - \sum_u b(s, u) = \sum_{f \in F}(|P(f)| - 4) + 8 + \sum_{v \in V}(\deg(v) - 4)$$

$$= 2|E| - 4|F| + 8 + 2|E| - 4|V|$$

$$= 4(|E| - |F| - |V| + 2) = 0.$$

在可行解的基础上可以进一步求解最小代价流问题 。举例说明。

4. 5. 采用的网络流算法.

4.5.1 计算最大网络流的算法之一：Ford and Fulkerson 算法

基本思想：给定一个网络和一个基于该网络的 flow。我们可以通过对网络图中的顶点进行 labelling 和 scanning 的处理找到一条从 source 到 sink 的可以增加流量的路径。不断重复以上过程，直到网络中不再存在从 source 到 sink 的可以增加流量的路径为止。

Definition. Let f be a flow function in a network **X.** We say that an edge of **X** is usable from v to w if either e is directed form v to w and the flow in e is less than the capacity of the edge, or e is directed from w to v and flow in e is >0.

给网络图中每一个顶点 v 一个三元组（u, ± , z）的标号。标号定义如下：

The ‘u’ part of the label of v is the name of the vertex that was being scanned when v was labeled.

The ‘±’ will be ‘+’ if v was labeled because the edge(u, v)was usable from u to v and it will be ‘－’ if v was labeled because the edge (v, u) was usable from v to u(i.e. if the flow from v to u was >0).

The ‘z’ component of the label represents the largest amount of flow that can be pushed from the source to the present vertex v along any augmenting path that has so far been found. At each step the algorithm will replace the current value of z by the amount of new flow that could be pushed through to z along the edge that is now being examined, if that amount is smaller than z.

算法初始时，将 source 结点标号为 (-∞,+,+∞). source 结点将具有标号状态 *labeled* 和扫描状态 *unscanned.*

Procedure scan(u:vertex;X:network;f:flow);

for every 'unlabeled' vertex v that is connected to u by an edge in either or both directions , do

if *the flow in (u,v) is less than cap(u,v)*

then

label v with (u,+,min{z(u),cap(u,v)-flow(u,v)})

else if *the flow in (v,u) is >0*

then

label v with (u, - , min{z(u), flow(v,u)}) and

change the label-status of v to 'labeled';

change the scan-status of u to 'scanned'

end.{scan}

一次完整的对网络图顶点的 labeling 和 scanning 算法如下：

*procedure labelandscan (**X**:network;f:flow;whyhalt:reason);*

give every vertex the scan-status 'unscanned';

u:=source;

label source with (-∞,+,+∞);

label-status of source:='labeled';

while *{there is a 'labeled' and 'unscanned' vertex v and sink is 'unlabeled'}*

*do scan(v, **X**, f);*

if *sink is unlabeled*

then *'whyhalt':'flow is maximum'*

else *'whyhalt':'it's time to augment'*

end.(labelandscan)

对网络图的一次 labelandscan 将获得一条可以增加网络流量的路径，可增加的流量即为 sink 顶点的 label 的 z 值。

增加整个网络流量的算法如下：

*procedure augmentflow(**X**:network;f:flow;amount:real);*

{assume that labelandscan has just been done}

```
v:=sink;
amount:=the 'z' part of the label of sink;
repeat
    (previous,sign,z):=label(v);
    if sign='+'
            then
        increase f(previous. v) by amount
            else
        decrease f(v, previous) by amount;
    v:=previous
until v=source
end.{augmentflow}
```

Ford and Fulkerson 算法就是不断重复以上过程，直到网络中不在存在可以增加网络流的路径为止。算法如下；

```
proedure fordfukerson(X:network;f:flow;maxflowvalue:real);
{find maximum flow in a given network X}
set f:=0 on every edge of X;
maxflowvalue:=0;
repeat
    labelandscan(X, f, whyhalt);
    if whyhalt='it's time to augment' then
        augmnetflow(X, f, amount);
        maxflowvalue:=maxflowvalue+amount
until whyhalt='flow is maximu'
end.{fordfulkerson}
```

最大流问题可用来计算最小代价流问题的初始可行解（feasible flow）。

4.6.正交表示的获得.

可以从 A_V 和 A_F 中边的流量分别计算 G 的正交表示中的 a-和 s-域。

对 A_V 中的每一条边，设 $r_i \in R(v, f), I=1,2,\ldots$。

$a[r_1]=(x(v, f)+1)*90$, $a[r_I]=90$ $i \neq 1$.

对 A_F 中的每一对边（f, g）和（g, f），设 $r_1, r_2, \ldots r_n$ 和 $r_n' \ldots, r_2', r_1'$ 为 f 和 g 的公共部分在 f 和 g 内的对应表示，则：

$$s[r_1]=\mathbf{0}^{x(f, g)}\mathbf{1}^{x(g,f)},\ s[r_1']=\mathbf{0}^{x(g, f)}\mathbf{1}^{x(f, g)}.$$

$s[r_i]=s[r_i']=\varepsilon$ $i \neq 1$.

4.7.边长和点坐标的计算

4.7.1.基本思想.

对获得的正交表示进行剖分,通过增加虚顶点和虚边的方法使得正交表示中只含矩形 face.通过网络流的算法分别计算具有同样走向(垂直或水平)的边长.最后计算顶点坐标.

4.7.2.剖分算法

A) 首先根据 face 的正交表示，构造串 S(f):

正交表示包括三项：

e 是图 G 的一条边；

s 是由 0,1 组成的串，表示边的形状；

a 表示该边与下一条边的夹角。

设 H(f)是一个面 f 的正交表示，H(f) = r_0，r_1，……，r_{n-1}。其中 r_i 的三项分别为 $e[r_0]$，$s[r_0]$，$a[r_0]$。

$S(f) = s[r_0]w(a[r_0])\cdots s[r_{n-1}]w(a[r_{n-1}])$;

这里

if a=90, w(a)=0;

if a=180, w(a)为空；

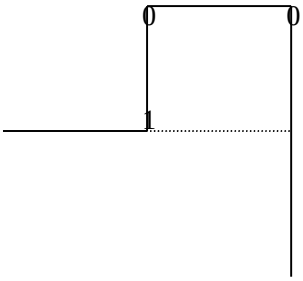
if a=270, w(a)=1;

if a=360, w(a)=11.

B) 进行剖分：

内部 face:

读串 s(f)，若出现 100，则进行剖分，如图所示：

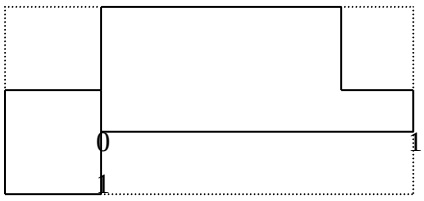


加入虚线所示的边，将一个面剖分成两个面。如果原来虚线两端没有顶点，则加入顶点，并修改数据结构中边，点，面的存储，加入的点和边要作出标记。

最后在串 s(f)中用 0 替换 100。

外部 face:

读串 s(f)，若出现 101，则进行剖分，如图所示：



加入点和虚线所示的边，从而加入一个面，并修改数据结构中边，点，面的存储，加入的点和边要作出标记。

最后在串 s(f)中用 1 替换 101。

附：参考剖分算法。

用 H 表示剖分前的正交表示，用 H′表示剖分后的正交表示，H′可以通过如下操作从 H 获得:

←增加孤立的顶点；

↑沿边增加顶点；

→增加边

详细步骤如下：算法开始在 H 的每一个 bend 处加入一个顶点。然后对每一个 face 做剖分，其中外部 face 的剖分稍有不同。假设 f 是一个内部 face, 如果 f 不是矩形，按如下操作：

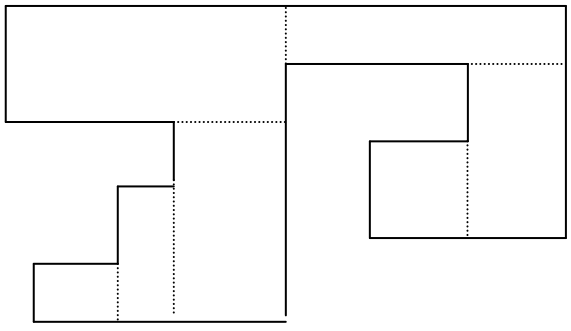
←对 f 的每一条边，用 next(e)表示逆时针遍历 f 的边界（即遍历所有属于该 face 的边）时 e 的下一条边，用 corner(e)表示边 e 和 next(e)的公共顶点，用 angel(e)表示边 e 和 next(e)在 f 所成的角度。

↑对 f 的每一条边 e，如果 angel(e)=90，则让 turn(e)=1；如果 angel(e)=180，则让 turn(e)=0；如果 angel(e)=270，则让 turn(e)=-1。

→对每条边 e，沿顺时针找到满足如下条件的第一条边 e’：在边 e(包含)和边 e’（不包含）之间所有的边的 turn 值之和为 1；记 front(e)=e’。

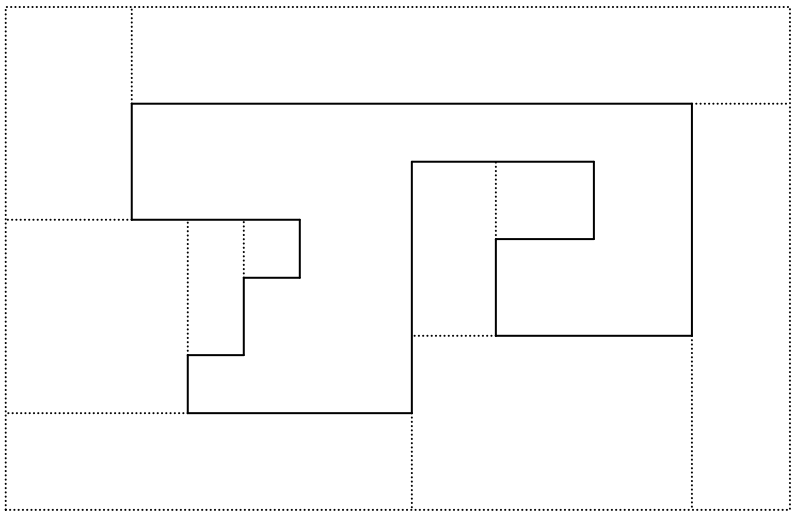
↓如果一条边 e，满足 turn(e)=-1，那么沿边 front(e)插入顶点，记为 project(e)，并增加一条没有 bend 的边 extend(e)=(corner(e)，project(e))。，并且 e 和 extend(e)所成的角度为 180 度。如果不同的两条边边 e’和 e”有 fornt(e’)=front(e”)=e”，那么当且仅当 e’，e”，和 front(e’)形成逆时针序列时，我们按逆时针顺序把 project(e”)放置在 project(e’)的前面。

下图为一个内部 face 剖分的例子：



外部 face 的剖分可以采用上述算法的一个变种。f 为外部 face 时，有 $\sum_e turn(e)=-4$，front(e)有可能不存在。为了处理这些边，我们在外部 face 的周围添加一个“矩形”，把不存在 front(e)的边延长“映射”到矩形的边上。

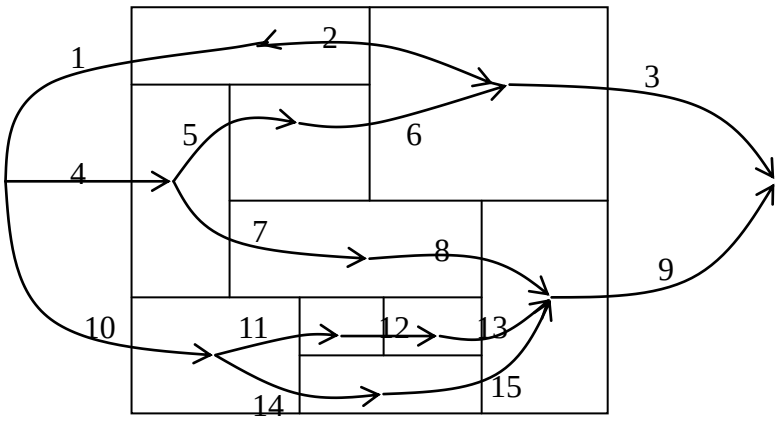
下图为一个外部 face 剖分例子



参考：文献[3]，158-161。

4.7.3.边长的计算

按照 face 的相邻关系采用递归的方法构造与 H 相关的网络 $N_{hor}(H)$，$N_{ver}(H)$；这里的网络和最大流里定义的网络相同。网络中的结点分别对应图的 face，其中 source 和 sink 结点对应图的外部 face。网络中每一条边的两个顶点分别对应于一个 face，而每条边上的流量则对应两个 face 的公共边的边长。下对 N_{ver} 举例加以说明。



如上图所示，直线段组成的是平面图的正交表示，有向曲线段为网络，构造网络时插入有向边的顺序如图所示。最左端为网络的 Source 结点，有箭头的地方对应网络的结点，最有端为网络的 Sink 结点。只要网络中不存在边的流量为 0 的边，并且对网络中每一个结点而言，流入量等入流出量，就对应一种边长的分派。本实验中采用如下分配方法。找出所有从 source 结点出发，到达 sink 结点的路径，网络的流量即为路径数。令每一条边的流量等于经过这一条边的路径数，将获得网络的一个流，从而对应一种图的边长的安排。

具体方法如下，首先对网络的每一条边编号，然后采用深度优先的算法对该网络的所有路径进行遍历，在一个矩阵中记录下所有路径所经过的边，矩阵的每一行对应一条路径，遍历所有路径后，形成了一个矩阵，访问它的所有元素，每个元素都代表一条边，一条边的编号在矩阵中出现几次，则表示经过这条边有几条路径，也即表示其流量，从而用这种方法就可以获得一种边长的分配方案。

4.7.4.点坐标的计算.

计算出图中所有边的边长后，就可以开始计算顶点坐标。

以外部 face 四个“拐角”之一作为参考点，令其坐标为（0，0），然后根据 face 的相邻关系采用深度优先的方法分别递归遍历所有具有同样走向的边（垂直或水平），递归过程和构造网络时是一致的；访问每一条边时，根据尾顶点的坐标和该边的边长，来确定头顶点的坐标；直到所有的边被遍历完毕。具体实现可参照源程序。

五．实验总结

本实验所遇到的几个难点如下：

1．由于初始的图采用的是利用随机数生成的算法，每次程序运行的过程都不一样，经常会遇到一些随机的，不可重现的错误，必须单步执行很多次才能重新遇到，从而为后面的调试带来较大困难；

2．程序的主体是将随机生成的图转为正交表示，但这一步无法直接验证，必须将程序全部编完后才能将其画出来，从而检查其正误，这样就只能是在程序全部编完后才能开始调试，与边编写边调试相比，这显然给程

序的调试带来了极大困难，前后的许多错误纠缠在一起，常常难以准确定位错误的 根源所在；

3．许多算法在论文中只是简要概述了其思想，经常是以定理的形式出现，具体过程都必须自己设计，使得编写代码花费了较长时间；

参考文献

【1】Herbert S.Wilf, Algorithms and Complexity, University of Pennsy Lvania Philadelphia, PA 19104-6395.

【2】Bavindra K.Ahuja, Thomas L.Magnanti, James B.Orlin, Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs, New Jersey 07632.

【3】G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing, Prentice-Hall, Upper Saddle River, NJ, 1999.

【4】R. Tamassia, On Embedding a graph in the grid with the minimum number of bends, SIAM J.Comput. 16(3)(1987)421-444.

【5】A. Garg, R. Tamassia, A new minimum cost flow algorithm with applications to graph drawing , in: S. North,(Ed.), Graph Drawing(Proc.GD'96), Lecture Notes in Computer Science, Vol.1547, Springer, 1998, pp. 138-152.

【6】G.Di Battista, G. Liotta, Upward planarity checking: "Faces are more than polygons", in:S.H. Whitesides(Ed.),Graph Drawing(Proc. GD'98), Lecture Notes in computer Science, vol.1547, Springer, 1998, pp.72-86.