Tạp chí Khoa học Trường Đại học Cần Thơ
website: sj.ctu.edu.vn

GIẢI THUẬT SỬA LỖI CHO CÁC TRÌNH TỰ ADN NGẮN

Nguyễn Văn Hòa[1] và Nguyễn Văn Đông[1]
[1] *Khoa Công nghệ Kỹ thuật Môi trường, Trường Đại học An Giang*

Thông tin chung:
Ngày nhận: 03/09/2013
Ngày chấp nhận: 21/10/2013

Title:
Algorithm of short-read error correction

Từ khóa:
chuỗi ADN, xác lập trình tự ADN, chuỗi ADN ngắn, chỉ mục, sửa lỗi

Keywords:
DNA sequence, DNA sequencing, short read, kmer, error correction

ABSTRACT

Today with the development of DNA sequencing technology, we have obtained a large amount of DNA sequences in a short time with low cost. Specially, the next-generation DNA sequecing can generate a huge amount of short DNA sequences, called short reads with length from 30 to 100 bp. The short reads have an error rate between 1% and 2%. Therefore, the error reads must be corrected before being assembled into the complete genome. There are several proposed algorithms for correcting the error reads such as SHREC and SOAP de Novo. However, SHREC needs a long computation time to correct errors while SOAP de Novo requires very high memory usage. In this paper, we present our algorithm (RCorrector) based on the index structure of KMER for detecting and correcting error reads. Compared to the SHREC algorithm, the RCorrector algorithm provides a speed up from 3 to 7 with the same sensitivity and specificity.

TÓM TẮT

Ngày nay với sự tiến bộ của kỹ thuật xác lập trình tự ADN (DNA Sequencing) chúng ta có thể tạo ra một số lượng lớn các chuỗi ADN trong khoảng thời gian ngắn với chi phí thấp. Đặc biệt thế hệ xác lập trình tự mới hiện nay tạo ra số lượng rất lớn chuỗi ADN ngắn, được gọi là short read, với chiều dài từ 30 đến 100 nulcotide. Các read này có tỉ lệ lỗi từ 1% đến 2%. Do đó các read lỗi này phải được sửa lỗi trước khi được lắp ráp thành bộ gien ADN hoàn chỉnh. Nhiều giải thuật sửa lỗi đã được đề xuất như SHREC, SOAP de Novo. Nhưng những giải thuật này vẫn còn những hạn chế như cần dung lượng bộ nhớ lớn hoặc thời gian sửa lỗi khá nhiều. Trong bài báo này chúng tôi đề xuất giải thuật hiệu chỉnh lỗi, được đặt tên là RCorrector, dựa trên cấu trúc chỉ mục kmer nhằm phát hiện lỗi và sửa lỗi trực tiếp trên các read. So sánh với giải thuật SHREC trên 8 tập dữ liệu, RCorrector đạt được hiệu suất sửa lỗi thông qua hai đặc trưng specificity và sensitivity là tương đương với SHREC nhưng nhanh hơn SHREC từ 3 đến 7 lần.

1 GIỚI THIỆU

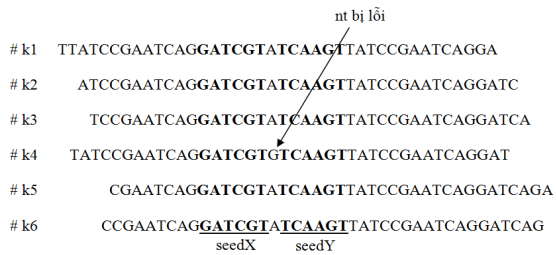
Xác lập trình tự ADN cho một gien của loài nào đó là một trong những công việc quan trọng trong lĩnh vực Sinh Tin học (Bioinformatics). Kỹ thuật xác lập trình tự thủ công đầu tiên được giới thiệu vào năm 1977 [6]. Với những cải tiến cho đến đầu những năm 1990, kỹ thuật xác lập trình tự có thể tạo ra các chuỗi ADN với độ dài từ 500bp đến 1000bp với độ chính xác 99%. Những năm gần đây kỹ thuật xác lập trình tự ADN đang trải qua một cuộc cách mạng. Đặc biệt là từ năm 2008 đến nay

với kỹ thuật xác lập trình tự song song lớn (massive), còn được gọi là NGS (Next Generation Sequencing) [4,7], cho phép thu được số lượng rất lớn dữ liệu trình tự ADN với tốc độ nhanh hơn và giá thành rẻ hơn so với các phương pháp xác lập trình tự trước đó. Chẳng hạn như công nghệ SOLEXA có thể tạo ra trên 10 Gbp trình tự ADN ngắn (short read 30-70 bp) trong khoảng thời gian 4 ngày với chi phí khoảng 60 đô la cho mỗi Mbp [7]. Với chi phí thấp và thời gian thực hiện nhanh nên ngày càng có nhiều dự án xác lập gien mới. Chẳng hạn phương pháp chuẩn đoán bệnh dựa theo dữ liệu cá nhân đã được đề nghị bởi Guy và *ctv* [1].

Sau khi xác lập được số lượng lớn các short read, gọi tắc là read, thì công việc tiếp theo là sắp dãy (assembly) các read này để thu được bộ gien hoàn chỉnh. Nhưng trong quá trình xác lập trình tự ADN của các kỹ thuật thế hệ mới này khả năng sinh lỗi cho từng nucleotide (nt) cao hơn. Tỷ lệ lỗi khoảng từ 1% đến 2% trên chiều dài của read [8]. Do số lượng các read sinh ra là rất lớn nên số lượng lỗi cũng rất nhiều. Các nucleotide lỗi phải được sửa lỗi để phục vụ cho việc sắp dãy lại thành một bộ gien hoàn chỉnh. Do đó cần phải có giải pháp sửa lỗi ở giai đoạn này nhằm chỉnh sửa các read bị lỗi. Các phương pháp sửa lỗi được đưa ra chủ yếu theo hai hướng sau đây. Hướng thứ nhất thực hiện sửa lỗi bằng cách xây dựng cây hậu tố (suffix tree) cho tất cả read. Phương pháp này được đề nghị bởi Schröder và *ctv* với tên gọi là giải thuật SHREC [8]. Để sửa lỗi thì trước hết SHREC sẽ xây dựng cây hậu tố cho tất cả các read. Tiếp theo là quá trình duyệt và phân tích các nút trên cây có hai nút con. Khi đó một nút con được xem là lỗi nếu trọng số (độ bao phủ) của nút đó nhỏ hơn trọng số mong đợi. Hướng thứ hai được đề nghị bởi Li và *ctv* với tên giải thuật là SOAP de novo [3]. Phương pháp này sử dụng đồ thị de Bruijn (de Bruijn graph) để sắp dãy các read thành các contig, các trình tự ADN có chiều dài lớn hơn các read. Các read được kiểm tra lỗi để sửa lỗi trước khi tiến hành sắp dãy.

Trong bài viết, chúng tôi đề xuất giải thuật sửa các nucleotide lỗi trong các trình tự ADN ngắn thu được từ kỹ thuật xác lập trình tự thế hệ mới. Giải thuật của chúng tôi sửa một nucleotide lỗi thành nucleotide đúng theo nguyên tắc biểu quyết theo số đông. Phương pháp sửa lỗi này hoạt động với sự hỗ trợ của cấu trúc chỉ mục kmer [10].

Phần tiếp theo của bài báo được tổ chức như sau: trong phần 2, chúng tôi sẽ trình bày ý tưởng chính của giải thuật sửa lỗi ở các chuỗi ADN ngắn do chúng tôi đề xuất. Quá trình thực nghiệm sẽ được trình bày trong phần 3 trước khi kết luận và hướng phát triển được trình bày trong phần 4.



Hình 1: Sửa lỗi dựa trên độ bao phủ của read

2 GIẢI THUẬT HIỆU CHỈNH LỖI K-MER

Trong phần này chúng tôi trình bày giải thuật sửa lỗi dựa trên kỹ thuật lập chỉ mục k-mer, được đặt tên là giải thuật RCorrector.

2.1 Nguyên tắc sửa lỗi

Giải thuật RCorrector thực hiện việc sửa lỗi một nucleotide nào đó của read dựa trên thông tin về độ bao phủ (coverage) khi xác lập các read từ một gien nào đó. Giả sử một gien có chiều dài là n và một tập R gồm m read $\{R_1,\ldots,R_m\}$ được xác lập từ một gien đó. Các read có chiều dài l. Độ bao phủ c được xác định qua công thức như sau: $c=(lm)/n$. Như vậy, nếu một gien có chiều dài n=10kbp và tập read R có 7000 read với độ dài của mỗi read l=50 thì độ bao phủ được tính như sau $c=(50x7000)/10000=35$. Giải thuật RCorrector chỉ có thể sửa lỗi tại một nucleotide nào đó của read khi độ bao phủ tại vị trí đó lớn hơn 2. Nguyên tắc sửa lỗi của giải thuật RCorrector dựa vào thông tin về độ bao phủ c của chuỗi con có độ dài bằng k, được gọi là k-mer [10] nằm trong c read như Hình 1. Trong Hình 1, chuỗi con có độ dài bằng 6 (6-mer) như là hai chuỗi seedX, seedY nằm trong 6 read tương ứng độ bao phủ $c=6$. Dựa vào sự giống nhau của các seedX và seedY trong 6 read này chúng tôi tiến hành kiểm tra các nucleotide (nt) nằm giữa hai seedY và seedY trong 6 read này. Nếu có sự khác biệt giữa các nt nằm giữa hai seed này thì chúng tôi tiến hành chỉnh sửa cho đúng. Chẳng hạn trong Hình 1 thì nucleotide của read thứ 4 (#k4) là G trong khi 5 nucleotide còn lại đều là A nên chúng tôi tiến hành sửa nt G thành A. Đây là nguyên tắc sửa lỗi biểu quyết theo số đông.

2.2 Giải thuật Rcorrector

Giải thuật RCorrector bao gồm ba giai đoạn chính: (1) lập chỉ mục cho tất cả các read của tập dữ liệu read vào; (2) sửa lỗi cho từng chỉ mục của INDEX theo nguyên tắc biểu quyết số đông; (3) loại bỏ các read không thể sửa lỗi.

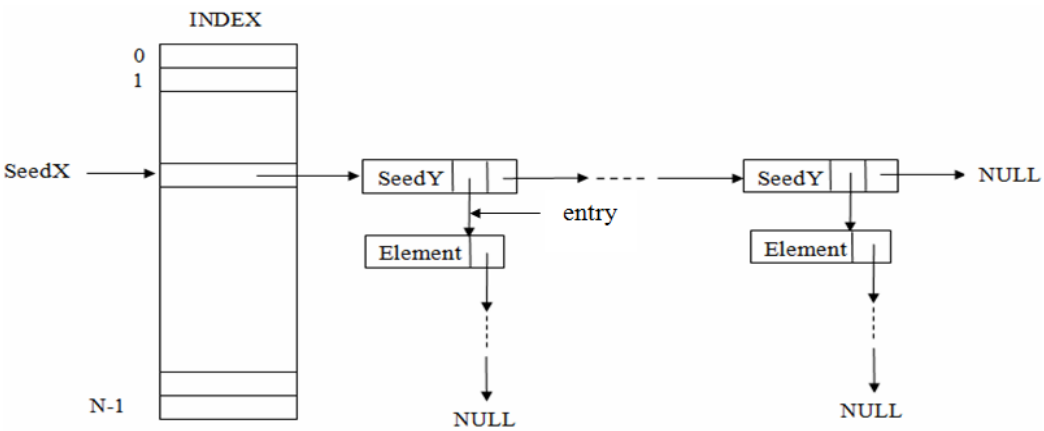
Giải thuật RCorrector

Đầu vào: tập tin chứa các read vào
Đầu ra: tập tin chứa các read ra
1: lập chỉ mục INDEX
2: **do** // {vòng lặp sửa lỗi}
3: **for each** mục của INDEX **do**
4: LR = danh sách read bị lỗi
5: **for each** phần tử read lỗi trong LR **do**
6: xóa các cặp chỉ mục trong INDEX
7: sửa lỗi nucleotide của read
8: chèn các cặp seed mới vào INDEX
9: **end for**
10: **end for**
11: **while** hết lỗi hoặc số lỗi không đổi
12: loại bỏ các read không thể sửa lỗi
13: xuất ra tập tin kết quả

2.2.1 *Lập chỉ mục INDEX*

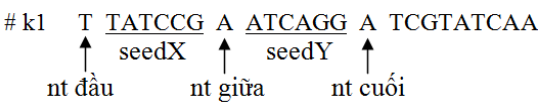
Để có thể xác định được các seedX và seedY giống nhau với độ bao phủ *c* chúng tôi sử dụng cấu trúc chỉ mục INDEX như Hình 2.

Cấu trúc chỉ mục này là một mảng được cấp phát động gồm *N* phần tử, với *N* được xác định dựa vào chiều dài của seedX. Gọi *s* là chiều dài của seedX thì số phần tử của cấu trúc INDEX là $N = 4^s$ bởi vì mỗi nucleotide của seedX ðược mã hóa bằng 2 bít nhị phân như sau *A=00, C=01, G=10* và *T=11*. Mỗi phần tử của mảng là một con trỏ, trỏ đến các node chứa seedY. Mỗi node của seedY có hai con trỏ: một tương ứng danh sách theo chiều ngang (Hnext) trỏ đến các seedY khác nhau, con trỏ còn lại tương ứng với danh sách theo chiều dọc (Vnext) chứa danh sách các phần tử có cùng giá trị seedY. Mỗi danh sách tương ứng với một seedY theo chiều dọc được gọi là một entry. Mỗi phần tử element của entry lưu thông tin gồm năm thành phần như sau: {#, pos, c_1, c_2, c_3}. Trong đó #: số thứ tự của read trong tập dữ liệu; pos: vị trí của nucleotide giữa seedX và seedY của read; c1, c2, c3 tương ứng lần lượt là ba nt kề đầu seedX, nt kề giữa seedX và seedY và nt kề cuối của seedY như Hình 3.



Hình 2: Cấu trúc chỉ mục INDEX

Quá trình lưu bộ element gồm 5 thành phần của một cặp {seedX, seedY} của read trong tập dữ liệu đầu vào được thực hiện như sau: đầu tiên cần xác định vị trí của seedX trong cấu trúc INDEX. Tiếp theo từ vị trí này RCorrector tiến hành xác định vị trí theo chiều ngang (Hnext) để kiểm tra sự tồn tại của chuỗi seedY này hay chưa. Kết quả tìm kiếm theo chiều ngang sẽ có hai khả năng:



Hình 3: Cấu trúc các thành phần của 1 element.

– Nếu seedY đã tồn tại, bộ element sẽ được chèn vào danh sách liên kết tương ứng theo chiều dọc (entry) của seedY này. Danh sách được trỏ bởi Vnext và được sắp xếp theo thứ tự tăng dần của chỉ số read nhằm giúp cho quá trình tìm kiếm khi xóa và lập chỉ mục mới hiệu quả hơn.

– Nếu không tìm thấy, thực hiện chèn thêm một nút mới ở đầu danh sách theo chiều ngang và gán giá trị seedY mới. Đồng thời chèn phần tử element vào nút mới này bởi con trỏ theo chiều dọc (Vnext).

Như vậy với một read có chiều dài *l* và kích thước của seed (X và Y) là *s* thì sẽ có tổng cộng $l-(2\times s+2)$ bộ element được đưa vào cấu trúc chỉ mục INDEX.

2.2.2 Sửa lỗi ở các read

Sau khi lập chỉ mục INDEX toàn bộ các read trong tập dữ liệu đầu vào, RCorrector tiến hành duyệt qua từng mục trong cấu trúc INDEX để tiến hành sửa lỗi. Tại một chỉ mục nào đó tương ứng với seedX, dữ liệu trong mục này là danh sách ngành với nhiều entry. Mỗi entry tương ứng với một seedY được biểu diễn bởi con trỏ Hnext. Dữ liệu trong mỗi entry được minh họa như Bảng 1. Cột đầu tiên là số thứ tự (#) của read được trong tập dữ liệu vào và được xếp theo thứ tự tăng dần; cột thứ hai là vị trí (pos) của nt nằm giữa seedX và seedY; bộ ba kí tự ở cột thứ ba tương ứng là ba nt kề đầu seedX, kề giữa seedX và seedY, kề cuối của seedY.

Bảng 1: Lỗi trong entry có thể được được sửa

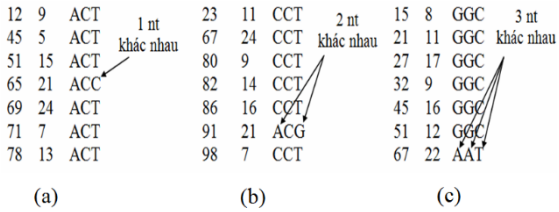
Số thứ tự (#k)	Vị trí nt giữa seedX và seedY (pos)	Ba kí tự kề trước, giữa và cuối của {seedX, seedY}
12	9	ACT
45	5	ACT
51	15	ACC ← nt bị lỗi
65	21	ACT
69	24	ACT
71	7	ACT
78	13	ACT

Cột bộ ba ký tự kề trước, kề giữa, kề cuối của seedX và seedY trong mỗi entry chính là cơ sở để giải thuật RCorrector tiến hành sửa lỗi theo nguyên tắc biểu quyết theo số đông. Chẳng hạn trong Bảng 1, cột này chỉ ra rằng read có số thứ tự 51 ở hàng thứ 3 có nt kề cuối của seedY là "C" được xem là bị lỗi do khác biệt với "T" của các bộ ba còn lại. Do đó, nt "C" sẽ được sửa lại thành "T" theo nt đúng của entry. Trong entry này các bộ ba ký tự đầu, giữa và cuối đồng nhất với nhau và chỉ có một ký tự "C" của dòng thứ ba là khác biệt nên dễ dàng được sửa lại thành "T".

Bảng 2: Lỗi trong entry không thể sửa được

Số thứ tự (#k)	Vị trí nt giữa seedX và seedY (pos)	Ba kí tự kề trước, giữa và cuối của {seedX, seedY}
23	11	GGT
67	24	GGT
80	9	GGT
82	14	GCT ← không thể sửa lỗi
86	16	CCT
91	21	CCT
98	7	CCT

Nhưng các bộ ba ký tự của một entry khác như ở Bảng 2 thì có 2 nhóm đại diện (GGT và CCT). Trong khi đó read có số thứ tự là 82 ở hàng thứ 4 có bộ 3 nt là "GCT". Nếu so sánh "GCT" với nhóm thứ nhất là "GGT" thì sẽ sửa nt kề giữa "C" thành "G". Còn nếu so sánh "GCT" với nhóm thứ hai là "CCT" thì sẽ sửa nt kề trước "G" thành "C". Vì vậy, RCorrector không thể xác định được read (số thứ tự 82) thuộc nhóm nào. Trong trường hợp này giải thuật sẽ không thực hiện sửa lỗi bởi vì xác suất sửa lỗi sai sẽ rất cao. Do đó, RCorrector phải kiểm tra sự đồng nhất của các bộ ba nt, được gọi là 3 nt đúng, trong entry với yêu cầu chỉ có một bộ 3 ký tự có sự khác biệt với các bộ ba ký tự còn lại như ở Hình 4. Hình 4 này minh họa ba khả năng xảy ra: (a) chỉ có một nt sai so với 3 nt đúng của entry; (b) hai nt sai so với 3 nt đúng của entry; (c) cả ba nt đều sai so với 3 nt đúng của entry. RCorrector chỉ tiến hành sửa lỗi cho trường hợp đầu tiên.



Hình 4: Các trường hợp khác biệt với bộ nt đúng

Khi tiến hành sửa lỗi cho một element của entry thì lỗi này có thể là xảy ra ở nt kề trước seedX, hoặc nt kề giữa seedX và seedY hoặc nt kề cuối với seedY. Trước khi tiến hành sửa lỗi một nt, RCorrector tiến hành quá trình xóa các bộ element của read được sửa lỗi và cập nhập lại các bộ element cho read này phụ thuộc vào vị trí của nt được sửa lỗi như sau:

– Nếu lỗi tại ký tự kề trước của seedX thì quá trình xóa các bộ element và lập các bộ element mới chỉ thực hiện cho các cặp {seedX, seedY} liên quan đến nt bị lỗi. Tổng số bộ element phải sửa phụ thuộc vào vị trí của nt lỗi trong read. Gọi e là vị trí của nt lỗi, s là kích thước của seed. Nếu $e \geq (2s+3)$ tổng số bộ element phải sửa là $(2s+3)$ ngược lại thì số bộ element phải sửa là $e+1$.

– Nếu sửa lỗi tại ký tự kề giữa seedX và seedY thì quá trình xóa các bộ element và lập các bộ element mới cho các cặp seedX và seedY phía trước và phía sau có liên quan đến nt bị lỗi. Tùy theo vị trí xảy ra lỗi trong read mà số bộ element có liên quan từ $s+3$ đến $2s+3$.

– Nếu sửa lỗi tại ký tự kề cuối của seedY thì số bộ element phải chỉnh sửa sẽ là một nếu $e=l-1$, trong đó l là chiều dài của read, ngược lại thì số bộ element được cập nhật nhiều nhất là $2s+3$.

2.2.3 Loại bỏ các read không thể sửa lỗi

Sau mỗi vòng lặp, số lỗi được phát hiện sẽ giảm dần và có thể sau một số vòng lặp thì giải thuật RCorrector sẽ không phát hiện thêm lỗi nào nữa. Khi đó giải thuật RCorrector sẽ kết thúc và không có read nào được loại bỏ khỏi tập dữ liệu đầu ra. Nhưng cũng có thể xảy ra trường hợp sau một số vòng lặp số lỗi mà giải thuật RCorrector phát hiện bị lặp lại. Điều này là vì một số nt của read có thể bị sửa nhầm thành nt sai. Do đó, sau khi cập nhập lại với nt mới thì các cặp {seedX, seedY} vừa được đưa vào bị trùng vào vị trí khác trên gien nhưng vị trí này không phải là vị trí đúng của read. Ở vòng lặp tiếp theo, nt được sửa theo entry này nhưng trong vòng lặp sau nt đó được sửa lại thành nt khác theo entry khác. Cho nên quá trình này bị lặp đi lặp lại gây ra số lỗi do giải thuật phát hiện cũng sẽ bị lặp lại.

Khi số lỗi do giải thuật phát hiện sau một số vòng bị lặp lại thì tất cả các read liên quan đến các lỗi này đều không thể được sửa lỗi. Số read này sẽ được loại bỏ khỏi tập dữ đầu ra.

3 KẾT QUẢ THỰC NGHIỆM

3.1 Các tiêu chí đánh giá giải thuật

Để đánh giá giải thuật RCorrector chúng tôi chọn các tiêu chí để đánh giá về thời gian thực thi của giải thuật và hai đặc trưng thể hiện khả năng sửa lỗi và độ tin cây là: Sensitivity và Specificity. Hai đặc trưng này được tính như sau:

Sensitivity = TP / (TP+FN)

Specificity = TN / (TN+FP)

Trong đó:

– True Positive (TP): read có nt lỗi và nt này được sửa lại thành đúng bởi giải thuật.

– False Positive (FP): read có nt đúng nhưng được sửa lại thành nt sai bởi giải thuật.

– True Negative (TN): read có nt đúng và vẫn được giữ nguyên nt đúng.

– False Negative (FN): read có nt lỗi và vẫn còn giữ nguyên nt lỗi sau khi thực hiện giải thuật.

Đặc trưng Sensitivity thể hiện khả năng sửa lỗi của giải thuật. Giải thuật sửa lỗi càng hiệu quả (Sensitivity càng gần bằng 100%) khi số read vẫn còn bị lỗi (FN) nhỏ hơn nhiều so với số read bị lỗi mà đã được sửa đúng (TP). Trong khi đó, đặc trưng Specificity thể hiện khả năng này thể hiện độ tin cậy của giải thuật sửa lỗi.

3.2 Các tập dữ liệu

Giải thuật RCorrector được đánh giá dựa trên các tập dữ liệu giả lập (simulated datasets). Các tập dữ liệu giả lập được xây dựng từ 4 tập dữ liệu được tải từ trang web NCBI: Saccharomyces cerevisiae chromosome 5 (S.cer5), chromosome 7 (S.cer7), gien vi khuẩn Haemophilus influenzae (H.inf) và Staphylococcus Aureus (S.aureus). Kích thước của 4 tập dữ liệu này từ 0.58 Mbp tới 2.8 Mbp. Với mỗi tập dữ liệu đã chọn, chúng tôi tạo ra hai tập dữ liệu read giả lập với tỷ lệ lỗi tương ứng là 1% và 2%. Thông tin chi tiết các tập dữ liệu read được trình bày ở Bảng 3.

Bảng 3: Các tập dữ liệu dùng để chạy thực nghiệm

ID	Reference genome (Genbank ID)	Genome length	Error - rate	Coverage	Read length	Number of reads
A1	S.cer 5 (NC_001137.3)	0.58M	1%	35	35	0.58M
A2			2%			
B1	S.cer 7 (NC_001139.9)	1.1M	1%			1.1M
B2			2%			
C1	H.inf (NC_007146.2)	1.9M	1%			1.9M
C2			2%			
D1	S.aureus (NC_003923.1)	2.8M	1%			2.8M
D2			2%			

3.3 Kết quả thực nghiệm

Để đánh giả giải thuật RCorrector, chúng tôi tiến hành chạy thực nghiêm giải thuật RCorrector và giải thuật SHREC với 8 tập dữ liệu ở Bảng 3 trên máy PC sử dụng CPU intel Genuine 2.13GHz với bộ nhớ RAM 3GB chạy trên hệ điều hành Linux Ubuntu 10. Giải thuật RCorrector được cài đặt bằng ngôn ngữ lập trình C. Quá trình đánh giá gồm có hai phần: phần thứ nhất tiến hành đánh giá giải thuật RCorrector với nhiều kích thước seed khác nhau. Từ đó chọn kích thước của seed tối ưu nhất để làm giá trị mặc định cho giải thuật RCorrector. Phần thứ hai tiến hành đánh giá so sánh giữa giải thuật RCorrector và giải thuật SHREC về thời gian chạy cũng như hai đặc trưng sửa lỗi Sensitivity và Specificity.

3.3.1 Kết quả đánh giá giải thuật Rcorrector với các chiều dài seed khác nhau

Để kiểm tra khả năng sửa lỗi của giải thuật RCorrector, chúng tôi tiến hành thực nghiệm nhằm đánh giá giải thuật với nhiều kích thước của seed khác nhau (từ 6 nt đến 10 nt) trên tập dữ liệu C1. Với chiều dài seed nhỏ hơn 6 thì khả năng xảy ra lặp lại (repeat) của các trình tự của seedX và seedY tại nhiều vị trí trên gien sẽ làm cho giải thuật sửa lỗi kém hiệu quả về mặt thời gian vì kích thước lưu trữ trong từng chỉ mục của INDEX rất lớn. Ngược lại, đối với chiều dài seed lớn hơn 10 kích thước lưu trữ trong từng chỉ mục INDEX rất ít nên khả năng phát hiện lỗi sẽ rất thấp. Vì vậy, chiều dài seed nhỏ hơn 6 hoặc lớn hơn 10 không được chúng tôi dùng trong thực nghiệm để đánh giá. Kết quả của giải thuật RCorrector với độ dài seed từ 6 đến 10 được thể hiện ở Bảng 4.

Bảng 4: Thời gian chạy (tính bằng giây) và hai đặc trưng sửa lỗi

Chiều dài seed	6	7	8	9	10
Thời gian	7385	2331	676	267	158
Sensitivity	76.23%	95.02%	95.39%	92.24%	87.66%
Specificity	99.67%	99.89%	99.93%	99.95%	99.96%

Bảng 4 chỉ ra rằng khi tăng chiều dài seed thì thời gian thực hiện của giải thuật giảm dần. Điều này được giải thích như sau: đối với một read, khi tăng chiều dài seed số lượng cặp {seedX, seedY} được tạo ra ít hơn cho nên thời gian xử lý lỗi sẽ nhanh hơn. Đặc trưng sửa lỗi Specificity của giải thuật RCorrector thể hiện giá trị Specificity cao (đạt trên 99.6%) đối với tất cả các seed. Đặc trưng Sensitivity đạt giá trị tốt nhất với chiều dài seed bằng 8. Khi chiều dài seed tăng dần từ 8 đến 10 thì chỉ số Sensitivity giảm xuống. Điều này là do khi chiều dài seed càng lớn thì số cặp {seedX, seedY} trong chỉ mục của INDEX càng ít. Do đó khả năng phát hiện và sửa lỗi của giải thuật Rcorrector sẽ giảm.

Tóm lại, với chiều dài seed bằng 8 là lựa chọn tối ưu cho giải thuật RCorrector bởi vì thời gian thực sửa khá nhanh (676s) nhưng đặc trưng sửa lỗi Sensitivity đạt giá trị tốt nhất (95.39%). Vì vậy, kích thước của seed bằng 8 được chọn làm giá trị mặc định cho giải thuật RCorrector.

3.3.2 So sánh giải thuật RCorrector với giải thuật SHREC

Tiếp theo chúng tôi trình bày kết quả đánh giá giải thuật RCorrector bằng cách so sánh với giải thuật SHREC về thời gian thực hiện và hai đặc trưng Sensitivity, Specificity. Cả hai giải thuật SHREC và RCorrector được tiến hành chạy thực nghiệm trên 8 tập dữ liệu với tham số mặc định.

a. Về thời gian chạy chương trình

Thời gian thực hiện của hai giải thuật được thể hiện ở Bảng 5. So với SHREC thì RCorrector chạy nhanh hơn từ 3 từ 7 lần.

Xét trên một tập dữ liệu, ví dụ S.cer 5, thời gian thực hiện của RCorrector phụ thuộc vào tỷ lệ lỗi ở các read. Nếu tỷ lệ lỗi càng cao thì RCorrector cần nhiều thời gian chỉnh sửa lỗi. Khi tỷ lệ lỗi tăng thì tổng số lỗi mà giải thuật phát hiện cũng sẽ tăng lên, do đó cần nhiều thời gian hơn để sửa lỗi. Thao tác sửa lỗi của SHREC là duyệt cây và trong lúc duyệt nếu phát hiện lỗi thì tiến hành sửa lỗi. Sau khi sửa lỗi SHREC phải cập nhật lại cây.

Bảng 5: Thời gian chạy (giây) của SHREC và Rcorrector trên 8 tập dữ liệu

Tập dữ liệu	A1	A2	B1	B2	C1	C2	D1	D2
SHREC	354	463	1046	1102	4734	4948	8544	10800
RCorrector	75	112	223	364	676	1221	1771	3423
Speedup	5	4	5	3	7	4	5	3

b. Về hiệu suất hiệu chỉnh sửa lỗi

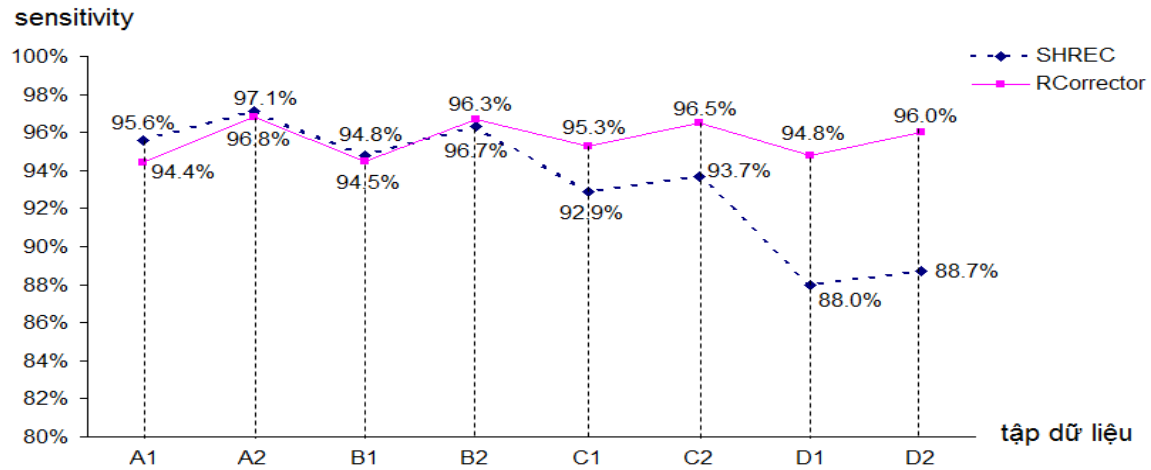
Hai đặc trưng sửa lỗi Sensitivity và Specificity của giải thuật RCorrector và SHREC được tính thông qua các đại lượng True Positive (TP), False Positive (FP), False Negative (FN) và True Negative (TN) được thể hiện ở Bảng 6 và Bảng 7. Trước hết, xét đặc trưng Specificity của hai giải thuật (cột cuối cùng của Bảng 6&7). Đặc trưng này thể hiện độ tin cậy của một giải thuật sửa lỗi. Nó phụ thuộc vào số read bị sửa nhầm (FP) so với số read đúng (TN). Đối với giải thuật SHREC, đặc trưng Specificity đạt tốt nhất (100%) trên tất cả các tập dữ liệu. Điều này là do số read bị sửa nhầm của giải thuật rất nhỏ. Đối với RCorrector, số read bị sửa nhầm có nhiều hơn so với SHREC nhưng cũng không đáng kể trên tổng số read đúng. Do đó, Specificity đạt cũng rất cao (hơn 99.6%) đối với tất cả các tập dữ liệu. Nói chung, cả hai giải thuật có độ tin cậy rất cao. Đặc điểm này là rất cần thiết đối với một giải thuật sửa lỗi.

Bảng 6: Các đặc trưng Sensitivity và Specificity trên 8 tập dữ liệu của Rcorrector

	TP	FP	FN	TN	Sensitivity	Specificity
A1	125482	312	7344	443988	0.944	0.999
A2	279094	201	9179	288522	0.968	0.999
B1	237351	1818	13636	839702	0.945	0.997
B2	527768	1091	17942	544951	0.967	0.998
C1	418988	1006	20265	1474961	0.953	0.999
C2	922982	495	32757	958333	0.965	0.999
D1	614505	3867	33172	2172240	0.948	0.998
D2	1354444	2516	55459	1409631	0.960	0.998

Bảng 7: Các đặc trưng Sensitivity và Specificity trên 8 tập dữ liệu của SHREC

	TP	FP	FN	TN	Sensitivity	Specificity
A1	127034	0	5803	444000	0.956	1
A2	280166	0	8143	288528	0.971	1
B1	238058	2	13015	839829	0.948	1
B2	525921	1	19945	545036	0.963	1
C1	408568	0	30868	1475016	0.929	1
C2	895823	0	60275	958354	0.937	1
D1	570705	1	77419	2172300	0.880	1
D2	1252574	1	158178	1409672	0.887	1



Hình 5: Đặc trưng Sensitivity của SHREC và RCorrector trên các tập dữ liệu

Đặc trưng Sensitivity của hai giải thuật được thể hiện ở cột thứ 6 của Bảng 6&7 và Hình 5. Đây là đặc trưng thể hiện khả năng phát hiện lỗi của các giải thuật. Với hai tập dữ liệu nhỏ (A1 và A2) thì Sensitivity của giải thuật SHREC tốt hơn giải thuật RCorrector. Như đối với các tập dữ liệu còn lại thì Sensitivity của RCorrector tốt hơn giải thuật SHREC. Điều này được giải thích như sau: Với dữ liệu nhỏ thì số lượng seed trong các mục trong cấu trúc INDEX không nhiều nên có thể không đạt ngưỡng để tiến hành sửa lỗi theo nguyên tắc biểu quyết theo số đông, còn đối với các tập dữ liệu lớn hơn thì nguyên tắc biểu quyết theo số đông hoạt động tốt hơn quá trình sửa lỗi của Rcorrector tốt hơn.

4 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Chúng tôi vừa trình bày trong bài báo giải thuật RCorrector sửa lỗi ở các chuỗi ADN ngắn thu được từ kỹ thuật xác lập trình tự thế hệ mới. Giải thuật RCorrector quyết định sửa một nucleotide lỗi thành nucleotide đúng theo nguyên tắc biểu quyết theo số đông. Phương pháp sửa lỗi này của RCorrector hoạt động hiệu quả với sự hỗ trợ của cấu trúc chỉ mục INDEX dựa trên kỹ thuật kmer. Kết quả thực nghiệm trên 8 tập dữ liệu giả lập cho thấy rằng giải thuật RCorrector nhanh hơn giải thuật SHREC từ 3

đến 7 với hai đặc trưng sửa lỗi Sensitivity và Specificty của hai giải thuật RCorrector và SHREC tương đương nhau. Trong tương lai, chúng tôi nghiên cứu để song song hóa giải thuật RCorrector trên PC cluster nhằm giảm thời gian sửa lỗi cho các tập dữ liệu lớn hơn.

TÀI LIỆU THAM KHẢO

1. Guy Haskin Fernald, Emidio Capriotti, Roxana Daneshjou,Konrad J. Karczewski, and Russ B. Altman. Bioinformatics challenges for personalized medicine. Bioinformatics (2011) 27 (13): 1741-1748
2. Li R., Zhu H., Ruan J., *et al*., De novo assembly of human genomes with massively parallel short read sequencing, Genome Research, volume 20, number 2, pp: 265–272, 2010.
3. Monya Baker, De novo genome assembly: what every biologist should know, Nature, volume 9, pp:333–337, 2012.
4. Pop M., Salzberg S. L., Bioinformatics challenges of new sequencing technology, Trends in Genetics, 24 (3), pp:142-149, 2008.
5. Salmela L., Correction of sequencing errors in a maxed set of reads. Bioinformatics 26(10) pp:1284-1290, 2010.
6. Sanger, F. *et al*. Nucleotide sequence of bacteriophage phi X174 DNA. Nature 265, 687–695 (1977).
7. Shendure J. and Ji H., Next-generation DNA sequencing, Nature biotechnology, volume 26, number 10, 1135-1145, 2008.
8. Schröder J., Schröder H., Simon J. P., Sinja R. and Schmidt B., SHREC: A short-read error correction method, Genome Analysis, volume 25, number 17, 2157-2163, 2009.
9. Tammi M. T., Arner E., Kindlund E., Andersson B., Correcting errors for shotgun sequencing, Nucleic Acid Research, 31, pp:4663-4672, 2003.
10. V.H. Nguyen, D. Lavenier, PLAST: parallel local alignment search tool for database comparison, BMC Bioinformatics 2009 10(329).