一般图的三角剖分问题

计算几何课程大作业
计研 4　朱军　2005310449
计研 6　赵军　2004310449

前言

在计算几何课程中我们学过多边形的三角剖分，但是对于一般的图也有三角剖分的问题。研究这个问题的意义及其应用存在于概率图模型和其他的一些相关领域，比如稀疏对称正定线性系统的高斯消元法等。这个问题虽然同属于三角剖分，但是和计算几何课程所介绍的三角剖分又有很大的区别，而且使用的方法以及评价准则也不尽相同。

我们选择这个题目原因如下：

第一，　这个问题虽然和计算几何课程内容有出入，但仍然是比较相关或者比较相似的，而且这个问题本身也比较有意思和富有挑战性。

第二，　通过介绍和比较几个比较著名的一般图三角剖分算法，希望能够帮助丰富和扩展计算几何课程的内容。

在完成这个大作业的过程中，我们有幸和邓老师进行了两次深入的交流，得到的宝贵反馈意见让我们的工作得到进一步完善。尤其值得提出的是在邓老师的建议下使用随机图生成算法来测试不同的剖分算法。才是并没有意识到这个问题的复杂性，经过调研才发现生成一定约束条件的随机图本身就是一个很有趣也很有挑战性的工作。虽然，我们只实现了一种简单的非精确算法，但是这个学习的过程让我们受益匪浅。这里非常感谢邓老师！

最后，我们要声明的是这里要用到很多图论和概率论的术语和方法，由于篇幅时间有限，我们只给出了关键性的原理介绍，更详细的内容我们都给出了参考文献，供有兴趣的读者参阅。

关键词：　图、弦、簇、三角剖分

1.　问题背景

在计算几何课程中我们学过多边形的三角剖分，但是对于一般的图也有三角剖分的问题。研究这个问题的意义及其应用之一存在于概率图模型，即 probabilistic graphical model 或者称为 probabilistic networks [1]。另外，它还和稀疏对称正定线性系统的高斯消元法有关[2]。

定义 弦图/三角化图：

图$G=(V,E)$是由顶点集合V和边集合E组成的偶对。根据边是否有方向可以把图分为有向图、无向图以及链图（Chain Graph），其中有向图和链图都可以转化为无向图[1]，因此这里只讨论无向图，图中的无向边记为$e=\langle v,w\rangle$。图G是一个弦图或者三角化的图（Chordal/Triangulated Graph），如果任何一个长度大于3的环都有一条弦（chord）。

给定一个图，可能为三角化图也可能不是三角化图。完全图是三角化图，最简单的非三角化图如图 1 所示是由四个顶点构成的图。判断一个图是否是三角化图可以在线性时间内完成 [5]。

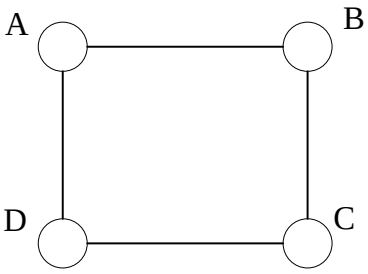


图 1。最简单的非三角化图

对于非三角化图，需要添加若干边将其转化为三角化图，这个过程叫做图的三角化/三角剖分（Triangulation）。比如对于图 1 中的简单非三角化图，添加边<A, C>或者<B, D>都可以将其三角化。三角剖分的形式化描述为：

如果图$G=(V,E)$是一个非三角化图，都可以通过适当的方式添加额外的边集F使得生成的图$G'=(V,E')$是三角化图，其中$E'=E\cup F$，这个过程定义为图的三角化/三角剖分。

事实上，对于任意一个无向图 G，如果它的顶点给定了一个序，即定义了一个一一映射 $\#:V\to\{1,2,\cdots,N\}$，其中 $N=|V|$。定义了顶点之间序关系的图称为有序图（Ordered Graph），记为$G_{\#}$。按照#定义的序关系可以得到 G 的一个三角剖分。具体做法是：

*从$\#^{-1}(N)$开始按逆序依次消除各个顶点$\#^{-1}(i)$，对于序号小于i的$\#^{-1}(i)$的邻居对如果它们不相邻，就添加一条边将它们连接起来。当所有顶点消除完毕就得到了图 G 的一个三角剖分。在下面的介绍中我们称这种方法为**消除法**。*

根据三角化图的定义，使用反证法很容易证明上述消除法得到的结果一定是个三角化图。现在要关心的问题是**不同的消除序对剖分结果的影响如何？**

由上面描述可以看出，三角剖分的结果和质量是和给定的顶点序#有关的，然而无向图的最优三角剖分问题被证明是 NP-难的[3]。为了理解这里的最优，需要定义另外一个概念，即簇（Clique）。图 G 的一个子图 C 是一个簇，如果 C 是完全图而且不存在另外一个包含 C 的完全子图。记所有簇的集合为□。不同的三角剖分方法得到的三角化图的簇集□可能不同，而且最大簇的顶点数 $\lambda=\max_{C\in\square}|C|$ 也可能不相同。而在概率图模型中，最大簇的顶点数将决定概率推理算法的复杂度。事实上，如果假设一个概率图模型中和每个顶点相关联的是一个离散随机变量，而且每个随机变量的状态空间为A，且$L=|A|$，则包含 m 个顶点的簇的联合

状态空间为迪卡尔积：$\underbrace{A \times A \times \cdots \times A}_{m}$，维数为$L^m$。那么，在这个概率图模型上的精确推理算法的复杂度将为$O(L^t)$[1]。因此好的三角剖分算法应该使最大簇顶点数较小，更确切的说应该是最大的联合状态空间的维数最小。这种优化准则也称为**最小权重准则（Minimum Weight）**，当然实际上计算权重时通常采用更复杂一点的公式，下面会具体介绍。

另外，除了最小权重准则，还有另外两个准则，即最小数目的添加边准则（Minimum Fill）和最小规模准则（Minimum Size）：

- **最小添加边准则：**即用于三角剖分所添加的边的数目最小。
- **最小规模准则：**即三角剖分后所得图的簇的顶点数目之和最小。

目前近似最优的一般图三角剖分算法有模拟退火方法[6]，以及确定的启发式算法等。

2．算法原理介绍

在这一节，我们简单介绍一下实验中使用的算法及其复杂度，这里要介绍的算法包括：使用不同优化准则的启发式算法，以及模拟退火算法。在这些算法中间要用到一些工具，如检查一个图是否已经是三角化图的最大集势搜索算法以及用于去除冗余边的算法等。

2．1．最大集势搜索算法（Maximum Cardinality Search）

最大集势搜索算法是用于检查一个给定的图是否是三角化图的算法，算法描述如下：（其中 $ne(v)=\{w|<w,v>\in E\}$）

Input: Graph $G(V,E)$
Output: whether *G* is triangulated.
Set **Output** := '*G* is triangulated'
Set counter *k* := 1
Set $L = \Phi$.
For all $v \in V$, set $c(v)=0$.
While $L \neq V$:
- Set $U := V \setminus L$
- Select any vertex v maximizing $c(v)$ over $v \in U$ and label it *i*.
- If $\prod_{v_i} := ne(v_i) \cap L$ is not complete in *G*, then set **Output** := 'G is not triangulated'.
- Otherwise, set $c(w)=c(w)+1$ for each vertex $w \in ne(v_i) \cap U$
- Set $L = L \cup \{v_i\}$.
- Increment *i* by 1.

Report **Output**.

显然这个算法可以在$O(N+|E|)$时间内实现，因为每个顶点访问 2 次而每条边需要访问 4 次。While 循环访问每个顶点一次，在判断子图是否是完全图时，每个顶点访问一次，但是根据完全图的子图也是完全图的性质，整个过程只需要对每个顶点检查一次。因此，每个顶点访问两次。每条边访问 4 次比较容易理解，因为每条边有两个顶点。另外，在每次访问时选取定点 v 只需要常数时间，这是因为由于开始时不需要排序（每个值都为 0），而每次更新 $c(v)$ 时都是局部的更新，且更新的权重相同（都是 1），因此不需要重新排序即可直接找到最大值。

2．2．随机消去算法

这种方法随机地产生一个消除序#，然后根据#定义的先后次序使用***消除法***即可得到一个三角剖分。

2．3．近似的启发式算法

前面提到一般图的最优化三角剖分是 NP-难的问题，因此人们设计了不同的启发式近似算法。根据上述三种不同的最优化准则，这里介绍三种不同的启发式算法。我们把基于上述三种最优化准则的启发式方法分别称为**最小权重启发式算法**、**最小数目添加边启发式算法**以及**最小规模启发式算法**。

虽然这三种启发式方法的启发规则不同，但是算法流程却是相同的。它们都是基于一步向前的思想，即使用**消除法**每次选择删除的那个顶点引入的代价最优。具体的来说，**最小权重启发式算法**是在每一次选择顶点 v，满足 $W(c_v)$ 最小，其中：c_v 为消除顶点 v 时产生的簇，而 $W(c_v)=\sum_{w\in c_v} W(w)$，每个顶点的权重通常取为以 2 为底的状态空间维数的对数；**最小数目添加边启发式算法**是在每一次选择顶点 v，使得新引入的边的数目最小；**最小规模启发式算法**是每次选择顶点 v，使得 $|c_v|$ 最小。可以看到，当每个节点相关联的状态空间的维数相同时，最小权重启发式算法和最小规模启发式算法实际上是一样的。

下面给出一步向前的算法描述：

Set all vertices unnumbered, set counter $i=|V|$

While there are still unnumbered vertices, do

- Select an unnumbered vertex v to optimize the criterion $\phi(v)$.
- Label it with the number i, i.e. set $v_i = v$.
- Form the set C_i consisting of v_i and its unnumbered neighbors.
- Fill in edges where none exists between all pairs of vertices in C_i.
- Eliminate v_i from the unnumbered vertices and decrement i by 1.

2．4．模拟退火算法

由上面的介绍可知，寻找最优化三角剖分实际上是寻找最优化的定点序#。上面介绍的几种启发式搜索方法都是确定性的方法，借鉴蒙特卡罗随机采样的思想，模拟退火算法给出一种很好的搜索算法。当然，模拟退火算法也有它的缺点，即搜索的效率低，而且算法的性能和收敛速度和参数的选择有关。

模拟退火算法来源于固体退火原理，将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据 *Metropolis* 准则，粒子在温度 *T* 时趋于平衡的概率为 $\exp\{-\Delta E/(kT)\}$，其中 *E* 为温度 *T* 时的内能，ΔE 为其改变量，*k* 为 Boltzmann 常数。用固体退火模拟组合优化问题，将内能 *E* 模拟为目标函数值 *f*，温度 *T* 演化成控制参数 *t*，即得到解组合优化问题的模拟退火算法：由初始解 *i* 和控制参数初值 *t* 开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减 *t* 值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。退火过程由冷却进度表(*Cooling Schedule*)控制，包括控制参数的初值 *t* 及其衰减因子 Δt、每个 *t* 值时的迭代次数 *L* 和停止条件 *S*。

模拟退火算法用到图的三角剖分问题中的具体描述如下：

- **解空间 *S***：在三角剖分问题中，前面已经介绍了一个三角剖分和一个消除序#对应，因此一个消除序#就是一个解。所以，解空间是所有可能消除序的集合 $S=\{\#:V\to\{1,2,\cdots,|V|\}\,|\,\#(w)=\#(v)\ iff\ w=v\}$。

- **目标函数 *W***：这里使用最小权重准则，则目标函数定义为 $W(\#)=W(G^{\#})$。前面定义了一个簇的权重：$W(c_v)=\sum_{w\in C_v}W(w)$，每个顶点的权重通常取为以 2 为底的状态空间维数的对数，一个**图的权重**则定义为 $W(G)=\log_2(\sum_{c\in\square(G)}2^{W(c)})$。

- **解之间的转化**：从一个解转化到另外一个不同的解实际上就是从一个序#变为另外一个序 $\#'$，这里采用的方法是在随机的选择两个下标 *i*, *j*，令 $\#'(\#^{-1}(i))=j$，$\#'(\#^{-1}(j))=i$。

- **最优解**：找到一个序 $\#^*:W(\#^*)\le W(\#),\forall\#\in S$。

模拟退火算法的性能和四个主要参数有关：初始温度、初始解、温度衰减速度以及容许的错误率。这里没有原理上的最优方案，即所谓的**最优冷却进度表**，但是可以根据不同的图例需求较好的方案。我们将在实验部分说明不同案例使用的具体方案。

2．5．去除冗余边算法

上面说过，各种启发式近似方法通常不能保证找到最优的三角剖分。这里介绍一种算法**在给定顶点消除序以及相应的三角剖分结果的情况下**可以检测和去除多余的边从而得到最优化的三角剖分，当然这种最优是在最小数目添加边以及最小权重的准则下得到的，对于最小规模准则来说去除冗余边是没有好处的，反而会增加剖分结果的规模。

这个算法的基础是如下定理及其推论：

定理 2（Rose et al., 1976）：*$G=(V,E)$ 是一个图，$G^{'}=(V,E^{'})$ 是对应的三角化图，其中 $E^{'}=E\cup F$ 。则 F 是最优化的三角剖分当且仅当 F 中的每一条边是图 $G^{'}=(V,E^{'})$ 中一个包含四个顶点的环路的唯一的弦。*

由上述定理可以证明如下推论：

推论 1（Kjærulff, 1990）：*$G=(V,E)$ 是一个图，$G^{'}=(V,E^{'})$ 是对应的三角化图，其中 $E^{'}=E\cup F$ 。则 F 是最优化的三角剖分当且仅当 F 中的每一条边 $<w,v>$，都存在另外一对不同的顶点 $\{x,y\}\subseteq ne(w,G^{'})\cap ne(v,G^{'})$ 使得 $<x,y>\notin E\cup F$，其中 $ne(w,G)$ 表示顶点 w 在图 G 中的邻居顶点集合。*

由推论 1 又可以证明如下推论：

推论 2（Kjærulff, 1990）：*$G=(V,E)$ 是一个图，$G^{'}=(V,E^{'})$ 是对应的三角化图，其中 $E^{'}=E\cup F$ 。对于 F 中的每一条边 $<w,v>$，$F\setminus\{<w,v>\}$ 仍然构成图 G 的一个三角剖分的充分必要条件是：集合 $S=ne(w,G^{'})\cap ne(v,G^{'})$ 中的顶点导出的子图 $G^{'}(S)$ 是图 $G^{'}$ 的一个完全子图。*

基于上述推论 2，可以得到如下的递归算法用于消除多余的添加边：

MINT $(F,G=(V,E\cup F),R)$，初始化 $R=F$ 。

Set $R^{'}=\{e_1\in F\mid \exists e_2\in R, s.t.\ e_1\cap e_2\neq\Phi\}$。

** Set $T^{'}=\{<w,v>\in R^{'}\mid G(ne(w,G)\cap ne(v,G))\ is\ complete\}$

If $T^{'}\neq\Phi$ then

return (MINT $(F\setminus T^{'},G=(V,E\cup F\setminus T^{'}),T^{'})$)

else

return (F)

可以看到，上述方法实际上是在不断地扫描添加的边，直到没有可以删除的多余边为止，

因此算法的复杂度为$O(c^2 \mid F \mid^2)$，其中 c 为$\max_{<w,v>\in F} \mid ne(w,G) \cap ne(v,G) \mid$。

另外可以看出这个算法和 F 中边的排列顺序即扫描顺序有关。这里要指出的一点是论文（**Kjærulff, 1990**）中给出的算法实际上是有一个错误的，即如上标所示的步骤（**）。应该改为：

For each $< w,v > \in R^{'}$,

if $G(ne(w,G) \cap ne(v,G))$ *is complete* then

Add $< w,v >$ to $T^{'}$, and update $G \leftarrow G(V, E \cup F \setminus \{< w,v >\})$.

else ;

下面是一个例子说明该算法对 F 中边的排列顺序的依赖性。如图 2 示，假设用于三角剖分的消除序列为 A, B, C, D，则所添加的边依次为<B, C>和<C, D>，显然这两条边都是多余边。使用上述算法，如果按照边的生成顺序依次扫描，算法在第一次扫描时只能删除边<C, D>，需要第二次扫描才能删除多余边<B, C>。相反，如果按照逆序扫描，则只需要一边扫描即可删除两条多余边。

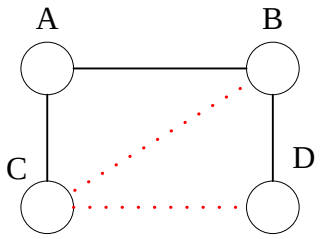


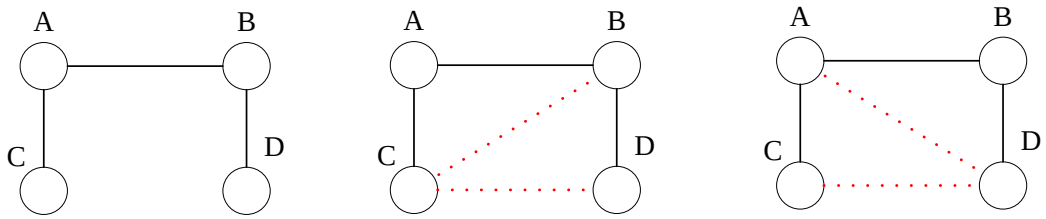
图 2。一个非最优的三角剖分结果，顶点的删除顺序为 A, B, C, D。

3．算法性能比较

在这一节，我们通过设计一些图例来说明不同近似算法的差别和性能。我们将分两部分来比较性能：第一部分主要是测试算法原理上的差别，因此设计的都是一些典型的小图；第二部分通过设计一个较大的图来全面比较不同算法性能。

3．1．随机消除算法

简单的随机算法由于每次产生的序有可能不同，因此会得到不同的剖分结果。图 3 所示是图 2 按照几种不同序关系进行三角剖分的结果。



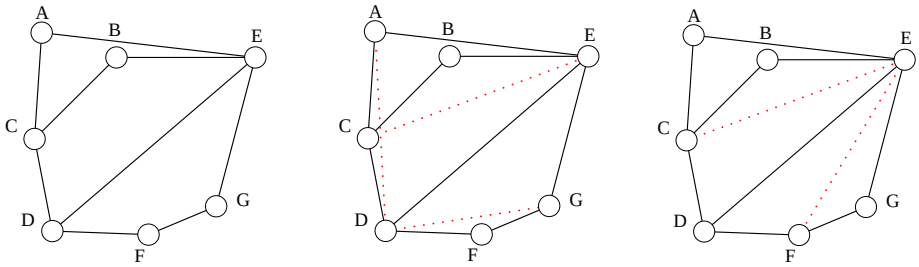
（a）　　（b）　　（c）

图 3。不同消除序关系产生的不同三角剖分结果。

当消除序为<C, A, B, D>或者<C, D, A, B>时，剖分结果如图 3（a）所示，即不需要添加任何边；当消除序为<A, B, C, D>或者<A, B, D, C>时，剖分结果如图 3（b）所示，需要依次添加边<B, C>和<C, D>；当消除序为<B, A, C, D>或者<B, A, D, C>时，剖分结果如图 3（c）所示，依次添加边<A, D>和<C, D>。

3．2．不同近似算法的比较

上面介绍的不同近似算法，在通常情况下即使在输入相同的时候，得到的结果也是不同的。当然，近似算法的差别还和不同顶点相关联的状态空间的维数有关，在这里我们假设各个顶点的状态空间的维数相同，不同的情况在下一节讨论。



（a）　　（b）　　（c）

图 4。a 为测试例图，b 为最小数目添加边启发式算法的结果，其中<A, D>是多余的添加边，c 为最小规模启发式算法以及最小权重启发式算法的结果。

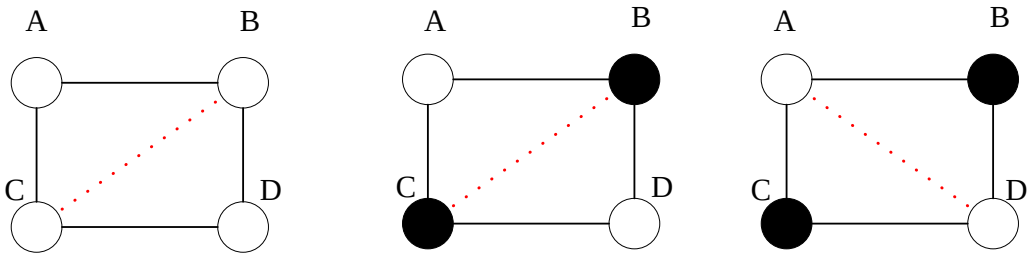
从图 4 可以看出，在通常情况下不同准则的启发式算法得到的结果是不同的，而且启发式算法得到的结果也不一定是最优结果，比如图（b）中的<A, D>就是多余的添加边，这时候使用去除冗余边算法可以删除<A, D>。

3．3．状态空间的维数对结果的影响

上面介绍了最小数目添加边准则、最小规模准则以及最小权重准则，在某些情况下不同方法得到的结果可能相同，下面以图 1 中的简单例子说明状态空间的维数产生的不同结果。

如果每个顶点相关联的状态的空间维数相同，三种不同的启发式算法得到的结果是相同的，如图 5（a）所示，其中红色的虚线表示进行三角剖分时添加的边，不同颜色的顶点用

来区别状态空间的维数。这里要说明的是因为算法中每次选择最优点的时候需要比较准则的大小，在出现多个顶点的数值相同时，仍然会出现和扫描的顺序相关，所以这里假设不同算法的**输入相同**。



（a）　　（b）　　（c）

图 5。使用不同启发式算法的得到的三角剖分结果。（a）：当各顶点的状态空间维数相同时，在输入相同的情况下三种近似算法所得结果相同；（b）：在 B、C 顶点的状态空间维数较大（比如 4）且 A、D 顶点的状态空间维数较小（比如 2）的情况下，最小数目添加边启发式算法和最小规模启发式算法的剖分结果相同，即添加边<B, C>；（c）：在 B、C 顶点的状态空间维数较大（比如 4）且 A、D 顶点的状态空间维数较小（比如 2）的情况下，最小权重启发式算法得到的不同于另外两个启发式算法。

上述是确定的近似算法，然而使用随机的模拟退火算法时，在各个顶点的状态空间维数相同的情况下，添加边<A, D>或者<B, C>都是有可能的；但是在（b）中当状态空间的维数不同而且最优解（最小权重准则）确定的时候，模拟退火算法总能搜索到结果（c）。

3．4．基于大图的性能比较

前面的实验部分主要是用来说明算法的正确性以及相互之间的差别的，因此使用的图例都是很典型的，而且图的大小也是很小。

在这一节，我们设计若干比较大的图来进一步测试不同算法的性能。这里要比较的指标包括如下几种：

- **添加边的条数：**进行三角剖分所添加的额外边的数目。
- **三角剖分图中簇的数目：**三角剖分生成的图中簇的数目。
- **最大簇顶点数目：**生成图中所有簇所含顶点的数目的最大值。
- **最大簇权重：**生成图中所有簇的权重最大值。前面介绍了簇的权重计算公式：$W(c)=\sum_{w\in c}W(w)$，其中每个顶点的权重通常取为以 2 为底的状态空间维数的对数。
- **所有簇的顶点数目之和：**生成图中所有簇的顶点数目之和。
- **所有簇的权重之和：**生成图中所有簇的权重之和。有前面的介绍可知，一个图的权重实际上是所有簇的权重之和的以 2 为底的对数。因此，由权重之和可以直接得到图的权重。
- **算法执行时间：**这里通过每种算法执行 80 次取平均时间，以秒为单位。

注意：上述评价指标对于随机算法以及随机模拟算法来说，都是指平均值。

测试图例如下，是一个含有 218 个顶点的图，其中边的数目为 331 条：这个测试图例的一个好处是，它包含了一下各种情况：

- **悬挂的顶点：**如 165。
- **多个连通分量：**整个图有 3 个连通分量（以红色线框起来的两个和余下顶点组成的一个）组成。这其中包含了含有孤立顶点的特殊情况。
- **多条交叉边：**最大的连同分量包含多条跨度很大的边，比如边<31, 47>、<41, 60>等，这些边的引入使得图中可能出现若干较大的环路，因此可能需要添加足够多的边才能得到三角剖分的图。后面的结果也验证了这一点。
- **三角剖分概念上的差异：**可以看出处于中间偏下的连通分量按照课程讲授的多边形的三角剖分的定义已经是“三角剖分”图，然而在这里它并不是一个三角剖分的图，因为显然存在长度大于等于 4 的环路不包含弦。

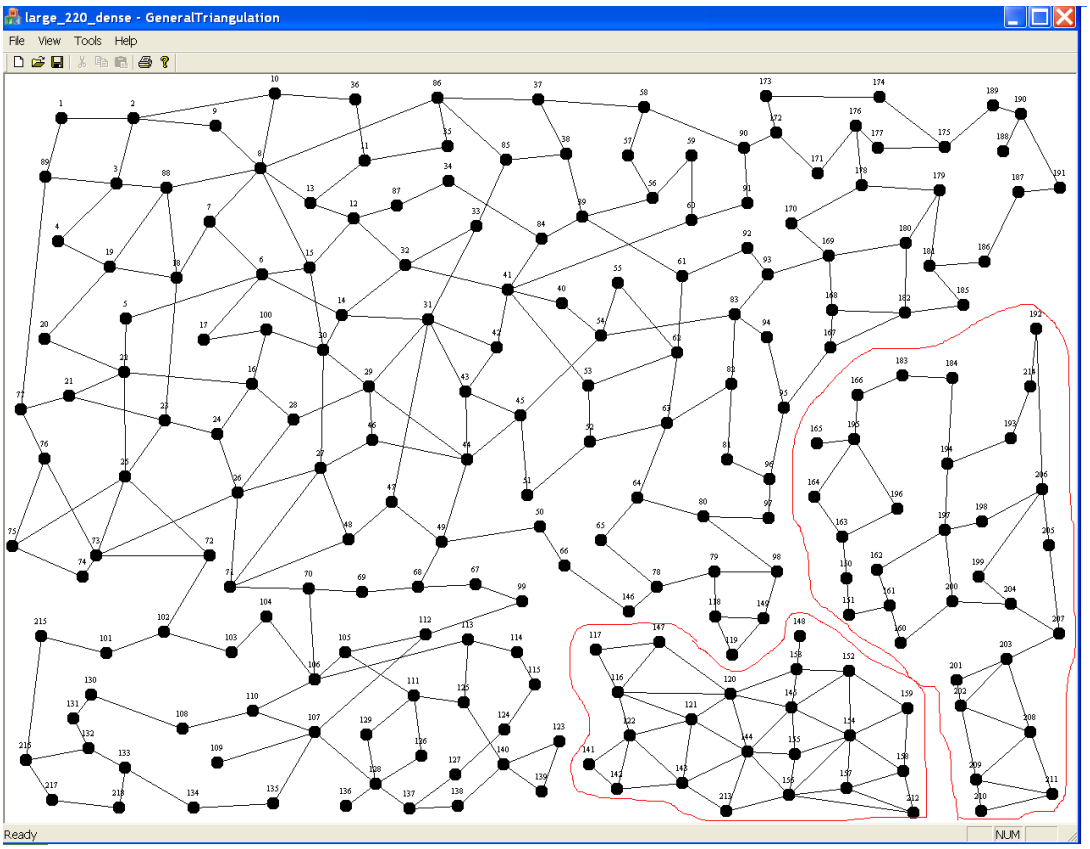


图 6。含有 218 个顶点，331 条边的测试图例。

下表是使用不同近似算法得到的结果：剖分图分别参见图 7、图 8、图 9 以及图 10，其中个顶点的状态空间维数设置为 2。

表 1。对图 6 所示样例使用不同算法得到的性能列表。其中模拟退火使用的参数为：初始温度 20，温度衰减指数 0.9，每个温度下迭代次数 1000，误差 0.2。

	添加边	簇数目	最大簇顶点数	最大簇权重	簇顶点数目和	簇权重和	平均时间
随机算法	2305	148	27	28.85	1918	31.00	0.04028
最小添加边	317	195	10	10	766	13.03	0.0403
最小规模	318	195	11	11	766	13.16	0.0549
最小权重	318	195	11	11	766	13.16	0.0535
模拟退火算法（随机）	1759	148	23	23	1551	27.65	2150.42
模拟退火算法（最小权重）	318	195	11	11	766	13.16	1794.87

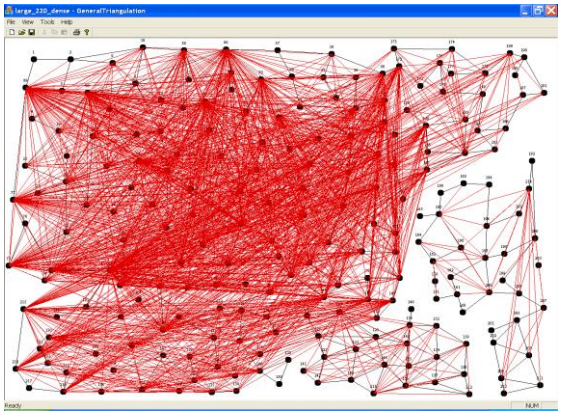


图 7（a）。随机算法得到的一个剖分结果

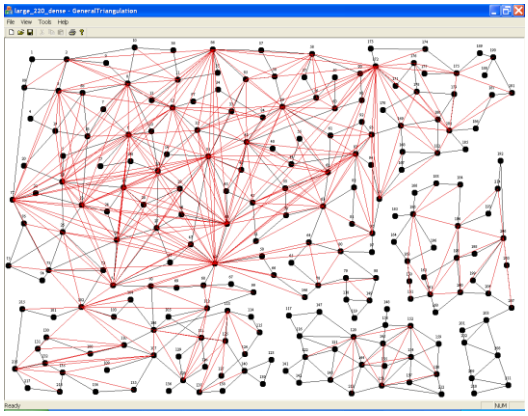


图 7（b）。最小添加边算法得到的剖分结果。

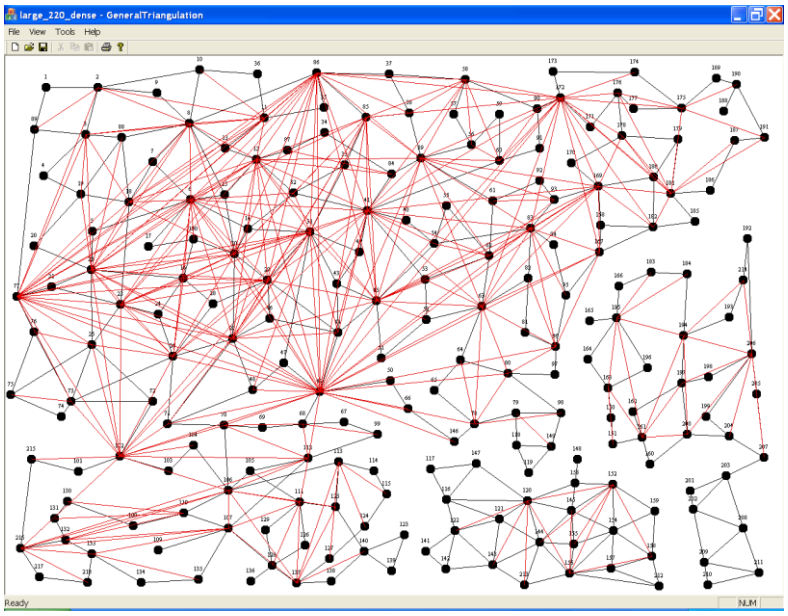


图 7（c）。在各个顶点相关联的状态空间维数相同的情况下，最小规模启发式算法和最小权重启发式算法得到的相同的剖分结果。

结论：由上述结果可以看出，在状态空间维数相同的情况下最小添加边算法在各个性能指标上表现的要好一些，另外两种启发式算法性能相同，稍差于最小添加边启发式算法。显然，随机消除算法的性能很差。从模拟退火的结果可以看出来，在这种定点个数特别多而且边数也很多的图中，随机的搜索最优解也是很困难的，从结果中可以看出来即使在 1794.87

秒内（相当于最小权重启发算法的约 35000 倍）仍然不能找到较好的解，另外从表 1 最后一行可以看出来，最小权重启发式算法得到的结果已经相对比较好了，因为在之后的尝试过程中没有发现更好的解。

下面一组结果是通过改变不同顶点的状态空间的维数来比较不同算法的性能。

从上面的结果可以看出来，顶点 6、31、41、49、77、86、102、172 等为在三角剖分结果中度比较高的顶点。**为了得到显著的效果，我们选择这些顶点来调整状态空间的分布**，因为它们会对结果产生较大的影响。具体调整值如下（其他顶点的状态空间维数不变仍为 2）：6（6）、31（8）、41（8）、49（4）、77（6）、86（6）、102（8）、172（8）。

表 2。对图 6 所示样例使用不同算法得到的性能列表。其中模拟退火使用的参数为：初始温度 20，温度衰减指数 0.9，每个温度下迭代次数 1000，误差 0.2。

	添加边	簇数	最大簇顶点数	最大簇权重	簇顶点数和	簇权重和	平均时间
随机算法	1808	**148**	24	28.87	1569	1811.45	0.04028
最小添加边启发式算法	317	195	**10**	17.75	**766**	19.15	0.0591
最小规模启发式算法	318	195	11	18.75	**766**	19.85	0.0833
最小权重启发式算法	368	193	12	**14**	797	**16.25**	0.0878
模拟退火算法（随机）	368	193	12	14	797	16.249	1999.30
模拟退火算法（最小权重）	368	193	12	**14**	797	**16.25**	1594.04

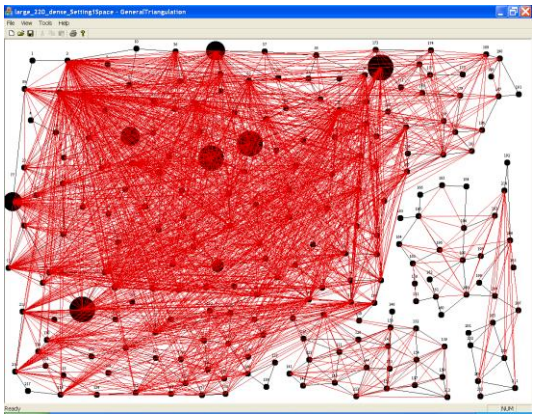
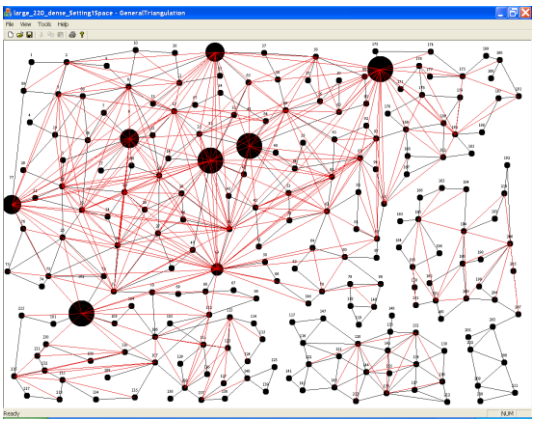


图 8（a）。随机消除算法的剖分结果



8（b）最小数目添加边算法剖分结果

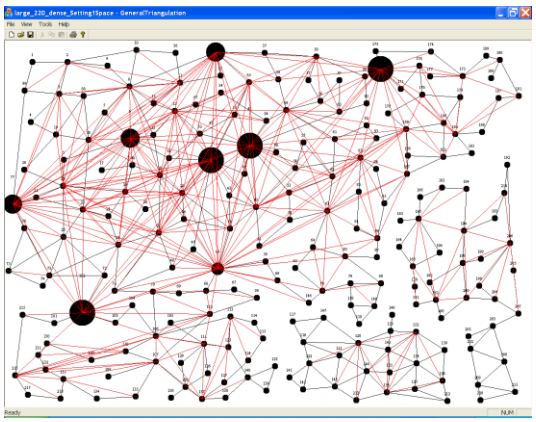


图 8（c）最小规模启发式算法剖分结果

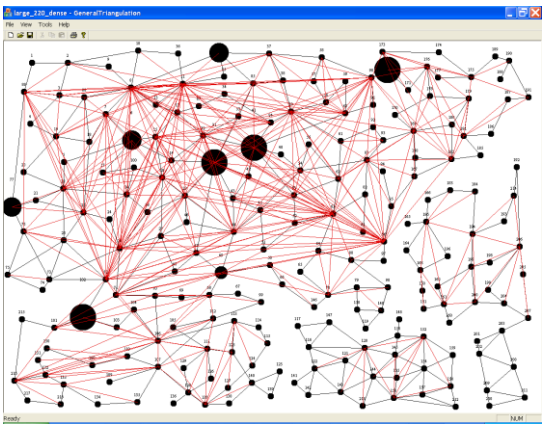


图 8（d）最小权重启发式算法的剖分结果。

结论：从上面的结果可以看出来，在最小权重准则下，最小权重启发式算法剖分结果要优于其他两种剖分算法。这是因为，最小权重准则每次选择下一个消除顶点时倾向于优先考虑状态空间比较小的顶点，因此状态空间比较大的顶点在剖分图中反而变成度比较低的顶点。对于模拟退火算法，从表 2 可以得出和表 1 同样的结论。

3．5．随机图模拟测试

上面对一个特定的大图我们给出了性能评价，下面通过随机生成具有特殊特性的图来测试各个算法平均性能。

关于随机图的生成，本身是一个很有趣也很有挑战性的问题，这方面的研究有很多。根据所要求不同的图特性，生成算法也不同。这里我们**只考虑具有一定顶点数和平均度的随机图。**即使是这一种特殊情况，已经发表的研究论文也是很丰富的。这里，我们只要求生成的是一个简单图而不限制生成的图是一个连通图。

具体方法是：对于设定的顶点数 N 和平均度 $\bar{d}$，我们通过随机采样选取 N 的度 $d(1),d(2),\cdots,(N)$。然后根据论文[9]中提供的方法生成简单的随机图。该算法主要包含两步：

1. 将每个顶点 v 拷贝 $d(v)$ 次形成集合 L，注：这里允许有重复元。
2. 随机地从集合 L 中选择一对顶点，在这对顶点之间添加一条边，然后将他们从 L 中删除。重复此步直至 L 为空。注：这里删除的只是一个拷贝。

这里要注意的是，论文[9]中的方法并不能保证生成简单的图，因此可能形成自环或者一对顶点之间形成多条边。虽然没有理论上的保证，这里我们采用一种简单的处理方法，即在上面的步骤 2 如果出现重复的边或者自环，将这次尝试放弃。

为了生成度序列，我们需要假设度的分布，最简单的方法是假设**均匀分布**。当然，现实世界里的网络顶点度分布可能是满足幂率分布（Power Law Distribution）。我们做这样的假设只是一种方案。

下表 3 给出了不同算法在不同规模图例下的性能测试结果：这两组分别为顶点数为 20，平均顶点度为 6，以及生成随机图数目为 20；顶点数为 60，平均顶点度为 10，生成随机图数目为 40。从表中可以得出如下结论：

1. 最小添加边算法的性能一般较好。对于第一组有两项指标最好，三项为次最好；对于第二组有四项指标为最好。其中很容易理解为何随机算法得到最小数目的簇？这是因为一般情况下，添加的额外边越多，得到的簇规模越大（顶点数大），因此，随机算法总是得到最少数目的簇，次最少的则是使用随机初始解的模拟退火算法；相对应的它们得到的最大簇的顶点数也是最多和次最多。
2. 对于较大的图，模拟退火算法在随机初始解的情况下很难获得较好的解。而在较好的初始解基础上可能得到进一步的改进。比如在两组测试中，以最小权重算法的结果为初始解，模拟退火可以得到更好的解。
3. 较之启发式的近似算法，模拟退火算法非常耗时。不过对于一个特定的图只需要进行一次剖分处理。因此，在特定的场合还是可以花费这个代价的。但是对于需要实时进行剖分处理的问题来说，模拟退火显现效率低下。这种情况下，近似算法可以选择使用。

表 3。使用生成随机图的测试方法得到的各个算法的测试结果。其中模拟退火算法根据初始解是随机的和以最小权重的结果作为初始解分为两种。以红色标记的表示在当前评价指标下的最好结果，蓝色标记的则为次最好结果。其中模拟退火使用的参数为：初始温度 20，温度衰减指数 0.9，每个温度下迭代次数 1000，误差 0.2。

设置	算法	边数	添加边	簇的数目	最大簇顶点数	最大簇权重	簇顶点数目和	簇权重和	平均时间
20 6 20	随机算法	32.7	39.05	**11.40**	7.40	16.5	57.75	18.04	**2.0e-3**
	最小添加边	32.7	**17.75**	14.45	**4.75**	**10.59**	**57.70**	13.14	**2.4e-3**
	最小规模	32.7	17.90	14.50	**4.70**	**10.45**	58.00	**13.10**	8.4e-3
	最小权重	32.7	17.90	14.50	**4.70**	**10.45**	58.00	**13.10**	5.0e-3
	模拟退火算法（随机）	32.7	23.50	**13.35**	5.05	11.24	58.20	13.95	1747.95
	模拟退火算法（最小权）	32.7	**17.80**	14.45	**4.70**	**10.45**	**57.70**	**13.12**	1594.24
60 10 40	随机算法	107.8	276.8	**36.15**	12.30	29.55	271.00	31.45	**1.7e-3**
	最小添加边	107.8	**85.08**	48.48	**6.60**	**15.85**	**214.80**	**18.21**	6.1e-3
	最小规模	107.8	**85.35**	48.70	**6.60**	15.86	216.08	18.26	6.0e-3
	最小权重	107.8	85.35	48.70	**6.60**	15.86	216.08	18.26	**5.8e-3**
	模拟退火算法（随机）	107.8	225.70	**37.60**	**10.48**	25.13	256.88	27.15	2190.84
	模拟退火算法（最小权重）	107.8	85.65	48.60	**6.60**	**15.86**	**215.90**	**18.23**	1778.88

3．6．其他测试样例

除了上面的测试之外，还设计了若干特殊的测试样例，比如图 9 所示是一个简单的由八边形和一个位于中心的顶点构成的图，其中位于中心的顶点和其他各顶点相联接。可以看出图中的剖分结果已经是最优的。

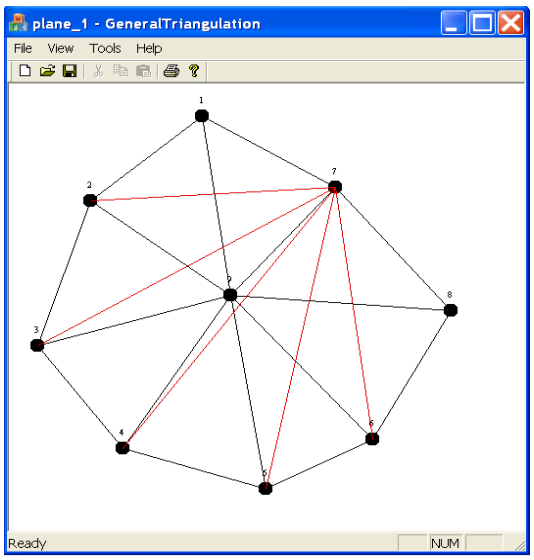


图 9。

图 10 所示是由 6 个子图相互连接而成，每个子图是由正六边形和一个位于中心的顶点组成，其中中心顶点和六边形的各顶点相连接。可以检验使用最小规模剖分算法对每个子图是最优的。

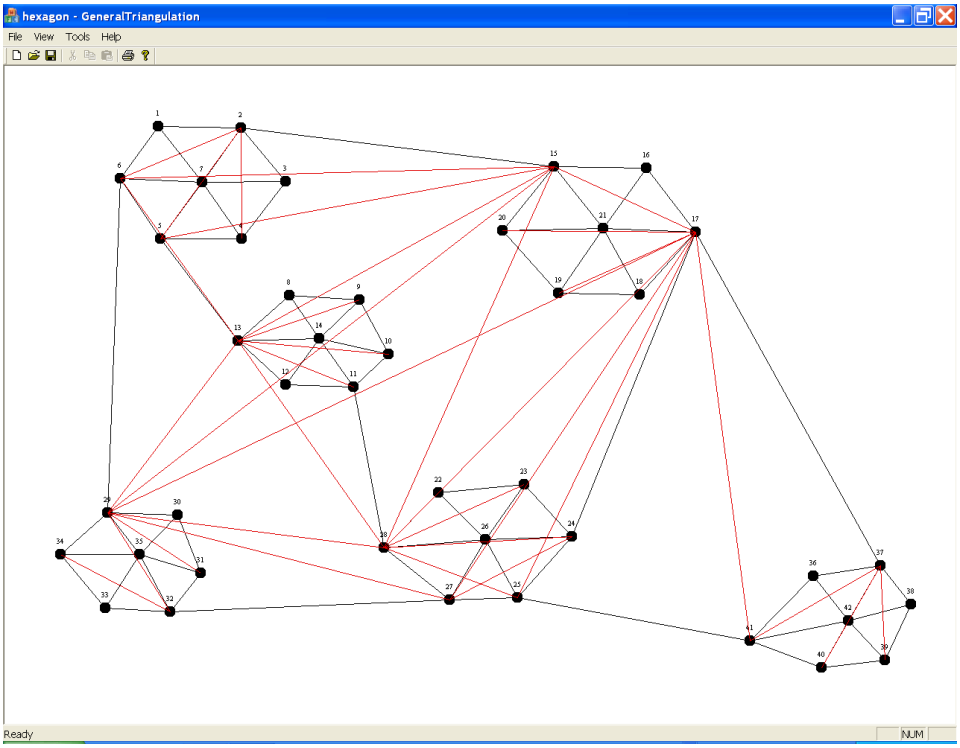


图 10。使用最小规模启发算法的剖分结果。

4. 总结

在这个报告中，我们通过对一般图的三角剖分问题的调研和对具体算法的实践和分析，进一步了解了三角剖分在计算几何课程之外的一个重要扩展和应用。在这个过程中，我们不但扩展了知识面，同时也增强了检索文献、阅读文献以及鉴别算法的准确性的能力。这些对我们将来开展很好的研究工作都是很有帮助的。在实验过程中，我们有幸和邓老师进行了两次深入的交流并且得到了宝贵反馈意见。尤其是使用随机图生成算法来测试不同的剖分算法的性能。开始的时候并没有意识到这个问题的复杂性，经过调研才发现生成一定约束条件的随机图本身就是一个很有趣也很有挑战性的工作。虽然，我们只实现了一种简单的非精确算法，但是这个学习的过程让我们受益匪浅。这里再次感谢邓老师！

当然如果这个报告能够给计算几何课程带来些许的丰富和补充，我们将感到无比的自豪。

5. 程序说明

我们在这个项目中开发了一套演示系统用于测试上述各种不同近似算法的性能。这套系统是在 Visual Studio .Net 2003 下开发的 MFC 应用程序，包括各个算法的实现。主界面如下：

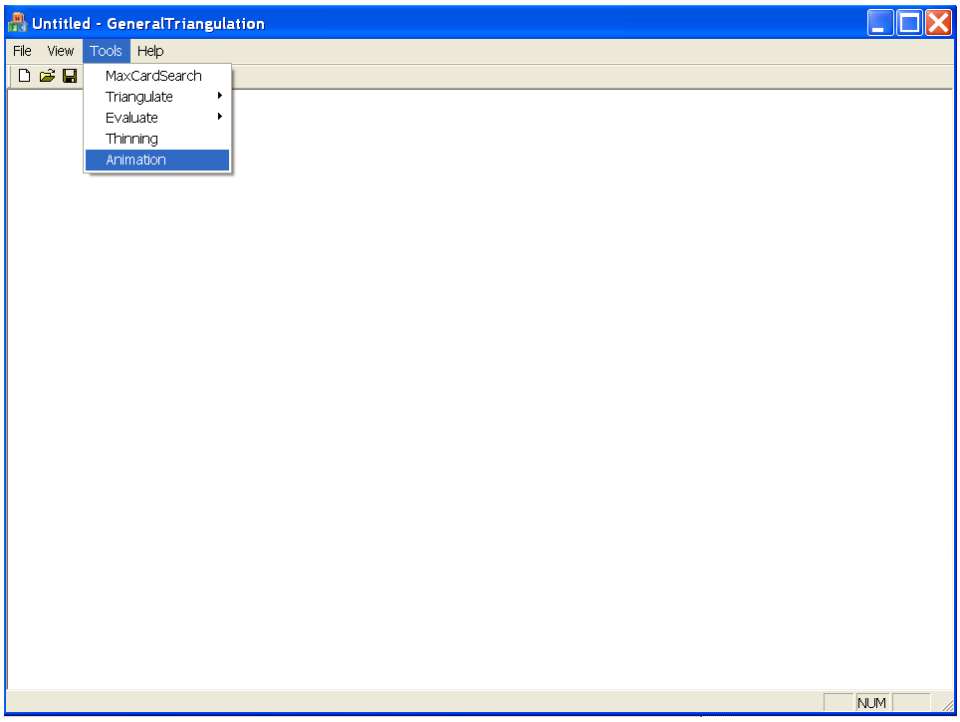


图 11。主界面

包括功能：

1. 文件存盘与读取
2. 图形的显示。
3. Tools 下面是各个算法的实现。
   a) MaxCardSearch 用于检查输入的一般图是否是一个三角化图。
   b) 如图 12（a）所示，Triangulate 子项包括：
      i. Random：随机消除序算法。
      ii. Minimum Fill：最小数目添加边启发算法。
      iii. Minimum Size：最小规模启发算法。
      iv. Minimum Weight：最小权重启发算法。
      v. Simulated Annealing：模拟退火算法。

      点击相应的项即可对输入的一般图进行不同准则下的三角剖分。
   c) 如图 12（b）所示，Evaluate 子项包括：
      i. Random Graph：通过生成指定顶点数以及平均度的随机图对不同算法进行评价，评价结果存到工作目录下的“EvaluationRes.txt”文件中。其中顶点数和平均度通过图 13（a）的对话框设定。
      ii. Determinant Graph：通过输入的特定的图例测试不同算法的性能。算法的选择和参数的设定通过对话框 13（a）完成，其中包括算法的选择，运行次数 Iter Num。测试结果会在弹出的对话框中显示，对话框如图 13（b）所示。
   d) Thinning：用于消除特定三角剖分结果中的冗余边。
   e) Animation：动态的显示各个顶点的消除顺序。以及添加边的添加顺序。
   f) 对于输入的图例，通过在指定的顶点上面点击右键会弹出如图 14（a）所示的不同的备选值，选中之后对应的顶点大小会随作状态空间的维数而变化，如图

14（b）所示当选中的顶点状态空间的维数设置为 8 时的结果。

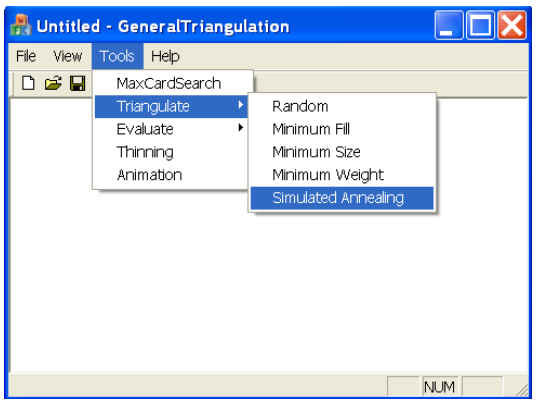


图 12（a）Triangulate 子项

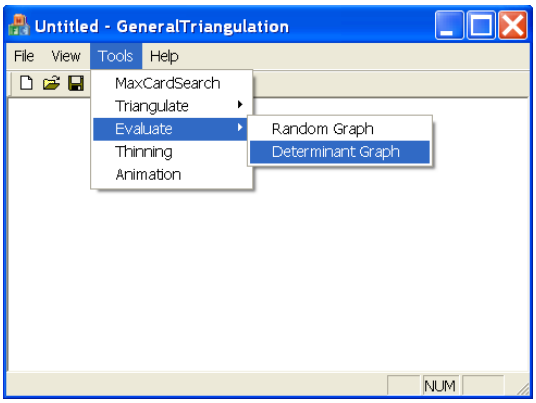


图 12（b）Evaluate 子项

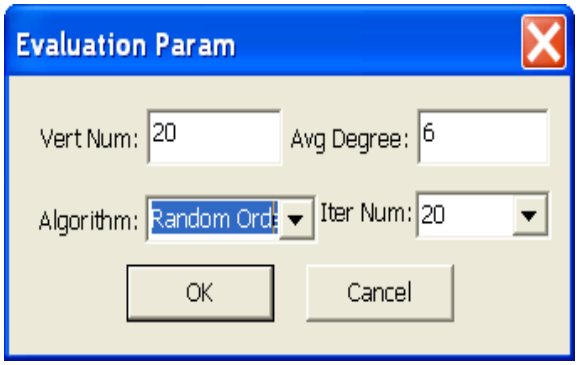


图 13（a）测试参数设置界面

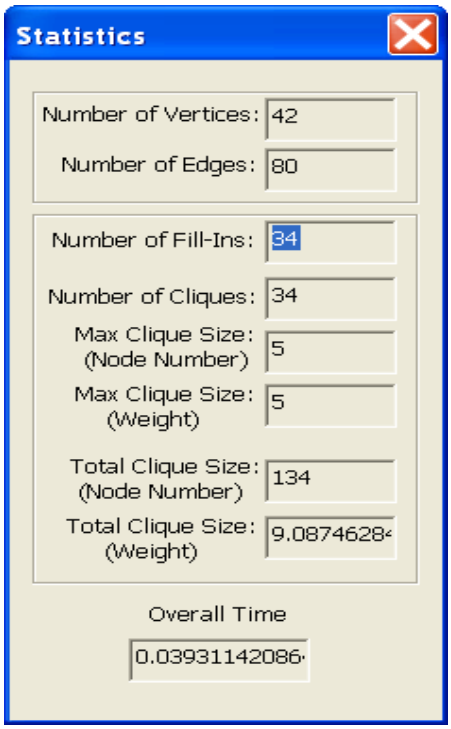


图 13（b）。测试结果显示界面

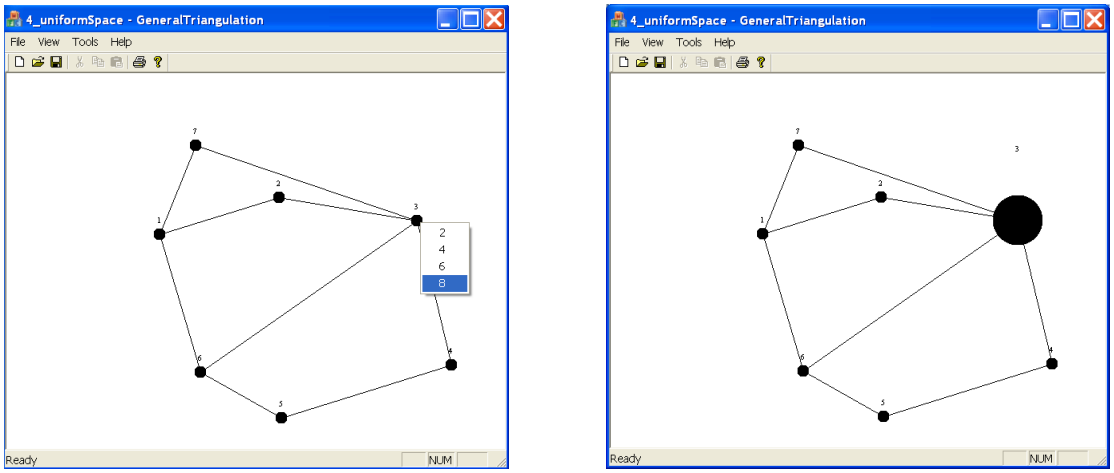


图 14。右键设置不同顶点的状态空间的维数。

参考文献：

[1]. Probabilistic Networks and Expert Systems. R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. Springer.

[2]. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. D. J. Rose. In *Graph Theory and Computing*, R. Read, ed., Academic Poress, New York, 1973, pp. 183-217.

[3]. Computing the Minimum Fill-in is NP-Complete. M. Yannakakis. *SIAM Journal on Algebraic and Discrete Methods*, Vol. 2, No. 1, 1981.

[4]. Complexity of Finding Embedding in a K-Tree. S. Arnborg, D. G. Corneil, and A. Proskurowski. *SIAM Journal on Algebraic and Discrete Methods*, Vol. 8, No. 2, 1987.

[5]. Simple Linear-Time Algorithms to Text Chordality of Graphs, Text Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. R. E. Tarjan, and M. Yannakakis. *SIAM Journal of Computing*, Vol. 13, No. 3, 1984.

[6]. Optimal Decomposition of Probabilistic Networks by Simulated Annealing. U. Kjærulff. *Statistics and Computing* 2: 7-17, 1992.

[7]. Triangulation of Graphs – Algorithms Giving Small Total State Space. U. Kjærulff. 1990.

[8]. Optimal Junction Tree. F. V. Jensen, and F. Jensen. In *Proceedings of Uncertainty in Artificial Intelligence*, 1994.

[9]. A Critical Point for Random Graphs with A Given Degree Sequence. M. Molloy, and B. Reed. *Random Structures and Algorithms*, 1995.

[10]. Fast Generation of Random Connected Graphs with Prescribed Degrees. F. Viger, and M. Latapy. Preprint, 2005.