arXiv:2304.04196v1 [cs.DS] 9 Apr 2023

凸包问题的一个简单高效预处理步骤

Mohammad Heydari
伊斯法罕大学 Khansar 校区计算机科学系，伊朗伊斯法罕 m.heydari@khc.ui.ac.ir

Ashkan Khalifeh亚兹德
大学统计学系，伊朗亚兹德
khalifeh68@yahoo.com

摘要

本文研究了一种递归算法，作为预处理步骤，用于求取平面上均匀分布的 n 个随机点的凸包。对于这样一个点集，研究表明，除去所有点中除 $O(\log n)$ 个点以外的其余点，可以得到与输入集合相同的凸包。最后将证明，该算法的运行时间为 $O(n)$。

关键词 算法 ·凸包 ·递归

1 引言

对于给定的点集 P in $\mathbb{R}^d$， P的凸包是包含 P的最小凸集。在许多领域中，例如生物学 [7],图像处理 [6, 12, 11] 和模式识别 [9]，计算凸包是一个基础问题。在文献中，寻找 n 中的一组 $\mathbb{R}^2$ 点的凸包这一问题已经得到了广泛研究，并提出了许多优雅的算法来解决该问题。在这些算法中，可以提及 Graham扫描算法 [10] 和Quickhull 算法 [4]，它们的运行时间分别为 $O(n\log\ n)$，以及运行时间为 $O(nh)$的Giftwrapping 算法 [13]，其中 h 是凸包上的点数。解决这一问题的一种方法包括一个预处理步骤，该步骤涉及寻找并消除位于凸包内部的一些点。然后将其余点输入任意凸包算法。Quickhull 算法采用了这种方法。此外，Akl和 Toussaint [1] 采用点剔除方法，在最坏情况下以 $O(n\log\ n)$ 的运行时间、在期望情况下以 $O(n)$的运行时间寻找凸包。然而，他们的工作中没有理论结果表示被消除点的数量级。An等人 [3] 和 An [2] 提出了使用点剔除方法的算法。

随机点的凸包有着悠久的历史，可追溯至 1864 年的 Sylvester。关于点服从均匀分布的情形，可参见 Bentley 等人 [5] 以及 Golin 和 Sedgewick [8], ，他们专门研究了预处理步骤。 [5] 和 [8] 中的作者研究了针对均匀分布点的点剔除方法。他们的预处理步骤能够识别并剔除所有点中除 $O(\sqrt{n})$ 之外的点。 [5] 的结果仅在点数超过 2,000,000 时才有效，因此这些结果具有理论研究价值。Golin 和 Sedgewick [8] 研究了单位正方形中的问题，并在 $O(n)$ 时间内剔除了 $O(\sqrt{n})$ 的点。
本文将考虑一种新的递归算法，作为对在 $\mathbb{R}^2$ 中均匀分布的 n 点集合进行点消除的预处理步骤。该算法能在 $O(n)$ 时间内识别并消除除 $O(\log n)$ 个点之外的所有点。与以往算法类似，我们可以将剩余的点输入任意凸包算法。大多数凸包算法处理少量点时比处理大量 [8]，点时效率高得多，因此缩小输入集合的规模具有实际意义。此外，由于输入集合的凸包与得到的 $O(\log n)$ 个点的凸包相同，该预处理算法估计了位于输入集合凸包上的点的期望数量上界。

本文其余部分安排如下：第2节给出预备引理。第3节介绍预处理算法。第4节分析结果并证明该算法的正确性。最后，第5节给出结论。

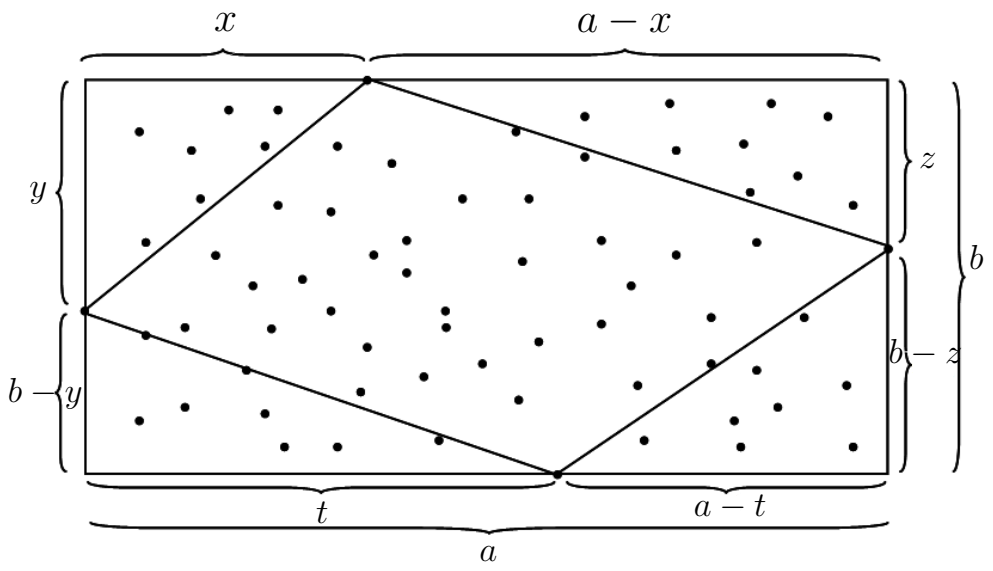


图1：连接极值点形成四边形的示意图。

2 预备知识

本节给出全文所需的定义和引理。

定义： 一组点中的极值点，是指其 x 或 y 坐标在该集合中取最小值或最大值的点，即最上方、最左方、最下方和最右方的点。

在下述引理中，我们考察平面上一组均匀分布点的包围盒，并研究其中点的分布。具体而言，在包围盒的每条边上随机选取一个点，并将其与相邻边上的点连接，由此形成的四边形称为四边形。包围盒内形成的直角三角形称为角部；见图1。

引理1。 考虑平面上一组均匀分布点及其包围盒，如图*1*所示。将包围盒一条边上的点与相邻边上的点连接，会在包围盒内形成一个四边形。四边形内部及各角部的点的期望数量分别为 $\frac{n}{2}$ 和 $\frac{n}{8}$。

证明。 在包围盒的每条边上随机选取一个点。令 $X, T, Y,$ 和 Z 分别表示所选点的随机变量，其观测值分别为 $x, t, y,$ 和 z。因此， X 和 T在区间 $[0, a]$ 上服从均匀分布，记为 $U(0, a)$，而 Y和 Z服从 $U(0, b)$；见图1。鉴于四边形的每个顶点都是随机变量，四边形的面积也是一个随机变量，定义如下

$$N = ab - \frac{1}{2}[XY + (a - X)Z + (b - Y)T + (a - T)(b - Z)].$$

因此，四边形面积的数学期望为

$$E(N) = ab\bigg(1 - \frac{1}{2ab}[E(X)E(Y) + aE(Z) - E(X)E(Z) + bE(T)$$

$$- E(Y)E(T) + ab - aE(Z) - bE(T) + E(T)E(Z)]\bigg).$$

由于任意区间 $[c_1, c_2]$ 上均匀随机变量的数学期望为 $\frac{c_1+c_2}{2}$， $E(X)$ 和 $E(T)$ 为 $\frac{a}{2}$。同理， $E(Y)$ 和 $E(Z)$ 为 $\frac{b}{2}$。因此， $E(N)$ 可表示为

$$\begin{aligned} E(N) &= ab\left(1 - \frac{1}{2ab}[\frac{ab}{4} + \frac{ab}{2} - \frac{ab}{4} + \frac{ab}{2} - \frac{ab}{4} + ab - \frac{ab}{2} - \frac{ab}{2} + \frac{ab}{4}]\right) \\ &= ab\left(1 - \frac{ab}{2ab}\right) \\ &= \frac{1}{2}ab. \end{aligned}$$

因此， N 的面积的期望值为 $\frac{ab}{2}$，其中 ab 是包围盒的面积。由于 N 的面积的期望值是包围盒面积的一半，且点均匀分布，因此 N 中点的数量的期望值是包围盒中点的数量的一半。接下来只需证明每个角的面积的期望值。令 Q 为左上角面积的随机变量。由此可得

$$Q = \frac{1}{2}XY.$$

鉴于 X 和 Y 是相互独立的随机变量，很容易证明

$$E(Q) = \frac{1}{2}E(X)E(Y) = \frac{1}{2} \times \frac{a}{2} \times \frac{b}{2} = \frac{ab}{8}.$$

用同样的方法，很容易看出每个角中点的数量的期望值为 $\frac{n}{8}$，由此证明得证。 □

在下面的引理中，我们研究直角三角形中点的分布。我们假设每条直角边上至少有一个点，因为否则我们可以找到极值点，并绘制经过这些极值点的新直角边；见图2b。

引理2。假设我们有一个直角三角形，其内部有 $n \geq 4$ 个均匀分布点。设其每条直角边上至少有一个点。若我们在每条直角边上随机取一个点并将它们连接起来，则内三角形内部点的期望值为 $\frac{n}{4}$。

证明。 考虑图2a。令 X_1 和 Y_1 为以下均匀随机变量

$$X_1 \sim U(O, a),$$
$$Y_1 \sim U(O, b)$$

对于所选的点。将三角形 ABC 的面积记为 $S(ABC)$，并将$S(OX_1Y_1)$的数学期望记为 H。然后可以很容易地证明

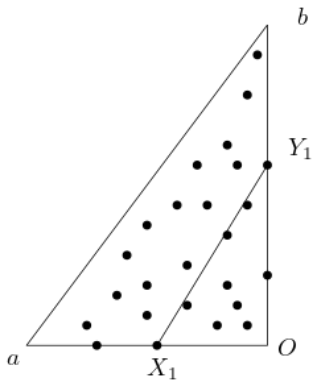
$$E(H) = E(\frac{1}{2}Y_1X_1) = \frac{1}{2}E(Y_1)E(X_1) = \frac{1}{2} \times \frac{b}{2} \times \frac{a}{2} = \frac{1}{8}ab = \frac{1}{4}S(Oab).$$

由于三角形 OY_1X_1 的面积期望值是三角形 Oab面积的四分之一，因此三角形 OY_1X_1 中点的数量期望值为 $\frac{n}{4}$。 □

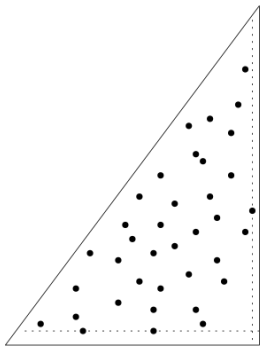
在接下来的 s 本节将介绍所提算法，然后证明其正确性并分析其运行时间。 en.

3 算法

本节将讨论预处理算法。该算法以点集 S 作为输入，并返回点集A，使得集合 S 与集合 A 的凸包保持一致。在全文中，对于给定的点集，将获得的极值点按顺时针顺序命名为 p_t、p_r、p_b、p_l，其中 p_t 的y坐标取最大值。如果存在多个点的 x或 y坐标达到最小值或最大值，则任取其中一点。



(a) 三角形中的点分布。



(b) 由新的直角边构成的直角三角形示意图。

图2：直角三角形。

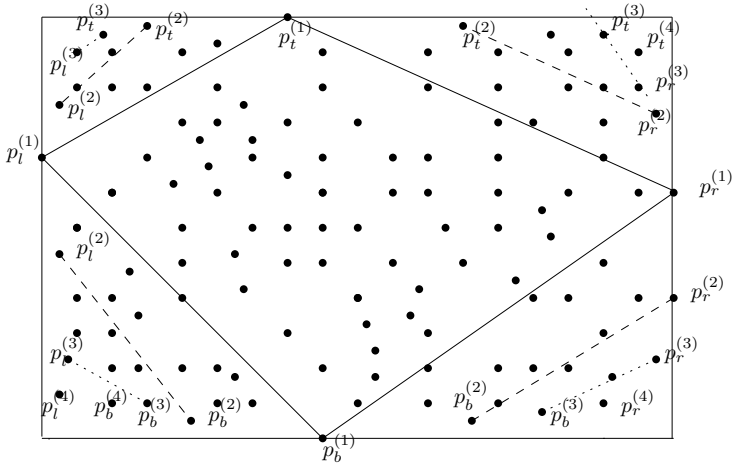


图3：算法执行示意图。

所提算法对点子集进行递归处理，并在 A中选择和存储一些极值点。第 i次递归中的极值点记为 $p_t^{(i)}$、$p_r^{(i)}$、$p_b^{(i)}$、$p_l^{(i)}$。如图3所示，算法首先寻找极值点，并将 $p_t^{(1)}$、$p_r^{(1)}$、$p_b^{(1)}$、$p_l^{(1)}$ 存储在集合 A中。然后，对于右上角进行递归处理，寻找极值点，并将 $p_t^{(2)}$、$p_r^{(2)}$存储在 A中。对于右下角、左下角和左上角，算法分别将 $p_r^{(2)}$、$p_b^{(2)}$，$p_b^{(2)}$、$p_l^{(2)}$ 以及 $p_l^{(2)}$、$p_t^{(2)}$ 存储在 A中。算法重复此过程：对于右上角的线段$p_t^{(2)}$、$p_r^{(2)}$上方的点，以及左上角的线段 $p_t^{(2)}$、$p_l^{(2)}$ 上方的点；对于右下角的线段 $p_r^{(2)}$、$p_b^{(2)}$ 下方的点，以及左下角的线段 $p_b^{(2)}$、$p_l^{(2)}$ 下方的点，持续进行，直到剩余点数不超过两个。

上述算法的伪代码见算法1。字符串 TR、TL、BR 和 BL 分别表示图1中的右上、左上、右下和左下直角三角形。

4 正确性与分析

本节将在定理3中证明该算法的正确性。在定理4中，将估算算法存储的点数的期望值。在定理5中，我们将证明该算法的运行时间为 $O(n)$。

定理3. 集合 A包含输入集合凸包上的每一个点。

证明。 在第一步中，算法存储输入集合的极值点。我们针对右上角证明该定理；同样的方法很容易推广到其他角部。在第 i次递归中，对于 $i \geq 2$，位于.....下方的点

算法1预处理算法

输入： 由 n 个点组成的集合 S

输出： 由点组成的集合 A

如果 $|S|<=3$ **则**
  返回 S
寻找极值点 p_t, p_r, p_b, p_l。
设

S_1： 线段 $p_t p_r$上方的点
S_2：线段下方的点 $p_r p_b$
S_3：线段下方的点 $p_b p_l$
S_4：线段上方的点 $p_l p_t$

$A = \{p_t, p_r, p_b, p_l\}$
点选择(S_1, "TR")
点选择(S_2, "BR")
点选择(S_3, "BL")
点选择(S_4, "TL")
过程 点选择(S_r, $corner$)
  如果 $|S_r|<=2$ **那么**
    $A = A \cup S_r$
    返回
  查找极值点 p_t,p_r,p_b,p_l。
  对于点集 S_r，令

S_1： 线段 $p_t p_r$上方的点
S_2： 线段 $p_r p_b$下方的点
S_3： 线段下方的点 $p_b p_l$
S_4： 线段上方的点 $p_l p_t$

  如果角 ==“TR”**则**
    $A = A \cup \{p_t, p_r\}$
    点选择(S_1, "TR")
  如果角 == “右下角”**则**
    $A = A \cup \{p_r, p_b\}$
    点选择(S_2, "BR")
  如果角 =="BL" **那么**
    $A = A \cup \{p_b, p_l\}$
    点选择(S_3, "BL")
  如果角 =="TL" **那么**
    $A = A \cup \{p_l, p_t\}$
    点选择(S_4, "TL")
结束过程

线段 $p_t^{(i)} p_r^{(i)}$ 位于四边形 $p_t^{(i-1)} p_t^{(i)} p_r^{(i)} p_r^{(i-1)}$ 内部，因此会从后续计算中移除。由于它们位于四边形内部，因此不可能出现在 S 的凸包上。因此，集合 A 包含 S 凸包上的所有点。

□

在定理4中，我们证明集合 A 的期望大小为 $O(\log n)$。根据定理3，由于集合 A 包含输入集合凸包上的每个点，因此可以得出，仅保留 $O(\log n)$ 个点，就能得到与输入集合相同的凸包。

定理4。 集合 A 的预期大小为 O(对数 n)。

证明。根据引理1，每个角处的期望数量为 $m = n/8$，并且在算法的第一步中有一半的点被消除。引理2还指出，在算法的每次递归中，每个角处进入下一次递归的期望数量为 $O(m_r/4)$，其中 m_r 是在第 r 次递归时角内剩余点的数量。因此，算法对一个角的期望迭代次数为 $O(\log_4 m)$。对于每个角，算法在每次递归中获取并存储2个点。因此， A中存储点总数的上界为

$$O(4 \times 2 \log_4 m) = O(\log n).$$

□

在以下定理中，我们证明该算法的预期运行时间是线性的 r.

定理5。 该算法的预期运行时间为 $O(n)$。

证明。算法的运行时间包括第一步的运行时间，即 $O(n)$，以及每次递归的运行时间。令 $m = \lceil n/8 \rceil$。算法的预期迭代次数为 $O(\log_4 m)$。算法第 15 行确定点相对于线段的位置，其运行时间为 $\Theta(m)$。寻找极值点 p_t， p_r， p_b， p_l也需要 $\Theta(m)$。因此，每次递归的运行时间为

$$T(m) = T(m/4) + \Theta(m),$$

并且根据主定理，这个递归方程的解为 $O(m)$。因此，总运行时间为:

$$T(n) = O(n) + O(m) = O(n) + O(4 \times n/8) = O(n).$$

□

5 结论

本文提出了一种高效而简单的递归算法，作为凸包问题的预处理步骤。我们假设点在平面上服从均匀分布、随机分布。该算法在 $O(n)$中消除除 $O(\log n)$ 个点之外的所有点；据我们所知，与所有以往保留 $O(\sqrt{n})$个点的类似工作相比，该算法有所改进。随后，将剩余的点输入任意现有的凸包算法。

参考文献

[1] S. G. Akl 和 G. T. Toussaint，快速凸包算法，信息处理快报**7**(5) (1978) 219–222.[2] P. T. An，用于确定平面上有限点集凸包的曲线定向方法，优化**59**(2) (2010) 175–179.[3] P. T. An、P. T. T. Huyen 和 N. T. Le，一种用于求有限平面点集最小连通正交凸包的改进 Graham 扫描算法，应用数学与计算**397**(1) (2021) 125889.[4] C. B. Barber、D. P. Dobkin 和 H. Huhdanpaa，用于凸包的 Quickhull 算法，*ACM* 数学软件汇刊**22**(4) (1996) 469–483.[5] J. L. Bentley、K. L. Clarkson 和 D. B. Levine，用于计算极大值和凸包的快速线性期望时间算法，算法学**9**(2) (1993) 168–183.[6] P. Bhaniramka、R. Wenger 和 R. Crawfis，使用凸包构造任意维等值面，*IEEE* 可视化与计算机图形学汇刊**10**(2) (2004) 130–141.[7] F. L. Bookstein，地标数据的形态测量工具：几何与生物学（剑桥大学出版社，1992）。[8] M. Golin 和 R. Sedgewick，简单而高效的凸包算法分析，*SCG '88*：第四届计算几何年度研讨会论文集(1988) 153–163.[9] C. Gope 和 N. Kehtarnavaz，使用凸包和 Hausdorff 距离对点集进行仿射不变比较，模式识别**40**(1) (2007) 309–320.[10] R. L. Graham，一种确定有限平面点集凸包的高效算法，信息处理快报**1**(4) (1972) 132–133.[11] M. A. Jayaram 和 H. Fleyeh，图像处理中的凸包：范围综述，美国智能系统期刊**6**(2) (2016) 48–58.[12] A. Sarkar、A. Biswasand、M. Dutt、P. Bhowmick 和 B. B.Bhattacharya，一种计算数字对象三角凸包的线性时间算法，离散应用数学**216**(2) (2017) 408–423.[13] K. Sugihara，三维凸包的稳健礼品包装算法，计算机与系统科学期刊**49**(2) (1994) 391–407.