A SEQUENTIAL ALGORITHM FOR CONSTRUCTING THE DELAUNAY TRIANGULATION

Trinh Minh Duc
TNU - University of Information and Communication Technology

ARTICLE INFO

Received: 28/12/2023
Revised: 28/3/2024
Published: 29/3/2024

KEYWORDS

Delaunay triangulation
Triangulation
Divide-and-conquer
Voronoi diagram
Computational geometry

ABSTRACT

The problem of constructing Delaunay triangulation is one of the important problems in Computational Geometry. The Delaunay triangulation has been used widely in computational geometry and extended to other multi-purpose domains, for example, GIS, terrain modelling, computer graphics, pattern recognition, finite element analysis… As a result, developing fast and effective algorithms to construct the Delaunay triangulation is becoming more and more important. There are a variety of sequential algorithms implemented effectively for constructing the Delaunay triangulation. In this paper, we present a sequential algorithm for constructing the Delaunay triangulation of a finite planar point set based on divide-and-conquer stratery. In particular, we use a restricted area of the examination of points for testing the circle criterion. The efficiency of the proposed algorithm is verified by comparing its performance with the Delaunay triangulation procedure given by O'Rourke. The results show that the implementation of our algorithm significantly achieves better speedups over O'Rourke's code.

MỘT THUẬT TOÁN TUẦN TỰ ĐỂ XÂY DỰNG LƯỚI TAM GIÁC DELAUNAY

Trịnh Minh Đức
Trường Đại học Công nghệ thông tin và Truyền thông – ĐH Thái Nguyên

THÔNG TIN BÀI BÁO

Ngày nhận bài: 28/12/2023
Ngày hoàn thiện: 28/3/2024
Ngày đăng: 29/3/2024

TỪ KHÓA

Lưới tam giác Delaunay
Sơ đồ Voronoi
Lưới tam giác
Chia để trị
Hình học tính toán

TÓM TẮT

Bài toán xây dựng lưới tam giác Delaunay là một trong các bài toán cơ bản trong hình học tính toán. Lưới tam giác Delaunay đã được sử rộng rãi trong hình học tính toán và được mở rộng sang nhiều lĩnh vực khác như hệ thống thông tin địa lý (GIS), mô hình hóa địa hình, đồ họa máy tính và đa phương tiện, phần tử hữu hạn, nhận dạng mẫu... Do đó, việc phát triển các thuật toán nhanh và hiệu quả để xây dựng lưới tam giác Delaunay ngày càng trở nên quan trọng. Có nhiều thuật toán tuần tự được cài đặt một cách hiệu quả để xây dựng tam giác Delaunay. Trong bài báo này, chúng tôi trình bày một thuật toán tuần tự để xây dựng lưới tam giác Delaunay của một tập điểm phẳng hữu hạn dựa trên chiến lược chia để trị. Cụ thể, chúng tôi đưa ra một vùng giới hạn để loại bỏ đi các điểm không cần thiết cho việc kiểm tra tiêu chí đường tròn trong quá trình ghép nối hai lưới tam giác Delaunay trong hai tập con liền kề nhau. Tính hiệu quả của thuật toán đề xuất được chứng minh bằng việc so sánh sự thực thi của nó với một sự cài đặt xây dựng lưới tam giác Delaunay được đưa ra bởi O'Rourke. Các kết quả thực nghiệm chỉ ra rằng sự cài đặt thuật toán chúng tôi thu được sự tăng tốc tốt hơn đáng kể so với sự cài đặt của O'Rourke.

DOI: https://doi.org/10.34238/tnu-jst.9487

* Corresponding author. *Email: tmduc@ictu.edu.vn*

1. Giới thiệu

Bài toán xây dựng lưới tam giác Delaunay là một trong các bài toán cơ bản trong hình học tính toán. Lưới tam giác Delaunay cùng với đồ thị đối ngẫu của nó sơ đồ Voronoi đã được ứng dụng rộng rãi trong nhiều lĩnh vực như hệ thống thông tin địa lý (GIS), mô hình hóa địa hình, đồ họa máy tính và đa phương tiện, phần tử hữu hạn, nhận dạng mẫu [1]-[3]... Vì những ứng dụng to lớn của lưới tam giác Delaunay trong các lĩnh vực nói trên nên nhiều nhà nghiên cứu đã đưa ra các thuật toán cả tuần tự lẫn song song để xây dựng lưới tam giác Delaunay. Điển hình là các công trình của Fortune [4] với thuật toán quét; Dwyer [5], Guibas và Stolfi [6] với thuật toán chia để trị; thuật toán tăng dần của Green và Sibson [7]...

Rất nhiều các thuật toán hiệu quả để xây dựng lưới tam giác Delaunay cả tuần tự lẫn song song [8]-[10] là dựa trên chiến lược chia để trị. Chi tiết thuật toán xây dựng lưới tam giác Delaunay dựa trên chiến lược chia để trị được trình bày trong [5], [6], [10], [11]. Thuật toán chia để trị được đề xuất bởi Shamos và Hoey [12] và được ứng dụng để xây dựng sơ đồ Voronoi với độ phức tạp thời gian là $O(n\log n)$. Thuật toán này sau đó được sử dụng bởi Guibas và Stolfi [6], Lee và Schachter [11] để xây dựng lưới tam giác Delaunay, thời gian chạy của nó là tiệm cận tối ưu (độ phức tạp thời gian là $O(n\log n)$). Dwyer [5] tiếp tục cải tiến thuật toán để làm giảm độ phức tạp thời gian tới $O(n\log\log n)$, tuy nhiên số điểm đầu vào lớn nhất cho sự cải tiến này là 65536 điểm.

Trong bài báo này, chúng tôi trình bày một thuật toán tuần tự mới để xây dựng lưới tam giác Delaunay của một tập điểm phẳng hữu hạn dựa trên chiến lược chia để trị của Guibas và Stolfi. Cụ thể, chúng tôi đưa ra một vùng giới hạn để loại bỏ đi các điểm không cần thiết cho việc kiểm tra tiêu chí đường tròn trong quá trình ghép nối hai lưới tam giác Delaunay trong hai tập con liền kề nhau.

Bài báo được tổ chức như sau: Trong Phần 2, chúng tôi sẽ trình bày một số khái niệm cơ bản và xem xét lại một vài công việc có liên quan, sau đó chúng tôi trình bày thuật toán đề xuất. Phần 3 đưa ra các kết quả thực nghiệm. Phần cuối dành cho kết luận.

2. Cở sở lý thuyết và thuật toán đề xuất

2.1. Cơ sở lý thuyết

Cho $V = \{v_1, v_2,..., v_n\}$ là một tập bao gồm n điểm phân biệt trong không gian Euclidean hai chiều $\mathbb{R}^2$. Giả sử rằng, tất cả các điểm là không thẳng hàng và không có bốn điểm nào nằm trên cùng một đường tròn. Với hai điểm p, q bất kỳ trong mặt phẳng, chúng tôi ký hiệu một đoạn thẳng hay cạnh $[p,q] = \{(1-\lambda)p + \lambda q : 0 \leq \lambda \leq 1\}$ và ký hiệu pq là đường thẳng đi qua hai điểm p và q [13]. Một đường thẳng có hướng là một đường thẳng được xác định bởi hai điểm đã cho trong một thứ tự cụ thể (p,q). Một điểm z nằm bên trái của (p,q) nếu diện tích của tam giác theo thứ tự ngược chiều kim đồng hồ Δpqz (Δpqz biểu thị một tam giác với ba đỉnh p,q,z) là dương [14]. Đặt E là tập các cạnh hay đoạn thẳng giữa các đỉnh trong V.

Định nghĩa 2.1 [14]: Lưới tam giác $T = (V,E)$ là một đồ thị phẳng mà mỗi cạnh không chứa điểm nào khác ngoài hai điểm đầu mút của nó, không có hai cạnh nào cắt nhau và tất cả các mặt là những tam giác với hợp của chúng là bao lồi của tập điểm V.

Định nghĩa 2.2 [14]-[16]: Một cạnh $[v_i,v_j] \in E$ được cho là thỏa mãn đặc tính đường tròn rỗng nếu tồn tại một đường tròn đi qua v_i, v_j sao cho đường tròn này không chứa bất kỳ điểm nào của V ở bên trong nó. Một lưới tam giác T của V là một lưới tam giác Delaunay ($DT(V)$) nếu mỗi cạnh của T thỏa mãn đặc tính đường tròn rỗng. Tam giác $\Delta v_iv_jv_k$ là một tam giác Delaunay của $DT(V)$ nếu và chỉ nếu đường tròn ngoại tiếp của nó không chứa bất kỳ điểm nào của V ở bên trong nó.

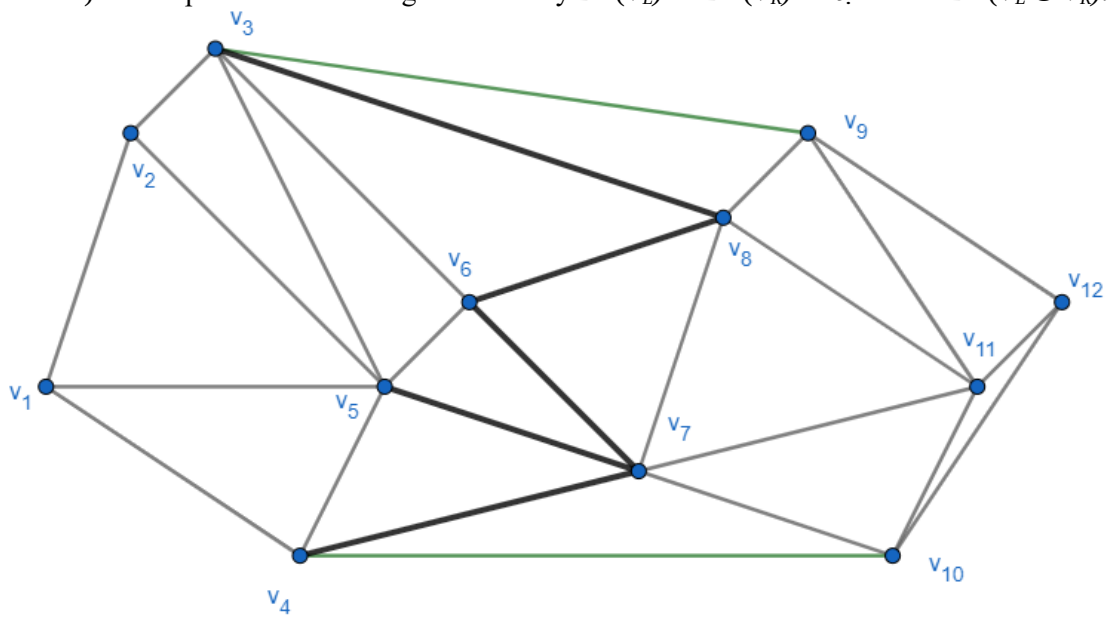
Đặc tính sau cùng trong định nghĩa trên được gọi là tiêu chí đường tròn. Nó thường được sử dụng như một quy tắc để xây dựng lưới tam giác Delaunay.

Định nghĩa 2.3 [14]-[16]: Một ***tiếp tuyến chung*** của hai đa giác lồi đơn giản là một đoạn thẳng nằm ở bên ngoài của cả hai đa giác, giao cắt mỗi đa giác tại một đỉnh duy nhất. Nếu được

kéo dài vô hạn theo bất kỳ hướng nào, tiếp tuyến chung sẽ không cắt vào phần bên trong của bất kỳ đa giác nào. ***Tiếp tuyến chung trên*** là đường thẳng giao cắt mỗi đa giác tại một đỉnh sao cho toàn bộ phần của cả hai đa giác nằm dưới đường tiếp tuyến này. ***Tiếp tuyến chung dưới*** là đường thẳng giao cắt mỗi đa giác tại một đỉnh sao cho toàn bộ phần của cả hai đa giác nằm trên đường tiếp tuyến này.

Thuật toán của Guibas và Stolfi [6] là một thuật toán tuần tự quan trọng được sử dụng để xây dựng lưới tam giác Delaunay của một tập điểm phẳng hữu hạn. Thuật toán này phù hợp với mô hình chia để trị tiêu chuẩn trong việc giải quyết một bài toán bằng cách chia đệ quy nó thành các bài toán con nhỏ hơn và sau đó kết hợp các kết quả của các bài toán con để giải bài toán ban đầu. Thuật toán của Guibas và Stolfi là hiệu quả (độ phức tạp thời gian trong trường hợp xấu nhất là $O(n\log n)$) với một cấu trúc dữ liệu **Quad-edge** để hỗ trợ cho việc ghép nối hai lưới tam giác Delaunay (DT) trong hai tập con liền kề nhau và do vậy chúng tôi đã chọn thuật toán chia để trị của Guibas và Stolfi. Các bước cơ bản như sau:

1) Sắp xếp tập điểm V theo thứ tự tăng dần bởi tọa độ x.
2) Chia tập điểm V thành các tập con trái và phải (V_L và V_R) sao cho mỗi tập con có số lượng điểm xấp xỉ nhau.
3) Xây dựng một cách đệ quy các lưới tam giác Delaunay $DT(V_L)$ và $DT(V_R)$.
4) Ghép nối hai lưới tam giác Delaunay $DT(V_L)$ và $DT(V_R)$ để tạo thành $DT(V_L \cup V_R)$.

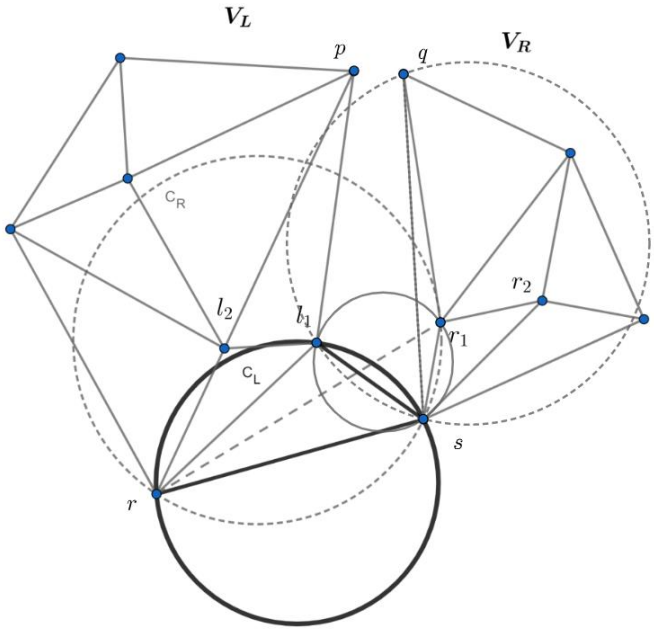


Hình 1. *Lưới tam giác Delaunay*

Trong thuật toán của Guibas và Stolfi, bước quan trọng và phức tạp nhất là ghép nối hai lưới tam giác trái và phải. Trong Hình 1, chúng ta thấy hai cạnh Delaunay được thêm mới liền kề nhau có chung một đỉnh, do đó một chuỗi các cạnh Delaunay nối tiếp nhau được tạo thành theo phương thẳng đứng, điều này làm cho quá trình ghép nối tuân theo một thứ tự từ dưới lên trên. Để ghép nối hai lưới tam giác Delaunay trái và phải, $DT(V_L)$ và $DT(V_R)$, chúng tôi cần sử dụng các tiếp tuyến chung trên và dưới của hai bao lồi $\text{CH}(V_L)$ và $\text{CH}(V_R)$. Hai tiếp tuyến chung này chính là hai cạnh Delaunay trong $DT(V_L \cup V_R)$ [11]. Quá trình ghép nối bắt đầu với tiếp tuyến chung dưới của hai bao lồi, gọi nó là đường cơ sở *basel* và chọn một điểm thứ ba trong các điểm liền kề với hai điểm đầu mút của tiếp tuyến chung dưới để tạo thành một tam giác Delaunay bằng cách kiểm tra tiêu chí đường tròn ngoại tiếp rỗng. Lúc này chúng ta có thể kết nối điểm đầu mút trái (trong V_L) tới một điểm liền kề với điểm đầu mút phải (trong V_R) của tiếp tuyến chung dưới hoặc kết nối điểm đầu mút phải (trong V_R) tới một điểm liền kề với điểm đầu mút trái (trong V_L)

của tiếp tuyến chung dưới để tạo thành một cạnh Delaunay mới trong $DT(V_L \cup V_R)$. Quá trình ghép nối được lặp lại với cạnh Delaunay mới này thay thế cho tiếp tuyến chung dưới và trở thành một đường *basel* mới. Khi đường *basel* này lên tới tiếp tuyến chung trên của hai bao lồi, quá trình ghép nối kết thúc.

Một ví dụ của quá trình này được minh họa trong Hình 2. Đường tròn ngoại tiếp C_R của tam giác Δrsr_1 không chứa điểm nào của V_R ở bên trong của nó và đường tròn ngoại tiếp C_L của tam giác Δrsl_1 không chứa điểm nào của V_L ở bên trong của nó. Bây giờ chúng ta có kết nối l_1 tới s hoặc r_1 tới r. Trong Hình 2, C_R chứa l_1 ở bên trong của nó, vì thế chúng ta chọn cạnh $[s,l_1]$ là một cạnh Delaunay trong $DT(V_L \cup V_R)$.



Hình 2. *Minh họa quá trình ghép nối $DT(V_L)$ và $DT(V_R)$ để tạo thành $DT(V_L \cup V_R)$*

2.2. Thuật toán đề xuất

Đặt $V = \{v_1, v_2,..., v_n\}$ là tập bao gồm n điểm phân biệt trong mặt phẳng và được sắp xếp tăng dần theo tọa độ x. Giả sử rằng tất cả các điểm là không thẳng hàng và không có bốn điểm nằm trên cùng một đường tròn. Với mỗi điểm v_i chúng tôi lưu một danh sách các điểm liền kề có thứ tự $v_{i1}, v_{i2},..., v_{ik}$ sao cho cạnh $[v_i,v_{ij}]$, $j = 1, ..., k$ là một cạnh Delaunay (cạnh của một tam giác Delaunay được gọi là cạnh Delaunay). Chúng ta biết rằng, quá trình ghép nối hai lưới tam giác trái và phải liền kề nhau là quan trọng và tốn nhiều chi phí nhất, vì vậy trong phần này chúng tôi đề xuất một thuật toán mới để ghép nối hai lưới tam giác Delaunay $DT(V_L)$ và $DT(V_R)$ để tạo thành $DT(V_L \cup V_R)$

Thuật toán

Input: Hai lưới tam giác Delaunay $DT(V_L)$ và $DT(V_R)$, tiếp tuyến chung trên và dưới của hai bao lồi $CH(V_L)$ và $CH(V_R)$.

Output: Lưới tam giác Delaunay $DT(V_L \cup V_R)$.

Bước 1: /* Khởi tạo */

a. **Gán** $r \leftarrow$ điểm đầu mút trái của tiếp tuyến chung dưới, $s \leftarrow$ điểm đầu mút phải của tiếp tuyến chung dưới.

Gọi ***ADD(r,s)*** để tạo thành một cạnh Delaunay $[r,s]$.

b. **Gán** $p \leftarrow$ điểm đầu mút trái của tiếp tuyến chung trên, $q \leftarrow$ điểm đầu mút phải của tiếp tuyến chung trên.

Gọi *ADD(p,q)* để tạo thành một cạnh Delaunay [*p,q*].

Bước 2: /* Bắt đầu với tiếp tuyến chung dưới */

a. Nếu đường tròn đi qua ba điểm *r, s, q* thỏa mãn tiêu chí đường tròn thì cạnh [*r,q*] là một cạnh Delaunay và nhảy tới **Bước 5**.

b. Trái lại nếu đường tròn đi qua ba điểm *r, s, p* thỏa mãn tiêu chí đường tròn thì cạnh [*s,p*] là một cạnh Delaunay và nhảy tới **Bước 6**.

c. Trái lại

i. Tìm một điểm *u* trong các danh sách liền kề của *r* và *s* sao cho *u* nằm về phía trái (*r,s*) và đường tròn đi qua ba điểm *r, s, u* thỏa mãn tiêu chí đường tròn.

ii. Nếu *u* ở trong danh sách liền kề của *r* thì cạnh [*s,u*] là một cạnh Delaunay và nhảy tới **Bước 3**.

iii. Trái lại, cạnh [*r,u*] là một cạnh Delaunay và nhảy tới **Bước 4**.

Bước 3:

a. /* Thêm một cạnh Delaunay vào trong $DT(V_L \cup V_R)$ */

Gọi **ADD(*s,u*)** để tạo thành một cạnh Delaunay [*s,u*]

b. */* Xóa các cạnh không phải là cạnh Delaunay trong $DT(V_L \cup V_R)$ */*

Với mỗi điểm *u'* nằm giữa *r* và *u* trong danh sách liền kề của *r* theo hướng ngược chiều kim đồng hồ (CCW): Nếu cạnh [*r,u'*] là một cạnh Delaunay thì **DEL(*r,u'*).**

c. Gán *r* ← *u* và nhảy tới **Bước 2**.

Bước 4:

a. /* Thêm một cạnh Delaunay vào trong $DT(V_L \cup V_R)$ */

Gọi **ADD(*r,u*)** để tạo thành một cạnh Delaunay [*r,u*]

b. / Xóa các cạnh không phải là cạnh Delaunay trong $DT(V_L \cup V_R)$ */*

Với mỗi điểm *u'* nằm giữa *s* và *u* trong danh sách liền kề của *s* theo hướng cùng chiều kim đồng hồ (CW): Nếu cạnh [*s,u'*] là một cạnh Delaunay thì **DEL(*s,u'*)**.

c. Gán *s* ← *u* và lặp lại **Bước 2**.

Bước 5:

a. /* Thêm một cạnh Delaunay vào trong $DT(V_L \cup V_R)$ */

Gọi **ADD(*r,q*)** để tạo thành một cạnh Delaunay [*r,q*]

b. */* Xóa các cạnh không phải là cạnh Delaunay trong $DT(V_L \cup V_R)$ */*

Với mỗi điểm *u'* nằm giữa *s* và *q* trong danh sách liền kề của *s* theo hướng cùng chiều kim đồng hồ (CW): Nếu cạnh [*s,u'*] là một cạnh Delaunay thì **DEL(*s,u'*)**. **STOP.**

Bước 6:

a. /* Thêm một cạnh Delaunay vào trong $DT(V_L \cup V_R)$ */

Gọi **ADD(*s,p*)** để tạo thành một cạnh Delaunay [*s,p*]

b. */* Xóa các cạnh không phải là cạnh Delaunay trong $DT(V_L \cup V_R)$ */*

Với mỗi điểm *u'* nằm giữa *r* và *p* trong danh sách liền kề của *r* theo hướng ngược chiều kim đồng hồ (CCW): Nếu cạnh [*r,u'*] là một cạnh Delaunay thì **DEL(*r,u'*)**. **STOP.**

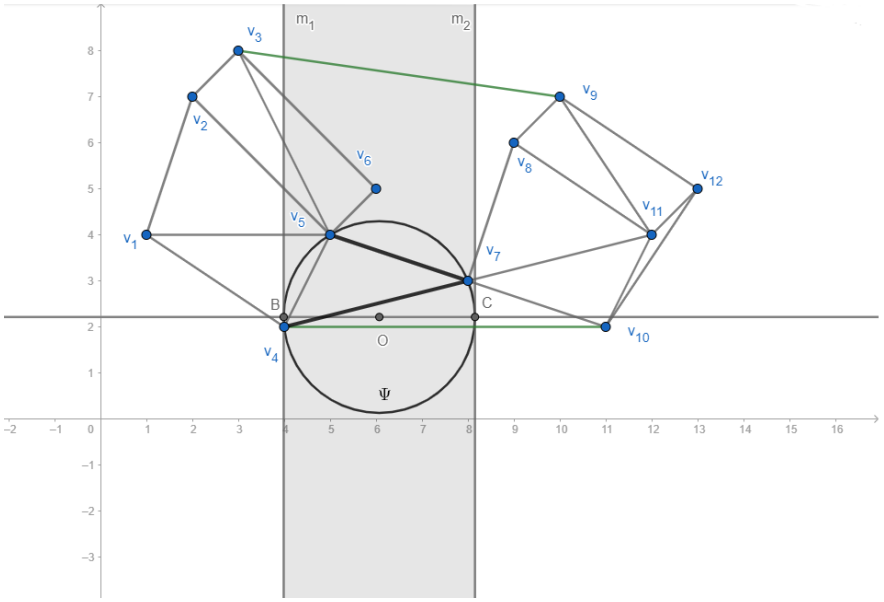
Trong thuật toán ở trên hàm **ADD(*a,b*)** có chức năng thêm *a* vào trong danh sách liền kề của *b* và *b* vào danh sách liền kề của *a* ở vị trí thích hợp. Hàm **DEL(*a,b*)** xóa *a* từ danh sách liền kề của *b* và *b* từ danh sách liền kề của *a*.

Một điều cần lưu ý là có những cạnh là cạnh Delaunay trong $DT(V_L)$ *hoặc* $DT(V_R)$ mà không là cạnh Delaunay trong $DT(V_L \cup V_R)$, những cạnh này sẽ bị xóa khỏi $DT(V_L \cup V_R)$ (các Bước 3b, 4b, 5b, 6b). Trong **Hình 1** chúng ta thấy [*s,q*] là một cạnh Delaunay trong $DT(V_R)$. Tuy nhiên, trong quá trình ghép nối, đường tròn đi qua ba điểm *s*, l_1 và r_1 thỏa mãn tiêu chí đường tròn, trong khi đường tròn đi qua ba điểm *s*, l_1 và *q* không thỏa mãn tiêu chí đường tròn đối với tập *V*. Thêm nữa, điểm *q* nằm giữa *s* và r_1 trong danh sách liền kề của *s* theo hướng cùng chiều kim đồng hồ và tung độ của *q* có giá trị lớn hơn tung độ của r_1, điều này dẫn đến cạnh [l_1,r_1] luôn cắt cạnh [*s,q*]. Do đó, cạnh [*s,q*] không là cạnh Delaunay trong $DT(V_L \cup V_R)$ và phải bị xóa khỏi $DT(V_L \cup V_R)$. Chúng ta thấy rằng, khi đường tròn đi qua ba điểm *s*, l_1 và r_1 thỏa mãn tiêu chí đường

tròn đối với tập *V*, nếu tồn tại một điểm *u'* nằm giữa *s* và r_1 trong danh sách liền kề của *s* theo hướng cùng chiều kim đồng hồ thì cạnh [*s,u'*] không là cạnh Delaunay trong $DT(V_L \cup V_R)$ và phải bị xóa khỏi $DT(V_L \cup V_R)$. Vì vậy, chúng ta khẳng định rằng các Bước 3b, 4b, 5b, 6b là hoàn toàn chính xác.

Để kiểm tra một tam giác nào đó có thỏa tiêu chí đường tròn, thông thường chúng ta phải kiểm tra tất cả các điểm của tập *V*, điều này dẫn tới làm tăng thời gian tính toán. Để giảm bớt số điểm cần kiểm tra, chúng tôi đưa ra một vùng giới hạn cho việc kiểm tra các điểm, khi đó chúng ta chỉ cần kiểm tra những điểm thuộc vùng giới hạn đó.

Trong Hình 3, chúng ta cần ghép nối hai lưới tam giác Delaunay trái và phải. Để kiểm tra liệu tam giác $\Delta v_4v_7v_5$ có là một tam giác Delaunay hay không, chúng ta cần kiểm tra xem đường tròn Ψ mà đi qua v_4, v_7, v_5 có thỏa mãn tiêu chí đường tròn hay không. Đặt *O* là tâm của đường tròn Ψ, *k* là đường thẳng đi qua *O* và song song với trục hoành, k cắt Ψ ở *B* và *C*. Đặt m_1 là đường thẳng đi qua *B* và song song với trục tung, m_2 là đường thẳng đi qua *C* và song song với trục tung. Bây giờ để kiểm tra xem đường tròn Ψ có thỏa mãn tiêu chí đường tròn hay không, chúng ta chỉ cần kiểm tra các điểm mà có hoành độ nằm giữa hoành độ của hai điểm *B* và *C* (vùng màu xám trong Hình 3 chính là vùng giới hạn cho việc kiểm tra). Rõ ràng rằng, khoảng cách từ một điểm bất kỳ nằm ngoài vùng giới hạn này đến *O* luôn luôn lớn hơn bán kính của đường tròn Ψ, và do vậy chúng bị bỏ qua khi kiểm tra tiêu chí đường tròn.



Hình 3. *Vùng giới hạn cho việc kiểm tra các điểm*

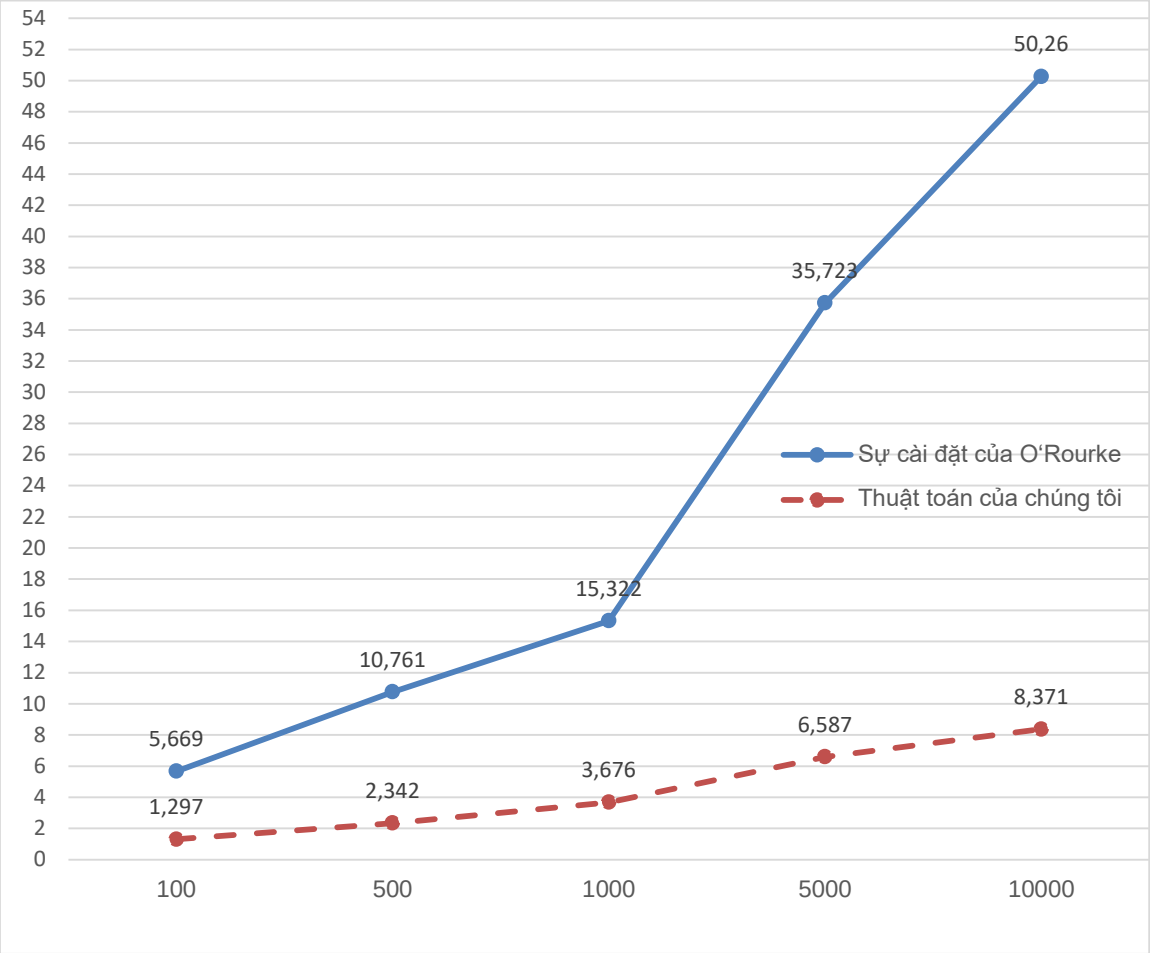
3. Kết quả thực nghiệm

Bảng 1. *Thời gian chạy thực tế thuật toán của chúng tôi và sự cài đặt của O'Rourke (tính theo giây)*

Số điểm đầu vào	Tổng số tam giác Delaunay	Sự cài đặt của O'Rourke	Thuật toán của chúng tôi
100	198	5.669	1.297
500	980	10.761	2.342
1000	1914	15.322	3.676
5000	9968	35.723	6.587
10000	19859	50.260	8.371

Thuật toán của chúng tôi được cài đặt bằng ngôn ngữ lập trình C và được so sánh với một sự cài đặt xây dựng lưới tam giác Delaunay được đưa ra bởi O'Rourke [14]. Mã nguồn được biên

dịch bằng trình biên dịch GNU C trên hệ điều hành Ubuntu 18.04, bộ vi xử lý Intel Core i3 2.67 GHz, RAM 8 GB. Thời gian chạy thực tế thuật toán của chúng tôi và sự cài đặt của O'Rourke trên một số tập hữu hạn các điểm được lấy ngẫu nhiên bên trong một hình vuông có kích cỡ 1000 x 1000 được đưa ra trong Bảng 1 và Hình 4. Các kết quả này chỉ ra rằng, thuật toán của chúng tôi chạy nhanh hơn đáng kể sự cài đặt của O'Rourke.



Hình 4. *Thời gian chạy thực tế thuật toán của chúng tôi và sự cài đặt của O'Rourke (tính theo giây)*

4. Kết luận

Trong bài báo này, chúng tôi đã đưa ra một thuật toán tuần tự để xây dựng lưới tam giác Delaunay của một tập điểm phẳng hữu hạn dựa trên chiến lược chia để trị. Bên cạnh đó, chúng tôi cũng đưa ra một vùng giới hạn (vùng màu xám trong Hình 3) cho việc kiểm tra các điểm khi sử dụng tiêu chí đường tròn. Chương trình được cài đặt bằng ngôn ngữ C và có thể dễ dàng biên dịch trên các hệ điều hành khác.

TÀI LIỆU THAM KHẢO/ REFERENCES

[1] K. H. Huebner, D. L. Dewhirst, D. E. Smith, and T. G. Byrom, *The Finite Element Method for Engineers*. Wiley, New York, NY, USA, 2001.

[2] L. Tang, C. Ren, Z. Liu, and Q. Li, "A Road Map Refinement Method Using Delaunay Triangulation for Big Trace Data," *ISPRS International Journal of Geo-Information*, vol. 6, no. 2, pp. 141-153, 2017.

[3] M. Flores, G. Torres, G. García, and M. Licona, "Fingerprint verification methods using delaunay triangulations," *International Arab Journal of Information Technology*, vol. 14, pp. 346-354, 2017.

[4] S. Fortune, "A sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, pp. 153-174, 1987.
[5] R. A. Dwyer, "A faster divide-and-conquer algorithm for constructing Delaunay triangulations," *Algorithmica*, vol. 2, pp. 137-151, 1987.
[6] L. Guibas and J. Stolfi, "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams," *ACM Transactions on Graphics*, vol. 4, no. 2, pp. 74-123, 1985.
[7] P. Green and R. Sibson, "Computing Dirichlet tessellations in the plane," *Computer Journal*, vol. 21, pp. 168-173, 1977.
[8] M. -B.Chen, T. -R. Chuang, and J. -J. Wu, "Parallel divide-and-conquer scheme for 2D Delaunay triangulation," *Concurrency and Computation: Practice and Experience*, vol. 18, pp. 1595-1612, 2006.
[9] P. T. An and L. H. Trang, "A Parallel Algorithm Based on Convexity for the Computing of Delaunay Tessellation," *Numerical Algorithms*, vol. 59, no. 3, pp. 347–357, 2012.
[10] W. Wu, Y. Rui, F. Su, L. Cheng, and J. Wang, "Novel parallel algorithm for constructing Delaunay triangulation based on a twofold-divide-and-conquer scheme," *GIScience & Remote Sensing*, vol. 51, no. 5, pp. 537-554, 2014.
[11] D. T. Lee and B. J. Schachter, "Two Algorithms for Constructing a Delaunay Triangulation," *International Journal of Computer and Information Sciences*, vol. 9, no. 3, pp. 219-242, 1980.
[12] M. I. Shamos and D. Hoey, "Closest-point Problems," *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, 1975, pp. 151-162.
[13] P. T. An, "Method of orienting curves for determining the convex hull of a finite set of points in the plane," *Optimization*, vol. 59, no. 2, pp. 175-179, 2010.
[14] J. O'Rourke, *Computational Geometry in C*, Second edition. Cambridge University Press, 1998.
[15] M. de Berg, *Computational Geometry Algorithms and Applications*. Springer, Germany, 2000.
[16] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of voronoi Diagrams*, First edition, John Winley and Sons Ltd, 1992.