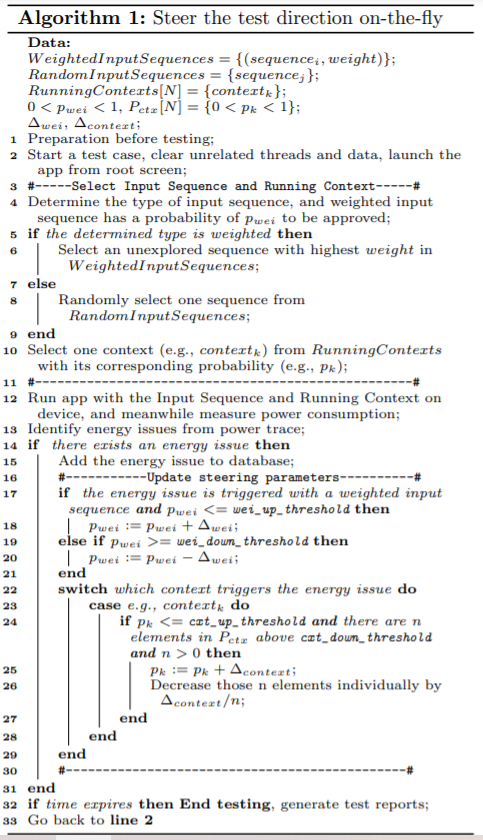
**5.2. Chỉ đạo hướng thử nghiệm on-the-fly**

Khung của chúng tôi điều khiển hướng thử nghiệm một cách linh hoạt dựa trên lịch sử kiểm tra. Thuật toán 1 cho thấy chi tiết về sơ đồ lái của chúng tôi. Lý do đằng sau là như này: khi một vấn đề năng lượng được phát hiện, Nó ngụ ý rằng loại trình tự đầu vào và bối cảnh đang chạy này có thể có cơ hội lớn hơn để gây ra các vấn đề về năng lượng so với trường hợp bình thường vì nó đã mang đến một vấn đề năng lượng cho ánh sáng. Do đó, khung của chúng tôi sẽ tạo ra nhiều hơn một chút các trường hợp thử nghiệm này để có cơ hội phát hiện các vấn đề năng lượng lớn hơn.



**Dữ liệu cho thuật toán**

Các chuỗi đầu vào ứng cử viên và các bối cảnh đang chạy được thiết kế dựa trên cách tiếp cận chúng tôi đã trình diễn trong Phần 5.1.1 và 5.1.2. Chúng tôi trình bày chúng trong các cấu trúc dữ liệu của trọng lượng có trọng lượng, đầu vào ngẫu nhiên và chạy bộ. N là số lượng bối cảnh chạy ứng cử viên. Trong thí nghiệm của chúng tôi, chúng tôi đã thiết lập 4 bối cảnh đang chạy, vì vậy n == 4.

Pwei và Pctx là các thông số lái để chỉ đạo bài kiểm tra. Pwei là xác suất chọn một chuỗi đầu vào có trọng số cho trường hợp thử nghiệm sắp tới. Trong thử nghiệm của chúng tôi, chúng tôi khởi tạo nó là 50%, do đó, trường hợp thử nghiệm có khả năng 50% để chạy với trình tự đầu vào có trọng số khi bắt đầu thử nghiệm và chúng tôi sẽ cập nhật và tinh chỉnh Pwei On-the-Fly trong toàn bộ thử nghiệm. Mặt khác, một phần tử PK trong PCTX đại diện cho xác suất chọn bối cảnh trong các bối cảnh đang chạy. Chúng tôi cũng sẽ cập nhật PCTX vào thời gian chạy. Lưu ý rằng, tổng kết các phần tử trong PCTX bị ràng buộc với 1.

∆wei là sự gia tăng được sử dụng để tăng hoặc giảm pwei để tinh chỉnh Pwei. ∆context đóng vai trò tương tự cho Pctx. Lớn hơn ∆wei và ∆context Chúng tôi sử dụng, tích cực hơn chúng ta sẽ điều chỉnh hướng thử nghiệm.

**Chi tiết về thuật toán**

Trước tiên, chúng tôi chuẩn bị dữ liệu và khởi tạo các tham số (i.e., Pwei, Pctx, Δ wei và Δ Context). Sau đó chúng tôi bắt đầu một trường hợp thử nghiệm, xóa các luồng và dữ liệu không liên quan, khởi chạy.

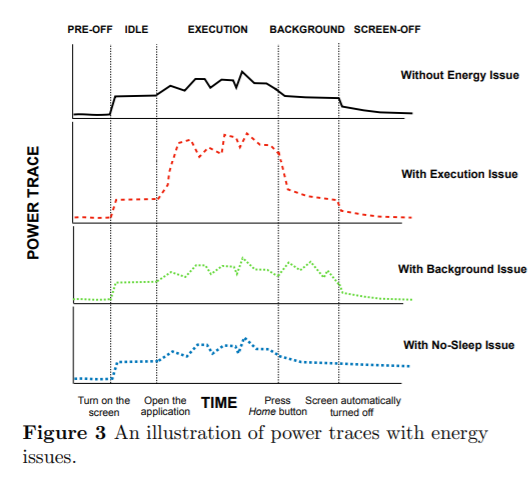
Ứng dụng từ màn hình gốc. Tiếp theo, chúng tôi sẽ quyết định loại trình tự đầu vào; Chuỗi đầu vào có trọng số có xác suất pwei để được chọn. Nếu loại đã chọn là " weighted ", sau đó chúng tôi chọn một chuỗi chưa được khám phá với *weight* cao nhất trong *WeightedInputSequences.* Nếu không, chúng tôi chọn ngẫu nhiên một từ *RandomInputSequences.*Tương tự như vậy để chạy bối cảnh, chúng tôi chọn một từ Run-ningContexts với xác suất tương ứng.

Chúng tôi cung cấp chủ đề ứng dụng với chuỗi đầu vào đã chọn và cài đặt bối cảnh trên thiết bị, và biện pháp tiêu thụ điện năng cùng một lúc. Power Trace sẽ được phân tích để xác nhận xem một vấn đề năng lượng xảy ra hay không. Nếu tồn tại một vấn đề năng lượng, nó ngụ ý rằng loại trình tự đầu vào và bối cảnh chạy tương ứng có thể mang lại lợi nhuận để kích động nhiều vấn đề năng lượng hơn, khung thử nghiệm của chúng tôi sau đó chỉ định một chút về hướng này. Cụ thể, nếu nó được kích hoạt với trình tự đầu vào có trọng số, chúng tôi tăng Pwei bởi Δwei và Pwei không được vượt *wei\_up\_threshold.* Nếu không, chúng tôi đã giảm pwei bởi Δwei. Ngoài ra, chúng tôi giữ Pwei > = *wei\_down \_threshold.*

Một cách tiếp cận tương tự được áp dụng để tinh chế Pctx. Chúng tôi kiểm tra theo đó đang chạy bối cảnh (e.g., contextk) vấn đề xảy ra, sau đó tăng xác suất thử nghiệm của nó (e.g., pk). Tuy nhiên, điều kiện tiên quyết là phải có ít nhất một yếu tố (ngoại trừ chính Pk) trong Pctx ở trên *cxt\_down \_threshold* bởi vì chúng tôi, on one hand, có ý định cân bằng lại xác suất, Mặt khác, chúng ta cũng nên để tất cả các bối cảnh có ít nhất một khả năng của *cxt\_down\_threshold* để được kiểm tra.

**5.3 Xác định các vấn đề năng lượng từ dấu vết điện**.

Chúng tôi chia tỷ lệ năng lượng thành năm giai đoạn, cụ thể là PRE-OFF, IDLE, EXECUTION, BACKGROUND và SCREEN-OFF. Bộ phận này có thể giúp chúng tôi xác định ba loại vấn đề năng lượng: các vấn đề thực hiện (bao gồm các vấn đề gây ra bởi khối lượng công việc không cần thiết và các hoạt động quá mức), các vấn đề về nền tảng và vấn đề không ngủ. Hình 3 cho thấy một minh họa về dấu vết năng lượng với ba loại vấn đề năng lượng này. giai đoạn PRE-OFF là giai đoạn bắt đầu nơi thiết bị được cung cấp nhưng màn hình tắt. Sau đó, trường hợp thử nghiệm sẽ được chuyển sang giai đoạn nhàn rỗi bằng cách bật màn hình.

**

Để vào giai đoạn EXECUTION, ứng dụng chủ đề sẽ được mở và chạy với một chuỗi đầu vào nhất định và bối cảnh đang chạy, được tạo như trong Phần 5.1 và 5.2. sau giai đoạn EXECUTION, Ứng dụng sẽ được cho ăn bằng một nút bấm Home để vào giai đoạn BACKGROUND. Giai đoạn cuối cùng là SCREENOFF, mà bắt đầu khi màn hình được phép tắt tự động, tuy nhiên, một phần của các vấn đề năng lượng sẽ giữ màn hình ngay cả tại giai đoạn SCREEN-OFF, ăn một lượng lớn năng lượng pin.

***Identifying execution issues****. (Xác định các vấn đề thực hiện)*

Đối với các vấn đề thực thi, như chúng ta đã thảo luận trong Phần 1, đánh giá sự cần thiết của khối lượng công việc ứng dụng là rất quan trọng để xác định chúng. Cụ thể, chúng tôi sử dụng thuật toán phân cụm DBSCAN [7] (Mật độ dựa trên các ứng dụng không gian với tiếng ồn) để thực hiện mục đích này. Mục tiêu của DBSCAN là phân loại các điểm dữ liệu đa chiều thành ba nhóm, cụ thể là các điểm cốt lõi, điểm biên và các điểm ngoại lệ. Sau khi phân cụm, các điểm dữ liệu sẽ có các thuộc tính sau:

1. Đối với một điểm cốt lõi, số lượng hàng xóm của nó không ít hơn một giá trị nhất định, MinPts. Nói chung, các điểm cốt lõi là “quite close and gathered”.
2. Đối với một điểm biên giới, hàng xóm của nó ít hơn MinPts, nhưng nó là một người hàng xóm có ít nhất một điểm cốt lõi hoặc một điểm biên giới khác.
3. Đối với một điểm ngoại lệ, hàng xóm của nó ít hơn MinPts và nó không có cốt lõi hoặc hàng xóm biên giới.

Chúng tôi đối xử với từng trường hợp thử nghiệm làm điểm dữ liệu và xử lý các trường hợp thử nghiệm trong cùng một danh mục ứng dụng dưới dạng tập dữ liệu để phân cụm. Kích thước của từng điểm dữ liệu mà chúng tôi sử dụng để phân cụm là lchpp, nchpp, µchpp, µexe , tất cả được trích xuất từ ​​dấu vết điện của giai đoạn EXECU-TION của mỗi trường hợp thử nghiệm. lchpp là tổng chiều dài của chu kỳ continuous-high-power (công suất cao liên tục). chu kỳ continuous-high-power là khi điện liên tục vượt quá một ngưỡng nhất định dài hơn một chiều dài nhất định. Nchpp là số lượng của những giai đoạn này. µchpp là sức mạnh trung bình của những giai đoạn này. µexe là sức mạnh trung bình của toàn bộ giai đoạn EXECUTION.

sau đó Dbscan phân loại các trường hợp thử nghiệm vào ba nhóm đó. Chúng tôi dán nhãn các trường hợp thử nghiệm ở các nhóm lõi và biên giới là “normal”, và gắn nhãn những người trong nhóm ngoại lệ là nghi phạm đối với các vấn đề về năng lượng. Sau đó, đánh giá của chúng tôi trên 89 ứng dụng (liên quan đến các trường hợp thử nghiệm 35600) cho thấy chỉ có 1,1% trường hợp thử nghiệm là ngoại lệ, có thể hiểu biết energy-consuming (tiêu tốn năng lượng). Chúng tôi cho rằng họ có “unnecessary” (khối lượng hoạt động công việc không cần thiết) không cần thiết. Cuối cùng, chúng tôi xác minh thủ công liệu có tồn tại các vấn đề thực thi thực sự không. Trong thí nghiệm, khung thử nghiệm của chúng tôi đã phát hiện 47 vấn đề thực thi ứng cử viên từ các trường hợp thử nghiệm 1,1% đó. Chỉ có ba (trong số 47) là dương tính giả, cho thấy độ tin cậy cao của phương pháp này. Ngược lại, công nghệ hiện tại [2] đã phát hiện 3 vấn đề thực thi ứng viên từ 30 ứng dụng, và vẫn là một trong số đó là một sự tích cực sai lầm.

**Identifying background and no-sleep issues. (Xác định nền tảng và vấn đề no-sleep.)**

Nếu ứng dụng miễn phí từ các vấn đề nền, dấu vết điện trong giai đoạn BACKGROUND được cho là tương tự như trong giai đoạn IDLE. Do đó, chúng tôi tính toán giá trị không giống nhau của hai dấu vết. Nếu giá trị nằm trên một ngưỡng nhất định (40% trong thí nghiệm của chúng tôi), chúng tôi đã dán nhãn cho trường hợp thử nghiệm này làm ứng cử viên cho một vấn đề nền tảng.

Chúng tôi xác định các vấn đề no-sleep theo cùng một cách. Chúng tôi so sánh PRE-OFF với SCREEN-OFF (so sánh trước khi tắt màn hình với tắt màn hình). Nếu sự khác biệt vượt quá ngưỡng nhất định (50% trong thí nghiệm của chúng tôi), chúng tôi suy đoán trường hợp thử nghiệm này đang bị một vấn đề no-sleep ứng cử viên.

**5.4 Manual verification (Xác minh thủ công)**

Sau khi các vấn đề ứng cử viên được tìm thấy, chúng tôi xác minh thủ công cho dù chúng là vấn đề về năng lượng thực tế. Các bước để xác minh thủ công như đã làm theo:

1. Chúng tôi chạy lại trường hợp thử nghiệm để kiểm tra nếu năng lượng vấn đề có thể được sao chép.
2. Chúng tôi phân tích dấu vết điện để khẳng định tác động đáng kể đến mức tiêu thụ năng lượng.
3. NS. Chúng tôi quan sát các triệu chứng của vấn đề, bao gồm UI nhấp nháy, thông báo bật lên, thời gian dài CPU thức dậy, sử dụng CPU cao và tốc độ kết xuất GPU cao.
4. Chúng tôi kiểm tra dấu vết thực thi (thu được từ Android TraceView11, ngay bây giờ TraceView không dùng nữa, các nhà phát triển có thể sử dụng CPU Profiler12) để kiểm tra xem có các phương thức nào được kích thích thường xuyên trong Pro-gram hay không. Trong thực tế, họ thường là các nghi phạm của mã bị lỗi. Theo các báo cáo vấn đề trong nghiên cứu thực nghiệm của chúng tôi, vì các vấn đề được chẩn đoán sử dụng thông tin này, mã bị lỗi đã được tìm thấy 100% và cố định. Dưới đây là hai ví dụ :

* <https://bugs.chromium.org/p/chromium/issues/detail?id=349059>
* <https://bugs.chromium.org/p/chromium/issues/detail?id=480522>

Chỉ khi vấn đề có thể được sao chép, gây ra chất thải năng lượng đáng chú ý, có các triệu chứng có thể nhìn thấy được, và mã bị lỗi của nó có thể được đặt, chúng tôi khẳng định sự tồn tại thực sự của nó. Nếu không, chúng tôi xác định nó như một tích cực sai.

**6. EXPERIMENTAL EVALUATION** (Đánh giá thử nghiệm)

Trong phần này, trước tiên chúng tôi trình bày các chi tiết cụ thể của thiết lập thử nghiệm của chúng tôi. Sau đó, chúng tôi đánh giá khung thử nghiệm của chúng tôi về các khía cạnh khác nhau, chẳng hạn như hiệu quả của việc phát hiện các vấn đề về năng lượng, so sánh với nghệ thuật tiên tiến, v.v. Kết quả cho thấy rằng khung thử nghiệm của chúng tôi chủ yếu vượt trội so với công nghệ hiện tại, được hưởng lợi từ âm thanh Nghiên cứu thực nghiệm và thực hiện Targetting của khung thử nghiệm.

**6.1** **Experimental setup** (Thiết lập thử nghiệm)

Chúng tôi sử dụng bảng phát triển Odroid-Xu4 (https://wiki.odroid.com/odroid-xu4/odroid-xu4), có bộ xử lý có bốn lõi lớn với tần số 2 GHz và bốn lõi nhỏ có tần số 1,3 GHz. Bộ nhớ chính lớn tới 2 GB. Hội đồng quản trị sở hữu một máy gia tốc 3D mạnh mẽ, GPU Mali-T628 MP6. Công suất cao của Hội đồng Odroid-XU4 đảm bảo hiệu suất của mình cho hầu hết các ứng dụng trên thị trường.

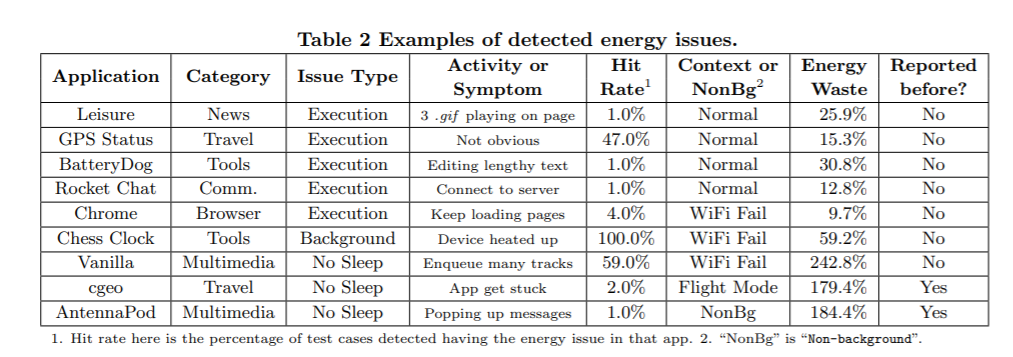
Nó cũng cung cấp một gói tùy chọn nhà phát triển. Ví dụ: nó có thể hiển thị sử dụng CPU và cấu hình kết xuất GPU trên màn hình sẽ hỗ trợ chúng tôi trong việc xác minh sự tồn tại thực sự của các vấn đề năng lượng. Nó cũng được trang bị màn hình nguồn, SmartPower216, để đo mức tiêu thụ điện năng thời gian thực. Tỷ lệ lấy mẫu là 100 Hz. Do các tính năng phong phú này, Odroid Board được sử dụng rộng rãi trong lĩnh vực tối ưu hóa năng lượng cho các thiết bị di động.

Chúng tôi sử dụng Android làm hệ điều hành mục tiêu của chúng tôi vì nó có nguồn gốc mở và thị phần của nó chiếm khoảng 85,9% 17 của khối lượng điện thoại thông minh trên toàn thế giới vào cuối quý 1 năm 2018. Chúng tôi đánh giá khung của chúng tôi trên cùng một tập hợp 89 đối tượng ứng dụng trong nghiên cứu thực nghiệm. Tổng thời gian thử nghiệm của chúng tôi cho 89 đối tượng ứng dụng là 2373,3 giờ, tức là 98,9 ngày (trung bình, 1,11 ngày đối với một chủ đề ứng dụng).

**6.2 The efficacy of our testing framework** (Hiệu quả của khung thử nghiệm của chúng tôi).

Kết quả thử nghiệm cho thấy bài kiểm tra của chúng tôi phát hiện 91 vấn đề năng lượng ứng cử viên, Trong đó chúng tôi xác nhận thủ công 83 vấn đề về năng lượng thực sự. Chúng tôi có 8 dương tính giả. Các dương tính giả là do tiếng ồn trong các mẫu điện trong một số trường hợp. Tiếng ồn có thể là kết quả của các hoạt động hệ điều hành tại nền, chẳng hạn như bộ sưu tập rác của Android Runtime18, tải xuống và cập nhật hệ thống ngẫu nhiên.

Bảng 2 cho thấy 9 ví dụ về các vấn đề năng lượng được phát hiện. Ví dụ, trong thời gian giải trí, ba hoạt hình .gif được tải và phát ở cuối một trang nhất định mặc dù chúng vô hình đối với người dùng hầu hết thời gian. Vấn đề thực hiện này lãng phí 25,9% sử dụng năng lượng. Nó có thể được khắc phục bằng cách đóng băng hoạt hình khi hình ảnh .gif không được hiển thị trên màn hình. Đối với một ví dụ khác, khi đồng hồ cờ là sau khi sử dụng và lướt qua, thiết bị sẽ được làm nóng từ 41,2 ◦c đến 60,9 ◦c do sử dụng CPU không hiệu quả và dài hạn. Sức mạnh trung bình của vấn đề này cao hơn 59,2% so với giai đoạn nhàn rỗi.



91,6% (76 trên 83) Các vấn đề về năng lượng được phát hiện trong bài kiểm tra của chúng tôi là mới được báo cáo. Thí nghiệm của chúng tôi cũng cho thấy những vấn đề này trung bình gấp đôi mức tiêu thụ năng lượng của các ứng dụng (xem sau trong Phần 6.5). Nếu không có bài kiểm tra của chúng tôi, những vấn đề năng lượng nghiêm trọng này sẽ không bao giờ nổi lên ngay cả khi pin thoát nước tuyệt vọng. Mặt khác, 94,5% (120 trên 127) các vấn đề về năng lượng trong nghiên cứu thực nghiệm của chúng tôi đã không được liệt kê trong các vấn đề được phát hiện bởi bài kiểm tra của chúng tôi. Xem xét rằng xét nghiệm rất lớn (2373,3 giờ), không khả thi để trải qua tất cả các trường hợp thử nghiệm để tìm ra lý do tại sao chúng không được phát hiện và trình bày tỷ lệ của chúng. Tuy nhiên, chúng tôi phỏng đoán các lý do chính có thể là như sau:

Đầu tiên, tiêu chuẩn xác định một vấn đề năng lượng cao hơn nhiều so với các nhà phát triển. Các vấn đề được phát hiện bởi thử nghiệm của chúng tôi thường có tác động lâu dài đến mức tiêu thụ năng lượng, chất thải năng lượng thường trên 10,0%. Tuy nhiên, nhiều vấn đề được phát hiện bởi các nhà phát triển có thể chỉ gây ra khối lượng công việc thoáng qua và chất thải năng lượng của chúng có thể khó hơn 10,0%. Thứ hai, một số (26,7%, 32 trên 120) các vấn đề không thể tái tạo, vì vậy bài kiểm tra của chúng tôi cũng không thể kích hoạt chúng. Thứ ba, do giới hạn thời gian, sự đa dạng của các chuỗi đầu vào và các bối cảnh đang chạy trong thử nghiệm của chúng tôi không đủ lớn để bao gồm tất cả chúng.

Sau tất cả, khung của chúng tôi vẫn cho thấy hiệu quả của nó trong việc phát hiện các vấn đề về năng lượng nghiêm trọng và không được báo cáo, vượt quá tầm nhìn của các nhà nghiên cứu trước đây.