ĐÁNH GIÁ VIỆC HỌC CÁC KHÁI NIỆM HÌNH HỌC

ĐỂ TẠO RA CÁC MIỀN TRÍ CH DẪN TRONG

PHÂN TÍ CH CHƯƠNG TRÌNH TĨNH

qua

Patrick Chadbourne

Một luận án

nộp một phần để hoàn thành

của các yêu cầu về mức độ

Thạc sĩ Khoa học Máy tính

Đại học Boise State

Tháng 8 năm 2023

TRƯỜNG CAO ĐẲNG ĐẠI HỌC BANG BOISE

ỦY BAN QUỐC PHÒNG VÀ PHÊ DUYỆT ĐỌC CUỐI CÙNG

của luận án được nộp bởi

Patrick Chadbourne

Tiêu đề luận án: Đánh giá việc học các khái niệm hình học để tạo ra các miền trừu tượng vị ngữ trong phân tích chương trình tĩnh

Ngày thi vấn đáp cuối kỳ: 12 tháng 6 năm 2023

Những cá nhân sau đây đã đọc và thảo luận luận án do sinh viên Patrick nộp Chadbourne, và họ đã đánh giá bài thuyết trình và phản hồi các câu hỏi trong kỳ thi vấn đáp cuối cùng. Họ thấy rằng sinh viên đã vượt qua kỳ thi vấn đáp cuối cùng.

Elena Sherman, Tiến sĩ	Chủ tịch, Ủy ban giám sát
Tiến sĩ Jim Buffenbarger	Thành viên, Ủy ban giám sát
Tiến sĩ Steven Cutchin	Thành viên, Ủy ban giám sát

Sự chấp thuận đọc cuối cùng của luận án đã được Elena Sherman, Tiến sĩ, Chủ tịch chấp thuận của Ủy ban giám sát. Luận án đã được Hội đồng sau đại học chấp thuận.

LỜI CẢM ƠN

Tôi biết ơn rất nhiều người mà nếu không có họ thì luận án này sẽ không được hoàn thành đã hoàn thành. Đứng đầu trong số này là cố vấn của tôi, Tiến sĩ Elena Sherman, vì sự tận tụy vô tận của cô ấy quyết tâm giúp tôi đi đúng hướng và tiến về phía trước, và mẹ tôi Laurie St. Amand là người đóng vai trò quan trọng không kém trong việc lắng nghe một cách chu đáo trong khi tôi đã giải thích công việc của tôi qua nhiều cuộc điện thoại.

Tôi cũng biết ơn các thành viên trong ủy ban của tôi, Tiến sĩ Jim Buffenbarger và Tiến sĩ Steven Cutchin, đã cung cấp những hiểu biết và phản hồi có giá trị trong quá trình hình thành các giai đoạn của luận án này.

Luận án này được thực hiện nhờ sự hỗ trợ tài chính của Boise State CS và Quỹ Khoa học Quốc gia Hoa Kỳ theo giải thưởng CCF-19-42044.

TÓM TẮT

Độ chính xác của phân tích tĩnh trên các miền trừu tượng của thuật ngữ phụ thuộc vào phân vùng của các vị ngữ. Các vị ngữ chính xác hơn xấp xỉ các giá trị cụ thể của các biến chương trình dẫn đến phân tích chính xác hơn. Lý luận thủ công về những điều này việc phân vùng phải được thực hiện theo từng trường hợp cụ thể, tốn thời gian và khó khăn.

Công trình này khám phá việc học các khái niệm hình học để tự động khám phá các vị từ ứng viên tên miền.

Khung đề xuất sử dụng dữ liệu thời gian chạy từ các lần thực thi chương trình để thu thập dữ liệu đào tạo cho người học PAC để tạo ra các siêu mặt phẳng tách biệt có thể được chiếu lên miền vị ngữ.

Luận án triển khai khung và thực hiện đánh giá hiệu quả của nó trên một tập hợp các chương trình chuẩn sử dụng nhiều công cụ tạo trường hợp thử nghiệm khác nhau.

Công trình nghiên cứu này thảo luận về một số thiếu sót trong tình trạng hiện tại của nghệ thuật trong việc tạo ra trường hợp thử nghiệm, biểu diễn chương trình trung gian và tính khả dụng của chuẩn mực chương trình phù hợp.

MỤC LỤC

DANH SÁCH CÁC BẢNG

DANH SÁCH CÁC HÌNH ẢNH

CHƯƠNG 1

GIỚI THIỆU

Mục tiêu của luận án này là đóng góp vào sự phát triển của các miền vị ngữ trong phân tích tĩnh (SA), một phương pháp kiểm tra mã mà không thực thi nó. SA sử dụng các miền vị ngữ rời rạc có lợi thế về tốc độ nhưng đòi hỏi phải được xác định trước miền để thực hiện phân tích. Một miền được tạo ngẫu nhiên là một tùy chọn có thể có thể được thực hiện một cách dễ dàng, mặc dù nó thường dẫn đến một phân tích không chính xác do bản chất rộng và không cụ thể của nó. Ngược lại, việc sử dụng một miền có thể mang lại mức độ chính xác cao hơn trong phân tích, vì nó có thể được điều chỉnh để các yêu cầu cụ thể của chương trình. Tuy nhiên, cách tiếp cận này đòi hỏi một sự hiểu biết về chương trình của người tạo miền, do đó làm tăng thêm sự phức tạp cho quá trình. Việc tạo thủ công tên miền, mặc dù có lợi cho độ chính xác, nhưng lại mất thời gian bước tiêu tốn và chủ quan có khả năng làm chậm quá trình phân tích. Điều này công việc khám phá một cách tiếp cận tự động tạo ra các miền trừu tượng của vị ngữ sử dụng dữ liệu từ việc thực thi chương trình.

1.1 Bối cảnh vấn đề

Phân tích tĩnh cho phép kỹ sư phần mềm suy luận về tất cả các khả năng thực thi của một chương trình mà không thực hiện nó trên mọi đầu vào có thể. SA lý do về ngữ nghĩa của chương trình bằng cách phân tích cấu trúc của mã. Trong so sánh

```
1 :func(int n){ //n = {  , 0, +}
2 :if(n < 2){ //n = {  , 0, +}
3 :n = 1/(n-2); //Cảnh báo về khả năng Chia cho số 0
4 :print(n);
5 :} else { //n = {+}
6 :print(n);}
```

Hình 1.1: Ví dụ mã được chú thích bằng miền dấu.

với thử nghiệm truyền thống, SA đảm bảo tính hợp lệ của kết quả. Trong khi thời gian chạy kiểm tra xác nhận sự tồn tại của hành vi lỗi sau khi phát hiện ra nó, SA đảm bảo không có hành vi không mong muốn nào xảy ra đối với tất cả các lần thực thi chương trình khi không phát hiện lỗi.

Sức mạnh này phát sinh từ khả năng của SA trong việc lý luận các yếu tố trừu tượng miền trái ngược với các giá trị cụ thể. Một phần tử của miền trừu tượng biểu diễn một tập hợp hữu hạn hoặc vô hạn các giá trị cụ thể. Bằng cách thay thế các giá trị có thể vô hạn trạng thái thực hiện chương trình với số lượng trạng thái hữu hạn trên một miền trừu tượng một vấn đề được đơn giản hóa. SA sử dụng các giá trị trừu tượng này thay cho các giá trị cụ thể, thực hiện phân tích trên tất cả các đường dẫn thực thi có thể bằng cách kiểm tra các giá trị trừu tượng đại diện cho chúng. SA sau đó có thể xác định xem một số thuộc tính có tồn tại hay không các khẳng định giữ nguyên cho các yếu tố trừu tượng. Các thuộc tính này sau đó có thể tồn tại trong các giá trị cụ thể có thể được biểu diễn bằng các yếu tố trừu tượng.

Mặc dù có nhiều lĩnh vực trừu tượng khác nhau tồn tại, công trình này đặc biệt sử dụng của các miền vị ngữ do hiệu quả của chúng [13]. Để tạo các phần tử của một vị ngữ miền, các giá trị cụ thể được phân chia thành các tập hợp con được xác định bởi các ràng buộc trên cụ thể giá trị. Ví dụ, phân vùng tập hợp tất cả các số nguyên thành các tập hợp con của các số nguyên nhỏ hơn, bằng và lớn hơn không tạo ra miền trừu tượng về dấu hiệu: { , 0, +}.
Mặc dù miền vị ngữ cung cấp hiệu quả lớn hơn nhiều bằng cách giới hạn số lượng các yếu tố phải được phân tích, độ chính xác của phân tích bị giới hạn bởi độ chính xác của các phân vùng được sử dụng.

```
1 :func(int n){ //n = {< 2, 2, > 2}
2 :if(n < 2){ //n = {< 2}
3 :n = 1/(n-2); //Không có cảnh báo ước lượng quá
mức
4 :print(n); 5 :} else { //
n = {2, > 2} 6 :print(n);}
```

Hình 1.2: Ví dụ mã được chú thích bằng miền Vị ngữ với phân vùng được chọn thủ công tại 2.

Để chứng minh những tác động đến độ chính xác của SA mà các phân vùng khác nhau mang lại, chúng ta hãy kiểm tra hai đoạn mã giống hệt nhau với một phân vùng ngây thơ và một phân vùng thủ công. Trong Hình 1.1 chúng ta thấy mã đơn giản với các chú thích nêu các giá trị trừu tượng có thể có của n. Khi bắt đầu func SA không biết gì về giá trị n đến và chú thích nó với các giá trị trừu tượng âm, dương và số không. Trên dòng 2 SA phân tích giá trị thực nhánh và xác định rằng n phải nhỏ hơn 2. Tuy nhiên vì phân vùng không thể diễn đạt chính xác, SA xấp xỉ nó một cách hợp lý với các giá trị dương, âm và bằng không một lần nữa. Ở dòng 3 SA đưa ra cảnh báo tiềm ẩn rằng mã có phép chia cho số không thất bại. Điều này là do SA bao gồm n = 2 như một giá trị có thể vì giá trị cụ thể 2 tại đó lỗi xảy ra được biểu thị bằng giá trị trừu tượng dương {+}.

Sự xấp xỉ quá mức này gây ra sự không chính xác trong SA có thể được giảm thiểu bằng cách sử dụng phân vùng chính xác hơn cho mã này. Ví dụ, trong Hình 1.2 các giá trị trừu tượng là dựa trên mối quan hệ của chúng với 2. Bằng cách thay đổi các phân vùng được sử dụng trong phân tích, giá trị trừu tượng của nhánh thực trên dòng 2 hiện đại diện chính xác cho tất cả các khả năng các giá trị của n nhỏ hơn 2. Vì các giá trị nhỏ hơn hai tránh được lỗi chia cho số không, SA xóa cảnh báo dương tính giả.

Trong ví dụ mã này, việc xác định lựa chọn phân vùng thứ hai là rất đơn giản tốt hơn cách đầu tiên, đặc biệt khi một trong các toán hạng quan hệ là hằng số.

Tuy nhiên, đối với các chương trình phức tạp hơn, việc xác định phân vùng tốt nhất theo cách thủ công là một thách thức.

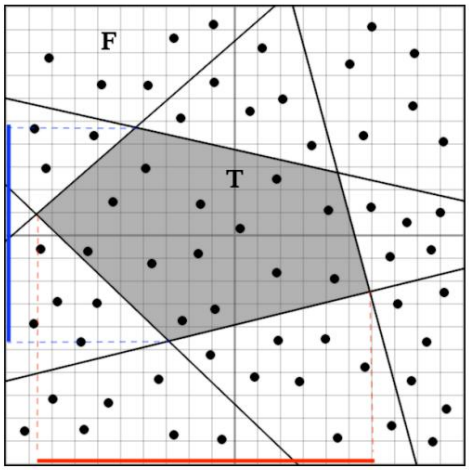
dài, đặc biệt là khi cả hai toán hạng đều là biến. Công trình này tìm kiếm một
phương pháp hiệu quả để tạo ra các phân vùng chính xác một cách tự động mà không cần phải có trước
kiến thức về chương trình hoặc kiểm tra mã thủ công.

Mặc dù có nhiều cách để xác định một cách ngây thơ các yếu tố vị ngữ như
khi tìm kiếm các hằng số mã, chúng thường bị hạn chế về độ chính xác khi xử lý
mối quan hệ giữa nhiều biến. Công trình này cố gắng suy ra các phân vùng như vậy
bằng cách phân tích các giá trị cụ thể được gán cho các biến trong quá trình thực thi chương trình.
Sử dụng chúng như tọa độ hình học trong một không gian mà mỗi chiều tương ứng
đối với một biến, kỹ thuật được đề xuất cố gắng tạo ra các phân vùng bằng cách học
khái niệm hình học: hình dạng trừu tượng hoặc vùng bao gồm các điểm.

1.2 Các khái niệm hình học

Các khái niệm hình học có thể biểu diễn một trạng thái trừu tượng, ví dụ, n {+} bằng cách ánh xạ tập hợp của nó
của các giá trị cụ thể (ví dụ, dương) vào một vùng trên trục n cho biến đó. Đối với kỳ thi-
Ví dụ, phân tích hai biến dẫn đến các vùng trong không gian hai chiều. Tăng
số lượng các biến được phân tích cũng làm tăng tính đa chiều của
không gian chứa các vùng. Công trình trước đây [3] đã phát triển các thuật toán học
hình dạng của các điểm dữ liệu. Theo cùng một lộ trình, chúng ta có thể xác định các phân vùng phù hợp
mà SA có thể sử dụng để tính toán các giá trị trừu tượng.

Để chứng minh khái niệm này, hãy xem xét Hình 1.3 cho thấy một hình ảnh hai chiều
không gian trong đó các giá trị cụ thể của biến u và w được biểu diễn trên tọa độ Descartes.
Các điểm dữ liệu này được dán nhãn là "đúng" hoặc "sai" tùy thuộc vào tương ứng-
nhánh ing được lấy tại một câu lệnh có điều kiện như u < w. Do đó tất cả các điểm dữ liệu trong
vùng tô bóng được gắn nhãn là đúng dẫn đến kết quả trong chương trình thực hiện đúng



Hình 1.3: Ví dụ về khái niệm hình học được sử dụng để phân vùng các giá trị cụ thể trong không gian hai chiều.

nhánh của câu lệnh điều kiện. Hình dạng của vùng tô bóng biểu diễn khái niệm hình học được học bởi thuật toán. Các đường tạo thành hình dạng và giao điểm của chúng tương ứng với các phân vùng trừu tượng quan hệ suy ra. Để tìm các vị từ, vùng tô bóng được chiếu lên trục u và w như thể hiện trong hình ở các đường màu đỏ và màu xanh lam, tương ứng. Sau đó, một miền vị ngữ bao gồm ba vị ngữ, "trước" phần bắt đầu của dòng, phần dòng đó và "sau" phần kết thúc của dòng.

Cách tiếp cận này thu thập các giá trị biến cụ thể dưới dạng các điểm dữ liệu thông qua công cụ thực hiện chương trình và chuyển chúng vào một thuật toán học một hình học khái niệm đại diện cho dữ liệu đó. Thuật toán sử dụng một Có lẽ-Xấp xỉ-Người học (PAC) chính xác [15], đòi hỏi một mẫu dữ liệu đủ lớn để tính toán chính xác hình học đại diện. Sau khi tạo ra, vùng kết quả sau đó được chiếu lên trục để tạo ra một miền vị ngữ với các phân vùng đó tách biệt chính xác các giá trị trong các câu lệnh có điều kiện.

1.3 Giải pháp đề xuất

Để cải thiện hiệu quả của việc tạo miền để sử dụng trong SA, chúng tôi kiểm tra một kỹ thuật mới để sử dụng thuật toán học tập PAC và phép chiếu không gian kép để học các khái niệm hình học có thể được diễn giải để tạo ra một miền vị ngữ bằng cách sử dụng dữ liệu đầu vào được lấy mẫu từ các lần thực hiện chương trình. Thông qua việc phát triển và đánh giá Với cách tiếp cận này, chúng tôi muốn xác thực luận điểm sau:

1.3.1 Luận đề

Thuật toán sử dụng các kỹ thuật học PAC có thể cung cấp dự đoán chính xác như thế nào? các miền trừu tượng để phân tích tĩnh bằng cách chiếu các khái niệm hình học 2D lên 1D miền trừu tượng.

Như một phương tiện để hỗ trợ cho luận điểm, các câu hỏi nghiên cứu sau đây sẽ được điều tra và trả lời:

1.3.2 Câu hỏi nghiên cứu

RQ1 Làm thế nào để thu thập dữ liệu hiệu quả từ các chương trình để cung cấp mẫu cho Người học PAC?

RQ2 Kích thước mẫu ảnh hưởng như thế nào đến đầu ra của thuật toán học PAC?

RQ3 Thuật toán học PAC có thể tạo ra các miền trừu tượng của vị từ trên một tập hợp không? của các chương trình chuẩn mực?

Luận án được tổ chức như sau: Chương 2 bao gồm bối cảnh có liên quan về các kỹ thuật được sử dụng trong công việc này và cách chúng được sử dụng trong nghiên cứu hiện có, bao gồm các kỹ thuật SA hiện tại và các thuật toán học tập PAC. Chương 3 cung cấp một

tổng quan cấp cao về khuôn khổ tạo miền từ người dùng ban đầu được cung cấp
chương trình đến miền được tạo ra kết quả. Chương 4 xem xét khuôn khổ này
chi tiết hơn, bao gồm cách thức tiến hành thu thập dữ liệu và công cụ lập trình,
cũng như thuật toán học tập PAC không gian kép mới lạ và miền kết quả
tổng hợp. Cuối cùng, Chương 5 kiểm tra khuôn khổ đã hoàn thành và kết quả của nó khi
được sử dụng để tạo ra các miền cho tập hợp đầu vào của các chương trình thử nghiệm của chúng tôi. Các phát hiện của luận án là
được biên soạn trong Chương 6, nơi cung cấp câu trả lời cho các câu hỏi nghiên cứu của chúng tôi và
những mở rộng tiềm năng của công trình này được đề xuất như là chủ đề nghiên cứu trong tương lai.

CHƯƠNG 2

BỐI CẢNH VÀ CÔNG VIỆC LIÊN QUAN

Chương tiếp theo sẽ xem xét lại bối cảnh cần thiết hỗ trợ cho luận án của chúng tôi
trong các chủ đề bao gồm phân tích tĩnh, học PAC và xác minh chương trình. Công việc trước đây
trong các lĩnh vực này cung cấp các phương pháp luận để khẳng định hành vi chương trình thông qua dữ liệu
phân tích dòng chảy, sử dụng các kỹ thuật học máy được hỗ trợ thống kê để tìm hiểu
ranh giới của các vùng phân tách các tập dữ liệu và mở rộng các khái niệm học tập PAC
để tạo điều kiện cho việc suy luận trực tiếp về việc xác minh chương trình.

2.1 Phân tích tĩnh với miền trừu tượng của vị ngữ

Mục tiêu của phân tích chương trình luồng dữ liệu A là tính toán các bất biến của chương trình
được thể hiện như các phần tử của miền trừu tượng D = po D; , , trong đó là phần tử nhỏ nhất
toán tử giới hạn trên. Một trường hợp phân tích A được định nghĩa bởi [7] sau:

Một mạng lưới hoàn chỉnh D (thỏa mãn điều kiện chuỗi tăng dần), mô tả
miền trừu tượng của A.

Luồng F gồm các câu lệnh của chương trình P, ví dụ như đồ thị luồng điều khiển của P.

Một tập hợp các hàm truyền đơn điệu F cho mỗi câu lệnh l F ánh xạ một
phần tử của D với chính nó, tức là, fl F : D D.

Một tập hợp các câu lệnh bên ngoài E trong P, có thể là nút nhập

nút vào (Chương trình) hoặc nút thoát (Chương trình).

Giá trị ban đầu ι D cho các câu lệnh trong E.

Sau đó, tập hợp các phương trình cho A được định nghĩa như sau khi vào và ra khỏi mỗi phát biểu l F:

$$Ain(l) = \{Aout(l') \mid (l', l) \in F\} \quad \iota_{\bar{E}} \tag{2.1}$$

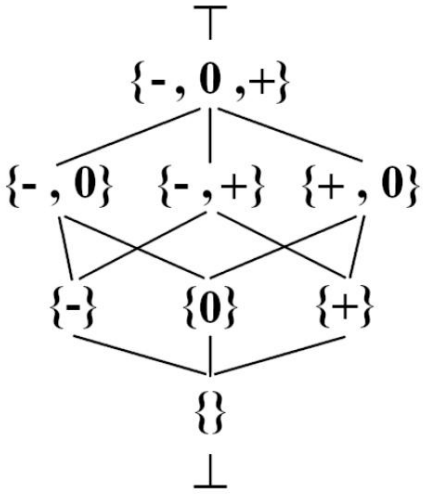
$$l \text{ ở đâu } \iota_E = \begin{cases} \text{nếu } l \in E \\ \text{nếu } l \notin E \end{cases} \tag{2.2}$$

$$Aout(l) = fl(Ain(l)), \; l \in F \tag{2.3}$$

Trong đó là toán tử giới hạn trên nhỏ nhất, là phần tử dưới cùng của D với các tính chất sau d D : d = d và (l) F : fl( ) = . Tập hợp trên của các phương trình được giải quyết bằng cách sử dụng phép tính điểm cố định như thuật toán danh sách công việc lặp đi lặp lại tính toán lại các luồng đến và đi cho mỗi câu lệnh, tức là Ain(l) và Aout(l) cho đến khi phát hiện không có thay đổi nào trong luồng dữ liệu.

Tùy thuộc vào loại phân tích, D có thể có biểu diễn khác nhau. Đối với ví dụ, trong miền trừu tượng DLV của phân tích Biến trực tiếp, lý do về biến chương trình, D là tập hợp lũy thừa của các biến P Var với là tập hợp hợp phép toán, là và là Var. Đối với phân tích dấu, D trong DSA là một tập hợp các bản đồ Var P(Dấu) trong đó như trước Dấu = { , 0, +}. Thông thường, một phần tử của D mà A tính toán cho một vị trí chương trình nhất định được gọi là trạng thái trừu tượng σ D.

Vì trọng tâm của công việc này là tạo ra các phân vùng cho các miền vị ngữ, chúng tôi đưa ra mô tả chi tiết về miền DP đỏ = po D; , . D là tập lũy thừa của



Hình 2.1: Mạng của miền dấu hiệu.

tất cả các vị từ có các phần tử từ p1 đến pk. , toán tử bao hàm, là

định nghĩa thứ tự của mạng. Nếu một phần tử p3 là siêu tập của phần tử p2

sau đó p3 sẽ là bậc cao hơn trên mạng. , toán tử giới hạn trên nhỏ nhất của chúng ta là

thiết lập hoạt động hợp nhất. Nó đảm bảo rằng tất cả các vị từ trong các luồng dữ liệu khác nhau được bảo toàn khi được hợp nhất, nghĩa là nếu các giá trị trong một phạm vi nhất định là hợp lệ trong quá trình thực thi chương trình, sau đó luồng dữ liệu hợp nhất sẽ giữ nguyên các giá trị hợp lệ sau khi hợp nhất.

Các giá trị ban đầu, ι, là tập hợp tất cả các vị từ có sẵn tại mục nhập (Chương trình) vì thế là phân tích vô điều kiện trong đó các giá trị đầu vào không bị hạn chế. Tại các nút không phải là điểm vào giá trị ban đầu của sẽ là tập rỗng. Tập hợp của các giá trị trong Ain(l) có hàm truyền fl được áp dụng cho nó, tạo ra Aout(l). Các hàm chuyển giao này được xác định cho từng loại câu lệnh và phản ánh những thay đổi của các giá trị so với các vị ngữ biểu diễn chúng [13].

2.2 PAC Học các khái niệm hình học

Thuật toán học tập PAC hoạt động theo nguyên tắc với một mẫu đủ lớn
kích thước thuật toán có thể trả về một kết quả chính xác trong phạm vi thống kê
biên độ sai số đáng kể [3]. Người học PAC có khả năng học nhiều hình học khác nhau
các khái niệm bằng cách phân loại dữ liệu thành bên trong hoặc bên ngoài hình dạng hình học từ
mà dữ liệu được thu thập. Baum [1] và Blumer et al. [2] đều trình bày tương tự
phương pháp cho các hợp nhất học tập PAC của các không gian nửa. Blumer et al. [2] cũng cung cấp một
thuật toán cho PAC học hợp các hình chữ nhật thẳng hàng theo trục.

Thuật toán 1 Mã giả cho thuật toán học PAC

```
hàm Learn C (s, ε, δ)
Vẽ mẫu có nhãn S có kích thước
    m = O( 1/ε lg 1/δ + ds/ε (câu ds/ε)^3 )
    Tạo U bằng cách thêm cạnh cho mọi cặp +/- trong S
F
    Tạo một không gian kép hình học chứa một siêu phẳng
      cho mỗi điểm trong S
    Tính toán trọng tâm y của mỗi đặc điểm trong không gian đối ngẫu
F   F   g  (y)
F   bao phủ tham lam U bằng F
    h  các vùng được hình thành bởi F
đối với mỗi vùng ρ  h thì
        nếu ρ ∩ S ≠  thì
            nhãn ρ theo S
        khác
            nhãn ρ tùy ý
        kết thúc nếu
kết thúc cho
    Trở về h
chức năng kết thúc
```

Bshouty et al. [3] đã mở rộng các phương pháp này bằng cách cung cấp một thuật toán cho
PAC học các tổ hợp boolean tùy ý của các không gian nửa. Những hình học chung này
các khái niệm ric được xác định bởi các tổ hợp boolean tùy ý của các khoảng trống nửa cung cấp

tính linh hoạt cần thiết để mô tả các giá trị cụ thể tương đối của các biến trong chương trình thực hiện và do đó có thể được áp dụng cho các vấn đề liên quan đến xác minh chương trình.

Thuật toán 1 trình bày mã giả từ Bshouty et al. [3] để tìm thông tin vùng thông tin từ dữ liệu điểm, mà việc triển khai của chúng tôi dựa trên. Dòng 2 và 3 vẽ mẫu các điểm dữ liệu từ một nguồn cho đến khi thu thập được m điểm. Giá trị của m là được tính toán sao cho có đủ kích thước để trả về kết quả chính xác trong một biên độ không đáng kể về mặt thống kê. Dòng 4 tạo ra một tập dữ liệu cạnh, U, giữa các điểm tích cực và tiêu cực sau đó được sử dụng để xác định thời điểm các điểm đó được tách biệt đúng cách.

Các dòng từ 5 đến 9 điền vào tập hợp các cạnh phân tách. Quá trình này bắt đầu bằng dòng 6 chuyển đổi dữ liệu điểm từ mẫu S của chúng tôi thành các siêu phẳng trong không gian đối ngẫu (quá trình này được giải thích chi tiết hơn trong Chương 3). Trên mặt phẳng kép giao điểm của các siêu phẳng tạo ra các đặc điểm mà từ đó các tâm được tính toán. Dòng 9 chuyển đổi các tâm, là các điểm trong không gian kép, thành một siêu phẳng trên không gian nguyên thủy và thêm chúng vào tập hợp F của các siêu phẳng được tính toán này.

Sau khi tất cả các tâm đã được tính toán và chuyển đổi thành siêu không gian nguyên thủy mặt phẳng, tập hợp U được bao phủ một cách tham lam bởi F. Mỗi siêu phẳng trong F bao phủ một cạnh trong U qua giao điểm được thêm vào F. Trên dòng 11, các siêu phẳng còn lại từ sau đó sử dụng tập hợp tham lam để chia không gian nguyên thủy và dán nhãn kết quả vùng là dương hoặc âm. Vì tất cả các cạnh giữa các điểm dương và âm có các siêu phẳng từ F ngăn cách chúng, các vùng còn lại chỉ có các điểm với nhãn tích cực hoặc tiêu cực. Sau đó ở các dòng 12-18, các vùng này được dán nhãn có cùng nhãn với các điểm chứa bên trong và các vùng trống là được gắn nhãn tùy ý. Thuật toán trả về các vùng này làm kết quả của nó.

2.3 Công việc liên quan đến việc học PAC trong Xác minh

Sharma et al. [11] đã tiên phong sử dụng phương pháp học PAC trong việc xác minh chương trình. Trong làm việc, các nhà nghiên cứu sử dụng Thuật toán 1 để phân biệt giữa các trạng thái có thể đạt được của một program và các trạng thái vi phạm các thuộc tính cụ thể. Cách tiếp cận này tạo ra cảm ứng những khẳng định mô tả những trạng thái "xấu" gây ra vi phạm tài sản. Tuy nhiên, Kỹ thuật này đòi hỏi các bước bổ sung để đảm bảo các bất biến được tạo ra là hợp lệ. Việc tạo ra các bất biến không hợp lệ sẽ khởi động lại quá trình thực thi với dữ liệu bổ sung để cải thiện độ chính xác của thuật toán học PAC. Các tác giả tương tự đã mở rộng khuôn khổ của họ để thực hiện các so sánh định tính về độ chính xác của các miền trừu tượng khác nhau [12].

Một nghiên cứu khác [5, 10] áp dụng thuật toán học PAC cùng với các thuật toán khác để tạo ra các khẳng định tại các điểm cụ thể trong một chương trình mục tiêu. Chúng được tìm thấy bằng cách kết hợp các tính năng của chương trình với các mẫu thiết lập bằng cách sử dụng trình học PAC hoặc Học viên dựa trên SVM. Trong khi cách tiếp cận sử dụng học viên PAC để xác định boolean sự kết hợp của các không gian nửa, nó áp dụng kỹ thuật học tập dựa trên SVM để tìm sự kết hợp của các đẳng thức tuyến tính. Cả hai người học sau đó được sử dụng để tạo ra mong muốn khẳng định. Những khẳng định này có thể được tạo ra tại các điểm cụ thể trong một chương trình bằng chú thích mã tại vị trí khẳng định mong muốn.

Công trình gần đây nhất [9] sử dụng cùng một thuật toán học PAC để tìm chương trình bất biến nhưng trong bối cảnh của các tính năng chương trình khác nhau. Sử dụng dữ liệu được trích xuất từ chương trình mục tiêu trước và sau khi gọi phương thức, kỹ thuật này tạo ra một biểu diễn đồ thị bộ nhớ để tìm hiểu các điều kiện trước và sau của các phương pháp đó. Nghiên cứu này tập trung cụ thể vào các phương pháp trong các chương trình xử lý đống do chúng sự phức tạp. Bằng cách biểu diễn dữ liệu được lưu trữ trong đống dưới dạng đồ thị và kiểm tra cách đồ thị đó được chuyển đổi giữa các điều kiện trước và sau, quá trình học PAC

thuật toán có thể phân loại các hành vi phương pháp. Tương tự như đã mô tả trước đó
làm việc bài báo gần đây này trực tiếp học các bất biến chương trình sau đó có thể được sử dụng cho
xác minh chương trình như một oracle cho hành vi mong đợi. Tạo ra bất biến trực tiếp
có những nhược điểm giống như trong nghiên cứu trên, trong đó tính hợp lý không được đảm bảo
bằng dữ liệu động.

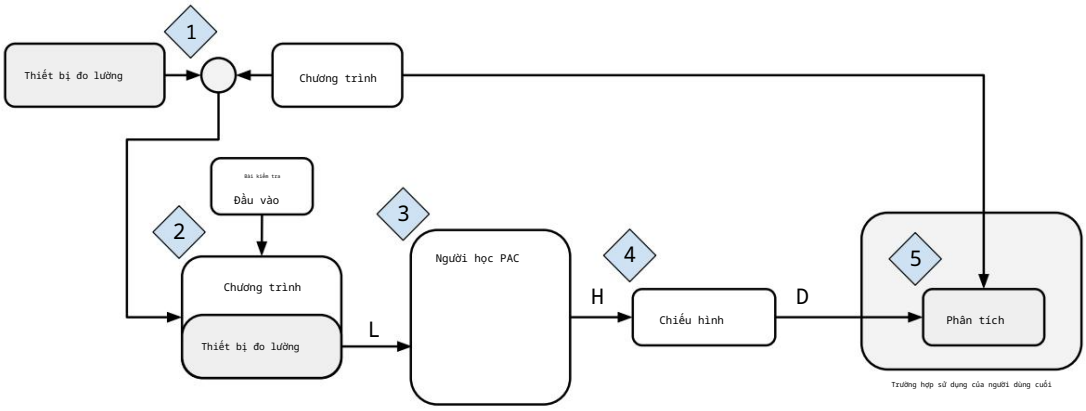
Công trình chúng tôi đề xuất khác với công trình liên quan trước đây theo cách tiếp cận này
học các tính năng khác nhau của chương trình mục tiêu hoặc bằng cách học trực tiếp các chương trình bất biến
kiến. Mặc dù vẫn sử dụng cùng một thuật toán học PAC, việc thu thập dữ liệu
và các tính năng học được kết quả khác nhau. Công việc của chúng tôi tập trung vào việc thu được các bất biến tại
các câu lệnh có điều kiện, tức là các vị từ sử dụng Thuật toán 1, lấy các giá trị cụ thể
của các biến tại các câu lệnh có điều kiện cũng như các nhãn dựa trên kết quả
của việc đánh giá biểu thức có điều kiện. Dữ liệu này sau đó được sử dụng để tìm các vùng
có thể được chiếu lên các vị từ tiềm năng sau đó có thể tạo ra các bất biến sau
thực tế thông qua các kỹ thuật phân tích tĩnh sử dụng các vị ngữ đó làm trừu tượng của chúng
miền. Chương 3 mô tả phương pháp luận và chi tiết triển khai.

CHƯƠNG 3

TIẾP CẬN

Hình 3.1 trực quan hóa tổng quan cấp cao về việc tạo ra các bất biến từ đầu vào chương trình. Khung này theo một đường dẫn tuần tự với đầu ra từ trước các thành phần được hiển thị theo các mũi tên tới các thành phần tiếp theo. quá trình này đòi hỏi hai thông tin riêng biệt để bắt đầu: chương trình dưới phân tích, được hiển thị trong các hộp có nhãn 'Chương trình' và một bộ JUnitTests, trong hộp được gắn nhãn 'T'. Kết quả cuối cùng của quá trình này là phân tích tĩnh được thực hiện bằng cách sử dụng miền được tạo ra, hiển thị trong hộp có nhãn 'Phân tích'. Trong trường hợp sử dụng thực tế này phân tích sẽ phản ánh nhu cầu của người dùng thực hiện nó, tuy nhiên cho mục đích đánh giá công trình này, phân tích được giới hạn trong việc so sánh độ chính xác giữa miền mới và một miền ngẫu nhiên đã tồn tại trước đó.

Ở Bước 1 trong hình, chương trình đầu vào được thiết kế sao cho dữ liệu được gắn nhãn được thu thập trong quá trình thực hiện các câu lệnh nhánh. Mã nguồn Java của chương trình mã được cung cấp làm đầu vào cho một chương trình đo lường sử dụng SOOT Java Khung [16]. Chương trình đo lường này lặp lại trên tất cả các phương pháp trong chương trình đầu vào và trên tất cả các câu lệnh có điều kiện trong các phương pháp. công cụ thay đổi các câu lệnh có điều kiện để các giá trị được lưu trữ trong các biến được tham chiếu bởi điều kiện được lưu trong một lớp báo cáo bên ngoài. Biến dữ liệu được ghép nối với các nhãn tích cực và tiêu cực biểu thị liệu điều kiện



Hình 3.1: Tổng quan cấp cao về phương pháp tạo bất biến.

được đánh giá là đúng hay sai. Chương trình đo lường cũng tìm ra những điểm mà
mã thoát khỏi phương thức chính và chèn thêm một lệnh để viết
thu thập dữ liệu được gắn nhãn vào một tệp đầu ra, nơi dữ liệu này có thể được sử dụng sau này.

Với chương trình đầu vào được biên dịch bằng mã được tích hợp, nó tiếp tục
Bước 2 nơi bộ kiểm thử được cung cấp thực thi mã được đo lường. Mặc dù
bất kỳ thực thi nào của mã được đo lường đều cung cấp dữ liệu đầu ra, các thử nghiệm JUnit được sử dụng
để thuận tiện cho họ. Một bài kiểm tra tham số thực hiện các phương pháp được đo lường
trên một phạm vi rộng các giá trị đầu vào ngẫu nhiên cho phép một lượng lớn dữ liệu được
được thu thập nhanh chóng mà không yêu cầu người dùng phải thực thi mã thủ công. Mỗi
chạy liên tiếp bộ kiểm tra cung cấp nhiều dữ liệu mẫu hơn được thêm vào
tệp đầu ra của thiết bị đo lường. Dữ liệu mẫu được gắn nhãn này, 'L', sau đó đã sẵn sàng để sử dụng
ở Bước 3.

Thuật toán học PAC lấy dữ liệu mẫu được gắn nhãn làm đầu vào và
tạo thành một không gian kép để hình thành các khái niệm hình học nhằm nắm bắt hành vi
của mã được thiết bị. Quá trình này được trình bày chi tiết hơn nhiều trong Phần 4.3.
Các khái niệm hình học là tập hợp các siêu phẳng tạo thành các vùng có nhãn tách biệt

dữ liệu đầu vào. Vì các siêu phẳng này có mật độ thông tin dày đặc hơn nhiều so với
các phân vùng được sử dụng trong miền vị ngữ phải được chiếu vào các vị ngữ.

Bước 4 lấy tập hợp các siêu phẳng, 'H', và tính toán các vị từ tương ứng
phân vùng mà đầu ra cuối cùng của miền, D, có thể được xây dựng. Như
một siêu phẳng biểu diễn một đường thẳng hai chiều và các phân vùng hoạt động như tương ứng
các điểm một chiều, chúng phải được đơn giản hóa đủ để có thể sử dụng làm đầu ra.
Các siêu phẳng có độ dốc lớn, các đường gần như thẳng đứng, được sử dụng
để tìm các phân vùng trong biến tương ứng với trục Y của khái niệm hình học.
Ngược lại, các siêu phẳng có độ dốc gần bằng không, các đường gần như nằm ngang, sau đó
được sử dụng để xác định các phân vùng trên biến tương ứng của trục X. Siêu mặt phẳng
rất gần với độ dốc 1 hoặc -1, những thứ biểu diễn các đường chéo, không phải là
được sử dụng như các phân vùng vì không có cách khả thi nào để nắm bắt thông tin một cách đầy đủ
chúng đại diện. Những đường chéo này thay vào đó được sử dụng để xác định tính hữu ích của
miền vị ngữ để thực hiện phân tích trên chương trình đầu vào. Siêu phẳng chéo
rằng việc tách thành công phần lớn dữ liệu mẫu được dán nhãn cho thấy rằng
mối quan hệ giữa hai biến hoàn toàn là quan hệ, điều này nằm ngoài
khả năng phân tích sử dụng các miền vị ngữ. Đối với các chương trình này, một cách tiếp cận khác
là cần thiết để đạt được phân tích chính xác hơn.

Nếu Bước 4 được hoàn thành với một miền vị từ được tạo thành công thì
người dùng có thể sử dụng miền đó để phân tích trong Bước 5. Cho mục đích
luận án này, các miền được tạo ra được sử dụng để tìm các bất biến chương trình sau đó
so sánh với các bất biến được tìm thấy bởi một miền đã tồn tại được chọn ngẫu nhiên.
Nếu miền được tạo ra dẫn đến các bất biến chính xác hơn cho chương trình theo
phân tích hơn là miền được chọn ngẫu nhiên thì quá trình này được coi là thành công.
Ngược lại nếu các bất biến ít chính xác hơn thì đó là một thất bại. Các bất biến cũng có thể

có độ chính xác như nhau hoặc không thể so sánh được, và đối với cả hai tình huống này, kết quả là mơ hồ, với khuôn khổ được trình bày không đưa ra lợi ích rõ ràng.

CHƯƠNG 4

THỰC HIỆN

Để hiện thực hóa khuôn khổ trên, từng thành phần riêng lẻ được triển khai riêng biệt như một chương trình Java độc lập. Công cụ chương trình được xử lý bằng cách sử dụng khuôn khổ SOOT Java, các bài kiểm tra jUnit được tạo và chạy để thu thập mẫu dữ liệu cho người học PAC và thuật toán học PAC là một chương trình Java lấy dữ liệu mẫu đó làm đầu vào và đưa ra các khái niệm hình học đã học. Như không cần thêm bất kỳ đầu vào bên ngoài nào để đưa các khái niệm hình học vào các miền vị ngữ diễn ra đồng thời, sử dụng dữ liệu có sẵn cho Thuật toán học tập PAC để hỗ trợ cho việc chiếu. Với tất cả các bước được thực hiện và thực hiện kết quả cuối cùng là một thông báo giải thích rằng không gian đã học là có khả năng quan hệ và không thể có miền vị ngữ được tạo cho nó hoặc miền được tạo miền vị ngữ chính nó. Các phần sau đây cung cấp thông tin chi tiết về việc triển khai của từng thành phần.

4.1 Công cụ chương trình

Trước khi tạo miền bằng thuật toán học PAC của chúng tôi, cần phải thu thập dữ liệu mẫu để học hỏi. Dữ liệu được thu thập thông qua thiết bị đo lường, chèn thêm các câu lệnh vào mã đã biên dịch. Bằng cách sửa đổi chương trình khi phân tích theo cách này, chúng tôi tạo ra một chương trình biên dịch hoạt động như ban đầu

dự định trong khi cũng thu thập dữ liệu bằng cách thực hiện các hướng dẫn được đo lường. Điều này đảm bảo rằng kết quả thực hiện không bị thay đổi và do đó dữ liệu thu thập phản ánh chính xác thông tin cần thiết để đào tạo PAC học tập thuật toán.

Để chạy thuật toán học PAC, chúng tôi yêu cầu dữ liệu mẫu của các giá trị của các biến trong biểu thức của một câu lệnh điều kiện, cũng như liệu đánh giá biểu thức đã dẫn đến "phân nhánh" hoặc "giảm dần". Các nhánh nhảy đến một nhãn trong khối sau khi câu lệnh điều kiện của nhánh hoạt động đánh giá là đúng và rơi vào khối khi điều kiện câu lệnh được đánh giá là sai. Ví dụ nếu trong quá trình thực hiện chương trình, một điều kiện câu lệnh so sánh một biến x với một biến y, cả hai đều là số nguyên với giá trị tương ứng là 3 và 5, và mã phân nhánh ra, điều này là cần thiết cho mã được thiết bị để lưu trữ các giá trị 3 và 5 cũng như nhãn để biểu thị rằng câu lệnh điều kiện dẫn đến một nhánh.

Lấy mẫu các giá trị biến là một nhiệm vụ khá đơn giản với Soot, vì mã ba địa chỉ cho Câu lệnh If trong Jimple tham chiếu đến cả hai toán hạng, từ đó các giá trị của chúng có thể dễ dàng đạt được. Xác định trạng thái chuyển tiếp của tình trạng phức tạp hơn vì đó không phải là thông tin dễ dàng tiếp cận qua Soot. Trong Hình 4.1, 4.2 và 4.3, một đoạn mã được kiểm tra trong bản gốc của nó Java, biểu diễn Jimple trung gian và Jimple sau nhạc cụ xảy ra. Hình 4.1 là đơn giản với một nhánh bắt đầu ở dòng 3. Chúng ta sẽ muốn thiết lập mã này để khi thực thi các giá trị x và y được lưu lại, cùng với việc nhánh đã được lấy hay chưa. Mã Jimple trong Hình 4.2 cũng đơn giản như vậy, tuy nhiên có một số khác biệt quan trọng cần lưu ý. Dòng 3 của Hình 4.2 vẫn là câu lệnh có điều kiện, tuy nhiên bây giờ điều kiện đã được

```
1: int x = arg0; 2:
int y = arg1; 3: if(x
< y){ 4:
System.out.println(x + y); 5: } 6: return;
```

Hình 4.1: Đoạn mã trước khi đo lường

```
1 :     i0 := @parameter0: int; i1 :=
2 :     @parameter1: int; nếu i0 >= i1 chuyển
3 :     đến nhãn1; virtualinvoke
4 :     r1.<java.io.PrintStream: void println chuyển đến nhãn2; 6 : nhãn1:
5 :

7 : nhãn2:
8 :     trở lại;
```

Hình 4.2: Biểu diễn trung gian của đoạn mã

lật ngược lại sao cho thay vì x < y thì nó là x(i0) >= y(i1). Điều này nghĩa là khi bản gốc Mã Java trong Hình 4.1 sẽ rơi vào một đánh giá thực sự, thay vào đó Jimple sẽ nhánh ra khỏi khối trên một đánh giá sai. Dòng 6 và 7 của Hình 4.2 cũng giữ một chi tiết quan trọng là nhãn được tạo ra cho cả việc bước qua điều kiện khối và để hoàn thành khối có điều kiện. Điều này tương đương với việc đưa ra tất cả nếu các câu lệnh trong Java là một khối else rỗng khi chưa có khối nào được chỉ định.

Hình 4.3 hiển thị mã Jimple đã hoàn thành với các hướng dẫn bổ sung được thêm vào bằng thiết bị đo lường. Dòng 3 hiện có lệnh gọi tĩnh của setData, dòng 5 lệnh gọi setFlag và phương thức storeArgs 10. Phương thức setData luôn luôn được gọi trước câu lệnh if để dữ liệu không bị ảnh hưởng bởi kết quả của statement. setData lấy i0 và i1, các giá trị x và y của chúng ta, và lưu trữ chúng trong một biến đối tượng bên ngoài. Phương thức setFlag được gọi bên trong điều kiện khối và yêu cầu mã được đo lường báo cáo các giá trị biến được lưu trữ là có dẫn đến một đánh giá 'đúng'. Sau đó cả hai nhánh hội tụ tại label2 trên dòng

```
1 :     i0 := @tham số0: int;
2 :     i1 := @tham số1: int;
3 :     staticinvoke <void setData(int,int)>(i0, i1);
4 :     nếu i0 >= i1 chuyển đến nhãn1;
5 :     staticinvoke <void setFlag()>();
6 :     virtualinvoke r1.<java.io.PrintStream: void println
7 :     chuyển đến nhãn2;
8 : nhãn1:
9 : nhãn2:
10 :     staticinvoke <void storeArgs()>();
11 :     trở lại;
```

Hình 4.3: Biểu diễn Jimple sau khi đo đạc

9, chỉ có các đánh giá 'đúng' được gọi là setFlag. Các giá trị trong điều kiện câu lệnh cũng như giá trị của cờ sau đó đều được thêm vào tập giá trị cuối cùng ở dòng 10.

Thuật toán 2 Mã giả cho thuật toán Branch Instrumentation

```
1: lớp BranchInstruter mở rộng BodyTransformer
2: Khởi tạo reporterClass, setData, setFlag, storeArgs, report
3: hàm internalTransform(body, phase, options)
4:     cho stmt trong body.units() làm
5:         nếu stmt là một thể hiện của IfStmt thì
6:             điều kiện    stmt.getCondition()
7:             insertBefore(setData, điều kiện.LHS, điều kiện.RHS, stmt)
8:             chènSau(đặtCờ, stmt)
9:             chènSau(storeArgs, stmt.getTarget())
10:         kết thúc nếu
11: kết thúc cho
12:     nếu body.isMain() thì
13:         đối với mỗi stmt trong đơn vị của cơ thể làm
14:             nếu stmt là một thể hiện của ReturnStmt thì
15:                 insertBefore(báo cáo, stmt)
16:             kết thúc nếu
17:         kết thúc cho
18: kết thúc nếu
19: chức năng kết thúc
```

Để thực hiện việc đo đạc và đạt được kết quả mong muốn, một algorithm được sử dụng để lặp lại biểu diễn Jimple của mã, chèn

các câu lệnh bổ sung khi cần thiết. Mã giả cho thuật toán được tạo ra cho mục đích này
mục đích được thể hiện trong Thuật toán 2. Bản thân mã mở rộng BodyTransformer của Soot
lớp, là công cụ của khung Soot để thực hiện các thay đổi đối với Body, nội bộ của Soot
biểu diễn các thuộc tính của phương pháp, bao gồm các tuyên bố rằng phương pháp là
bao gồm. Dòng 2 khởi tạo một đối tượng reporterClass cũng như bốn phương thức
rằng reporterClass có: setData, setFlag, storeArgs và report.
reporterClass là một lớp trợ giúp chứa cấu trúc dữ liệu để theo dõi
của dữ liệu đang được thu thập trong khi mã được đo lường được thực thi. Nó cũng
chứa các phương thức đã đề cập ở trên mà Soot coi là các đối tượng để truyền làm tham số
vào các phương thức insertBefore và insertAfter của nó.

Vì chúng ta đang sử dụng BodyTransformer, chúng ta đã biết trên Dòng 3 rằng chúng ta có
truy cập vào Body đang được chuyển đổi và trên Dòng 4 có thể bắt đầu lặp lại tất cả
các câu lệnh trong phương pháp. Dòng 5 kiểm tra từng câu lệnh trong thân phương pháp
để xem liệu nó có phải là IfStmt hay không, đó là biểu diễn nội bộ của Soot cho tất cả các điều kiện
các câu lệnh. Khi chúng ta biết câu lệnh hiện tại là một câu lệnh có điều kiện, chúng ta có thể
sử dụng hàm getCondition để lấy tất cả dữ liệu có liên quan đến chính điều kiện đó
trên Dòng 6. Các Dòng 7 đến 9 sau đó gọi insertBefore đã đề cập ở trên của Soot và
insertAfter phương pháp. Như chúng ta biết rằng thiết bị đo lường hiện đang ở trên
câu lệnh điều kiện, sử dụng insertBefore dẫn đến phương thức được chèn là
được gọi ngay trước khi điều kiện xảy ra. Điều này được sử dụng trên Dòng 7 để thu thập
hai giá trị được so sánh trong điều kiện của câu lệnh có điều kiện. Điều này sẽ cẩn thận
của việc thu thập các giá trị biến liên quan đến câu lệnh có điều kiện, nhưng nó vẫn còn
cần thiết để xác định xem kết quả là “phân nhánh” hay “thụt lùi”.

Bằng cách sử dụng insertAfter trên Dòng 8, chúng ta thêm phương thức setFlag vào chương trình
thực hiện tại điểm mã ngay sau câu lệnh điều kiện. Như

một câu lệnh có điều kiện nhảy đến một điểm mã khác khi thực thi các nhánh
ra kết quả này trong phương thức setFlag chỉ được gọi khi có điều kiện
câu lệnh dẫn đến một sự thất bại. Bản thân phương pháp sau đó chỉ cần đặt một cờ trong
Lớp phóng viên.

Phương pháp cuối cùng được chèn vào quá trình thực thi chương trình tại thời điểm này là
phương thức storeArgs trên Dòng 9. Mặc dù nó cũng sử dụng phương thức insertAfter của Soot,
nó không lấy câu lệnh hiện tại làm tham số, mà lấy mục tiêu của câu lệnh
stmt.getTarget(). Một lợi ích bổ sung khi biết rằng câu lệnh hiện tại là một
IfStmt là nó có một mục tiêu, đó là điểm mã mà lệnh thực thi nhảy tới khi
câu lệnh có điều kiện dẫn đến một "phân nhánh". Trong trường hợp mục tiêu là
sớm hơn trong quá trình thực thi mã so với câu lệnh hiện tại (chẳng hạn như trong các vòng lặp), thì đó là
đáng chú ý là một "fall-through" cũng đạt đến điểm mã của mục tiêu, mặc dù sau đó
"fall-through" đã được thực hiện. Vì cả "branch-out" và "fall-through"
dẫn đến việc thực hiện đạt được mục tiêu của điều kiện, đây là nơi an toàn để gọi
Phương thức storeArgs lưu các giá trị biến được lưu trữ cùng với cờ
đã được thiết lập hay chưa.

Cuối cùng, thuật toán kiểm tra Dòng 12 và 14 xem câu lệnh hiện tại có phải là
trả về phương thức Main và nếu vậy nó sẽ chèn phương thức report của reporterClass
xuất toàn bộ dữ liệu thu thập được vào một tệp văn bản để sử dụng sau này.

4.2 Thu thập dữ liệu

Sau khi chương trình đang được phân tích được trang bị đầy đủ, nó sẽ được chạy trên các đầu vào thử nghiệm để
tạo dữ liệu mẫu L của chúng tôi. Chất lượng và số lượng đầu vào thử nghiệm rất quan trọng đối với
quá trình cuối cùng vì nó phải bao gồm một phạm vi đầu vào đủ rộng. Nếu không có đủ

dữ liệu thuật toán học tập PAC không thể xác định chính xác và rõ ràng
nơi đặt các khoảng cách, cả hai đều do thiếu các đường chiếu kép để
tách biệt, và do tính thưa thớt của dữ liệu được tách biệt. Để có được
dữ liệu mong muốn, các trường hợp thử nghiệm phải thực hiện các phân đoạn mã được đo lường và thu thập
dữ liệu mẫu đa dạng. Mặc dù phân tích tĩnh truyền thống không yêu cầu thực hiện
của mã trong quá trình phân tích, khía cạnh học máy của luận án này làm cho nó
cần thiết để thu thập dữ liệu đào tạo khi chạy.

Như đã đề cập, luận án này sử dụng các trường hợp thử nghiệm như một phương tiện để thực hiện các công cụ
chương trình và thu thập dữ liệu mẫu cần thiết, tuy nhiên như thử nghiệm Java truyền thống
tập trung vào việc xác nhận hoặc bác bỏ các khẳng định, nhu cầu của chúng tôi khác với nhu cầu của
kiểm tra truyền thống. Khi chúng tôi tạo ra các bài kiểm tra để thực thi mã Java, phần lớn
các phương pháp tạo thử nghiệm cũng tạo ra các khẳng định, tuy nhiên đây là sản phẩm phụ của
quá trình. Lý tưởng nhất là các bài kiểm tra nên chạy càng nhiều mã càng tốt trong khi sử dụng cùng một
kiểm tra đầu vào để tăng khả năng thực hiện các phương pháp nội bộ với đầu vào hợp lệ.
Trong khi các thử nghiệm cụ thể hơn thực hiện một phương pháp duy nhất vẫn hữu ích, chúng có thể
đóng góp dữ liệu ít có khả năng gặp phải trong quá trình thực thi toàn bộ chương trình.

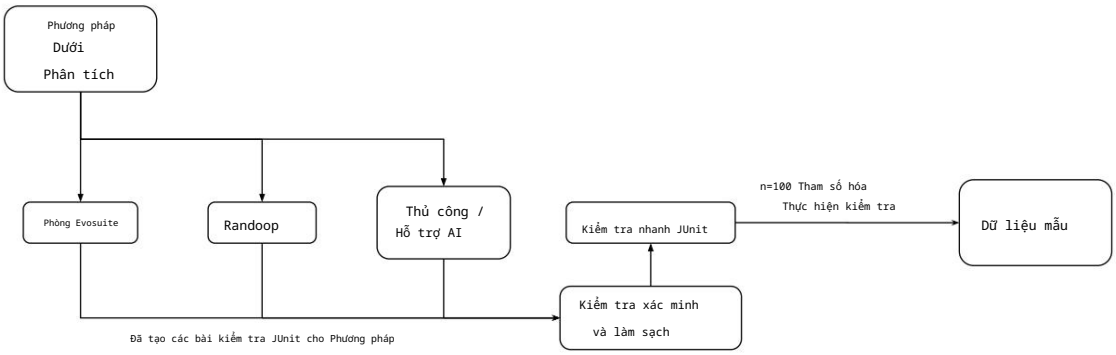
Do tính phức tạp của nhiệm vụ và nhu cầu tìm ra các kỹ thuật hiệu quả cho
nhiều chương trình khác nhau, nhiều phương pháp tạo thử nghiệm được sử dụng. Hai phương pháp tự động
máy phát thử nghiệm, Evosuite [4] và Randoop [8], được sử dụng cho mục đích này. Cả hai công cụ
phân tích các lớp Java đã biên dịch được cung cấp và các bộ đầu ra của các trường hợp thử nghiệm JUnit [14]
thực hiện các phương pháp trong các lớp đó. Để thu thập dữ liệu đa dạng hơn, các trường hợp thử nghiệm bổ sung
có thể được tạo ra hoặc các trường hợp thử nghiệm hiện có có thể được sửa đổi với các giá trị đầu vào khác nhau.
Trong nhiều trường hợp, việc tái sử dụng các trường hợp thử nghiệm hiện có và cung cấp dữ liệu đầu vào mới sẽ hiệu quả hơn
giá trị để thử nghiệm. JUnit-QuickCheck [6], một thư viện triển khai dựa trên thuộc tính
kiểm tra cho JUnit, được sử dụng để cung cấp các giá trị đầu vào mới này cho các giá trị đã tạo trước đó của chúng tôi

kiểm tra.

4.2.1 Phòng EvoSuite

Evosuite là một công cụ tạo trường hợp thử nghiệm tự động cho các chương trình Java, chủ yếu nhằm mục đích trong việc phát hiện lỗi thông qua việc đáp ứng các yêu cầu về phạm vi phủ sóng [4]. Nó sử dụng một phương pháp di truyền thuật toán để tạo ra các trường hợp thử nghiệm, thay đổi bộ thử nghiệm để tối đa hóa phạm vi bao phủ của mã mục tiêu. Evosuite có một số chức năng thể dục khác nhau hướng dẫn tìm kiếm quá trình hướng tới việc bao phủ càng nhiều nhánh thực thi càng tốt. Vì điều đó là cần thiết để người học PAC có dữ liệu được thu thập từ việc thực hiện cả hai nhánh của một câu lệnh có điều kiện, chiến lược tối đa hóa này hỗ trợ trực tiếp cho trường hợp sử dụng của chúng tôi các bài kiểm tra được tạo ra.

Mặc dù có khả năng tạo trường hợp thử nghiệm mạnh mẽ, Evosuite cũng cung cấp các thử thách dài trong việc sử dụng và kết hợp vào đường ống của chúng tôi. Một trong những khó khăn là tạo ra các trường hợp thử nghiệm không hợp lệ. Vì trọng tâm chính của Evosuite là đạt được độ phủ mã cao, đôi khi nó có thể tạo ra các trường hợp thử nghiệm không tuân thủ những ràng buộc hoặc yêu cầu của hệ thống đang được thử nghiệm. Một trong những thử nghiệm như vậy là phổ biến để Evosuite tạo ra cung cấp dữ liệu đầu vào đa dạng nhưng không thực hiện phương pháp



Hình 4.4: Tổng quan cấp cao về quy trình thu thập dữ liệu mẫu.

sử dụng nó, thay vào đó khẳng định rằng giá trị được cung cấp vẫn không thay đổi. Mặc dù
kiểm tra hợp lệ để phát hiện và báo cáo các phương pháp có tác dụng phụ không mong muốn, kết quả
khi chạy nó cung cấp dữ liệu mẫu trùng lặp cho các lần thực thi trên tất cả các đầu vào.

Điều này đòi hỏi phải kiểm tra thủ công và cắt tỉa các trường hợp thử nghiệm được tạo ra để
đảm bảo tính hợp lệ của chúng, với số lượng lớn các chương trình không nhận được bất kỳ giá trị nào
các trường hợp thử nghiệm sau khi quá trình tạo hoàn tất. Điều này được bù đắp một phần bằng cách thực hiện
ing làm sạch mã thủ công trước khi cung cấp cho Evosuite, vì nó có thể gặp khó khăn khi
phân tích mã với cấu trúc dữ liệu phức tạp hoặc mã phụ thuộc nhiều vào bên ngoài
thư viện. Thỉnh thoảng cần phải loại bỏ một số khía cạnh này trước khi Evosuite
sẽ chạy thành công. Tuy nhiên, làm như vậy có nguy cơ làm thay đổi hành vi của
phương pháp đang được phân tích và có thể làm giảm độ tin cậy của dữ liệu thu thập được.

Cuối cùng, Evosuite là một công cụ nghiên cứu và không có cùng độ tin cậy như
một công cụ công nghiệp, và do đó đã gặp phải nhiều sự cố hoặc các lỗi khác
tạo ra các bài kiểm tra tùy thuộc vào độ phức tạp của mã đầu vào. Giới hạn này được thực hiện
việc sử dụng Evosuite hiệu quả khó khăn như một giải pháp chung cho việc tạo đầu vào thử nghiệm
trong nhiều chương trình khác nhau.

Bất chấp những thách thức này, Evosuite vẫn có giá trị trong việc tạo trường hợp thử nghiệm tự động
Khả năng tạo ra một tập hợp đa dạng các trường hợp thử nghiệm bao gồm nhiều loại mã
các nhánh làm cho nó trở thành một thành phần thiết yếu của quá trình tạo thử nghiệm. Tuy nhiên,
do những vấn đề đã đề cập ở trên, các công cụ bổ sung cũng được sử dụng cho các trường hợp khi
Evosuite không đủ hiệu quả.

4.2.2 Vòng lặp ngẫu nhiên

Do số lượng chương trình lớn nên Evosuite không thể tạo ra bất kỳ
các bài kiểm tra hoặc các bài kiểm tra được tạo ra không thể sử dụng để thu thập dữ liệu đào tạo, Randoop [8] là

được coi là một giải pháp thay thế.

Các chương trình mà Evosuite không thành công sau đó được cung cấp cho Randoop [8] để tạo ra kiểm tra. Randoop là một công cụ tạo trường hợp kiểm tra tự động khác cho các chương trình Java, tập trung vào việc tạo ra các bài kiểm tra đơn vị bằng cách sử dụng một bài kiểm tra ngẫu nhiên hướng phản hồi phương pháp tiếp cận thế hệ. Phương pháp của Randoop bao gồm việc xây dựng thử nghiệm theo từng bước lặp lại các trường hợp, thực hiện chúng và sử dụng kết quả để hướng dẫn việc tạo ra các bài kiểm tra tiếp theo trường hợp. Nó sử dụng phương pháp tìm kiếm để tăng khả năng tạo ra các trường hợp thử nghiệm phát hiện lỗi, nhằm mục đích cải thiện chất lượng phần mềm.

Tương tự như Evosuite, Randoop cũng đưa ra những thách thức riêng của mình. Một trong những vấn đề gặp phải khi sử dụng Randoop là tạo ra các trường hợp thử nghiệm tầm thường. Khi sử dụng trong bộ sưu tập các chương trình mẫu của chúng tôi, Randoop thường xuyên tạo ra các trường hợp thử nghiệm không thực hiện có ý nghĩa các câu lệnh có điều kiện đang được phân tích, tương tự như vậy để Evosuites tạo ra các bài kiểm tra theo dõi tác dụng phụ. Điều này dẫn đến phần lớn của dữ liệu thời gian chạy được thu thập từ việc thực hiện các trường hợp thử nghiệm do Randoop tạo ra trùng lặp, chẳng hạn như dữ liệu được lấy mẫu bao gồm hàng trăm mục nhập chính xác cặp giá trị và nhãn giống nhau, hoặc hoàn toàn ngẫu nhiên, trong đó bản thân phương pháp không có ảnh hưởng đến dữ liệu thu thập được và đánh giá sâu hơn cho thấy nhiều hơn về Randoop tạo ra số lượng lớn hơn chương trình đang được phân tích.

Thật không may, mặc dù có kết quả đầy hứa hẹn với thử nghiệm ban đầu, thách thức những nỗ lực thất bại trong việc tạo ra bài kiểm tra đã vượt xa những thành công lẻ tẻ. Như Randoop như vậy không được sử dụng cho bất kỳ thế hệ thử nghiệm nào dẫn đến dữ liệu đào tạo đã được chuyển đến thuật toán học tập PAC. Với các giải pháp tự động chứng minh không khả thi đối với việc tạo ra các chương trình ngẫu nhiên thử nghiệm quy mô lớn, cần phải dựa vào về các phương pháp tạo thử nghiệm phù hợp hơn với mã nguồn.

4.2.3 Tạo bài kiểm tra thủ công và bán tự động

Trong những tình huống mà các phương pháp trước đây không tạo ra được các trường hợp thử nghiệm đầy đủ, con người việc tạo ra các trường hợp thử nghiệm thông thường trở nên cần thiết để cho phép thu thập các dữ liệu có ý nghĩa dữ liệu thời gian chạy cho thuật toán học PAC. Viết các trường hợp thử nghiệm theo cách thủ công, trong khi tốn thời gian, đảm bảo việc tạo ra các trường hợp thử nghiệm phù hợp hơn với việc thu thập dữ liệu đào tạo cho thuật toán học tập PAC. Trong luận án này, quá trình này được tăng tốc được thực hiện bằng cách tận dụng khả năng của Chat-GPT, một mô hình ngôn ngữ AI được đào tạo để hiểu và hỗ trợ nhiều nhiệm vụ khác nhau, bao gồm phân tích mã và trường hợp thử nghiệm thế hệ. Chat-GPT có thể đưa ra các gợi ý cho các trường hợp thử nghiệm hoặc các giá trị đầu vào cụ thể có nhiều khả năng thực hiện các đường dẫn thực thi khác nhau và phát hiện ra các lỗi tiềm ẩn, do đó cải thiện chất lượng và tính đa dạng của các trường hợp thử nghiệm được tạo ra. Mặc dù kết quả khi sử dụng Mô hình ngôn ngữ lớn như Chat-GPT là không thể đoán trước, nó thường là có thể đạt được kết quả có giá trị thông qua kỹ thuật nhanh chóng, một quá trình tương tác của việc điều chỉnh các yêu cầu để giải quyết tốt hơn những thiếu sót trong các phản hồi trước đó.

Ví dụ sau đây là lời nhắc chung có thể được đưa ra cho Chat-GPT cùng với bên cạnh mã nguồn của phương pháp đang đề cập:

Viết một bài kiểm tra JUnit cho phương pháp sau:

Các bài kiểm tra jUnit thu được từ lời nhắc trên bao gồm tất cả các phép đun sôi cần thiết mã erplate và sẽ biên dịch và chạy mà không có vấn đề gì. Chat-GPT cũng cung cấp các giá trị đầu vào có vẻ ngẫu nhiên cho các phương pháp đang được thử nghiệm, đáp ứng nhu cầu của quá trình tạo thử nghiệm bằng cách cung cấp các đầu vào khác nhau cho các chương trình được trang bị công cụ để để họ tạo ra dữ liệu đào tạo.

Điều đáng chú ý là tất cả các phương pháp tạo trường hợp thử nghiệm, bao gồm cả những phương pháp được tạo ra bởi Evosuite, Randoop và Chat-GPT, yêu cầu một số mức độ dọn dẹp thủ công

và kiểm tra để đảm bảo tính hợp lệ và sự liên quan của các trường hợp thử nghiệm được tạo ra. Điều này nỗ lực thủ công là rất quan trọng để duy trì chất lượng tổng thể của bộ kiểm tra và đảm bảo rằng dữ liệu thời gian chạy được thu thập là đáng tin cậy và hữu ích cho việc học PAC thuật toán.

4.2.4 Kiểm tra nhanh JUnit

Ngoài việc dọn dẹp thủ công các trường hợp thử nghiệm được tạo và viết, cần phải chạy các trường hợp thử nghiệm đó trên nhiều giá trị hợp lệ khác nhau để dữ liệu được tạo ra không bị giới hạn trực tiếp ở số lượng các trường hợp thử nghiệm có sẵn. JUnit-QuickCheck là một thư viện tích hợp với JUnit và triển khai thử nghiệm dựa trên thuộc tính, cho phép các thử nghiệm được thực hiện trên một loạt các đầu vào được tham số hóa [6]. Chạy Các thử nghiệm jUnit trên các đầu vào được tham số hóa này cho phép một số ít các thử nghiệm cung cấp hàng trăm dữ liệu đầu vào cho các phương pháp đang được phân tích.

Bằng cách chỉ định các thuộc tính mà các giá trị đầu vào phải đáp ứng, chẳng hạn như các ràng buộc về phạm vi hoặc loại giá trị, JUnit-QuickCheck nhanh chóng tạo ra nhiều đầu vào khác nhau đáp ứng các điều kiện này. Khi các trường hợp thử nghiệm được thực hiện, chúng được chạy lặp đi lặp lại với các giá trị khác nhau cho các đầu vào được tham số hóa cho đến khi các điều kiện đặt ra được đáp ứng. Vì dữ liệu thu thập được bằng quy trình này phụ thuộc vào cấu trúc của chính phần mềm, các thử nghiệm tham số hóa sau đó có thể được chạy lại cho đến khi có đủ lượng dữ liệu cho tất cả thực hiện nhánh được tạo ra với mỗi lần lặp lại cung cấp đầu vào ngẫu nhiên mới trong các tham số. Quá trình thu thập một loạt dữ liệu tham số rộng lớn cho phép quan sát hành vi của các câu lệnh có điều kiện bằng cách sử dụng PAC thuật toán học tập và tạo điều kiện cho việc tạo ra một loạt các đầu vào thử nghiệm khi để chạy bộ thử nghiệm đã tạo trước đó của chúng tôi.

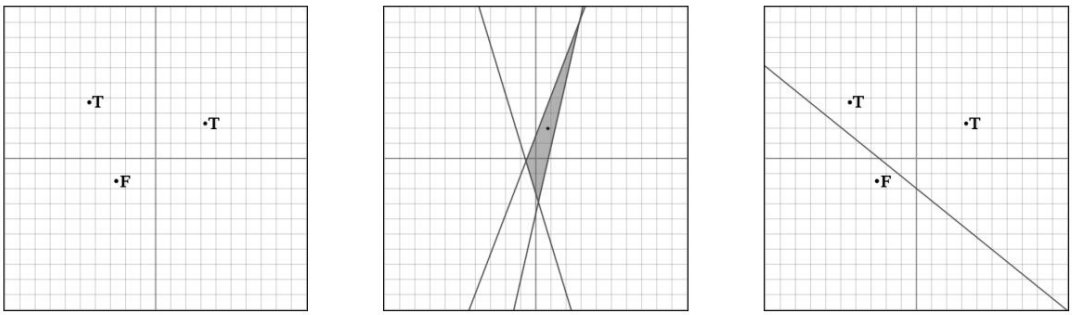
Quá trình tham số hóa khá đơn giản để thực hiện thủ công, tuy nhiên khi Chat-
GPT hiện đang được sử dụng để tạo ra các bài kiểm tra, một thay đổi đơn giản
lời nhắc trên dẫn đến các trường hợp thử nghiệm được tạo ra đã có các yêu cầu cần thiết
mã mẫu và những thay đổi để cung cấp đầu vào có tham số:

Viết một bài kiểm tra JUnit cho phương pháp sau sử dụng jUnitQuickcheck
để giới hạn các tham số phương pháp giữa -1000 và 1000:

Điều này chắc chắn dẫn đến một thử nghiệm jUnit có thể chạy ngay lập tức bằng cách sử dụng jU-
nitQuickcheck để chọn các biến đầu vào ngẫu nhiên cho các phương pháp đang được thử nghiệm. Một
thực hiện thử nghiệm jUnit dẫn đến 100 lần thực hiện thử nghiệm tham số hóa, trong đó
sau đó dẫn đến một lượng dữ liệu đào tạo khác nhau từ chương trình được đo lường
tùy thuộc vào tần suất các hướng dẫn được đo lường được chạy trong một lần
gọi phương thức.

4.3 Thuật toán học tập PAC

Với dữ liệu mẫu L từ các lần thực hiện thử nghiệm của chúng tôi, chúng tôi có thể cung cấp khả năng học tập PAC
thuật toán với đầu vào để tạo vùng. Thuật toán học tập PAC ac-
hoàn thành nhiệm vụ của mình bằng cách sử dụng phép chiếu không gian kép để tạo ra các vùng bao bọc
ra khỏi dữ liệu được lấy mẫu. Có hai không gian, mặt phẳng (x, y) ban đầu được gọi là
nguyên thủy V', và không gian mới (u, v) là không gian đối ngẫu V . Khi một điểm trên một không gian đối ngẫu
mặt phẳng có nhiều chiều được chiếu lên một không gian kép, nó trở thành một đường thẳng trên mặt phẳng kép đó.
Nghĩa là, phép chiếu này lấy một điểm có tọa độ (2, 3) và đặt chúng vào
công thức độ dốc-đoạn chắn cho một đường thẳng, với giá trị tọa độ x trở thành độ dốc
và giá trị tọa độ y bị phủ định và được sử dụng làm điểm cắt, tức là v = 2u 3.



Hình 4.5: Ví dụ trực quan về phép chiếu không gian kép được sử dụng trong Thuật toán học tập PAC

Sử dụng phương pháp này, dữ liệu mẫu L đầu tiên được chiếu từ không gian nguyên thủy V ′
vào không gian kép để tạo thành một tập hợp các đường phân tách không gian kép V thành các vùng
R . Một vùng r R được định nghĩa là một phép kết hợp của các khối đa diện, mà chúng ta tạo thành chúng bằng
các điểm mà các đường thẳng trong V giao nhau. Để "phân tách" các điểm trong V ′ chúng ta cần phải tìm
tâm của mỗi vùng trong R . Cho một vùng r R được xác định bởi tập hợp các điểm
(ui , vi), với i = 1, . . . n, tọa độ của tâm (ur, vr) là số học
có nghĩa là tất cả các điểm tạo thành vùng, tức là, ur = (n $\sum_{tôi=1}$ ui)/n và vr = (n $\sum_{tôi=1}$ vi)/n. Các
tâm điểm trong V được chiếu trở lại V ′ điều này một lần nữa dẫn đến các dòng, do đó (ur, vr)
trong V trở thành y = urx vr trong V ′. Các dòng này xác định tập hợp cuối cùng của các vùng R′ trong
không gian nguyên thủy, được sử dụng trong phần còn lại của luận án để xác định một tập hợp các vị ngữ P.

Để minh họa quá trình chiếu không gian kép, hãy xem xét Hình 4.5, trong đó
hình ảnh bên trái chứa ba điểm mẫu được dán nhãn. Hình ảnh ở giữa cho thấy dự án
sự sắp xếp của chúng vào mặt phẳng kép nơi ba điểm trở thành ba điểm chồng lên nhau
các đường thẳng. Các điểm giao nhau của chúng tạo thành một vùng, được tô bóng và một điểm bên trong vùng đó
là tâm được tính toán. Tâm này sau đó được chiếu trở lại mặt phẳng nguyên thủy
như được mô tả trên hình ảnh bên phải, nơi nó trở thành một đường thẳng. Trong ví dụ này, tâm này
đường phân tách ba điểm dữ liệu ban đầu.

Thuật toán 3 cung cấp mã giả cho quá trình tìm vùng mà chúng tôi triển khai

được đề cập. Thuật toán lấy tập hợp các siêu phẳng H làm đầu vào và trả về một tập hợp
của vùng S′, trong đó mỗi vùng là một tập hợp các điểm giao nhau. Một giao điểm κ là
được biểu diễn bằng tọa độ và tập hợp các siêu phẳng tạo nên giao điểm này.

Trong quá trình triển khai của chúng tôi, h sẽ là một đường thẳng được biểu diễn bằng công thức chặn độ dốc,
và chứa các tham chiếu đến tất cả các dòng khác giao nhau được sắp xếp theo giá trị x của
tọa độ giao điểm (trong hệ tọa độ xy).

Thuật toán 3 Mã giả cho thuật toán tìm vùng.

```
1: hàm Find-Regions(H)
2: S
3:     với mọi siêu phẳng h    H thì
4:         với mọi giao điểm κ    h thì
5:             giờ′    siêu phẳng giao nhau với h tại κ′
6:             κn    giao lộ tiếp theo dọc theo h
7:             S    {κ}
8:             lặp lại
9:                 S′   S    {κn}
10:                giờ    siêu phẳng giao nhau h    ′tại kn
11:                κn    giao lộ tiếp theo dọc theo h′
12:            cho đến khi κ = κn
13:            S′    S′    {S}
14:        kết thúc cho
15: kết thúc cho
16:    Quay lại S′
17: chức năng kết thúc
```

Khi tất cả các giao điểm được tính toán, từ dòng 4 đến dòng 14, thuật toán lặp lại
qua các giao lộ này với mỗi giao lộ đóng vai trò là điểm khởi đầu để tìm kiếm
một vùng tiềm năng. Các dòng 8 đến 12 lấy điểm bắt đầu κ và tăng κn lên
giao lộ gần nhất tiếp theo dọc theo đường hiện tại h′. Trên dòng 10 h′ được cập nhật với
siêu phẳng giao nhau gần nhất.

Thuật toán tiếp tục cập nhật κn và h′, cho đến khi κn trở lại trạng thái ban đầu
cắt ngang cho vùng. Theo trực giác, thuật toán 'đi bộ' xung quanh chu vi của
các cạnh của vùng sẽ ghé thăm và lưu trữ từng đỉnh khi thuật toán tiến triển. Khi

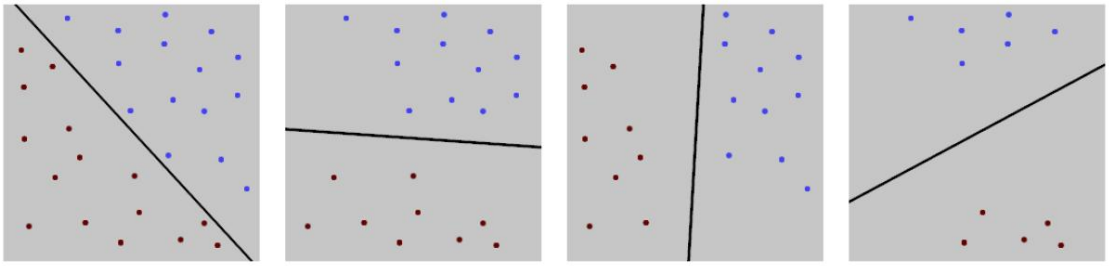
kết thúc vòng lặp ở dòng 12, thuật toán đã duyệt qua toàn bộ vùng,
nó lưu trữ như tập hợp các điểm giao nhau này và thêm các điểm này vào kết quả trả về
đặt trên dòng 13. Khi tất cả các giao điểm bắt đầu có thể được đi qua, thuật toán
đã lấy được tập hợp đầy đủ các vùng và trả về nó.

Khi các vùng cuối cùng của chúng ta trong không gian nguyên thủy được tìm thấy, chúng ta có thể kiểm tra các nhãn
trên dữ liệu và bắt đầu dán nhãn các vùng là dương và âm. Phương pháp sắp xếp
sử dụng thuật toán bao phủ tập tham lam để tìm vùng chứa giá trị lớn nhất
số điểm dữ liệu được gắn nhãn tích cực và loại bỏ vùng đó và những điểm đó
từ sự cân nhắc. Thuật toán bao phủ tập hợp sau đó tiếp tục cho đến khi tất cả các số dương
các điểm dữ liệu đã bị xóa. Bộ vùng cuối cùng bao gồm các vùng tích cực đó
các điểm tạo thành một tổ hợp boolean của các halfspace H được đưa ra bước tiếp theo
của quá trình.

4.4 Phép chiếu miền

Sau khi thuật toán đã tạo ra tập hợp cuối cùng của các nửa không gian H, sau đó nó có thể tạo ra
phân vùng tạo thành miền vị ngữ của chúng ta. Cho một siêu phẳng h trong H được định nghĩa bởi
một độ dốc m và một giao điểm Y b, chúng ta cần phát triển một phương pháp tìm ra một liên kết
giá trị số nguyên để hoạt động như một phân vùng trong miền. Thuật toán xác định những gì
giá trị phù hợp để tạo phân vùng cũng như biến nào là phân vùng
được định nghĩa cho. Vì h biểu diễn một vị trí tiềm năng để tách các giá trị của một biến, chúng ta
sử dụng các siêu phẳng ngang làm phân vùng cho các giá trị dọc theo trục dọc và trục dọc
siêu phẳng cho các giá trị dọc theo trục ngang.

Có một số cân nhắc cần phải được tính đến để đạt được
phép chiếu này. Vì một siêu phẳng được xác định bởi độ dốc, đường thẳng tương ứng có thể



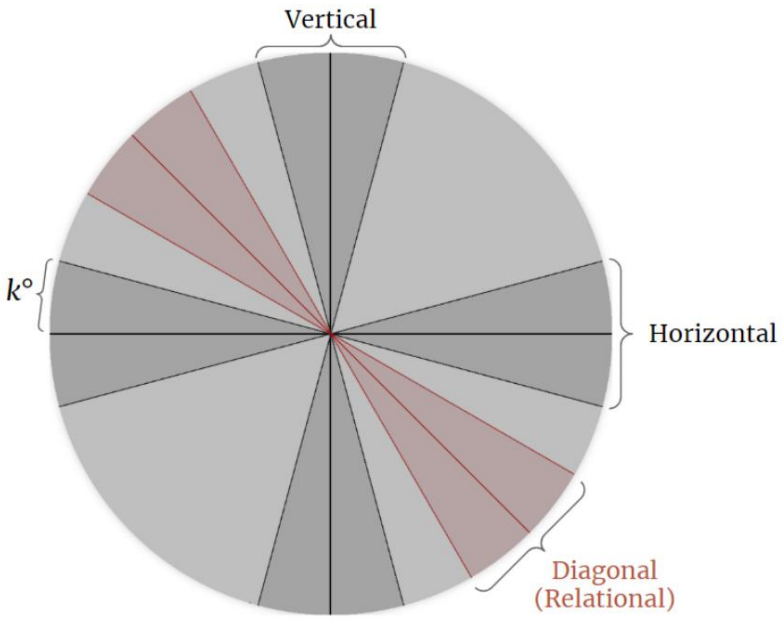
Hình 4.6: Các đường chéo, ngang, dọc và đường siêu phẳng không rõ ràng.

không bao giờ thẳng đứng khi m< ∞. Hơn nữa, một siêu phẳng với m= 0, tương ứng
theo một đường nằm ngang hoàn hảo, không có khả năng xảy ra. Điều này là do bản chất của
phép chiếu không gian kép, trong đó m trong không gian nguyên thủy tương ứng với giá trị u của a
tâm của vùng trong không gian kép. Mặc dù tâm này có thể là
tại chính xác u = 0, nó có nhiều khả năng bị lệch một chút, dẫn đến không nằm ngang
siêu phẳng.

Cả hai vấn đề này đều được giải quyết bằng cách đưa ra ngưỡng K , số lượng
độ mà đường thẳng tương ứng của h có thể phân kỳ từ mỗi trục trong khi vẫn là
được coi là được căn chỉnh theo trục. Tổng cộng có bốn trường hợp riêng biệt mà thuật toán
xem xét khi chiếu các siêu phẳng của thuật toán học PAC lên một vị từ
miền; những trường hợp này được thể hiện trong Hình 4.6 như là ví dụ. Một biểu diễn trừu tượng
của việc xác định các trường hợp được thể hiện trong Hình 4.7.

4.4.1 Tách biệt theo chiều dọc

Đối với trường hợp thứ ba được thể hiện trong Hình 4.6 khi độ dốc của h nằm trong K của phương thẳng đứng
nó đang gợi ý một sự tách biệt tiềm năng của các giá trị tại điểm cắt X của h. Giá trị này
được tính bằng cách giải cho X khi Y = 0 và sau đó được thêm vào danh sách các giá trị tiềm năng
phân vùng. Chặn Y được sử dụng làm dữ liệu được tách biệt trong công việc này luôn luôn



Hình 4.7: Đường tròn đơn vị trực quan hóa các phạm vi góc đường tạo ra các phép chiếu siêu phẳng khác nhau.

tập trung xung quanh (0, 0). Tuy nhiên, đối với dữ liệu được biết là bù trừ thì chặn cũng nên được bù trừ để tính đến sự thay đổi từ giá trị chặn do K gây ra .

4.4.2 Phân tách theo chiều ngang

Trường hợp thứ hai trong Hình 4.6 tuân theo một quá trình tương tự khi độ dốc của h nằm trong K của phương ngang. Điều này ngụ ý một sự tách biệt tiềm tàng tại điểm cắt Y h, trong đó Giao điểm Y được sử dụng thay cho giao điểm X làm điểm phân tách. Giá trị này đã được tính toán như một phần của h và cũng có thể được thêm vào danh sách các tiềm năng phân vùng.

4.4.3 Phân tách theo đường chéo

Xử lý các đường gần như chéo như thể hiện trong trường hợp đầu tiên trong Hình 4.6, khi độ dốc của h nằm trong K của (ngang ± 45), cần phải xem xét thêm. Vì một phân vùng bắt nguồn từ h sẽ không liên quan trực tiếp đến bất kỳ biến nào riêng lẻ không phù hợp để sử dụng h để cung cấp một phân vùng. Hơn nữa, dữ liệu có thể được làm sạch được phân tách bằng một đường quan hệ thuần túy, được xác định bởi mối quan hệ X = Y hoặc X = Y gợi ý hành vi chương trình tự nó có tính quan hệ. Đối với các chương trình như thế này miền vị ngữ không đủ để thực hiện phân tích. Do đó thuật toán cung cấp cho người dùng cảnh báo khi gặp phải siêu phẳng như vậy trong quá trình quá trình tạo phân vùng, gợi ý rằng một miền trừu tượng biểu cảm hơn có thể là cần thiết.

4.4.4 Định hướng không rõ ràng

Đối với các trường hợp còn lại khi h không phù hợp với các danh mục trước đó, siêu phẳng bị loại bỏ vì bất kỳ phân vùng nào được tạo ra từ nó sẽ không có khả năng cung cấp bất kỳ giá trị nào để phân tích. Một dòng như vậy được hiển thị như trường hợp thứ tư trong Hình 4.6, trong đó mặt phẳng tách biệt không phù hợp cho phép chiếu.

4.5 Tổng hợp các miền hoàn chỉnh

Các phân vùng được tạo ra cho mỗi biến tạo thành cơ sở của toàn bộ miền. quá trình hợp nhất các phân vùng này bị ảnh hưởng bởi thứ tự mà các siêu phẳng được lựa chọn bởi thuật toán học tập PAC và bản chất có thể có mối quan hệ của hành vi của chương trình. Hơn nữa, vì mục tiêu của luận án này là cụ thể

tạo ra các miền rời rạc, các giá trị phân vùng riêng lẻ phải được tạo thành các miền rời rạc vị ngữ.

4.5.1 Ưu tiên phân vùng

Khi các phân vùng được tạo ra trước đó trong quá trình thuật toán học PAC tách biệt nhiều điểm hơn, chúng được gán mức độ ưu tiên cao hơn. Điều này đảm bảo rằng những phần sớm này Các thông tin có xu hướng mang nhiều thông tin hơn có ảnh hưởng lớn hơn đến hình dạng của miền cuối cùng.

4.5.2 Hành vi của chương trình quan hệ

Khả năng hành vi của chương trình là quan hệ được tính đến. Nếu phân vùng có trọng số lớn nhất cho thấy một hành vi quan hệ, tức là phân vùng là đường chéo và tách cả hai biến một cách bình đẳng, thì rất khó có thể có mức độ ưu tiên thấp hơn phân vùng có lợi cho việc phân tích.

Trong trường hợp như vậy, thuật toán sẽ dừng quá trình phân vùng và cảnh báo người dùng, như một miền vị ngữ biến đơn sẽ không đủ để nắm bắt các mối quan hệ biến và cần có một phạm vi trừu tượng toàn diện hơn.

4.5.3 Xây dựng miền

Khi tất cả các phân vùng đã được ưu tiên và bất kỳ hành vi quan hệ nào đã được thêm vào được mặc quần áo, miền vị ngữ hoàn chỉnh được xây dựng. Các phân vùng được sắp xếp theo theo thứ tự giảm dần của số cặp được tách ra và mỗi cặp được thêm vào miền xác định. Một giá trị số nguyên duy nhất được tạo ra trong quá trình chiếu được chia thành ba vị ngữ riêng biệt cho các giá trị nhỏ hơn, lớn hơn và chính xác bằng giá trị được tạo ra.

Việc bổ sung các phân vùng này sẽ tạo ra một phân đoạn mới của miền. Đối với
Ví dụ, việc thêm một phân vùng tại giá trị j sẽ chia phạm vi của biến đó thành ba
các phân đoạn: (∞, j), [j] và (j, +∞). Mỗi phân vùng được thêm vào sẽ chia nhỏ hơn nữa
phạm vi biến của nó.

Trong một số trường hợp, không đủ phân vùng được tạo ra cho một miền có giá trị. Đối với
trong những trường hợp này, các phân vùng được tạo cho một biến có thể được sử dụng cho cả hai biến.
Chiến lược này đảm bảo rằng ít nhất một số phân vùng được thực hiện, mặc dù nó có thể
không phải là tối ưu.

CHƯƠNG 5

SỰ ĐÁNH GIÁ

Để xác định hiệu quả của phương pháp tiếp cận của chúng tôi đối với việc tạo miền phân tích tĩnh
chúng tôi thực hiện đánh giá sau: chúng tôi thu thập dữ liệu mẫu từ các lần thực hiện thử nghiệm của
các chương trình được trang bị, cung cấp dữ liệu mẫu đó làm tài liệu đào tạo cho PAC của chúng tôi
thuật toán học tập để tìm hiểu các phân vùng tiềm năng trong miền của chúng tôi và kiểm tra các phân vùng mới
miền được tạo ra dựa trên các miền chung được chọn ngẫu nhiên bằng cách đánh giá chúng
độ chính xác so sánh trên tất cả các đường dẫn thực hiện. Những kết quả này sau đó được kiểm tra
để xác định xem miền được tạo ra có chính xác hơn miền được chọn ngẫu nhiên hay không
miền trung bình và nếu vậy thì cần bao nhiêu dữ liệu mẫu để đào tạo
Thuật toán học PAC để nhận được miền như vậy.

Các phần sau đây trình bày kết quả cho từng bước đánh giá cũng như các ghi chú
về việc tạo miền đã cố gắng. Các kết quả được trình bày để hỗ trợ trả lời
các câu hỏi nghiên cứu sau:

RQ1 Làm thế nào để thu thập dữ liệu hiệu quả từ các chương trình để cung cấp mẫu cho
Người học PAC?

RQ2 Kích thước mẫu ảnh hưởng như thế nào đến đầu ra của thuật toán học PAC?

RQ3 Thuật toán học PAC có thể tạo ra các miền trừu tượng của vị từ trên một tập hợp không?
của các chương trình chuẩn mực?

5.1 Lọc ban đầu các chương trình thử nghiệm

Đánh giá được thực hiện trên một tập hợp gồm 33 tệp chương trình Java chứa 209 phương pháp
ods [13]. Bộ chuẩn mực này được lấy từ công trình trước đây liên quan đến luận án này,
bao gồm nghiên cứu SA sử dụng các miền vị ngữ rời rạc. Mặc dù được sử dụng trong các
làm việc có nhiều chương trình và phương pháp không có các tính năng có thể học được
đào tạo thuật toán học PAC của phép chiếu không gian kép.

Thuật toán học PAC được áp dụng cho các bài toán so sánh hai hằng số
các biến có quan hệ ≤ hoặc ≥. Khi một trong các biến là hằng số, nó dẫn đến
một không gian một chiều trong phép chiếu không gian kép, và hơn nữa có thể là
được phân tích bằng các phương pháp đơn giản hơn như trích xuất giá trị trong đó hằng số
giá trị được sử dụng như phân vùng. Sự bằng nhau hoặc bất bình đẳng dẫn đến dữ liệu được gắn nhãn trong đó
phần lớn các kết quả chia sẻ cùng một nhãn. Dữ liệu không cân bằng như vậy không thể được sử dụng
để học cách phân tách chính xác.

Một khó khăn bổ sung trong các chương trình Java được cung cấp phát sinh từ các chuyển đổi
được áp dụng cho chúng để cho phép các chương trình biên dịch và chạy mà không cần thêm bên ngoài
mã. Một phép biến đổi phổ biến là gán các giá trị ngẫu nhiên cho các biểu thức không thể giải quyết.
Đối với đánh giá SA, điều này không gây ra vấn đề gì vì phân tích được thực hiện mà không cần thực hiện.
cắt mã và do đó các giá trị ngẫu nhiên là những chỗ giữ chỗ phù hợp. Tuy nhiên,
công việc này tận dụng các giá trị thực tế của các biến cần thiết cho dữ liệu đào tạo,
do đó, một sự chuyển đổi như vậy tạo ra nhiều tình huống trong đó dữ liệu đào tạo được lấy mẫu
sẽ là so sánh trực tiếp của hai số nguyên ngẫu nhiên. Mặc dù dữ liệu này sẽ là
về mặt kỹ thuật có thể sử dụng trong thuật toán học PAC, nó sẽ không cung cấp bất kỳ hiểu biết nào
vào hành vi của chương trình đang được phân tích vì nó sẽ tương đương với việc đào tạo
thuật toán học PAC hoàn toàn dựa trên các giá trị ngẫu nhiên.

Để loại bỏ những tình huống như vậy, các chương trình ban đầu được lọc thủ công tại mã nguồn chỉ dành cho những phương thức có chứa các câu lệnh if so sánh hai hoặc nhiều các biến không hằng số. Hơn nữa các biến không hằng số đó không được được đặt thành các giá trị ngẫu nhiên mà không có bất kỳ phép tính nào được thực hiện để thay đổi chúng trước so sánh. Nếu các biến ban đầu được đặt thành các giá trị ngẫu nhiên nhưng sau đó được thay đổi theo hành vi của chương trình, các phương thức được giữ lại.

Bảng 5.1: Kết quả lọc lớp

Loại	Các lớp Phương pháp	
Tổng cộng	33	209
Chỉ là những so sánh tầm thường	16	158
Không có so sánh hợp lệ	8	16
Không thể tạo bộ kiểm tra	1	9
Chương trình chuẩn mực hợp lệ	8	11

Bảng 5.1 hiển thị số lượng chương trình được lọc ra khỏi điểm chuẩn được thiết lập cùng với lý do lọc. Các chương trình được liệt kê là 'Chỉ so sánh tầm thường' là những phép so sánh của hai hoặc nhiều biến không đổi, nhưng các giá trị trong số các biến đó sẽ được tạo ra hoàn toàn ngẫu nhiên, độc lập với chương trình thực hiện. Các chương trình này có thể được trang bị và có thể có PAC thuật toán học tập cố gắng tạo ra các miền cho chúng. Tuy nhiên, những kết quả như vậy là không có khả năng nắm bắt chính xác hành vi quan hệ của các biến đang được phân tích. 'Không có so sánh hợp lệ' là các chương trình không có câu lệnh if so sánh hai hoặc nhiều biến không hằng số hơn. Các chương trình này không cung cấp dữ liệu cho việc học PAC thuật toán để đào tạo. Một chương trình bị xóa vì thực hiện nó đòi hỏi bên ngoài dữ liệu bản đồ để chương trình thực hiện các phép tính, nếu không có nó sẽ hoạt động như chương trình 'Chỉ so sánh tầm thường'. Vì không khả thi để có được bản đồ này dữ liệu, nó sẽ bị xóa khỏi tập hợp.

Sau khi lọc xong, tám chương trình Java còn lại sẽ được thiết lập để hướng dẫn.
được đề cập và sử dụng trong khuôn khổ tạo miền và có các bài kiểm tra jUnit
được tạo ra cho họ để có thể thu thập dữ liệu mẫu.

5.1.1 Thiết bị đo lường

Quá trình đo lường rất đơn giản với quy trình trong Chương 4
thêm các câu lệnh báo cáo giá trị vào mỗi chương trình hợp lệ. Phần lớn
của các chương trình có một câu lệnh điều kiện duy nhất được thực hiện bằng công cụ dẫn đến
một tập hợp dữ liệu mẫu duy nhất. Các lớp GeoEngine và Sort là ngoại lệ,
với GeoEngine chứa ba phương pháp riêng biệt với các so sánh hợp lệ
được trang bị, trong đó GeoEngine.moveCheck() chứa hai điều kiện hợp lệ riêng biệt
các câu lệnh ngôn ngữ được thiết lập. Sort chứa cả Sort.selectSort()
và Sort.bubbleSort() đều chứa các câu lệnh điều kiện hợp lệ
được trang bị dụng cụ.

5.2 Tạo bài kiểm tra

Vì bước tạo thử nghiệm của khuôn khổ đòi hỏi sự giám sát thủ công đáng kể,
kết quả rất cụ thể đối với từng chương trình mục tiêu. Vì vậy, khác nhau
các chương trình có các bước khác nhau được thực hiện trên chúng. Các kết quả sau đây chứng minh
khía cạnh này với nhiều lớp báo cáo không có dữ liệu cho một số bước, vì những lớp đó là
được xử lý theo cách khác.

Bảng 5.2 cung cấp kết quả của Evosuite tạo ra các bài kiểm tra jUnit cho các giá trị hợp lệ
chương trình chuẩn. Bảng hiển thị số lượng các bài kiểm tra mà Evosuite tạo ra
cũng như phạm vi tuyên bố được báo cáo của các bài kiểm tra đó. Cột có nhãn 'Hữu ích'

Bảng 5.2: Kết quả tạo bài kiểm tra Evosuite

Tên lớp	Các bài kiểm tra tạo ra	Phạm vi (%) Hữu ích	
AmyFastSimplex	không có	không	
Tìm kiếm nhị phân	8 6	có	
Ví dụ1M	8	39	8
Địa lý	không	34	không có
Khối InfBlock	có/có/	không	không có
Mảng đa năng	có 3	có 44	0
Trình đọc dữ liệu QRCodeBlock	1	0	0
Loại	không có	không có	không có

biểu thị số lượng các bài kiểm tra jUnit được tạo ra có thể được sử dụng để chạy các phương pháp chứa các so sánh được đo lường. Khi Evosuite tạo ra các bài kiểm tra cho toàn bộ tệp lớp, không phải tất cả các so sánh được đo lường đều được thực hiện trong quá trình chạy thử nghiệm jUnit. Do đó, các bài kiểm tra jUnit có thể khác biệt so với các phần có liên quan của chương trình.

Đối với các chương trình MultidementionalArray, QRCodeDataBlockReader và bốn cái mà Evosuite không thể tạo ra các bài kiểm tra, thay vào đó các bài kiểm tra được tạo ra thủ công hoặc với sự hỗ trợ từ ChatGPT. Đối với BinarySearch và Example1M một của các bài kiểm tra jUnit hữu ích được chọn và tăng cường bằng cách sử dụng jUnitQuickCheck để có thể để chạy lặp đi lặp lại với các đầu vào được tham số hóa. Khi tất cả các chương trình có jUnit các bài kiểm tra đã được tăng cường bởi jUnitQuickCheck các bài kiểm tra có thể được chạy để thu thập dữ liệu mẫu để đào tạo thuật toán học PAC.

Bảng 5.3 hiển thị kết quả của 5 bộ 100 lần thực thi tham số của phương pháp đang được thử nghiệm, với tổng cộng 500 lần thực hiện phương pháp trên nhiều giá trị đầu vào khác nhau. bảng cho thấy binarySearchUsingDivision() có nhiều mẫu được thu thập hơn nhiều so với 500 lần thực hiện được tham số hóa, vì mỗi lần thực hiện đều dẫn đến kết quả được đo lường mã được gọi nhiều lần do phương thức đệ quy gọi chính nó.

Bảng 5.3: Kết quả thực hiện phương pháp

Tên phương pháp	Mẫu
AmyFastSimplex.noise()	0
binarySearchUsingDivision() ví	2611
dụ1M()	500
GeoEngine.moveCheck() 1	789
GeoEngine.moveCheck() 2	500
GeoEngine.nLOS	0
GeoEngine.nGetNSWE	487
InfBlocks.proc	0
MultidementionalArray.maxProduct() 0	
QRCodeDataBlockReader.getNextBits() 0	
Sắp xếp.selectSort() 0	
Sắp xếp.bong bóngSắp xếp 0	

phần lớn các phương pháp thử nghiệm cho thấy không có mẫu nào được thu thập. Điều này là do bởi mã được đo lường không được thực thi trong quá trình chạy trường hợp thử nghiệm. Mặc dù những phương pháp này có chứa các câu lệnh có điều kiện phù hợp với các yêu cầu của quá trình lọc, kết quả thực tế của thiết bị đo lường không khớp kỳ vọng do cách thức quy trình đo lường xử lý mã ba địa chỉ.
Thông tin chi tiết hơn về sự khác biệt này được thảo luận ở phần cuối của chương này trong Mục 5.4. Trong trường hợp mã được trang bị đã cung cấp đào tạo có thể sử dụng được dữ liệu sau đó được chuyển đến thuật toán PAC Learning để tạo ra một thuật ngữ lãnh địa.

Những kết quả này cung cấp cái nhìn sâu sắc về “RQ1: Làm thế nào để dữ liệu có thể được thu thập hiệu quả từ chương trình cung cấp mẫu cho người học PAC? ”. Với bộ lọc nghiêm ngặt và nhiều công cụ được sử dụng để thu thập dữ liệu, quá trình này có thể sai sót và không nhất quán. Thủ công sự can thiệp của người dùng là cần thiết cho hầu hết các bước, đáng chú ý nhất là quá trình lọc và việc tăng cường các bài kiểm tra jUnit cho các thực thi được tham số hóa. Với ít hơn một nửa của các chương trình chuẩn được lọc cung cấp các bài kiểm tra có thể sử dụng được, nó có vẻ không linh hoạt

đủ để được đưa vào một đường ống phân tích không phụ thuộc vào chương trình. Hơn nữa
với các bài kiểm tra được viết cho toàn bộ các chương trình chuẩn được lọc, thiết bị đo lường
thu thập dữ liệu đào tạo cho ít hơn một nửa số điều kiện hợp lệ.

5.3 Thuật toán học tập PAC

5.3.1 Dữ liệu tổng hợp

Các thí nghiệm ban đầu với thuật toán học PAC được thực hiện trên một Java tầm thường
phương pháp MinOfThree lấy ba giá trị đầu vào và so sánh chúng, trả về
giá trị tối thiểu. Các thí nghiệm sử dụng MinOfThree được gọi trực tiếp với ngẫu nhiên
giá trị của các phạm vi khác nhau. Trong khi các thử nghiệm trên bộ chuẩn mực của chương trình là
tất cả đều được thực hiện bằng cách sử dụng một phạm vi chuẩn hóa các giá trị đầu vào, các thử nghiệm trên MinOfThree
sử dụng dữ liệu tổng hợp với các biến cụ thể được đặt thành các giá trị trong phạm vi chỉ định.
Hình 5.1, 5.2 và 5.3 hiển thị dữ liệu mẫu được thu thập trên các phạm vi được chọn lọc thủ công cho
các giá trị ngẫu nhiên của các biến đầu vào. Mặc dù MinOfThree lấy ba đầu vào
biến, mỗi lần so sánh chỉ xử lý hai biến cùng một lúc, với vế trái
(trái) của phép so sánh trên trục ngang và phía bên phải (phải) trên
trục dọc.

Hình 5.1 cho thấy kết quả chạy thuật toán học PAC trên mẫu
dữ liệu từ hai bài kiểm tra riêng biệt của chương trình MinOfThree, một lần với kích thước mẫu là
50 ở cột bên trái và một lần với kích thước mẫu là 200 ở cột bên phải. Vì mỗi
thực hiện MinOfThree có ba phép so sánh, nó tạo ra ba tập hợp khác nhau
đầu ra. Mỗi hàng hiển thị dữ liệu được lấy mẫu từ một trong ba phép so sánh, với
hàng đầu tiên có một điểm dữ liệu được lấy mẫu cho mỗi lần thực hiện. Là phép so sánh thứ hai
chỉ xảy ra khi giá trị đầu tiên được đánh giá là 'Đúng' và giá trị thứ ba được đánh giá là 'Sai', mỗi giá trị

những hàng đó có số lượng mẫu chỉ bằng khoảng một nửa. Các mẫu được tô màu dựa trên
nhãn từ thiết bị liên quan đến việc liệu câu lệnh có điều kiện
dẫn đến một 'sự sụp đổ' hoặc một 'sự phân nhánh', và các đường thẳng là các siêu phẳng được tạo ra
bằng thuật toán học PAC.

Hình 5.2 áp dụng một thiết lập khác với Hình 5.1 với các giá trị đầu vào được kẹp chặt
giữa -1000 và -500 khi âm hoặc giữa 500 và 1000 khi dương. Điều này
cung cấp các giá trị rõ ràng cho các điểm phân vùng lý tưởng trong các giá trị của dữ liệu mẫu,
mặc dù như được thể hiện bằng các đường biểu diễn các siêu phẳng được chọn, không có sự tách biệt nào như vậy
được tìm thấy.

Hình 5.3 lặp lại thí nghiệm với các giá trị đầu vào được kẹp trên một phương án thay thế
việc thực hiện chương trình MinOfThree. Việc thực hiện này cũng giống như
đầu vào và cung cấp cùng một đầu ra, tuy nhiên sử dụng cấu hình bên trong khác nhau
của các câu lệnh có điều kiện, dẫn đến sự phân bố khác nhau của các điểm dữ liệu được lấy mẫu.
Chuỗi thử nghiệm này cung cấp kết quả gần nhất với kết quả mong muốn của chiều ngang
và các đường thẳng đứng tại các giá trị đầu vào quan trọng.

Việc kiểm tra các số liệu này cung cấp cái nhìn sâu sắc ban đầu về “RQ2: Lấy mẫu như thế nào
kích thước ảnh hưởng đến thuật toán học tập PAC? ”. Với kích thước mẫu nhỏ hơn có nhiều hơn
sự thay đổi trong dữ liệu được lấy mẫu, được cân bằng bởi số lượng cặp ít hơn phải được
tách biệt. Kích thước mẫu lớn hơn cung cấp dữ liệu chứng minh chương trình rõ ràng hơn
hành vi, với khả năng cao hơn của các cặp gần nhau và khó khăn hơn
để tách ra. Những điểm khó tách ra này xảy ra thường xuyên hơn khi cả LHS
và biến RHS gần như bằng nhau. Dữ liệu có số lượng lớn khó phân tách
điểm có nhiều khả năng yêu cầu số lượng siêu phẳng lớn hơn để tạo thành một
sự tách biệt.

Các hình vẽ cũng cung cấp các ví dụ về siêu phẳng dọc theo đường X = Y. Do

với những hạn chế của mã ba địa chỉ, bất kỳ siêu phẳng nào dọc theo đường này sẽ hoàn hảo tách biệt tất cả các điểm. Mặc dù không có mô hình xác định nào xuất hiện, nhưng có vẻ như với số lượng điểm dữ liệu mẫu lớn hơn, phép chiếu không gian kép sẽ hiệu quả hơn có khả năng tạo ra một siêu phẳng dọc theo X = Y dẫn đến sự tách biệt tầm thường này.

5.3.2 Đối xứng biến đầu vào

Vì thuật toán học PAC không đối xứng về mặt LHS và RHS biến, luận án này đã kiểm tra kết quả từ việc chuyển đổi phía hiệu quả của biến khi cung cấp dữ liệu mẫu cho thuật toán học tập. Trong khi đó, bằng cách mặc định thuật toán lấy các biến LHS và RHS và gán chúng cho trục ngang và trục dọc tương ứng, thì việc gán trục trái cũng có giá trị như nhau biến đổi theo trục tung và ngược lại. Mặc dù sử dụng cùng các giá trị dữ liệu mẫu và thực hiện cùng một kỹ thuật, điều này mang lại kết quả khác với không gian kép phép chiếu có hiệu ứng khác nhau đối với mỗi trục.

Hình 5.4 cho thấy bốn bộ kết quả khác nhau từ việc áp dụng học tập PAC thuật toán cho chương trình ExampleM1. Cột bên trái hiển thị thuật toán mặc định kết quả, cột trung tâm hiển thị kết quả thuật toán lật ngược và bên phải cột cung cấp lớp phủ của cả hai, cho thấy các điểm màu xanh lam ở cột bên trái tương ứng với các điểm màu vàng ở cột giữa. Quá trình này có thể được nhìn thấy dẫn đến một tập hợp các siêu phẳng khác nhau tùy thuộc vào việc nó là mặc định hay bị lật hành vi. Đáng chú ý nhất là hàng thứ ba chứng minh rằng ngay cả với đầu vào giống hệt nhau giá trị kết quả cuối cùng có thể thay đổi rất nhiều.

Kết quả của việc lật ngược các mặt biến đầu vào được thu thập để cố gắng giải quyết sự không nhất quán của các siêu phẳng được tạo ra được thể hiện trong các hình trước. Tuy nhiên, như Hình 5.4 cho thấy ở cột thứ ba, các siêu phẳng chồng lên nhau không hiển thị rõ ràng

sự hội tụ hướng tới một kết quả thống nhất. Mặc dù sự tồn tại của các siêu phẳng dọc theo X = Y vẫn đáng chú ý và được củng cố bằng cách lật ngược các biến đầu vào, không rõ ràng sự tách biệt theo chiều dọc hoặc chiều ngang đều thể hiện ở cả bản gốc và bản lật ngược kết quả.

Thuật toán được chạy trên nhiều chương trình mẫu với cùng kết quả không chứng minh được lợi ích rõ ràng khi cố gắng thống nhất kết quả từ bản gốc và bản lật ngược biến đầu vào. Tuy nhiên, như dữ liệu sau này cho thấy thuật toán học tập PAC thực hiện khác nhau tùy thuộc vào chương trình đầu vào được đánh giá khi đánh giá lật biến. Điều này sẽ được xem xét kỹ hơn ở phần sau.

5.3.3 Kết quả chuẩn

Hình 5.5 cung cấp kết quả của bốn lần chạy riêng biệt của thuật toán học PAC trên các tập dữ liệu mẫu khác nhau từ phương pháp binarySearchUsingDivision. Đối với mỗi lần chạy thuật toán, một đường chéo rõ ràng được tìm thấy để phân tách các điểm với các nhãn khác nhau, dẫn đến việc đánh giá chương trình chỉ mang tính quan hệ thuần túy. Mặc dù các đường thẳng đứng đã được tìm thấy nhưng chúng có mức độ ưu tiên thấp hơn trong tập tham lam bao phủ và như vậy không được coi là quan trọng. Hơn nữa những đường thẳng đứng này luôn xuất hiện phía trên các giá trị của biến tương ứng với trục X.
Với kiến thức bên ngoài về hành vi của thuật toán tìm kiếm nhị phân, một giá trị như vậy sẽ không cung cấp một phân vùng có ý nghĩa cho phân tích chương trình. Thay vào đó là một đường thẳng đứng tương ứng với phân vùng tại X = 0 sẽ là mong muốn.

Bảng 5.4, 5.5, 5.6 chứa các đoạn trích dữ liệu từ việc sử dụng thuật toán học tập PAC với dữ liệu được lấy mẫu từ chương trình BinarySearch. Bốn cột đầu tiên của mỗi bảng cung cấp các giá trị cho số lượng siêu phẳng được đánh giá là hướng đường thẳng như được mô tả trong Phần 4.4. Cột Chính liệt kê hướng nào được chọn bởi

Bảng 5.4: Kết quả thuật toán PAC của BinarySearch (Kích thước mẫu 50, K=15)

Dọc	Ngang	Đường chéo	Không rõ	Tổng	chính		
0	1	1	2	Không rõ ràng	4		
0	0	1	3	Đường chéo	4		
0	0			Đường chéo	3		
0	0	1 2	2 2	Đường chéo	4		
0	0	2	2	Không rõ ràng	4		
0	0	3	2	Đường chéo	5		
0	0	2	0	Đường chéo	2		
1	1	0	3	Không rõ ràng	5		
0	1	2	1	Đường chéo	4		
0	0	1	3	Đường chéo	4		

Bảng 5.5: Kết quả thuật toán PAC của BinarySearch (Kích thước mẫu 50, K=15, Lật Biến đầu vào)

Dọc	Ngang	Đường chéo	Không rõ	Tổng	chính		
0	1	1	0	Đường chéo	2		
0	0	2	0	Đường chéo	2		
0	0	2	1	Đường chéo	3		
0	0	1	0	Đường chéo	1		
8	2	0		Nằm ngang	11		
0	0	1	1 1	Đường chéo	2		
1	0	2	1	Đường chéo	4		
0	0	1	0	Đường chéo	1		
0	0	2	0	Đường chéo	2		
0	0	1	1	Đường chéo	2		

thuật toán bao phủ tập tham lam đầu tiên và tương ứng với hướng của đường thẳng có tác dụng tách biệt số điểm lớn nhất trong phép chiếu không gian kép.

Khi so sánh Bảng 5.4 và 5.5 để xác định hiệu quả của việc đánh giá ban đầu các phép gán biến cho các biến đầu vào bị đảo ngược, có một sự khác biệt rõ ràng giữa hai bộ dữ liệu. Bộ siêu phẳng ban đầu trong Bảng 5.4 có số lượng lớn hơn nhiều của các dòng 'Không rõ ràng' không được căn chỉnh theo trục hoặc theo đường chéo. Tương ứng dữ liệu bị lật có ít siêu phẳng được tạo ra trung bình với mức giảm đáng kể

Bảng 5.6: Kết quả thuật toán PAC của BinarySearch (Kích thước mẫu 200, K=15)

Dọc	Ngang	Đường chéo	Không rõ	Tổng chính		
1	2	0	3	Không rõ ràng		6
0	0	1	2	Đường chéo		3
0	0	3	0	Đường chéo		3
1	0	1	0	Đường chéo		2
0	0	1	0	Đường chéo		1
1	0	1	0	Đường chéo		2
0	0	2	0	Đường chéo		2
0	0	2	0	Đường chéo		2
0	0	1	1	Đường chéo		2
0	2	1	2	Không rõ ràng		5

Bảng 5.7: Kết quả thuật toán PAC của Example1M (Kích thước mẫu 50, K=15)

Dọc	Ngang	Đường chéo	Không rõ	Tổng chính		
1	0	1	0	Đường chéo		2
0	1	1	0	Đường chéo		2
3	2	1	1	Đường chéo		7
2	1	1	1	Đường chéo		5
0	2	2	1	Đường chéo		5
1	0	1	0	Đường chéo		2
1	1	2	1	Đường chéo		4
0	0		0	Đường chéo		3
0	0	3 2	0	Đường chéo		2
0	0	2	0	Đường chéo		2

số dòng 'Không rõ ràng'.

Sự khác biệt này cho thấy rằng khi xem xét RQ1, nên cung cấp
Thuật toán học PAC với dữ liệu trong cả cấu hình ban đầu cũng như
cấu hình đảo ngược vì hành vi chỉ có thể học được ở một trong hai. Tuy nhiên,
sử dụng cả hai cấu hình cùng nhau để xác định bất kỳ đặc điểm thống nhất nào không
có vẻ khả thi vì hành vi của hai kết quả có vẻ không liên quan.

Bảng 5.7, 5.8, 5.9 chứa cùng dữ liệu được thu thập từ chương trình Example1M.
Hành vi của chương trình này rất có thể chỉ liên quan đến hầu hết mọi thứ

Bảng 5.8: Kết quả thuật toán PAC của Ví dụ 1M (Kích thước mẫu 50, K=15, Đầu vào đảo ngược Biến)

Dọc	Ngang	Đường chéo	Không rõ	Tổng chính		
	0	1	1	Đường chéo	2
0 0	0	2	0	Đường chéo	2
0	0	2	1	Đường chéo	3
0	0	1	0	Đường chéo	1
0	0	2	0	Đường chéo	2
0	0		0	Đường chéo	1
1	0	1	1	Đường chéo	5
0	0	3 1	1	Đường chéo	2
0	0	3	0	Đường chéo	3
3	0	0	4	Không rõ ràng	7

Bảng 5.9: Kết quả thuật toán PAC của Example1M (Kích thước mẫu 200, K=15)

Dọc	Ngang	Đường chéo	Không rõ	Tổng chính		
0	0	3	0	Đường chéo	3
0	0	1	0	Đường chéo	1
1	0	3	1	Đường chéo	5
1	0	2	2	Đường chéo	5
0	0		0	Đường chéo	1
0	0	1	0	Đường chéo	3
0	0	3	0	Đường chéo	3
0	0	3 2	0	Đường chéo	2
0	3	2	0	Đường chéo	5
13	4	0	2	Thẳng đứng	19

chạy thuật toán học PAC dẫn đến một siêu phẳng tương ứng với một chẩn đoán dòng onal. Sự khác biệt giữa các biến ban đầu và biến bị lật được thấy trong BinarySearch không có ở đây vì dữ liệu mẫu được phân bổ đồng đều trên cả LHS và các biến RHS.

So sánh Bảng 5.7 và 5.9 cung cấp thêm xác nhận về trước đó kết luận liên quan đến RQ2. Cả số lượng siêu phẳng được chọn cho cặp sep- sự phân bố và hướng của các đường liên quan xuất hiện phân bố đều bất kể

Bảng 5.10: Kết quả thuật toán GeoEngine PAC (Kích thước mẫu 50, K=15)

Dọc Ngang Đường chéo Không rõ Tổng chính					
0	0	0	1	Không rõ ràng	1
1	0	0	0	Thẳng đứng	1
	0	0		Không rõ ràng	1
0 2	0	0	1	Thẳng đứng	2
1	1	0	0 0	Nằm ngang	2
0	2	0	1	Không rõ ràng	3
1	0	0	0	Thẳng đứng	1
0	0	0	1	Không rõ ràng	1
0	1	0	1	Không rõ ràng	2
0	0	0	2	Không rõ ràng	2

Bảng 5.11: Kết quả thuật toán GeoEngine PAC (Kích thước mẫu 50, K=15, Đầu vào đảo ngược Biến)

Dọc Ngang Đường chéo Không rõ Tổng chính					
0	0	0	1	Không rõ ràng	1
0	0	0	1	Không rõ ràng	1
1	0	0	1	Không rõ ràng	2
0	1	0	1	Không rõ ràng	2
0	0		0	Đường chéo	1
0	0	1	1	Không rõ ràng	1
0	0	0	1	Không rõ ràng	1
0	0	0 0	1	Không rõ ràng	1
0	0	0	1	Không rõ ràng	1
1	0	0	0	Thẳng đứng	1

của kích thước mẫu. Mặc dù không được hiển thị ở đây, các thử nghiệm bổ sung được thực hiện trong suốt quá trình đánh giá sử dụng nhiều kích cỡ mẫu khác nhau, cho kết quả tương tự nhau.

Kích thước mẫu trên 500 dường như không mang lại lợi ích nhưng lại làm tăng thời gian thực hiện của thuật toán học tập phụ thuộc đáng kể vào độ phức tạp của chương trình. Như vậy kích thước mẫu dường như không có bất kỳ tác động có thể đo lường nào đến kết quả của PAC thuật toán học tập hoặc khả năng tạo ra miền của nó.

Cuối cùng, các bảng 5.10, 5.11, 5.12 chứa dữ liệu thu thập được từ GeoEngine

Bảng 5.12: Kết quả thuật toán GeoEngine PAC (Kích thước mẫu 200, K=15)

Dọc Ngang Đường chéo Không rõ Tổng chính					
0	0	0	1	Không rõ ràng	1
0	0	1	0	Đường chéo	1
	0	0		Không rõ ràng	2
1	0	0	1 1	Không rõ ràng	1
0 0	1	0	1	Không rõ ràng	2
1	0	0	0	Thẳng đứng	1
1	0	0	0	Thẳng đứng	1
0	0	0	1	Không rõ ràng	1
1	0	0	1	Thẳng đứng	2
1	0	0	1	Không rõ ràng	2

chương trình. Một sự khác biệt đáng chú ý trong chương trình này là số lượng siêu phẳng ít hơn nhiều cần thiết để tách các điểm mẫu. Các bảng cho thấy một số lần chạy dẫn đến các đường thẳng đứng riêng biệt phân tách các điểm dữ liệu có vẻ đầy hứa hẹn, tuy nhiên lý do đằng sau điều này chỉ ra nhiều hơn về thuật toán học tập PAC hành vi trái ngược với bất kỳ điều gì học được từ chương trình đang được phân tích.

Mặc dù được chạy trên dữ liệu được thu thập từ cùng một phạm vi giá trị đầu vào như các chương trình chuẩn mực khác (-1000 đến 1000), các điểm dữ liệu được lấy mẫu bao phủ một phạm vi lớn hơn nhiều (khoảng từ 0 đến 1.500.000.000). Khi được cung cấp một phạm vi rộng như vậy phạm vi giá trị tiềm năng phép chiếu không gian kép tạo ra các siêu phẳng có độ dốc thay đổi đáng kể hơn. Tương ứng, các giá trị chặn mà miền phép chiếu được sử dụng để chọn phân vùng cũng có phương sai cao hơn nhiều. Với độ dốc thay đổi trên một phạm vi rộng hơn, số lượng đường được đánh dấu là thẳng đứng nhiều hơn, vì bất kỳ giá trị nào trên hoặc dưới ngưỡng nhất định đều được coi là theo chiều dọc. Điều này khác nhau từ hành vi của các siêu phẳng tương ứng với các đường ngang luôn luôn bị giới hạn trong cùng một phạm vi độ dốc có tâm là giá trị bằng không.

5.4 Thách thức

Sau khi dữ liệu đã được xử lý bằng phép chiếu không gian kép, cần phải
diễn giải các dòng và vùng kết quả để có được phương pháp phù hợp cho
phép chiếu lên miền vị ngữ. Điều này đặt ra nhiều thách thức phải
được giải quyết một cách độc đáo, vì công trình trước đây sử dụng các kỹ thuật có độ phức tạp cao hơn
cung cấp các phương pháp tạo ra bất biến mà chúng ta không có. Trong công việc
bởi Sharma et al. [11] các vùng kết quả từ trình học PAC được sử dụng trực tiếp
bất biến như miền được sử dụng cho các phân tích tĩnh của họ cho phép các vị từ phức tạp
có thể biểu diễn chính xác vùng được tạo ra. Vì mục tiêu của công việc này là
tạo ra các phân vùng trong miền vị ngữ, cũng cần phải tìm một ý nghĩa
cách để chiếu vùng được tạo ra lên miền này.

Một thách thức bổ sung phát sinh từ bản chất của các công cụ được sử dụng để thu thập
dữ liệu từ các chương trình đang được phân tích. Trong công trình của Sharma và cộng sự, mỗi nhánh đang được
được đánh giá có thể có nhiều câu lệnh có điều kiện có thể có. Do sử dụng
SOOT [16] trong công trình này và bản chất mã ba địa chỉ của Jimple, nội bộ của SOOT
biểu diễn mã, các câu lệnh có điều kiện có sẵn để đánh giá là rất xa
đơn giản hơn. Trong mã nguồn gốc, một điều kiện có thể bao gồm bất kỳ số lượng biến nào
và các giá trị hằng số được so sánh với nhau, ví dụ trong một câu lệnh Nếu
(X ≥ 50 VÀ X ≤ 100) trong đó giá trị bị giới hạn giữa 50 và 100 đối với tất cả các giá trị đúng
hành quyết. Tuy nhiên, một khi được thể hiện trong Jimple, điều này sẽ dẫn đến hai
các câu lệnh có điều kiện đang được đánh giá, một câu lệnh kiểm tra xem X có bằng hoặc lớn hơn không
hơn 50 và một cái khác kiểm tra xem nó bằng hay nhỏ hơn 100. Bằng cách đánh giá những
hai điều kiện riêng biệt thông tin rằng chúng có liên quan không phải là ngay lập tức
có sẵn cho thuật toán học PAC.

Khi xem xét sự so sánh của hai giá trị biến, trường hợp có nhiều sự khác biệt nhất
Mặc dù chúng ta có thể linh hoạt sử dụng dữ liệu, nhưng dữ liệu cũng bị giới hạn bởi số lượng thao tác có sẵn.
Với các phép so sánh lớn hơn và nhỏ hơn, cả hai tập dữ liệu kết quả đều có các vùng
được tách biệt hoàn toàn bằng cách biểu diễn tuyến tính của hai giá trị là
bằng nhau. So sánh bằng nhau và bất bằng nhau cũng có sự tách biệt xung quanh đó
dòng, tách các giá trị trên dòng với các giá trị không nằm trên dòng.

Có các tập dữ liệu kết quả có thể tách biệt được theo đường thẳng bình đẳng dẫn đến
các vùng phần lớn không bị giới hạn cũng như một số lượng lớn các mối quan hệ tách biệt
các dòng được tạo ra. Các dòng quan hệ này thường tách biệt hiệu quả hai dữ liệu
nhưng khó có thể chuyển thành các phân vùng vị từ hữu ích, vì mỗi phân vùng chỉ có thể
cung cấp một bất biến cho một biến duy nhất chứ không phải là mối quan hệ giữa hai biến.
Sharma et al. [11] giải quyết vấn đề này trong công trình của họ bằng cách chọn cụ thể các đường có độ dốc
có thể được sử dụng như các đường thẳng dọc theo một biến duy nhất và sau đó dịch chuyển chúng
các đường thẳng ở vị trí thích hợp. Ví dụ, tìm một đường ngang có
độ dốc = 0 tại Y = 3 và dịch chuyển đường thẳng đó thành Y = 10 tại đó tất cả các điểm dữ liệu là
tách biệt sạch sẽ tại thời điểm này. Tuy nhiên, kỹ thuật này có thể được áp dụng tương tự bởi
ban đầu tạo ra các đường thẳng tại các vị trí đó trước khi thực hiện phép chiếu không gian kép,
và do đó kỹ thuật này không có lợi cho công việc này.

5.4.1 Tạo tên miền

Về RQ3: “Thuật toán học PAC có thể tạo ra miền trừu tượng của thuật ngữ không
trên một tập hợp các chương trình chuẩn”, những thách thức này cho thấy rằng khuôn khổ sử dụng
thuật toán học tập PAC không phù hợp cho mục đích này trong phạm vi giới hạn của
công việc này. Bộ chương trình giới hạn mà cả hai đều nằm trong bộ chuẩn mực sau
lọc và tạo dữ liệu đào tạo sau khi đo lường bị giới hạn ở ba phương pháp.

Trong ba phương pháp khả thi để sử dụng thuật toán học PAC, mỗi phương pháp trả về kết quả nhất quán nêu rõ phân tích quan hệ là cần thiết để cải thiện độ chính xác của SA. Trong trường hợp thuật toán học PAC tạo ra các siêu phẳng có thể được sử dụng để phân vùng, các giá trị dự kiến không có ý nghĩa cũng như có lợi.

Chương trình Example1M được chạy nhiều lần thông qua thuật toán học tập PAC nhịp điệu với kích thước mẫu lớn cho đến khi một số siêu phẳng được tạo ra có thể được sử dụng để điền vào một miền duy nhất. Kết quả của việc này là miền (∞, 110](110, 0)01(1, 35)[35,∞) với hai phân vùng được tạo ra, tại -110 và 35, cùng với các phân vùng ở 0 và 1 luôn được cho là có lợi.

Sau đó, phân tích được thực hiện trên chương trình Example1M bằng cách sử dụng cả hai chương trình này tên miền và một số tên miền khác đã tồn tại trước đó được chọn ngẫu nhiên. Đối với mọi so sánh giữa miền được tạo ra và miền được chọn ngẫu nhiên, hai người thực hiện phân tích với độ chính xác như nhau. Do đó, vấn đề do RQ3 đặt ra có vẻ như không được trả lời bằng luận án này, và nếu thử với kết quả đó được cung cấp độ chính xác không được cải thiện.

5.5 Mối đe dọa đến tính hợp lệ

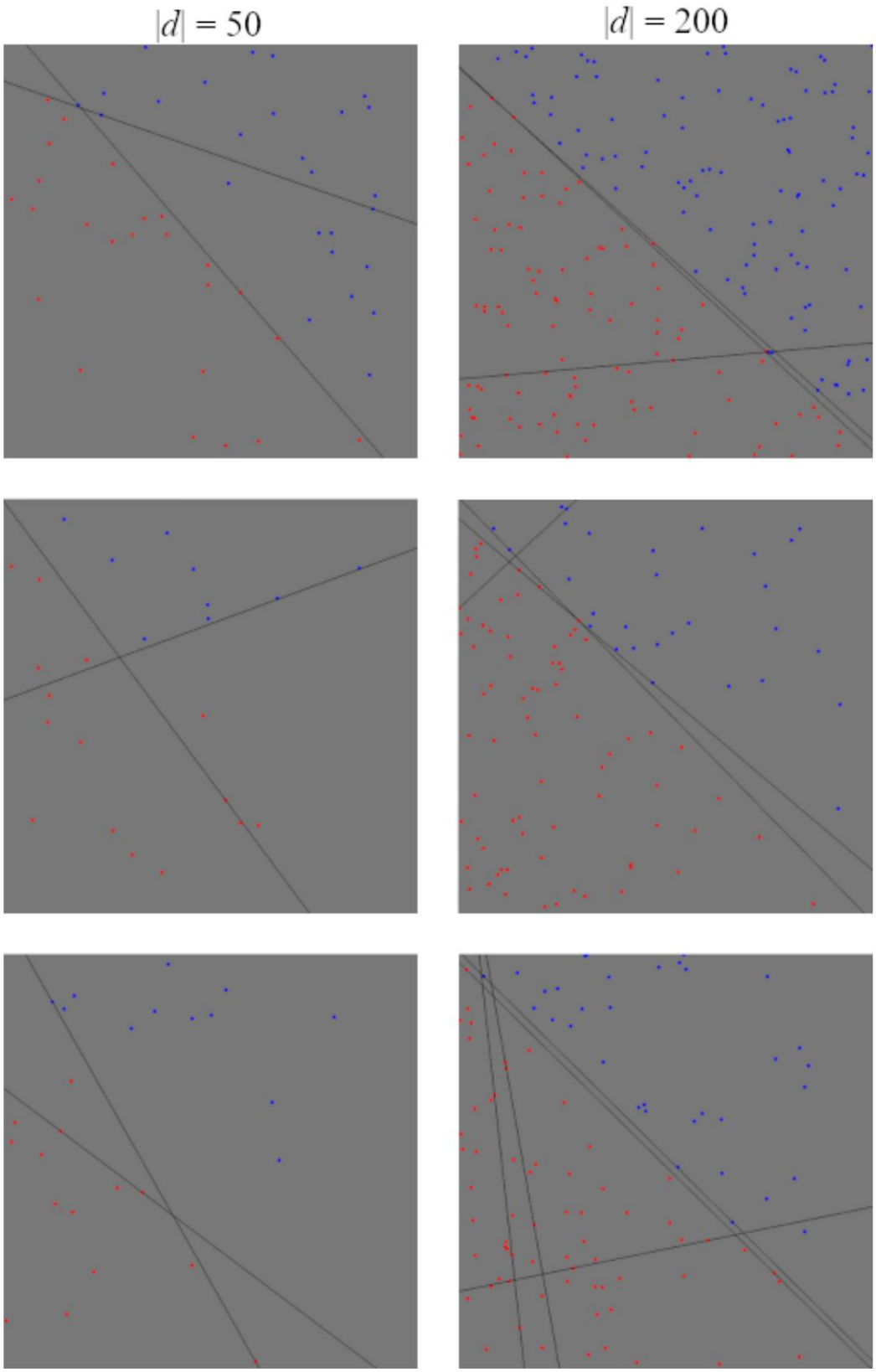
5.5.1 Nội bộ

Do tỷ lệ thành công thấp khi sử dụng Evosuite để tạo thử nghiệm nên có khả năng là công cụ không được vận hành trong những hoàn cảnh lý tưởng và có thể dẫn đến kết quả không tối ưu jUnit kiểm tra để thu thập dữ liệu. Điều này được giảm nhẹ bằng quy trình xem xét thủ công của jUnit kiểm tra cùng với việc tăng cường thủ công xảy ra sau đó. Tương tự như vậy các giá trị đầu vào được tham số hóa được tạo ra bởi jUnitQuickCheck có thể không chỉ ra

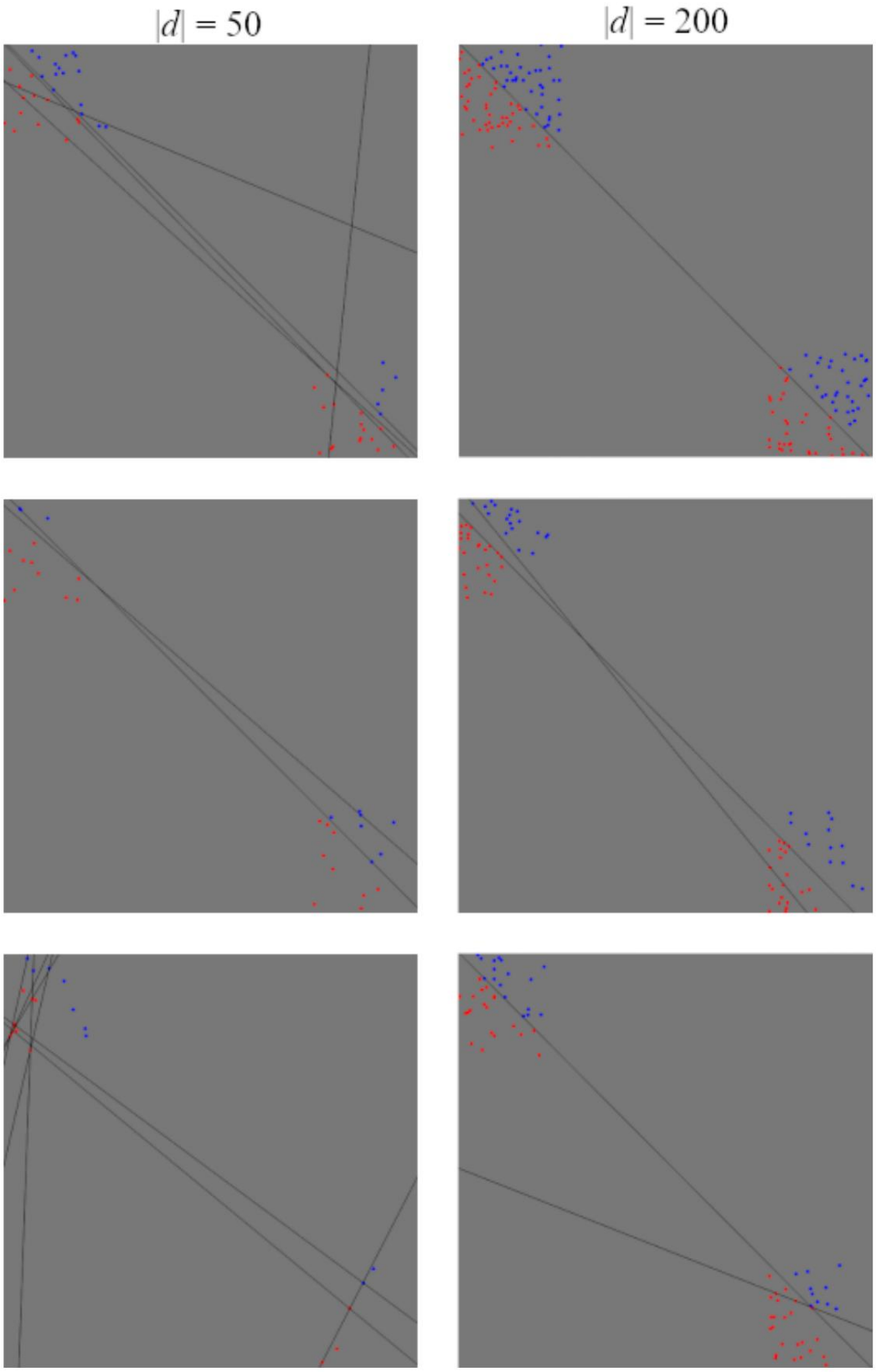
đầu vào của người dùng thực tế và có thể ảnh hưởng tiêu cực đến tính hợp lệ của dữ liệu học tập được lấy mẫu từ họ.

5.5.2 Bên ngoài

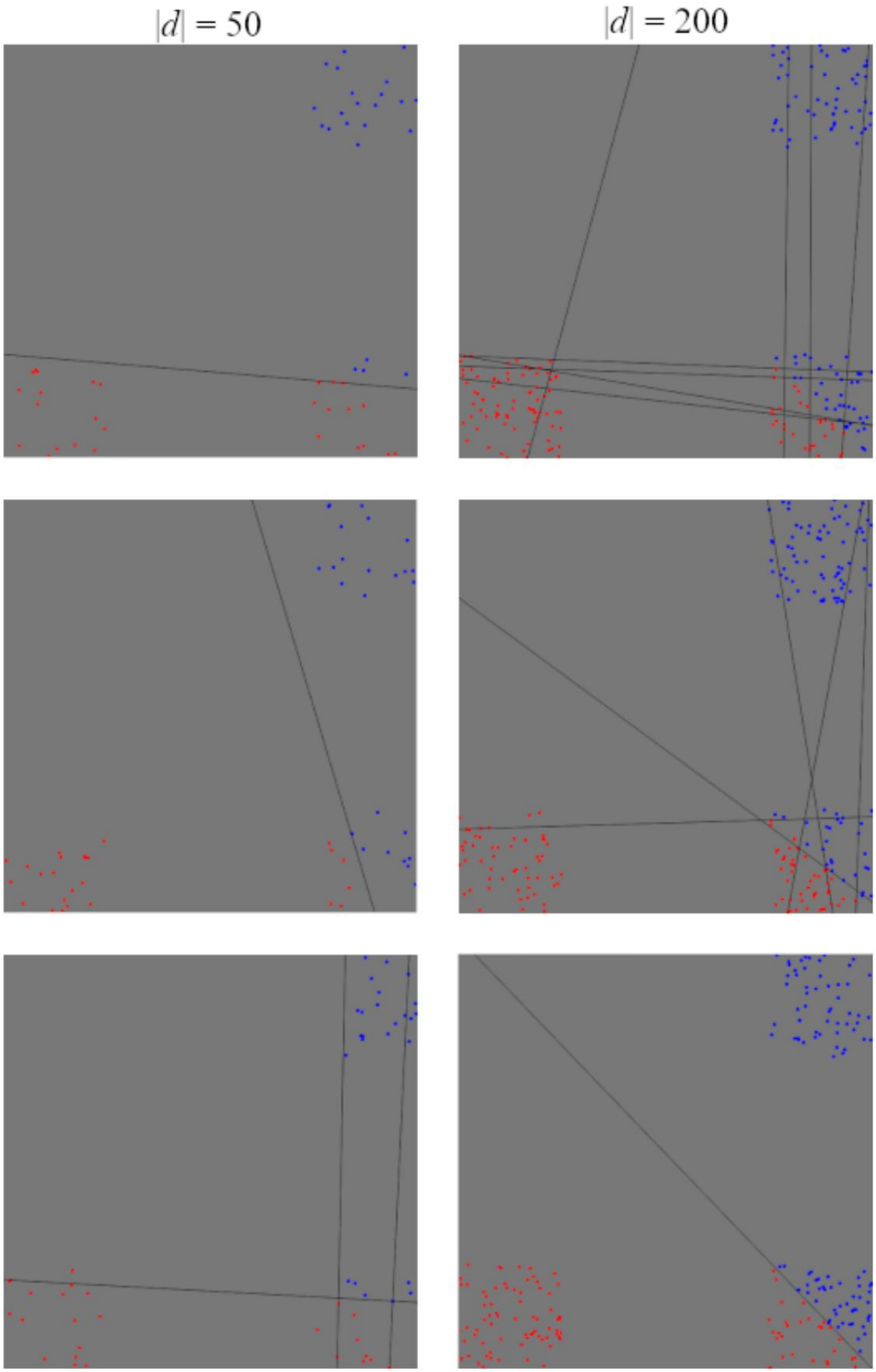
Các chương trình được sử dụng trong chuẩn mực đánh giá có thể không đại diện cho các vấn đề sẽ được hưởng lợi từ hình thức phân tích này, rõ ràng là do số lượng lớn đã được lọc thủ công. Quá trình lọc làm giảm thiểu điều này vấn đề mặc dù các chương trình bổ sung vượt qua quá trình lọc chắc chắn sẽ cải thiện độ tin cậy của kết quả. Việc tìm ra những chương trình như vậy chứng tỏ là một thách thức và không thể thực hiện được trong quá trình thực hiện công việc này.



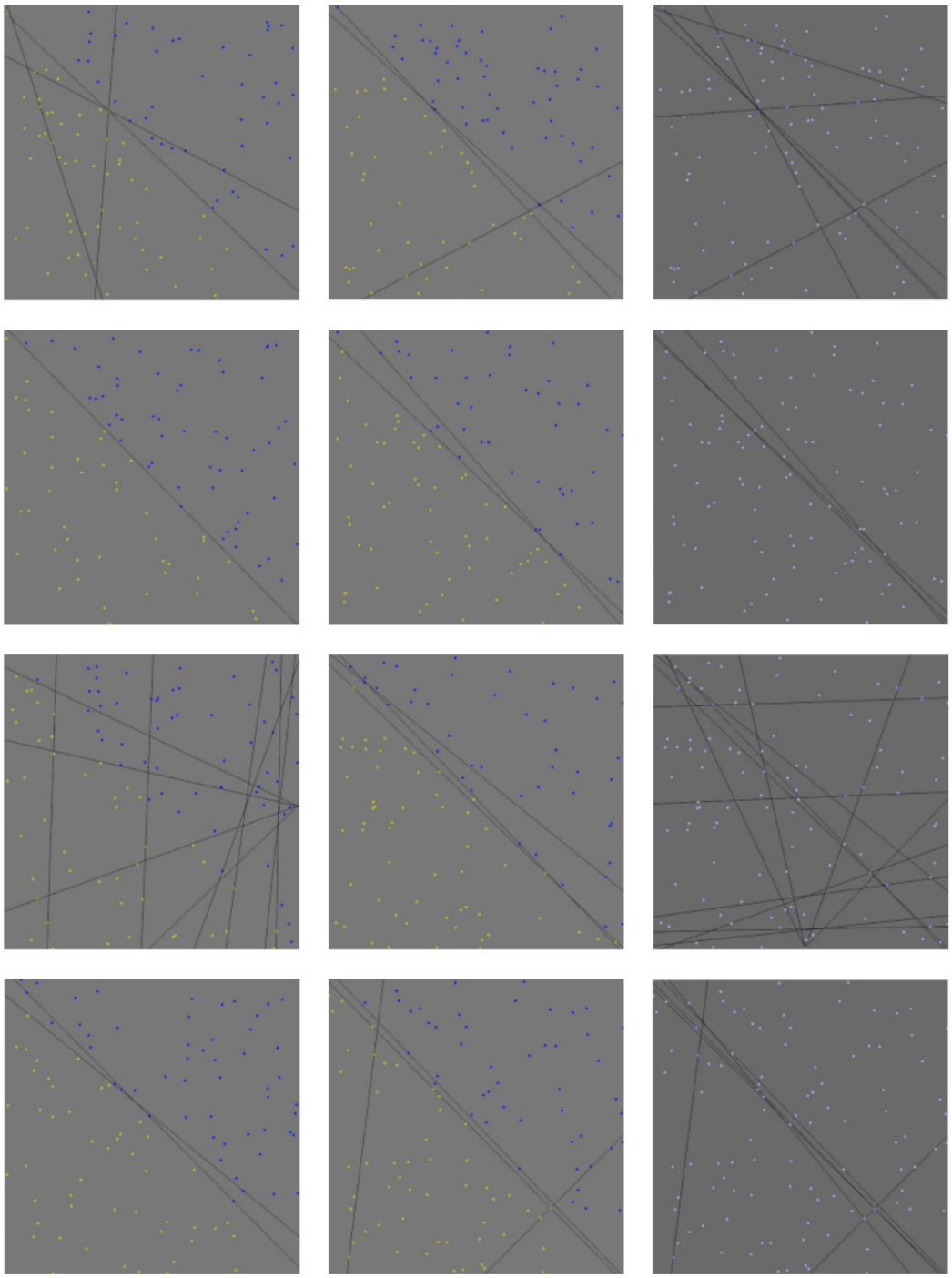
Hình 5.1: Kết quả của PAC Learning chương trình MinOfThree với các giá trị đầu vào ngẫu nhiên trong khoảng từ -1000 đến 1000



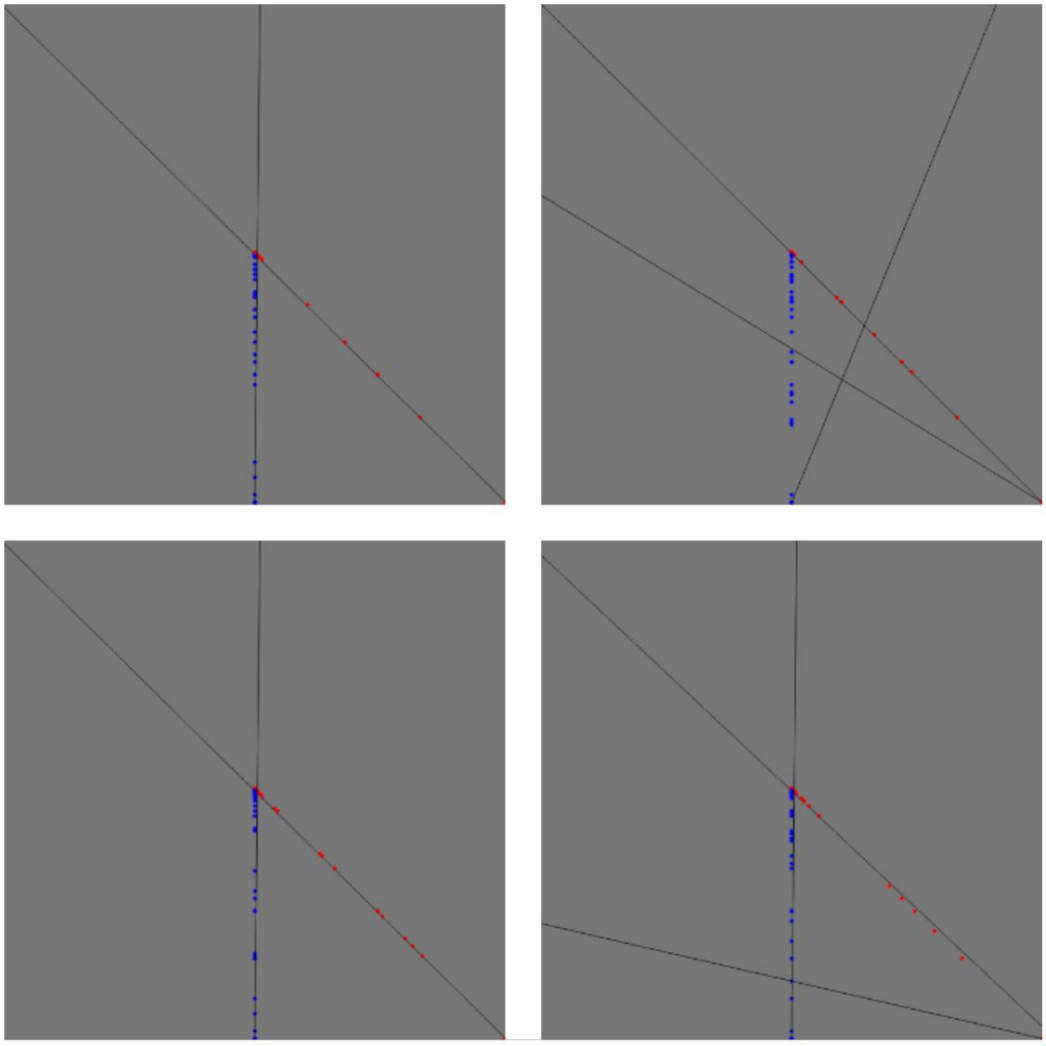
Hình 5.2: Kết quả của PAC Learning chương trình MinOfThree với các giá trị đầu vào ngẫu nhiên trong khoảng từ -1000 đến -500 hoặc trong khoảng từ 500 đến 1000



Hình 5.3: Kết quả của PAC Learning một chương trình MinOfThree thay thế với các giá trị đầu vào ngẫu nhiên trong khoảng từ -1000 đến -500 hoặc trong khoảng từ 500 đến 1000



Hình 5.4: Kết quả của PAC Learning chương trình ExampleM1 với phép gán LHS/RHS ban đầu và đảo ngược.



Hình 5.5: Kết quả học PAC của chương trình BinarySearch.

CHƯƠNG 6

PHẦN KẾT LUẬN

Nghiên cứu được tiến hành đã mang lại một số phát hiện quan trọng góp phần vào
hiểu rõ hơn về thế hệ miền phân tích tĩnh và xác định cơ hội mới
cơ hội cho nghiên cứu trong tương lai.

Đầu tiên, vấn đề thu thập dữ liệu thời gian chạy từ các chương trình tùy ý vẫn là một
vấn đề chưa được giải quyết. Cần có một cuộc điều tra riêng biệt để giải quyết triệt để tình trạng thiếu hụt này
vì hiện tại nó đặt ra một trở ngại đáng kể đối với khả năng lặp lại của phương pháp luận
được sử dụng trong nghiên cứu này. Nếu không có cách tiếp cận đáng tin cậy để tạo ra các bài kiểm tra cho một pro- tùy ý
gram và nhận dữ liệu mẫu để đào tạo thuật toán học tập PAC khuôn khổ
đòi hỏi nỗ lực thủ công đáng kể, khiến cách tiếp cận này kém khả thi hơn. Tuy nhiên,
công trình này đã khám phá ra rằng các công cụ như jUnitQuickcheck và ChatGPT có thể hỗ trợ
yêu cầu tạo trường hợp thử nghiệm. Đây có thể là một hướng đi tiềm năng để khám phá thêm
và cải thiện trong lĩnh vực này, với tiềm năng tự động hóa hoàn toàn khuôn khổ
trong tương lai.

Thứ hai, thuật toán học tập PAC thể hiện mức độ linh hoạt cao trong
về mặt kích thước mẫu. Đánh giá cho thấy thuật toán có thể hoạt động hiệu quả với
không có sự phụ thuộc rõ ràng vào kích thước mẫu, ngoại trừ hiệu quả thời gian chạy giảm
đối với kích thước lớn. Điều này cho thấy thuật toán học tập PAC tự nó là mạnh mẽ và
có khả năng thích ứng, có khả năng xử lý nhiều loại dữ liệu đầu vào ở cấp độ trừu tượng

được sử dụng trong luận văn này.

Công trình này cũng tiết lộ phạm vi hẹp các chương trình có thể được hưởng lợi từ phương pháp tiếp cận được đề xuất. Do những hạn chế do mã ba địa chỉ và phân tích dựa trên số nguyên, số lượng chương trình từ chuẩn mực có sẵn với đặc điểm cần thiết bị hạn chế đáng kể. Điều này hạn chế khả năng áp dụng của khuôn khổ cho các tập hợp chương trình lớn. Hơn nữa, tính khả dụng của chương trình hạn chế này ngăn chặn mọi đánh giá xác định những thay đổi trong độ chính xác của miền được tạo ra.

Về mặt phép chiếu của các siêu phẳng lên các miền vị ngữ, người ta đã tìm thấy có thể khả thi khi sử dụng phân loại được đề xuất, sử dụng phạm vi độ của siêu phẳng góc. Tuy nhiên, việc đánh giá độ chính xác của các miền vị ngữ được tạo ra vẫn còn phải được khám phá. Để giải quyết vấn đề này, bộ chuẩn mực hiện tại nên là được tăng cường bằng các chương trình chuẩn mực với các hành vi không liên quan hoặc PAC kỹ thuật học tập và trình chiếu cần phải thay đổi.

Tóm lại, trong khi nghiên cứu đã mang lại những hiểu biết có giá trị và thúc đẩy hiểu biết về thế hệ miền phân tích tĩnh, nó cũng đã làm nổi bật một số các lĩnh vực cho công việc trong tương lai. Những hạn chế do mã ba địa chỉ và các miền vị ngữ rời rạc có thể ngăn cản khả năng thực hiện hai chiều phân tích hình học, và do đó khả năng học được các khái niệm hình học này với PAC thuật toán học tập có vẻ không khả thi trên tập hợp các chương trình nhất định. Tuy nhiên, điều này luận án đề xuất một số gợi ý để cải thiện cách tiếp cận với hy vọng tìm thấy kết quả hứa hẹn hơn.

6.1 Công việc trong tương lai

Những phát hiện từ nghiên cứu này đã mở ra nhiều hướng đi cho các cuộc khám phá trong tương lai và cải tiến trong lĩnh vực tạo miền trừu tượng tự động.

Một trong những lĩnh vực chính cho công việc trong tương lai là cải thiện tự động hóa để thử nghiệm tạo trường hợp để thu thập đủ mẫu dữ liệu hiệu quả hơn. Sử dụng các công cụ hiện có làm cho quá trình hiện tại tốn nhiều công sức. Bước này có thể được hưởng lợi từ sự phát triển của các công cụ hoặc kỹ thuật tự động hóa tinh vi hơn. Ngoài ra, nghiên cứu trong tương lai có thể khám phá các phương pháp khác nhau để thực thi mã tùy ý để thu thập dữ liệu mẫu, có khả năng khắc phục một số hạn chế gặp phải trong nghiên cứu này.

Một lĩnh vực khám phá đầy hứa hẹn khác là phát triển một phương pháp luận cho đánh giá toàn bộ hành vi của các biến tại một vị trí chương trình nhất định. Bằng cách nắm bắt thông tin phong phú hơn liên quan đến các mối quan hệ biến đổi có tiềm năng cho một máy thuật toán học tập để tổng hợp các kết quả tốt hơn. Điều này có thể dẫn đến ít hạn chế hơn tiêu chí lựa chọn áp đặt bởi mã ba địa chỉ, được tìm thấy để hạn chế số lượng chương trình phù hợp. Ngoài ra, có tiềm năng mở rộng chương trình học PAC thuật toán tự nó xử lý nhiều hơn hai biến bằng cách sử dụng đa chiều phép chiếu không gian kép. Điều này có thể tăng cường tính linh hoạt và khả năng áp dụng của Thuật toán học PAC.

Cuối cùng, nghiên cứu trong tương lai cũng có thể điều tra các phương pháp học tập PAC khác nhau gies. Ví dụ, việc tìm kiếm các giới hạn hình chữ nhật của các mẫu dữ liệu có thể cung cấp một cách tiếp cận thay thế để tạo ra miền và cung cấp những hiểu biết mới về khả năng của việc học PAC trong bối cảnh này, đồng thời cũng tránh được các vấn đề phát sinh trong chiếu các siêu phẳng lên miền vị từ.

TÀI LIỆU THAM KHẢO

[1] Eric B Baum. Về việc học hợp nhất của các không gian nửa. Tạp chí phức tạp, 6(1):67–101, 1990.

[2] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler và Manfred K War-muth. Khả năng học hỏi và chiều kích vapnik-chervonenkis. Tạp chí của ACM (JACM), 36(4):929–965, 1989.

[3] Nader H Bshouty, Sally A Goldman, H David Mathias, Subhash Suri và Hisao Tamaki. Học tập không phân phối chịu được tiếng ồn của các khái niệm hình học chung. Tạp chí ACM (JACM), 45(5):863–890, 1998.

[4] Gordon Fraser và Andrea Arcuri. Evosuite: tạo bộ kiểm thử tự động cho phần mềm hướng đối tượng. Trong Biên bản báo cáo của hội thảo ACM SIGSOFT lần thứ 19 và hội nghị châu Âu lần thứ 13 về Nền tảng của kỹ thuật phần mềm, trang 416–419, 2011.

[5] Long H. Pham, Ly Ly Tran Thi, và Jun Sun. Tạo ra sự khẳng định thông qua học tập tích cực. Trong Zhenhua Duan và Luke Ong, biên tập viên, Phương pháp hình thức và Kỹ thuật phần mềm, trang 174–191, Cham, 2017. Springer International Publishing.

[6] Paul Holser. Kho lưu trữ github JUnit-Quickcheck.

[7] Flemming Nielson, Hanne R Nielson và Chris Hankin. Nguyên tắc phân tích chương trình. Springer Science & Business Media, 2004.

[8] Carlos Pacheco và Michael D. Ernst. Randoop: thử nghiệm ngẫu nhiên theo hướng phản hồi cho Java. Trong OOPSLA 2007 Companion, Montreal, Canada. ACM, tháng 10 năm 2007.

[9] Long H Pham, Jun Sun, và Quang Loc Le. Xác minh thành phần của các chương trình thao tác heap thông qua học tập có hướng dẫn thuộc tính. Trong Hội thảo Châu Á về Ngôn ngữ lập trình và Hệ thống, trang 405–424. Springer, 2019.

[10] Long H. Pham, Ly Ly Tran Thi, và Jun Sun. Tạo ra khẳng định thông qua học tập tích cực. Trong Hội nghị quốc tế lần thứ 39 về Kỹ thuật phần mềm của IEEE/ACM (ICSE-C) năm 2017, trang 155–157, 2017.

[11] Rahul Sharma, Saurabh Gupta, Bharath Hariharan, Alex Aiken và Aditya V. Nori. Kiểm chứng như học các khái niệm hình học. Trong Francesco Logozzo và Manuel F¨ahndrich, biên tập viên, Phân tích tĩnh, trang 388–411, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[12] Rahul Sharma, Aditya V Nori và Alex Aiken. Sự đánh đổi giữa độ lệch và phương sai trong phân tích chương trình. Thông báo ACM SIGPLAN, 49(1):127–137, 2014.

[13] Elena Sherman và Matthew B Dwyer. Khai thác cấu trúc miền và chương trình để tổng hợp các phân tích luồng dữ liệu hiệu quả và chính xác (t). Năm 2015, Hội nghị quốc tế lần thứ 30 về Kỹ thuật phần mềm tự động (ASE) của IEEE/ACM, trang 608–618. IEEE, 2015.

[14] Nhóm JUnit. Junit.

[15] LG Valiant. Một lý thuyết về khả năng học được. Cộng đồng. ACM, 27(11):1134–1142, tháng 11 1984.

[16] Raja Vall´ee-Rai, Phong Co, Etienne Gagnon, Laurie Hendren, Patrick Lam và Vijay Sundaresan. Soot - một khuôn khổ tối ưu hóa bytecode java. Trong Biên bản báo cáo Hội nghị năm 1999 của Trung tâm Nghiên cứu Nâng cao về Nghiên cứu Hợp tác, CASCON '99, trang 13. IBM Press, 1999.