ÁP DỤNG HỆ THỐNG VIẾT LẠI SỐ HẠNG ĐỂ TỔ HỢP CÁC DỊCH VỤ WEB TỰ ĐỘNG

Huỳnh Tấn Khải[1]

Tóm tắt: *Với các ưu điểm của dịch vụ web so với ứng dụng truyền thống trong việc giải quyết các vấn đề như độc lập nền tảng, khả năng tái sử dụng, dễ triển khai,... số lượng dịch vụ web ngày càng tăng. Thay vì phải phát triển một phần mềm mới cho yêu cầu công việc, người dùng có thể sử dụng các dịch vụ web phù hợp sẵn có. Người dùng có thể tìm kiếm và tổ hợp các dịch vụ web sẵn có theo yêu cầu của mình. Do đó, tổ hợp dịch vụ web đáp ứng yêu cầu người dùng là một vấn đề cần phải giải quyết. Tổ hợp tự động các dịch vụ web là lĩnh vực đang thu hút sự quan tâm của các nhà nghiên cứu. Tuy nhiên, các phương pháp hiện tại còn nhiều hạn chế như tốn thời gian, không xét đến các giá trị chất lượng dịch vụ,... Nghiên cứu trong bài báo này cung cấp một hướng đi mới, đó là sử dụng phương pháp viết lại số hạng để tự động hóa việc tổ hợp các dịch vụ web.*

Từ khoá: *Viết lại số hạng, Dịch vụ Web, Tổ hợp Dịch vụ Web, Tái sử dụng Dịch vụ Web, Term Rewriting*

1. Mở đầu

Ngày nay, khi các nhu cầu trong doanh nghiệp ngày càng tăng trưởng cả về số lượng lẫn chất lượng, việc một doanh nghiệp tự phát triển các phần mềm riêng biệt cho công ty trở nên tốn kém, khó mở rộng, phát triển và chia sẻ với các doanh nghiệp bên ngoài. Dịch vụ web được tạo ra thỏa mãn các yêu cầu nêu trên vì có các tính chất sau [1]:

- Khả năng kết hợp lẫn nhau: các dịch vụ web không nằm trong mỗi doanh nghiệp mà được chia sẻ ra bên ngoài. Do đó, chúng có khả năng kết hợp lẫn nhau để thỏa mãn các nhu cầu ngày càng phức tạp của người dùng. Hơn thế nữa, dịch vụ web được tổ hợp không phụ thuộc vào ngôn ngữ lập trình và hệ điều hành. Từ đó, việc phát triển một dịch vụ web phức tạp dựa trên các dịch vụ đơn giản, sẵn có được thực hiện một cách dễ dàng hơn.

- Khả năng sử dụng: từ kho các dịch vụ mà người dùng có thể tìm được trên mạng, họ có thể tự do lựa chọn dịch vụ mà họ cần, sử dụng ngôn ngữ lập trình hoặc các công cụ để thay đổi dịch vụ đó theo hướng mà họ muốn.

- Khả năng triển khai: dịch vụ web được triển khai dựa trên các tiêu chuẩn công nghệ mạng. Do đó, việc sử dụng các dịch vụ web được thực hiện dễ dàng từ bất kỳ nơi đâu có kết nối mạng.

Các dịch vụ web trên mạng rất cơ bản, chỉ phục vụ cho các yêu cầu đơn giản. Tuy nhiên, yêu cầu của người dùng lại phức tạp, thường phải kết hợp nhiều dịch vụ có sẵn lại với nhau để thực hiện được công việc mong muốn. Ví dụ với các dịch vụ có sẵn như: dịch vụ chọn và đặt vé máy bay, dịch vụ vận chuyển, chọn khách sạn,… Một người dùng mỗi khi đi du lịch lại cần một dịch vụ phức tạp được tổ hợp từ các dịch vụ trên, giải quyết các vấn đề về thủ tục di chuyển, về việc chọn nơi ở,… Đồng thời, dịch vụ tổ hợp thỏa mãn

1. TS., Khoa Toán – Tin, Trường Đại học Quảng Nam

tiêu chí được đặt ra về thời gian, chi phí,… Các tiêu chí này được gọi là chất lượng dịch vụ (Quality of Service - QoS) [2]. Như vậy, người dùng cần một giải pháp giúp tổng hợp được một dịch vụ web phức hợp từ các dịch vụ đã có nhằm thỏa mãn yêu cầu của mình.

Hiện nay, rất nhiều giải pháp đã được đưa ra để giải quyết vấn đề trên. Tuy nhiên, các giải pháp này còn khá nhiều nhược điểm như:

- Các giải pháp này tính toán để có thể đưa ra tất cả các cách kết hợp có thể có để từ đó đưa ra được kết quả tối ưu nhất thỏa yêu cầu người dùng. Vì số lượng dịch vụ web hiện nay quá nhiều, phương pháp này rất tốn thời gian và không thích hợp khi số lượng dịch vụ web được cập nhật (vì chương trình phải tính toán lại để đưa ra danh sách mọi sự tổ hợp có thể).

- Các tính toán trên dựa vào khả năng kết hợp của các dịch vụ mà không tính đến các tiêu chí về đặc điểm chất lượng dịch vụ. Do đó, giải pháp đưa ra có thể không thỏa mãn nhu cầu của người dùng.

- Với nhiều loại chất lượng dịch vụ khác nhau, công thức tính toán chất lượng dịch vụ lại khác nhau. Ví dụ: nếu tính theo chi phí dịch vụ thì chi phí của dịch vụ web tổ hợp bằng tổng chi phí các dịch vụ thành phần, nếu tính theo độ sẵn có (availability) thì độ sẵn có của dịch vụ web tổ hợp bằng tích độ sẵn có của các dịch vụ thành phần. Các công thức này hiện nay được mô tả riêng khỏi các dịch vụ web và được gán cứng trong mã nguồn của chương trình tổ hợp. Mỗi khi muốn thay đổi công thức này, người dùng cần phải sửa chúng trong mã nguồn và chạy lại chương trình tổ hợp.

Giải pháp được đề nghị trong nghiên cứu này dựa trên các luật mô tả dịch vụ web. Các luật này có thể mô tả các đặc điểm của dịch vụ web như đầu vào, đầu ra, các thông số về chất lượng dịch vụ, điều kiện để thực hiện. Công thức tính toán chất lượng dịch vụ web tổ hợp được mô tả ngay trong đặc tả của các luật tương ứng với các dịch vụ web đơn cấu tạo nên chúng. Chương trình sẽ dựa trên tập luật này để chọn luật phù hợp với yêu cầu của người dùng. Với phương pháp như trên, chương trình có khả năng tổ hợp dịch vụ web thỏa mãn yêu cầu cho trước và cho phép thay đổi tập luật trong quá trình tính toán.

Trong bài báo này, chúng tôi đưa ra giải pháp áp dụng hệ thống viết lại số hạng (term rewriting) để tổ hợp các dịch vụ web dựa trên các tiêu chí ràng buộc cứng (tiêu chí về đầu vào và đầu ra), ràng buộc mềm (tiêu chí về chất lượng dịch vụ). Để minh họa kết quả của quá trình nghiên cứu, tác giả đã xây dựng nên một chương trình có khả năng tổ hợp các dịch vụ web sẵn có nhằm thỏa mãn yêu cầu của người dùng. Sau đây là những đóng góp chính của bài báo:

- Đưa ra một phương pháp mới để có thể tổ hợp các dịch vụ web sử dụng hệ thống viết lại số hạng. Nhờ vào phương pháp này, các thay đổi trên tập dịch vụ web (thêm, bớt các dịch vụ) có thể được thực thi ngay trong quá trình tính toán.

- Cho phép xác định được khả năng thực thi khi kiểm tra quá trình tổ hợp bằng cách xác định điều kiện của các dịch vụ thông qua giá trị chất lượng dịch vụ. Ví dụ: dịch vụ vận chuyển sẽ không được áp dụng nếu chi phí vượt quá một mức yêu cầu người dùng. Đối với

các phương pháp khác, điều kiện này chỉ có thể xem xét sau khi đã biết được cách tổ hợp các dịch vụ web.

- Cải tiến việc lựa chọn các dịch vụ web: thay vì tạo ra tất cả các mô hình kết hợp có thể có của tất cả các dịch vụ web, chương trình tổ hợp có thể dựa trên các yêu cầu của người dùng để loại bỏ các dịch vụ không thỏa mãn ở từng bước lựa chọn dịch vụ, sử dụng các giải thuật heuristic để tăng tốc quá trình tìm kiếm.

Phần còn lại của bài báo được tổ chức như sau: Phần 2 khảo sát về các công trình nghiên cứu có liên quan. Trong phần 3, bài báo sẽ đi sâu vào việc phân tích và diễn giải các kiến thức nền tảng về hệ thống viết lại số hạng để phục vụ cho việc tổ hợp các dịch vụ web. Phần 4 trình bày chi tiết về ý tưởng xây dựng chương trình tổ hợp các dịch vụ web và đưa ra một ví dụ thực tế để minh họa cho chương trình. Phần 5 trình bày về kết quả thực nghiệm và đưa ra một số đánh giá về hệ thống. Phần 6 đưa ra kết luận và đề xuất một số hướng phát triển tiếp theo trong tương lai.

2. Nội dung

2.1. Các công trình nghiên cứu có liên quan

Các nghiên cứu vấn đề tổ hợp dịch vụ web có thể được phân thành 2 nhóm như sau:

2.1.1. Tổ hợp dịch vụ web dựa trên các ràng buộc cứng

Tổ hợp dịch vụ web chỉ liên quan đến các thuộc tính chức năng (các ràng buộc cứng) là bài toán kinh điển của kiến trúc hướng dịch vụ (SOA), phần lớn các nghiên cứu thuộc nhóm này dựa trên lý thuyết về Lập kế hoạch của lĩnh vực trí tuệ nhân tạo (AI planning), chẳng hạn như [3]. Một số nghiên cứu gần đây đề xuất sử dụng các mô hình trừu tượng như Petri net hoặc Colored Petri net [4] để tổ hợp và xác minh các dịch vụ web. PORSCE II [3] là một khung thức thực hiện tổ hợp các dịch vụ web dựa trên các yêu cầu về đầu vào và đầu ra của các dịch vụ (ràng buộc về chức năng hay ràng buộc cứng). Tương tự, OWLS-XPlan [5] cũng sử dụng các dịch vụ web được thể hiện bởi ngôn ngữ OWL-S để chuyển bài toán từ lĩnh vực tổ hợp dịch vụ web sang lĩnh vực lập kế hoạch và sử dụng bộ lập kế hoạch có tên XPlan để xử lý.

2.1.2. Thành phần dựa trên ràng buộc cứng và mềm

Phương pháp tổ hợp dịch vụ web kết hợp các thuộc tính chức năng (ràng buộc cứng) và thuộc tính chất lượng dịch vụ - QoS (ràng buộc mềm) đã được đề xuất trong [2]. Trong [2], các tác giả áp dụng thuật toán di truyền (Genetic Algorithm - GA) để giải quyết vấn đề tổ hợp với mỗi cách tổ hợp được mã hóa thành một gen. Tuy nhiên, nghiên cứu này chỉ cung cấp cho chúng tôi một cơ chế để chọn cách tổ hợp tốt nhất (có thể) từ một tập hợp các tổ hợp (lược đồ tổ hợp đầy đủ) chứ không phải thực hiện từng bước tổ hợp từ các dịch vụ web thành phần. Bên cạnh đó, việc áp dụng thuật toán di truyền làm tăng độ phức tạp của bài toán và do đó rất khó áp dụng vào thực tế.

Một cách tiếp cận khác trong [6] đề xuất khôi phục tự động tổ hợp khi việc thực thi các tổ hợp bị rơi vào trạng thái lỗi (không thể truy cập được đến dịch vụ web hoặc các dịch vụ web không đáp ứng được yêu cầu của người dùng). Trong cách tiếp cận này, chúng ta

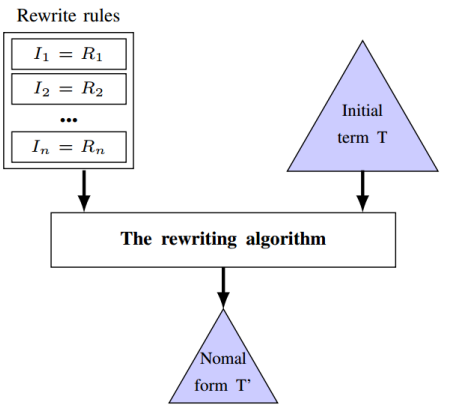
phải có một lược đồ tổ hợp đầy đủ được mô tả bằng ngôn ngữ BPEL, được chuyển đổi thành Hệ thống chuyển tiếp được gắn nhãn (Lebelled Transition System - LTS), được giám sát bởi một một giám sát chuyển trạng thái tự động (monitor automata). Khi phát sinh lỗi (trạng thái không thể đạt đến được, tương ứng với dịch vụ web không thể truy cập được), hệ thống sẽ bắt đầu tính toán phương án khôi phục bằng giải thuật di truyền. Sự khác biệt giữa [6] và [2] là kích thước của gen trong [6] là không cố định, điều này phụ thuộc vào số bước đã thực hiện qua cũng như số bước phải quay lui khi việc thực thi tổ hợp bị rơi vào trạng thái lỗi.

2.2. Hệ thống viết lại số hạng (Term Rewriting System – TRS)

Trong phần này, chúng tôi sẽ đi sâu vào việc phân tích và diễn giải các kiến thức nền tảng của việc xây dựng hệ thống viết lại số hạng (Term Rewriting System – TRS) [7] để có thể áp dụng vào việc tổ hợp dịch vụ web. Để tăng tốc quá trình tổ hợp nhằm thỏa mãn các yêu cầu về chất lượng dịch vụ, chúng tôi sử dụng phương pháp tìm kiếm tốt nhất trước (best-first search) dựa trên giá trị của các thuộc tính chất lượng dịch vụ ở từng bước của quá trình tổ hợp.

Để có thể hiểu rõ về hệ thống viết lại số hạng, chúng ta bắt đầu với biểu thức đơn giản $(9 - 5)^2 \times (7 + 4)$, biểu thức này sẽ được đơn giản hóa theo một số bước, sử dụng các quy tắc cơ bản số học như $(9 - 5)^2 \times (7 + 4) \rightarrow 4^2 \times 11 \rightarrow 16 \times 11 \rightarrow 176$.

Có nhiều hình thức đơn giản hóa (rút gọn) biểu thức trên. Quy tắc đơn giản hóa thường bắt nguồn từ các quy tắc toán học, chẳng hạn như $a^2 + 2ab + b^2 = (a + b)^2$ hoặc các quy tắc nghiệp vụ của mỗi vấn đề mà chúng ta cần giải quyết. Một cách tổng quát, một quy tắc rút gọn sẽ có hai vế, vế bên trái là một công thức (biểu thức) phức tạp, biểu thức này có thể được đơn giản hóa thành biểu thức ở vế bên phải. Các biểu thức này được gọi là các ***số hạng - term*** và việc áp dụng các quy tắc rút gọn này được gọi là viết lại ***số hạng – term rewriting*** [8].



Hình 1. Hệ thống viết lại số hạng (Term Rewriting System – TRS)

Một cái nhìn đơn giản về một hệ thống viết lại số hạng (Term Rewriting System – TRS) được thể hiện trong Hình 1. Cho một tập hợp các quy tắc viết lại và một biểu thức ban đầu T, thuật toán viết lại số hạng được áp dụng và sẽ trả về một biểu thức T', là biểu thức đơn giản hóa của T. T' được gọi là dạng đơn giản hóa hay dạng chuẩn của T.

2.3. Tổ hợp dịch vụ web tự động dựa trên hệ thống viết lại số hạng - TRS

2.3.1. Vấn đề tổ hợp dịch vụ web như một bài toán TRS

Trong phần 2.2., chúng tôi đã giới thiệu về cách sử dụng TRS để rút gọn các bài toán. Vấn đề tổ hợp dịch vụ web cũng là một vấn đề rút gọn khi chúng ta cố gắng thay thế biểu thức mô tả yêu cầu của người dùng bằng biểu thức mô tả các dịch vụ web thành phần có

sẵn. Để áp dụng TRS để giải quyết vấn đề tổ hợp dịch vụ web, chúng ta cần chuyển đổi các dịch vụ web thành các *luật* trong lĩnh vực TRS.

Một dịch vụ web bao gồm hai thành phần chính: đầu vào (dữ liệu cần thiết để thực hiện các dịch vụ) và đầu ra (dữ liệu thu được sau khi thực hiện một dịch vụ). Tính năng này hoàn toàn giống với khái niệm *luật* trong TRS. Khi một biểu thức khớp với phía bên trái của một luật, chúng ta có thể viết lại biểu thức đó bằng vế bên phải của luật này. Do đó, mỗi dịch vụ web thành phần có sẵn có thể được mô tả bằng một luật trong TRS.

Tổ hợp dịch vụ web là một quá trình để tìm ra thứ tự áp dụng các dịch vụ web thành phần để đáp ứng các yêu cầu của người dùng. Việc tính toán trên TRS cũng dựa trên các luật có sẵn để có được dạng rút gọn hoặc nhận được biểu thức thỏa mãn yêu cầu của người dùng được mô tả. Do đó, bài toán tổ hợp dịch vụ web chính là một bài toán trong TRS, trong đó các luật là các biểu thức mô tả các dịch vụ web thành phần và mục tiêu là biểu thức mô tả các yêu cầu của người dùng. Việc áp dụng các quy tắc thu gọn sẽ dừng lại khi chúng ta đạt đến biểu thức thỏa mãn yêu cầu của người dùng hoặc đạt đến dạng tối giản.

Một vấn đề khác là khi các thuộc tính QoS (chẳng hạn như chi phí thực thi, thời gian phản hồi, tính khả dụng, v.v.) được tính đến, giá trị của thuộc tính QoS sẽ được thay đổi dựa trên công thức cho từng thuộc tính cụ thể và giá trị của từng dịch vụ web tham gia tổ hợp. Sử dụng TRS, chúng ta có thể "đính kèm" thông tin này trong đặc tả của từng dịch vụ web giúp chúng ta tính toán giá trị của các thuộc tính QoS của từng dịch vụ thành phần.

Khi giá trị đầu vào và đầu ra (thuộc tính chức năng) của một dịch vụ web không thay đổi, chúng có thể được biểu diễn bằng hằng số. Để dễ xử lý, chúng tôi chỉ định các hằng số bắt đầu bằng chữ hoa; các biến và hàm bắt đầu bằng chữ thường; các công thức toán học được đặt trong dấu ngoặc vuông ("[" và "]"). Các điều kiện của dịch vụ (ví dụ: dịch vụ sẽ chỉ được áp dụng nếu tổng chi phí thực hiện không vượt quá ngưỡng nhất định), được đặt ở cuối đặc tả dịch vụ web và được phân tách bằng ký hiệu "|". Khi điều kiện gồm nhiều điều kiện con, chúng tôi sử dụng hàm "and".

Ví dụ 1: Đặc tả của dịch vụ web thực hiện chức năng giao hàng *ShippingService* nhận vào thông tin đặt hàng của khách hàng (*OrderData*) và địa chỉ giao hàng (*Address*); trả về thông tin ngày giao hàng (ShippingDate); chi phí thực hiện (qosCost) của dịch vụ này là 3 đơn vị tiền tệ trong TRS như sau:

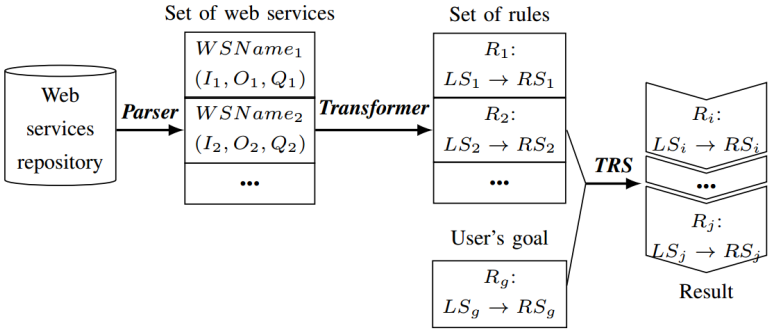
ShippingService: and(OrderData, Address, qosCost(x)) → and(ShippingDate, qosCost([x + 3]))

Trong đó, *x* đại diện cho giá trị hiện tại của *qosCost* của dịch vụ web tổ hợp trước khi tổ hợp thêm dịch vụ này, *qosCost ([x + 3])* cho biết giá trị của *qosCost* sẽ tăng lên 3 đơn vị sau khi áp dụng dịch vụ này. Lưu ý rằng, khi có nhiều thuộc tính QoS, người dùng cần chỉ định tất cả các công thức đó.

Ở mỗi bước tổ hợp, nếu bất kỳ dịch vụ web nào đáp ứng yêu cầu đầu vào của người dùng, nó có thể được sử dụng. Khi đó, đầu ra của dịch vụ web này sẽ được thêm vào danh sách đầu vào ở trạng thái hiện tại, giá trị của thuộc tính QoS cũng sẽ được cập nhật. Quá trình

này diễn ra liên tục cho đến khi đầu ra của tổ hợp thỏa mãn yêu cầu của người dùng. Quá trình sử dụng TRS để giải quyết vấn đề tổ hợp dịch vụ web được minh họa trong Hình 2.

Trong hệ thống, các dịch vụ web được mô tả bằng ngôn ngữ OWL-S và được lưu trữ trong kho dịch vụ Web. Trình phân tích cú pháp (Parser) sẽ đọc các dịch vụ web này và trả về thông tin gồm tên dịch vụ web (WSName), đầu vào (I), đầu ra (O) và các thuộc tính QoS (Q) của mỗi dịch vụ web. Sau đó, trình chuyển đổi (Transformer) sẽ chuyển đổi thông tin này thành các luật trong miền TRS. Mỗi luật có dạng như sau: tên luật (R) theo sau là dấu hai chấm (:), vế trái (LS) suy ra (→) vế phải (RS). Yêu cầu của người dùng cũng được mô tả như một luật (gọi là luật mục tiêu) và TRS sẽ thực hiện quá trình viết lại số hạng để tạo ra biểu thức thỏa mãn kết quả đầu ra của luật mục tiêu.



Hình 2. Sử dụng TRS để giải bài toán về tổ hợp dịch vụ web

Ví dụ 2: Minh họa bộ chuyển đổi sẽ biến đổi các dịch vụ web thành các luật trong TRS như trong Bảng 1 sau:

Bảng 1. Kho dịch vụ web đơn hiện có

Tên dịch vụ	Đầu vào	Đầu ra	Chi phí	Mô tả	Luật trong TRS
BookTo Publisher Service	Book, Author	Publisher	1	Trả về thông tin nhà xuất bản của sách	*BookToPublisherService: and(Book, Author, qosCost(x)) →and(Publisher, qosCost([x+1]))*
Customs Cost Service	Publisher, OrderData	CustomsCost	2	Trả về chi phí của đơn hàng	*CustomsCostService: and(Publisher, OrderData, qosCost(x)) → and (CustomsCost,qosCost([x+2]))*
Shipping Service	Address, OrderData	ShippingDate	3	Trả về ngày giao cho từng đơn hàng	*ShippingService: and(Address, OrderData, qosCost(x)) → and(ShippingDate, qosCost([x+3]))*
CreditCard ChargeService	OrderData, CreditCard	Payment	1	Trả về thông tin tín dụng cho từng đơn hàng	*CreditCardChargeService: and(OrderData, CreditCard, qosCost(x)) → and(P ayment, qosCost([x + 1]))*

WaysOf OrderService	Publisher	Electronic	2	Trả về các đơn hàng hiện có của nhà XB	*WaysOfOrderService: and(Publisher, qosCost(x)) → and(Electronic,qosCost([x+2]))*
Electronic OrderService	Electronic	OrderData	3	Thực thi đơn hàng	*ElectronicOrderService: and(Electronic, qosCost(x)) → and(OrderData,qosCost([x+3]))*

Giả sử chúng ta có yêu cầu là người dùng sẽ cung cấp Tên tác giả (*Author*), Tên sách (*Book*) và Địa chỉ giao hàng (*Address*), người dùng mong muốn các thông tin về giá tiền (*CustomCost*) và ngày giao hàng (*ShippingDate*). Đó là các ràng buộc về mặt chức năng (ràng buộc cứng). Ngoài ra, người dùng còn mong muốn chi phí để thực hiện dịch vụ này không vượt quá 12 đơn vị tiền tệ (đây là ràng buộc về mặt chất lượng dịch vụ). Yêu cầu này của người dùng cũng được biến đổi thành một luật (luật mục tiêu) trong hệ thống TRS như sau:

and(Author, Book, Address, qosCost(0)) → and(CustomCost, ShippingDate, qosCost(12))

Quá trình tổ hợp được thể hiện trong Hình 3. Ở mỗi bước, chỉ một luật được chọn và thay thế, quá trình tổ hợp kết thúc khi nó đạt đến biểu thức đáp ứng yêu cầu đầu ra của người dùng hoặc không thể tìm thấy bất kỳ luật nào được áp dụng. Trong ví dụ này, quá trình tổ hợp sẽ dừng lại khi nó đạt đến các thuộc tính *CustomsCost* và *ShippingDate* như trong yêu cầu của người dùng. Khi quá trình tổng hợp dừng lại, giá trị của thuộc tính qosCost là 11.

and(Author, Book, Address, qosCost(0))
→ and(Author, Book, Address, Publisher, qosCost(1)) [BookToPublisherService]
→ and(Author, Book, Address, Publisher, Electronic, qosCost(3)) [WaysOfOrderService]
→ and(Author, Book, Address, Publisher, Electronic, OrderData, qosCost(6)) [ElectronicOrderService]
→ and(Author, Book, Address, Publisher, Electronic, OrderData, CustomsCost, qosCost(8)) [CustomsCostService]
→ and(Author, Book, Address, Publisher, Electronic, OrderData, CustomsCost, ShippingDate, qosCost(11)) [ShippingService]

Hình 3. Minh họa quá trình tổ hợp dịch vụ web dựa trên TRS

2.3.2. Áp dụng giải thuật Tìm kiếm tốt nhất trước (Best-First Search – BFS) để tối ưu cho vấn đề tổ hợp dịch vụ web

Nói chung, phương pháp áp dụng TRS cho vấn đề tổ hợp dịch vụ web như mô tả trong Phần 2.3.1. lấy các luật để xử lý theo thứ tự xuất hiện của chúng trong kho lưu trữ. Nếu luật đã chọn không phải là luật tốt nhất (theo thuộc tính QoS), hệ thống sẽ tạo ra kết quả không tối ưu.

Ví dụ, nếu chúng ta thêm một dịch vụ web mới là *BookElectronicOrderService*, dịch

vụ này cũng nhận vào Sách (*Book*), Tác giả (*Author*); trả về nhiều thông tin như dữ liệu đơn hàng (*OrderData*) và chi phí (*CustomCost*). Tuy nhiên, chi phí để sử dụng dịch vụ này là rất lớn, đến 20 đơn vị tiền tệ. Với trường hợp này, phương pháp được trình bày trong Phần 2.3.1. có thể tìm ra tổ hợp dịch vụ nhưng không phải là phương án tối ưu. Vì vậy, phương pháp tìm kiếm tốt nhất trước (Best First Search - BFS) sẽ được áp dụng để khắc phục nhược điểm này.

Thuật toán BFS cho việc chọn luật trong TRS được mô tả như trong Thuật toán 1. Thuật toán này dử dụng hàm đánh giá *f* dựa trên các thuộc tính QoS. Thuật toán trả về danh sách các luật được xếp thứ tự ưu tiên theo hàm chi phí *f* tăng dần.

Thuật toán 1. Thuật toán BFS cho TRS

Đầu vào: Tập hợp các luật đại diện cho các dịch vụ web thành phần (R_i: $I_i \rightarrow O_i$), luật mục tiêu đại diện cho yêu cầu của người dùng ($I_r \rightarrow O_r$)

Đầu ra: Các luật đại diện cho tổ hợp dịch vụ web với giá trị QoS tốt nhất ($R_m \bullet ... \bullet R_n$)

1: Thêm I_r vào danh sách *OPEN*

2: **If** *OPEN* rỗng **then return** *unsuccessful error*

3: Lấy luật *x* từ *OPEN* với giá trị hàm mục tiêu *f* là tốt nhất

4: Thêm *x* vào danh sách *CLOSED*

5: **if** *x* thỏa O_r **then return** danh sách CLOSED

6: **for each** I_i thỏa *x* **do**

7: Tính hàm mục tiêu $f(I_i)$

8: **if** I_i chưa có trong *OPEN* và *CLOSED* **then** Thêm I_i vào *OPEN*

9: **else**

10: Cập nhật giá trị của $f(I_i)$ nếu giá trị mới tốt hơn

11: **if** I_i đã có trong *CLOSED* **then** Đưa I_i vào *OPEN*

12: **goto** bước 2

Áp dụng Thuật toán 1 vào hệ thống gồm có 6 dịch vụ web như trong Bảng 1 và yêu cầu của người dùng như mô tả ở trên, hàm tối ưu (heuristic) *f* của dịch vụ web tổ hợp được tính toán ở mỗi bước. Quá trình tổ hợp dịch vụ web dựa trên TRS sử dụng thuật toán BFS diễn tiến qua các bước như sau:

Bước 1: *OPEN* = {[*Author, Book, Address*]}

Bước 2: *OPEN* không rỗng

Bước 3&4: *x* = [*Author, Book, Address*], *f(x)* = 0. Chọn *x*, thêm *x* vào *CLOSED*,

CLOSED={[*Author, Book, Address*]}

Bước 5: *x* chưa thỏa mãn mục tiêu

Bước 6: Chúng ta có 2 luật có vế trái thỏa mãn *x*, đó là *BookToPublisherService* và *BookElectronicOrderService*

Bước 7: Ta tính được *f(BookToPublisherService)=1, (BookElectronicOrderService)=20*

Bước 8: Cả hai luật này chưa có trong *OPEN* nên thêm chúng vào *OPEN*

Bước 9: Quay lại Bước 2, chọn luật *BookToPublisherService* cho nó có giá trị hàm f tốt nhất.

Quá trình này lặp lại cho đến khi thỏa yêu cầu người dùng. Kết quả chúng ta có một chuỗi các luật tối ưu như sau: *BookToPublisherService • WaysOfOrderService • ElectronicOrderService • CustomsCostService • ShippingService*. Đây cũng là danh sách các dịch vụ web thành phần trong dịch vụ web tổ hợp kết quả.

2.4. Kết quả thực nghiệm

Bảng 2. Tập dữ liệu thực nghiệm

Tập dữ liệu	Số lượng dịch vụ	Mô tả
Travel Booking(TB)	20	Gồm các dịch vụ web cung cấp thông tin để phục vụ việc đặt vé du lịch
Book Store(BS)	40	Gồm các dịch vụ mua sách, thanh toán trực tuyến và giao sách cho khách hàng
Online Film Store(OFS)	60	Bao gồm các dịch vụ hỗ trợ tìm kiếm, mua và xem phim trực tuyến
Medical Services (MS)	80	Bao gồm các dịch vụ cung cấp thông tin để tra cứu bệnh viện, điều trị, thuốc men, v.v.
Education Services(EDS)	100	Bao gồm các dịch vụ liên quan đến giáo dục như học bổng, khóa học, bằng cấp, v.v.

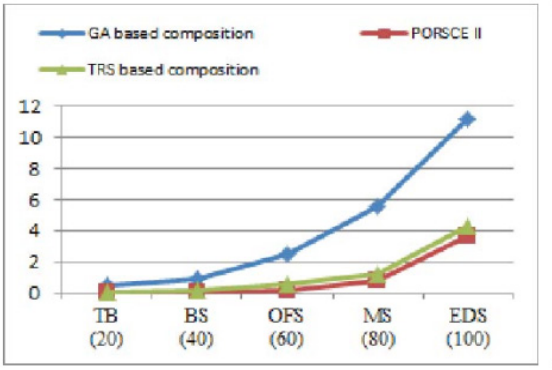
Trong phần này, chúng tôi sẽ trình bày các kết quả thực nghiệm của phương pháp tiếp cận của chúng tôi. Để đánh giá, chúng tôi thử nghiệm cách tiếp cận của mình trên các bộ dữ liệu thực thu được từ dự án OWLS-TC [9]. OWLS-TC cung cấp hơn 1000 dịch vụ web ngữ nghĩa được phân loại thành các miền khác nhau và được mô tả bằng ngôn ngữ OWL-S . Trong tập dữ liệu này, chúng tôi chọn năm tập dữ liệu con với số lượng dịch vụ web trong mỗi nhóm dao động từ 20 đến 100 dịch vụ như trong Bảng 2. Bên cạnh đó, chúng tôi cũng đánh giá và so sánh với các cách tiếp cận khác để cho thấy sự hiệu quả của cách tiếp cận của mình.

Chúng tôi tiến hành các kịch bản thử nghiệm dựa trên ba cách tiếp cận khác nhau, được đặt tên là *Lược đồ tổ hợp đầy đủ dựa trên giải thuật di truyền (GA)* [2]; Tổ hợp dịch vụ web dựa trên lập kế hoạch trí tuệ nhân tạo, đại diện bởi khung thức *PORSCE II* [3]; và cách tiếp cận của chúng tôi – *tổ hợp dịch vụ web dựa trên TRS*. Cách thứ nhất là cách tiếp cận điển hình với việc chọn tổ hợp tối ưu bằng các thuật toán tối ưu hóa như thuật toán di truyền. Lưu ý rằng, các hướng tiếp cận dựa trên lập kế hoạch hiện tại không thể giải quyết các ràng buộc mềm và PORSCE II cũng không phải là ngoại lệ. Cách tiếp cận của chúng tôi hỗ trợ cả hai ràng buộc: ràng buộc về thuộc tính chức năng (ràng buộc cứng) và ràng buộc về thuộc tính QoS (ràng buộc mềm). Với mỗi tập dữ liệu và mỗi cách tiếp cận, chúng

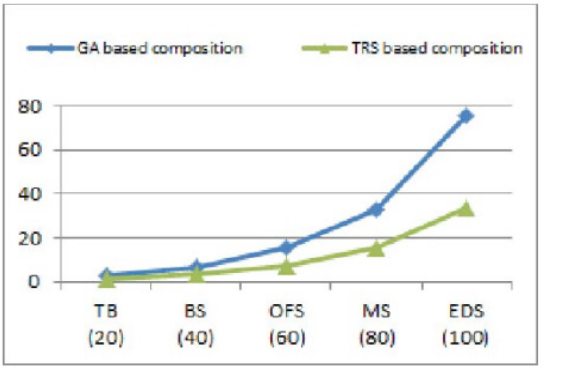
tôi chạy 10 yêu cầu người dùng khác nhau với giá trị ngẫu nhiên của thuộc tính QoS. Giá trị về thời gian thực thi và lượng bộ nhớ đã sử dụng được thu thập trong Bảng 3, được minh họa trực quan trong Hình 4.

Bảng 3. Kết quả thực nghiệm đo theo thời gian xử lý (giây) và lượng bộ nhớ sử dụng (Kb)

Hướng tiếp cận tổ hợp	Loại R. buộc	TB(20)		BS(40)		OFS(60)		MS(80)		EDS(100)	
		Thời gian	*Bộ nhớ*	*Thời gian*	*Bộ nhớ*	*Thời gian*	*Bộ nhớ*	*Thời gian*	*Bộ nhớ*	*Thời gian*	*Bộ nhớ*
Sử dụng GA	Cứng	0.50	6,539	0.90	7,192	2.53	8,066	5.59	9,521	11.20	11,348
	Mềm	2.51	10,427	6.80	11,373	15.60	12,971	32.85	15,172	75.85	19,391
PORSCE II	Cứng	0.02	6,293	0.04	6,552	0.13	7,066	0.85	7,801	3.70	8,818
	Mềm	-	-	-	-	-	-	-	-	-	-
Dựa trên TRS	Cứng	0.04	4,257	0.15	4,334	0.55	4,616	1.25	4,850	4.33	5,528
	Mềm	1.25	4,307	3.35	4,503	7.53	4,932	15.25	5,210	33.25	6,288



Ràng buộc cứng



(b) Ràng buộc mềm

Hình 4. Minh họa trực quan kết quả thực hiện về thời gian xử lý (giây)

3. Kết luận

Cách tiếp cận tổ hợp dịch vụ web được đề cập trong bài báo này là cách tiếp cận hoàn toàn mới dựa trên lý thuyết về viết lại số hạng (term rewriting). Phương pháp này chuyển đổi các dịch vụ web thành phần có sẵn thành các luật đi kèm với thông tin về các thuộc tính chức năng và lẫn các thuộc tính QoS. Hệ thống có khả năng chọn ra một luật tốt nhất ở mỗi bước viết lại cho đến khi đạt được biểu thức thỏa mãn yêu cầu của người dùng. Trong quá trình tổ hợp, phương pháp này cho phép cập nhật số lượng các dịch vụ mà không ảnh hưởng đến việc thực thi hệ thống.

Bên cạnh đó, nghiên cứu này cũng cho phép tổ hợp dịch vụ web đáp ứng cả ràng buộc cứng lẫn ràng buộc mềm. Trong tương lai, để có được hiệu suất tốt hơn, phương pháp này sẽ được mở rộng bằng cách áp dụng các cơ chế như lập chỉ mục và (hoặc) phân cụm các dịch vụ web để truy xuất các dịch vụ web (hay các luật) hiệu quả hơn. Việc tích hợp các kỹ thuật khai phá dữ liệu đó vào TRS được xem là hướng mở rộng trong tương lai của nghiên cứu này.

TÀI LIỆU THAM KHẢO

[1]. Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decades overview," Information Sciences, vol. 280, pp. 218–238, 2014.

[2]. M. AllamehAmiri, V. Derhami, and M. Ghasemzadeh, "Qos-based web service composition based on genetic algorithm," Journal of AI and Data Mining, vol. 1, no. 2, pp. 63–73, 2013.

[3]. O. Hatzi, D. Vrakas, and I. Vlahavas, "The PORSCE II framework: Using AI planning for automated semantic web service composition," The knowledge Engineering Review, vol. 28, no. 02, pp. 137–156, 2013.

[4]. Y. W. M. Maung and A. A. Hein, "Colored petri-nets (CPN) based model for web services composition," IJCCER, vol. 2, pp. 169–172, 2014.

[5]. M. Klusch, A. Gerber, and M. Schmidt, "Semantic web service composition planning with OWLS-XPlan," in Proceedings of Semantic Web and Agents, AAAI Press, 2015.

[6]. H. Jingjing, Z. Wei, Z. Xing, and Z. Dongfeng, "Web service composition automation based on timed automata," Appl. Math, vol. 8, no. 4, pp. 2017–2024, 2014.

[7]. M. Bezem, J. W. Klop, and R. de Vrijer, Term rewriting systems., Cambridge University Press, 2003.

[8]. F. Baader and T. Nipkow, Term rewriting and all that. Cambridge university press, 1999.

[9]. M. Klusch, "Owls-tc: Owl-s service retrieval test collection, version 2.1," available at: http://projects.semwebcentral.org/projects/owls-tc/.

THE APPLICATION OF TERM REWRITING SYSTEM TO AUTOMATIC WEB SERVICE COMPOSITION

HUYNH TAN KHAI

Quang Nam University

Abstract: *With the powerful advantages of web services compared to traditional ways to solve the problems such as platform-independence, reusability, deployment,..., the number of web services increases day by day. Instead of developing a new software which satisfies the requirements, users can search for available appropriate web services. Users can search and combine available web services in order to meet their requirements. Therefore, the composition of web services that respond to users' requirements is a problem that needs solving. Composing web services automatically attracts much interest from researchers. However, current methods have some limitations such as time consumption, inconsideration of the quality of services, ... The research in this paper proposes a new approach of basing on the Term Rewriting System (TRS) to automate the web service composition process.*

Keywords: *Term Rewriting System, Web service, Web service composition, Web service recycling.*