STUDYING TECHNIQUES OF COMPUTATIONAL GEOMETRY FOR TWO-DIMENSION RANGE SEARCH ALGORITHM ASSISTING THE QUERY FOR DATABASE

Le Thi Thuan
University of Khanh Hoa

ARTICLE INFO

Received: 15/02/2023
Revised: 31/3/2023
Published: 07/4/2023

KEYWORDS

Computational geometry
Range searching
KD-trees
General sets of points
Geometric data structures

ABSTRACT

Nowadays, the rapid development of science and technology leads to the creation of enormous multimedia databases. Therefore, optimizing database queries has much attention from communication research. At first sight it seems that databases have little to do with geometry. Nevertheless, queries about data in a database can be interpreted geometrically. To this end we transform records in a database into points in a multi-dimensional space, and we transform the queries about the records into queries on this set of points. This paper proposed an approach based on computational geometry techniques for building KD-trees, two-dimensional range search algorithm (search KD trees algorithm) and general sets of points to improve the algorithm. The algorithm also applies to data query optimization, a problem that is still challenging today. Applying build KD-trees algorithm and search KD-trees algorithm to build a search application on two-dimensional range search shows the potential of this research for many real-world problems related to database query optimization.

NGHIÊN CỨU CÁC KỸ THUẬT CỦA HÌNH HỌC TÍNH TOÁN CHO THUẬT TOÁN TÌM KIẾM PHẠM VI HAI CHIỀU HỖ TRỢ BÀI TOÁN TRUY VẤN CƠ SỞ DỮ LIỆU

Lê Thị Thuấn
Trường Đại học Khánh Hòa

THÔNG TIN BÀI BÁO

Ngày nhận bài: 15/02/2023
Ngày hoàn thiện: 31/3/2023
Ngày đăng: 07/4/2023

TỪ KHÓA

Hình học tính toán
Phạm vi truy vấn
Cây KD
Tập điểm chung
Cấu trúc dữ liệu hình học

TÓM TẮT

Ngày nay, với sự phát triển mạnh về khoa học và công nghệ, cơ sở dữ liệu đa phương tiện ra đời với dung lượng rất lớn, vì vậy việc tối ưu hóa truy vấn dữ liệu là một bài toán nhận được sự quan tâm của các nhà nghiên cứu. Hãy xem xét mối tương quan giữa cơ sở dữ liệu với hình học bằng phương pháp chuyển đổi các bản ghi trong một cơ sở dữ liệu thành các điểm trong không gian đa chiều và chuyển đổi các truy vấn về các bản ghi thành các truy vấn lên tập các điểm này. Nghiên cứu đã ứng dụng các kỹ thuật của hình học tính toán để xây dựng cây KD và thuật toán tìm kiếm phạm vi 2 chiều - cây KD, đồng thời đề xuất phương pháp tập điểm chung để cải tiến thuật toán được đề xuất. Thuật toán được đề xuất sẽ áp dụng trong việc tối ưu hóa truy vấn dữ liệu, một vấn đề còn nhiều thách thức hiện nay. Áp dụng thuật toán để xây dựng chương trình tìm kiếm trên phạm vi hai chiều – cây KD cũng cho thấy tiềm năng của nghiên cứu này cho nhiều vấn đề trong thực tế liên quan đến bài toán tối ưu hóa truy vấn dữ liệu.

DOI: https://doi.org/10.34238/tnu-jst.7336

Email: lethithuan84@gmail.com

1. Giới thiệu

Những tiến bộ về kỹ thuật trong những năm gần đây tạo ra các cơ sở dữ liệu đa phương tiện có thể lưu trữ hình ảnh, phim và tiếng nói. Các hệ thống thông tin địa lý có thể lưu trữ và phân tích các bản đồ, các dữ liệu về thời tiết và các ảnh vệ tinh [1], [2]. Kho dữ liệu và các hệ thống phân tích trực tuyến được sử dụng trong nhiều công ty để lấy ra và phân tích những thông tin có lợi từ các cơ sở dữ liệu rất lớn nhằm đưa ra các quyết định [3]. Các kỹ thuật cơ sở dữ liệu động và thời gian thực được sử dụng trong việc kiểm tra các tiến trình công nghiệp và sản xuất [4]. Vì vậy các hệ cơ sở dữ liệu cũng theo đó tăng lên cả về dung lượng lẫn tính phức tạp của dữ liệu. Lúc này nảy sinh vấn đề là làm thế nào để việc tìm kiếm trên các cơ sở dữ liệu, đặc biệt là trên các cơ sở dữ liệu lớn, truy vấn một cách nhanh chóng, chính xác và tốn ít tài nguyên? Đây là bài toán luôn được các nhà khoa học cũng như các nhà ứng dụng quan tâm [5]. Một hướng tiếp cận khác là ứng dụng hình học tính toán để giải quyết bài toán này. Để xây dựng một thuật toán ứng dụng kĩ thuật hình học tính toán bao gồm các bước: (i) Phân tích và thiết kế thuật toán sơ bộ ban đầu; (ii) Hiệu chỉnh sao cho thuật toán đúng đắn khi xuất hiện các trường hợp suy biến; (iii) Thực thi thuật toán [6]. Một phương pháp hình học tính toán đề xuất trong nghiên cứu này là xây dựng thuật toán tìm kiếm trên phạm vi 2 chiều - cây KD, nhiều nghiên cứu đã áp dụng thuật toán này như Madmood áp dụng thuật toán này để tối ưu hóa tìm kiếm trên bản đồ số ở Ai Cập [7], Otair sử dụng cấu trúc dữ liệu cây KD để phân cụm dữ liệu [8].

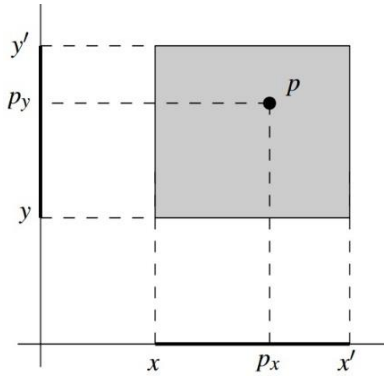
Hãy xem xét ví dụ về một dạng truy vấn thông thường lên cơ sở dữ liệu nói trên là liệt kê tất cả các nhân viên sinh từ năm 1950 đến năm 1955 và có thu nhập hàng tháng từ 3000 đến 4000 USD. Để thực hiện truy vấn cơ sở dữ liệu theo phương pháp hình học, phải chuyển đổi các bản ghi trong một cơ sở dữ liệu thành các điểm trong không gian đa chiều và chuyển đổi các truy vấn về các bản ghi thành các truy vấn lên tập các điểm này [9], [10]. Bài toán sẽ sử dụng đại diện mỗi nhân viên bởi một điểm trong mặt phẳng, với mỗi điểm lưu trữ các thông tin khác của nhân viên, chẳng hạn như tên và địa chỉ. Nghiên cứu đề xuất sử dụng phương pháp tìm kiếm trên phạm vi hai chiều - cây KD. Tuy nhiên, không thể tránh khỏi việc có một số nhân viên có những thông tin trùng nhau. Vì vậy, phương pháp tập điểm chung được đề xuất để cải tiến thuật toán tìm kiếm trên cây KD trong trường hợp có các tập điểm trùng nhau. Thuật toán được trình bày chi tiết bởi các bước và được minh họa bằng tập các điểm trong không gian hai chiều.

2. Xây dựng thuật toán

2.1. Phương pháp tìm kiếm trên phạm vi hai chiều – cây KD

Trong mục này, xét bài toán tìm kiếm trên phạm vi hai chiều.

Cho P là tập hợp của n điểm trong mặt phẳng, giả thiết rằng không có hai điểm nào của P có cùng tọa độ x hoặc y. Giả thiết này không thực tế nhưng sẽ giải quyết những hạn chế này ở mục 2.2.



Hình 1. *Truy vấn trên phạm vi hai chiều*

Một truy vấn trên phạm vi hai chiều lên P, yêu cầu liệt kê các điểm từ P mà nằm trong phạm vi truy vấn chữ nhật $[x: x'] \times [y: y']$. Một điểm $p \coloneqq (p_x, p_y)$ nằm trong phạm vi này nếu và chỉ nếu $p_x \in [x: x']$ và $p_y \in [y: y']$ (hình 1).

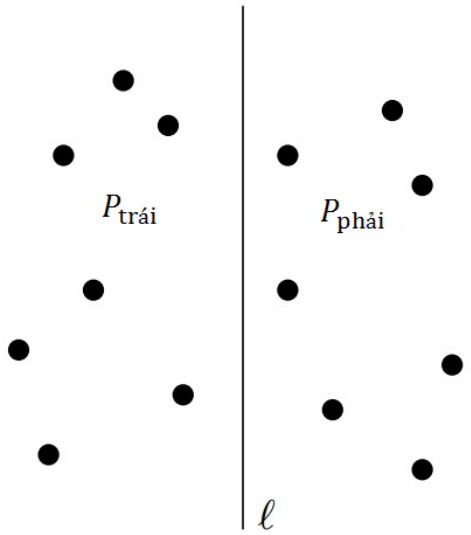
Một truy vấn trên phạm vi hai chiều là tổ hợp của hai truy vấn con một chiều, một trên tọa độ x của các điểm và một trên tọa độ y.

2.1.1. Thuật toán xây dựng cây KD

Xây dựng cây KD theo cách sử dụng đệ quy để tạo cây nhị phân tìm kiếm cân bằng: tập các điểm được phân thành hai tập con, một tập con chứa các điểm nhỏ hơn hoặc bằng giá trị phân tách, tập con còn lại chứa các điểm lớn hơn giá trị phân tách (hình 2). Giá trị phân tách được lưu trữ tại gốc và hai tập con được lưu trữ bằng cách đệ qui trong hai cây con [11], [12].

Trong trường hợp mỗi điểm có hai giá trị là tọa độ x và tọa độ y, trước tiên phân tách theo tọa độ x, tiếp theo là tọa độ y, sau đó một lần nữa theo tọa độ x, ... quá trình này diễn ra như sau:

- Tại gốc phân tách tập P bởi một đường thẳng đứng ℓ thành hai tập con với kích thước gần như tương đương. Đường thẳng phân tách được lưu trữ tại gốc. $P_{\text{trái}}$ là tập con chứa các điểm nằm bên trái hoặc nằm trên đường thẳng phân tách, $P_{\text{trái}}$ được lưu trữ trong cây con bên trái. $P_{\text{phải}}$ là tập con chứa các điểm nằm bên phải đường thẳng phân tách, $P_{\text{phải}}$ được lưu trữ trong cây con bên phải.



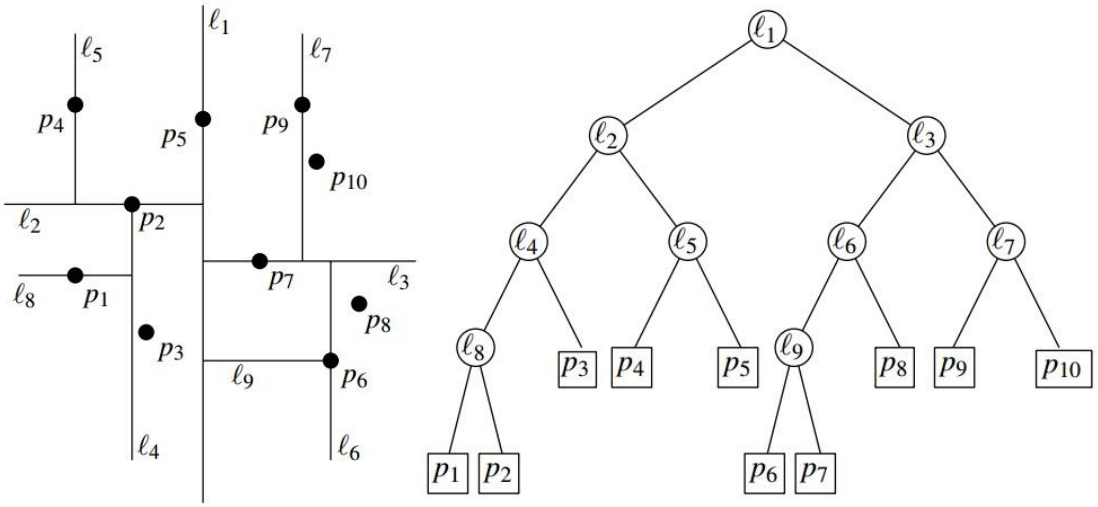
Hình 2. *Phân tách tập P bởi đường thẳng* $\boldsymbol{\ell}$

- Tại nút con bên trái của gốc thực hiện phân tách $P_{\text{trái}}$ thành hai tập con bởi một đường thẳng nằm ngang; các điểm nằm phía dưới đường thẳng được lưu trữ trong cây con bên trái của nút con này, tương tự thì các điểm nằm phía trên đường thẳng được lưu trữ trong cây con bên phải. Đường thẳng phân tách được lưu trữ tại nút con bên trái của gốc.

- Tương tự, tập $P_{\text{phải}}$ được phân tách bởi một đường thẳng nằm ngang thành hai tập con, hai tập này được lưu trữ trong cây con bên trái và bên phải của nút con bên phải của gốc.

- Tại nút cháu của gốc, phân tách một lần nữa bởi một đường thẳng đứng,...

Tổng quát thuật toán, phân tách những nút có chiều sâu chẵn bởi một đường thẳng đứng, và phân tách những nút có chiều sâu lẻ bởi một đường thẳng nằm ngang, cây tạo ra được gọi là cây KD. Việc phân tách này được minh họa hình 3.



Hình 3. *Phân tách tập hợp và cây nhị phân tương ứng (cây KD)*

Thuật toán Xay_dung_cay_KD(P,$chieu\_sau$)

Đầu vào: Một tập điểm P và chiều sâu hiện thời $chieu\_sau$

Đầu ra: Gốc của cây KD lưu trữ P

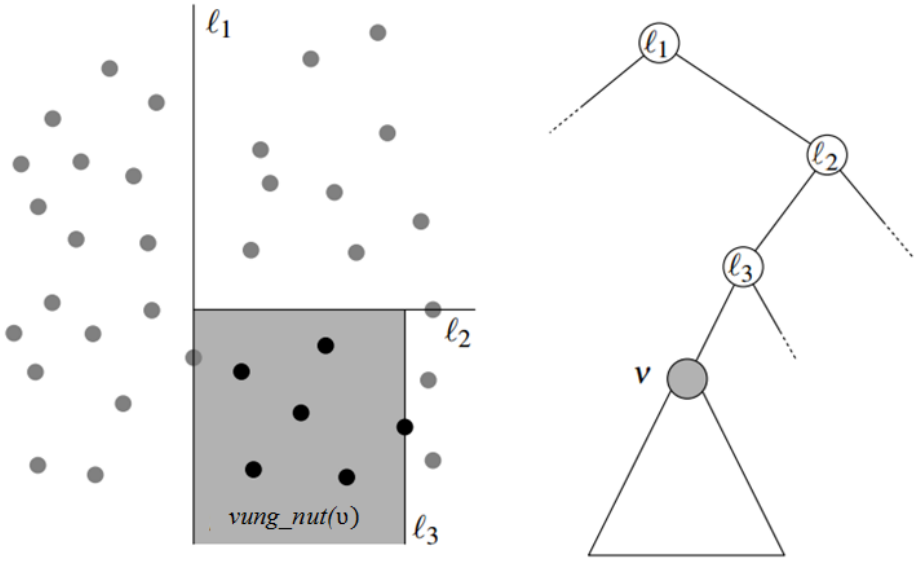
Các bước:

1. **IF** P chỉ chứa một điểm
2. **THEN RETURN** một lá chứa điểm này
3. **ELSE IF**$chieu\_sau$ là chẵn
4. **THEN** Phân tách P thành hai tập con bởi một đường thẳng đứng ℓ đi qua tọa độ trung bình theo x của các điểm trong P. Gọi P_1 là tập các điểm nằm bên trái ℓ, và P_2 là tập các điểm nằm bên phải ℓ.
5. **ELSE** Phân tách P thành hai tập con bởi một đường thẳng nằm ngang ℓ đi qua tọa độ trung bình theo y của các điểm trong P. Gọi P_1 là tập các điểm nằm phía dưới ℓ, và P_2 là tập các điểm nằm phía trên ℓ.
6. $\upsilon_{\text{trái}} \leftarrow$ Xay_dung_cay_KD($P_1, chieu\_sau + 1$)
7. $\upsilon_{\text{phải}} \leftarrow$ Xay_dung_cay_KD($P_2, chieu\_sau + 1$)
8. Tạo một nút υ lưu trữ ℓ, $\upsilon_{\text{trái}}$ là nút con bên trái của υ và $\upsilon_{\text{phải}}$ là nút con bên phải của υ.
9. **RETURN** υ

2.1.2. Xây dựng thuật toán tìm kiếm trên cây KD

Đường thẳng phân tách lưu trữ tại gốc chia mặt phẳng thành hai nửa mặt phẳng. Các điểm nằm nửa mặt phẳng bên trái được lưu trữ trong cây con bên trái, còn các điểm nằm trong nửa mặt phẳng bên phải được lưu trữ trong cây con bên phải. Như vậy, nút con trái của gốc tương ứng với nửa mặt phẳng bên trái và nút con phải của gốc tương ứng với nửa mặt phẳng bên phải. Nói một cách tổng quát vùng mặt phẳng tương ứng với một nút υ là một vùng chữ nhật bị chặn bởi đường phân tách lưu trữ tại các nút tổ tiên của υ được minh họa hình 4.

Ký hiệu vùng mặt phẳng tương ứng với một nút là *vung_nut(*υ). Vùng mặt phẳng của gốc cây KD là toàn bộ mặt phẳng. Sau đây, gọi vùng mặt phẳng tương ứng với một nút là vùng nút của nút đó.



Hình 4. *Vùng mặt phẳng tương ứng với một nút*

Chú ý rằng, một điểm được lưu trữ trong cây con có gốc tại nút υ nếu và chỉ nếu nó nằm trong *vung_nut(*υ). Ví dụ, cây con của nút υ trong hình 4 lưu trữ những điểm được đánh dấu là các chấm đen. Theo đó, chỉ phải tìm kiếm những cây con có gốc tại υ nếu vùng tìm kiếm giao với *vung_nut(*υ). Như vậy các bước thực hiện thuật toán truy vấn sau:

- Cắt cây KD nhưng chỉ duyệt những nút mà vùng nút của nó giao với vùng tìm kiếm.
- Khi một vùng nút nằm hoàn toàn trong vùng tìm kiếm thì liệt kê tất cả các điểm được lưu trữ trong con của nó.
- Khi phần giao nhau có chứa một lá nào đó thì cần phải kiểm tra xem điểm được lưu trữ tại lá đó có được chứa trong vùng tìm kiếm hay không, nếu có thì liệt kê nó.

Thuật toán tìm kiếm được mô tả bởi thủ tục đệ qui dưới đây, với tham số là gốc của cây KD và phạm vi truy vấn R. Nó sử dụng thủ tục con Liet_ke_cay_con(υ) để duyệt qua cây con có gốc tại υ và liệt kê tất cả các điểm lưu trữ tại các lá của nó. Ký hiệu $lc(\upsilon)$ và $rc(\upsilon)$ tương ứng là nút con bên trái và bên phải của nút υ.

Thuật toán TimkiemPhamvi2Chieu_CayKD(υ, R)
Đầu vào: Gốc (một cây con) của một cây KD, và phạm vi R
Đầu ra: Tất cả các điểm tại những lá dưới υ và thuộc phạm vi R
Các dòng lệnh:

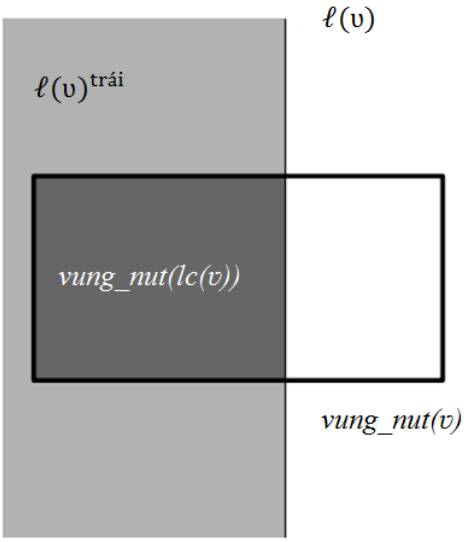
1. **If** υ là một lá
2. **then** Liệt kê điểm chứa tại υ nếu nó thuộc R
3. **else if** *vung_nut*($rc(\upsilon)$)được chứa hoàn toàn trong R
4. **then** Liet_ke_cay_con($lc(\upsilon)$)
5. **else if** *vung_nut*($lc(\upsilon)$)giao với R
6. **then**TimkiemPhamvi2Chieu_CayKD($lc(\upsilon)$,R)
7. **If** *vung_nut*($rc(\upsilon)$) được chứa hoàn toàn trong R
8. **then** Liet_ke_cay_con($rc(\upsilon)$)
9. **else if** *vung_nut*($rc(\upsilon)$)giao với R
10. **then**TimkiemPhamvi2Chieu_CayKD($rc(\upsilon)$, R)

Bước kiểm tra chính trong thuật toán này là kiểm tra xem vùng tìm kiếm có giao với vùng nút của một nút υ nào đó hay không. Để thực hiện bước kiểm tra này, có thể tính toán *vung_nut*(υ) cho tất cả các nút υ ở giai đoạn tiền xử lý và lưu trữ nó, nhưng điều này là không cần thiết vì có thể xác định vùng hiện hành thông qua các lời gọi đệ qui bằng cách sử dụng những đường thẳng

phân tách lưu trữ tại những nút trong. Chẳng hạn như vùng tương ứng với nút con trái của một nút υ tại chiều sâu chẵn có thể được tính từ *vung_nut*(υ)như sau:

$$vung\_nut(lc(\upsilon)) = vung\_nut(\upsilon) \cap \ell(\upsilon)^{trái} \text{ (hình 5)}$$

với $\ell(\upsilon)$ là đường thẳng phân tách lưu trữ tại υ và $\ell(\upsilon)^{trái}$ là nửa mặt phẳng nằm bên trái của $\ell(\upsilon)$ (bao gồm cả $\ell(\upsilon)$).



Hình 5. *Cách xác định vùng nút hiện hành*

Chú ý rằng thuật toán tìm kiếm ở trên không giả định rằng vùng tìm kiếm có dạng hình chữ nhật, cho nên nó làm việc tốt với mọi dạng phạm vi tìm kiếm.

2.2. Giải pháp cải tiến

Trong phần trước, khi tìm kiếm trên phạm vi hai chiều giả thiết rằng không có bất kỳ hai điểm nào có cùng tọa độ *x* hay *y*, đây là một giả thiết phi thực tế. Ví dụ, khi các điểm lưu trữ thông tin về mức thu nhập và số con của nhân viên thì không thể tránh khỏi việc có một số nhân viên có những thông tin trùng nhau. Để giải quyết vấn đề này, nghiên cứu đề xuất giải pháp tập điểm chung để cải tiến thuật toán tìm *kiếm trên cây KD*, được thực hiện như sau: thay thế các tọa độ số thực bởi các phần tử của không gian hợp số. Các phần tử của không gian này là các cặp số thực. Hợp số của hai số thực a và b được ký hiệu là (a/b). Định nghĩa thứ tự trong không gian hợp số như sau: với hai hợp số (a/b) và (a'/b'):

$$(a/b) < (a'/b') \Leftrightarrow a < a' \text{ hoặc } (a = a' \text{ và } b < b').$$

Giả thiết *P* là tập *n* điểm đã cho trong mặt phẳng. Các điểm riêng biệt, nhưng nhiều điểm có thể có cùng tọa độ *x* hoặc *y*. Thay thế mỗi điểm $p := (p_x, p_y)$ bởi một điểm mới $\hat{p} := ((p_x|p_y), (p_y|p_x))$ sao cho có các hợp số như là các giá trị tọa độ. Bây giờ dựa vào tập $\hat{P}$ hãy xây dựng cây KD.

Giả sử như muốn liệt kê các điểm của *P* mà nằm trong một phạm vi $R := [x: x'] \times [y: y']$. Để thực hiện điều này thì phải tìm kiếm trên cây đã xây dựng cho $\hat{P}$. Có nghĩa rằng phải thực hiện việc biến đổi phạm vi tìm kiếm sang không gian hợp số mới. Phạm vi biến đổi $\hat{R}$ được định nghĩa như sau:

$$\hat{R} := [(x|-\infty): (x'|+\infty)] \times [(y|-\infty): (y'|+\infty)].$$

3. Kết quả thực nghiệm

Công cụ lập trình được lựa chọn trong nghiên cứu này là Microsoft Visual Studio 2015.

Đây là chương trình thực nghiệm tính hiệu quả của thuật toán tìm kiếm sử dụng cấu trúc dữ

liệu cây KD khi chưa áp dụng phương pháp cải tiến thuật toán và khi áp dụng cải tiến thuật toán bằng phương pháp tập điểm chung để khắc phục tình huống các điểm trong tập điểm đầu vào có cùng tọa độ x hoặc tọa độ y.

Dữ liệu vào là một tập P chứa các điểm trong mặt phẳng. Tọa độ x hay tọa độ y của mỗi điểm trong tập P là một số thực ngẫu nhiên trong phạm vi từ 0 tới 1000.

Nhiệm vụ của chương trình là liệt kê tất cả các điểm thuộc tập P mà nằm trong phạm vi tìm kiếm $[x{:}\,x'] \times [y{:}\,y']$ nào đó được người sử dụng nhập vào thông qua giao diện người dùng.

Thử nghiệm với phạm vi tìm kiếm $[0{:}\,100] \times [0{:}\,100]$ và $[0{:}\,200] \times [0{:}\,200]$ kết quả thu được như mô tả trong bảng 1.

Bảng 1. *Kết quả thuật toán tìm kiếm trên phạm vi 2 chiều – Cây KD*

Phạm vi tìm kiếm	Số điểm có cùng tọa độ của tập P	Số điểm nằm trong tập P khi chưa cải tiến TT	Số điểm nằm trong tập P khi cải tiến TT
$[0{:}\,100] \times [0{:}\,100]$	5	26	27
$[0{:}\,200] \times [0{:}\,200]$	9	39	45

Đối với thuật toán Tìm kiếm phạm vi 2 chiều – cây KD khi chưa áp dụng phương pháp tập điểm chung sẽ không liệt kê được những điểm mà tọa độ x hoặc y trùng nhau của tập P, sau khi áp dụng phương pháp tập điểm chung thuật toán sẽ cho kết quả chính xác ngay cả khi tập P có điểm trùng nhau.

Qua thực nghiệm, thuật toán đề xuất cho kết quả tìm kiếm chính xác và khắc phục tình huống các điểm trong tập điểm đầu vào có cùng tọa độ x hoặc tọa độ y, đồng thời tốc độ tìm kiếm của thuật toán tỉ lệ thuận với số điểm được liệt kê.

4. Kết luận

Thuật toán tìm kiếm phạm vi hai chiều được đề xuất dựa vào các kỹ thuật hình học tính toán. Nghiên cứu cũng đề xuất phương pháp tối ưu thuật toán tìm kiếm phạm vi hai chiều – cây KD bằng tập điểm chung để giải quyết trường hợp suy biến cho các điểm có cùng tọa độ và xây dựng chương trình trực quan bằng thực nghiệm. Hướng nghiên cứu tiếp theo của bài báo sẽ tiếp tục tập trung vào các phương pháp tìm kiếm trên miền trực giao trên cơ sở dữ liệu tương ứng được trình bày trong nghiên cứu này để xây dựng một cơ sở dữ liệu mới đáp ứng nhu cầu ngày càng cao về tốc độ trong truy vấn hiện nay.

TÀI LIỆU THAM KHẢO/ REFERENCES

[1] M. McKenney, R. Frye, M. Dellamano, K. Anderson, and J. Harris, "Multi-core parallelism for plane sweep algorithms as a foundation for GIS operations," *GeoInformatica*, vol. 21, no. 1, pp. 151–174, 2017.

[2] H. Da, N. Alechins, M. Jackson, and Glen Hart, "A Method for Maching Caned arced and Authoritative Geospatial Data," *Transactions in GIS*, vol. 21, no. 2, 2017, Art. no. 406427.

[3] A. Eldawy, Y. Li, M. F. Mokbel, and R. Janardan, "CG_Hadoop: computational Geometry in mapreduce," in *Proceedings of the international conference on advances in geographic information systems*, ACM SIGSPATIAL, 2013, pp. 284–293.

[4] H. K. Chen and W. S. Chen, "GPU-accelerated blind and robust 3D mesh watermarking by geometry image," *Int. J. Multimedia Tools Appl.*, vol. 75, no. 16, pp. 10077–10096, 2016.

[5] K. G. Larsen and R. Williams, "Faster online matrix-vector multiplication," in *Proceedings of the 28th ACM–SIAM Symposium on Discrete Algorithms (SODA'17)*, SIAM, Philadelphia, 2017, pp. 2182–2189.

[6] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry*, Springer, 2008.

[7] M. A. Mahmood, "A Proposed Hybrid Spatial Data Structure based on KD Tree and Quad Tree," *Jokull*, vol. 69, no. 6, pp. 2-6, 2019.

[8] M. Otair, "Approximate K-Nearest Neighbour Based Spatial Clustering Using K-D Tree," *International Journal of Database Management Systems (IJDMS)*, vol. 5, no. 1, pp. 97-108, 2013.

[9] T. M. Chan and Z. Huang, "Dynamic Colored Orthogonal Range Searching," *Schloss Dagstuhl-Leibniz-Zentrum Informatik*, vol. 204, pp. 1-13, 2021.

[10] T. M. Chan, "Orthogonal Range Searching in Moderate Dimensions: k-d Trees and Range Trees Strike Back," *Discrete & Computational Geometry*, vol. 61, pp. 899–922, 2019.

[11] J. E. Goodman, J. O'Rourke, and C. D. Tóth, *Handbook of Discrete and Computational Geometry*, CRC Press LLC, 2017.

[12] M. Skrodzki, *The k-d tree data structure and a proof for neighborhood computation in expected logarithmic time*, Cornell University, 2019, doi: 10.48550/arXiv.1903.04936.