벡터에 기반한 휴리스틱을 이용한 RNA 이차 구조의 시각화

김도형, 한경숙
인하대학교 공과대학 자동화 공학과

Visualization of RNA secondary structure using vector-based heuristics

Dohyung Kim, Kyungsook Han
Automation Engineering Department, Inha University

요약

RNA 분자의 이차 구조를 예측하고 예측된 구조를 분석, 평가하기 위하여 시각화하는 작업은 RNA의 구조에 대한 연구에 있어서 가장 필수적인 과정이다. 본 논문은 이차 구조를 시각화하는 알고리즘과 이를 구현한 프로그램을 소개한다. 이 알고리즘은 vector와 vector space를 이용하여 RNA 분자의 구조 요소가 배치될 방향과 공간을 나타낸다. 구조 요소가 겹치지 않도록 배치될 방향과 공간을 효율적으로 찾기 위하여, 구조 요소를 배치하는 순서에 관한 휴리스틱과 구조 요소를 배치하는 방법에 관한 휴리스틱을 사용한다. 기존의 시각화 알고리즘이 구조 요소를 순차적으로 배치하면서 겹침 현상이 발생하면, 이를 제거하기 위하여 이미 배치한 구조 요소들을 재배치하거나 변형하는 것에 반하여, 이 알고리즘은 사용자의 판단이나 수작업에 의존하지 않고 겹침 현상이 거의 없는 이차 구조를 효율적으로 생성한다는 점에서 기존 방법을 많이 개선하였다

1. 서론

RNA 분자의 이차 구조는 본질적으로, 구성 염기간의 연결관계가 결정되면 유일하게 결정되는 topological structure이지, geometric structure가 아니다. 이것을 굳이 그래픽 형태로 나타내려는 목적 중의 하나는, 사람이 육안으로 보아서 쉽게 파악하고 비교할 수 있도록 표현하기 위한 것으로서, 많은 염기를 포함하는 RNA 분자일수록 적절한 시각화를 하지 않고 이차 구조를 파악하는 것은 거의 불가능하다.

대부분의 RNA 이차 구조 시각화 프로그램은 일단 구조 요소들이 겹침 (overlap)이 있는 구조를 생성하고, (1) 사용자의 수작업, 혹은 프로그램에 의하여 구조 요소들을 변형하는 작업을 (구조 요소의 휨, 찌그러짐, 길이 변형 등) 수행하여 겹침 현상을 제거하거나 (Devereux *et al.*, 1984; Shapiro *et al*, 1984), (2) 프로그램 스스로 backtracking을 하거나, 점진적으로 겹침이 줄어드는 구조를 반복적으로 생성함으로써 겹침을 제거한다 (Bruccoleri and Heinrich, 1988; Lapalme *et al*, 1982; Stüber 1985; Muller *et al.*, 1993; Perochon-Dorisse *et al*, 1995). 프로그램에 의하여 겹침 현상을 제거하는 경우, 구조 요소의 겹침을 피하기 위하여 구조 요소를 변형하게 되는데, 이 때 도입되는 변형 규칙이 모든 구조 요소에 일률적으로 적용되다 보니, 왜곡되게 보이는 (예를 들면, 특정 구조 요소가 지나치게 휘거나 찌그러진 모양) 이차 구조를 생성할 때가 많다. 또한, 이차 구조 시각화 프로그램이 필요로 하는 computational power가 높다 보니, 그 운용 환경은 대부분 PC가 아닌 mainframe 컴퓨터나 workstation 급 컴퓨터이어서, RNA를 연구하는 사람들에게 널리 보급되어 사용되기에 제약이 있다. 최근의 알고리즘 (Nakaya *et al.*, 1996)은 Barnes and Hut에 의하여 개발된 O(NlogN) force-calculation을 적용하여 전체적인 관점에서 겹침 현상을 제거하였다. 그러나 병렬 컴퓨터에서 병렬 언어를 사용하여 구현되었다는 단점이 있다. Nakaya의 알고리즘의 염기간의 force를 계산하는 방법을 개선한 연구가 본 저자들에 의하여 행해진 바 있다 (김도형과 한경숙, 1997).

본 논문은 사용자의 판단이나 개입을 요구하지 않으면서, 최소한의 왜곡만을 허용하는, RNA 이차 구조의 시각화 알고리즘과 이를 PC/Windows 환경에서 구현한 프로그램에 관하여 논한다.

2. 알고리즘

2.1 Input과 output

알고리즘의 input data 형식은 그림 1과 같다.

Data format I	형식	\$ <starting number> <base sequence> x <starting number> <matching parentheses and minus symbols>
	예	\$ 1 CAGgUCUCUCUGGUUUUAGACCAGA x 1 -----(((((((((-----)))))) \$ 26 UcUGAG x 26 ---)))
Data format II	형식	<base sequence> <matching parentheses and minus symbols>
	예	CAGgUCUCUCUGGUUUUAGACCAGAUcUGAG -----(((((((((-----))))))---)))

그림 1. Input data format

알고리즘의 output 형식은, loop과 helix를 각각 circle과 straight line으로 보여주는 outline view (그림 2)와 이차 구조를 이루는 염기까지 보여주는 standard view (그림 3)의 두 기지가 있다.

2.2 용어 정의

RNA 분자의 이차 구조 요소인 loop과 helix를 각각 circle과 straight line으로 간주한다. RNA 분자의 이차 구조를 straight line에 의하여 상호 연결된 circle의 관점에서 보여, circle을 배치하는 과정에서 straight line의 위치가 결정되도록 한다. circle과 straight line의 반지름과 길이는 그것을 이루는 염기의 개수에 의하여 결정된다.

loop v에 대한 adjacent helix라 함은 v에 직접적으로 연결된 helix를 의미한다. loop v에 대한 adjacent loop은 v

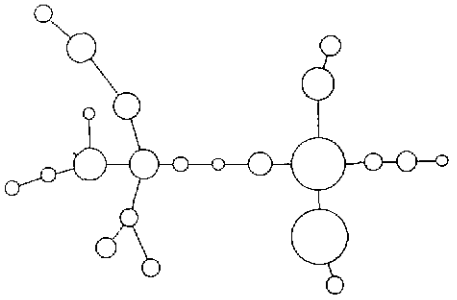

그림 2. 343개의 염기로 구성된 C. reinhardii chloroplast 16S like RNA의 domain 2에 대한 이차 구조의 outline view

그림 3 C. reinhardii chloroplast 16S like RNA의 domain 2의 이차 구조의 standard view

와 하나의 helix에 의하여 연결된 loop이다. loop v에 대한 seed loop은 v의 adjacent loop이며 동시에 이미 위치가 결정된 loop이다.

Vector space는 두 개의 고유한 vector에 의하여 정의되는 wedge region이다. 두 개의 고유한 vector는 두 개의 vector space를을 정의하므로, 원하는 공간에 대한 indicator로써 middle vector를 사용한다 (그림 4). circle이 놓일 위치를 탐색하고 결정하는 단계에서 vector space가 사용된다.

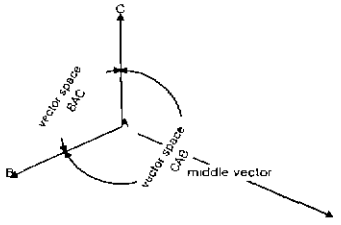

그림 4 vector space들과 middle vector

이미 위치가 결정된 circle (loop)과 straight line (helix)에 의하여 방해받지 않으며 개방된 공간을 open vector space라 한다. open vector space는 circle을 위치할 이상적인 공간이나, seed loop의 adjacent helix들과 seed loop을 이루는 염기에 의하여 항상 가능하지는 않다. 따라서 위치를 결정하고자 하는 대상 loop이 실질적으로 놓일 수 있는 공간을 vector space를 사용하여 따로 정의하며 이를 allowed vector space라 한다. 이 두 개의 vector space들을 고려하여 위치를 결정하고자 하는 loop이 실제로 놓일 방향을 결정하며, 이를 feasible vector라 한다.

2.3 알고리즘 개요

a. Regularizing a secondary structure
Pseudo 염기를 삽입하여 bulge loop, dangling end, 직접 이웃한 helix를 이차 구조에서 제거한다.

b. Determining positioning priority
Loop의 크기와 연결 정보를 사용하여 positioning priority를 결정한다.
1. 가장 큰 loop을 priority queue에 삽입한다.
2. priority queue와 연결되어 있으며, 삽입되어 있지 않은 loop들을 wait queue에 저장한다.
3. wait queue에 저장된 가장 큰 loop을 priority queue에 삽입한다.
4. 최종적으로 삽입된 loop과 priority queue에 삽입된 loop 사이에 존재하는 helix를 priority queue에 삽입한다.
5. 모든 loop과 stem이 priority queue에 삽입될 때까지 2, 3, 4과정을 반복한다.

c. Positioning and drawing structural elements
1. priority queue에서 target loop을 꺼내온다.
2. target loop의 open vector space를 계산한다.
3. target loop의 allowed vector space를 계산한다.
4. target loop의 feasible vector를 계산한다.
5. target loop의 위치를 결정하고 seed loop과 target loop 사이의 adjacent helix의 위치를 결정한다.
6. priority queue가 빌 때까지 1, 2, 3, 4, 5의 과정을 반복한다.

위의 알고리즘 c2 단계에서 연산량을 줄이기 위하여 exact open vector space가 아닌 approximate open vector space를 사용한다. open vector space의 계산 방법은 다음과 같다. open vector space의 left vector는 adjacent loop들 중 가장 오른쪽에 위치한 loop을 재귀적으로 방문함으로써 계산되어지며 right vector는 가장 왼쪽에 위치한 loop에 대하여 같은 과정을 수행함으로써 획득되어진다. 그림 5의 경우 open vector space의 left vector를 결정하기 위하여 D가 처음 방문된다. D에 연결되어 있으며 방문되지 않은 loop은 오직 F로 F를 방문한다. F의 경우 방문 가능한 것은 E와 H이다. 그들 중 E가 방문 방향에 대하여 가장 우측에 존재하므로 E를 방문한다 E의 경우 더 이상 방문 가능한 loop이 존재하지 않으므로 E는 left vector의 end point가 되며 left vector로 AE가 결정되어 진다. right vector를 결정하는 방법도 방문방향에 관하여 가장 좌측의 loop을 방문한다는 것을 제외하고는 동일하다. 그림 5에서 AC가 right vector로 결정되어 EAC가 최종적인 open vector space가 된다. BAC가 아닌 EAC가 선택된 이유는 approximate open vector space를 사용하였기 때문이다.

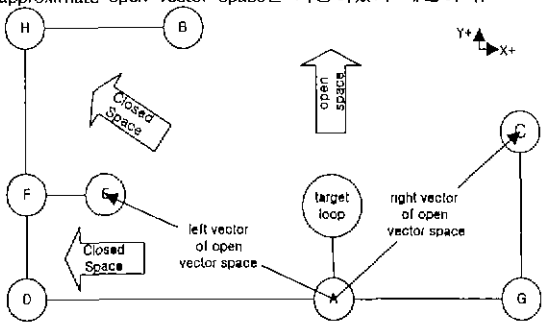

그림 5. Open vector space의 예

알고리즘 c3 단계의 allowed vector space의 계산 방법은 다음과 같다. seed loop의 adjacent helix들 중 target loop과 가장 인접한 helix들이 allowed vector space의 left vector와 right vector가 된다. 그림 6은 target loop이 양의 x,

양의 y 방향으로 위치되어야 할 경우의 original left, right vector와 seed loop의 염기에 의하여 새로이 계산된 left, right vector를 보여준다.

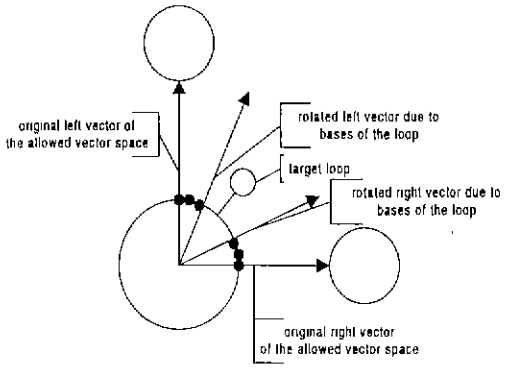

그림 6 allowed vector space의 예

c4 단계의 feasible vector는 다음과 같이 계산된다.

case 1 allowed vector space내에 open vector space의 middle vector가 존재하는 경우에는 feasible vector는 open vector space의 middle vector값을 취한다.

case 2 case 1이 아닌 경우에는 allowed vector space의 left, right vector중 open vector space의 middle vector와 가까운 것을 feasible vector의 값으로 결정한다

위 알고리즘은 n개의 염기로 구성된 RNA의 이차 구조의 시각화과정에 $O(n^2)$ 시간과 O(n) 공간량을 소요한다.

3. 구현 및 실험 결과

개발된 기법은 Borland C++ Builder를 사용하여 구현되었으며 (그림 7), Windows 95/98이 운용되는 IBM-PC 호환기종에서 동작된다.

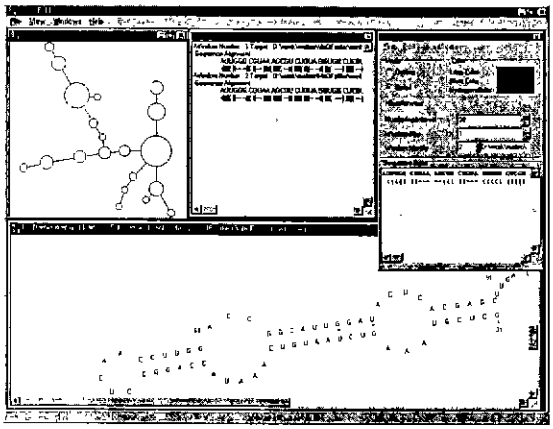

그림 7 알고리즘을 사용하여 구현된 프로그램. 왼쪽 상단 window로부터 시계 방향으로 outline view, history window, option window, standard view

	response time = visualization of sequence + profiler + window related code (cursor, caption, bitmap and etc.).
그림 2	86.35 ms
그림 3	288.105 ms

그림 8. 그림 2, 3의 실행 시간 (4회 측정값의 평균값) 동일한 input data에 대한 수행시간의 차이는 bitmap을 create 하는 시간 및 bitmap에 text, line, circle을 그리는 시간의 차이에 기인함

동작을 위한 최소환경으로 640×480 이상의 화면 해상도와 Intel Pentium이상의 CPU를 요구한다. 그림 2의 3의 생성에 소요된 시간을 Intel Pentium MMX 200MHz, RAM 64M의 환경에서 Zprofiler 2.20 (Baars, 1998)을 사용하여 측정하였다 (그림 8).

본 알고리즘은 간혹 부분적인 겹침을 허용하는데, 이는 다음과 같은 이유에 기인한다.

1. time complexity를 $O(n^3)$에서 $O(n^2)$으로 줄이기 위하여 approximate open vector space를 사용하였다.
2. allowed vector space의 계산 과정에서 loop (target loop, seed loop의 adjacent loop)의 반경이 고려되지 않는다.
3. backtracking등의 후처리 과정이 없다.

부분적인 겹침의 허용에도 불구하고 본 알고리즘은 다음의 기여를 하였다.

1. 기존 알고리즘은 염기 서열상의 존재하는 순서대로 loop과 helix의 위치를 순차적으로 결정하나, 본 알고리즘은 중요도가 큰 loop과 helix의 순서로 위치를 결정한다
2. 최소한의 distortion인 helix의 회전만으로 겹침을 제거한다.
3. 시각화 과정에 vector space의 개념을 도입하였다.
4. 사용자의 개입을 요구하지 않는다.
5. 일반적인 환경인 PC/Windows에서 구현되었다.

후기

본 연구는 1996년도 한국과학재단 핵심전문연구 (과제번호 961-0911-062-2) 연구비 지원을 받았음.

참고 문헌

Baars, A. (1998) Zprofiler, a Delphi component for high-resolution timing (version 2.20). Published electronically on the Internet.

Barnes, J. and Hut, P. (1986) A hierarchical O(NlogN) force-calculation algorithm. *Nature*, 324, 446-449.

Bruccoleri, R., E, and Heinrich, G. (1988) An improved algorithm for nucleic acid secondary structure display *CABIOS*, 4, 167-173.

Devereux, J., Haeberli, P., and Smithies, O. (1984) A comprehensive set of sequence analysis programs for the Vax. *Nucleic Acids Res.*, 12, 387-395.

Lapalme, G., Cedergren, R. J., and Sankoff, D (1982) An algorithm for the display of nucleic acid secondary structure. *Nucleic Acids Res*, 10, 8351-8356.

Muller, G., Gaspin, Ch., Etienne, A, and Westhof, E. (1993) Automatic display of RNA secondary structures *CABIOS*, 9, 551-561.

Nakaya, A, Taura, K., Yamamoto, K., and Yonezawa, A. (1996) Visualization of RNA secondary structures using highly parallel computers. *CABIOS*, 12, 205-211.

Perochon-Dorisse, J., Chetouani, F., Aurel, S., and Iscolo, N (1995) RNA_d2: a computer program for editing and display of RNA secondary structures *CABIOS*, 11, 101-109.

Shapiro, B. A., Maizel, J., Lipkin, L.E., Currey, K., and Whitney, C. (1984) Generating non-overlapping displays of nucleic acid secondary structure. *Nucleic Acids Res.*, 12, 75-88.

Stuber, K. (1985) Visualization of nucleic acid sequence structural information. *CABIOS*, 1, 35-42

김도형, 한경숙 (1997) 윈도우 95 환경에서 운용되는 RNA 이차 구조 예측 및 시각화 프로그램. 한국 정보과학회 가을 학술발표대회 논문집 (II), 695-698