探求二维凸包及其应用

许瑞广，余志伟
中国矿业大学(北京)资源学院(100083)
E-mail：lucky_xrg@163.com

摘 要：凸包是计算几何中最普遍、最基本的一种结构，本文介绍了二维凸包的概念和性质，并介绍几种求二维凸包的方法:Gift-Wrapping、Graham-Scan 算法，以及这几种算法的正确性和时间复杂度的分析，最后通过两个实例来简要介绍二维凸包的应用。
关键字：凸包、Gift-Wrapping、Graham-Scan

作为计算几何中第一个被深入研究的几何模型，凸包以其优美的性质带来了广泛的应用，本文将对这个几何模型进行简要的介绍。

1. 凸包的概念和性质

我们首先从一个实例来引入凸包：假设你种了很多树，想用一个篱笆把所有的树都包在里面，出于经济考虑，显然这个篱笆应该是越小越好。给出树的坐标，求出篱笆的最小周长。如图 1－1 所示的篱笆是一个最小的篱笆，而这个篱笆就是这些树的凸包(Convex Hull)。

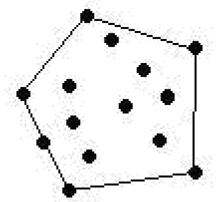

图 1－1 凸包(Convex Hull)

要定义凸包，首先我们来研究一下凸多边形。

定义 1 凸多边形 ⇔ 整个图形在任一条边的一侧。

定义 2 D 是凸多边形 ⇔ $\forall x_1, x_2 \in D, \frac{x_1 + x_2}{2} \in D$，即对于一个凸多边形任意两个内点的中点也在此图形内。我们不仅考虑中点，还考虑所有内分点，于是有如下定义。

定义 3 D 是凸多边形 ⇔ $\forall x_1, x_2 \in D, \forall \lambda \in [0,1], \lambda x_1 + (1-\lambda) x_2 \in D$

因此定义 2 是定义 3 的一种特殊形式。如图 1－2 给出了凸图形和凹图形的图示：

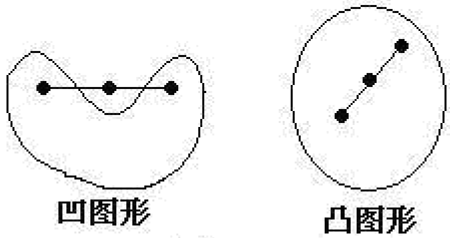

图 2－2 凸图形和凹图形

设S是平面(E^2)上的点集，用CH(S)表示点集S的凸包，BCH(S)表示S的凸包边界。

定义 4 平面点集 S 的凸包 CH(S)是包含 S 的最小凸集，凸包上的顶点称为极点。

点集 S 的凸包是包含 S 的所有凸集的并，或者 CH(S)是包含 S 的所有半空间的交。二维中的半空间是半平面，它是指位于一条直线上及该线一侧的点的集合。

定义 5 平面点集 S 的凸包边界 BCH(S)是一凸多边形，其顶点为 S 中的点。

BCH(S)是包围 S 的最小凸多边形 P，即不存在多边形 P'，使得 $P \supset P' \supset S$ 成立。

BCH(S)是具有最小面积并且封闭的凸多边形西 P，或者是具有最小周长并封闭的凸多边形 P。

由此也验证了实例中最省材料的篱笆即为这些树的凸包。

2. 凸包的实现

2.1 Gift-Wrapping 算法

Gift-Wrapping算法是凸包问题的最直观的一种解法，它是Chand和Kapur于 1970 年提出的，其基本思想如下：首先过y坐标最小的点p_1，画一水平直线l，显然该点是凸包的一个顶点。然后l绕p_1按逆时针方向旋转，碰到S中的第二个点p_2时，直线l改绕p_2按逆时针方向旋转而在p_1与p_2之间留下一条线段，该线段为凸包的一条边。继续旋转下去，最后直线l旋转 360°回到p1，便得到所要求的凸包。

直线l绕点p_1，的旋转是通过如下方法实现的：首先连接pi与非凸顶点p1，$j = \overline{i+1, n}$，得到线段$\overline{p_i, p_j}$，然后求这些线段与l（$\overline{p_{i-1}, p_j}$）的夹角，组成最小夹角的另一端点p_{i+1}即凸包顶点。如图 2－1 所示，Gift-Wrapping算法的基本思想[2]：

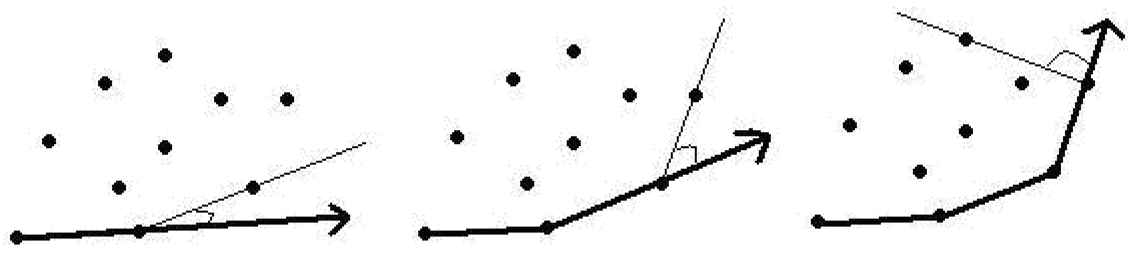

图 2－1 Gift-Wrapping 算法的基本思想

下面给出Gift-Wrapping算法的实现步骤[3]：

Step 1：计算y_1，$y_2 \ldots y_n$，中的最小值，其对于点设为p_1。

Step 2：从p_1向右引一条水平射线，记为l_{p1}。

Step 3：计算 $\overrightarrow{p1pi}$ 与l_{p1}的夹角的最小值所对应的点，记为p_2。

Step 4：j<-1，k<-3，m<-2。

Step 5：以 $\overrightarrow{pjpj+1}$ 代替 $\overrightarrow{pj-1pj}$，计算 $\overrightarrow{pj+1pi}$，$\overrightarrow{pjpj+1}$ 的夹角最小值所对应的点，记为p_k。

Step 6：j<-j+1，k<-k+1，m<-m+1， goto **step 5**，直至p_k为p_1。

算法中的**Step 1** 耗费n-1 此比较，**Step 2** 需要常数时间，**Step 3** 需计算夹角n-1 次，然后耗费n-2 次比较可以求得α_1。**Step 4**～**6** 循环n-2 次，每次循环需要计算夹角n-i-1 次，比较n-i-2 次，i=$\overline{1, n-2}$。**Step 7** 耗费常数时间。因此，算法的时间复杂度为O(n^2)。

2.2 Graham-Scan 算法

刚才讲的 Gift-Wrapping 算法虽然直观易懂，但是时间复杂度很高，那么有没有其他更高效率的算法呢?Gift-Wrapping 算法的每一步都确定性的得到一条最终凸包上的边，能否换个思路，不追求每步必朝最终凸包前进一步，而是考虑“临时凸包”或“局部凸包”。这种凸包是试探性增长的，目前凸包上的边不一定都是将来凸包上的边，而是通过逐步纠正错误来逼近“最终凸包”的。

例如对于一个简单的凹多边形，我们尝试从p_1开始，沿着多边形的顺序，试探性地增长凸包，如图 2－2 所示：

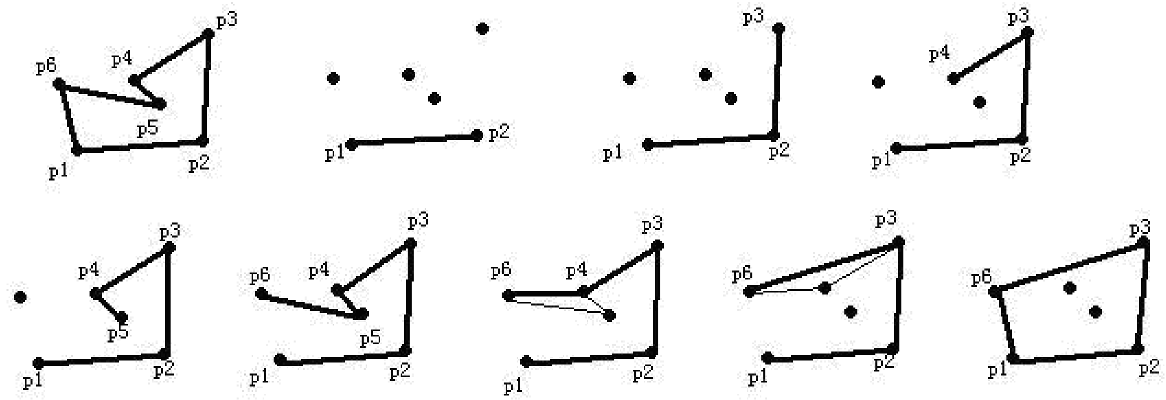

图 2－2 Graham-Scan：试探性增长凸包

选择最低点 p1 作为起点，只要向左转就将这个点添加至临时凸包中，继续扩展，否则回溯，一直到向左转为止。因为用到了回溯，所以算法易用堆栈来实现。

2.3 有关极角序的几个问题

很明显，对于任意一组平面点集，都需要一个序才可以运用 Graham-Scan 算法来解决那么应该是一个什么序呢？最简单的直观的序应该是内部一点的极角序，如图 2－3 所示：

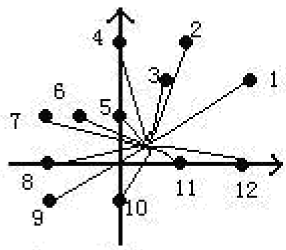

图 2－3 极角序

先假设最简单的情况：没有三点共线。则任意三点组成的三角形内部的点必是凸包内的点，以某个这样内部点 x 为中心，点集中所有点按关于极角逆时针排序，至于排序的其实位置，其实可以任意三点定一条射线，为了简单起见，就以 x 点出发向右延伸，平行于 x 轴的一条射线(xA)为其实位置，于是得到如图 2－3 所示的顺序。具体实现这个排序时，有几种可选的策略。

（1）由于这个序是循环的（绕了 360°），Gift-Wrapping 中的哪个简单的叉积用于此极角比较法以不适用于此种排序。

（2）一种简单的策略是回到原始的极角值。C 语言种有个 arctan 函数，它可以方便的求出每个点相对于 xi 射线的极角（在 0°～360°），但是这会带来浮点误差，对于那种输入全是整数，

不允许任何浮点误差的情况不适用。（当然我们可以设计分数比较的方法，但是显然不漂亮）

（3）比较可行的还是用叉积，不过要注意循环的判断。例如可以得到两个向量分别在 xA 所在的直线的上半平面还是下半平面；如果是同一半，再算两个向量的叉积。

另外，这个算法还有不少问题：起始点和终点的确定、x 点的确定以及三点共线。

（1）起始点和终点的选定

刚才的例子只是一种比较幸运的情况－起点恰好在凸包上，事实上很容易发现第一个入栈的点永不退栈，也就是肯定在凸包中。若起始点事实上不在凸包上时，这个算法就会出错；另外，终点也有这个问题，它也在凸包上，所以我们必须保证起点和终点都在土包上。

（2）排序参考点 x 的确定

前面假设了不存在三点共线，而这种假设时不现实的，就要找到一组不共线的三点，再求其重心，一种简单的做法是每次取三个点，是共线则删去中间那个点，再选。由于每个点最多被删去一次，所以是线性的，不过还是不够简单。

第一种算法针对上述两个问题，有一种很简单的有效的解决方案：把排序的参考点选在一定在凸包上的点，例如最左下的点。这样就很自然的在线性时间内解决了问题(2)。而且对于极角的比较也更简单了，因为这样一来就不存在循环的问题，可以直接用叉积比较。这也就同时解决了问题 1。至此已经得到了一种简单而且高效的平面点集的凸包的求法。

下面给出Graham-Scan算法的伪代码[1]：

```
GRAHAM-SCAN(S)
let p0 be the point in S with the minimum y-coordinate,
    or the left most such point in case of a tie
let(p1,p2…,pm) be the remaining points in S.
    sorted by polar angle in counterclockwise order around p0
    (if more than one point has the same angle, remove all but
    the one that is farthest from p0)
PUSH(p0,STACK)
PUSH(p1,STACK)
PUSH(p2,STACK)
for i<-3 to m
do while the angle formed by points NEXT-TO-TOP(S),TOP(S),
    and pi makes a nonleft turn
do POP(SRACK)
PUSH(pi,STACK)
return S
```

以上算法的排序时间复杂度为 O(nlogn)，根据均摊分析（有关均摊分析，可参见参考文献[1]）得出了扫描的时间复杂度为 O(n)，因此 Graham-Scan 算法的时间复杂度为 O(nlogn)。

水平序和第二种算法以上的算法还存在一定的不足，就是当三点共线的时候解决的还不是很完美，因此我们突破极角序，改用水平序，即按 y 坐标排序，坐标相同的再按 x 坐标排序。如图 2－4 所示，上文按极角排序的可重排：

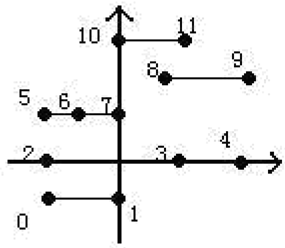

图 2－4 水平序

这样做有两个明显的优势：

（1）排序更简单了，只是简单的比较，没有运算。

（2）其实点更好找了，就是排序后的 0 点。

这样做同样保持了 Graham-Scan 的性质，只不过要把扫描的过程分成两步，右链和左链。先做右链，以 0 点排序最后点（即最高点 11），再反向做左链（已经生成在右链上的点不考虑），从最高点 11 到 0。

最关键的，这样做在几个特殊之处也不错。起边(0-1)、终边(5-2-0)、最高边(11-10)都没问题，对于凸包上共线点的取舍，可以方便地采用与 Gift-Wrapping 相同的严格、不严格比较来控制。进一步观察发现，这种新的 Graham-Scan，甚至连所有点都共线时（最特殊的情况）也成立，可以说时最简单，高效，优美而通用的算法。

水平序 Graham-Scan 的程序见附录。

3. 算法效率的讨论

在上一节中，我们已经对这个算法的时间复杂度做了分析，排序为 O(nlogn)，扫描用均摊分析法，为 O(n)，所以总的时间复杂度为 O(nlogn)。还有没有更快的方法？

定理 设 S 是平面上 n 个点的点集，则计算 CH(S)至少耗费 O(nlogn)的时间。

证明： 设p_1，p_2，…p_n是S中n个点，分类这些点的坐标，设$x_i \neq x_j$，$i \neq j$，$j=\overline{1,n}$。对应于x_i构造点(x_i，x_i^2)，即将点pi移至抛物线$y=x^2$上，用q_l表示。这样，CH（S）由所有点q_l($l=\overline{1,n}$)组成。分类xi需要O(nlogn)时间，而由x_i构造点(x_i，x_i^2)耗费O(n)时间，因此计算CH(S)至少耗费O(nlogn)的时间。

另外，由于凸包的输出可以用于排序，因此如果求凸包的时间小于 O(nlogn)，则突破了排序时间的下界，所以凸包的算法不可能比排序的下界还要快。

4. 凸包的应用

本小节通过两个实例来简要介绍凸包的应用。

【例 4.1】点集分割

判断平面上红蓝两个点集，是否可以被某条直线分割开来，如图 4－1 所示。

原始的做法：任意两个红点连线，称为红线集，任意两个蓝点也连线，称为蓝线集，只要红线、蓝线不存在相交，原点集就是可以分离的，但是这个算法的时间复杂度极高，为$O(n^4)$。那么，有没有更好的方法？

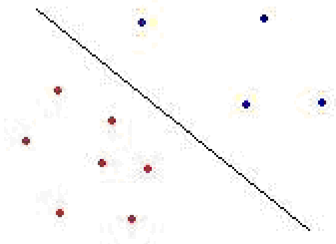

图 4－1 点集分割

首先，我们可以分别求出红、蓝两个点集的凸包，而求凸包之后，只要判断两个凸多边形是否分离，也就是两个凸多边形相交的问题了。这里的做法就是简单的套用一般多边形的相交：是否存在相交的边，若有，则不分离。若没有，也不代表分离，因为可能是包含关系（如图 4－2 所示），只需取一红顶点，看是否也落在蓝凸包内，若没有，再取一蓝顶点，看是否落在红凸包内。

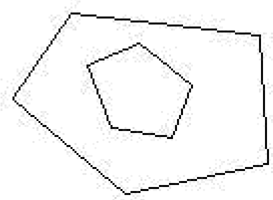

图 4－2 凸包包含情况

上述算法的时间复杂度为$O(n^2)$。

有关凸包分离⇔原点集可被直线分离的证明可参见参考文献[2]。

【例 4.2】合金制造问题

假设你有一些金－银合金，它们的含金量和含银量各不相同，现在要求你通过按某种比例混合，制造出新的合金，那么哪些含金是可能被合成的？更精确的，通过这些现有的合金制造出的新合金，它们的含金量，含银量在什么范围内？

这个问题初看有些难，那么就现看其简化版：

单纯考虑含金量。显然，新造出的合金的含金量只能在原来的合金含金量范围之内，即在原来的最大和最小之间，如图 4－3 所示。

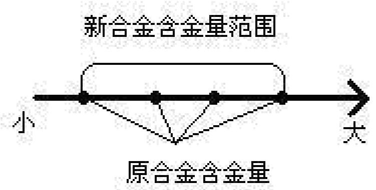

图 4－3 合金制造问题

进一步，如果加上含银量，但假设原来只有两种合金，也很简单，新合金的金/银比例比在原来的合金之间的范围内，精确的讲，原来合金为 A、B。其含金/银比例分别为(xa，ya)，(xb，yb)，则新合金 C:C=λA+(1-λ)B。

从几何上讲，C 必然在线段内，如图 4－4 所示。

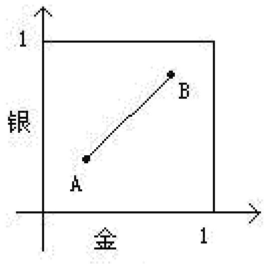

图 4－4 合金为 A、B

那么如果原来 3 个合金 A，B，C 则直观得新合金必在△ABC 内，如图 4－5 所示。

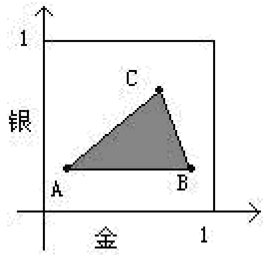

图 4－5 合金为 A,B,C

推广到一般情况，n个合金$A_1A_2…A_n$，则新合金必在这些点的凸包内，如图 4－6 所示。

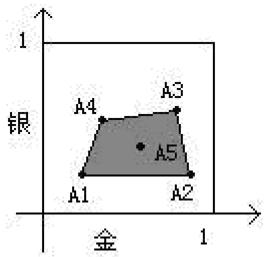

图 4－6 n 个合金 A1A2…An

到这一步，我们已经把一个原本不相关的问题成功转换为凸包问题。

另外更多的有关凸包的例子，可以参见参考文献[2]。

附录

水平序的 Graham-Scan 的程序：

```
/********************************
* Programme: Convex Hull          *
* Algorithms: Graham-Scan         *
* Author: Xu Ruiguang              *
* Date: 2006-05-17                *
********************************/

#include <cstdio>
#include <cstdlib>
#include <cassert>
#include <cmath>
#include <ctime>
#include <cstring>
```

```
#define Limit 100000
#define Precision 0.00001

typedef struct { double x, y; } Point;

int N, _N;
Point S[Limit];                        // Point set S.
Point CH[Limit];                       // The convex hull of point set S.

int InConvexHull[Limit];               // Judge whether the point is in the convex hull.

int stack[Limit];
int top;                               // A stack.

void Init() {                          // Read in the point set S.
    int i;
    scanf("%d", &N);
    for(i = 0; i < N; i ++)
        scanf("%lf %lf", &S[i].x, &S[i].y);
}

int Partition(int l, int r) {
    int k = rand() % (r - l + 1) + l;
    double x = S[k].y, y = S[k].x;
    double tmp;
    l --;
    r ++;
    while(1) {
        do { r --; } while(S[r].y > x || S[r].y == x && S[r].x > y);
        do { l ++; } while(S[l].y < x || S[l].y == x && S[l].x < y);
        if(l < r) {
            tmp = S[l].x; S[l].x = S[r].x; S[r].x = tmp;
            tmp = S[l].y; S[l].y = S[r].y; S[r].y = tmp;
        } else return r;
    }
}

void QuickSort(int l, int r) {
    int x;
    if(l < r) {
        x = Partition(l, r);
        QuickSort(l, x);
        QuickSort(x + 1, r);
    }
}

void EmptyStack() {
    top =   0;
}

void Push(int x) {
    stack[top ++] = x;
}
```

```
void Pop() {
    stack[-- top] = 0;
}

int StackEmpty() {
    return top == 1? 1 : 0;
}

int Dblcmp(double x) {
    if(fabs(x) < Precision)
        return 0;
    return x > 0? 1 : -1;
}

double Det(double x1, double y1, double x2, double y2) {
    return x1 * y2 - x2 * y1;
}

double Cross(Point a, Point b, Point c) {
    return Det(b.x - a.x, b.y - a.y, c.x - a.x, c.y - a.y);
}

void GrahamScan() {                    // Find the convex hull of point S with Graham-Scan.
    time_t seed;                       // rand seed.
    int i;
    srand((unsigned) time (&seed));
    QuickSort(0, N - 1);               // Order the point set S.
    EmptyStack();
    Push(0);
    Push(1);
    i = 2;
    while(i < N) {                     // Find the right chain.
        if(Dblcmp(Cross(S[i], S[stack[top - 1]], S[stack[top-2]])) > 0 || StackEmpty() == 1)
            Push(i++);
        else Pop();
    }
    memset(InConvexHull, 0, sizeof(InConvexHull));
    for(_N = 1; _N <= top; _N ++) {
        CH[_N - 1].x = S[stack[_N - 1]].x;
        CH[_N - 1].y = S[stack[_N - 1]].y;
        InConvexHull[stack[_N - 1]] = 1;
    }
    _N --;
    InConvexHull[0] = 0;
    EmptyStack();
    Push(N - 1);
    Push(N - 2);
    i = N - 3;
    while(i >= 0) {                    // Find the left chain.
        if(InConvexHull[i] == 1) {
            i --;
            continue;
        }
        if(Dblcmp(Cross(S[i], S[stack[top - 1]], S[stack[top-2]])) > 0 || StackEmpty() == 1)
```

```
                Push(i--);
            else Pop();
        }
        for(i = 1; i < top; i ++) {
            CH[_N].x = S[stack[i]].x;
            CH[_N ++].y = S[stack[i]].y;
        }
}

void Output() {                        // Output the convex hull CH.
        int i;
        printf("%d\n", _N);
        for(i = 0; i < _N; i ++)
            printf("%lf %lf\n", CH[i].x, CH[i].y);
}

int main() {
        Init();
        GrahamScan();
        Output();
        return 0;
}
```

参考文献

[1] Thomas H. Cormen, Charles E.Leiserson, Ronald L. Rivest. Introduction to Algorithms. Second Edition. The MIT Press
[2] 刘汝佳, 黄亮. 算法艺术与信息学竞赛. 北京: 清华大学出版社 2004
[3] 周培德.. 计算几何—算法设计与分析(第二版). 北京: 清华大学出版社 2005

Exploring two-dimensional Convex hull and its applications

Xu Ruiguang，Yu Zhiwei
China University of Mining and Technology , Beijing , China ,（100083）

Abstract
Convex hull is the most universal, a basic structure in geometric terms. This article introduced the concept and nature of two-dimensional convex hull, and introduced several approach to protrude two-dimensional convex hull: Gift-Wrapping, Graham-Scan algorithms, and analyze these types of algorithms correctness and time complexity, finally brief introduce the applications of the two-dimensional convex hull through the adoption of two examples.
Keywords: Convex Hull; Gift-Wrapping; Graham-Scan

作者简介：许瑞广（1982--），男，河北黄骅，中国矿业大学(北京)资源学院硕士研究生，专业为环境科学，主要研究方向是环境信息技术。