ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

CAO THỤC TUYẾT TRINH

NGHIÊN CỨU PHƯƠNG PHÁP NÉN DỮ LIỆU ĐỂ TĂNG HIỆU QUẢ LƯU TRỮ CHUỖI DNA

Ngành: Hệ thống thông tin
Chuyên ngành: Hệ thống thông tin
Mã số: 60 48 01 04

TÓM TẮT LUẬN VĂN THẠC SĨ HỆ THỐNG THÔNG TIN

HÀ NỘI – 2016

GIỚI THIỆU

Những tiến bộ kỹ thuật trong việc sắp xếp các chuỗi đa lượng đã và đang tạo ra một khối lượng khổng lồ dữ liệu các chuỗi gen phục vụ cho y sinh học hiện đại. Kích thước dữ liệu ngày càng tăng đặt ra vấn đề về chi phí cho không gian lưu trữ và tốc độ truy cập, truyền tải.

Bộ gen của con người gồm khoảng 3 tỉ đặc trưng trên 23 cặp nhiễm sắc thể. Cơ sở dữ liệu hệ gen là vô cùng lớn và phức tạp. Để lưu trữ, truy cập và xử lý dữ liệu này một cách hiệu quả là một nhiệm vụ rất khó khăn. Do vậy cần một thuật toán nén hiệu quả để lưu trữ khối lượng dữ liệu khổng lồ này. DNA là tên hóa học chỉ các phân tử mang cấu trúc gen trong tất cả các thực thể sống. DNA gồm một chuỗi được tạo nên từ 4 loại đơn vị nucleotide, mỗi loại gồm: 1 đơn vị đường carbon 5, 1 nhóm phốt phát và 1 trong 4 thành phần cơ bản adenine, cystosine, guanine và thymine gọi là các bazơ. Mỗi phân tử đường được gắn với ¼ thành phần cơ bản. Dạng đơn giản nhất của DNA trong 1 tế bào là 1 cấu trúc dây xoắn đôi, trong đó 2 sợi DNA đơn xoắn quanh nhau theo hình xoắn ốc thuận tay phải. Do chuỗi DNA gồm 4 thành phần A, T, G, C nên cách hiệu quả nhất để biểu diễn chúng là sử dụng 2 bits cho mỗi kí hiệu. Tuy nhiên, nếu ứng dụng phần mềm nén tiêu chuẩn như “Unix\compress and \compact” thì các tệp sẽ bị mở rộng ra hơn 2 bit trên mỗi thành phần. Những phần mềm này được thiết kế để nén văn bản, trong khi đó những quy tắc trong chuỗi DNA thì lại phức tạp hơn. Mã hóa 2 bit là cách hiệu quả nếu các bazơ xuất hiện ngẫu nhiên trong chuỗi. Nhưng cuộc sống của một sinh vật là không ngẫu nhiên, do đó chuỗi DNA xuất hiện trong 1 sinh vật là không ngẫu nhiên và có một số ràng buộc. Nén chuỗi DNA là một nhiệm vụ rất thách thức. Đặc trưng phức tạp của một chuỗi DNA nằm ở chỗ đó là một chuỗi các chỉ số độ dài khác nhau biểu diễn một phạm vi có thể dự đoán được của các thành phần cơ bản cấu tạo nên DNA. Những đặc trưng phức tạp này cho phép tìm kiếm những cấu trúc lặp bên trong một nhiễm sắc thể hoặc qua nhiều nhiễm sắc thể. Và cũng chính những đặc trưng này được sử dụng để tìm ra khoảng cách tiến hóa và cấu trúc nên cây phát sinh loài. Do sự cấu tạo phức tạp này mà có thể thấy là trong thực tế không có 1 chương trình nén tệp thông thường nào có thể nén chuẩn được chuỗi DNA. Nhiều thuật toán nén dành riêng cho chuỗi DNA đã được phát triển từ khoảng 10 năm trước. Sự thật là nén chuỗi DNA là một việc khó đối với các thuật toán nén cơ bản, nhưng từ quan điểm của lý thuyết nén thì nó là một đề tài thú vị cho việc tìm hiểu thuộc tính của nhiều thuật toán nén. Ở đây chúng ta nói về phương pháp luận của các phương pháp nén một cách ngắn gọn.

Hiện nay, kỹ thuật nén dữ liệu chuỗi gen được sử dụng rộng rãi trong lưu trữ dữ liệu sinh học. Có hàng trăm thuật toán đã được đề xuất cho nén dữ liệu DNA nhưng nhìn chung các thuật toán nén được chia thành một số cách tiếp cận như sau: (1) mã hóa bit, (2) nén dựa trên bộ từ điển, (3) nén thống kê, và (4) nén tham chiếu [1,2]. Trong khuôn khổ luận văn, người viết chỉ trình bày một số thuật toán tiêu biểu cho từng phương pháp đã nêu và hầu hết các phương pháp đều nhằm hai mục đích chính: đạt được tỉ lệ nén cao nhất có thể để tiết kiệm không gian lưu trữ và đạt được tốc độ nén/giải nén cũng như truy cập thông tin nhanh chóng.

- *Thuật toán mã hóa bit*: sử dụng mã hóa độ dài cố định hai hoặc nhiều kí tự trên một byte đơn [38].
- *Thuật toán nén dựa trên bộ từ điển*: hay còn gọi là thuật toán thay thế, thuật toán thay thể các chuỗi lặp bằng việc tham chiếu tới một từ điển, từ điển này được xây dựng trong thời gian chạy hoặc ngoại tuyến [39, 40].
- *Thuật toán nén thống kê:* hay còn gọi là thuật toán mã hóa entropy, bắt nguồn từ một mô hình lấy xác suất dữ liệu đầu vào. Dựa trên các chuỗi khớp từng phần của tập con đầu vào, mô hình dự đoán kí tự tiếp theo trong chuỗi. Tỉ lệ nén cao có thể đạt được nếu mô hình luôn chỉ ra được xác suất cao cho kí tự tiếp theo, nghĩa là dự đoán đáng tin cậy [15, 41].
- *Thuật toán nén tham chiếu:* tương tự nén dựa trên bộ từ điển, thuật toán thay thế các chuỗi con dài của đầu vào với tham chiếu tới chuỗi khác. Tuy nhiên, tham chiếu này trỏ tới các chuỗi bên ngoài mà không phải là một phần của dữ liệu nén.

Trung bình thuật toán mã hóa bit đạt tỉ lệ 4:1, thuật toán nén dựa trên bộ từ điển đạt 4:1 đến 6:1, thuật toán xác suất đạt 4:1 tới 8:1, riêng thuật toán nén tham chiếu có thể đạt tới tỉ lệ 400:1 [2] hoặc có thể cao hơn với điều kiện lý tưởng về chuỗi tham chiếu và chỉ số nén.

Thuật toán nén tham chiếu mang tới một tiềm năng lớn cho nén chuỗi đa lượng, điển hình là chuỗi DNA. Tương tự như thuật toán nén dựa trên bộ từ điển nhưng do các chuỗi mã hóa tham chiếu tới tập hợp chuỗi tham chiếu bên ngoài nên tốc độ nén cao hơn và giải mã cũng thuận lợi hơn. Các chuỗi DNA được nén tham chiếu bao gồm các phần khớp nhau về khoảng và có thể đạt tới tốc độ nén cao nhất đối với nén trong cùng loài. Tuy vẫn còn một số bất lợi cho nén các hệ gen khác loài nhưng nén tham chiếu rõ ràng đã cho thấy lợi thế về tỉ lệ nén và tốc độ nén nếu đạt được một số điều kiện lý tưởng. Vì việc tìm ra chuỗi tham chiếu phù hợp là điều khá khó khăn do các chuỗi gen nghiên cứu là các mẫu

được lấy ngẫu nhiên từ một tập hợp lớn các loài. Bên cạnh việc tìm kiếm chuỗi khớp xác định thì việc khớp giữa đầu vào và chuỗi tham chiếu cũng khá là phức tạp. Tuy nhiên, phương pháp tìm kiếm một chuỗi tham chiếu tốt có thể dựa trên *băm k-mer*. Sự tương đồng cao của *k*-mers đưa ra một tiềm năng lớn cho việc nén dựa trên tham chiếu. Có nhiều khung nén đã được phát triển dựa trên thuật toán nén tham chiếu. Qua thời gian, mỗi phương pháp nén dựa trên tham chiếu đều được cải tiến về phương thức lưu trữ dữ liệu, đánh chỉ số chuỗi gen, thuật toán tìm kiếm chuỗi tham chiếu tốt nhất hay viết lại tham chiếu và tìm kiếm chuỗi khớp tối ưu. Tất cả những cải tiến này đều cho thấy những hiệu quả khả quan đạt được về tỉ lệ cũng như tốc độ nén/giải nén chuỗi gen của thuật toán nén dựa trên tham chiếu. Đây cũng chính là lý do mà trong luận văn này, người viết tập trung nghiên cứu, thực nghiệm so sánh kết quả nén chuỗi đa lượng DNA dựa trên thuật toán nén tham chiếu với thuật toán nén tiêu biểu là JDNA, phát triển dựa trên thuật toán được sử dụng bởi FRESCO [25], được tối ưu với 3 phương pháp cải tiến là lựa chọn tham chiếu, viết lại tham chiếu và nén thứ tự hai. Ngoài ra JDNA còn thêm hai cải tiến để tối ưu về tỉ lệ nén và thời gian nén/giải nén là (1) sử dụng tính tương đương và (2) thay thế chỉ số tham chiếu hoàn toàn bằng một phương thức chỉ số theo yêu cầu. Những cải tiến này cho kết quả rất tốt về tỉ lệ nén, có thể đạt tỉ lệ nén tới 1000:1 với điều kiện lý tưởng. Người viết cũng thực hiện thực nghiệm bổ sung so sánh thuật toán tham chiếu JDNA với thuật toán nén dựa trên phương thức khác là Lempel-Ziv, nén dựa trên bộ từ điển và Huffman, nén dựa trên xác suất thống kê để thấy rõ được tính ưu việt của thuật toán tham chiếu này về cải thiện tỉ lệ nén, tốc độ giải nén và dung lượng lưu trữ. Tuy kết quả đạt được về tỉ lệ nén và thời gian nén của thực nghiệm bổ sung chưa đạt được tỉ lệ mong đợi cao nhất của thuật toán nén tham chiếu do còn hạn chế về môi trường thực nghiệm nhưng đã góp phần chứng minh những nhận định về hiệu quả của thuật toán nén tham chiếu đối với việc nén chuỗi gen mà người viết đã nghiên cứu.

Bố cục luận văn được chia thành 3 chương. Chương 1 trình bày về tổng quan các phương thức nén dữ liệu sử dụng cho nén chuỗi DNA. Thuật toán nén tham chiếu cụ thể mà người viết luận văn tập trung nghiên cứu, thuật toán nén tham chiếu JDNA, được trình bày ở chương 2. Chương 3 của luận văn mô tả môi trường thực nghiệm so sánh thuật toán nén tham chiếu JDNA với hai thuật toán thuộc phương thức nén khác và một số phân tích đánh giá của người viết về kết quả đạt được. Cuối cùng là kết luận về hiệu quả cũng như hạn chế còn tồn tại và hướng phát triển trong tương lai.

CHƯƠNG 1 – TỔNG QUAN VỀ THUẬT TOÁN NÉN DỮ LIỆU

1.1. THUẬT TOÁN MÃ HÓA BIT (Naïve Bit)

Thuật toán mã hóa bit sử dụng các bit trạng thái để biểu diễn dữ liệu nén. 4 bazơ đặc trưng của DNA được mã hóa bởi 2 bit (4 trạng thái). Kỹ thuật nén thẳng dữ liệu chuỗi DNA là mã hóa 4 bazơ vừa đủ trong một byte (8 bit) theo mã hóa bit.

Mỗi kí tự ở đầu vào được thay thế bởi 2 bit sử dụng phép thay thế {A = 00, C = 01, G = 10, T = 11}.

Tỉ lệ nén của thuật toán mã hóa bit là 4:1 nếu kích thước của chuỗi kí tự đầu vào là 4 hoặc ít hơn 4:1 nếu nhiều hơn 4 kí tự [2].

1.1.1. Mã hóa trực tiếp phần khác biệt (thuật toán 2D)

Thuật toán 2D được thiết kế để cung cấp một giao thức nén chuỗi nucleotit thông thường. Giao thức này có thể phân biệt dữ liệu chuỗi và dữ liệu phần bù, từ đó đưa ra sự điều chỉnh phù hợp giữa nén dữ liệu chung chung và cụ thể. Thuật toán 2D có những mục tiêu như sau:

- Thời gian thực hiện tuyến tính cho việc hỗ trợ các tập dữ liệu lớn.
- Hỗ trợ bao gồm cả những kí tự phụ mà không phải thành phần của tập bazơ nucleotit.
- Mã hóa trực tiếp pha đơn.
- Nén không mất dữ liệu.
- Không phân biệt loại chuỗi.
- Giải nén chuỗi polipeptit (mỗi peptit gồm 10 tới 100 amino axit).
- Sử dụng được cùng với phương pháp nén khác.

1.1.2. Thuật toán nén DNABIT

Thuật toán DNABIT nén các chuỗi DNA của toàn bộ hệ gen (cả các chuỗi lặp và không lặp) theo 2 pha. Hai pha lần lượt sử dụng 5 kỹ thuật được gọi là *kỹ thuật 2 Bit, 3, 5, 7* và *9 Bit.*

2 pha: (1) Kỹ thuật Bit chẵn: kỹ thuật 2Bit; (2) Kỹ thuật Bit lẻ: kỹ thuật 3Bit, kỹ thuật 5Bit, kỹ thuật 7Bit, kỹ thuật 9Bit.

- **Kỹ thuật Bit chẵn:**

Chuỗi DNA được gán 2 bit cho mỗi bazơ đơn. Các bazơ đó là {A, C, G, T}. 2 bit được gán tới các bazơ của vùng không lặp. Các bit được gán tới các bazơ đơn.

- **Kỹ thuật Bit lẻ**

Kỹ thuật Bit lẻ sử dụng 4 kỹ thuật đó là Kỹ thuật 3Bit, Kỹ thuật 5Bit, Kỹ thuật 7Bit và Kỹ thuật 9Bit.

Tuy nhiên, thuật toán DNABIT chỉ quan tâm tới nén các chuỗi lặp chính xác hoặc nghịch đảo và không lặp mà chưa tính tới các trường hợp có phần bù hoặc kí tự đặc biệt. Giả sử có một kí tự bù như N cho bazơ không phân biệt được thì việc mã hóa sẽ trở nên phức tạp. Ngoài ra sử dụng các bit nhị phân để mã hóa các bazơ như kí tự trong văn bản cũng khiến lãng phí không gian.

1.2. THUẬT TOÁN NÉN DỰA TRÊN BỘ TỪ ĐIỂN

Nén dựa trên bộ từ điển là một khung nén độc lập với các thuộc tính cụ thể của dữ liệu đầu vào. Phương thức chính của thuật toán là thay thế các phần tử dữ liệu lặp (các chuỗi con DNA) của đầu vào với tham chiếu tới một bộ từ điển. Những phần lặp thường được phát hiện bằng cách lưu lại các chuỗi xuất hiện trước đó. Bộ từ điển sẽ được tái cấu trúc theo nhiều cách khi thực hiện trong quá trình giải nén.

1.2.1. LZ77

Trong thuật toán LZ77, từ điển là phần mã hóa chuỗi trước tiên. Bộ mã hóa sẽ kiểm tra chuỗi đầu vào bằng cách nhấn vào dịch vụ cửa sổ trượt gồm 2 phần: bộ đệm tìm kiếm và bộ đệm xem thẳng. Một bộ đệm tìm kiếm gồm một phần chuỗi mới được mã hóa và bộ đệm xem thẳng gồm phần tiếp theo của chuỗi sẽ được mã hóa.

- ***Một số hạn chế của thuật toán LZ77***

- Trong thuật toán LZ77 gốc, Lempel và Ziv đã đề xuất toàn bộ chuỗi được mã hóa là độ dài và phần bù (offset), thậm chí cả chuỗi tìm thấy mà không khớp.
- Trong LZ77, bộ đệm tìm kiếm dài hàng nghìn bytes, trong khi bộ đệm xem thẳng chỉ 10 bytes
- Quá trình mã hóa tiêu tốn thời gian do phải thực hiện số lượng so sánh lớn để tìm mẫu khớp.
- LZ77 không có từ điển ngoài nên tạo ra vấn đề trong khi giải mã trên máy khác. Việc tạo ra phần bù làm tốn không gian và bước không cần thiết này cũng làm tăng thời gian thực hiện thuật toán.

1.2.2. LZ78

LZ78 là một thuật toán nén dựa trên từ điển mà duy trì một từ điển rõ ràng. Đầu ra được mã hóa gồm hai thành phần: một chỉ số tham chiếu tới từ điển mẫu khớp đầu vào và kí tự không khớp đầu tiên. Thuật toán cũng bổ sung cặp chỉ số và kí tự cho từ điển. Với phương pháp này, thuật toán tạo nên từ điển.

Thuật toán LZ78 có khả năng bắt được các mẫu và giữ chúng xác định nhưng nó cũng có một hạn chế nghiêm trọng. Từ điển cứ lớn dần lên mãi mà không có điểm dừng.

Có thể nói thuật toán nén Lempel Ziv (LZ77, LZ78) dựa trên từ điển là một thuật toán nén không mất dữ liệu, ứng dụng tốt trong nén văn bản và các chuỗi khớp ngẫu nhiên. Nhưng để nén chuỗi DNA thì thuật toán này không thỏa mãn và không phù hợp.

1.3. THUẬT TOÁN NÉN XÁC SUẤT THỐNG KÊ

Thuật toán thống kê tạo một mô hình thống kê dữ liệu đầu vào mà trong hầu hết các trường hợp được biểu diễn như một cấu trúc dữ liệu cây tiền tố hoặc xác suất. Các chuỗi con với tần suất cao trong gen được biểu diễn với mã ngắn hơn. Tỉ lệ nén phụ thuộc chất lượng của mô hình cũng như sự tồn tại của những mẫu có thể phát hiện được ở đầu vào.

Tỉ lệ nén của thuật toán nén xác suất thường là trong khoảng 4:1 và 8:1. Tỉ lệ nén phụ thuộc chủ yếu vào sự phân bố các kí tự đầu vào và bộ nhớ sẵn có cho cấu trúc phân bố tần suất.

1.3.1. Thuật toán nén HuffBit sử dụng cây nhị phân mở rộng với mã Huffman

Nén Huffbit sử dụng khái niệm cây nhị phân mở rộng để nén; gán 0 và 1 cho các cây con trái và phải. Huffbit là một thuật toán xử lý 2 chiều; tạo ra 1 cây nhị phân mở rộng ở pha khởi tạo. Trong trường hợp tốt nhất thì thuật toán sẽ đạt tỉ lệ nén là 1.006 bit trên mỗi bazơ. Thuật toán đơn giản nên tỉ lệ nén đạt được không thỏa mãn và nó cần quét toàn bộ chuỗi gen để đạt được tần suất của các bazơ đơn trước khi bắt đầu trình tự nén.

Thuật toán đưa ra là một quá trình xử lý 2 chiều:

(a) Thực hiện cấu trúc cây nhị phân mở rộng

(b) Mã hóa Huffman từ cây nhị phân mở rộng được sử dụng để tính số bit của độ dài chuỗi mã hóa. Tỉ lệ nén là kích thước đã nén trên kích thước chưa nén.

Thuật toán chưa kiểm tra được trên hệ gen lớn.

1.3.2. Thuật toán Expert Markov (XM)

Thuật toán mã hóa mỗi kí tự bằng cách đánh giá xác suất dựa trên thông tin có được từ kí tự trước nó. Là một phương pháp thống kê, thuật toán XM nén mỗi kí tự bằng cách xác định phân bố xác suất cho kí tự và sau đó sử dụng một khung nén sơ cấp để mã hóa nó [15]. Phân bố xác suất tại một vị trí dựa trên kí tự nhìn thấy trước nó. Tương ứng với nó, bộ giải mã cũng tìm tất cả các kí tự đã

giải mã trước nó để có thể tính phân bố xác suất đồng nhất và có thể khôi phục lại kí tự tại vị trí đó.

- **Các loại chuyên gia**

Một chuyên gia có thể là bất kỳ thứ gì mà đưa ra được một phân bố xác suất hợp lý cho một vị trí trong chuỗi. Một chuyên gia đơn giản có thể là một mô hình Markov (*Markov expert*). Một chuyên gia Markov thứ tự *k* sẽ đưa ra xác suất của một kí tự ở một vị trí kí tự trước *k.* Về cơ bản, *Markov expert* cung cấp phân bố xác suất cơ sở của các kí tự trên chuỗi. Ở đây sử dụng *Markov expert* thứ tự 2 cho DNA và thứ tự 1 cho protein.

Một loại chuyên gia khác đó là chuyên gia Markov ngữ cảnh (*context Markov expert)*, phân bố xác suất của chuyên gia này không dựa trên toàn bộ lịch sử của chuỗi mà dựa trên ngữ cảnh hạn chế trước nó.

Khả năng nén các chuỗi sinh học xuất phát từ các chuỗi con lặp. Bởi vậy, các chuyên gia có thể sử dụng được đặc tính này là rất quan trọng. XM sử dụng một sao chép chuyên gia (*copy expert)* mà coi kí tự tiếp theo như một phần của vùng sao chép từ một phần bù cụ thể.

1.4. THUẬT TOÁN NÉN THAM CHIẾU

1.4.1. Đặc trưng thuật toán tham chiếu

Cũng như kỹ thuật nén dựa trên từ điển, thuật toán này thay thế các chuỗi con dài của đầu vào được nén với tham chiếu tới chuỗi khác. Tuy nhiên, tham chiếu này trỏ tới các chuỗi bên ngoài, không phải là một phần của dữ liệu đầu vào được nén. Ngoài ra, tham chiếu của thuật toán này là tĩnh, còn tham chiếu của thuật toán cơ sở từ điển được mở rộng trong quá trình nén dữ liệu. Do các chuỗi mã hóa tham chiếu tới tập hợp chuỗi tham chiếu bên ngoài nên tốc độ nén cao hơn và giải mã cũng thuận lợi hơn.

Thuật toán nén tham chiếu có thể đạt tỉ lệ nén 400:1 hoặc tốt hơn nếu đưa ra được chuỗi tham chiếu khớp hiệu quả.

1.4.2. Các thuật toán nén tham chiếu

Có nhiều khung nén tham chiếu đã được phát triển dựa trên thuật toán nén tham chiếu. Khung nén [32] đề xuất chỉ lưu sự khác nhau giữa chuỗi đầu vào nén và chuỗi tham chiếu. Thuật toán xem xét 3 loại thay đổi bazơ đơn: chèn, xóa và thay thế. Kết quả chính đạt được là sự phân tích về cách mã hóa các số nguyên cho vị trí tham chiếu tương đối và tuyệt đối. Tuy nhiên, thuật toán này nhấn mạnh việc lựa chọn chuỗi tham chiếu có ảnh hưởng tới tỉ lệ nén nhiều hơn khung mã hóa số nguyên trên thực tế.

GRS [20] là một công cụ nén tham chiếu khác dựa trên chương trình Unix *diff,* tìm kiếm chuỗi con dài nhất trong hai chuỗi đầu vào.

RLZ [30] mô tả một phương pháp dựa trên việc tự đánh chỉ số. Thuật toán hoạt động như sau: thuật toán nén chuỗi đầu vào với mã hóa LZ77 liên quan tới mảng hậu tố của chuỗi tham chiếu. Không lưu các chuỗi thô kể cả đó có là chuỗi khớp ngắn với tham chiếu được mã hóa. Thuật toán coi việc xem xét các chuỗi tham chiếu là vấn đề sống còn.

GreEn [31] một khung nén tham chiếu dựa trên hệ chuyên gia mới được đề xuất gần đây. Lấy ý tưởng từ khung nén XM (eXpert Markov) không tham chiếu, GreEn sao chép hệ chuyên gia tìm kiếm chuỗi khớp *k-mers* giữa đầu vào và tham chiếu.

Vượt lên trên nhiều khung nén tham chiếu với nhiều cải thiện đạt được về tỉ lệ nén và không gian lưu trữ, JDNA được biết đến là một khung nén tham chiếu hiệu quả, xây dựng dựa trên một thuật toán nén tham chiếu nhanh và mã nguồn được mở để cộng đồng cùng sử dụng và cải tiến. JDNA sử dụng một bảng *k-mer* để đánh chỉ số cho chuỗi tham chiếu, JDNA tốn ít thời gian nén, phần lớn thời gian thực hiện là dùng cho việc đánh chỉ số. JDNA sử dụng thư viện mã nguồn mở được dùng bởi FRESCO [21] đáp ứng được 4 thiết kế chính khi thực hiện một thuật toán nén tham chiếu đó là: (1) định dạng đầu vào, (2) cấu trúc chỉ số tham chiếu, (3) thuật toán nén và (4) định dạng thứ tự chuỗi cho các tệp nén [25]. Bên cạnh đó JDNA còn thêm hai cải tiến để tối ưu về thời gian nén/giải nén và dung lượng lưu trữ là (1) sử dụng tính tương đương và (2) thay thế chỉ số tham chiếu hoàn toàn bằng một phương thức chỉ số theo yêu cầu. Giao diện chuỗi của khung nén cho phép tải chuỗi từ tệp và viết chuỗi vào tệp với định dạng dữ liệu RAW hoặc FASTA. Một chỉ số *index* dựa trên chỉ số băm k-mer được sử dụng để tìm chuỗi khớp cho chuỗi nén dựa trên tham chiếu. Tiếp theo giao diện nén sẽ cung cấp sự thực hiện hai hàm, một cho nén chuỗi vào danh sách các đầu vào khớp tham chiếu và một hàm khác cho giải nén. Cuối cùng là thực hiện chuỗi hóa, sắp xếp một danh sách khớp tham chiếu vào một tệp với xử lý 3 chuẩn: (1) định dạng ASCII, (2) mã hóa DELTA, và (3) dạng nhị phân rút gọn (COMPACT). Có thể nói JDNA đạt được hiệu quả nén chuỗi đa lượng đáng mong đợi và có tiềm năng mở rộng, cải tiến khá lớn. Ở chương 2, người viết sẽ tập trung trình bày chi tiết về khung nén JDNA, cách mà khung nén này kế thừa những đặc trưng của FRESCO và những cải tiến mà thuật toán đã thực hiện để mang lại hiệu quả về tỉ lệ nén cũng như dung lượng lưu trữ khi nén chuỗi gen.

CHƯƠNG 2 – THUẬT TOÁN NÉN THAM CHIẾU JDNA

Do những lợi ích và hiệu quả mà JDNA đạt được đối với nén chuỗi gen theo phương pháp nén tham chiếu mà ở chương này, người viết luận văn lựa chọn trình bày thuật toán nén tham chiếu JDNA, một khung nén tham chiếu xây dựng trên thuật toán nén tham chiếu nhanh và mã nguồn mở của Fresco được phát hành miễn phí cho cộng đồng sử dụng và mở rộng. Hiệu quả thực hiện đạt được tỉ lệ nén cao hơn các thuật toán thuộc các phương thức khác và cao hơn cả Fresco một bậc. Thêm vào đó, người viết sẽ trình bày những đặc trưng của Fresco mà JDNA đã kế thừa, đồng thời trình bày những đặc trưng mà JDNA đã cải tiến và mang lại hiệu quả thực sự về tỉ lệ nén và dung lượng lưu trữ cũng như tốc độ nén chuỗi gen.

Thuật toán được đánh giá trên tập dữ liệu từ 3 loài: 1092 gen người, 180 gen loài cỏ Arabidopsis thaliana và 38 gen khuẩn men.

2.1. THUẬT TOÁN JDNA – Nén tham chiếu các chuỗi gen đã sắp xếp

Phương pháp đánh chỉ số theo yêu cầu được phát triển và cung cấp một công cụ nén dùng ngôn ngữ Java, gọi là JDNA cho nén tham chiếu sử dụng phương pháp này. JDNA được thiết kế để áp dụng nén tham chiếu cho các tệp gen. Có bốn loại biến gen được JDNA hỗ trợ là: SNPs, thay thế, chèn và xóa.

- SNP: một sai khác cặp bazơ đơn ở cùng vị trí từ các gen khác nhau.
- Thay thế (*Substitution)*: một chuỗi các cặp bazơ mà khác nhau ở cùng vùng của hai gen.
- Chèn (*Insertion*): một chuỗi các cặp bazơ mà được thêm vào một vùng gen. Do đó, trong cùng vùng của DNA khác thì chuỗi cặp bazơ đó không được biểu diễn.
- Xóa (*Deletion)*: một chuỗi cặp bazơ không được biểu diễn trong một vùng gen nhưng được biểu diễn trong phần chính của gen.

2.1.1. Thuật toán nén

Nén thực hiện ở các khối tệp hoặc với toàn bộ tệp trong bộ nhớ. JDNA được thiết kế để tải tệp lên tới 250MB lên bộ nhớ với các tệp lớn hơn tải vào các khối 250MB. Cả tệp tham chiếu và đầu vào đều được tải theo cách này. Mỗi khối được nén riêng biệt, bởi vậy không có sự kết nối nào giữa các khối nén thậm chí chúng ở cùng một tệp. Nếu nén khối được sử dụng thì chỉ một tệp nén được tạo ra gồm các khối được nén.

Trong JDNA, chuỗi khớp được tìm thấy trong 3 bước:

1. Kiểm tra xác định có một SNP hay không.

2. Nếu nó không phải một SNP thì thực hiện tìm kiếm cục bộ khoảng cặp bazơ cho một chuỗi khớp – khoảng này gồm hầu hết SDs; là 6 lần K, trong đó K là kích thước K-mer.
3. Nếu không tìm thấy chuỗi khớp nào thì đánh chỉ số một phần tham chiếu trong bảng K-mer và tìm kiếm vị trí hiện tại của đầu vào trong bảng.

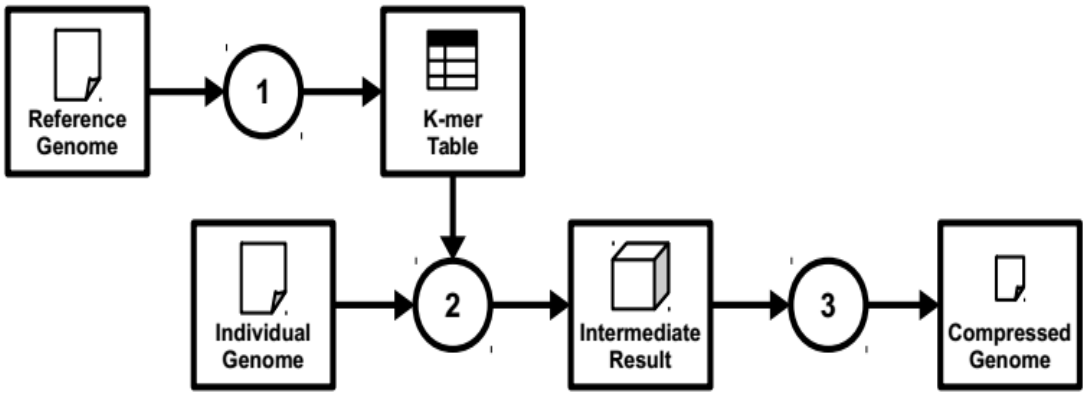
Nếu sau các bước trên mà vẫn không tìm thấy chuỗi khớp nào thì ghi lại một cặp bazơ của đầu vào và lặp lại quá trình.

Tệp đầu vào được xử lý tìm kiếm chuỗi khớp trong tham chiếu cho đến khi toàn bộ đầu vào đã được so sánh.

Mã hóa Huffman. Sử dụng mã hóa Huffman [45] khi ghi lại chuỗi khớp để tăng nén bằng cách giảm kích thước của những phần ghi vào đĩa.

2.1.2. Thư viện FRESCO

FRESCO là một thư viện nén tham chiếu không mất dữ liệu cho các tệp gen được sắp xếp, lưu ở định dạng RAW hoặc FASTA. Được viết bằng C++ và là mã nguồn mở [25]. FRESCO gồm những biến phát sinh sau: *single nucleotide polymorphisms* (SNPs), phép chèn, phép xóa và phép thay thế. Nó thực hiện ba bước bên trong (hình 2.1 [21]): (1) đánh chỉ số; (2) nén chính nó và (3) mã hóa.



Hình 2.1. Mô hình các bước thực hiện của FRESCO.
(1) Đánh chỉ số gen tham chiếu. (2) Nén gen đầu vào, sử dụng tham chiếu đã đánh chỉ số. (3) Mã hóa các kết quả sơ bộ để tạo ra tệp cuối cùng

- **FRESCO – Mã nguồn mở**

FRESCO – Framework for REferential Sequence Compresion, là tên một mã nguồn mở. Phần mềm có tại https://github.com/hubsw/FRESCO.git. FRESCO viết bằng C++, sử dụng thư viện BOOST, CST và libz. FRESCO được thiết kế theo một module để dễ thay thế các phần của thuật toán nén, ví dụ cấu trúc chỉ số với các thực nghiệm khác nhau. Những lựa chọn thiết kế chính khi thực hiện một thuật toán nén tham chiếu là 1) định dạng đầu vào, 2) cấu trúc chỉ

số cho tham chiếu, 3) thuật toán nén, ví dụ: tham ăn (greedy) và 4) định dạng thứ tự chuỗi cho các tệp nén, nghĩa là mã hóa thực tế các chuỗi khớp. FRESCO gồm giao diện cho mỗi phần trong 4 phần trên và cho phép sử dụng những thực hiện thay thế khác nhau và thêm vào các thuật toán mới chuyên dụng.

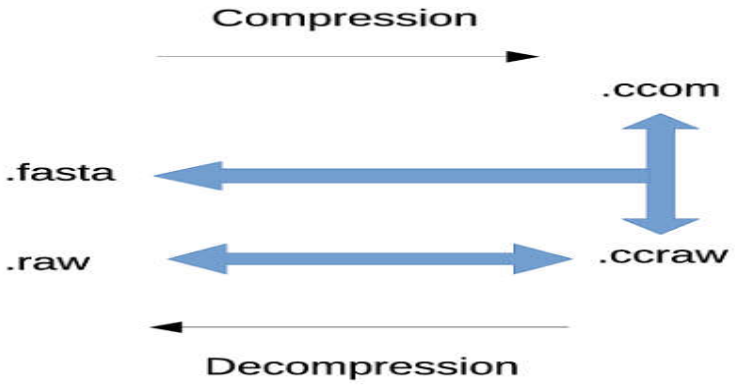
2.1.3. Bảng K-mer

Việc nén dữ liệu cần một cấu trúc đánh chỉ số. Đầu tiên, JDNA sử dụng đối tượng cổ điển (như HashMap) để đánh chỉ số. Tuy nhiên, bộ nhớ sử dụng cho các đối tượng phức tạp là rất lớn. Khai báo một *HashMap* 50000 ngăn và cung cấp cấu trúc tương ứng làm cho việc sử dụng bộ nhớ tăng quá 20GB.

Một bảng băm gọi là bảng K-mer được sử dụng để lưu thông tin liên quan tới thuật toán nén. Bảng K-mer giống với một cấu trúc *MultiValueMap,* lưu trữ nhiều giá trị trên mỗi khóa thay vì chỉ một giá trị. Bảng K-mer là một bảng *table* ma trận số nguyên và một mảng đếm *integer* số nguyên (mục đích sẽ được giải thích sau). Trong Java, một ma trận số nguyên là một mảng sơ cấp của nhiều mảng thứ cấp số nguyên. Mỗi vị trí trong mảng sơ cấp bắt đầu với giá trị *null.* Sau đó, dựa theo một *ArrayList* trong mỗi vị trí mảng sơ cấp để chèn giá trị vào *table* bằng cách tạo ra một mảng kích thước cố định và sử dụng biến đếm *counter* để điều khiển số phần tử ở mỗi vị trí sơ cấp. Mảng *counter* được sử dụng để truy cập các giá trị được lưu trữ và để biết khi nào mở rộng mảng.

2.1.4. Định dạng tệp

JDNA chấp nhận hai loại tệp cho đầu vào nén: RAW hoặc FASTA. Nếu tệp là RAW thì một tệp CRAW (cho RAW nén) được tạo ra; nếu tệp là FASTA thì một CRAW và một CCOM được tạo ra. Tệp CCOM có chú thích trong tệp FASTA và số dòng của những chú thích này. Với giải nén, một tệp CRAW được gán như đầu vào. JDNA sẽ tìm kiếm một tệp CCOM cùng tên với tệp CRAW. Nếu tìm thấy CCOM thì FASTA được tạo ra, nếu không thì RAW được tạo. Thông tin này được tóm tắt trong hình 2.4.



Hình 2.4. Định dạng file đầu vào, đầu ra của JDNA

2.2. Đánh giá

Ở phần đánh giá này, người viết trình bày kết quả nén đạt được về tỉ lệ nén, thời gian nén và vùng nhớ để so sánh hiệu suất của thuật toán tham chiếu FRESCO với các thuật toán cùng loại khác như GDC và RLZ. Đồng thời so sánh hiệu suất của JDNA với Fresco để thấy được những cải tiến của JDNA đã thực sự mang lại hiệu quả về tỉ lệ nén và dung lượng lưu trữ.

2.2.1. Cải thiện tỉ lệ nén

Chuỗi tham chiếu là nhân tố chính xác định tỉ lệ nén, cho một mã hóa đầu vào khớp tham chiếu cố định.

Ngay cả khi trong cùng một loài thì chuỗi tham chiếu cũng có ảnh hưởng quan trọng tới tỉ lệ nén. Với việc làm tăng sự giống nhau giữa tham chiếu và chuỗi nén thì đầu vào khớp tham chiếu dài hơn có thể được tìm thấy và tỉ lệ nén sẽ được tăng lên.

- **Lựa chọn một tham chiếu tốt**

Đầu tiên, nói về sự lựa chọn một chuỗi tham chiếu tốt nhất cho một chuỗi nén đơn.

Có thể tồn tại hơn một tham chiếu tốt nhất, trong trường hợp đó một tham chiếu sẽ được chọn ngẫu nhiên. Theo thực nghiệm thì trường hợp này không bao giờ xảy ra.

Một phương thức đơn giản để tìm chuỗi tham chiếu tốt nhất là nén tất cả các chuỗi dựa trên tất cả chuỗi tham chiếu có thể và chọn ra tham chiếu mà cho ra số đầu vào khớp tham chiếu ít nhất, gọi là Rsbest. Nếu các chuỗi dài như trong trường hợp đưa ra thì sẽ tốn khá nhiều thời gian để tính toán nén tham chiếu $n * m$, trong đó m là số chuỗi tham chiếu ứng cử và n là số chuỗi nén. Nếu muốn nén 1000 chuỗi mà chọn chuỗi tham chiếu tốt nhất theo phương thức này thì sẽ tốn vài tuần; tuy nhiên, ta sẽ sử dụng phương thức này trên một mẫu để đánh giá các phương pháp được mô tả tiếp theo.

- **Viết lại tham chiếu**

Một phương thức khác là viết lại chuỗi tham chiếu theo cách mà nó biểu diễn một đường đi (chuỗi hành động) hầu như giống nhau qua tất cả các chuỗi trong tập hợp chuỗi nén. Trong phương thức này, số các chuỗi tham chiếu ứng cử được cố định là một. Viết lại chuỗi có động lực sinh học: các SNP khác nhau thường xảy ra với tần suất khác nhau. Bằng việc viết lại tham chiếu để xác định và gắn các SNP thường xuyên nhất tới tham chiếu.

Hình 2.6 mô tả thuật toán viết lại tham chiếu [21].

Algorithm 4 Reference Rewriting Algorithm

Input: set of referential compressions $S = \{rc_1, ..., rc_n\}$, a reference string ref, and a threshold t
Output: rewritten reference $result$

Let $result$ be an empty string
for $1 \leq p \leq |ref|$ **do**
  if there exists a most frequent rewrite candidate (X, p, c) for p in S, with $freq((X, p, c), S) \geq t$ **then**
    if X=REPL **then**
      Append c to $result$
    else if X=INS **then**
      Append c to $result$
      Append $ref(p)$ to $result$
    else if X=DEL **then**
      do nothing
    end if
  else
    Append $ref(p)$ to $result$
  end if
end for

Hình 2.6. Thuật toán viết lại tham chiếu

Phép viết lại cho mỗi vị trí có tần suất cao nhất trong tham chiếu được sử dụng để viết lại chuỗi tham chiếu. Đầu vào của thuật toán là một tập nén tham chiếu *S*, một chuỗi tham chiếu sẽ-được-viết-lại *ref* và một giới hạn *t.* Giới hạn được sử dụng chỉ để chọn ra phép viết lại mà có ít nhất một tần suất tương đối trong *S.* Theo thực nghiệm thì việc lựa chọn chuỗi tham chiếu ban đầu chỉ có một tác động nhỏ tới tỉ lệ nén. Hơn nữa, tính toán lại mỗi nén tham chiếu đối với chuỗi tham chiếu được viết lại đã được thực nghiệm. Việc cập nhật nén tham chiếu để phản ánh những thay đổi trong tham chiếu viết lại mà không cần phải nén lại sẽ là một hướng đáng quan tâm trong tương lai.

(1) <u>So sánh FRESCO với hai thuật toán cùng loại GDC và RLZ</u>

So sánh sự thực hiện của các thuật toán nén tham chiếu với FRESCO. Hai đối thủ của FRESCO là GDC [19] và RLZ [24]. Có thể thấy RLZ là ngang hàng trong nén tham chiếu, trong khi GDC là chương trình tốt nhất khi so về tốc độ nén và tỉ lệ nén.

Kết quả nén 10 chuỗi được chỉ ra ở Hình 2.7.

	Compressed size (in MB)			Runtime (in s)			Compression factor			Compression speed (MB/s)		
Dataset	GDC	RLZ	FRESCO	GDC	RLZ	FRESCO	GDC	RLZ	FRESCO	GDC	RLZ	FRESCO
H-1	**3.7**	15.5	4.2	495.2	224.0	**20.0**	**680.0**	160.8	590.6	5.0	11.1	**124.3**
H-2	**3.9**	15.9	4.5	454.9	199.4	**19.4**	**625.5**	152.9	542.8	5.3	12.2	**125.5**
H-3	**3.3**	13.4	3.8	314.6	165.5	**14.9**	**593.6**	147.5	513.9	6.3	11.9	**132.4**
H-4	**3.5**	13.8	4.1	247.0	159.4	**15.0**	**543.8**	138.4	466.1	7.7	12.0	**127.1**
H-5	**3.0**	12.0	3.4	243.4	144.0	**13.9**	**608.2**	150.6	526.3	7.4	12.6	**130.2**
H-6	**3.0**	11.9	3.6	248.0	143.8	**15.3**	**566.1**	143.7	475.1	6.9	11.9	**112.0**
H-7	**2.7**	10.7	3.1	403.1	121.1	**12.8**	**591.2**	148.7	508.8	3.9	13.1	**124.7**
H-8	**2.5**	10.1	2.9	171.8	122.9	**11.6**	**577.5**	144.8	500.5	8.5	11.9	**126.3**
H-9	**2.0**	8.4	2.3	130.0	102.2	**11.0**	**714.3**	168.0	618.2	10.9	13.8	**128.8**
H-10	**2.4**	9.4	2.7	183.6	109.8	**10.9**	**572.2**	144.1	493.4	7.4	12.3	**124.7**
H-11	**2.5**	9.6	2.8	153.6	118.3	**11.0**	**548.3**	140.5	474.3	8.8	11.4	**122.2**
H-12	**2.3**	8.9	2.6	199.2	113.5	**10.0**	**593.0**	150.4	514.1	6.7	11.8	**133.5**
H-13	**1.9**	7.5	2.2	65.5	90.9	**9.2**	**602.5**	153.4	532.2	17.6	12.7	**124.5**
H-14	**1.6**	6.4	1.8	68.5	77.0	**8.6**	**664.7**	167.6	591.1	15.7	13.9	**124.2**
H-15	**1.4**	5.9	1.6	72.2	70.7	**8.1**	**710.1**	173.7	636.9	14.2	14.5	**126.9**
H-16	**1.4**	5.4	1.6	103.1	68.9	**6.9**	**638.5**	167.1	552.5	8.8	13.1	**131.4**
H-17	**1.3**	5.1	1.5	140.3	68.9	**6.5**	**635.3**	159.1	552.8	5.8	11.8	**125.4**
H-18	**1.4**	4.8	1.6	44.6	66.7	**6.6**	**565.2**	162.5	487.0	17.5	11.7	**118.3**
H-19	**1.1**	4.0	1.3	116.8	50.8	**5.3**	**546.7**	147.8	468.0	5.1	11.6	**111.1**
H-20	**1.0**	4.0	1.2	43.8	49.5	**4.5**	**623.7**	157.4	542.5	14.4	12.7	**139.3**
H-21	**0.7**	2.8	0.9	12.3	33.3	**3.5**	**684.3**	171.8	553.0	39.1	14.5	**138.2**
H-22	**0.6**	2.7	0.7	19.3	32.0	**3.7**	**816.9**	189.7	735.9	26.5	16.0	**137.1**
H-X	**1.7**	7.7	2.0	168.2	96.3	**12.1**	**903.6**	201.6	789.0	9.2	16.1	**128.0**
AT-1	**2.0**	6.5	2.3	8.3	41.3	**2.5**	**154.2**	105.3	133.2	36.7	7.4	**123.1**
AT-2	**1.4**	4.5	1.7	4.2	25.4	**1.4**	**145.0**	98.5	119.0	46.9	7.8	**136.8**
AT-3	**1.7**	5.5	2.0	5.5	32.1	**1.6**	**139.8**	96.0	117.2	42.7	7.3	**145.1**
AT-4	**1.3**	4.3	1.6	3.7	24.4	**1.5**	**139.5**	97.2	116.7	50.2	7.6	**126.5**
AT-5	**1.9**	6.1	2.2	6.3	37.5	**1.9**	**144.6**	99.5	121.3	42.8	7.2	**141.2**
Y-WG	**1.0**	86.8	1.4	2.8	47.6	**1.0**	**127.3**	1.4	89.0	44.5	2.6	**124.7**
AVG	**2.0**	10.7	2.3	142.4	90.9	**8.6**	**532.9**	142.8	460.7	18.0	11.5	**128.0**

Hình 2.7. Thống kê nén 10 chuỗi ngẫu nhiên dựa trên một tham chiếu cố định (kết quả tốt nhất được bôi đậm)

GDC đạt được kết quả nén tốt nhất cho mỗi tập dữ liệu dùng để đánh giá (trung bình 2.0MB cho 10 chuỗi). FRESCO đạt được hiệu quả nén tốt thứ hai (trung bình 2.3MB cho 10 chuỗi). RLZ đạt hệ số nén thấp đối với Y-WG dường như là do kỹ thuật tối ưu hạn chế trong nó.

FRESCO có thời gian nén ngắn nhất (trung bình 8.6 giây cho 10 chuỗi), trong khi RLZ chậm hơn khoảng 10 lần và GDC chậm hơn khoảng 16 lần.

Lưu ý là tốc độ đọc cao nhất của đĩa cứng ở thực nghiệm là khoảng 145 MB/s. Nén với FRESCO dường như có giới hạn vào/ra: thực hiện các thực nghiệm bổ sung với các chuỗi trong bộ nhớ chính.

Tóm lại, GDC luôn đạt được kết quả nén tốt nhất, trong khi FRESCO thì đạt tốc độ nén nhanh hơn RLZ và GDC. Hình 2.8 tóm tắt kết quả của nén tham chiếu FRESCO so với GDC và RLZ.

	GDC		RLZ		FRESCO		FRESCO (reference selection)		FRESCO (reference rewriting)		FRESCO (second-order compression)	
	CF	C.Speed	CF	C.Speed	CF	C.Speed	CF	C.Speed	CF	C.Speed	CF	C.Speed
H-*	635.0	11.2	158.4	12.8	550.7	126.8	594.7	81.2	742.4	90.6	3,057.2	58.4
AT-*	144.6	43.9	99.3	7.5	121.5	134.5	126.9	56.3	123.6	60.6	407.7	53.7
Y-WG	127.3	44.5	1.4	2.6	89.0	124.7	89.0	21.1	91.9	21.5	712.8	41.4
AVERAGE	302.3	33.2	86.4	7.6	253.7	128.7	270.2	52.8	319.3	57.5	1,392.6	51.1

Hình 2.8. Tóm tắt kết quả nén đạt được từ FRESCO, GDC và RLZ (CF: hệ số nén, C.speed: tốc độ nén MB/s)

(2) So sánh hiệu quả của JDNA và Fresco

Tỉ lệ nén hầu hết là xác định giữa JDNA và FRESCO. JDNA sử dụng một phiên bản mã hóa Huffman chỉnh sửa và Gzip, trong khi FRESCO chỉ sử dụng Gzip để nén mỗi chuỗi khớp.

Bảng 2.1. Bảng so sánh JDNA/FRESCO

Chr	Original Size (MB)	Compression JDNA (KB)	Compression FRESCO (KB)	Index Time JDNA (s)	Index Time FRESCO (s)	Indexes K	%
1	238	306	306	0.02	62.56	40.48	1.6
2	232	319	318	0.01	57.6	33.55	1.4
3	189	269	406	0.01	47.02	34.99	1.8
4	182	296	299	0.03	45.5	63.79	3.3
5	173	234	235	0.02	42.58	38	2.1
6	163	243	249	0.05	40.03	150.23	8.8
7	152	215	217	0.02	37.2	41.96	2.6
8	140	202	202	0.01	33.89	26.87	1.8
9	135	161	162	0.03	28.97	94.12	6.7
10	129	192	194	0.01	30.94	15.5	1.1
11	129	204	204	0.02	30.92	30.89	2.3
12	128	186	186	0.02	30.65	34.81	2.6
13	110	157	159	0.01	23.3	15.55	1.4
14	102	126	128	0.01	21.44	19.96	1.9
15	98	115	115	0.01	19.91	10.13	1
16	86	113	112	0.01	18.63	15.91	1.8
17	77	100	100	0.01	18.01	16.13	2
18	74	114	115	0.01	17.56	7.74	1
19	56	79	80	0.01	12.81	26.87	4.5
20	60	79	79	0.01	14.06	22.16	3.5
21	46	58	61	0.01	8.93	25.11	5.2
22	49	49	50	0.01	8.89	24.28	4.7
X	148	109	109	0.01	35.93	18.23	1.2

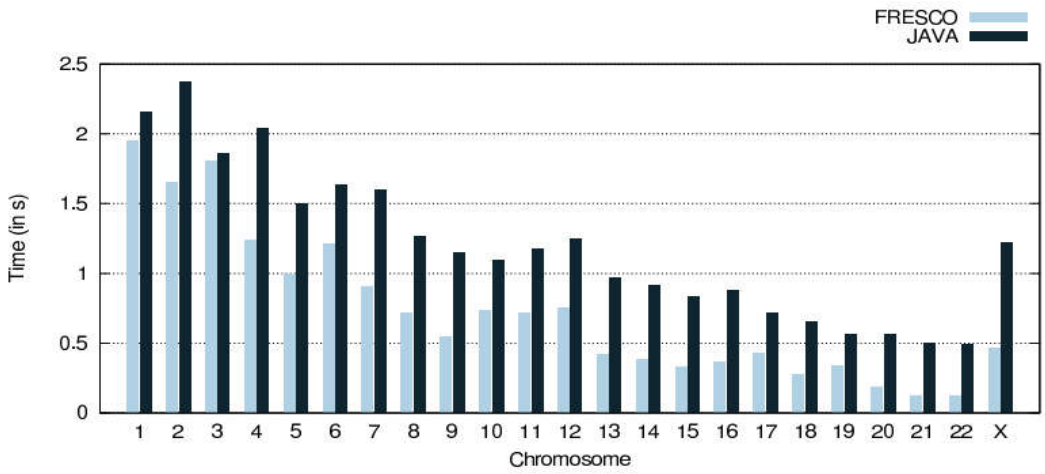
2.1.1. Cải thiện thời gian

Thực nghiệm ở trên đã chứng minh Fresco hiệu quả hơn các thuật toán cùng loại và cũng đã cho thấy sự vượt trội của JDNA so với Fresco. Ở phần tiếp theo, người viết sẽ chỉ trình bày so sánh hiệu quả về thời gian và vùng nhớ của JDNA với Fresco.

Thực nghiệm này so sánh hiệu quả về thời gian của cả hai công cụ cho việc nén và giải nén hệ gen người. Việc đánh giá thời gian được chia thành 4 phần:

- Nén đầy đủ
- Đánh chỉ số thời gian
- Thời gian nén
- Thời gian giải nén

Thời gian nén đầy đủ đo được sử dụng dòng lệnh *time*, kết quả có thể thấy ở hình 2.10.



Hình 2.10. Thời gian nén

Thời gian mỗi thư viện dùng để đánh chỉ số gen tham chiếu được đo trong quá trình thực hiện chương trình, kết quả có thể thấy ở bảng 2.1. JDNA hầu như không tốn thời gian đánh chỉ số, đặc biệt là so với thời gian đánh chỉ số luôn lớn hơn ở FRESCO.

2.1.2. Cải thiện vùng nhớ

Vùng nhớ nén. Một công cụ ngoài được sử dụng để đo việc sử dụng bộ nhớ lớn nhất của hai phương pháp. Hình 2.17 cho thấy việc sử dụng bộ nhớ của JDNA và FRESCO. JDNA thực hiện cơ chế tái sử dụng đối tượng và giảm việc tạo ra đối tượng. Tuy nhiên, JDNA và FRESCO sử dụng vùng nhớ tương tự nhau, ngay cả sau khi đã nỗ lực giảm sử dụng vùng nhớ đáng kể. Việc sử dụng bộ nhớ trong JDNA phụ thuộc bảng K-mer. Mặc dù JDNA đã giảm đánh chỉ số và bảng K-mer chỉ là một ma trận số nguyên, do mỗi dòng ma trận là một đối tượng mới nên bộ nhớ sử dụng vẫn lớn so với FRESCO, phương thức mà đánh chỉ số toàn bộ tham chiếu.

Vùng nhớ giải nén. Giải nén sử dụng một lượng vùng nhớ cố định cho tham chiếu, kết quả trong một hằng số sử dụng vùng.

Điểm tương đồng giữa gen tham chiếu và gen đầu vào sẽ quyết định kết quả của FRESCO và JDNA, trong đó sự tương đồng càng lớn thì tỉ lệ nén càng cao. Các kết quả được trình bày là những giá trị trung bình. Nén toàn bộ một hệ gen người cho kết quả trong một tệp kích thước từ 4 tới 10MB.

Kết quả chỉ ra ở phần đánh giá chứng minh rằng thuật toán đánh chỉ số theo yêu cầu có thể được sử dụng để xây dựng một công cụ có thể so sánh với các công cụ khác mà đánh chỉ số tham chiếu hoàn toàn. Các kết quả có tính cạnh tranh cho những thuộc tính được kiểm thử và cho thấy sự cải thiện về tổng thời

gian thực hiện và tỉ lệ nén. JDNA đã kế thừa và những cải tiến cho thấy thuật toán đã đạt được hiệu quả khả quan trong việc nén chuỗi gen và cả hệ gen.

Thuật toán nén tham chiếu dù chỉ mới phát triển gần đây và được biết đến như một loại thuật toán thứ tư cho nén chuỗi đa lượng nhưng đã cho thấy hiệu quả vượt trội hơn hẳn so với ba loại thuật toán nén được biết đến trước đó là (1) thuật toán nén mã hóa bit, (2) thuật toán nén dựa trên bộ từ điển và (3) thuật toán nén xác suất thống kê. Trong luận văn này, người viết thực hiện thực nghiệm bổ sung so sánh JDNA với thuật toán thuộc phương thức xác suất thống kê Huffman và thuật toán nén dựa trên bộ từ điển Lempel-Ziv để làm rõ hơn tính ưu việt của thuật toán nén tham chiếu như đã nhận định. Chi tiết thực nghiệm so sánh sẽ được trình bày ở chương 3 của luận văn.

CHƯƠNG 3 – THỰC NGHIỆM SO SÁNH THUẬT TOÁN JDNA VỚI THUẬT TOÁN MÃ HÓA HUFFMAN VÀ LEMPEL - ZIV

Ở chương này, người viết trình bày thực nghiệm bổ sung để minh họa thêm về tính hiệu quả của thuật toán nén tham chiếu đối với nén chuỗi gen DNA mà tiêu biểu là thuật toán JDNA so với hai thuật toán thuộc loại khác là Lempel-Ziv, thuật toán nén dựa trên từ điển và Huffman, thuật toán nén dựa trên xác suất thống kê.

3.1. Môi trường thực nghiệm

Tất cả thực nghiệm được thực hiện trên máy tính cá nhân Dell Latitude E6420 với cấu hình như sau:

- CPU: Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz / L2 cache
- Bộ nhớ: 6GB RAM (1x4GB, 1x2GB)/ DIMM
- Dung lượng: 250GB/ SCSI/ Disk drives WDC WD2500BEKT-75PVMT0

Phần mềm sử dụng: Các chương trình được chạy trên nền Linux kernel (64-bit). JDNA mã nguồn mở được viết và chỉnh sửa bằng ngôn ngữ Java sử dụng Oracle Java 7 JVM (build 1.7.0 40-b43). Huffman và Lempel Ziv (LZW) được viết và chỉnh sửa bằng ngôn ngữ C++.

Các tập dữ liệu thực nghiệm: Người viết thực hiện so sánh ba thuật toán nén trên ba tập dữ liệu sinh học: một tập hợp gen người, một tập hợp gen từ cây Arabidopsis thaliana và một tập hợp gen khuẩn men. Dữ liệu trong tệp gen nén có dạng chuỗi.

3.2. Thực nghiệm so sánh JDNA với Mã hóa Huffman và Lempel – Ziv

So sánh sự thực hiện của các thuật toán mã hóa Huffman và Lempel-Ziv với nén tham chiếu JDNA. Kết quả cho thấy tốc độ nén của Huffman khá tốt, trong khi JDNA đạt được hiệu quả vượt trội về hệ số nén và kích thước tệp nén.

So sánh đầu tiên như sau: với mỗi loài và mỗi nhiễm sắc thể, lựa chọn ngẫu nhiên một số chuỗi và áp dụng mỗi thuật toán lựa chọn cho các chuỗi ngẫu nhiên đó. Kết quả được thống kê và so sánh về kích thước gen sau khi nén, thời gian nén và hệ số nén của từng thuật toán cho một hoặc nhiều chuỗi gen cụ thể.

Các chương trình thuật toán được chạy trên máy ảo Linux bằng các dòng lệnh tương ứng.

Trong nhiều trường hợp, việc nén dữ liệu thành công không đồng nghĩa với việc giải nén cũng thành công và đạt hiệu quả tốt như mong đợi. Vì lý do này mà trong khuôn khổ thực nghiệm so sánh bổ sung, người viết cũng đã chỉnh sửa

chương trình và thực hiện giải nén các chuỗi gen đã được nén. Ở phần nén thuật toán JDNA đã đạt được hiệu quả tốt hơn các thuật toán thuộc loại khác, kết quả sẽ được phân tích ở phần 3.3 dưới đây nên khi thực hiện so sánh hiệu quả giải nén, người viết sẽ chỉ thống kê kết quả về thời gian thực hiện để chứng minh tính ưu việt về thời gian giải nén của thuật toán JDNA so với hai thuật toán được lựa chọn để so sánh. Việc giải nén cũng được thực hiện bằng các dòng lệnh tương ứng chạy trên nền Linux.

3.3. Phân tích và đánh giá kết quả thực nghiệm

Bộ dữ liệu được tải về khá lớn, tổng cộng gần 100GB nhưng do môi trường thực nghiệm có hạn nên người viết chỉ lựa chọn ra một số chuỗi với dung lượng phù hợp để thực hiện quá trình nén và so sánh. Hình 3.7 thể hiện bảng thống kê kết quả đạt được khi nén các tập dữ liệu sử dụng thuật toán nén Huffman, Lempel-Ziv và JDNA.

	Compressed size (in MB)			Runtime (in s)			Compression factor			Compression speed (in MB/s)		
Dataset	Huffman	LZW	JDNA	Huffman	LZW	JDNA	Huffman	LZW	JDNA	Huffman	LZW	JDNA
hs_alt_CHM1_1.1_chr1.fa	79.62	61.90	**24.12**	8	118	**10518**	3.1166	4.0088	**10.2880**	9.9525	0.5246	**0.0023**
hs_ref_GRCh38.p2_chr21.fa	14.62	13.02	**11.11**	1	18	**59**	3.1646	3.5535	**4.1644**	14.6200	0.7233	**0.1883**
hs_ref_GRCh38.p2_chr22.fa	16.22	72.80	**10.75**	2	26	**68**	3.0216	0.6732	**4.5591**	8.1100	2.7999	**0.1581**
hs_ref_GRCh38.p2_chrY.fa	14.58	8.32	**7.36**	2	24	**178**	3.8879	6.8131	**7.7018**	7.2900	0.3467	**0.0413**
hs_alt_CHM1_1.1_chr22.mfa	19.69	60.80	**5.57**	2	29	**2935**	2.5796	0.8352	**9.1167**	9.8425	2.0966	**0.0019**
hs_ref_GRCh38.p2_chr21.mfa	19.00	14.45	**6.03**	2	18	**3524**	2.4357	3.2019	**7.6779**	9.4975	0.8028	**0.0017**
hs_ref_GRCh38.p2_chrY.mfa	17.58	9.03	**3.60**	2	24	**2913**	3.2242	6.2802	**15.7415**	8.7905	0.3761	**0.0012**
hs_alt_CHM1_1.1_chrX.mfa	62.16	45.75	**10.26**	6	77	**12783**	2.4729	3.3598	**14.9888**	10.3595	0.5942	**0.0008**
hs_alt_CHM1_1.1_chr15.mfa	41.41	28.97	**12.82**	5	72	**5027**	2.4492	3.5001	**7.9122**	8.2810	0.4024	**0.0025**
hs_alt_CHM1_1.1_chr16.mfa	37.83	23.72	**12.42**	3	63	**6290**	2.4027	3.8320	**7.3173**	12.6100	0.3765	**0.0020**
hs_ref_GRCh38.p2_chr19.mfa	22.83	16.97	**7.54**	2	40	**6752**	2.5432	3.4214	**7.7056**	11.4150	0.4243	**0.0011**
AT_Aedal-1	41.30	25.90	**7.33**	12	55	**10805**	1.7167	2.7375	**9.6829**	3.4417	0.4709	**0.0007**
AT_Aedal-3	33.20	21.00	**5.90**	9	49	**6742**	1.7078	2.7000	**9.6178**	3.6889	0.4286	**0.0009**
AT_Ale-Stenar-44-4	19.50	16.80	**4.40**	7	31	**5835**	2.1744	2.5238	**9.6455**	2.7857	0.5419	**0.0008**
AT_Algutsrum	27.90	15.90	**5.14**	8	37	**6790**	1.7814	3.1258	**9.6654**	3.4875	0.4297	**0.0008**
AT_App1-12	25.40	15.80	**4.57**	7	32	**10825**	1.6850	2.7089	**9.3681**	3.6286	0.4938	**0.0004**
AT_Baa1-2	22.90	15.30	**4.09**	6	29	**6850**	1.6419	2.4575	**9.1988**	3.8167	0.5276	**0.0006**
AT_Bil-5	20.00	13.70	**3.60**	5	32	**5750**	1.6300	2.3796	**9.0608**	4.0000	0.4281	**0.0006**
AT_Broet1-6	23.10	15.70	**4.08**	6	50	**6850**	1.6190	2.3822	**9.1871**	3.8500	0.3140	**0.0006**
AT_Doer-10	19.90	12.90	**3.52**	5	46	**4792**	1.7136	2.6434	**9.6958**	3.9800	0.2804	**0.0007**
AT_Dra2-1	30.10	19.20	**5.35**	9	88	**8515**	1.6711	2.6198	**9.4208**	3.3444	0.2182	**0.0006**
AT_Eden-1	22.10	14.50	**3.86**	6	22	**6850**	1.6878	2.5724	**9.6843**	3.6833	0.6591	**0.0006**
AT_Eds-1	21.50	13.50	**3.78**	6	20	**6665**	1.7023	2.7111	**9.6699**	3.5833	0.6750	**0.0006**
AT_Faeb-2	20.30	13.90	**3.67**	6	19	**6164**	1.6256	2.3741	**9.0106**	3.3833	0.7316	**0.0006**
AT_Gro-3	29.30	17.90	**5.24**	9	30	**7360**	1.7611	2.8827	**9.8633**	3.2556	0.5967	**0.0007**
Yst-1	32.60	20.60	**5.76**	10	32	**10200**	1.7117	2.7087	**9.8000**	3.2600	0.6438	**0.0006**
Y-WG4	0.44	0.42	**0.22**	1	1	**5**	3.5320	3.6746	**7.1620**	0.4380	0.4210	**0.0432**
AVERAGE	**27.22**	**22.55**	**6.74**	**5.44**	**40.07**	**6001.67**	**2.25**	**3.06**	**9.14**	**6.09**	**0.64**	**0.02**

Hình 3.7. Thống kê kết quả nén của các thuật toán Huffman, Lempel-Ziv và JDNA.

JDNA đạt hiệu quả về kích thước tệp nén (trung bình 6.74MB cho 27 tập dữ liệu) và tỉ lệ nén khoảng 9:1 cho các tập dữ liệu thực nghiệm. Lempel-Ziv đạt hiệu quả nén tốt thứ hai (trung bình 22.55MB cho 27 tập dữ liệu) và hệ số nén trung bình là 3.06. Huffman đạt được tốc độ nén khá tốt (trung bình 5.44 giây) nhưng lại chưa hiệu quả về kích thước tệp nén, hệ số nén cũng như không gian lưu trữ.

JDNA không chỉ là thuật toán hiệu quả về tỉ lệ nén và tối ưu dung lượng lưu trữ mà còn rất hiệu quả về thời gian khi thực hiện giải nén. Hình 3.9 dưới đây thể hiện so sánh thời gian giải nén của JDNA so với Huffman và LZW. Kết quả cho thấy thời gian giải nén của JDNA nhanh hơn hai thuật toán Huffman và LZW một bậc.

	Compressed size (in MB)			Decompresion Runtime (in s)		
Dataset	Huffman	LZW	**JDNA**	Huffman	LZW	**JDNA**
hs_alt_CHM1_1.1_chr1.fa	79.62	61.90	**24.12**	14	28	**2**
hs_ref_GRCh38.p2_chr21.fa	14.62	13.02	**11.11**	2	9	**2**
hs_ref_GRCh38.p2_chr22.fa	16.22	72.80	**10.75**	2	38	**2**
hs_ref_GRCh38.p2_chrY.fa	14.58	8.32	**7.36**	6	5	**1**
hs_alt_CHM1_1.1_chr22.mfa	19.69	60.80	**5.57**	4	33	**0.001**
hs_ref_GRCh38.p2_chr21.mfa	19.00	14.45	**6.03**	3	10	**1**
hs_ref_GRCh38.p2_chrY.mfa	17.58	9.03	**3.60**	9	6	**1**
hs_alt_CHM1_1.1_chrX.mfa	62.16	45.75	**10.26**	5	21	**0.001**
hs_alt_CHM1_1.1_chr15.mfa	41.41	28.97	**12.82**	5	41	**0.001**
hs_alt_CHM1_1.1_chr16.mfa	37.83	23.72	**12.42**	6	12	**2**
hs_ref_GRCh38.p2_chr19.mfa	22.83	16.97	**7.54**	2	8	**1**
AT_Aedal-1	41.30	25.90	**7.33**	3	14	**3**
AT_Aedal-3	33.20	21.00	**5.90**	2	11	**2**
AT_Ale-Stenar-44-4	19.50	16.80	**4.40**	3	11	**0.001**
AT_Algutsrum	27.90	15.90	**5.14**	3	10	**2**
AT_App1-12	25.40	15.80	**4.57**	2	11	**0.001**
AT_Baa1-2	22.90	15.30	**4.09**	2	9	**2**
AT_Bil-5	20.00	13.70	**3.60**	2	11	**0.001**
AT_Broet1-6	23.10	15.70	**4.08**	4	14	**3**
AT_Doer-10	19.90	12.90	**3.52**	3	12	**3**
AT_Dra2-1	30.10	19.20	**5.35**	4	14	**3**
AT_Eden-1	22.10	14.50	**3.86**	6	8	**1**
AT_Eds-1	21.50	13.50	**3.78**	6	20	**2**
AT_Faeb-2	20.30	13.90	**3.67**	6	19	**0.001**
AT_Gro-3	29.30	17.90	**5.24**	3	12	**3**
Yst-1	32.60	20.60	**5.76**	3	14	**3**
Y-WG4	0.44	0.42	**0.22**	5	12	**0.001**
AVERAGE	**27.22**	**22.55**	**6.74**	**4.26**	**15.30**	**1.44**

Hình 3.9. Thống kê kết quả giải nén của các thuật toán Huffman, Lempel-Ziv và JDNA.

Thuật toán nén tham chiếu JDNA tuy phải dùng tới thời gian nén lâu hơn do phải xử lý các phần bù trong chuỗi gen (những thành phần không phải A, T, G, C) và cả những phần chưa tương đồng trong chuỗi gen nhưng đã cho thấy hiệu quả về thời gian khi giải nén là tốt hơn rất nhiều so với hai thuật toán Lempel-Ziv và Huffman, trung bình chỉ mất 1.44 giây cho giải nén các tập dữ liệu thực nghiệm đã nén.

Thực nghiệm đã chỉ ra thuật toán nén tham chiếu JDNA không chỉ đạt được hiệu quả về tỉ lệ nén cao, kích thước tệp gen nén giảm rõ rệt, tiết kiệm dung lượng lưu trữ mà còn đạt được sự ưu việt về thời gian giải nén đáng mong đợi.

KẾT LUẬN

Mặc dù những thách thức đối với lưu trữ thông tin chuỗi DNA cho tới nay đã có thể kiểm soát phần nào nhưng việc cải tiến trong sắp xếp chuỗi đa lượng và phương pháp nén tốt hơn cho chuỗi DNA vẫn là một vấn đề quan trọng đối với cộng đồng sinh học, nhất là những tiềm năng trong việc kiểm soát việc mất thông tin trong hoặc sau quá trình nén/giải nén chuỗi gen.

Sắp xếp chuỗi đa lượng (HTS) tạo nên một cuộc cách mạng trong nghiên cứu sinh học phân tử [44]. Công nghệ cung cấp những phương thức nén hiệu quả cho tập dữ liệu DNA khổng lồ. Thêm vào đó là những thách thức trong việc hiểu cấu trúc, chức năng và tiến hóa của hệ gen, những phương pháp sắp xếp chuỗi đa lượng cũng đặt ra câu hỏi và tập trung vào việc biểu diễn, lưu trữ, truyền tải, truy vấn và bảo vệ thông tin chuỗi gen.

Trong luận văn này, người viết đã trình bày các phương thức và thuật toán nén tiêu biểu cho mỗi phương thức nén dữ liệu chuỗi DNA. Trong đó, người viết chọn phương thức nén tham chiếu và thuật toán nén tiêu biểu JDNA làm mục tiêu nghiên cứu chính vì những hiệu quả mà thuật toán này mang lại cho nén DNA như tiết kiệm không gian lưu trữ, tỉ lệ nén đạt được cao hơn các thuật toán nén loại khác một bậc. JDNA được phát triển dựa trên thuật toán được sử dụng bởi FRESCO [25]. Thuật toán đã đạt được hiệu quả trong việc tăng tỉ lệ nén chuỗi đa lượng bằng 3 phương pháp kế thừa: (1) lựa chọn tham chiếu, (2) viết lại tham chiếu và (3) nén thứ tự hai. Tỉ lệ nén có thể đạt 400:1 hoặc cao hơn với những kế thừa ở điều kiện lý tưởng về chuỗi tham chiếu lựa chọn phù hợp hay chuỗi gen cùng loài có độ tương đồng cao. Bên cạnh những đặc trưng kế thừa từ thuật toán nén tham chiếu Fresco, JDNA còn thực sự hiệu quả khi sử dụng phương pháp đánh chỉ số theo yêu cầu để tiết kiệm thời gian nén thực và tăng tỉ lệ nén đáng kể. Đóng góp chính của JDNA là sử dụng phương thức đánh chỉ số theo yêu cầu. Cơ chế này kết hợp được hai đặc tính tốt nhất đó là: một cấu trúc chỉ số khá đơn giản xử lý những khác biệt chính giữa các tệp gen và nén nhanh các chuỗi khớp trực tiếp.

Đạt được những ưu việt về tỉ lệ nén, thời gian giải nén và không gian lưu trữ. Đồng thời xử lý và nén được nhiều định dạng tệp gen. Nhưng JDNA lại gặp bất lợi về thời gian nén do phải xử lý những chuỗi gen có sự tương đồng chưa cao, gồm nhiều những kí tự khác các bazơ đặc trưng (A, T, G, C) và chỉ đạt được tỉ lệ nén cao với các chuỗi DNA đã được sắp xếp. JDNA cũng bị hạn chế hiệu suất bởi JVM, trong đó việc quản lý bộ nhớ phức tạp của JVM cũng làm tăng độ khó khăn trong việc tạo ra một ứng dụng bộ nhớ hiệu quả. Để nén toàn

bộ hệ gen, hiệu suất JDNA có thể được tăng nhờ cơ chế song song. JDNA nén các tệp lớn theo các khối độc lập mà được nén riêng biệt. Cơ chế song song có thể làm tăng việc sử dụng vùng nhớ nhưng sẽ giảm được thời gian nén đáng kể. Tuy gặp một số bất lợi về thời gian nén và dung lượng máy ảo JVM do sử dụng ngôn ngữ Java làm công cụ phát triển nhưng JDNA đã chứng minh được tính hiệu quả trong việc nén chuỗi gen của thuật toán nén tham chiếu. Trong tương lai JDNA có thể được tiếp tục cải tiến để đạt được tốc độ nén và hiệu suất lưu trữ đáng mong đợi.

Cùng với những nghiên cứu và nhận định đã trình bày, người viết cũng đã thực hiện thực nghiệm so sánh thuật toán tham chiếu JDNA với hai thuật toán nén thuộc phương thức khác là nén dựa trên bộ từ điển Lempel-Ziv và thuật toán nén xác suất thống kê Huffman để bổ sung cho kết quả nghiên cứu đạt được. Kết quả thực nghiệm tuy chưa đạt được tỉ lệ nén hay thời gian mong đợi cao nhất của thuật toán nén tham chiếu do một số hạn chế về môi trường thực nghiệm, nhưng đã bước đầu khẳng định được sự tối ưu của thuật toán nén tham chiếu mà tiêu biểu là JDNA cho nén chuỗi gen. Những kết quả thực nghiệm này sẽ là tiền đề để người viết tiếp tục những nghiên cứu và cải tiến cho việc nén chuỗi gen trong tương lai.

TÀI LIỆU THAM KHẢO

[1] Samantha Woodward BIOC 218. *A Critical Analysis of DNA Data Compression Methods, 2011.*

[2] Sebastian Wandelt, Marc Bux, and Ulf Leser. *Trends in Genome Compression, 2013.*

[3] P. Raja Rajeswari, Allam Apparo, and V. K. Kumar. *Genbit compress tool(gbc): A javabased tool to compress dna sequences and compute compression ratio(bits/base) of genomes.* CoRR, abs/1006.1193, 2010

[4] Rajendra Kumar Bharti, Archana Verma, and R.K. Singh. *A biological sequence compression based on cross chromosomal similarities using variable length lut.* International Journal of Biometrics and Bioinformatics, 4:217 – 223, 2011.

[5] Ateet Mehta and Bankim Patel. *Dna compression using hash based data structure.* International Journal of Information Technology & Knowledge Management, 3:383 – 386, 2010.

[7] Pothuraju Rajarajeswari, Allam Apparao. *DNABIT Compress – Genome compression agorithm*, Journal on Bioinformation, Volume 5, Issue 8, January 2011.

[13] B. G. Chern, I. Ochoa, A. Manolakos, A. No, K. Venkat and T. Weissman,Department of Electrical Engineering, Stanford University, Stanford CA 94305. *Reference Based Genome Compression.*

[14] Suman M. Choudhary, Anjali S. Patel, Sonal J. Parmar. *Study of LZ77 and LZ78 Data Compression Techniques,* International Journal of Engineering Science and Innovative Technology (IJESIT), Volume 4, Issue 3, May 2015.

[15] M. D. Cao, T. Dix, L. Allison, and C. Mears. *A simple statistical algorithm for biological sequence compression*. In Data Compression Conference, 2007. DCC '07, pages 43 –52, march 2007.

[16] P.Raja Rajeswari, Dr. Allam Apparao, Dr. R.Kiran Kumar. *Huffbit Compress – Algorithm To Compress Dna Sequences Using Extended Binary Trees,* Journal of Theoretical & Applied Information Technology, Vol. 13 Issue 1/2, pages 101-106, 2010.

[19] S. Deorowicz and S. Grabowski. *Robust relative compression of genomes with random access*. Bioinformatics, 27(21):2979–2986, 2011.

[20] C. Wang and D. Zhang. *A novel compression tool for efficient storage of genome resequencing data*. Nucleic Acids Research, 39(7):e45, Apr. 2011.

[21] Jim Dowling, KTH. *Reference Based Compression Algorithm*, Scalable, Secure Storage of Biobank Data, Work Package 2, pages 23 – 44, June 2014.

[24] S. Kuruppu, S. J. Puglisi, and J. Zobel. *Optimized relative Lempel-Ziv compression of genomes*. In Proceedings of the Thirty-Fourth Australasian Computer Science Conference - Volume 113, ACSC '11, pages 91–98, Darlinghurst, Australia, Australia, 2011.

[25] S.Wandelt and U.Leser. *Fresco: Referential compression of highly similar sequences*. Computational Biology and Bioinformatics, IEEE/ACM Transactions on, 10(5):1275–1288, Sept 2013.

[30] Shanika Kuruppu, Simon J. Puglisi, and Justin Zobel. *Relative lempel-ziv compression of genomes for large-scale storage and retrieval*. In Proceedings of the 17th International Conference on String Processing and Information Retrieval, SPIRE'10, pages 201 – 206, 2010.

[31] A. J. Pinho, D. Pratas, and S. P. Garcia. *GReEn: a tool for efficient compression of genome resequencing data*. Nucleic Acids Research, December 2011.

[32] Marty C. Brandon, Douglas C. Wallace, and Pierre Baldi. *Data structures and compression algorithms for genomic sequence data*. Bioinformatics, 25(14):1731 – 1738, 2009.

[38] Stéphane Grumbach and Fariza Tahi. *A new challenge for compression algorithms: genetic sequences.* Information Processing & Management, 30(6):875 – 886, 1994.

[39] Jesper Larsson and Alistair Mofat. *Offline dictionary-based compression*. In Proceedings of the 1999 Conference on Data Compression, DCC'99, pages 296 – 305, 1999.

[40] John G. Cleary, Ian, and Ian H. Witten. *Data compression using adaptive coding and partial string matching.* IEEE Transactions on Communications, 32:396 – 402, 1984.

[44] M. L. Metzker. *Sequencing technologies — the next generation*, Nat. Rev. Genet., 11(1):31–46, January 2010.

[45] M. R. Wick. *An object-oriented refactoring of Huffman encoding using the Java Collections Framework.* SIGCSE Bull., 35(1):283–287, January 2003.