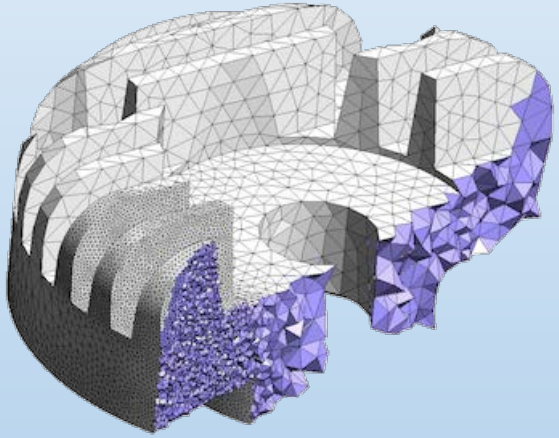
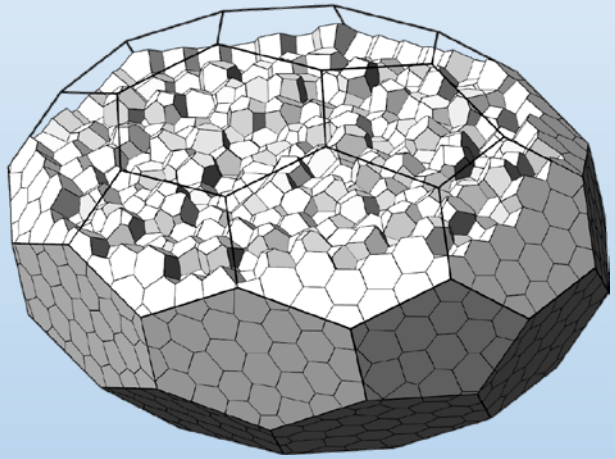
中国科学技术大学2024暑期课程

Delaunay网格与Voronoi图



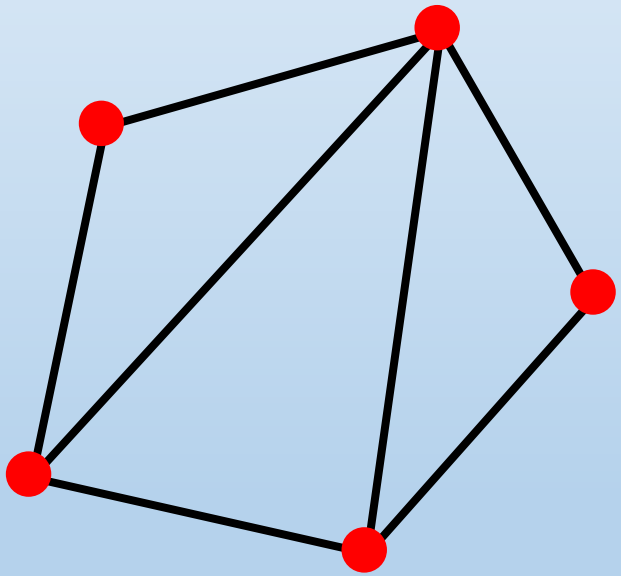


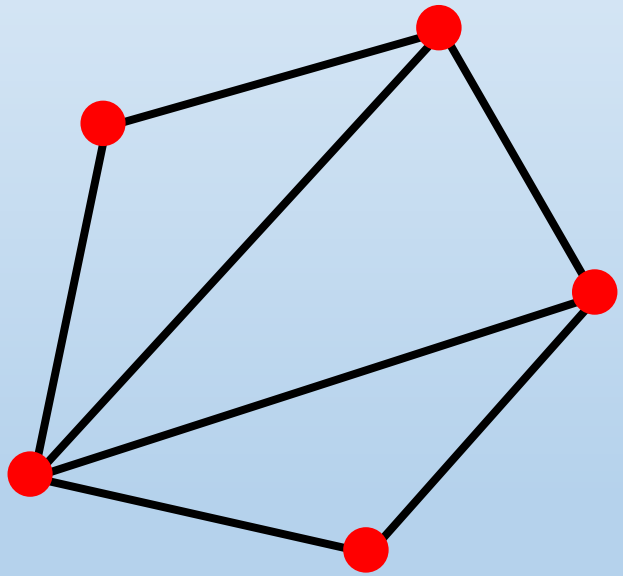
肖艳阳
南昌大学数学与计算机学院
xiaoyanyang@ncu.edu.cn
yanyangxiao.github.io

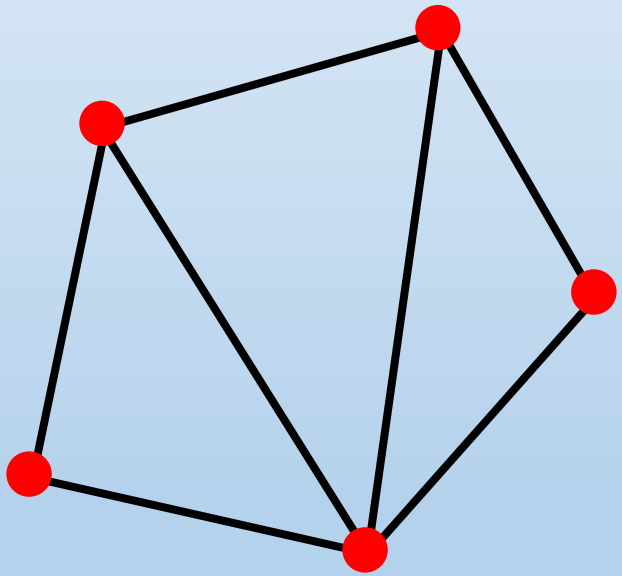
目录

点集→三角网格

- 同一组点有多种不同的连接关系

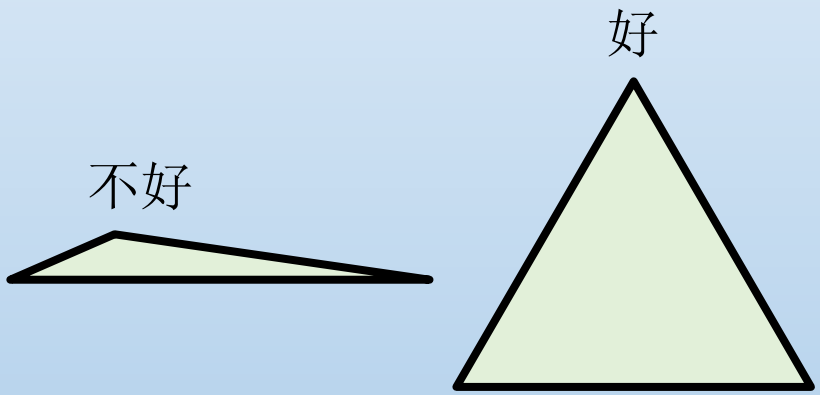




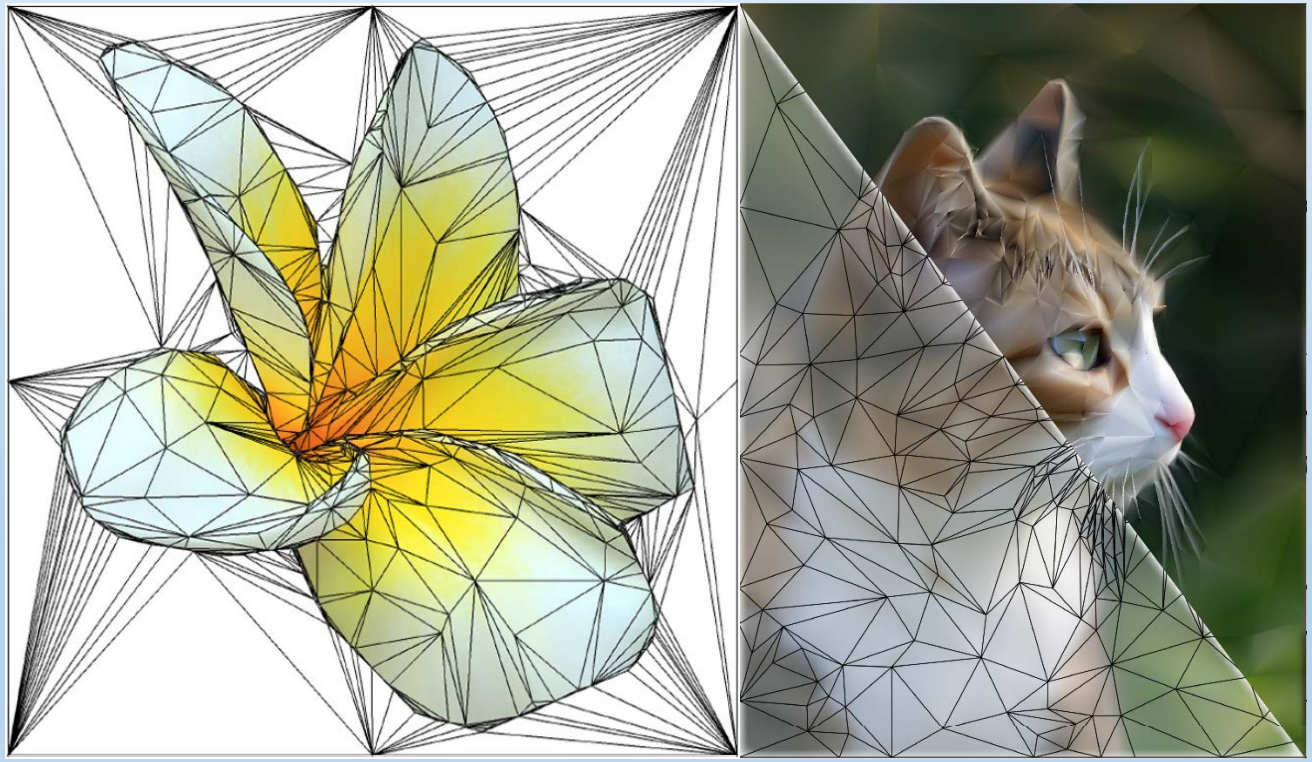


点集→三角网格

- 哪种连接关系“最好”？根据应用需求确定
- 希望获得角度尽可能大的网格→构造Delaunay triangulation



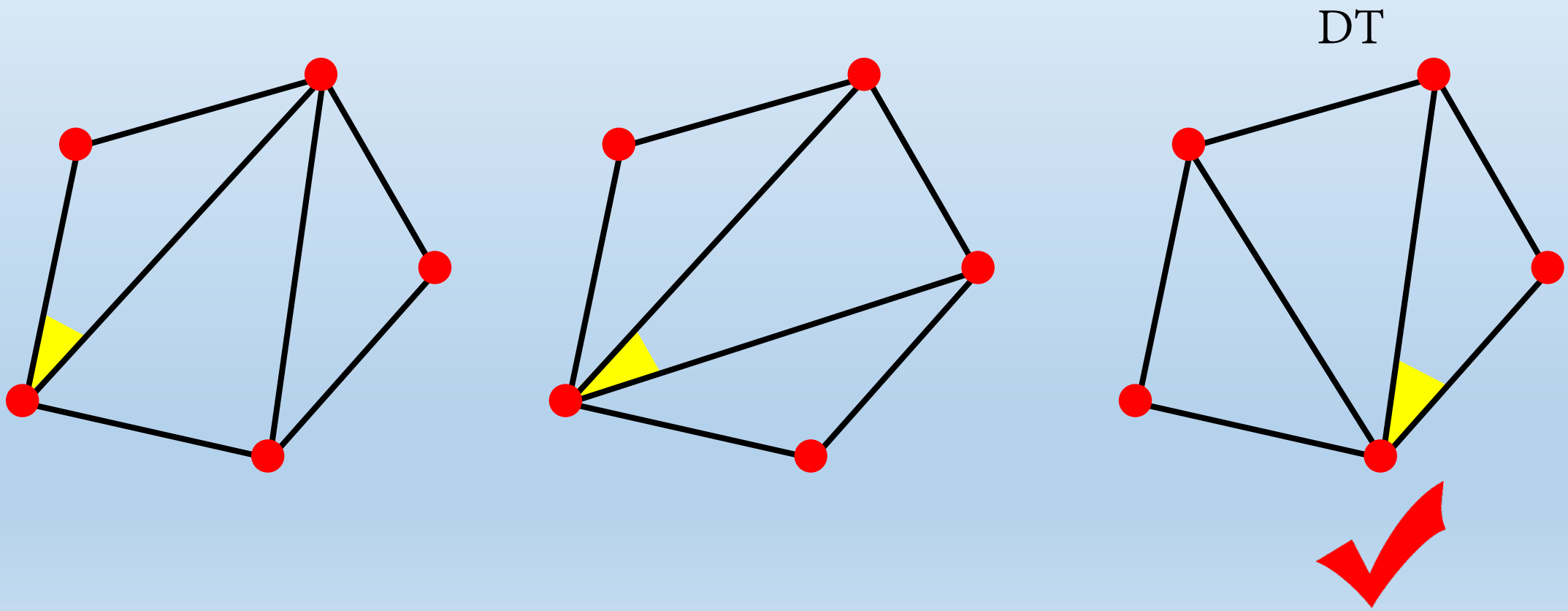
网格生成：不喜欢狭长三角形



图像矢量化/逼近：重建图像与原图误差最低

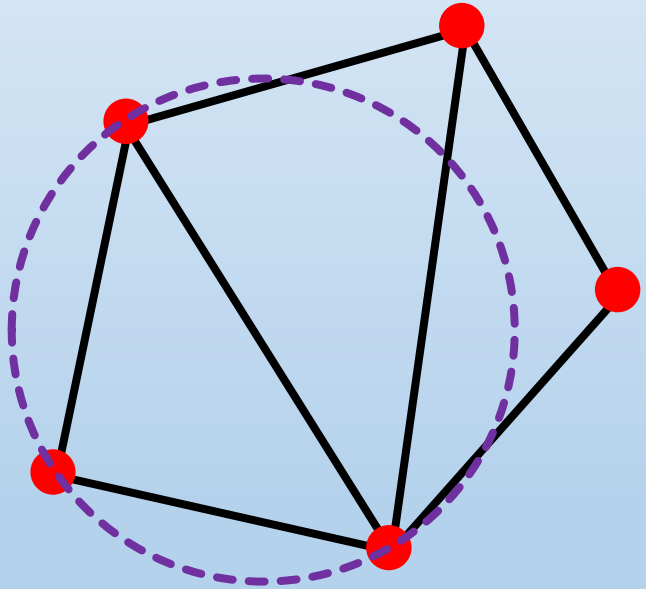
Delaunay Triangulation (DT)

- 性质1：在点集所有可能的三角网格中，DT的最小角度最大



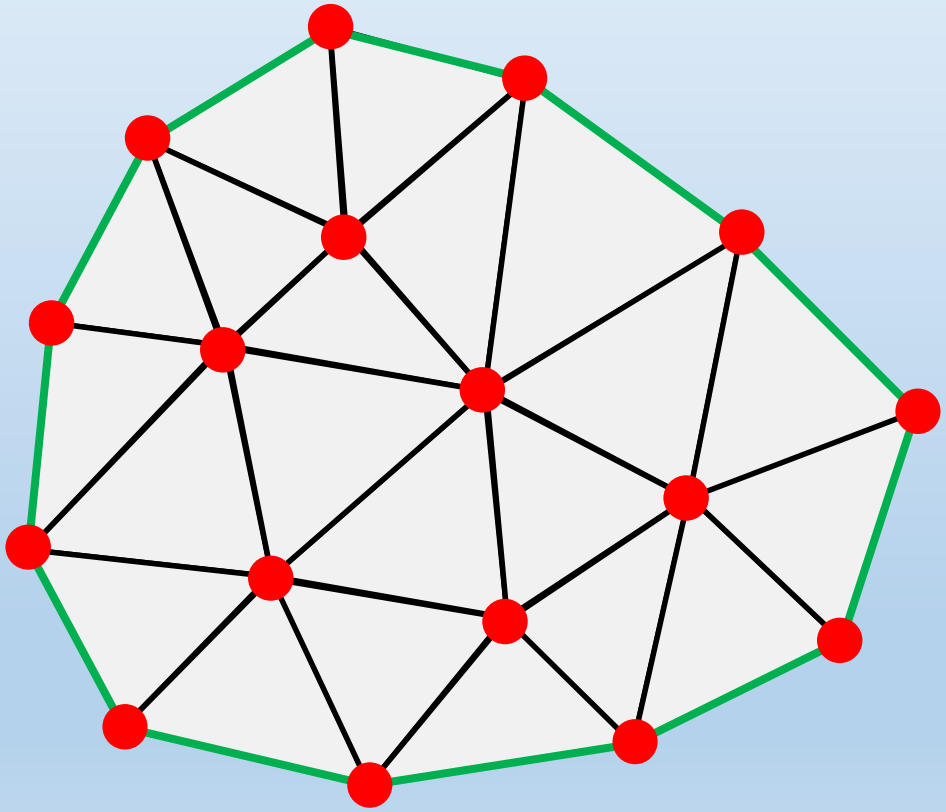
Delaunay Triangulation (DT)

- 性质2：DT每个三角形的外接圆内都不含点集中的点



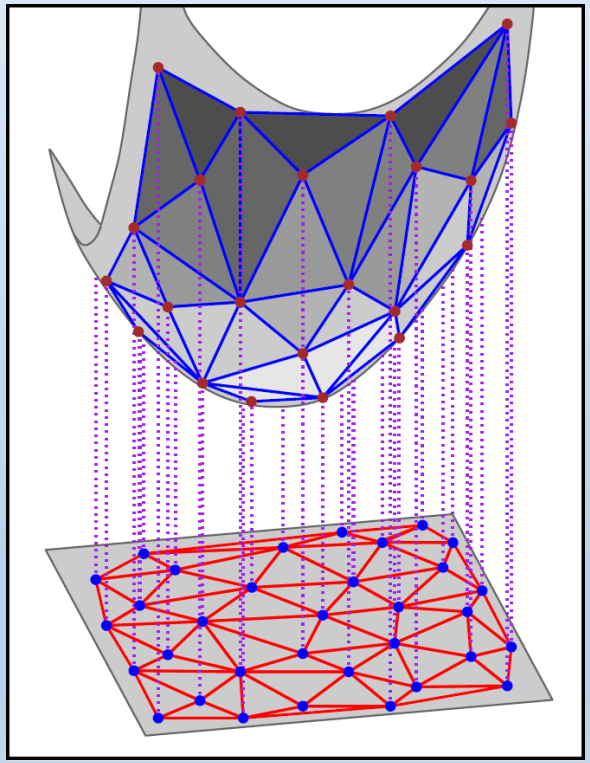
Delaunay Triangulation (DT)

- 性质3：DT网格的边界是该点集的凸包



Delaunay Triangulation (DT)

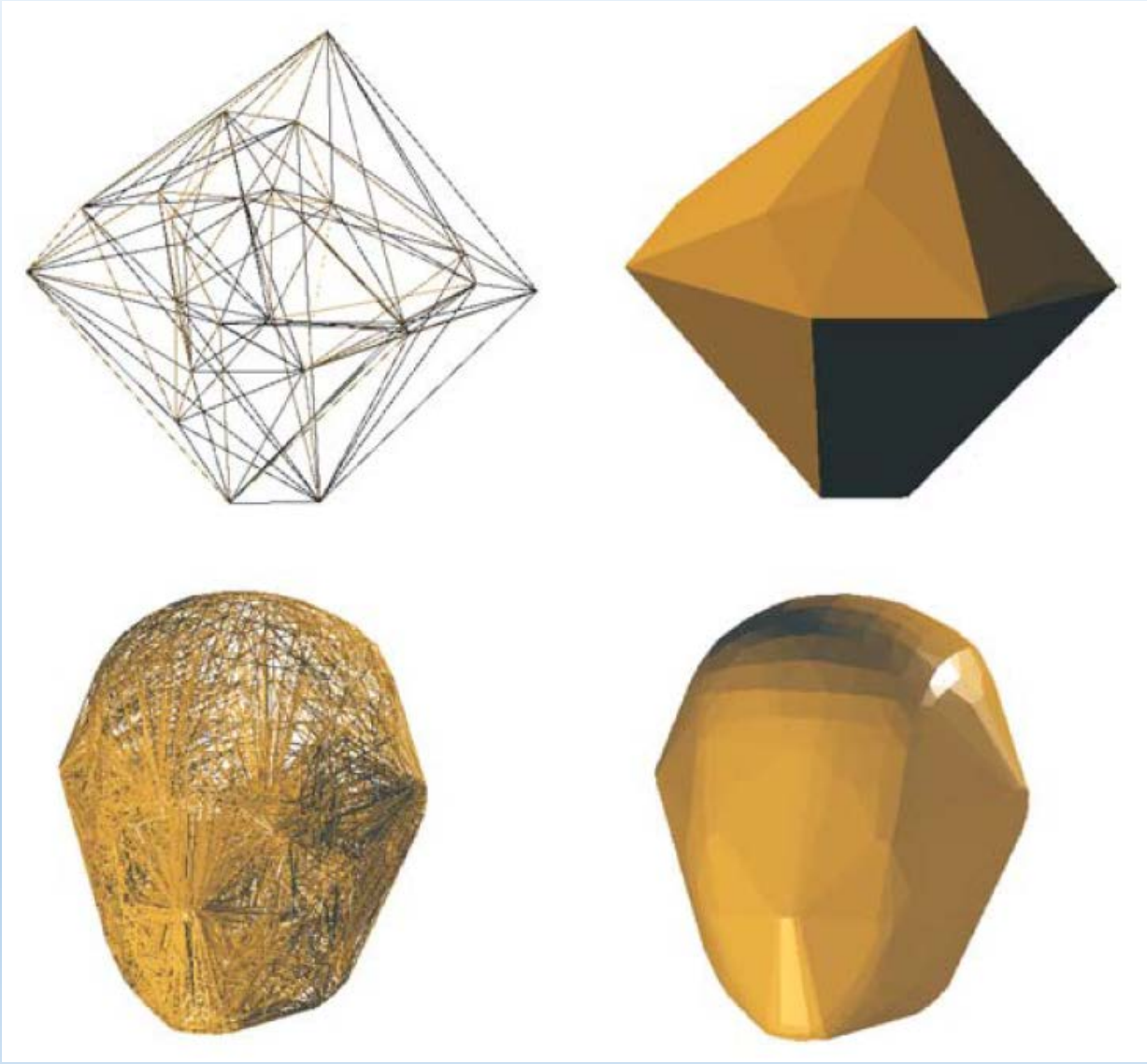
- 性质4：点集每个点提升到高一维抛物面函数上，提升点的凸包是一个多面体，将朝下的面片投影回原空间 = 原点集的DT



H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. Discrete & Computational Geometry, 1986, 1: 25-44.

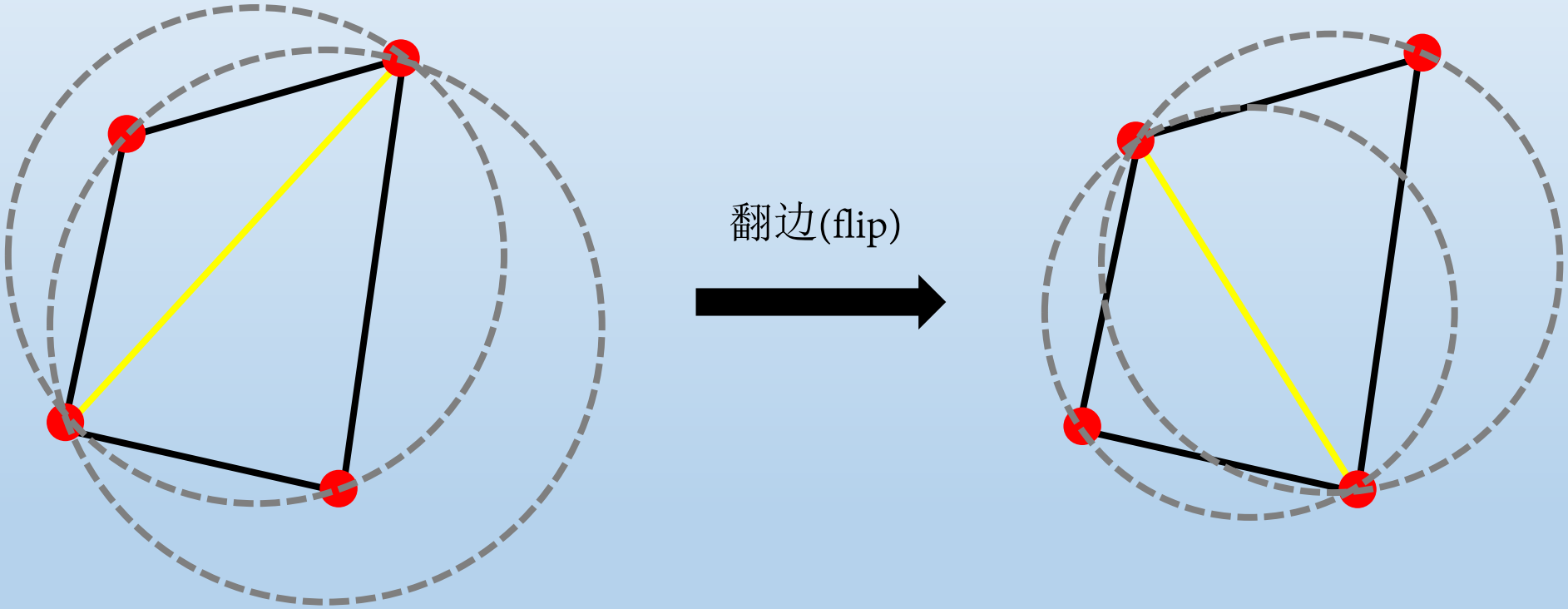
Delaunay Triangulation (DT)

- 高维DT



构造DT： 1-翻边算法 (Lawson/flipping/local improvement)

- 2D翻转操作

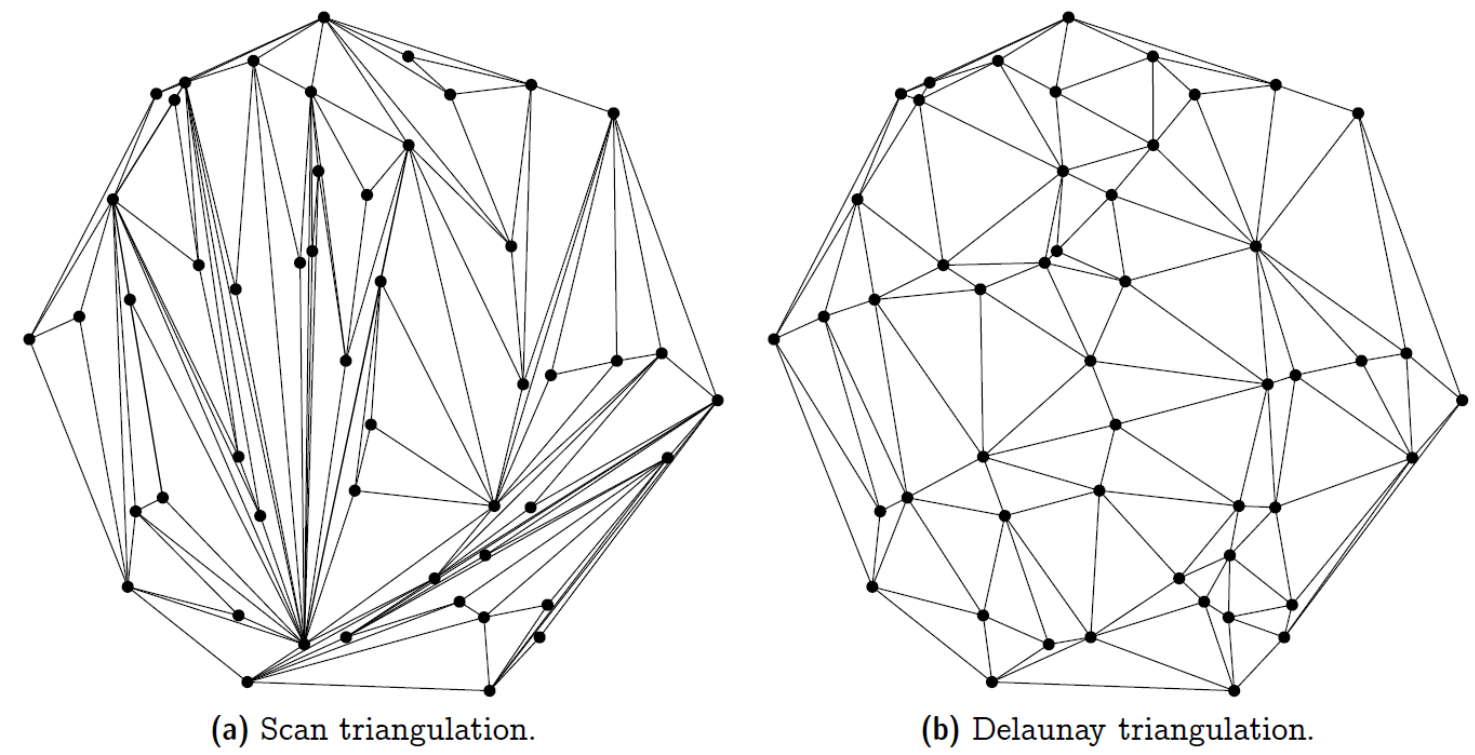


构造DT： 1-翻边算法 (Lawson/flipping/local improvement)

- [Lawson1972]：2D，任意三角网格可经有限次翻边到达DT

-Step1： 构造一个初始三角网格

-Step2： 若存在边不满足Delaunay性质，则对其翻转

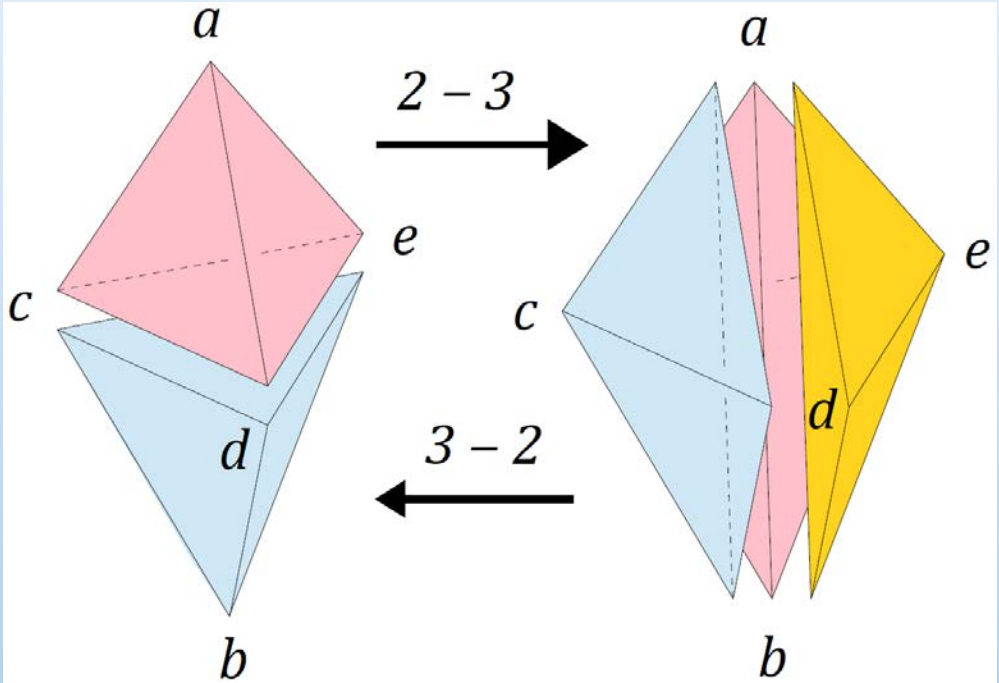


(a) Scan triangulation. (b) Delaunay triangulation.

Charles L. Lawson. Transforming triangulations. Discrete Mathematics, 1972, 3(4): 365-372.

构造DT： 1-翻边算法 (Lawson/flipping/local improvement)

- 3D翻转操作 (bilateral flips)

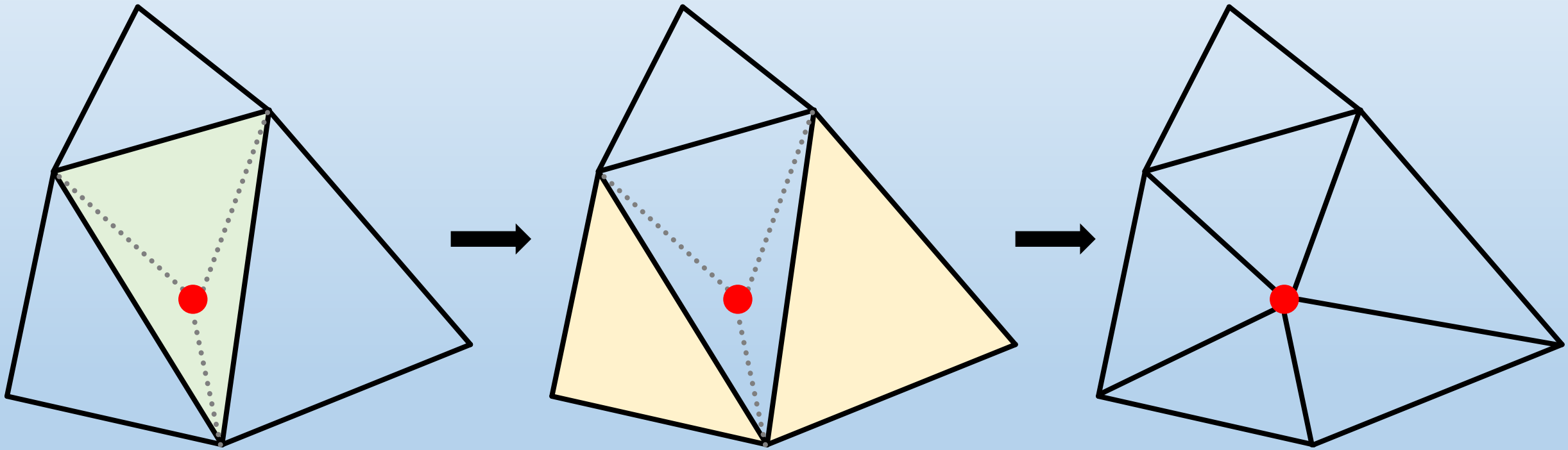


- [Joe1989]： 3D， 该算法可能在到达DT前卡住

B. Joe. Three-dimensional triangulations from local transformations. SIAM J. on Scientific and Statistical Computing, 1989, 10(4): 718.

构造DT：2-增长类算法 (incremental insertion)

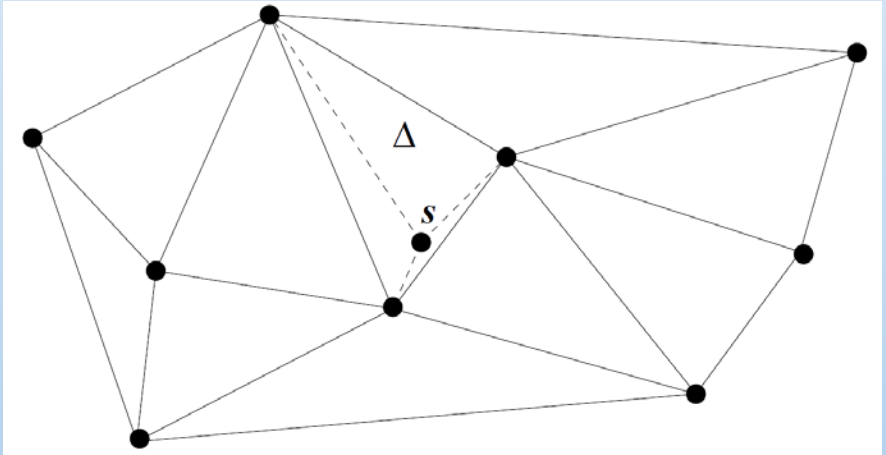
- 逐点插入到网格，并局部更新使其保持为DT

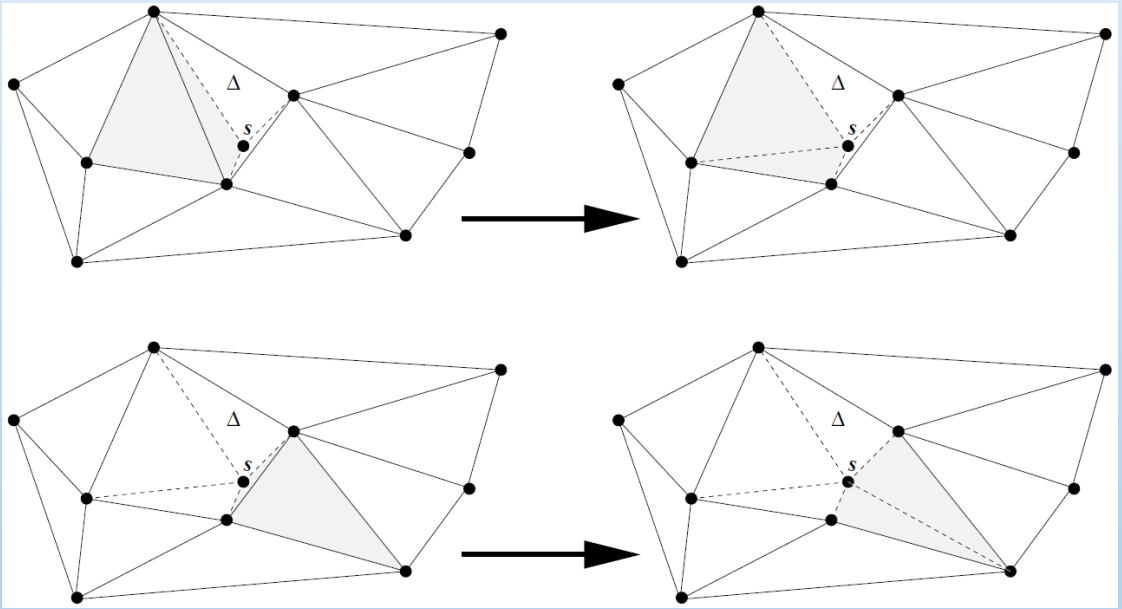


构造DT：2-增长类算法 (incremental insertion)

- 局部更新策略：(1) flip

-点插入到网格后，不断翻转非Delaunay边





需向外扩散测试

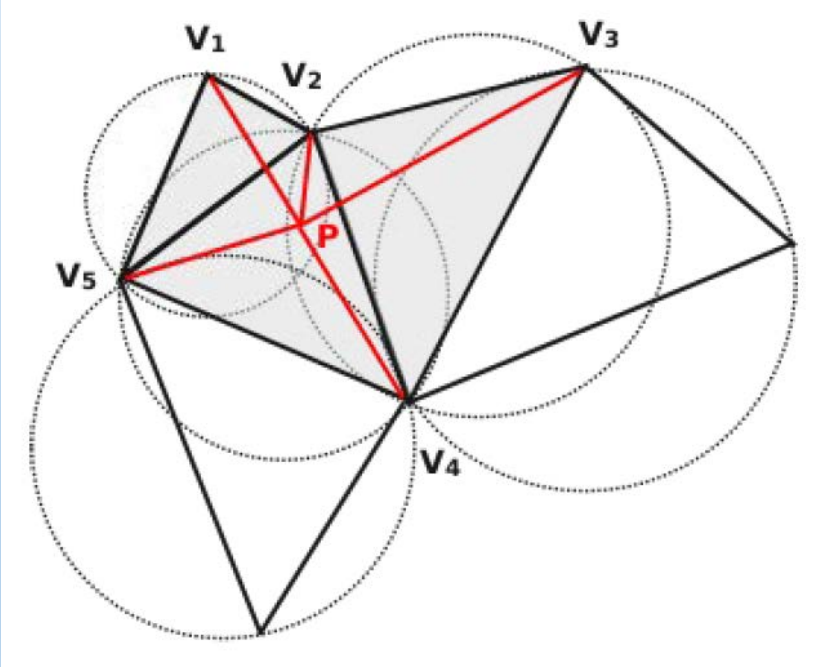
L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. ACM Transactions on Graphics, 1985, 4(2): 75-123.

构造DT：2-增长类算法 (incremental insertion)

- 局部更新策略：(2) Bowyer-Watson

-Step1：找出外接圆包含新点的所有三角形，并删除 ⟶ 空腔 cavity

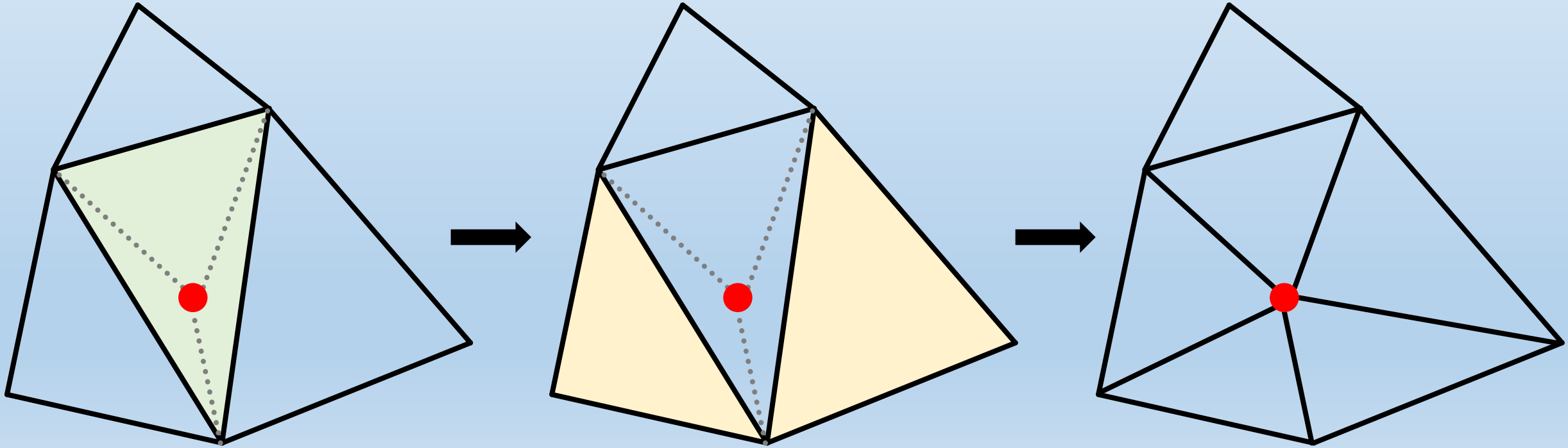
-Step2：对新点与空腔多边形顶点三角化



A. Bowyer. Computing Dirichlet tessellations. The computer journal, 1981, 24(2): 162-166.
D. F. Watson. Computing the n-dimensional Delaunay triangulation with application to Voronoi polytopes. The computer journal, 1981, 24(2): 167-172.

构造DT：2-增长类算法 (incremental insertion)

- 逐点插入到网格，并局部更新使其保持为DT
- [Guibas et al.1992]：插入顺序随机时，翻边次数与点数呈线性，而与点的分布无关
- 适用于构造任意维度的DT



L. Guibas, D. Knuth, M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. Algorithmica, 1992, 7: 381-413.

构造DT：3-高维嵌入算法 (higher dimensional embedding)

- 构建高一维凸包来获得原点集的DT：

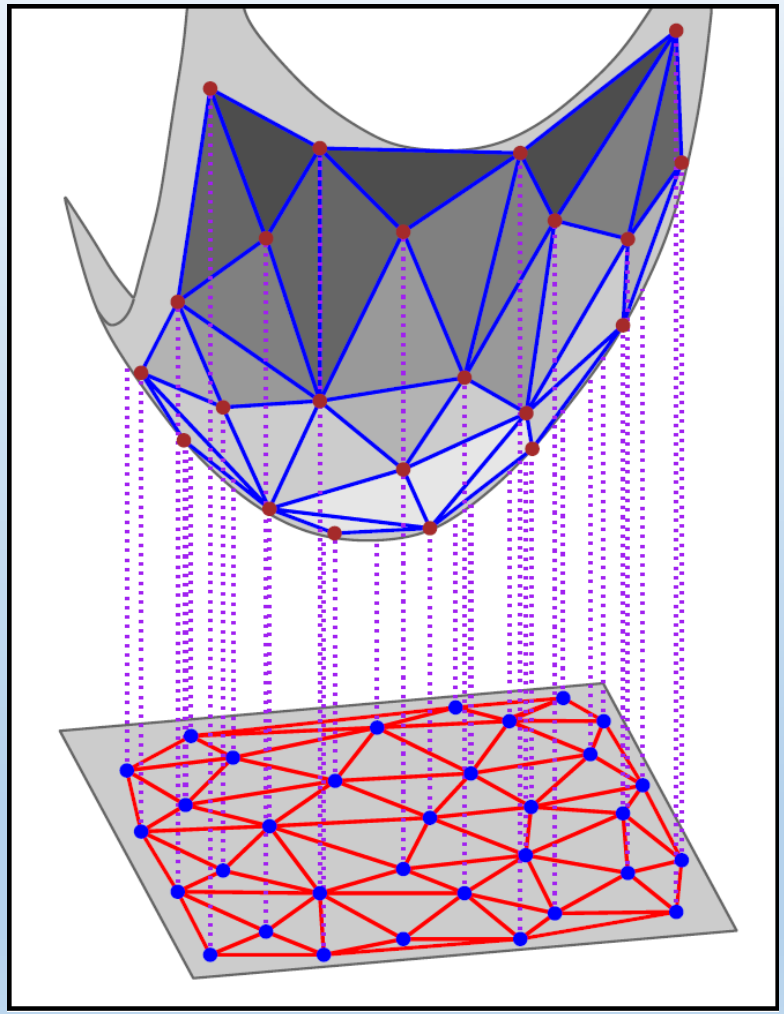
-Step1：每个点提升到高一维抛物面函数上

-Step2：计算提升点的凸包

-Step3：凸包朝下的面片投影回原空间

- 关键：高维凸包的计算

- 适用于构造任意维度的DT

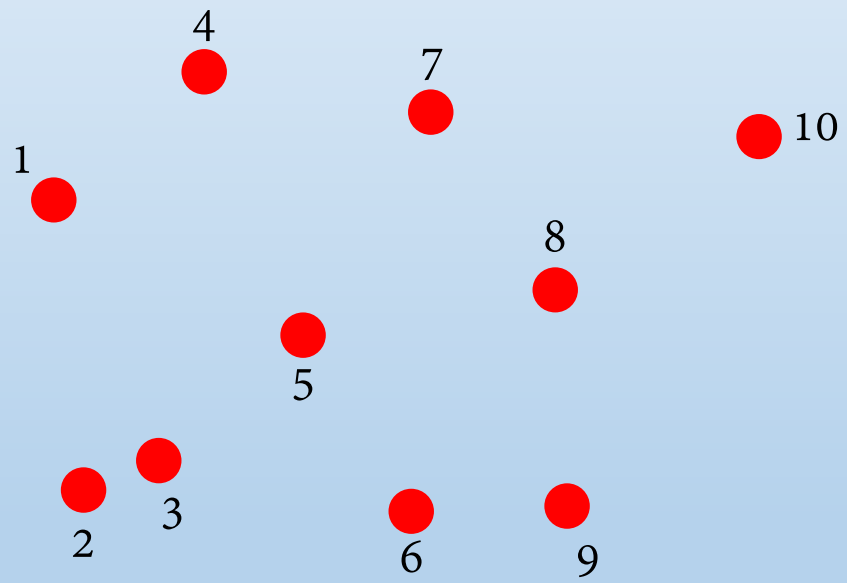


H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. Discrete & Computational Geometry, 1986, 1: 25-44.

构造DT：4-分治法

- 划分点集，独立构造DT，再合并相邻DT形成更大的DT：

-Step1：对所有点按照x坐标升序排序，x相同时按照y坐标升序排序



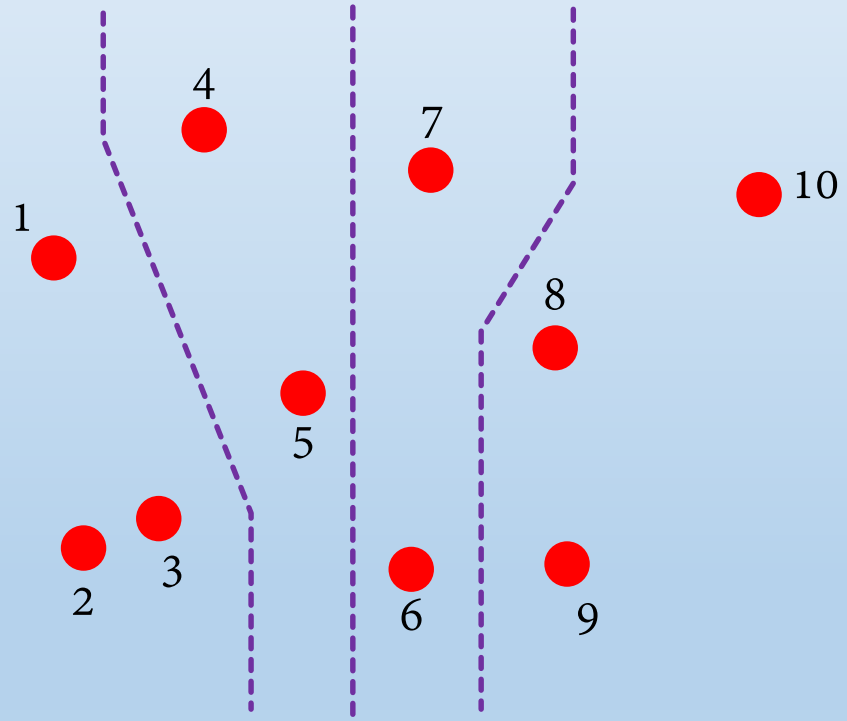
L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. ACM Transactions on Graphics, 1985, 4(2): 75-123.

谢增广． 平面点集Delaunay三角剖分的分治算法. 计算机工程与设计, 2012, 33(7): 2562-2568.

构造DT：4-分治法

- 划分点集，独立构造DT，再合并相邻DT形成更大的DT：

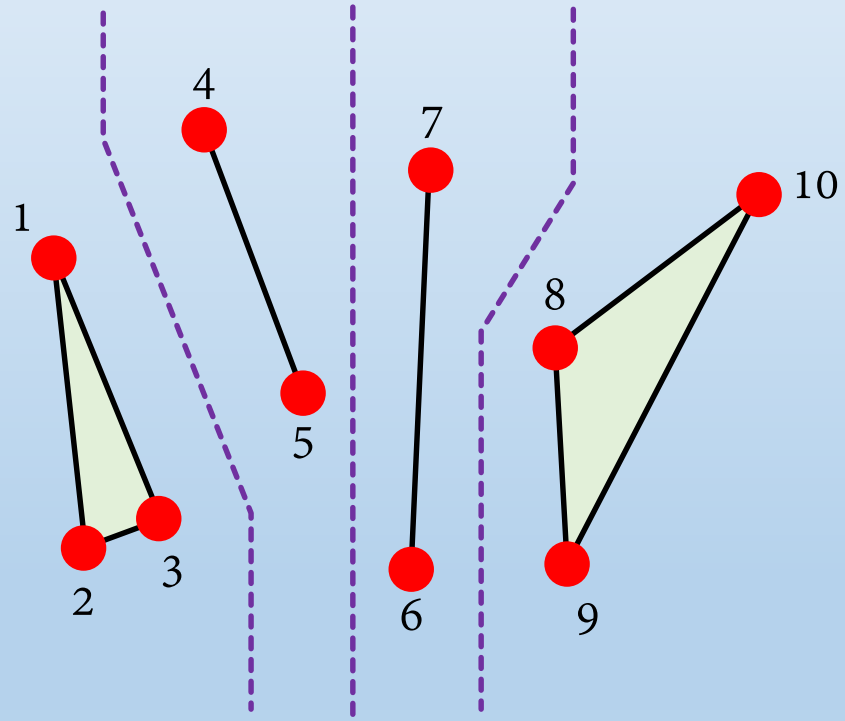
-Step2：划分点集，直至每个子集点的数量不多于3



构造DT：4-分治法

- 划分点集，独立构造DT，再合并相邻DT形成更大的DT：

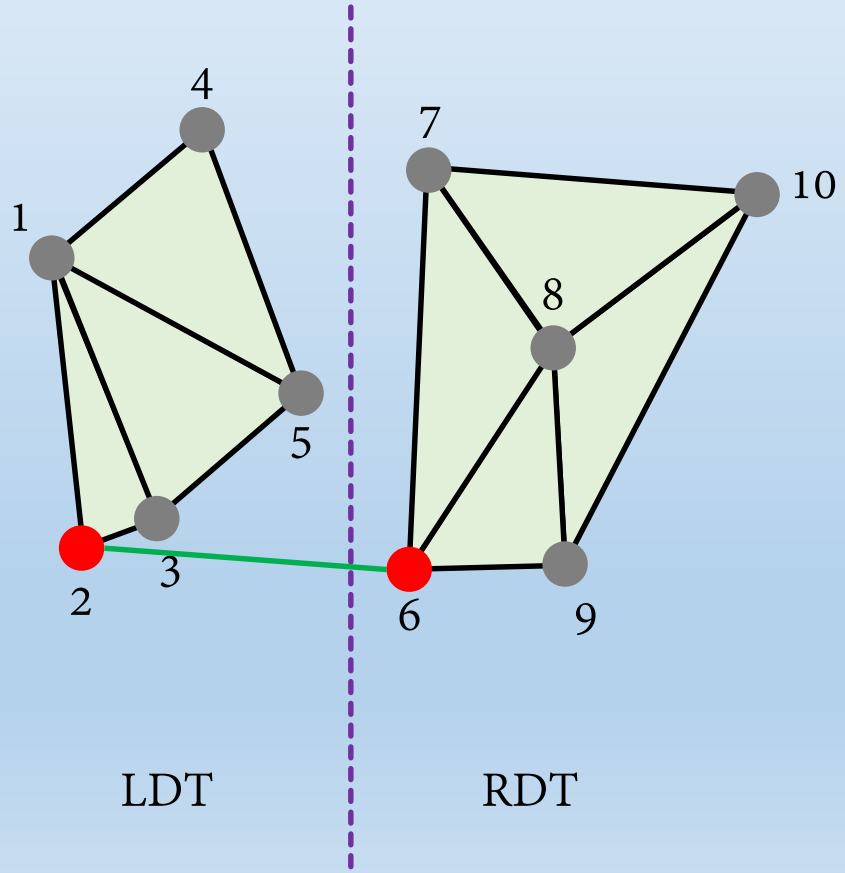
-Step3：每个子集独立构造DT（必为三角形或线段）



构造DT：4-分治法

- 划分点集，独立构造DT，再合并相邻DT形成更大的DT：

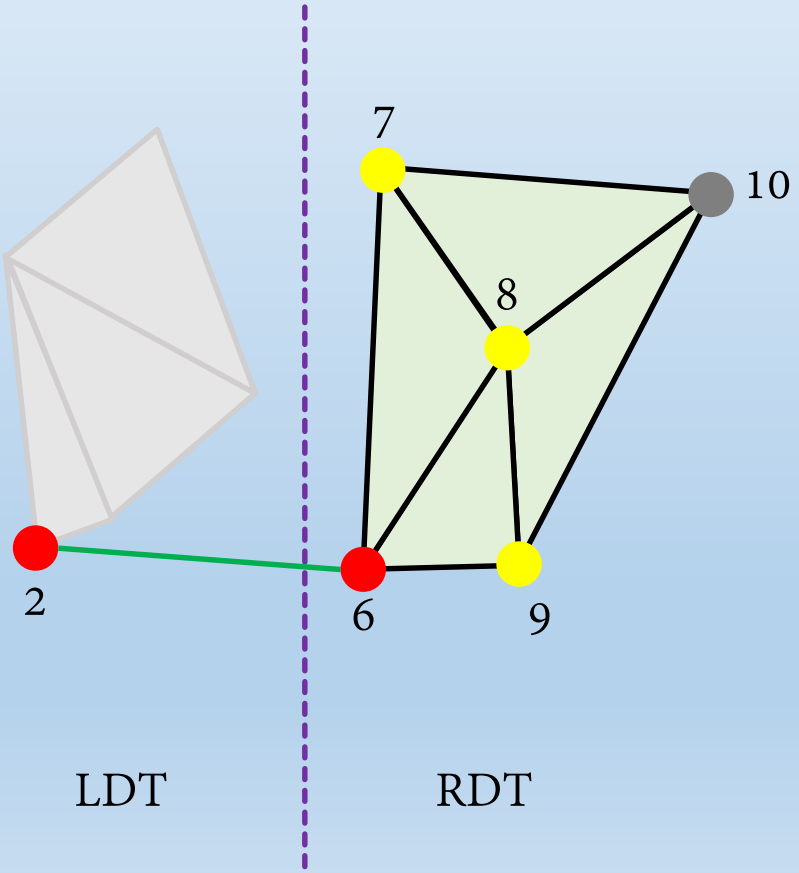
-Step4：不断合并相邻的DT，直至剩余1个DT结果



构造DT：4-分治法

- 划分点集，独立构造DT，再合并相邻DT形成更大的DT：

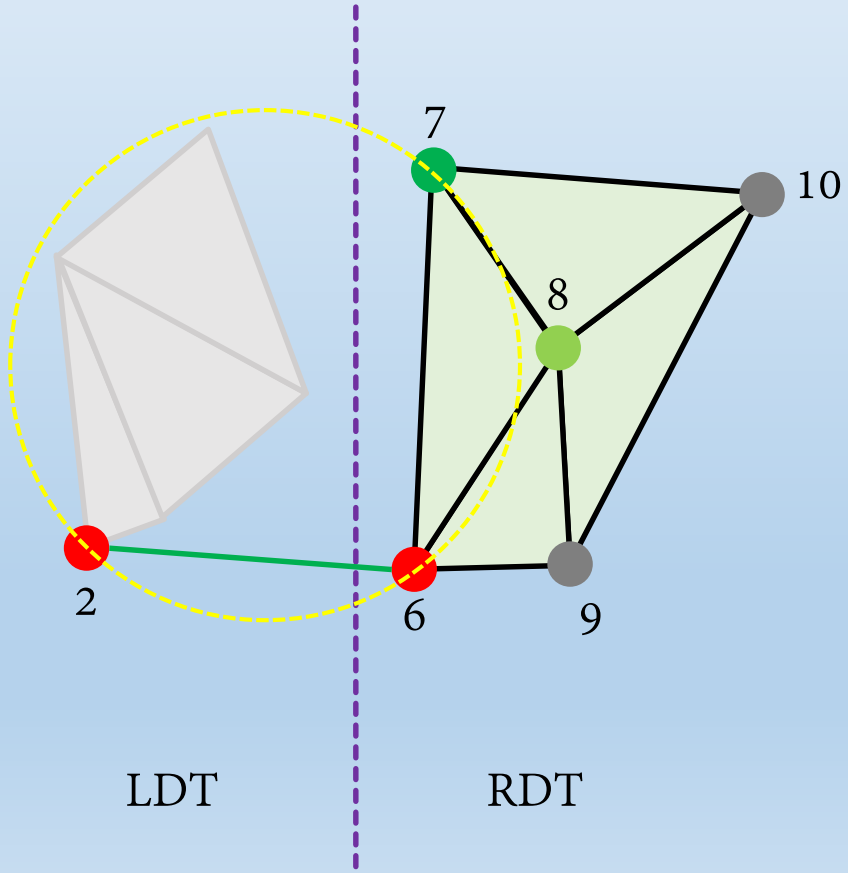
-Step4：不断合并相邻的DT，直至剩余1个DT结果



构造DT：4-分治法

- 划分点集，独立构造DT，再合并相邻DT形成更大的DT：

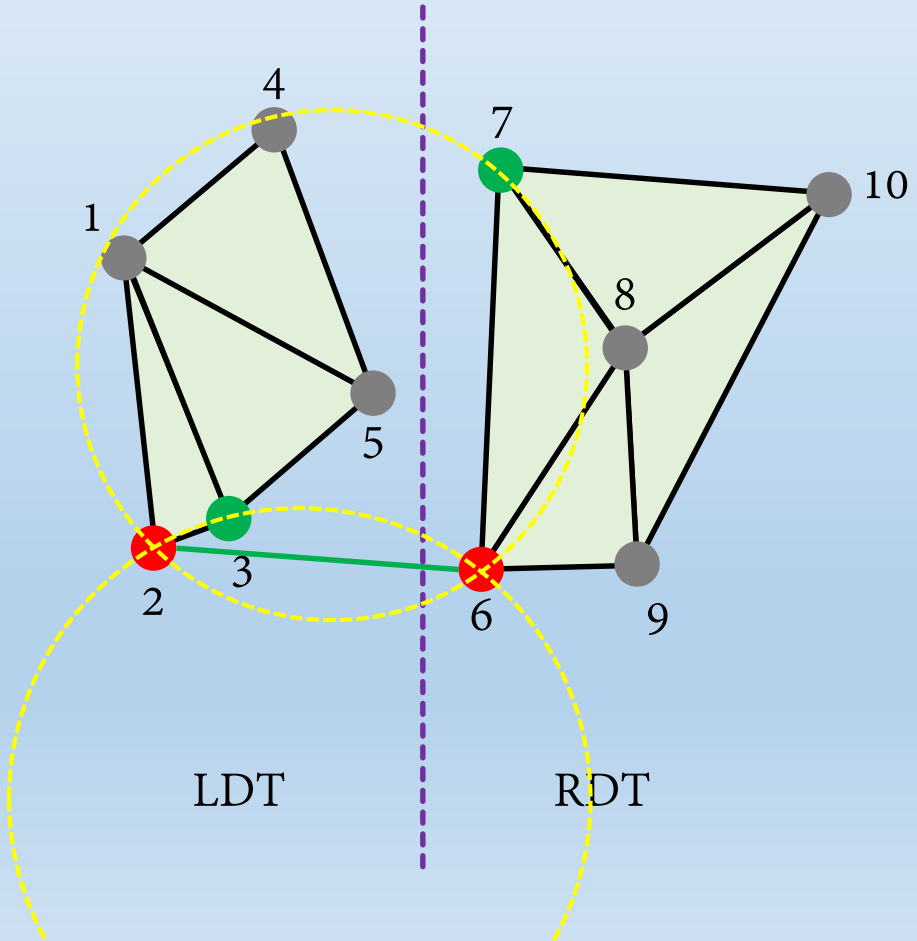
-Step4：不断合并相邻的DT，直至剩余1个DT结果



构造DT：4-分治法

- 划分点集，独立构造DT，再合并相邻DT形成更大的DT：

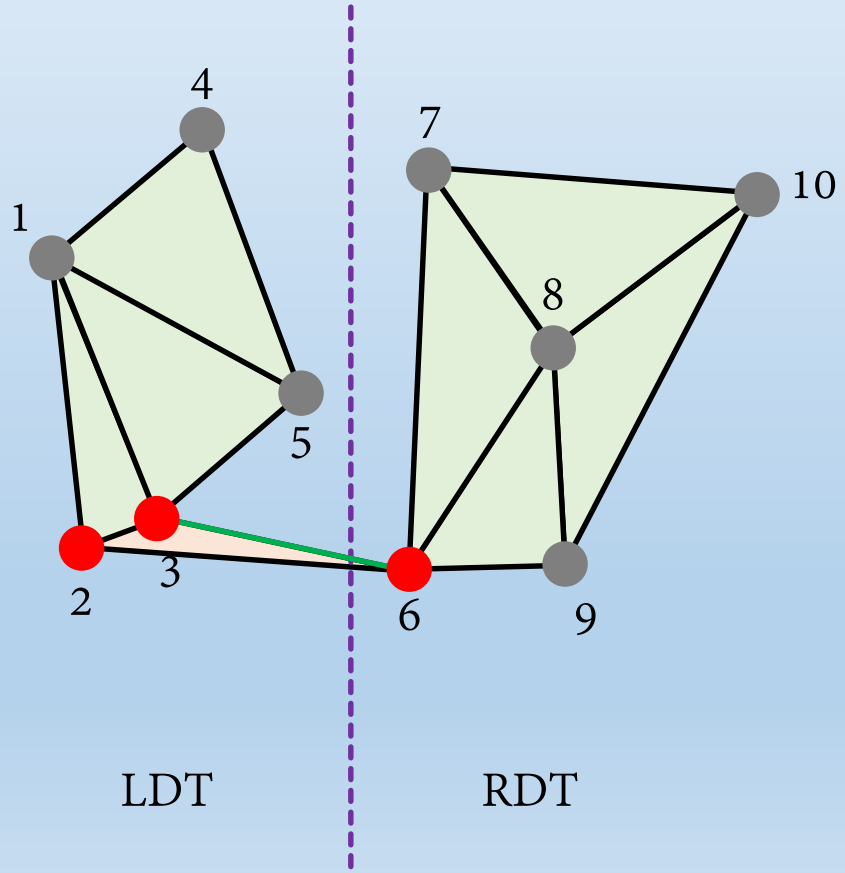
-Step4：不断合并相邻的DT，直至剩余1个DT结果



构造DT：4-分治法

- 划分点集，独立构造DT，再合并相邻DT形成更大的DT：

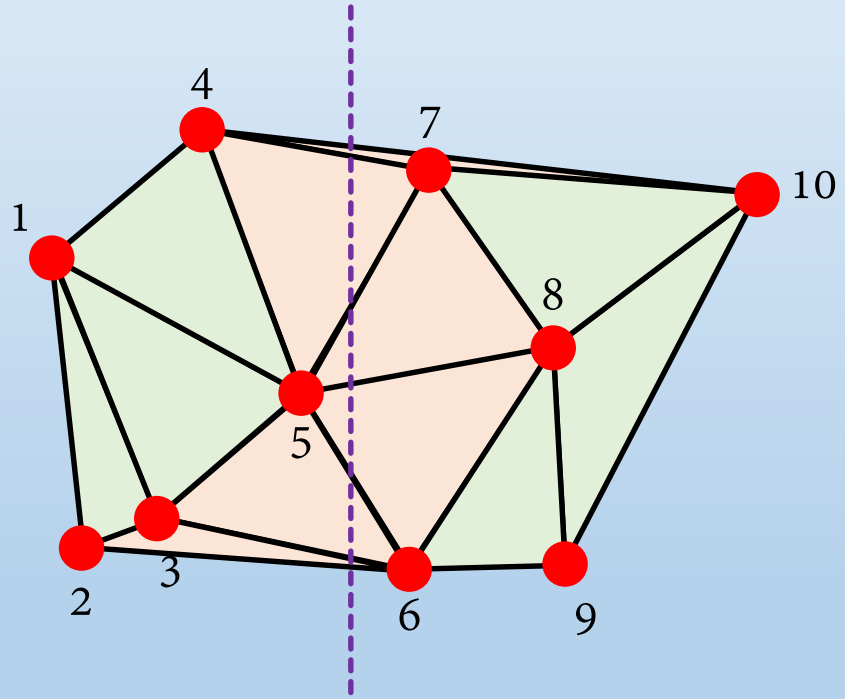
-Step4：不断合并相邻的DT，直至剩余1个DT结果



构造DT：4-分治法

- 划分点集，独立构造DT，再合并相邻DT形成更大的DT：

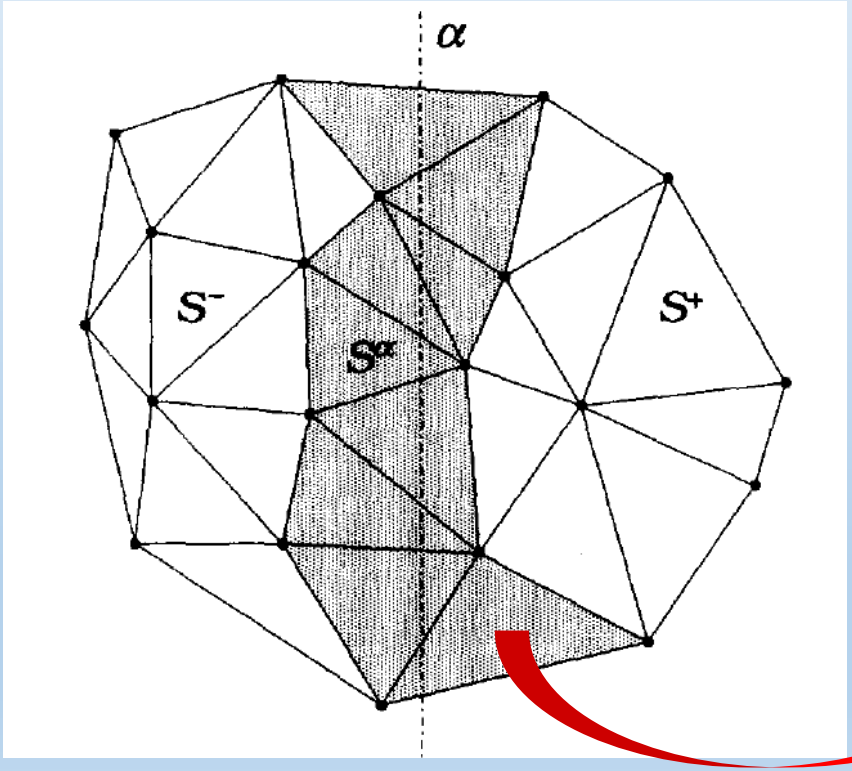
-Step4：不断合并相邻的DT，直至剩余1个DT结果



- 高维DT的合并较复杂：无显式的顶点邻接顺序

构造DT：5-DeWall算法

- 复杂的划分过程，无合并，适用于构造任意维度的DT
- 与一般分治法的过程相反

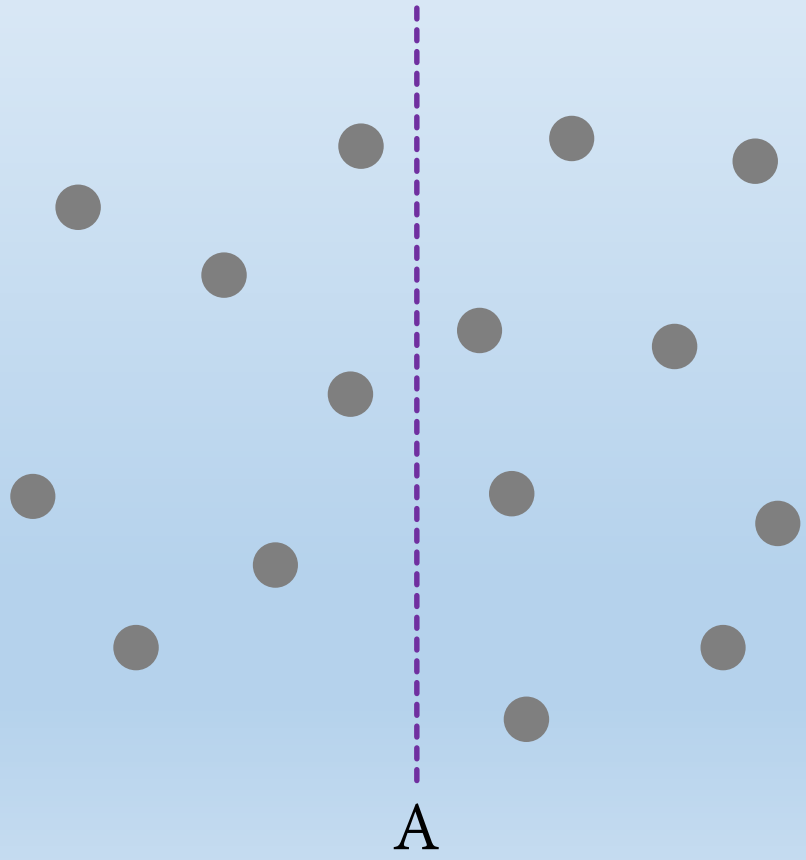


Delaunay Wall

P. Cignoni, C. Montani, R. Scopigno. DeWall: A fast divide and conquer Delaunay triangulation algorithm in E^d. Computer-Aided Design, 1998, 30(5): 333-341.

构造DT：5-DeWall算法

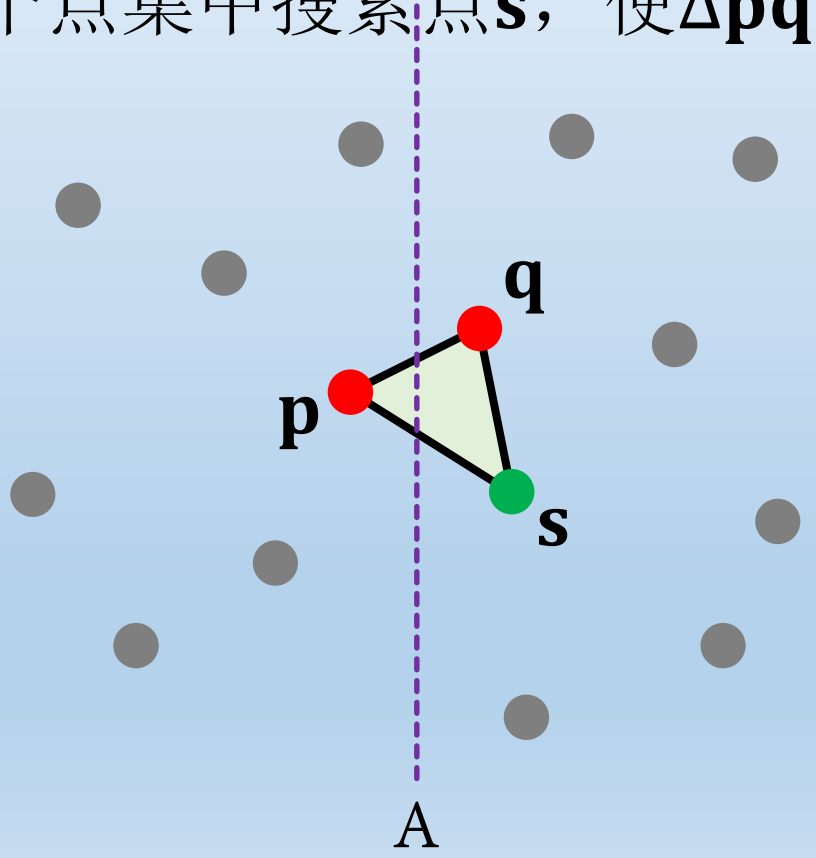
-Step1：确定分割平面，划分点集



构造DT：5-DeWall算法

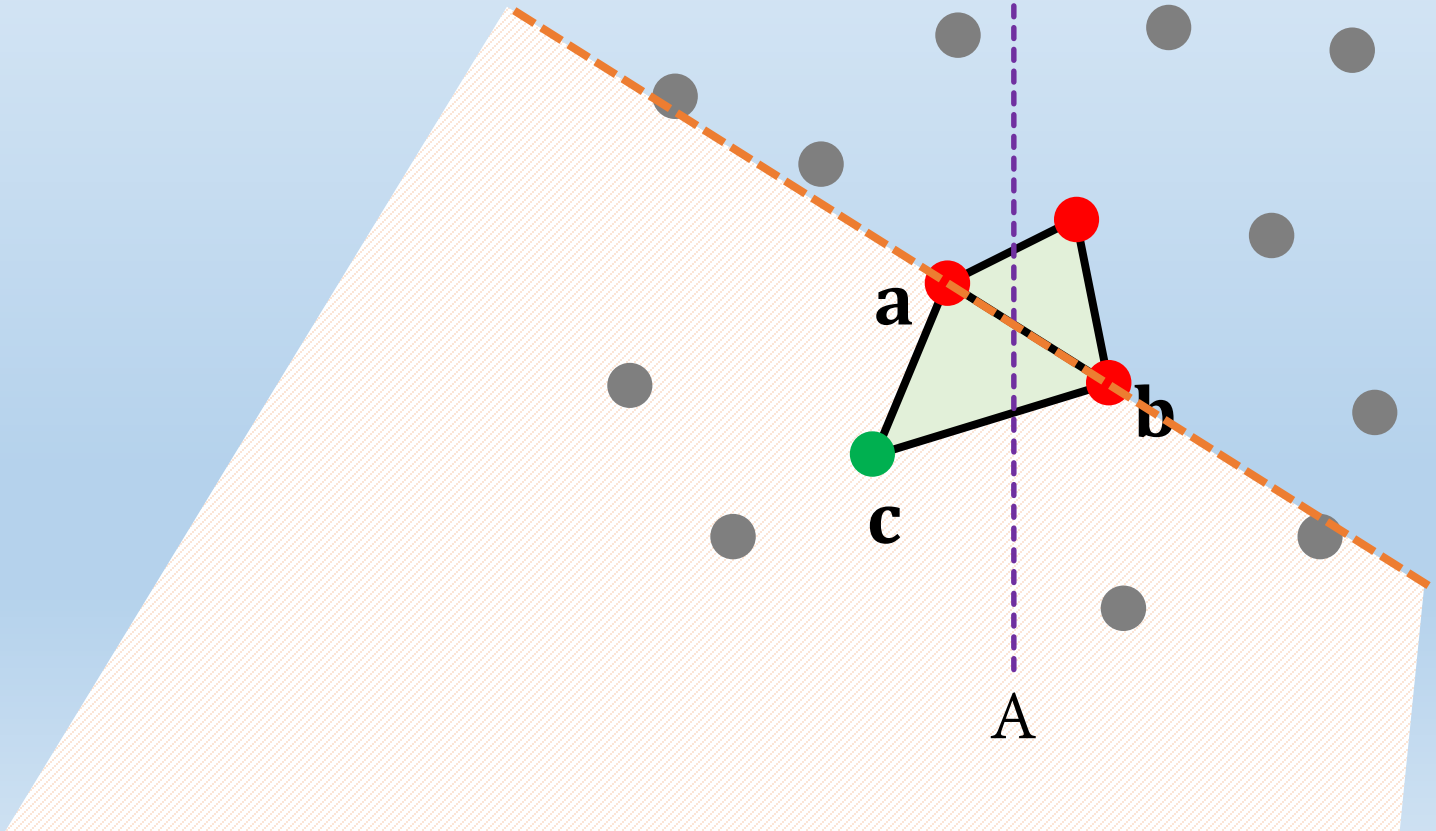
-Step2：构造第一个三角形并入DeWall：

从左子集中找到离分割平面最近的点**p**，再从右子集中找到离**p**最近的点**q**，再从整个点集中搜索点**s**，使**Δpqs**的外接圆半径最小



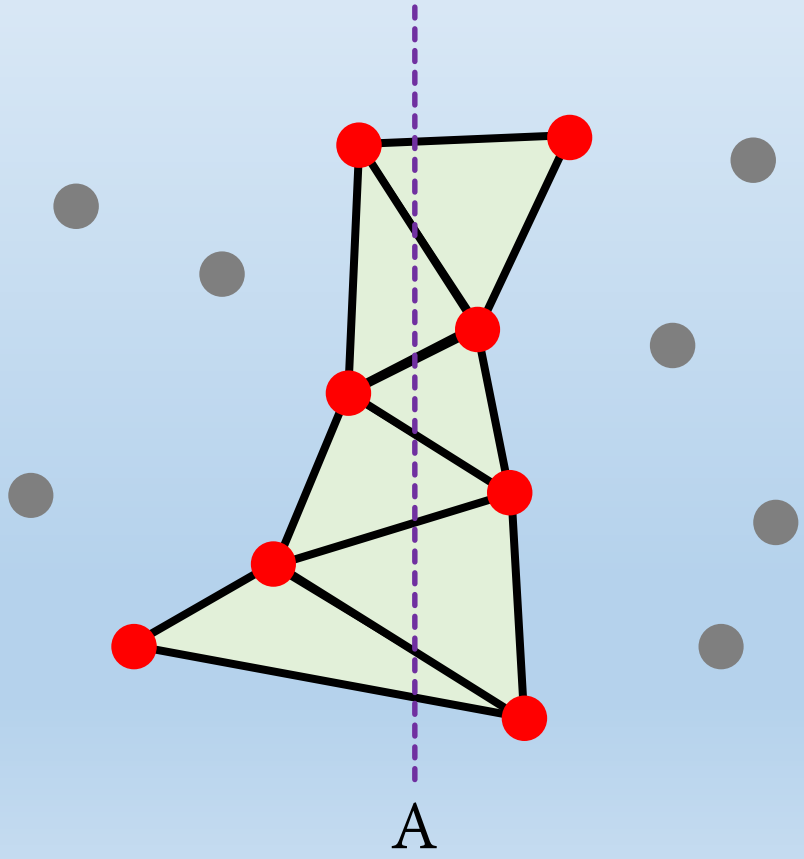
构造DT：5-DeWall算法

-Step3：维护一个列表AFL，初值为第一个三角形的三条边，每次从中取出一条边**ab**，若它与分割平面相交，则从**ab**外半空间中搜索点**c**，使**Δacb**的外接圆半径最小，构造新三角形**acb**并入DeWall，并将其边界边并入AFL



构造DT：5-DeWall算法

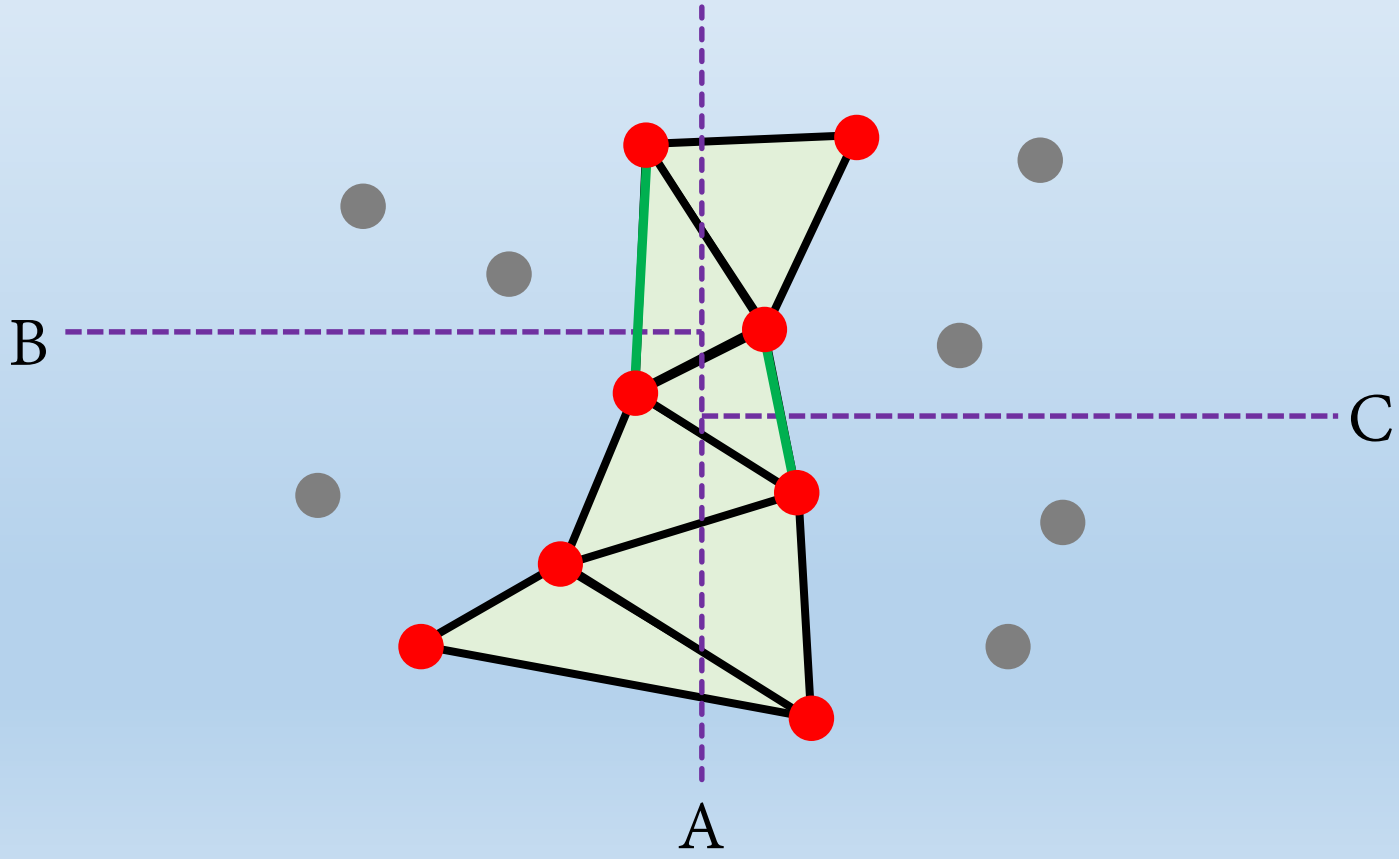
-Step4：迭代Step3直至AFL为空，可获得该分割平面对应的DeWall，并入结果DT



构造DT：5-DeWall算法

-Step5：对两个子集分别应用新分割平面并计算各自的DeWall

上一层DeWall结果的边界可用于AFL的初始化



构造DT：其他算法

- 扫描线算法 sweepline algorithm
- 卷包裹算法 gift wrapping algorithm

P. Su, R. L. S. Drysdale. A comparison of sequential Delaunay triangulation algorithms. Computational Geometry, 1997, 7: 361-385.

构造DT：主流代码库

- CGAL　https://www.cgal.org/

  2D/3D DT, randomized incremental insertion using Bowyer-Watson

- TetGen　https://www.wias-berlin.de/software/tetgen/1.5/index.html

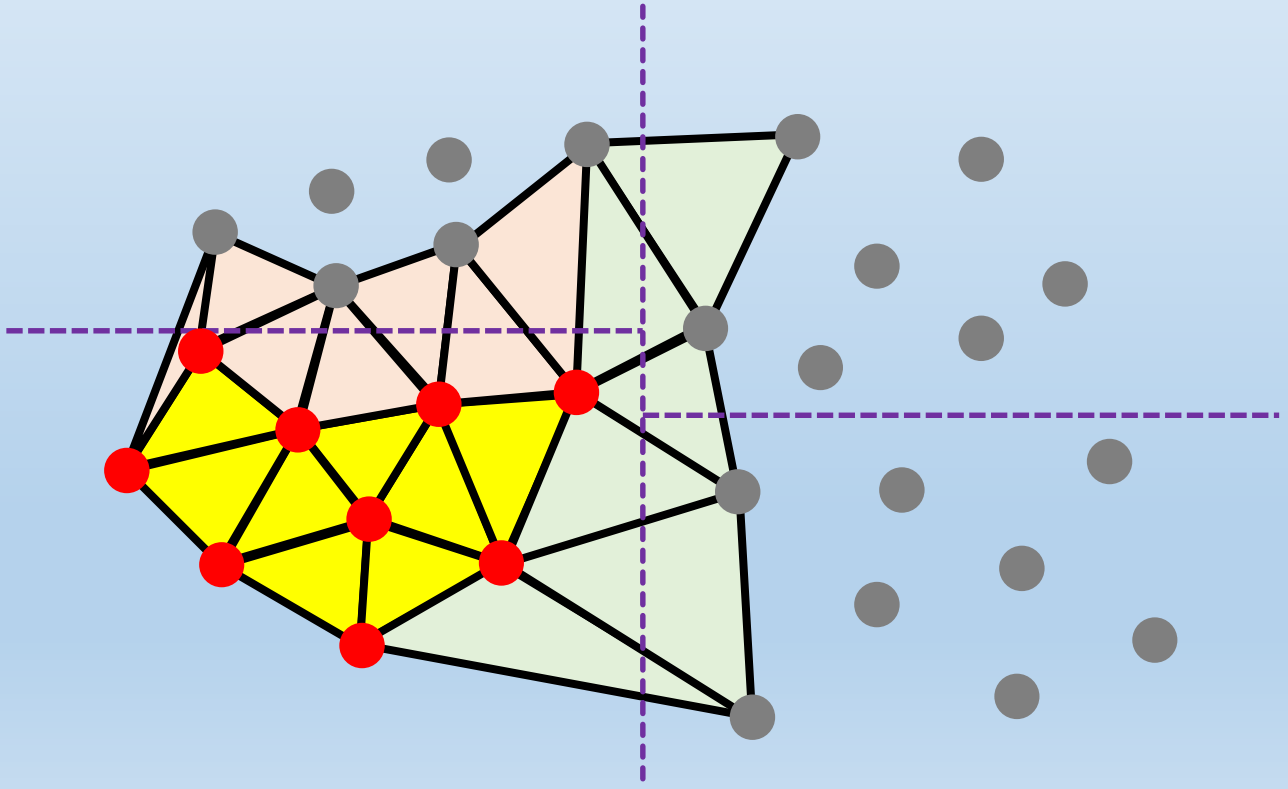
  3D DT, randomized incremental insertion using flipping

- Qhull　http://www.qhull.org/

  2D/3D DT, higher dimensional embedding

并行构造3D DT：1-并行DeWall算法

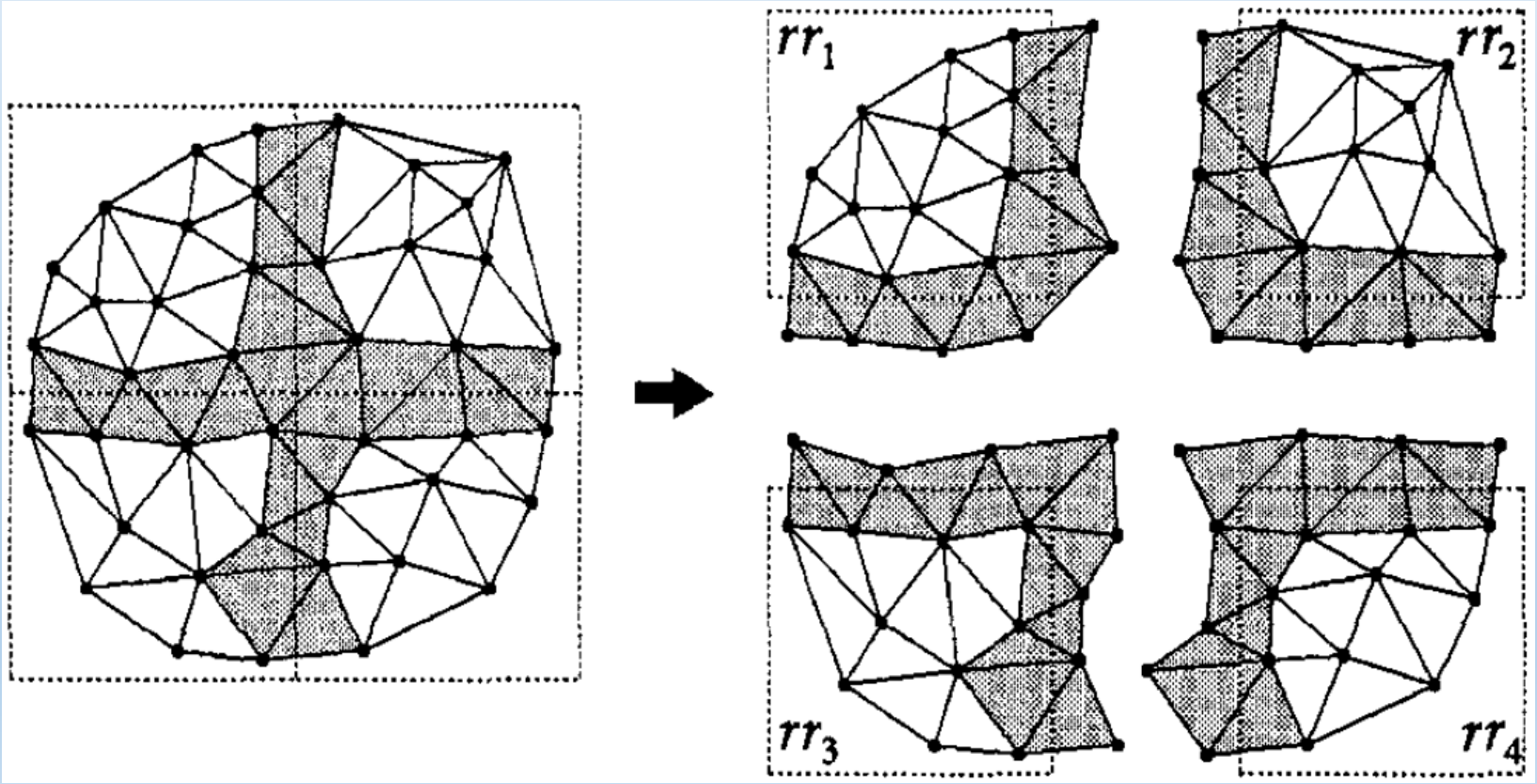
- 划分点集，记与某个子集相关的所有分割平面的集合P_k
- 每个线程需构造：P_k中每个平面的DeWall+子集内部的DT四面体
- 分割平面上的DeWall会被多个线程重复计算



P. Cignoni, C. Montani, R. Perego, R. Scopigno. Parallel 3D Delaunay Triangulation. Computer Graphics Forum, 1993, 12: 129-142.

并行构造3D DT：2-并行构造算法

- 划分点集，在每个格子里分别用扩散构造的方式获得DT四面体
- 格子边界的四面体会被重复计算



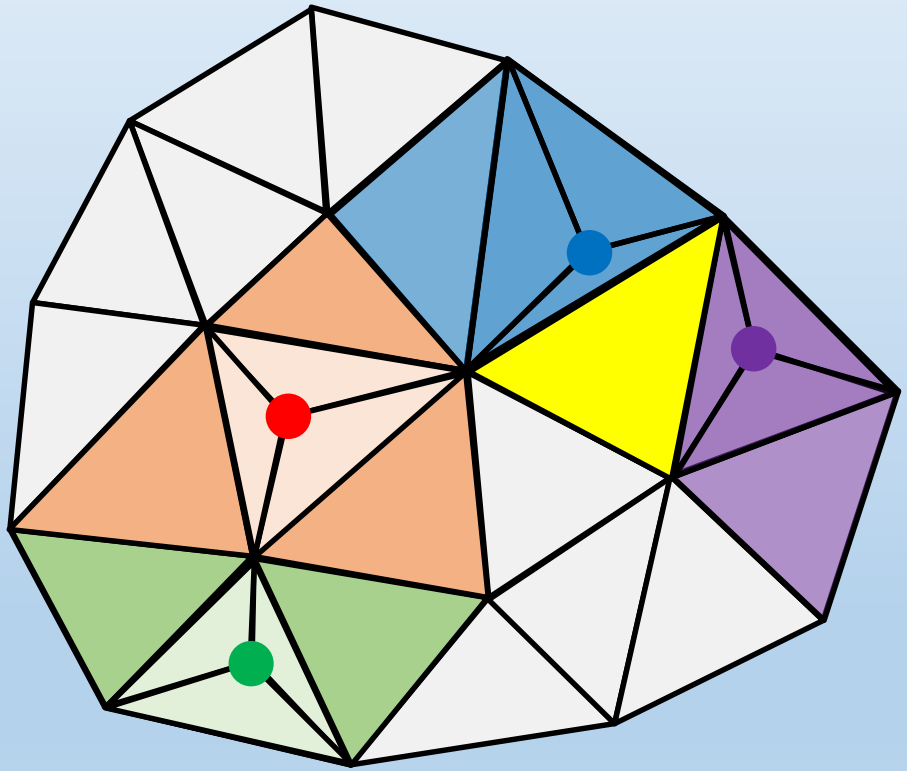
P. Cignoni, C. Montani, R. Perego, R. Scopigno. Parallel 3D Delaunay Triangulation. Computer Graphics Forum, 1993, 12: 129-142.

并行构造3D DT：3-并行增长算法

- 并行向多个四面体中同时插入新点，并通过局部更新保持DT
- 受影响四面体/顶点需要加锁、解锁以防冲突
- 局部更新策略：

-[Kohout2003]：翻面

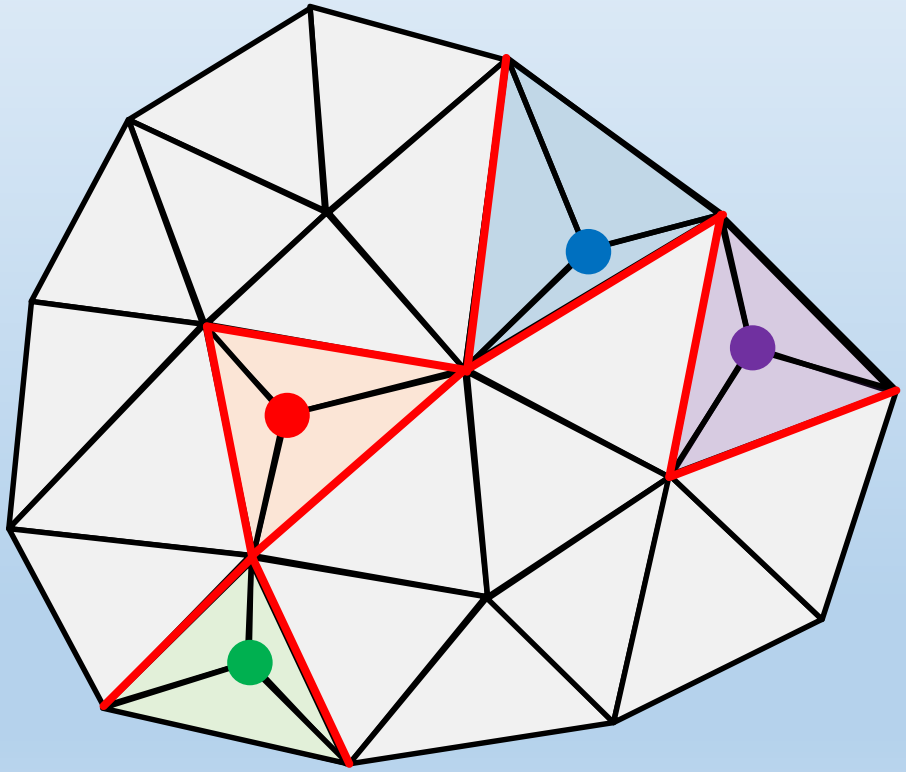
-[Foteinos2012]：Bowyer-Watson

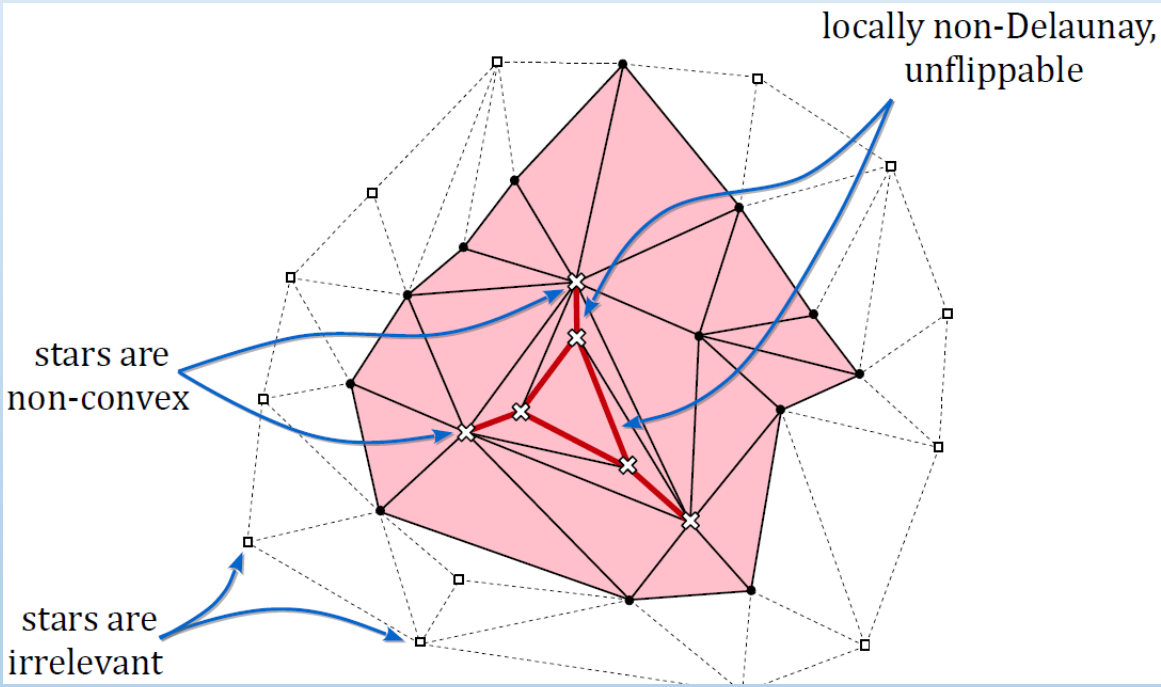


J. Kohout, I. Kolingerova. Parallel Delaunay triangulation in E^3: Make it simple. The Visual Computer, 2003, 19: 532-548.
P. Foteinos, N. Chrisochoides. Dynamic Parallel 3D Delaunay Triangulation. In Proceedings of the 20th International Meshing Roundtable, W. Quadros Ed., Berlin, 2012: 3-20.

并行构造3D DT：4-GPU增长+CPU矫正

- 并行插入新点，迭代标记需要翻转的面，并行翻转 (未加锁)
- GPU端结果存在非Delaunay的面 → CPU端用[Shewchuck2005]矫正



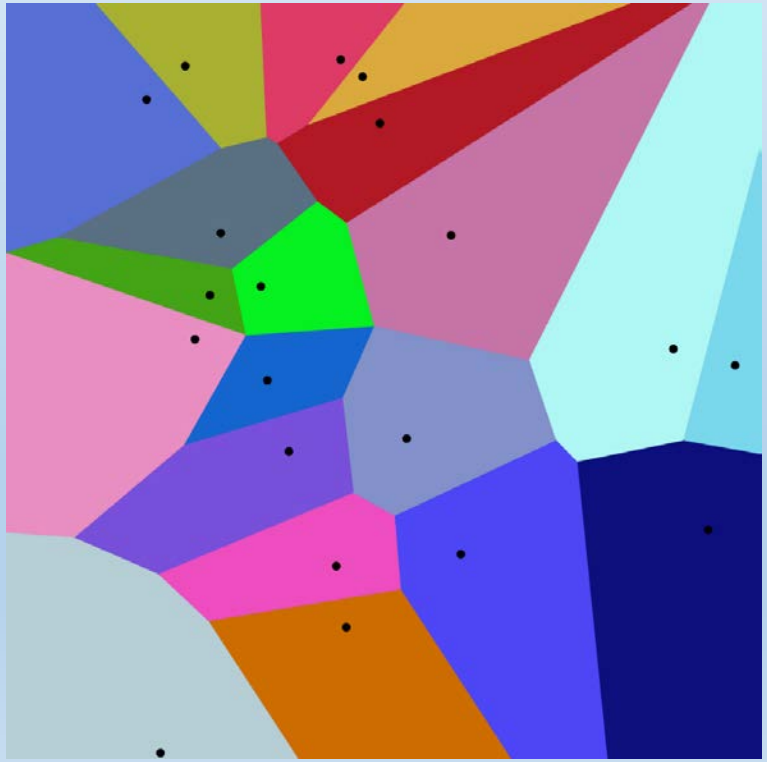


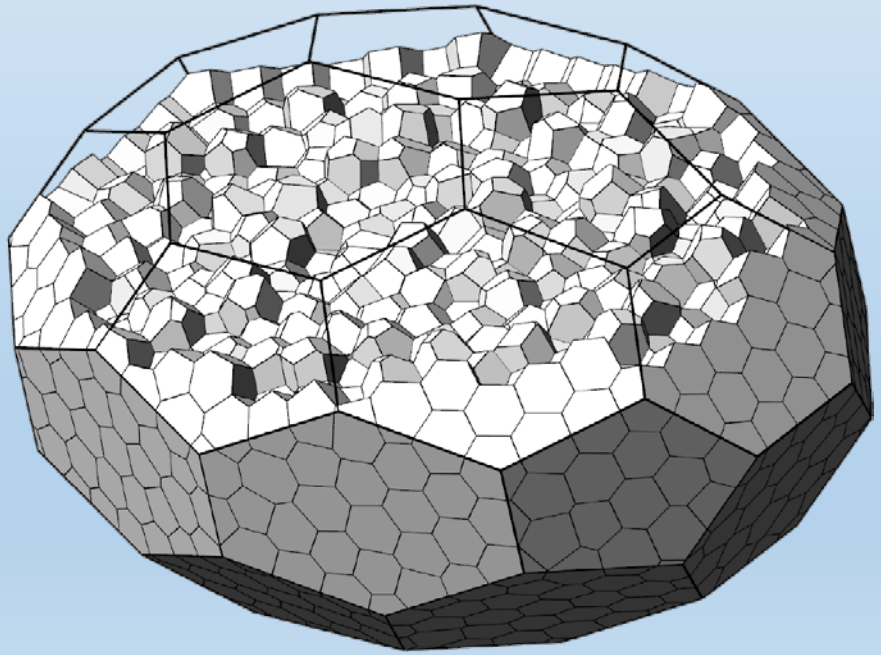
T. Cao, A. Nanjappa, M. Gao, T. Tan. A GPU accelerated algorithm for 3D Delaunay triangulation. In Proceedings of I3D, 2014: 47-54.
J. Shewchuk. Star Splaying: An Algorithm for Repairing Delaunay Triangulations and Convex Hulls. In Proc. 21st Symp. on Computational Geometry, 2005: 237–246.

目录

Voronoi Diagram (VD)

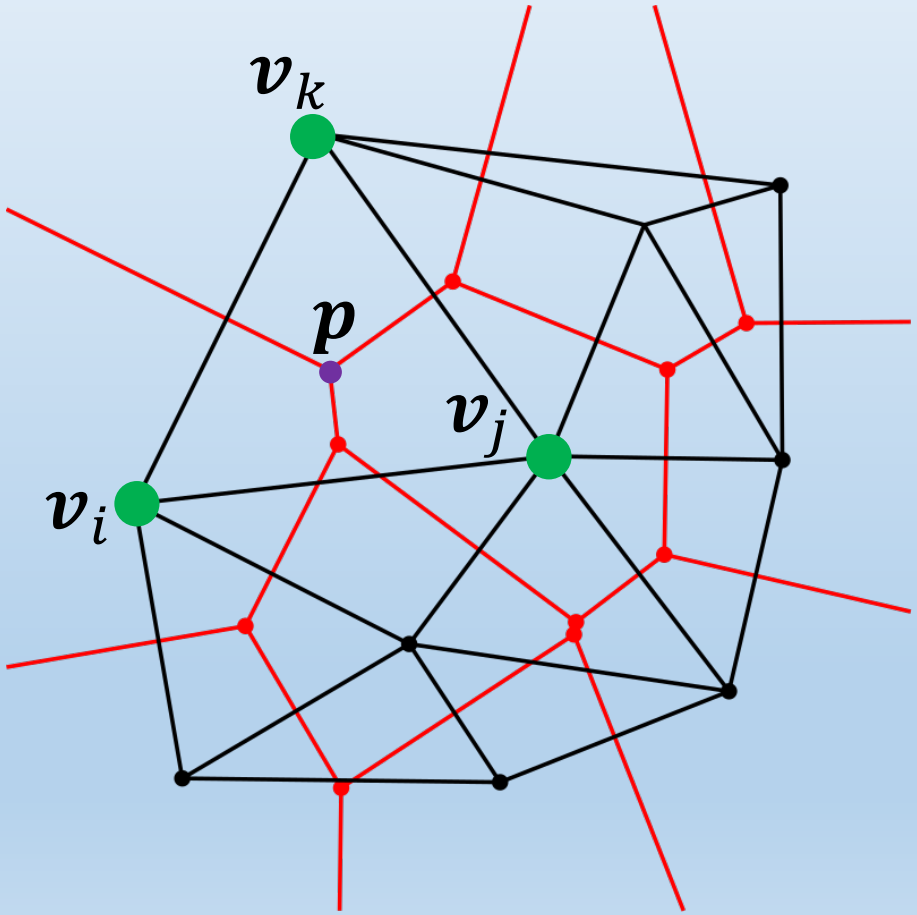
- 种子点集$\{\boldsymbol{v}_i \in R^d\}_{i=1}^n$，Voronoi图将空间划分成$n$个子区域
- $\boldsymbol{v}_i$的子区域 $\Omega_i = \left\{\boldsymbol{x} \in R^d \middle| \|\boldsymbol{x} - \boldsymbol{v}_i\| \leq \|\boldsymbol{x} - \boldsymbol{v}_j\|, \forall j \neq i\right\}$ (胞元)
- 高维VD





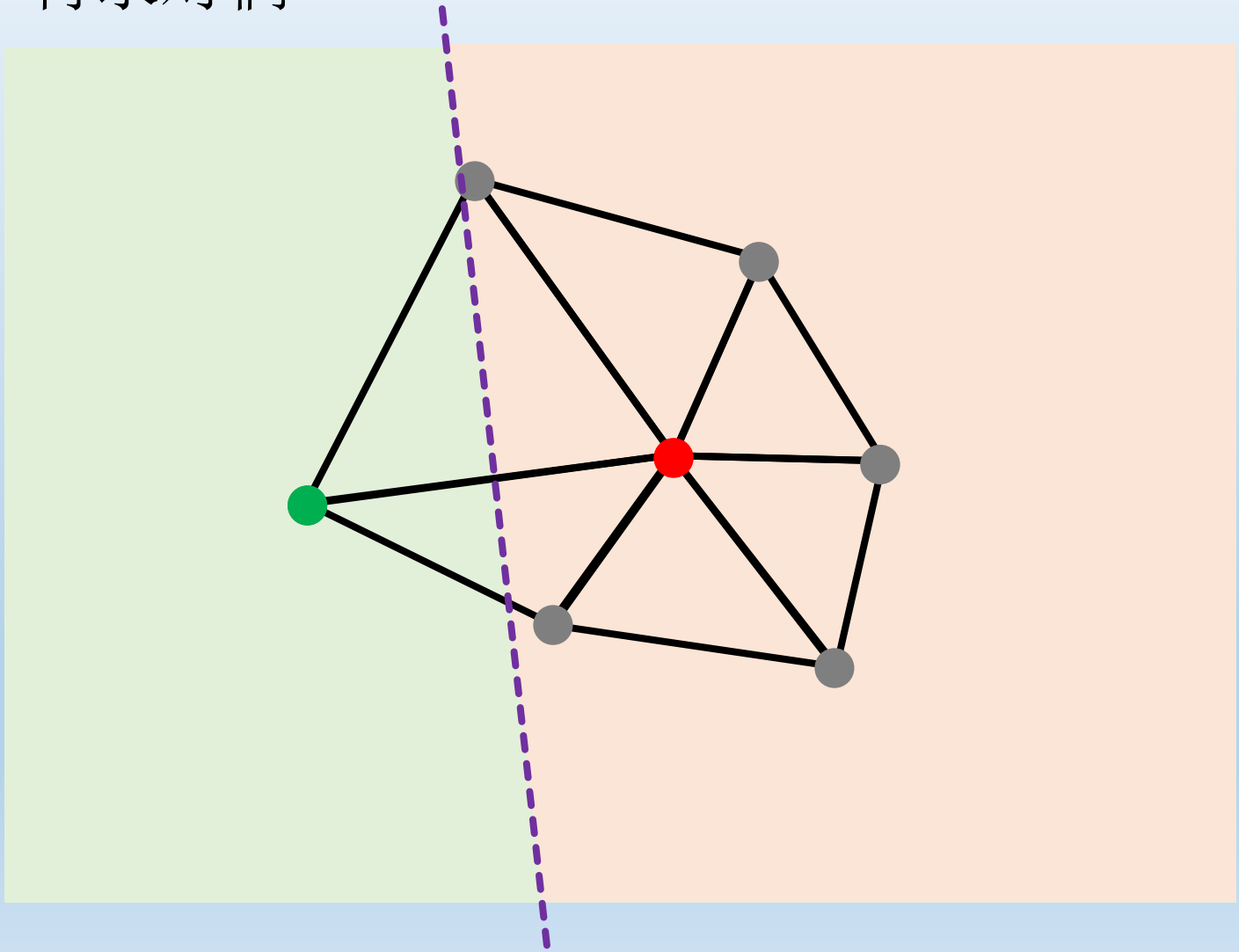
VD与DT的对偶关系

- 三个相邻Voronoi胞元对应的种子点构成一个DT三角形



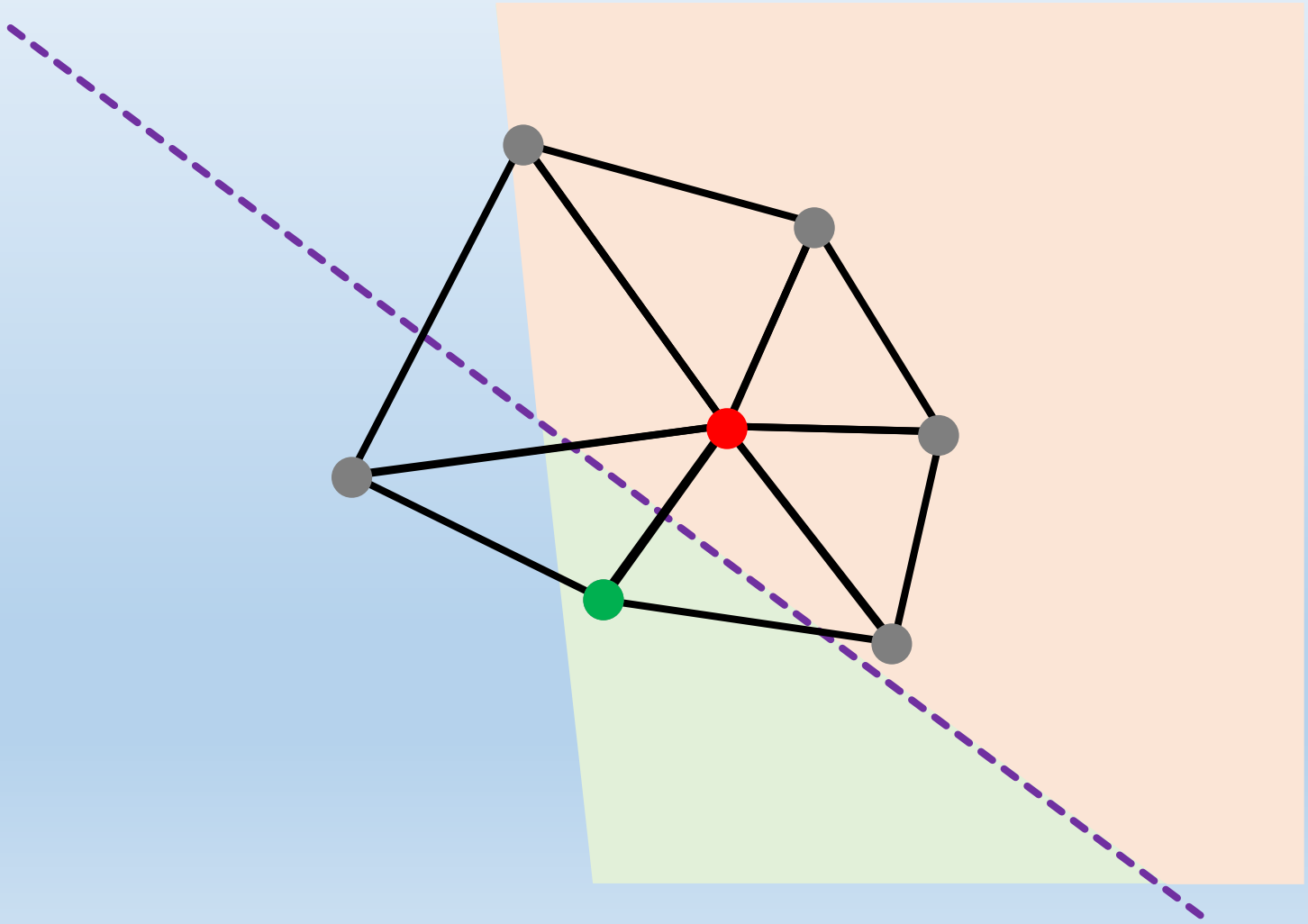
构造VD：1-基于DT的方法

- 先构造DT，再求对偶



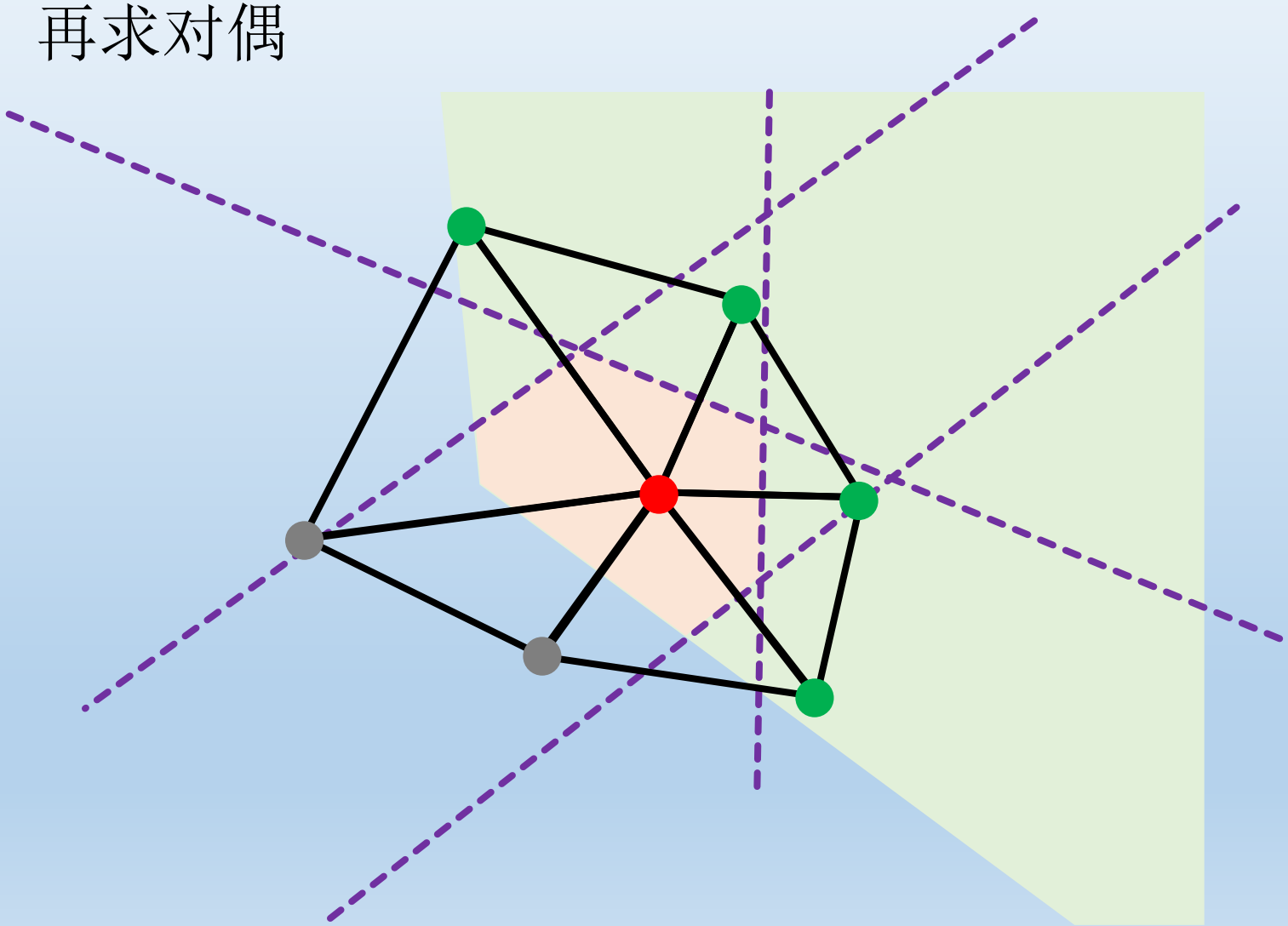
构造VD：1-基于DT的方法

- 先构造DT，再求对偶



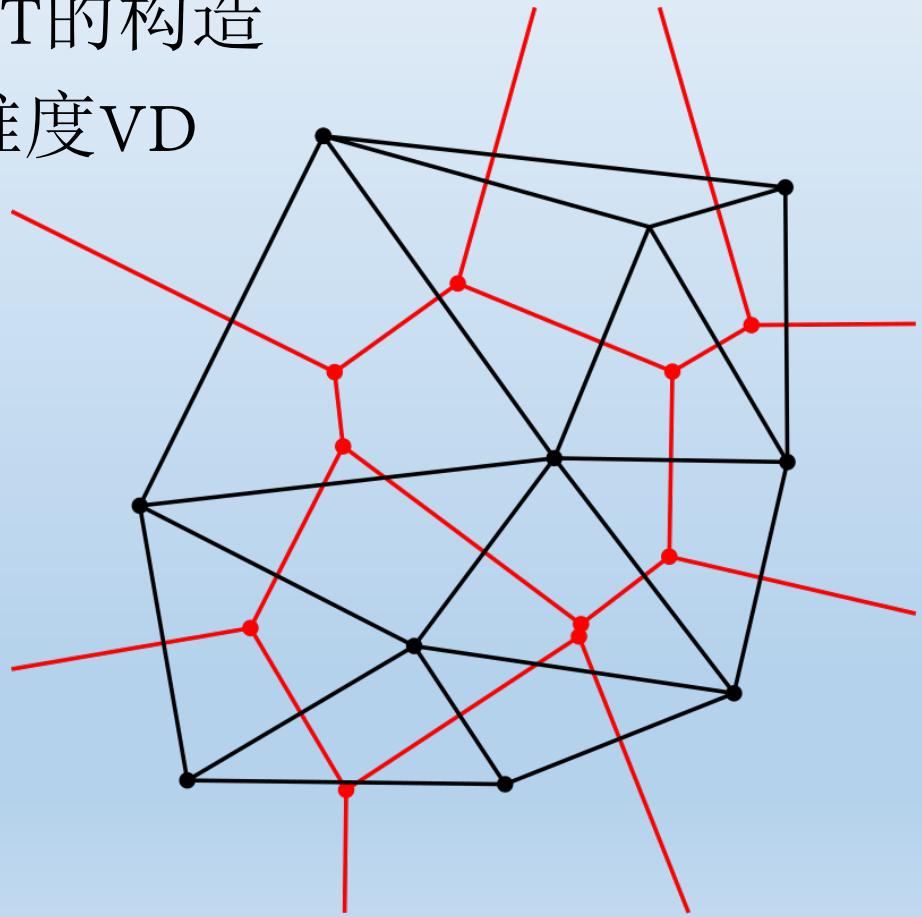
构造VD：1-基于DT的方法

- 先构造DT，再求对偶



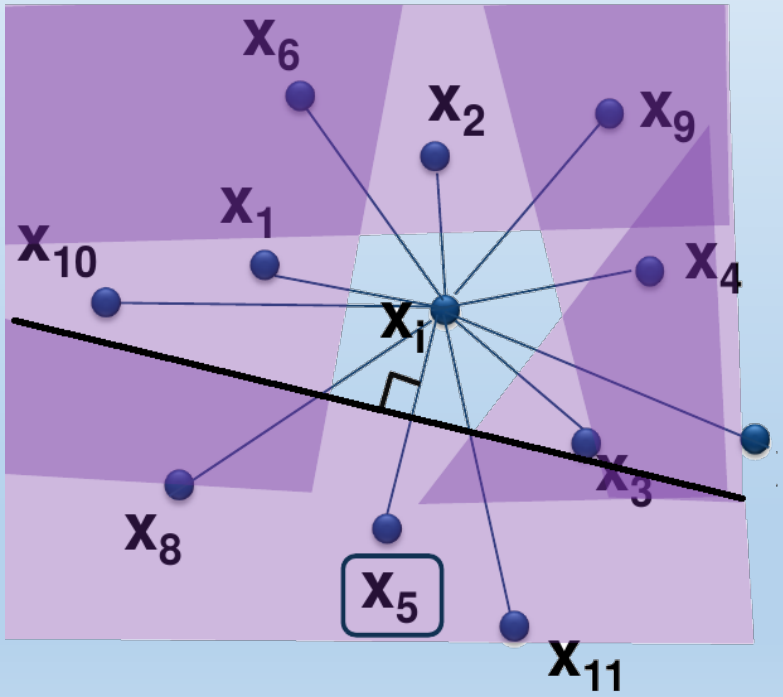
构造VD：1-基于DT的方法

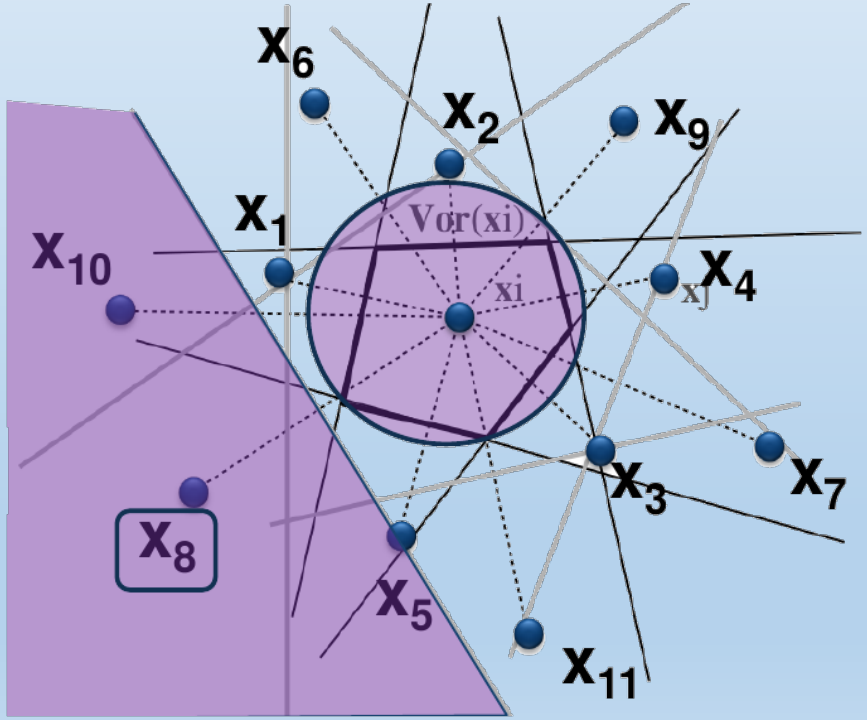
- 先构造DT，再求对偶
- 效率主要取决于DT的构造
- 适用于构造任意维度VD



构造VD：2-基于kNN的方法

- [Levy2013]：每个Voronoi胞元是其种子点与所有其他种子点之间半空间的交集
- 计算Ω_i：对其他种子点排序N_i，由近及远裁剪，至满足结束条件

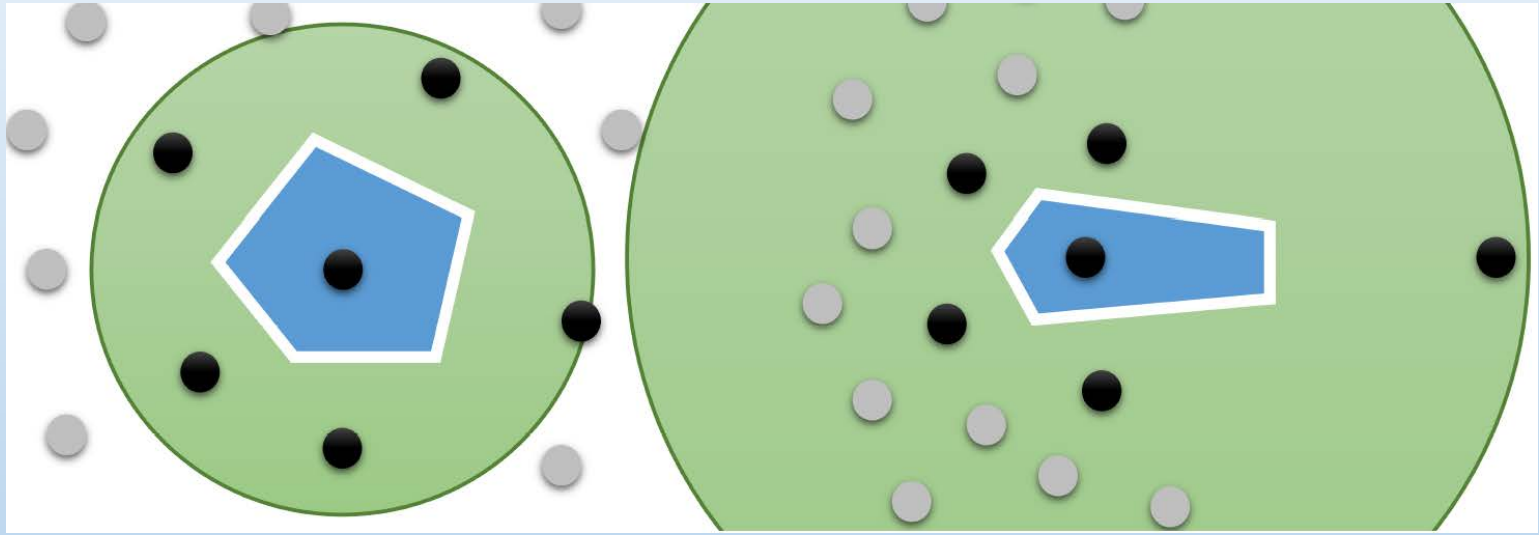


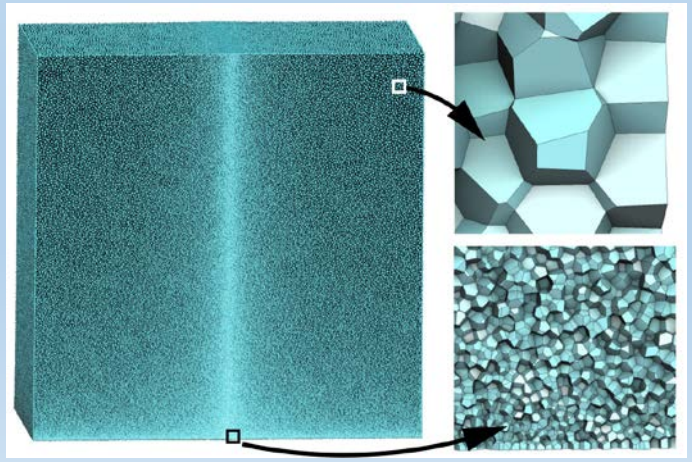


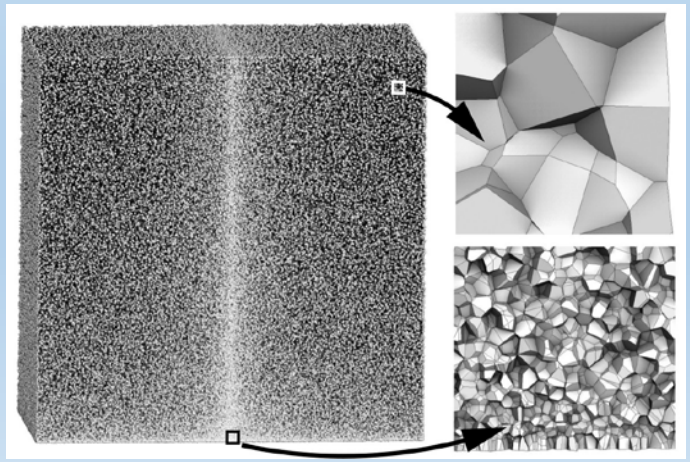
B. Lévy, N. Bonneel. Variational Anisotropic Surface Meshing with Voronoi Parallel Linear Enumeration. In: Jiao, X., Weill, JC. (eds) Proceedings of the 21st International Meshing Roundtable. Springer, Berlin, Heidelberg, 2013

构造VD：2-基于kNN的方法

- 缺点：效率受点分布影响 (胞元裁剪结束条件是否快速达到)







构造VD：其他算法

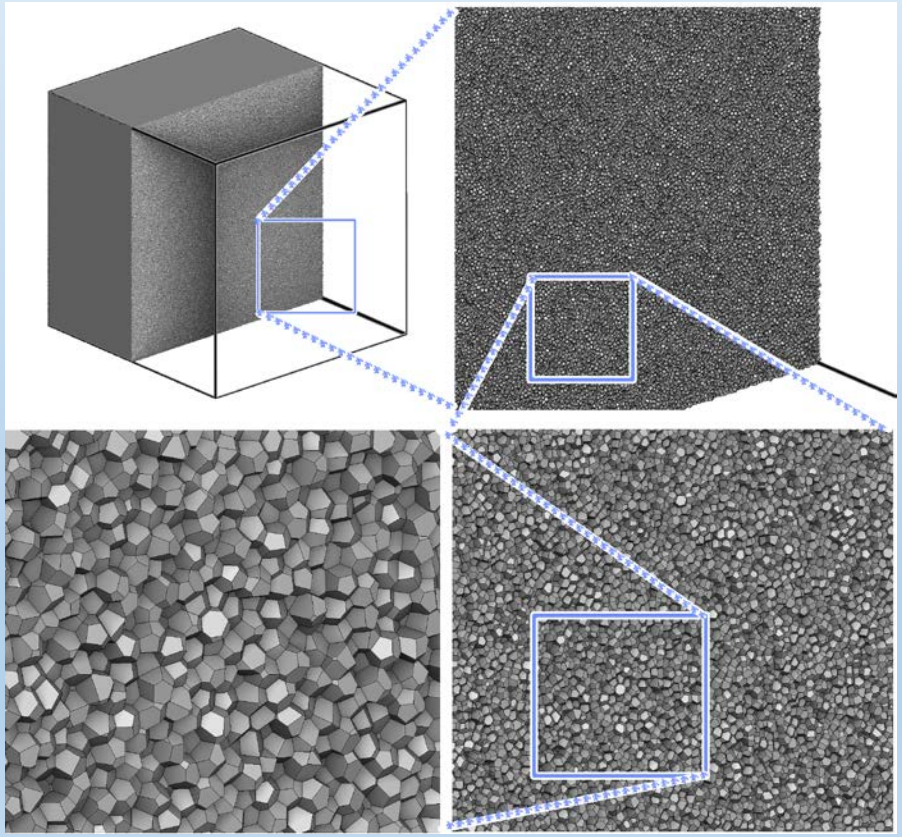
- 高维嵌入算法 [Brown1979]
- 扫描线算法 [Fortune1987]
- 分治法 [Shamos1975]

K. Brown. Voronoi diagrams from convex hulls. Information Processing Letters, 1979, 9(5): 223–228.
S. Fortune. A sweepline algorithm for Voronoi diagrams. Algorithmica, 1987, 2: 153-174.
M. Shamos, D. Hoey. Closest-point problems. In: 16th Annual symposium on foundations of computer science, 1975: 151–62.

并行构造VD：基于kNN的方法

-Step1：并行搜索每个种子点各自的N_i，$|N_i|$由用户给定

-Step2：并行裁剪每个胞元



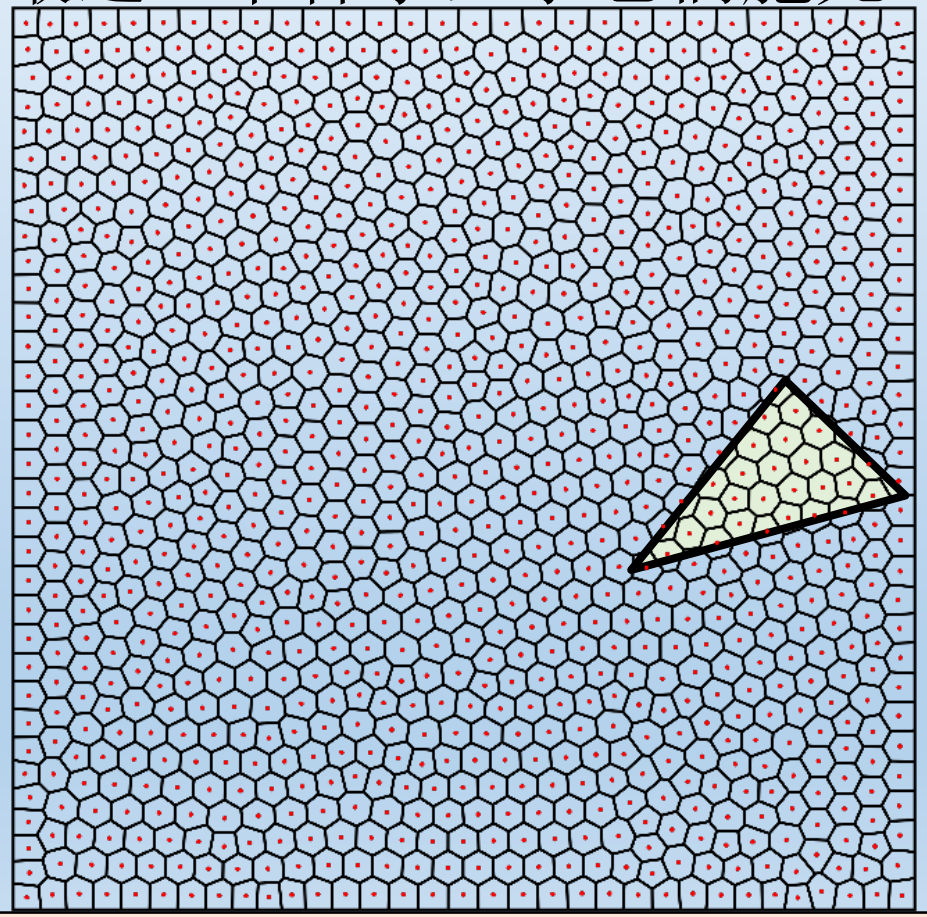
NVidia V100, 1千万点, 0.8s

N. Ray, D. Sokolov, S. Lefebvre, B. Levy. Meshless Voronoi on the GPU. ACM Transactions on Graphics, 2018, 37(6): 1-12.

并行构造VD：基于kNN的方法

- [Liu2022]：进一步求3D VD与不规则求解域(多个四面体)的交集

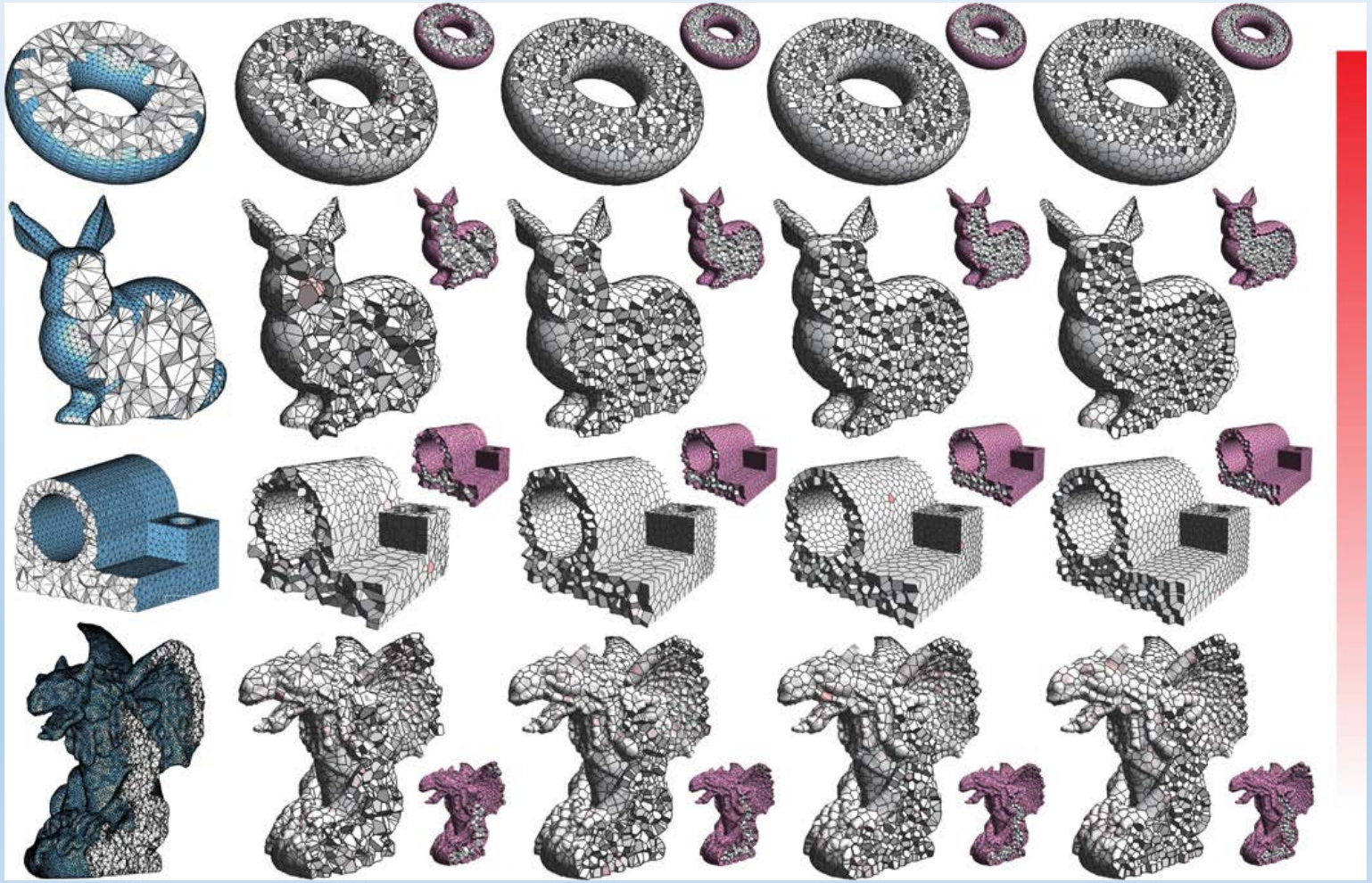
-离每个四面体重心最近20个种子，求它们胞元与该四面体的交集



X. Liu, L. Ma, J. Guo, D. Yan. Parallel Computation of 3D Clipped Voronoi Diagrams. TVCG, 2022, 28(2): 1363-1372.

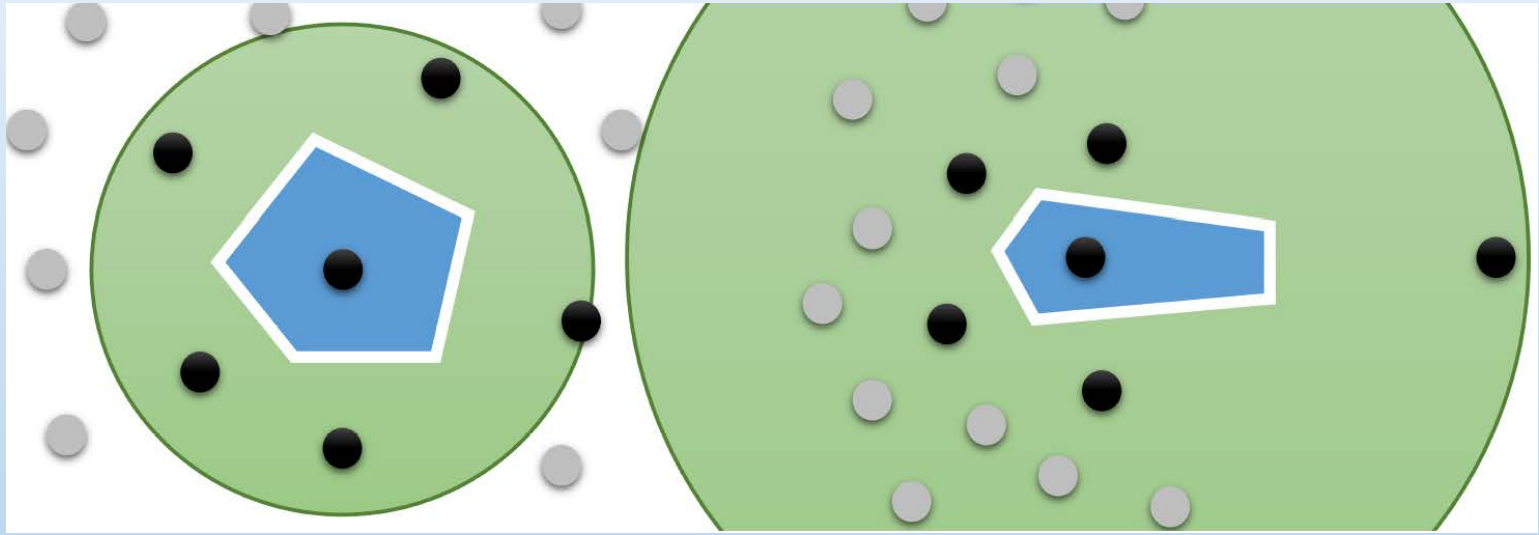
并行构造VD：基于kNN的方法

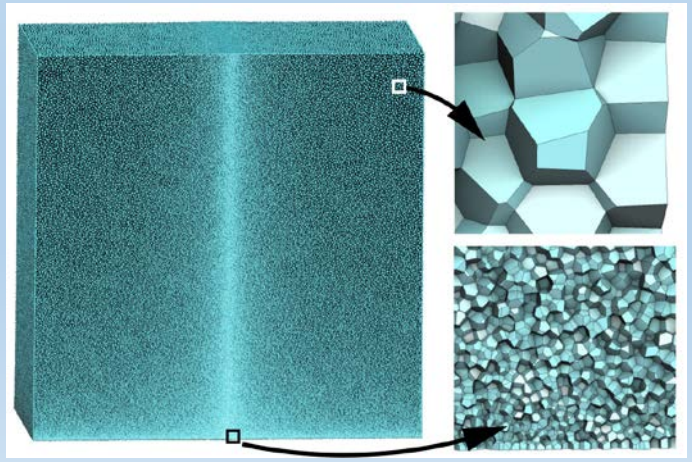
- [Liu2022]：进一步求3D VD与不规则求解域(多个四面体)的交集

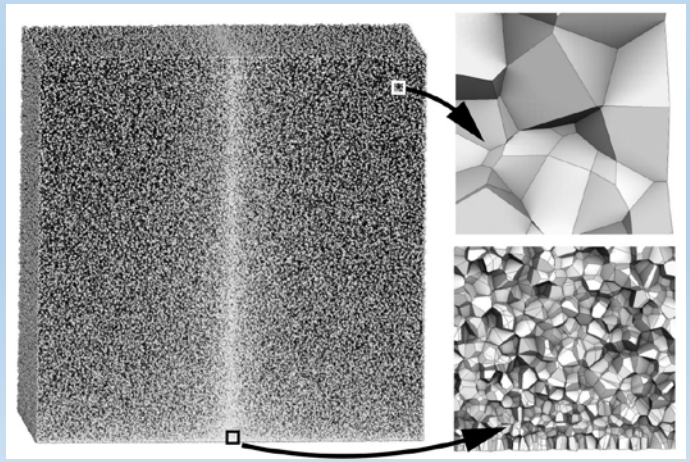


并行构造VD：基于kNN的方法

- 缺点1：效率受点分布影响 (胞元裁剪结束条件是否快速达到)





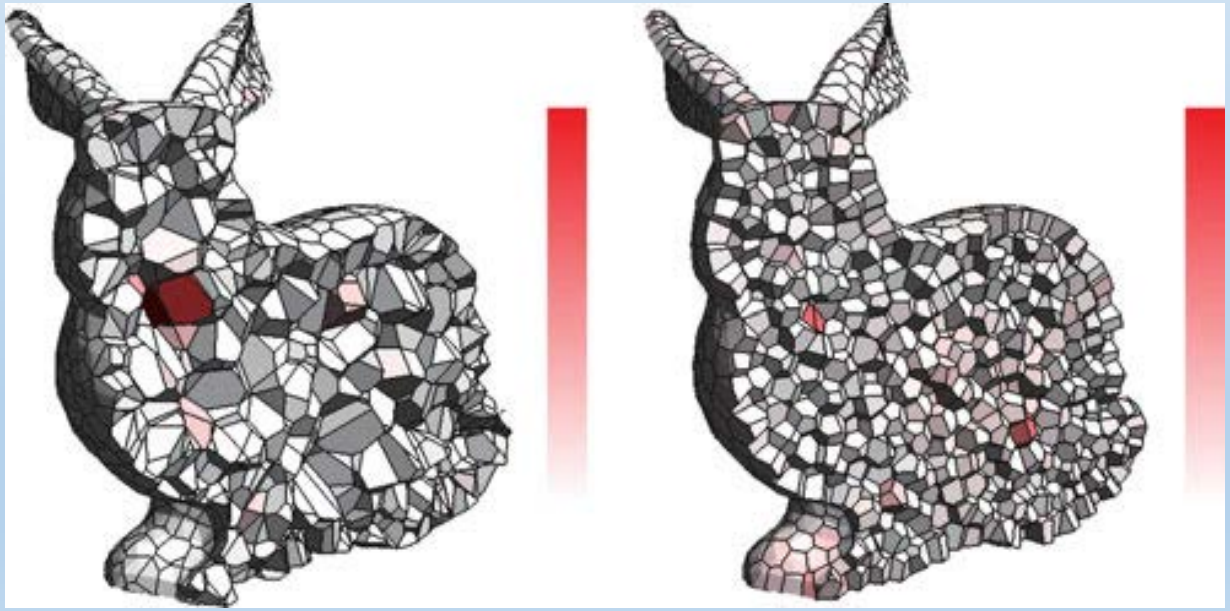


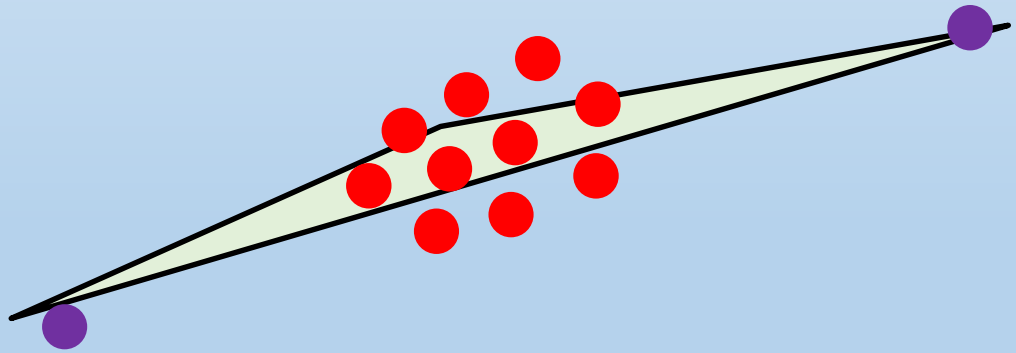
并行构造VD：基于kNN的方法

- 缺点2：部分胞元是近似解

-原因1：未达到裁剪结束条件 (除非$|N_i|$足够大)

-原因2：狭长四面体可能与更多胞元有交集却未被计算到



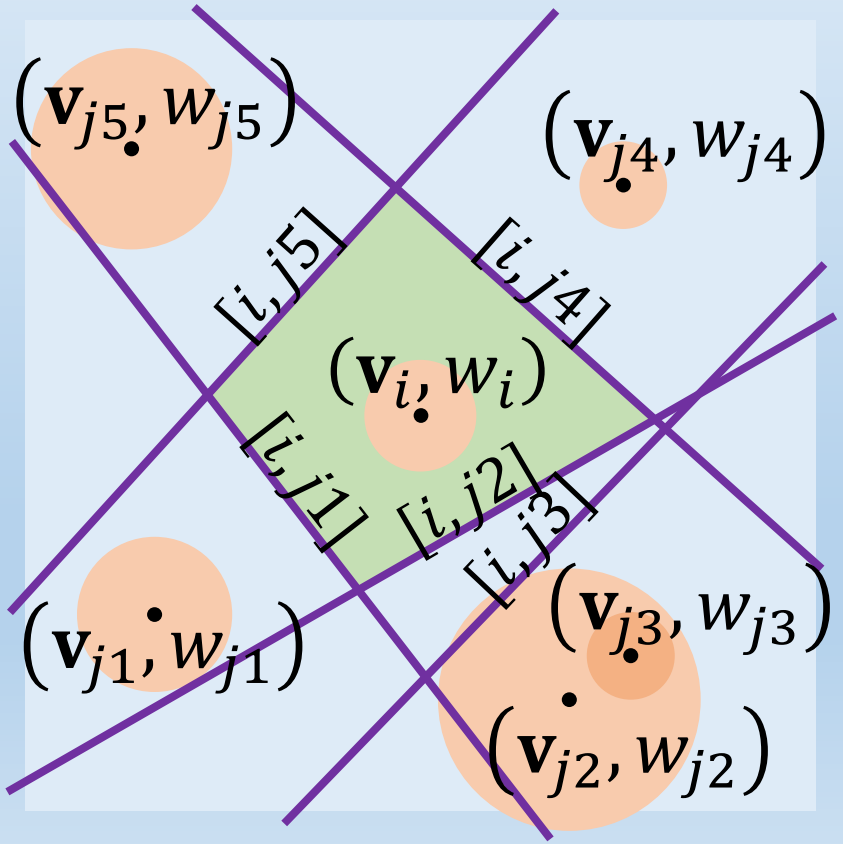


Power图的快速构造
基于kNN的方法

Yanyang Xiao, Juan Cao, Shaoping Xu, Zhonggui Chen. Meshless Power Diagrams. Computers & Graphics (Proc. SMI), 2023, 114: 247-256.

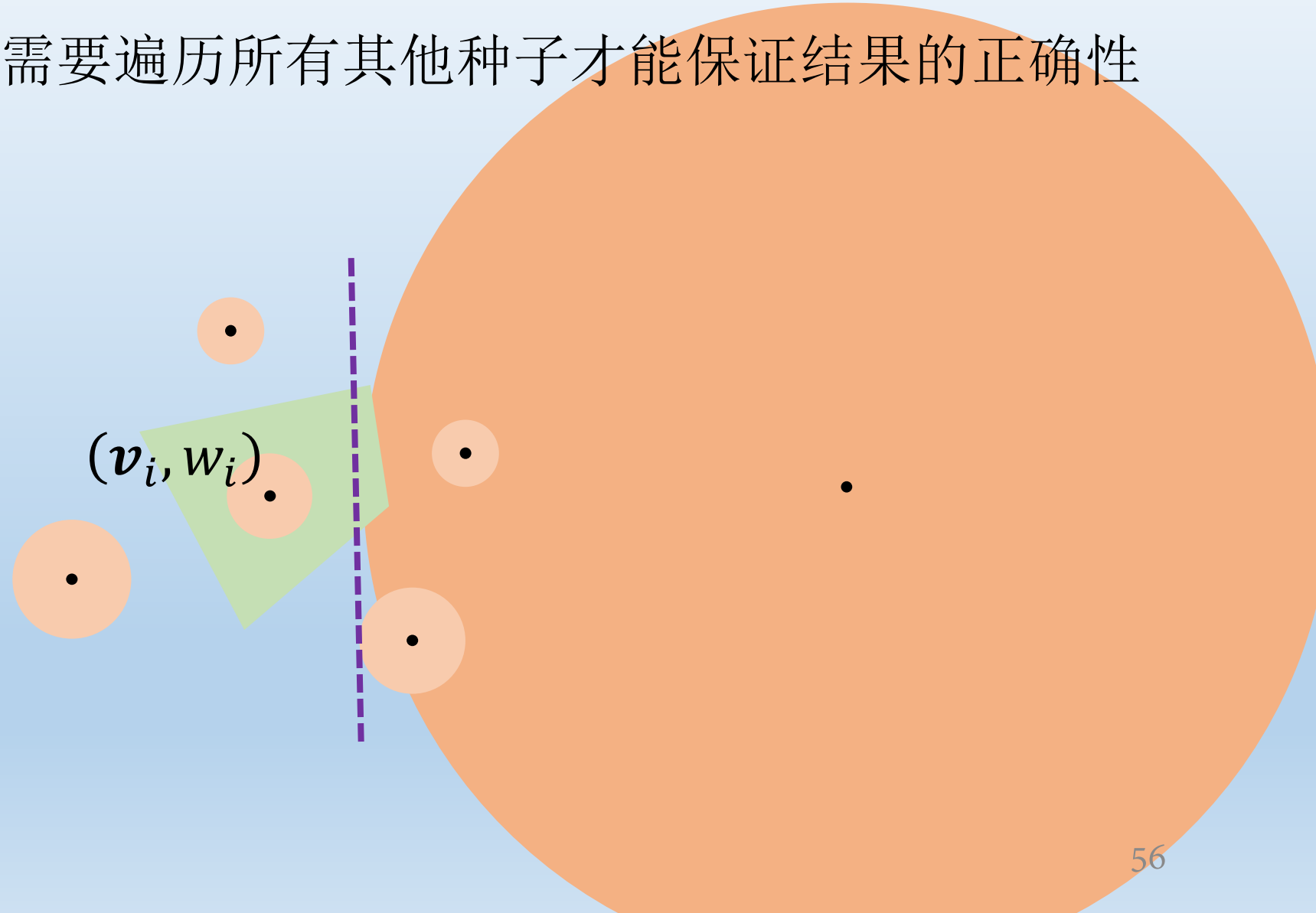
Power图

- 种子点 $X = \{\boldsymbol{v}_i\}_{i=1}^n$，权重 $W = \{w_i\}_{i=1}^n$
- Power胞元$\Omega_i = \left\{\boldsymbol{x} \in R^d \middle| \|\boldsymbol{x} - \boldsymbol{v}_i\|^2 - w_i \leq \left\|\boldsymbol{x} - \boldsymbol{v}_j\right\|^2 - w_j, \forall j \neq i\right\}$



基于kNN的方法不能直接应用于Power图的构造

- 每个胞元的构造都需要遍历所有其他种子才能保证结果的正确性



Power图 = 高一维受限Voronoi图

- Power胞元

$$\Omega_i = \left\{ \boldsymbol{x} \in R^d \middle| \|\boldsymbol{x} - \boldsymbol{v}_i\|^2 - w_i \leq \left\|\boldsymbol{x} - \boldsymbol{v}_j\right\|^2 - w_j, \forall j \neq i \right\}$$

- 引入$\eta \geq w_{max} = \max\{w_1, \ldots, w_n\}$

$$(\Omega_i)_\eta = \left\{ \boldsymbol{x} \in R^d \middle| \|\boldsymbol{x} - \boldsymbol{v}_i\|^2 + \eta - w_i \leq \left\|\boldsymbol{x} - \boldsymbol{v}_j\right\|^2 + \eta - w_j, \forall j \neq i \right\}$$

$$(\Omega_i)_\eta = \Omega_i$$

Power图 = 高一维受限Voronoi图

- Power胞元

$$\Omega_i = \left\{ \boldsymbol{x} \in R^d \middle| \|\boldsymbol{x} - \boldsymbol{v}_i\|^2 + \underbrace{\eta - w_i}_{\geq 0} \leq \|\boldsymbol{x} - \boldsymbol{v}_j\|^2 + \underbrace{\eta - w_j}_{\geq 0}, \forall j \neq i \right\}$$

$$\Omega_i = \left\{ \boldsymbol{x} \in R^d \middle| \|\boldsymbol{x} - \boldsymbol{v}_i\|^2 + \left(0 - \pm\sqrt{\eta - w_i}\right)^2 \leq \|\boldsymbol{x} - \boldsymbol{v}_j\|^2 + \left(0 - \pm\sqrt{\eta - w_j}\right)^2, \forall j \neq i \right\}$$

- 令 $\boldsymbol{y} = (\boldsymbol{x}, 0) \in R^{d+1}, \boldsymbol{p} = \left(\boldsymbol{v}, \pm\sqrt{\eta - w}\right) \in R^{d+1}$ (提升点)

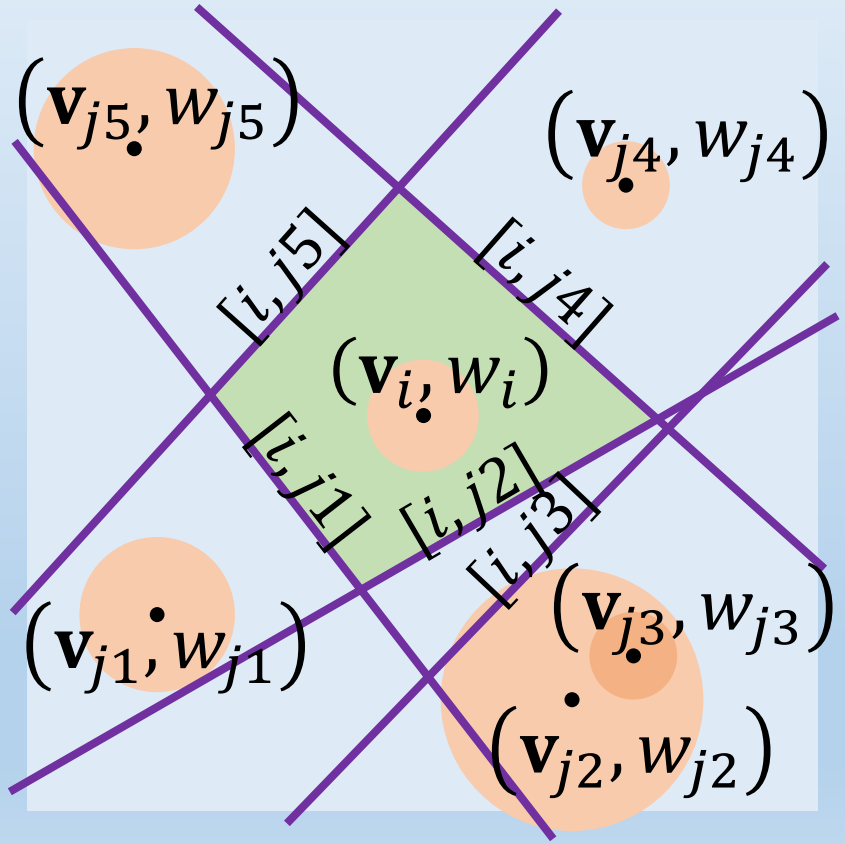
$$\Omega_i = \left\{ \boldsymbol{x} \in R^d \middle| \|\boldsymbol{y} - \boldsymbol{p}_i\|^2 \leq \|\boldsymbol{y} - \boldsymbol{p}_j\|^2, \forall j \neq i \right\}$$

= 提升点$\boldsymbol{p}_i$的Voronoi胞元约束在 R^d 的部分

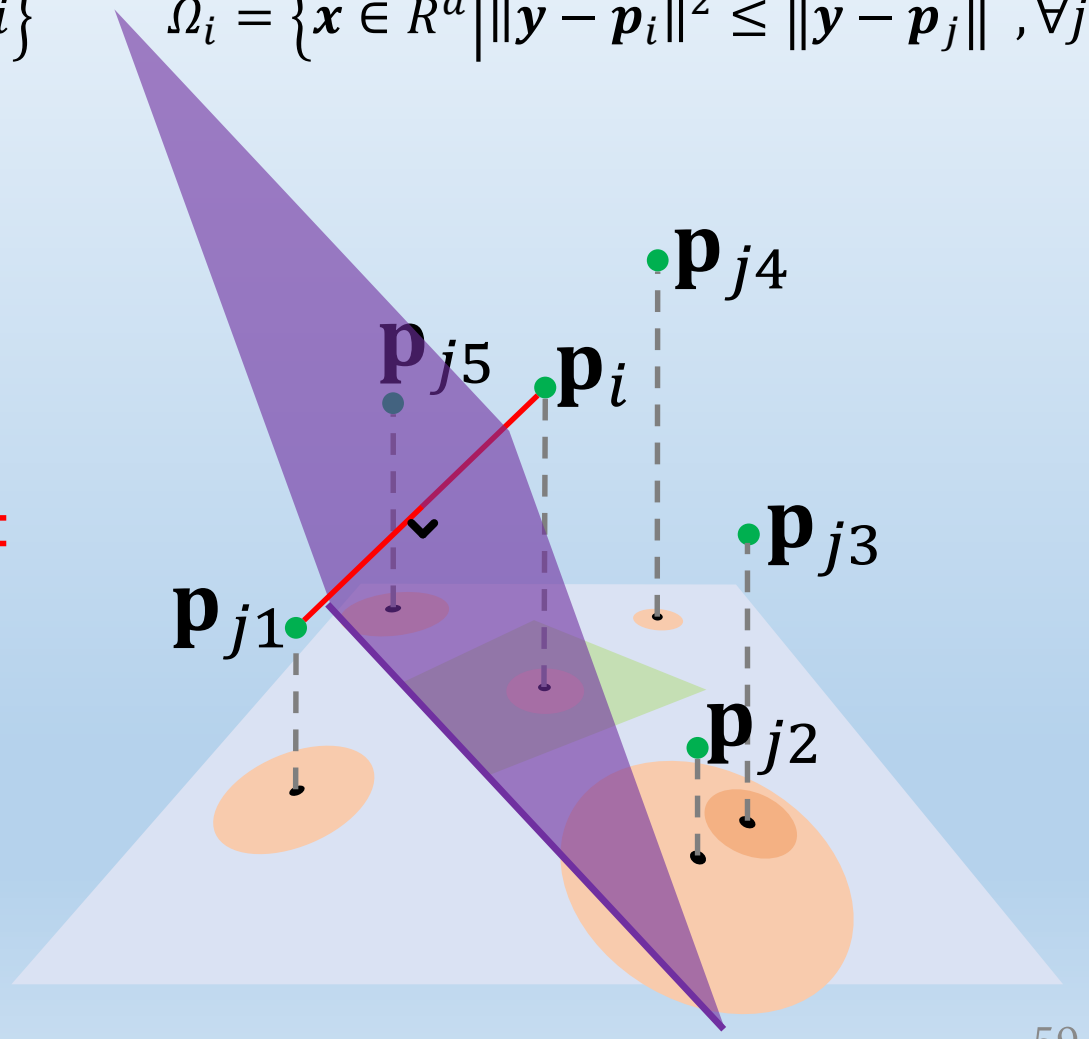
Power图 = 高一维受限Voronoi图

$$\Omega_i = \left\{ \boldsymbol{x} \in R^d \middle| \|\boldsymbol{x} - \boldsymbol{v}_i\|^2 - w_i \le \|\boldsymbol{x} - \boldsymbol{v}_j\|^2 - w_j, \forall j \ne i \right\}$$

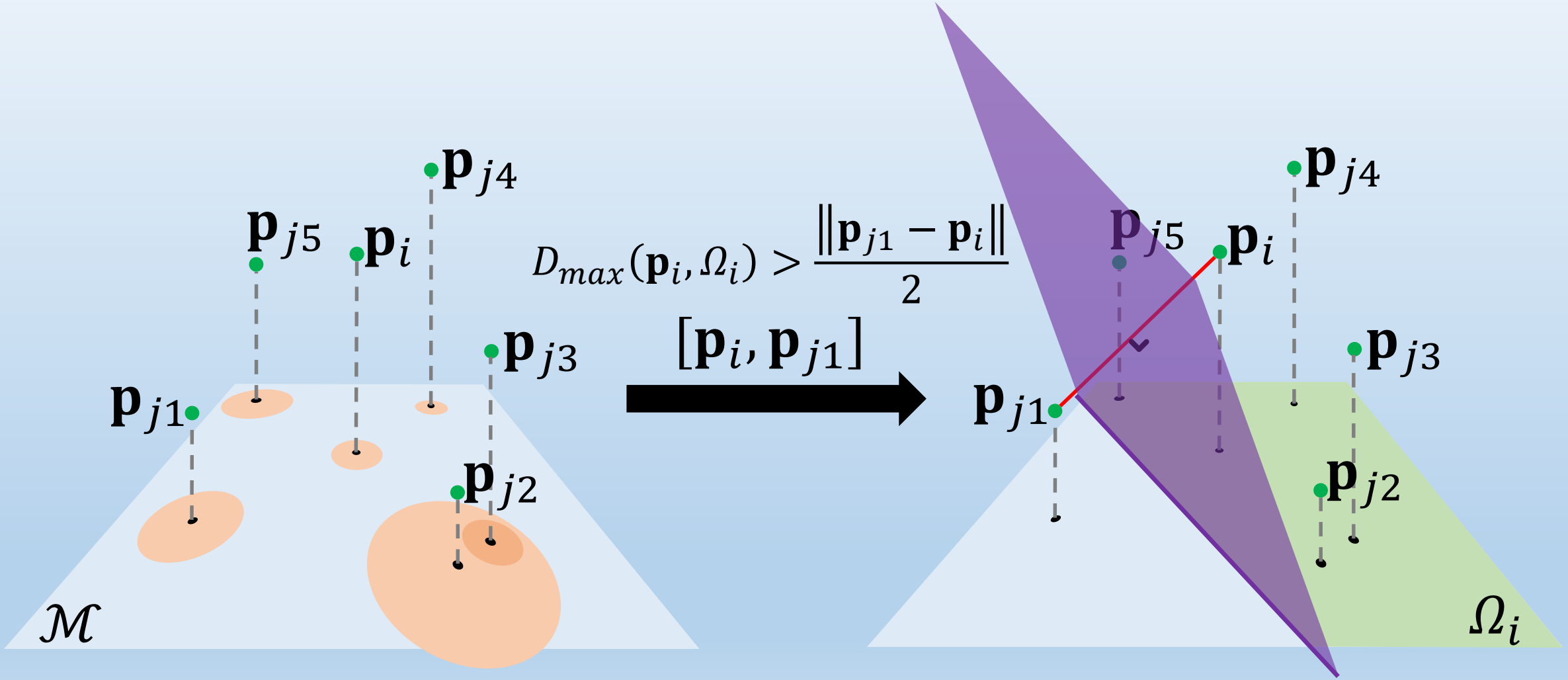
$$\Omega_i = \left\{ \boldsymbol{x} \in R^d \middle| \|\boldsymbol{y} - \boldsymbol{p}_i\|^2 \le \|\boldsymbol{y} - \boldsymbol{p}_j\|^2, \forall j \ne i \right\}$$



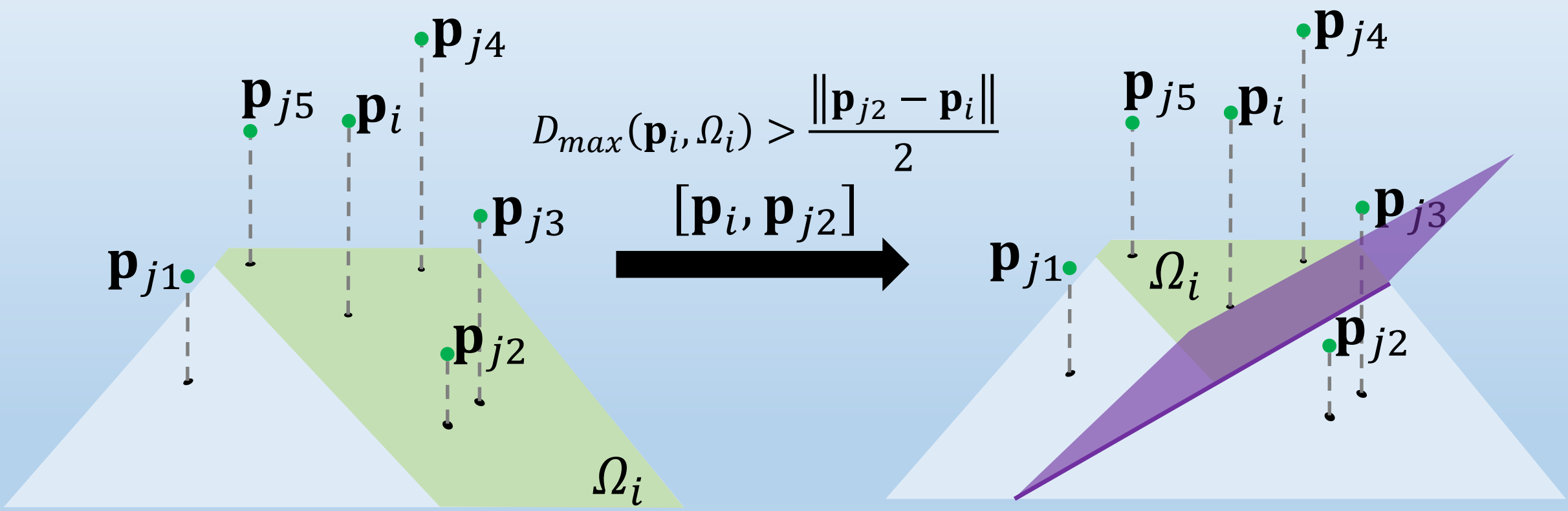
=



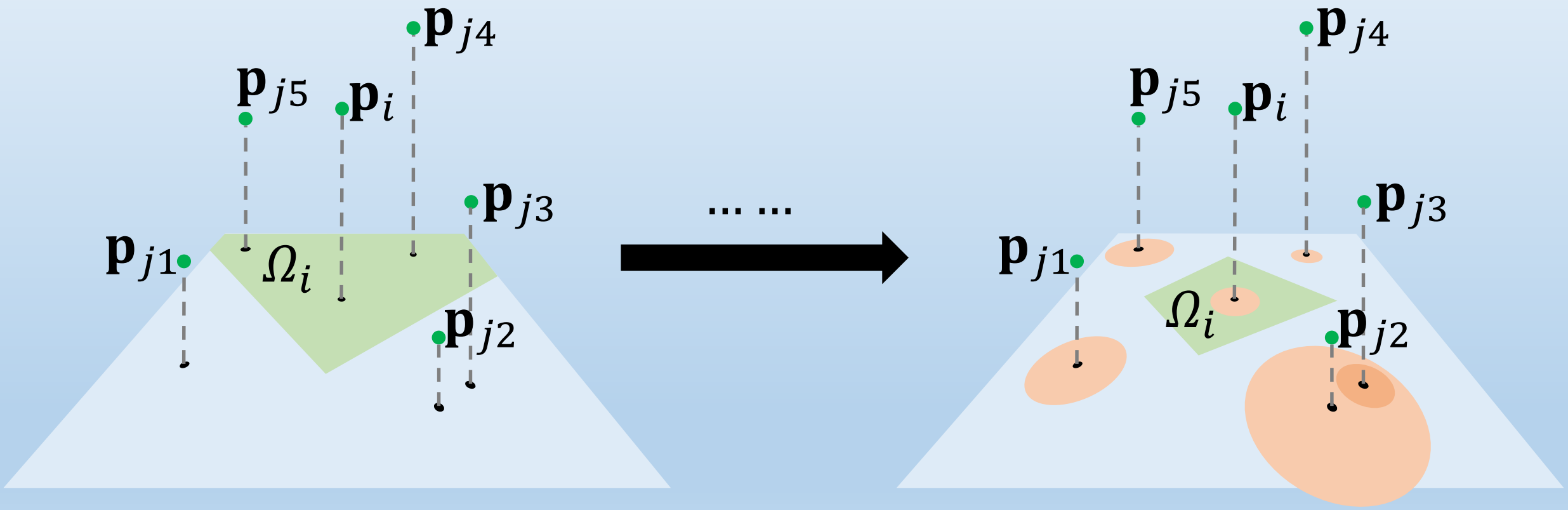
Power图的快速构造：基于kNN的方法



Power图的快速构造：基于kNN的方法

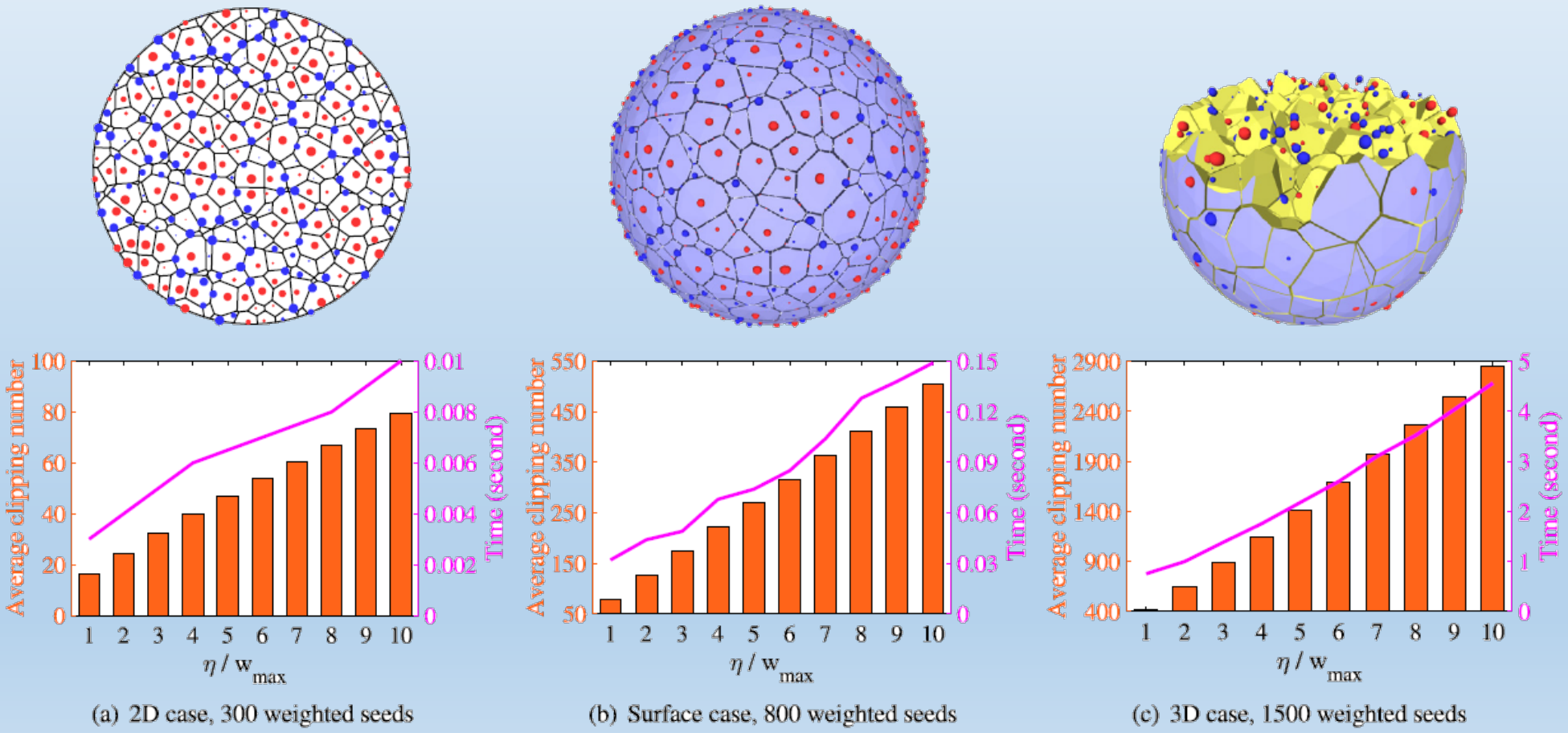


Power图的快速构造：基于kNN的方法



Power图的快速构造：基于kNN的方法

- η控制提升点到R^d的高度，其值越小power胞元构造得越快



Power图的快速构造：加速技巧

- 每个种子有两个提升点可选，它们的胞元相同

$$\mathbf{p} = (\mathbf{v}, \sqrt{\eta - w}), \mathbf{q} = (\mathbf{v}, -\sqrt{\eta - w})$$

$$\Omega_i = \left\{ \mathbf{x} \in R^d \middle| \|\mathbf{y} - \mathbf{p}_i\|^2 \leq \|\mathbf{y} - \mathbf{p}_j\|^2, \forall j \neq i \right\}$$

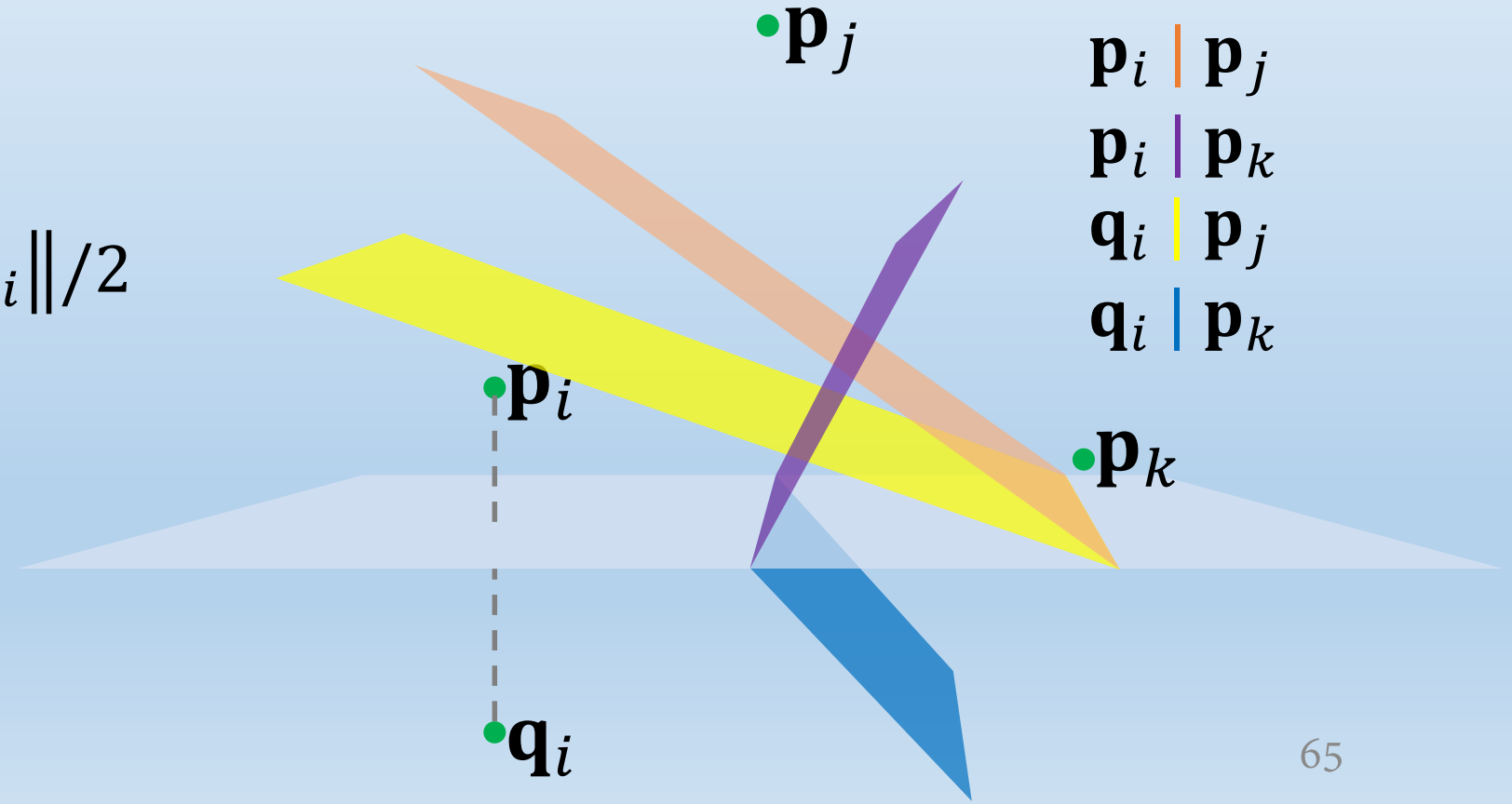
$$\Omega_i = \left\{ \mathbf{x} \in R^d \middle| \|\mathbf{y} - \mathbf{q}_i\|^2 \leq \|\mathbf{y} - \mathbf{p}_j\|^2, \forall j \neq i \right\}$$

Power图的快速构造：加速技巧

- 如果构造$\boldsymbol{p}_i$的胞元：先用$\boldsymbol{p}_i\boldsymbol{p}_j$中垂面，再用$\boldsymbol{p}_i\boldsymbol{p}_k$中垂面
- 如果构造$\boldsymbol{q}_i$的胞元：先用$\boldsymbol{q}_i\boldsymbol{p}_k$中垂面，再用$\boldsymbol{q}_i\boldsymbol{p}_j$中垂面

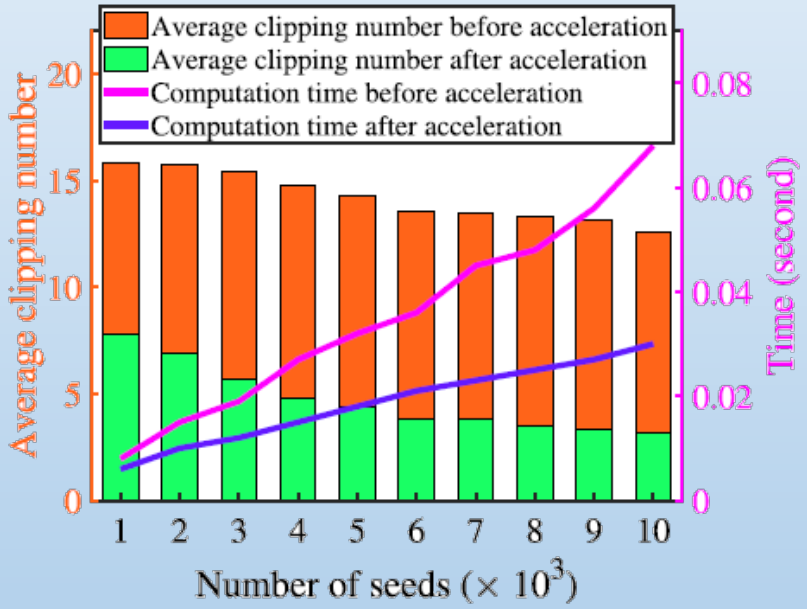
$$D_{max}(\boldsymbol{q}_i, \Omega_i) \leq \|\boldsymbol{p}_j - \boldsymbol{q}_i\|/2$$

可以更早达到

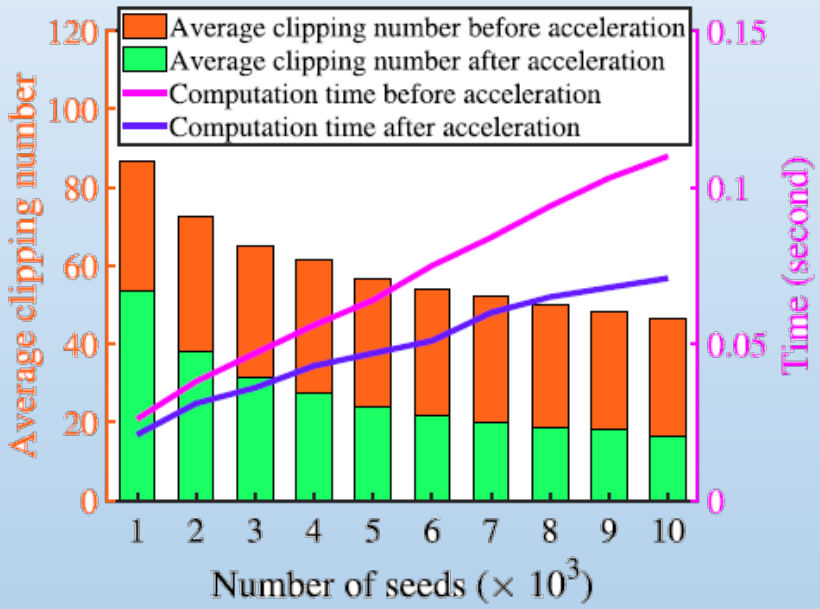


Power图的快速构造：加速技巧

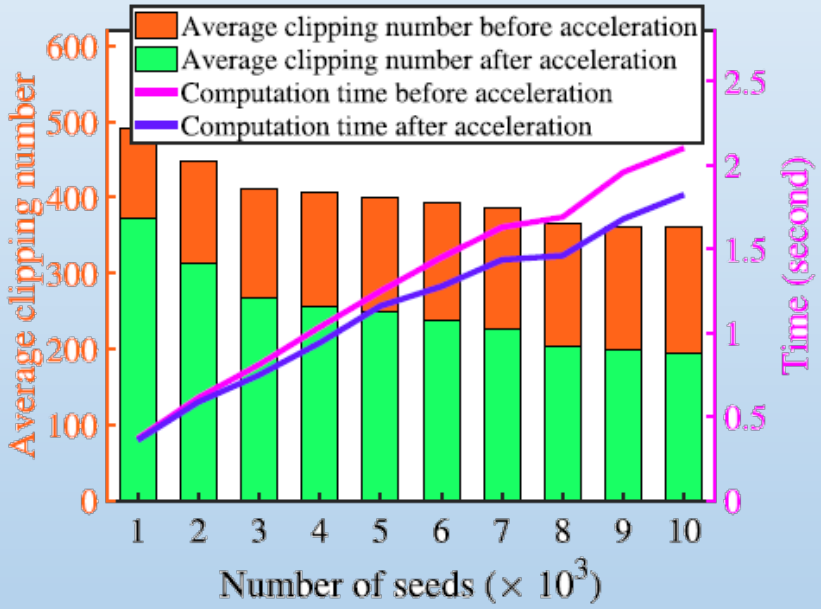
- 实施：提升点集$\{\boldsymbol{p}_i\}_{i=1}^n$构建kd树，搜索$\boldsymbol{q}_i$的k近邻以构造$\Omega_i$



(a) 2D case



(b) Surface case

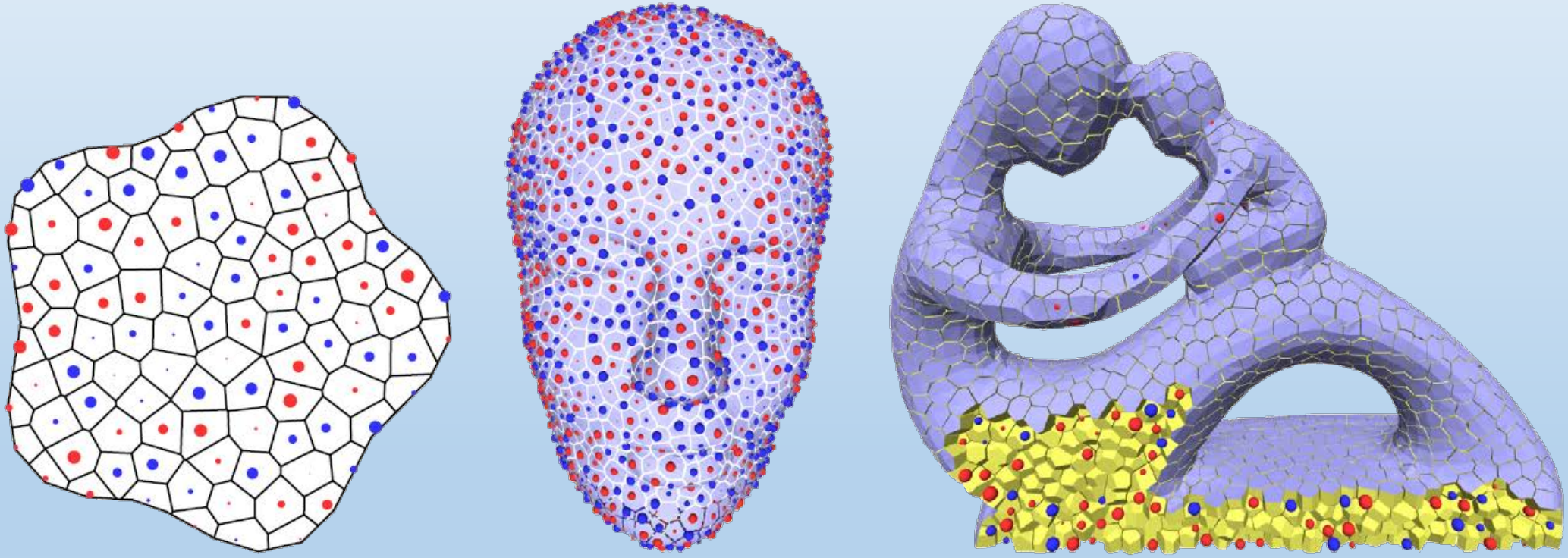


(c) 3D case

加速前后的效率对比

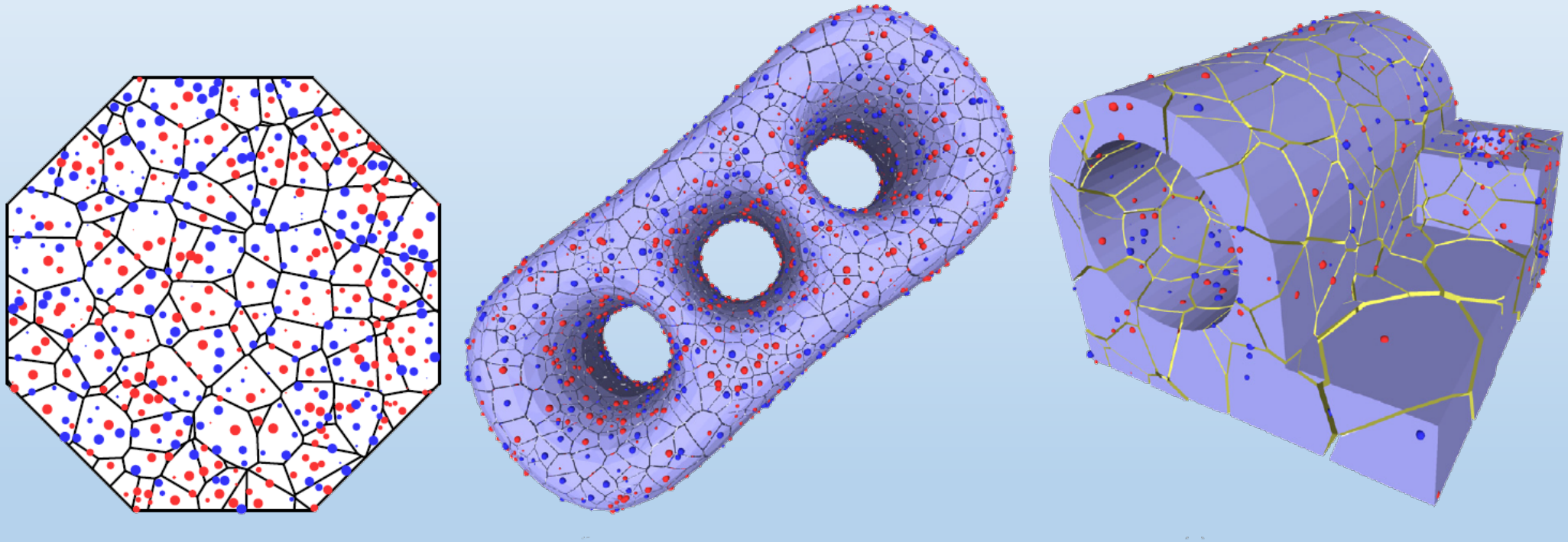
Power图的快速构造：基于kNN的方法

- 结果：均匀位置随机权重



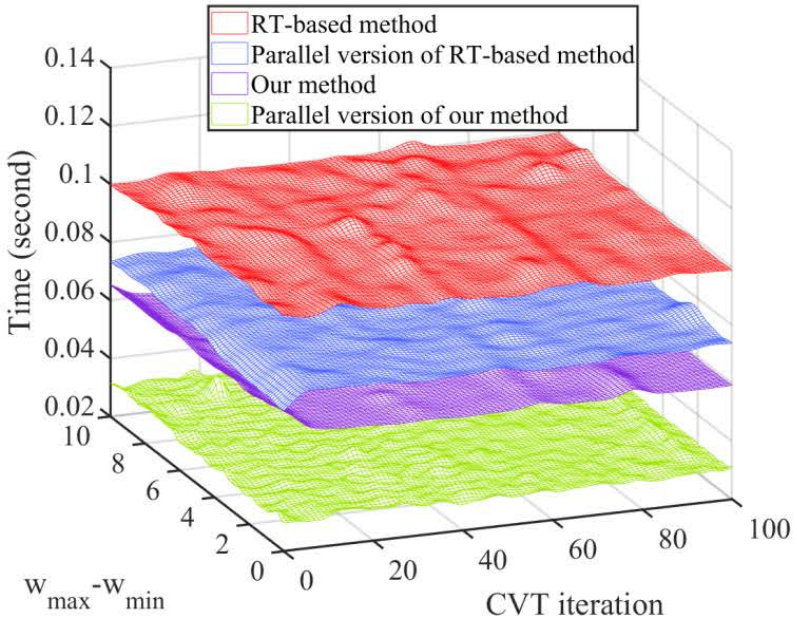
Power图的快速构造：基于kNN的方法

- 结果：随机位置随机权重

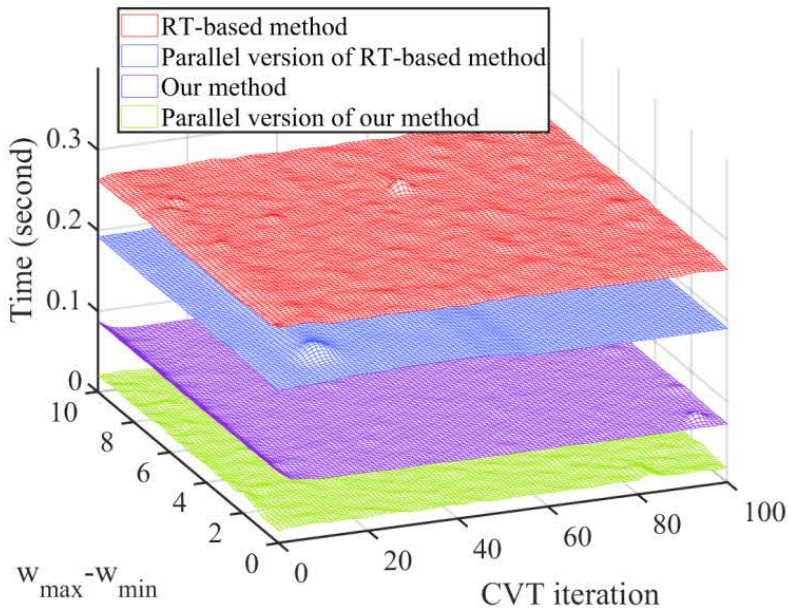


Power图的快速构造：基于kNN的方法

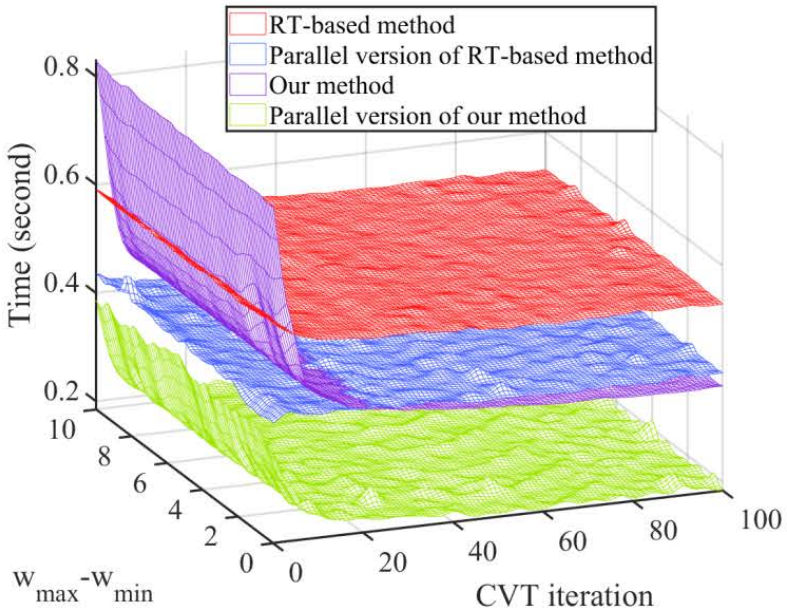
- 结果：运行速度比较



(a) 2D case



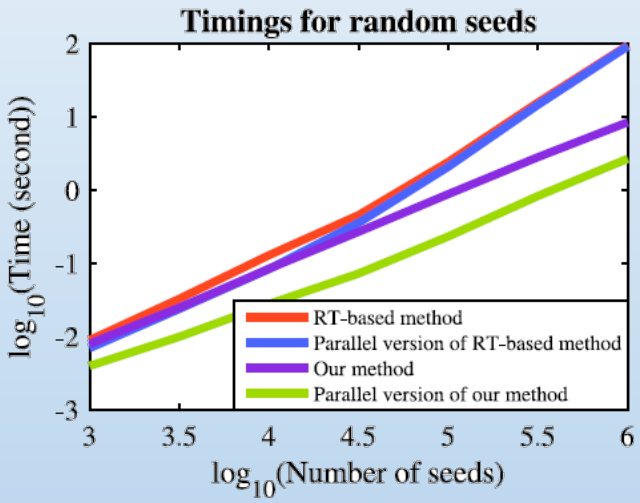
(b) Surface case

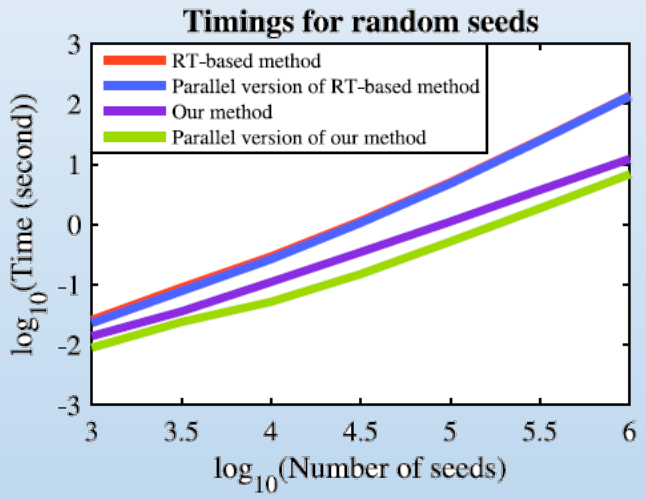


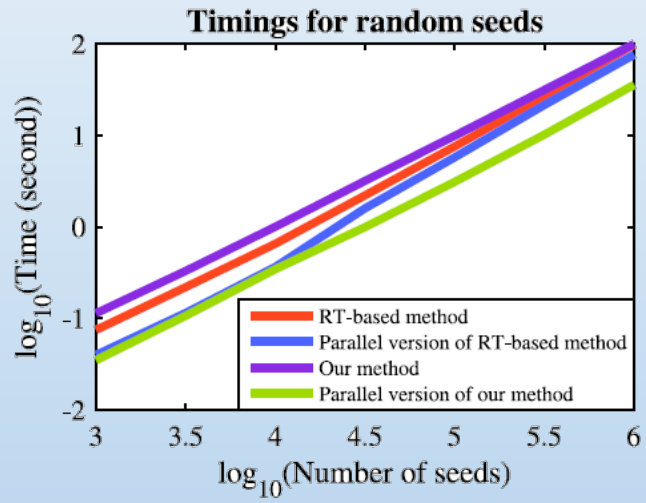
(c) 3D case

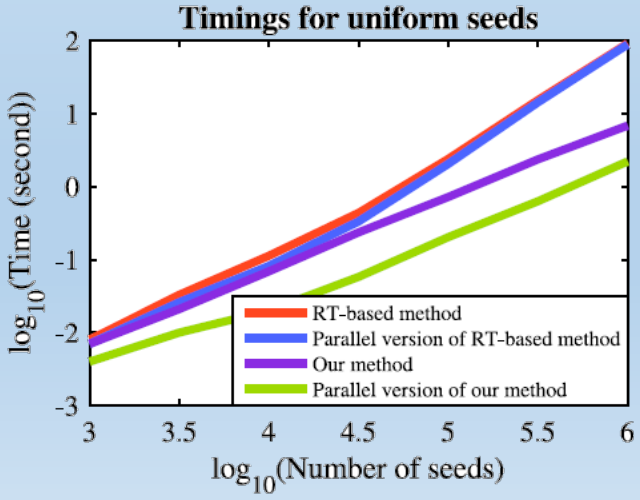
Power图的快速构造：基于kNN的方法

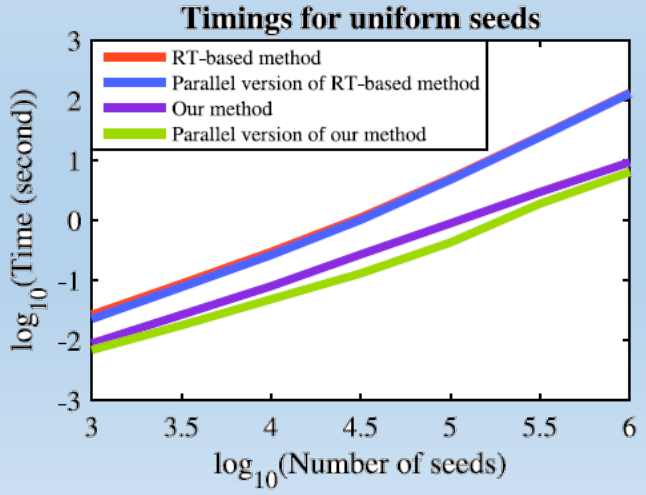
- 结果：运行速度比较

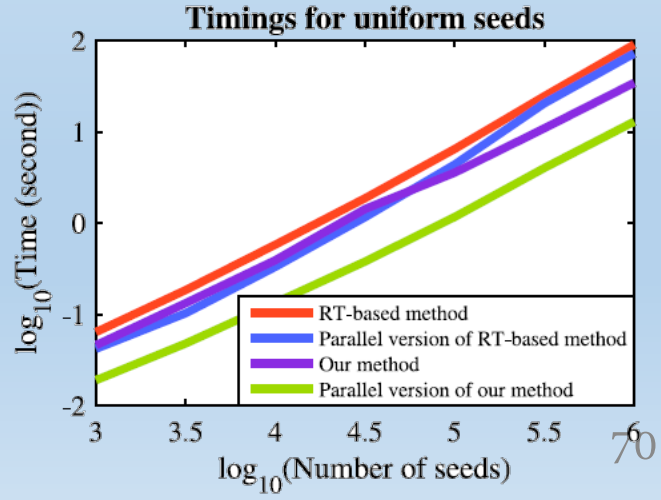












谢谢