ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN THU TRANG

BÀI TOÁN TÌM KIẾM MOTIF VÀ PHƯƠNG PHÁP TỐI ƯU ĐÀN KIẾN

LUẬN VĂN THẠC SĨ NGÀNH CÔNG NGHỆ THÔNG TIN

Hà Nội, năm 2016

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN THU TRANG

BÀI TOÁN TÌM KIẾM MOTIF VÀ PHƯƠNG PHÁP TỐI ƯU ĐÀN KIẾN

Ngành : Công nghệ thông tin
Chuyên ngành : Hệ thống thông tin
Mã số : 60480104

LUẬN VĂN THẠC SĨ NGÀNH CÔNG NGHỆ THÔNG TIN

Người hướng dẫn khoa học: PGS. TS Hoàng Xuân Huấn

Hà Nội, năm 2016

LỜI CẢM ƠN

Trước tiên, tôi xin gửi lời cảm ơn chân thành và lòng biết ơn sâu sắc nhất tới thầy giáo, PGS.TS. Hoàng Xuân Huấn, người thầy đáng kính đã tận tình chỉ bảo, hướng dẫn, động viên và giúp đỡ tôi trong suốt quá trình tìm hiểu, nghiên cứu và hoàn thiện luận văn. Thầy cũng đưa ra những góp ý chi tiết, tỉ mỉ hết sức quý báu giúp cho tôi có thể hoàn thành quyển luận văn này.

Thứ hai, tôi cũng xin được gửi lời cảm ơn sâu sắc tới em Dương Thị Ánh Tuyết, người đã giúp đỡ tôi giải quyết những khúc mắc trong quá trình viết chương trình để chạy thực nghiệm.

Thứ ba, tôi xin gửi lời cảm ơn tới các thầy cô trường Đại Học Công Nghệ - Đại Học Quốc Gia Hà Nội – những người đã tận tình giúp đỡ, cổ vũ và góp ý cho tôi trong suốt thời gian tôi học tập và nghiên cứu tại trường.

Thứ tư, tôi xin gửi lời cảm ơn tới các bạn học viên cùng học tập nghiên cứu tại trường Đại học Công nghệ đã hỗ trợ tôi rất nhiều trong quá trình học tập cũng như thực hiện luận văn.

Thứ năm, tôi xin gửi lời cảm ơn tới gia đình và bạn bè, những người thân yêu luôn bên cạnh, quan tâm, động viên tôi giúp tôi vượt qua khó khăn trong quá trình học tập và thực hiện luận văn tốt nghiệp này.

Cuối cùng tôi cũng bày tỏ lòng biết ơn về sự giúp đỡ của lãnh đạo trường, khoa Công nghệ thông tin – Trường cao đẳng Thống Kê cơ quan nơi tôi công tác đã tạo điệu kiện tốt nhất cho tôi về thời gian cũng như động viên tôi sớm hoàn thành bài luận văn.

Hà Nội, tháng 10 năm 2016

LỜI CAM ĐOAN

Tôi xin cam đoan rằng đây là công trình nghiên cứu của cá nhân tôi dưới sự hướng dẫn giúp đỡ của PGS.TS. Hoàng Xuân Huấn. Các kết quả được viết chung với các tác giả khác đều được sự đồng ý của tác giả trước khi đưa vào luận văn. Trong toàn bộ nội dung nghiên cứu của luận văn, các vấn đề được trình bày đều là những tìm hiểu và nghiên cứu của chính cá nhân tôi hoặc là được trích dẫn từ các nguồn tài liệu có ghi tham khảo rõ ràng, hợp pháp.

Trong luận văn, tôi có tham khảo đến một số tài liệu của một số tác giả được liệt kê tại mục tài liệu tham khảo

Hà nội, tháng 10 năm 2016

Nguyễn Thu Trang

MỤC LỤC

DANH MỤC KÝ HIỆU VÀ TỪ VIẾT TẮT

STT	Từ viết tắt	Từ hoặc cụm từ
1	ACO	*Ant Colony Optimization* (Tối ưu hóa đàn kiến)
2	AS	*Ant System* (Hệ kiến AS)
3	ACS	*Ant Colony System* (Hệ kiến ACS)
4	MMAS	*Max-Min Ant System* (Hệ kiến MMAS)
5	SMMAS	*Smooth-Max Min Ant System* (Hệ kiến MMAS trơn)
6	TSP	*Travelling Salesman Problem* (Bài toán người chào hàng)
7	TƯTH	*Tối ưu tổ hợp*
8	PMS	*Planted Motif Search*

DANH MỤC CÁC BẢNG

DANH SÁCH CÁC HÌNH VẼ

MỞ ĐẦU

Tin sinh học có ứng dụng cao trong cuộc sống, đặc biệt trong lĩnh vực y – dược. Về cơ bản, tin sinh học tập trung vào nghiên cứu và áp dụng các phương pháp cũng như các kĩ thuật trong tin học để giải quyết các bài toán trong sinh học phân tử. Tìm kiếm motif trong các chuỗi gene là một trong những bài toán quan trọng nhất của tin sinh học và thuộc loại NP-khó.

Các thành phần điều hòa gene (gene regulatory elements) được gọi là các DNA motif (về sau gọi là motif cho gọn), chúng chứa nhiều thông tin sinh học quan trọng. Vì vậy việc nhận dạng DNA motif đang là một trong những bài toán quan trọng nhất trong tin sinh học và thuộc loại NP-khó. Chủ yếu, có 2 cách tiếp cận để tìm kiếm motif: các phương pháp thực nghiệm và các phương pháp tính toán. Vì chi phí cao và tốn thời gian nên các phương pháp thực nghiệm ít hiệu quả. Phương pháp tính toán đang được dùng rộng rãi cho dự đoán motif.

Người ta đưa ra nhiều phát biểu cho bài toán tìm kiếm motif, và có nhiều thuật toán nghiên cứu và công bố giải quyết bài toán tìm kiếm motif. Trong luận văn này, tôi trình bày bài toán (ℓ,d) motif. Có nhiều thuật toán đưa ra để giải quyết bài toán (ℓ,d) motif, các thuật toán này có thể chia thành 2 loại đó là thuật toán chính xác và thuật toán xấp xỉ. Các thuật toán chính xác luôn luôn tìm ra những motif trong những chuỗi DNA đầu vào nhưng chỉ hiệu quả với các dữ liệu có kích thước nhỏ và thực hiện mất nhiều thời gian. Các thuật toán xấp xỉ có thể không tìm ra được tất cả các motif nhưng nó chạy hiệu quả với các dữ liệu lớn.

Luận văn đề xuất giải quyết bài toán (ℓ,d) motif theo thuật toán xấp xỉ, bằng việc đề xuất thuật toán tối ưu đàn kiến Ant colony optimization (ACO) để giải quyết bài toán (ℓ,d) motif. Đây là thuật toán mới và lần đầu được đưa vào để giải bài toán (ℓ,d) motif. Thuật toán được đặt tên là F-ACOMotif. Và trong thực nghiệm đã chỉ ra được thuật toán F-ACOMotif tối ưu hơn các thuật toán PairMotif+ và MEME về độ chính xác khi tìm ra (ℓ,d) motif.

Ngoài phần kết luận, cấu trúc nội dung của luận văn bao gồm 4 chương như sau:

Chương 1: Trình bày sơ lược các khái niệm về tin sinh học, bài toán tối ưu tổ hợp và phát biểu bài toán (ℓ,d) motif.

Chương 2: Giới thiệu thuật toán Ant colony optimization (ACO) và một vài thuật toán cập nhật mùi khác nhau trong ACO.

Chương 3: Đề xuất thuật toán, đó là thuật toán Ant colony optimization (ACO) để giải quyết bài toán (ℓ,d) motif.

Chương 4: Đưa ra kết quả thực nghiệm của luận văn, so sánh kết quả của thuật toán ACO với các thuật toán PairMotif+ và thuật toán MEME.

CHƯƠNG 1: TIN SINH HỌC VÀ BÀI TOÁN TÌM KIẾM (L,D) MOTIF

1.1. Tin sinh học

1.1.1 Giới thiệu về tin sinh học

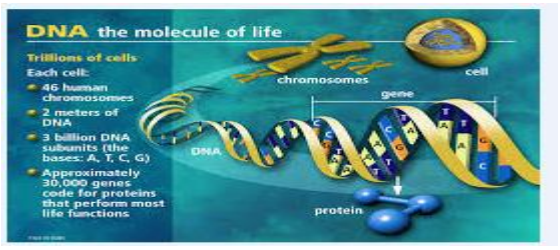
Tin sinh học (Bioinformatics) được tạo thành bởi cụm từ "Bio" là tương ứng với "Molecular Biology" nghĩa là sinh học phân tử còn "Informatics" thì tương đương với "Computer science" chính là khoa học máy tính. Ngoài ra Computational biology, Computational molecular biology, Biocomputing cũng đồng nghĩa với "Bioinformatics" [1]. Vậy Tin sinh học là gì? Fredj Tekaia Thuộc viện Pasteur đã đưa ra một định nghĩa về tin sinh học như sau:

"Tin sinh học là sử dụng toán học, thống kê và khoa học máy tính để giải quyết các vấn đề về sinh học với DNA, chuỗi axit amin và các thông tin có liên quan".

1.1.2 Khái niệm trong sinh học

Mọi cơ thể sống đều được cấu thành từ một lượng rất lớn các tế bào. Mỗi tế bào đều được cấu tạo gồm hạt nhân, ribôxom và nội bào. Hạt nhân của tế bào chứa các nhiễm sắc thể đặc trưng cho mỗi tế bào đó. Nhiễm sắc thể lại được tạo thành bởi các axit nucleic và protein. Axit nucleic là những đại phân tử có cấu trúc đa phân, đơn phân của nó là các nucleotide. Axit nucleic được chia làm 2 loại là DNA (deoxyribonucleic acid) và RNA. Một thành phần rất quan trọng khác của tế bào là protein, được tạo ra từ các axit amin, là các thành phần thiết yếu của mọi cơ quan và hoạt động hóa học liên quan đến toàn bộ hoạt động của tế bào, chúng được biểu hiện thành những đặc điểm về cấu tạo và chức năng của tế bào, hay chính là những tính trạng của sinh vật. Giữa protein và DNA có quan hệ chặt chẽ với nhau, cụ thể là mỗi loại protein đều được xác định bởi một đoạn trên dãy DNA gọi là gen.

1.1.2.1 DNA

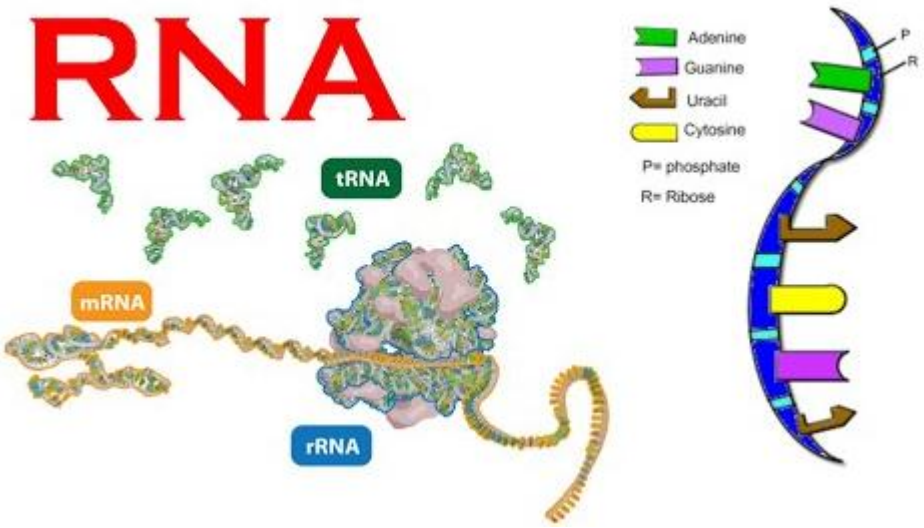


Hình 1.1: DNA phân tử của sự sống

Vào năm 1944, Oswald Avery phát hiện ra DNA là một loại nguyên liệu thô chứa gen. Bắt nguồn từ phát hiện này, một vài nhóm nghiên cứu đã tập trung nghiên cứu về DNA và các thành phần hóa học cấu thành. DNA là một phân tử được cấu tạo bởi đường, photphat và bốn nitrogenous bases: adenine, cytosine, guanine và thiamine, được lần lượt viết tắt là A, C, G, và T. Sau này, các nhà khoa học quan niệm rằng bốn nitrogen bases này là các nucleotide là cơ sở của mã di truyền.

Vào năm 1953, hai nhà sinh vật học là J.Wáton và F.Crick làm việc tại trường đại học Cambridge đã xây dựng thành công mô hình không gian của phân tử DNA(deoxyribonucleic acid), đánh dấu một bước ngoặt quan trọng trong sự phát triển của sinh học phân tử theo mô hình này DNA là một đại phân tử sinh học có cấu trúc như một chuỗi xoắn kép gồm hai mạch đơn, mỗi mạch đơn là một chuỗi nucleotide. Mỗi nucleotide gồm nhóm phosphate, đường desoxyribose và một trong bốn thành phần lần lượt được biểu thị bởi các chữ cái A, C, G và T. Hai mạch đơn kết hợp với nhau nhờ các liên kết hydro hình thành giữa các thành phần bổ sung nằm trên hai mạch. A bổ sung cho T, C bổ sung cho G.

1.1.2.2 RNA



Hình 1.2: Hình ảnh về RNA

RNA (Ribonucleic Acid) là 1 loại acid nucleic (như DNA), RNA cũng có cấu trúc đa phân mà đơn phân là 4 loại nucleotide, tuy nhiên trong RNA nucleotide loại T (pyrimidine thymine) được thay thế bằng U (uracil). RNA tồn tại ở dạng chuỗi đơn và được phân chia làm 3 loại chính dựa trên chức năng của

chúng:

mRNA (RNA thông tin): là một mạch sao chép nguyên từ một mạch đơn của DNA trong đó T được thay bằng U và làm nhiệm vụ truyền đạt thông tin cấu trúc protein được tổng hợp.

rRNA (RNA riboxom): là thành phần cấu tạo nên riboxom.

tRNA (RNA vận chuyển): có chức năng vận chuyển amino acid tương ứng đến nơi tổng hợp protein

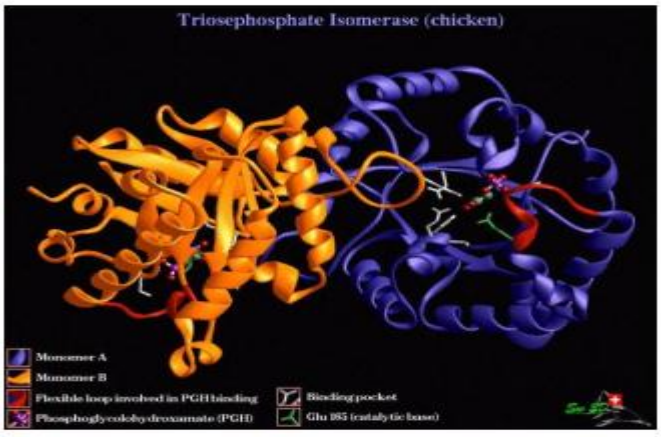
snRNA: có chức năng hỗ trợ việc ghép mã mRNA.

gRNA: sử dụng để điều khiển việc thay đổi mRNA.

RNA có thể liên kết với một dải đơn của một phân tử DNA, bằng cách thay T bằng U, và các phân tử kiểu này có vai trò quan trọng trong các quá trình sống và công nghệ sinh học [1].

C-G-A-T-T-G-C-A-A-C-G-A-T-G-C DNA
| | | | | | | | | | | | | |
G-C-U-A-A-C-G-U-U-G-C-U-A-C-G RNA

1.1.2.3 Protein



Hình 1.3: Cấu trúc Protein

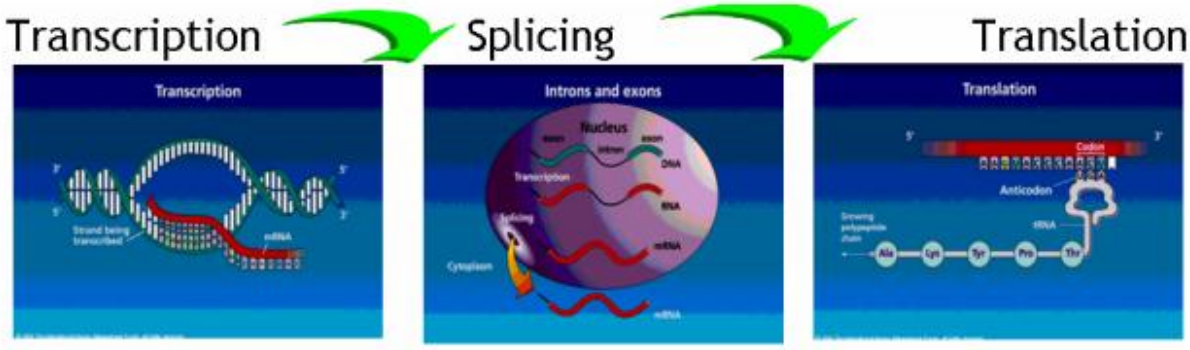
Protein là một đại phân tử sinh học được hình thành từ 1 hay nhiều chuỗi polypeptide sắp xếp theo một thứ tự đặc biệt, thứ tự này được xác định bởi dãy cơ sở (peptide là một chuỗi nối tiếp nhiều axit amin với số lượng ít hơn 30, với số lượng axit amin lớn hơn chuỗi được gọi là polypeptide) được hình thành từ 20 loại axit amin khác nhau lần lượt được biểu thị bằng 20 kí tự khác nhau trong bảng chữ cái. Từ " protein" dùng để chỉ một cấu trúc phức tạp trong không gian chứ không đơn thuần chỉ là một trình tự axit amin. Các nucleotide trong gene mã

hóa cho protein. Các protein cần thiết cho cấu trúc, chức năng và điều chỉnh tế bào, mô và tổ chức, mỗi protein có một vai trò đặc biệt.

Cấu trúc protein bao gồm 4 mức độ tổ chức: Cấu trúc bậc 1 là trình tự sắp xếp các axit amin trong chuỗi polypeptid, cấu trúc bậc 2 phát sinh từ sự uốn các thành phần của chuỗi polypeptid thành những cấu trúc đều đặn trong không gian (dạng xoắn α (alpha helix) hay lớp mỏng β (Beta sheets)). Cấu trúc bậc 3 quy định sự kết hợp các chuỗi xoắn hay lớp mỏng đó thành hình dạng ba chiều trong không gian. Cấu trúc bậc 4 là sự tổ chức nhiều chuỗi polypeptid thành một phân tử protein.

1.1.2.4 Quá trình tổng hợp protein

Tổng hợp protein là quá trình tạo ra protein dựa trên thông tin được mã hóa trong gen (là các đoạn mã đặc biệt của DNA có chức năng điều khiển cấu trúc và hoạt động của tế bào, là đơn vị chức năng của sự di truyền) gồm ba giai đoạn chính : (1) Transcription (phiên mã) (2) Splipcing (ghép mã) (3) Translation (dịch mã) [1] có thể được mô tả như hình dưới:



Hình 1.4: Quá trình tổng hợp Protein [1]

1.1.2.5 Một số bài toán trong tin sinh học

Việc hỗ trợ của công nghệ thông tin trong nghiên cứu cấu trúc các thành phần, quá trình hoạt động, đặc tính và vai trò của từng loại thành phần cùng liên kết giữa chúng dẫn đến phải giải quyết nhiều bài toán học máy phức tạp, thường là các bài toán tối ưu tổ hợp NP-khó và có tính bất định.

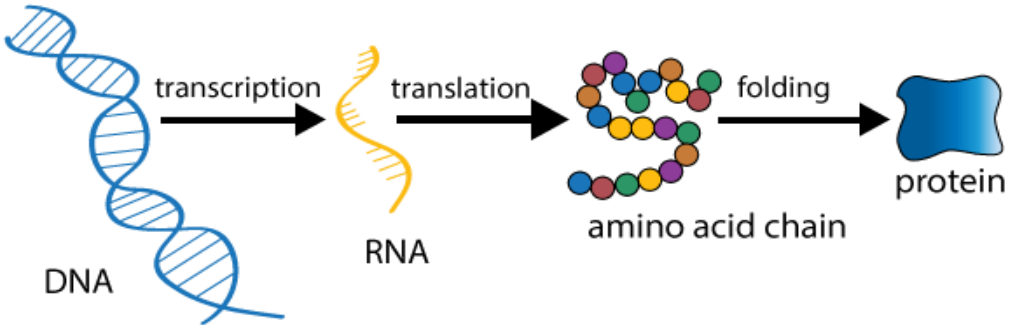
Một số bài toán hiện đang được quan tâm nghiên cứu là: So sánh tích hợp bộ gene (comparative genome assembly), xây dựng cây phân loài (phylogenetic tree reconstruction), tìm kiếm motif (motif finding), suy diễn haplotype, dự báo hoạt động điều tiết gene, xây dựng ma trận biến đổi axít amin, phân tích chức năng protein dựa trên cấu trúc bậc cao,….

Luận văn sẽ tập trung nghiên cứu "Bài toán tìm kiếm motif sử dụng phương pháp tối ưu đàn kiến"

1.1.3 Motif

1.1.3.1 Quá trình điều hòa gen

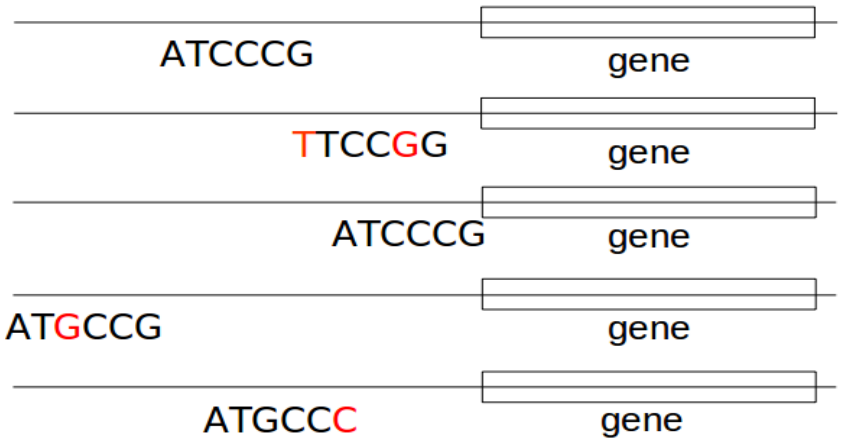
Các vị trí điều hòa trên DNA tương ứng với một chuỗi hợp nhất từ các vùng quy định của mỗi gen. Chúng ta gọi đó những *motif* hoặc DNA signals. Vị trí quy định trên mỗi DNA tương ứng với một motif được gọi là *instances* của motif đó. Xác định được các *motif* và các *instance* tương ứng của nó có ý nghĩ rất quan trọng, từ đó các nhà nghiên cứu sinh học có thể phát hiện ra các tương tác giữa DNA và protein, điều hòa gen cũng như sự phát triển và tương tác trong một tế bào.



Hình 1.5: Quá trình tổng hợp Protein

Motif là những đoạn trình tự có kích thước ngắn, có thể là nucleotide hoặc amino axit và mang ý nghĩa sinh học. Một vài đặc điểm của motif [15]:

- Motif là những mẫu có kích thước từ 10-25 base và lặp lại nhiều lần qua các chuỗi khác nhau.
- Motif là những đoạn trình tự đại diện cho vùng điều hòa của gen.
- Motif có kích thước nhỏ, cố định, lặp lại rất nhiều lần và thường xuyên.



Hình 1.6: Ví dụ về Motif

Khó khăn trong việc tìm kiếm motif [15]:

- Các Motif không bao giờ chính xác như chuỗi được bảo tồn. Luôn có những sự thay đổi ở một vài base.
- Kích thước của Motif quá ngắn so với kích thước của chuỗi DNA đang được xem xét.
- Vùng điều hòa bao gồm Motif có thể ở trị trí rất xa so với vùng mã hóa của gen khiến cho việc tìm kiếm trở nên khó khăn hơn rất nhiều.

Vùng điều hòa có thể nằm trên mảnh DNA đối diện với vùng mã hóa trong quá trình phiên mã

1.1.3.2 Ý nghĩa của Motif

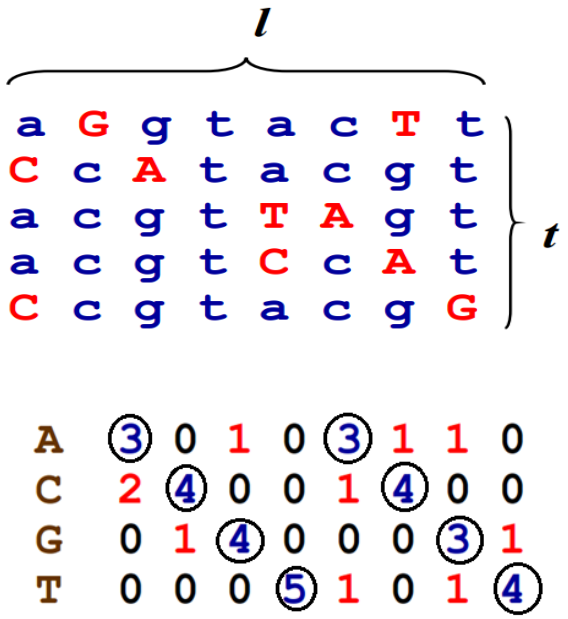
Ngoài những vùng mã hóa quan trọng, trong hệ gen còn có những vùng chứa các tín hiệu như tín hiệu khởi đầu phiên mã, tín hiệu cắt để xác định cùng intron exon …

Phần tử điều hòa (Regulatory element) được chia làm 2 loại: promoter và enhancer. Promoter là vùng gần với exon đầu tiên và là vị trí gắn (binding site) cho enzim điều khiển quá trình phiên mã (Transcription factor). Enhancer, trái lại, thường xuất hiện ở vị trí khá xa so với vùng mã hóa. Cả 2 vùng này đều có ý nghĩa trong việc kiểm soát sự biểu hiện của gen.

1.1.3.3 Biểu diễn Motif

1.1.3.3.1 Chuỗi hợp nhất và ma trận đặc trưng (Consensus sequence)

Chuỗi hợp nhất thường được dùng để đại diện cho vị trí gắn của emzim điều khiển quá trình phiên mã (Transcription factor binding). Là chuỗi gần như khớp hoàn toàn với trình tự gắn nhưng không chính xác hoàn toàn.



Hình 1.7: Chuỗi hợp nhất

Như ví dụ ở trên 'ACGTACGT' là chuỗi hợp nhất.

1.1.3.3.2 Ma trận

Có 3 cách biểu diễn ma trận

- Ma trận tần số: thể hiện tần số xuất hiện của từng base trên tất cả các trình tự xuất hiện.
- Ma trận tần suất: thể hiện tần suất xuất hiện của từng base
- Ma trận trọng số: trọng số mỗi bị trí base được tính theo công thức sau :

$$\sum_{\beta \epsilon \{A,C,G,T\}} f_{\beta k} \, log \frac{f_{\beta k}}{q_\beta}$$

Aligned Sites

site1	C	T	A	A	T	T	G	G	C	A	A	A	T	G
site2	A	T	A	A	T	A	A	A	C	A	A	A	A	C
site3	G	A	C	A	T	A	G	A	C	A	A	G	A	C
site4	G	T	C	T	T	T	C	A	C	A	A	A	T	A
site5	G	T	G	A	A	A	G	A	C	A	A	G	T	T
site6	A	T	A	A	T	A	A	A	C	A	A	A	A	T
site7	G	T	T	G	A	A	A	A	C	A	A	T	A	G
site8	T	C	A	A	A	T	A	T	C	A	A	A	T	C
site9	A	G	A	A	T	A	G	A	A	A	G	G	T	A

Regular Expression (RE)

N -- N -- N -- D -- W -- W -- V -- D -- M -- A -- R -- D -- W -- N

Position-specific Count Matrix (PSCM)

A	[	3	1	5	7	3	6	4	7	1	9	8	5	4	2]
C	[	1	1	2	0	0	0	1	0	8	0	0	0	0	3]
G	[	4	1	1	1	0	0	4	1	0	0	1	3	0	2]
T	[	1	6	1	1	6	3	0	1	0	0	0	1	5	2]

Position-specific Weight Matrix (PWM)

A	[	0.33	0.11	0.56	0.78	0.33	0.67	0.44	0.78	0.11	1.00	0.89	0.56	0.44	0.22]
C	[	0.11	0.11	0.22	0.00	0.00	0.00	0.11	0.00	0.89	0.00	0.00	0.00	0.00	0.33]
G	[	0.44	0.11	0.11	0.11	0.00	0.00	0.44	0.11	0.00	0.00	0.11	0.33	0.00	0.22]
T	[	0.11	0.67	0.11	0.11	0.67	0.33	0.00	0.11	0.00	0.00	0.00	0.11	0.56	0.22]

LOGO

Hình 1.8: Biểu diễn Motif

1.1.3.3.3 Biểu tượng

Biểu tượng là cách dùng hình ảnh biểu diễn cho Motif.

Hình 1.9: Biểu diễn Motif dạng sequence

1.2. Bài toán tối ưu tổ hợp và bài toán tìm kiếm (ℓ,d) motif

1.2.1 Bài toán tối ưu tổ hợp

1.2.1.1 Giới thiệu bài toán tối ưu tổ hợp

Mỗi bài toán tối ưu tổ hợp ứng với bộ ba (S, f, Ω), trong đó S là tập hữu hạn các trạng thái (lời giải tiềm năng hay phương án), f là hàm mục tiêu xác định trên S và Ω là tập các ràng buộc. Mỗi phương án $s \in S$ thỏa mãn các ràng buộc Ω gọi là phương án chấp nhận được. Mục tiêu của chúng là tìm ra phương án s^* tối ưu hóa toàn cục đối với hàm mục tiêu f, nói cách khác chính là tìm phương án s^* sao cho $f(s^*) \leq f(s)$ với mọi $s \in S$. Đối với bài toán này ta có 3 cách giải quyết đó là: vét cạn, kỹ thuật ăn tham hoặc phương pháp tối ưu trong lĩnh vực NP-khó.

Các thuộc tính của tập S, C và Ω như sau:

1) Ký hiệu X là tập các vectơ trên C có độ dài không quá h: $X = \{< u_0, \ldots, u_k > u_i \in C \ \forall i \leq k \leq h\}$. Khi đó, mỗi phương án s trong S được xác định nhờ ít nhất một vectơ trong X.
2) Tồn tại tập con X^* của X và ánh xạ φ từ X^* lên S sao cho $\varphi^{-1}(s)$ không rỗng với mọi $s \in S$, trong đó tập X^* có thể xây dựng được từ tập con C_0 nào đó của C nhờ thủ tục mở rộng tuần tự dưới đây.
3) Từ C_0 ta mở rộng tuần tự thành X^* như sau:

   i) Ta xem $x_0 = < u_0 >$ là mở rộng được với mọi $u_0 \in C_0$.

   ii) Giả sử $x_k = < u_0, \ldots, u_k >$ là mở rộng được và chưa thuộc X^*. Từ tập ràng buộc Ω, xác định tập con $J(x_k)$ của C, sao cho với mọi $u_{k+1} \in J(x_k)$

   thì $x_{k+1} = < u_0, \ldots, u_k, u_{k+1} >$ là mở rộng được.

   iii) Áp dụng thủ tục mở rộng từ các phần tử $u_0 \in C_0$ cho phép ta xây dựng được mọi phần tử của X^*.

1.2.1.2 Giới thiệu bài toán người chào hàng

Bài toán người chào hàng (*Traveling Salesman Problem - TSP*) là bài toán TƯTH điển hình, được nghiên cứu và xem như là bài toán chuẩn để đánh giá về hiệu quả lời giải các bài toán TƯTH.

Bài toán được phát biểu như sau:

Có một tập gồm n thành phố (hoặc điểm tiêu thụ) $C = \{c_1, c_2, \ldots, c_n\}$ *độ*

dài đường đi trực tiếp từ c_i đến c_j là $d_{i,j}$. Một người chào hàng muốn tìm một hành trình ngắn nhất từ nơi ở, đi qua mỗi thành phố đúng một lần để giới thiệu sản phẩm cho khách hàng, sau đó trở về thành phố xuất phát.

Có thể thấy đây chính là bài toán tìm chu trình Hamilton với đồ thị đầy đủ có trọng số $G = (V, E)$, với V là tập các đỉnh với nhãn là các thành phố trong C, E là tập các cạnh nối các thành phố tương ứng, độ dài mỗi cạnh chính là độ dài đường đi giữa hai thành phố tương ứng. Trong trường hợp này, tập S sẽ là tập các chu trình Hamilton trên G, f là độ dài của chu trình, Ω là ràng buộc đòi hỏi chu trình là chu trình Hamilton (qua tất cả các đỉnh, mỗi đỉnh đúng một lần), C là tập thành phố được xét, C_0 trùng với C, tập X là vectơ độ dài n: $x = (x_1, \dots, x_n)$ với $x_i \in C \; \forall \; i \leq n$, còn X^* là các vectơ trong đó x_i khác x_j đối với mọi cặp (i, j).

Do đó, lời giải tối ưu của bài toán TSP là một hoán vị π của tập đỉnh $\{c_1, c_2, .., c_n\}$ sao cho hàm độ dài $f(\pi)$ là nhỏ nhất, trong đó $f(\pi)$ được tính theo (1):

$$f(\pi) = \sum_{i=1}^{n-1} d(\pi(i), \pi(i+1)) + \; d(\pi(n), \pi(1)) \qquad (1.1)$$

1.2.1.3 Các cách tiếp cận giải quyết bài toán tối ưu tổ hợp

Như phần trên ta đã thấy các bài toán TƯTH có thể đưa về bài toán tìm kiếm trên đồ thị. Với những bài toán cỡ nhỏ hoặc những bài toán đặc biệt thì ta hoàn toàn có thể tìm lời giải tối ưu nhờ tìm kiếm vét cạn cũng như xây dựng những lời giải đặc thù riêng. Tuy nhiên hầu hết các bài toán trong số đó là bài toán NP-khó, nên với các bài toán cỡ lớn người ta phải tìm lời giải gần đúng. Các thuật toán gần đúng đối với các bài toán TƯTH khó thường dựa trên 2 kỹ thuật cơ bản: heuristic cấu trúc (construction heuristic) và tìm kiếm địa phương (local search).

1.2.1.3.1 Heuristic cấu trúc

Khi không thể tìm lời giải tối ưu của bài toán với thời gian đa thức, chúng ta hướng đến việc tìm lời giải gần đúng. Kỹ thuật hay dùng trong việc tìm lời giải gần đúng là heuristic cấu trúc, lời giải của bài toán được xây dựng thông qua việc mở rộng tuần tự. Từ thành phố khởi tạo trong tập C_0, từng bước mở rộng không quay lui, thêm vào các thành phần mới theo phương thức ngẫu nhiên hay tất định dựa trên những quy tắc heuristic. Các quy tắc heuristic này khác nhau tùy vào thuật toán cụ thể được xây dựng dựa trên toán học kết hợp với kinh

nghiệm. Chúng ta có thể khái quát hóa để mô phỏng dưới dạng thuật toán như sau:

Procedure Heuristic cấu trúc;

Begin

$\boldsymbol{s_p}$ ← chọn thành phần $\boldsymbol{u_0}$ trong $\boldsymbol{C_0}$;

While (chưa xây dựng xong lời giải) **do**

$\boldsymbol{c}$ ←GreedyComponent($\boldsymbol{s_p}$);

$\boldsymbol{s_p} \leftarrow \boldsymbol{s_p}$ ^ $\boldsymbol{c}$;

end-while

$\boldsymbol{s} \leftarrow \boldsymbol{s_p}$;

Đưa ra lời giải $\boldsymbol{s}$;

End;

Hình 1.10: Phương pháp heuristic cấu trúc

Trong đó GreedyComponent(s_p) có nghĩa là chọn thành phần bổ sung vào $s_p$theo quy tắc heuristic đã có. Ký hiệu s_p ^ c là kết quả phép toán thêm thành phần c vào s_p.

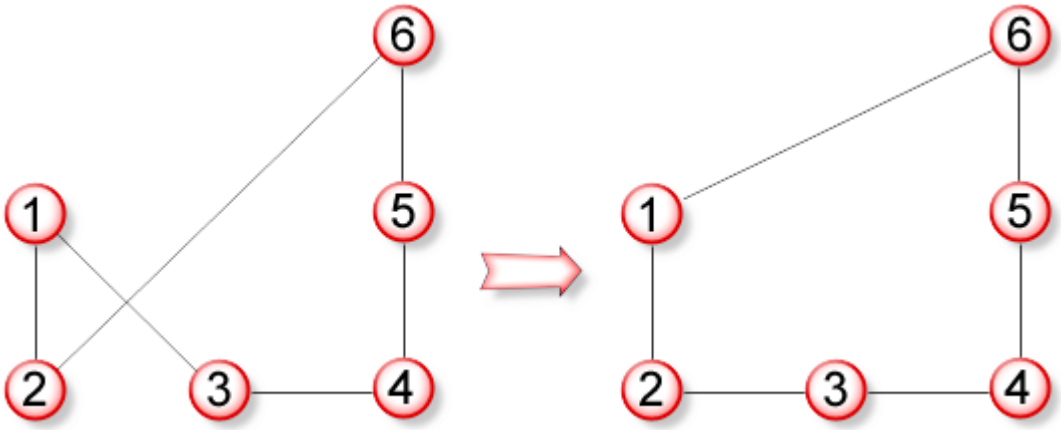
Với phương pháp trên ta có thể áp dụng cho bài toán TSP với đồ thị đầy đủ và sử dụng quy tắc heuristic láng giềng gần nhất để chọn đỉnh thêm vào (đỉnh láng giềng nhỏ nhất chưa đi qua để thêm vào). Thuật toán kiểu này có ưu điểm là thời gian tính toán nhanh nhưng lại không có khả năng cải tiến lời giải qua mỗi bước lặp.

1.2.1.3.2 Tìm kiếm địa phương

Kỹ thuật tìm kiếm cục bộ hay còn gọi là tìm kiếm địa phương, thực hiện bằng cách bắt đầu từ một phương án chấp nhận được, lặp lại bước cải tiến lời giải nhờ các thay đổi cục bộ. Để thực hiện kỹ thuật này, ta cần xác định được *cấu trúc lân cận* của mỗi phương án (lời giải) đang xét, tức là những phương án chấp nhận được, gần với nó nhất, nhờ thay đổi một số thành phần. Cách thường dùng là lân cận *k-thay đổi*, tức là *lân cận* bao gồm các phương án chấp nhận được khác với phương án đang xét nhờ thay đổi nhiều nhất k thành phần.

Ví dụ. Lân cận *2-thay đổi* của một lời giải *s* trong bài toán TSP bao gồm tất cả các lời giải s' có thể nhận được từ *s* bằng cách đổi hai cạnh. Hình 1.11 chỉ ra một ví dụ một lời giải nhận được bằng cách thay hai cạnh (1,3), (2,6) bằng hai cạnh (2,3), (1,6).

Việc cải tiến trong các bước lặp thường chọn theo phương pháp leo đồi dựa theo hai chiến lược: Chiến lược *tốt nhất* và chiến lược *tốt hơn*. Với chiến lược *tốt nhất*, người ta thực hiện chọn lời giải tốt nhất trong lân cận để làm lời giải cải tiến. Tuy nhiên, khi bài toán cỡ lớn có thể không tìm được lời giải tốt nhất do bị hạn chế về thời gian. Còn với chiến lược *tốt hơn*, ta chọn phương án đầu tiên trong lân cận, cải thiện được hàm mục tiêu. Nhược điểm của tìm kiếm địa phương là thường chỉ cho cực trị địa phương.



Hình 1.11: Lời giải nhận được thông qua tìm kiếm địa phương

Các kỹ thuật trên thường được kết hợp, tạo thành các hệ lai trong các phương pháp mô phỏng tự nhiên dựa trên quần thể, chẳng hạn như thuật toán di truyền (GA) hoặc tối ưu đàn kiến (ACO).

1.2.1.3.3 Phương pháp metaheuristic

Phương pháp metaheuristic là một phương pháp heuristic tổng quát được thiết kế, định hướng cho các thuật toán cụ thể (bao gồm cả heuristic cấu trúc và tìm kiếm địa phương). Như vậy, một metaheuristic là một lược đồ thuật toán tổng quát ứng dụng cho các bài toán tối ưu khác nhau, với một chút sửa đổi cho phù hợp với từng bài toán.

1.2.1.3.4 Phương pháp Memetic

Memetic là một mô hình theo phương pháp metaheuristic. Trong các thuật toán được thiết kế theo memetic, người ta tạo ra nhiều thế hệ quần thể lời giải chấp nhận được. Trong mỗi quần thể của thế hệ tương ứng, ta chỉ chọn ra một số lời giải (chẳng hạn lời giải tốt nhất) để thực hiện tìm kiếm địa phương nhằm cải thiện chất lượng. Quá trình tiến hóa này cho ta tìm được lời giải tốt nhất có thể. Hình 1.12 mô tả một thuật toán memetic sử dụng tính toán tiến hóa (*Evolutionary Computing - EC*):

Proedure Thuật toán memetic-EC;

Begin

Initialize: Tạo ra quần thể đầu tiên;

while điều kiện dừng chưa thỏa mãn **do**

Đánh giá các cá thể trong quần thể;

Thực hiện tiến hóa quần thể nhờ các toán tử cho trước;

Chọn tập con $\boldsymbol{\Omega_{il}}$ để cải tiến nhờ thủ tục tìm kiếm địa phương;

for mỗi cá thể trong $\boldsymbol{\Omega_{il}}$**do**

Thực hiện tìm kiếm địa phương;

end-for

Chọn phần tử tốt nhất;

end-while;

Đưa ra lời giải tốt nhất;

End;

Hình 1.12: Thuật toán memetic sử dụng EC

Trong ứng dụng thực tế, các thuật toán ACO thường được kết hợp với tìm kiếm địa phương theo mô hình memetic này

1.2.2 Phát biểu bài toán tìm kiếm (ℓ,d) motif

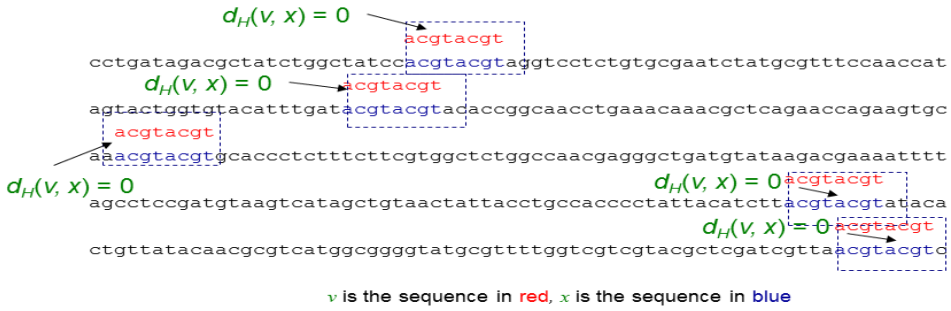
Trước khi đưa ra bài toán, luận văn đưa ra định nghĩa sau:

Định nghĩa: (Hamming distance)

Cho x và y tương ứng là hai xâu độ dài ℓ và n, khoảng cách Hamming $d_H(x,y)$ được xác định như sau:

a) $d_H(x,y)$ = số vị trí khác nhau của x và y nếu $\ell = n$

b) $d_H(x,y)$ = min{$d_H(x,m)$/ m là xâu con độ dài ℓ của y} nếu $\ell < n$



- $TotalDistance(v, DNA) = 0$

Hình 1.13: Ví dụ khoảng cách hamming

Xác định được các motif và các instance tương ứng của nó có ý nghĩa rất quan trọng, từ đó các nhà nghiên cứu sinh học có thể phát hiện ra các tương tác giữa DNA và protein, điều hòa gen cũng như sự phát triển và tương tác trong một tế bào. Các bài toán tìm kiếm motif đã thu hút được nhiều nhà nghiên cứu. Có nhiều phát biểu cho bài toán tìm kiếm motif. Điển hình có thể kể đến 3 bài toán tìm kiếm motif như sau [14]: Simple Motif Search, (ℓ,d) Motif Search (Planted Motif Search) và Edited Motif Search

Trong luận văn này, chúng tôi sẽ tập trung nghiên cứu bài toán (ℓ,d) Motif Search (LDMS) hay chính là bài toán Planted Motif Search (PMS) từ nay sẽ gọi là bài toán PMS.

Bài toán PMS được phát biểu như sau:

Cho một tập hợp N chuỗi S =$\{S_1, S_2, .., S_N\}$*, trong đó mỗi phần tử được lấy ra từ tập* $\sum = \{A, C, G, T\}$ *và hai số nguyên không âm* ℓ *và d, thỏa mãn* $0 \leq d < \ell < n$.

Bài toán (ℓ,d)-motif là tìm chuỗi m độ dài ℓ *từ* $\sum$ *và một tập chuỗi con* $M=\{m_1, m_2, .., m_N\}$ *trong đó,* m_i *tương ứng là chuỗi con của* S_i *có cùng độ dài* ℓ *sao cho* $d_H(m, S_i) \leq d$

<u>*Ví dụ:*</u>

Mô tả cho việc tìm kiếm (ℓ,d) – motif. Giả sử S là tập gồm 3 chuỗi S_1, S_2, S_3 trong đó:

S_1: GCGCGAT

S_2: CAGGTGA

S_3: CGATGCC

Giả sử cho 2 tham số đầu vào ℓ = 3; và d = 1. Sau khi S được kiểm tra bằng một thuật toán tìm kiếm (ℓ,d) – motif, ta có thể tìm được motif m là: GAT và GTG

Hiện nay có hai phương pháp để tìm kiếm motif:

- Bằng thực nghiệm trong sinh học: Tốn thời gian, chi phí cao, mất nhiều công sức, độ chính xác cao.
- Bằng tính toán trong tin học: Hoàn toàn có thể thực hiện được trong thời gian và chi phí thấp nhưng chỉ đưa ra được các chuỗi có khả năng là motif.

Với hướng tiếp cận bằng tính toán, có hai phương pháp tìm kiếm là chính xác và gần đúng. Các thuật toán chính xác luôn luôn tìm ra những motif trong những chuỗi DNA đầu vào nhưng chỉ hiệu quả với các dữ liệu có kích thước nhỏ và thực hiện mất nhiều thời gian. Một số thuật toán chính xác phổ biến hiện nay: PMS6, PMS5, Pampa, PMSPrune, Voting, RISSOTO, MITRA, PairMotif. Các thuật toán xấp xỉ có thể không tìm ra được tất cả các motif nhưng nó chạy hiệu quả với các dữ liệu lớn, tiêu biểu có: MEME, Gibbs sampler, Genetic Algorithm (GA), PairMotif+.

CHƯƠNG 2. GIỚI THIỆU VỀ THUẬT TOÁN ANT COLONY OPTIMIZATION (ACO)

2.1 Giới thiệu về thuật toán ACO

Tối ưu đàn kiến (*Ant Colony Optimization – ACO*) là một phương pháp metaheuristic được đề xuất bởi Dorigo vào năm 1991dựa trên ý tưởng mô phỏng cách tìm đường đi từ tổ tới nguồn thức ăn và ngược lại của các con kiến tự nhiên để giải gần đúng bài toán TƯTH NP-khó.

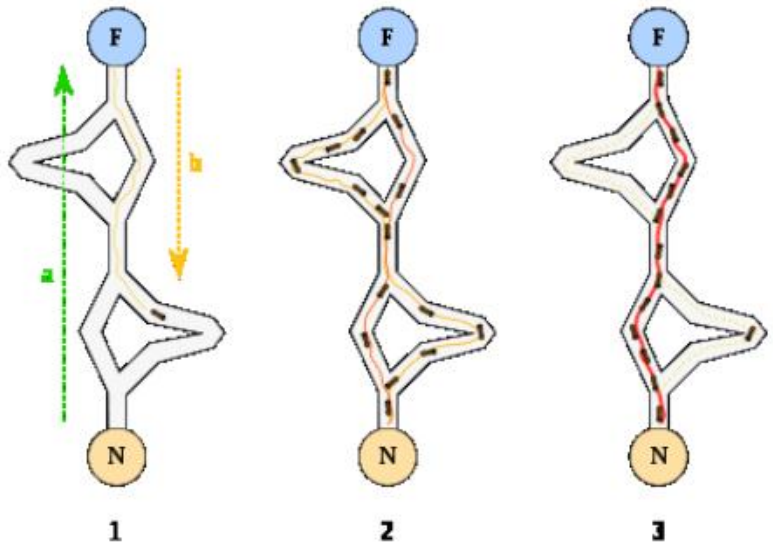
Trên đường đi của mình các con kiến thực để lại một vết hóa chất được gọi là vết mùi (*pheromone trail*), đặc điểm sinh hóa học của vết mùi này là có khả năng ứ đọng, bay hơi và là phương tiện giao tiếp báo cho các con kiến khác thông tin về đường đi đó một cách gián tiếp. Các con kiến sẽ lựa chọn đường đi nào tồn đọng lượng mùi hay có cường độ vết mùi lớn nhất tại thời điểm lựa chọn để đi, nhờ cách giao tiếp mang tính gián tiếp và cộng đồng này mà đàn kiến trong tự nhiên tìm được đường đi ngắn nhất trong quá trình tìm thức ăn mang về tổ và ngược lại. Sử dụng mô hình kiến nhân tạo này Dorigo (1991) [13] đã xây dựng thuật toán *hệ kiến* (AS) giải bài toán người chào hàng. Thuật toán này đã được chứng minh tính hiệu quả thông qua thực nghiệm so với các mô phỏng tự nhiên khác như SA (mô phỏng luyện kim) và GA (giải thuật di truyền). Thuật toán này về sau được phát triển và có nhiều áp dụng phong phú trong thực tế, được gọi chung là phương pháp ACO.

Theo ý tưởng này, các thuật toán ACO sử dụng thông tin heuristic kết hợp thông tin học tăng cường qua các vết mùi của các con kiến nhân tạo (*artificial ant*) để giải các bài toán tối ưu tổ hợp khó bằng cách đưa về bài toán tìm đường đi tối ưu trên đồ thị cấu trúc tương ứng được xây dựng từ đặc điểm của từng bài toán cụ thể. Thuật toán ACO đầu tiên là hệ kiến (*Ant System – AS)* giải bài toán Người chào hàng TSP, đến nay các thuật toán ACO đã áp dụng một cách phong phú để giải nhiều bài toán tối ưu tổ hợp khác nhau và hiệu quả nổi trội của nó đã được chứng tỏ bằng thực nghiệm.

2.2 Mô hình mô phỏng của thuật toán

2.2.1 Kiến tự nhiên

Khi tìm đường đi, đàn kiến trao đổi thông tin gián tiếp và hoạt động theo phương thức tự tổ chức. Phương thức này tuy đơn giản nhưng đã giúp cho đàn kiến có thể thực hiện được những công việc phức tạp vượt xa khả năng của từng con kiến, đặc biệt là khả năng tìm đường đi ngắn nhất từ tổ đến nguồn thức ăn [12, tr.16] (xem hình 2.1) (mặc dù, kiến không có khả năng đo độ dài đường đi). Kiến chịu ảnh hưởng của các vết mùi của các con kiến khác chính là ý tưởng thiết kế thuật toán ACO.



Hình 2.1: Thể hiện hành vi của mỗi con kiến trong tự nhiên

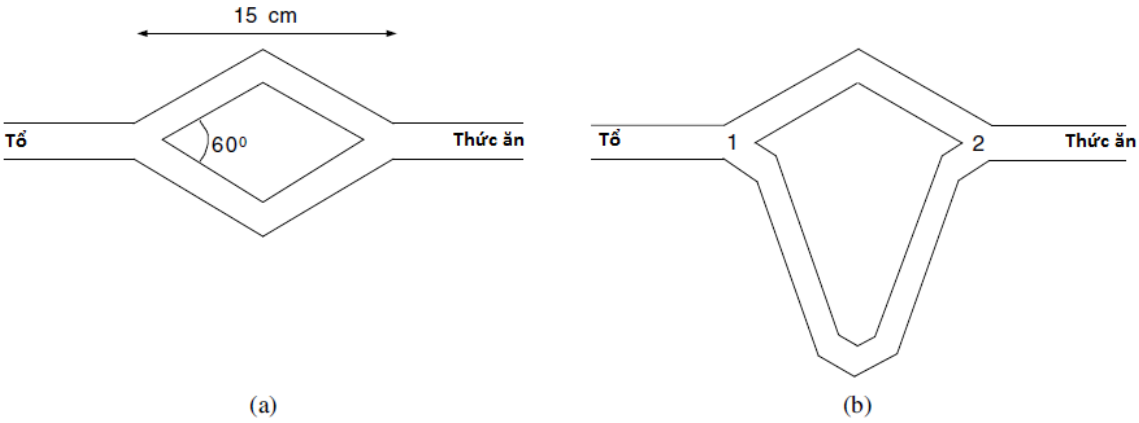
Để làm được điều đó, trên đường đi, mỗi con kiến để lại vết mùi dùng để đánh dấu đường đi. Bằng cách cảm nhận vết mùi, con kiến có thể lần theo đường đi đến nguồn thức ăn được các con kiến khác khám phá theo phương thức chọn ngẫu nhiên, có định hướng theo nồng độ vết mùi.

Con kiến chịu ảnh hưởng của các vết mùi của các con kiến khác, đây là ý tưởng chính để thiết kế thuật toán ACO.

Thí nghiệm trên cây cầu đôi

Có nhiều thực nghiệm nghiên cứu về hành vi để lại vết mùi và đi theo vết mùi của loài kiến. Thực nghiệm, được thiết kế bởi Deneubourg và các đồng nghiệp [12, tr.17-19], dùng một chiếc cầu đôi nối từ tổ kiến tới nguồn thức ăn, như minh họa trong hình 2.2. Họ đã thực nghiệm với tỉ lệ độ dài đường $r = \frac{l_l}{l_s}$ giữa hai nhánh khác nhau của chiếc cầu đôi, trong đó l_l là độ dài của nhánh dài còn l_s là độ dài của nhánh ngắn.

Trong thực nghiệm thứ nhất, chiếc cầu đôi có hai nhánh bằng nhau (r = 1, hình 2.2.a). Ban đầu, kiến lựa chọn đường đi một cách tự do đi từ tổ đến nguồn thức ăn, cả hai nhánh đều có kiến đi, nhưng sau một thời gian các con kiến này tập trung đi theo cùng một nhánh. Kết quả có thể được giải thích như sau: Ban đầu không có vết mùi nào trên cả hai nhánh, do đó kiến lựa chọn nhánh bất kỳ với xác suất như nhau. Một cách ngẫu nhiên, sẽ có một nhánh có số lượng kiến lựa chọn nhiều hơn nhánh kia. Do kiến để lại vết mùi trong quá trình di chuyển, nhánh có nhiều kiến lựa chọn sẽ có nồng độ mùi lớn hơn nồng độ mùi của nhánh còn lại. Nồng độ mùi trên cạnh lớn hơn sẽ ngày càng lớn hơn vì ngày càng có nhiều kiến lựa chọn. Cuối cùng, hầu như tất cả các kiến sẽ tập trung trên cùng một nhánh.Thực nghiệm này cho thấy là sự tương tác cục bộ giữa các con kiến với thông tin gián tiếp là vết mùi để lại cho phép điều chỉnh hoạt động vĩ mô của đàn kiến.



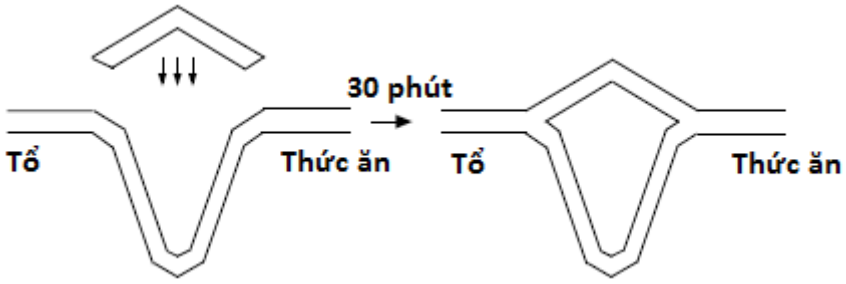
Hình 2.2: Thực nghiệm cây cầu đôi

(a) Hai nhánh có độ dài bằng nhau. (b) Hai nhánh có độ dài khác nhau.

Trong thực nghiệm thứ hai (xem hình 2.2 b), độ dài của nhánh dài gấp đôi độ dài nhánh ngắn (tỉ lệ $r = 2$). Trong trường hợp này, sau một thời gian tất cả các con kiến đều chọn đoạn đường ngắn hơn. Cũng như trong thực nghiệm thứ nhất, ban đầu đàn kiến lựa chọn hai nhánh đi như nhau, một nửa số kiến đi theo nhánh ngắn và một nửa đi theo nhánh dài (mặc dù trên thực tế, do tính ngẫu nhiên có thể một nhánh nào đó được nhiều kiến lựa chọn hơn nhánh kia). Nhưng thực nghiệm này có điểm khác biệt quan trọng với thực nghiệm thứ nhất: Những kiến lựa chọn đi theo nhánh ngắn sẽ nhanh chóng quay trở lại tổ và khi phải lựa chọn giữa nhánh ngắn và nhánh dài, kiến sẽ thấy nồng độ mùi trên nhánh ngắn cao hơn nồng độ mùi trên nhánh dài, do đó sẽ ưu tiên lựa chọn đi theo nhánh

ngắn hơn. Tuy nhiên, trong thời gian đầu không phải tất cả các kiến đều đi theo nhánh ngắn hơn. Phải mất một khoảng thời gian tiếp theo nữa bầy kiến mới lựa chọn đi theo nhánh ngắn. Điều này minh chứng bầy kiến đã sử dụng phương thức thăm dò, tìm đường mới.

Một điểm thú vị nữa là quan sát xem sẽ xảy ra điều gì khi quá trình tìm kiếm đang hội tụ, lại xuất hiện một đường mới từ tổ đến nguồn thức ăn. Việc này được thực nghiệm như sau: Ban đầu từ tổ đến nguồn thức ăn chỉ có một nhánh dài và sau 30 phút, thêm một nhánh ngắn (xem hình 2.3). Trong trường hợp này, nhánh ngắn thường không được kiến chọn mà chúng tập trung đi trên nhánh dài. Điều này có thể giải thích như sau: nồng độ vết mùi trên cạnh dài cao và sự bay hơi của vết mùi diễn ra chậm nên đại đa số các con kiến vẫn lựa chọn nhánh dài (có nồng độ vết mùi cao). Hành vi này tiếp tục được củng cố kiến chọn đi theo nhánh dài, ngay cả khi có một nhánh ngắn xuất hiện. Việc bay hơi vết mùi là cơ chế tiện lợi cho việc tìm đường mới, nghĩa là việc bay hơi có thể giúp kiến quên đi đường đi tối ưu địa phương đã được tìm thấy trước đây để tìm khám phá đường đi mới, tốt hơn.



Hình 2.3: Thí nghiệm bổ xung

(Ban đầu chỉ có một nhánh và sau 30 phút thêm nhánh ngắn hơn)

2.2.2 Kiến nhân tạo (Artificial Ant)

Thực nghiệm cây cầu đôi cho thấy đàn kiến tự nhiên có thể sử dụng luật di chuyển theo xác suất, dựa trên thông tin địa phương để tìm được đường đi ngắn nhất giữa hai địa điểm. Vết mùi của đàn kiến cho phép liên tưởng tới cách học tăng cường (reinforcement learning) trong bài toán chọn tác động tối ưu[6], gợi mở mô hình mô phỏng cho bài toán tìm đường đi ngắn nhất giữa hai nút (tương ứng là tổ và nguồn thức ăn) trên đồ thị, trong đó các tác tử (agent) là đàn kiến nhân tạo.

Tuy nhiên, trong các bài toán ứng dụng các đồ thị thường phức tạp hơn. Từ mỗi đỉnh có thể có nhiều cạnh, nên nếu mô phỏng thực sự hành vi của đàn kiến tự nhiên nhiều con kiến sẽ đi luẩn quẩn và do đó hiệu quả thuật toán sẽ rất kém. Vì vậy, người ta dùng kỹ thuật đa tác tử (multiagent) mô phỏng đàn kiến nhân tạo, trong đó mỗi con kiến nhân tạo có khả năng nhiều hơn so với kiến tự nhiên. Kiến nhân tạo (về sau trong luận văn ta sẽ gọi đơn giản là kiến) có bộ nhớ riêng, có khả năng ghi nhớ các đỉnh đã thăm trong hành trình và tính được độ dài đường đi nó chọn. Ngoài ra, kiến có thể trao đổi thông tin với nhau, thực hiện tính toán cần thiết, cập nhật mùi…

Sử dụng mô hình kiến nhân tạo này, Dorigo (1991) [13] đã xây dựng thuật toán Hệ kiến (AS) giải bài toán người chào hàng. Hiệu quả của thuật toán so với các phương pháp mô phỏng tự nhiên khác như SA và GA đã được kiểm chứng bằng thực nghiệm. Thuật toán này về sau được phát triển và có nhiều ứng dụng phong phú, được gọi chung làphương pháp ACO.

2.3 Trình bày giải thuật

Khi áp dụng ACO cho các bài toán cụ thể, có bốn yếu tố quyết định hiệu quả của thuật toán:

- *Xây dựng đồ thị cấu trúc thích hợp*: Tùy thuộc vào đặc thù của bài toán
- *Xây dựng lời giải tuần tự:* Tùy thuộc vào đặc thù của bài toán
- *Chọn thông tin heuristic*: Thông tin heuristic tốt sẽ làm tăng hiệu quả của thuật toán. Tuy nhiên có nhiều bài toán không có thông tin này thì có thể đánh giá chúng như nhau.
- *Chọn quy tắc cập nhật mùi*: Quy tắc cập nhật mùi thể hiện chiến lược học của thuật toán. Hai yếu tố đầu: Đồ thị cấu trúc và thông tin heuristic phụ thuộc vào bài toán cụ thể, còn quy tắc cập nhật mùi là yếu tố phổ dụng và thường dùng làm tên để phân biệt cho các thuật toán ACO.

2.3.1 Đồ thị cấu trúc

Xét bài toán TƯTH tổng quát được nêu trong mục 1.2.1.1 dưới dạng bài toán cực tiểu hóa (S, f, Ω), trong đó S là tập hữu hạn trạng thái, f là hàm mục tiêu xác định trên S, còn Ω là các ràng buộc để xác định tập S có các thành phần được lấy từ tập hữu hạn C. Các tập S, C và Ω có các đặc tính sau:

1) Ký hiệu X là tập các dãy trong C độ dài không quá h: $X = \{< u_0, \ldots, u_k > : u_i \in C, \forall i \leq k \leq h\}$. Khi đó, mỗi phương án s trong S được xác định bởi ít nhất một vectơ trong X.

2) Tồn tại tập con X^*của X và ánh xạ φ từ X^* lên S sao cho $\varphi^{-1}(s)$ không rỗng với mọi $s \in S$, trong đó tập X^*có thể xây dựng được từ tập con C_0của C nhờ mở rộng tuần tự dưới đây.

3) Từ C_0 ta mở rộng tuần tự thành X^*như sau:

i) Ta xem $x_0 = < u_0 >$là mở rộng được với mọi $u_0 \in C_0$.

ii) Giả sử $x_k = < u_0, \ldots, u_k >$ là mở rộng được và chưa thuộc X^*. Từ tập ràng buộc Ω, xác định tập con $J(x_k)$ của C, sao cho với mọi $u_{k+1} \in J(x_k)$ thì $x_{k+1} = < u_0, \ldots, u_k, u_{k+1} >$ là mở rộng được.

iii) Áp dụng thủ tục mở rộng từ các phần tử $u_0 \in C_0$ cho phép ta xây dựng được mọi phần tử của X^*.

Xây dựng đồ thị cấu trúc

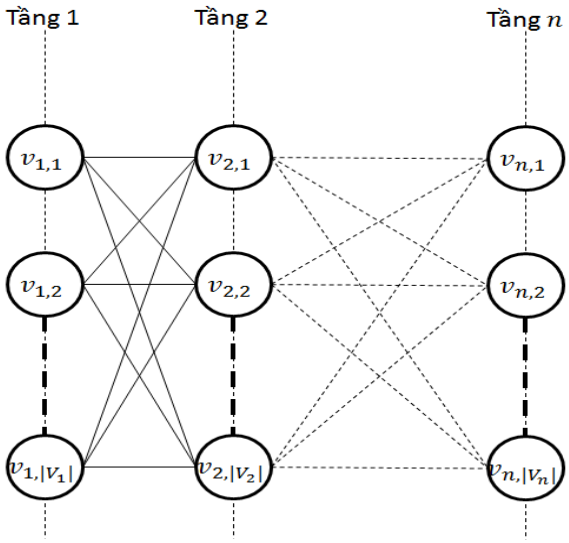
Mỗi bài toán TƯTH được xem như một bài toán tìm kiếm vectơ độ dài không quá h trên đồ thị đầy đủ có các đỉnh được gán nhãn trong tập C. Để tìm các lời giải chấp nhận được, ta xây dựng đồ thị đầy đủ với tập đỉnh V, mỗi đỉnh của nó tương ứng với mỗi thành phần của C. Các lời giải chấp nhận được sẽ là các vectơ được xác định theo thủ tục mở rộng tuần tự hay mở rộng ngẫu nhiên

Thông thường, đối với các bài toán thuộc loại NP-khó, người ta đưa ra các phương pháp heuristic tìm lời giải đủ tốt cho bài toán. Các thuật toán ACO kết hợp thông tin heuristic này với phương pháp học tăng cường, mô phỏng hành vi của đàn kiến, để tìm lời giải tốt hơn.

Ta gọi đồ thị $G = (V, E, H, \tau)$ là *đồ thị cấu trúc* của bài toán tối ưu tổ hợp, trong đó V là tập đỉnh, E là tập cạnh, H và τlà các thông tin gắn với cạnh. Từ các cạnh, xây dựng tập X^* nhờ mở rộng tập C_0 theo thủ tục tuần tự. Nếu không có thông tin heuristic thì ta xem H có các thành phần như nhau và bằng 1.

Trường hợp tổng quát, G là đồ thị đầy đủ.Tuy nhiên, tùy theo ràng buộc của bài toán, các cạnh có thể lược bớt để giảm miền tìm kiếm lời giải theo thủ tục mở rộng tuần tự. Chẳng hạn, với bài toán tìm cực trị của hàm giải tích $f(x_1, \ldots, x_n)$,với x_i thuộc tập giá trị hữu hạn V_i, đồ thị cấu trúc có n tầng, tầng i

chứa các đỉnh thuộc tập V_i, còn tập cạnh E chỉ gồm các cạnh nối các đỉnh thuộc tầng i với các đỉnh thuộc tầng $i+1$ $(i = 1,2,..,n-1)$ như trong hình 2.4. Khi đó tập C_0 là tập V_1, mỗi mở rộng tuần tự của lời giải sẽ được xây dựng từ một đỉnh thuộc tập này.



Hình 2.4: Đồ thị cấu trúc tổng quát cho bài toán cực trị hàm $\mathbf{f(x_1, \dots x_n)}$

2.3.2 Trình bày thuật toán ACO cơ bản

Tiến hành sử dụng m con kiến để xây dựng lời giải trên đồ thị cấu trúc. Quá trình tìm kiếm lời giải trên đồ thị kết thúc theo số bước lặp hoặc giới hạn thời gian chạy.

Xây dựng lời giải

Lời giải trên đồ thị cấu trúc $G = (V, E, H, \tau)$ như sau: Khởi tạo với m con kiến, tại mỗi lần lặp kiến sẽ chọn ngẫu nhiên một đỉnh để làm khởi tạo ban đầu $x_0 = < u_0 >$ với $u_0 \in C_0$. Sau đó các con kiến sẽ đi xây dựng lời giải theo thủ tục bước ngẫu nhiên.

Theo như trình bày ở trên điểm 3 phần iii mục 2.3.1. Từ đỉnh u_0 ta tiến hành mở rộng các đỉnh cho đến khi thuộc vào X^*, nghĩa là tìm được lời giải chấp nhận được. Giả sử con kiến đang ở đỉnh $i = u_k$ $(x_k = < u_0, \dots, u_k >)$ và có một đỉnh $j = u_{k+1}$ $(u_{k+1} \in J(x_k))$ để mở rộng (hay có thể hiểu con kiến từ đỉnh i sẽ lựa chọn đỉnh j) được chọn với xác suất như sau:

$$P(j) = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in J(x_k)}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} & j \in J(x_k) \\ 0 & j \notin J(x_k) \end{cases} \qquad (2.1)$$

Trong đó :

$\tau_{i,j}, \eta_{i,j}$: Giá trị thông tin mùi và thông tin heuristic.

α, β: Hai tham số quyết định sự ảnh hưởng tương quan giữa thông tin mùi và thông tin heuristic. Nếu $\alpha = 0$ không có học tăng cường. Nếu $\beta = 0$ chỉ có thông tin học tăng cường biểu thị qua vết mùi được sử dụng, không có thông tin heurisric.

l: Đỉnh lân cận của đỉnh i mà kiến có thể đi đến.

Cập nhật mùi

Dựa trên lời giải tìm được, đàn kiến sẽ thực hiện cập nhật mùi theo cách học tăng cường.

$$\tau_{i,j} \leftarrow (1-\rho)\tau_{i,j} + \Delta(i,j) \qquad (2.2)$$

Trong đó: ρ: hệ số bay hơi (tỷ lệ lượng mùi bị bay hơi), là hằng số thuộc khoảng (0,1).

$\Delta(i,j)$: lượng mùi do kiến để lại

Các bước thực hiện của thuật toán ACO được mô tả trong hình 2.5:

Procedure Thuật toán ACO;
Begin
Khởi tạo tham số, ma trận mùi, khởi tạo $\boldsymbol{m}$ con kiến;
repeat
    for $\boldsymbol{k = 1}$ **to** $\boldsymbol{m}$ **do**
        Kiến $\boldsymbol{k}$ xây dựng lời giải;
    end-for
    Cập nhật mùi;
    Cập nhật lời giải tốt nhất;
Until (Điều kiện kết thúc);
Đưa ra lời giải tốt nhất;
End;

Hình 2.5: Đặc tả thuật toán ACO

2.3.3 Thông tin Heuristic

Ta biết rằng thuật toán ACO mà không sử dụng tìm kiếm cục bộ, thông tin heuristic sẽ rất cần thiết để có được lời giải tốt. Trên thực tế, ở giai đoạn đầu vết mùi được khởi tạo như nhau. Khi đó vết mùi không thể giúp kiến tìm đường đi dẫn tới các lời giải tốt, vì chưa khác nhau nhiều. Vai trò chính của thông tin heuristic là để khắc phục điều này, giúp kiến có thể xây dựng được các hành trình tốt ngay trong giai đoạn đầu. Trong nhiều trường hợp, nhờ sử dụng tìm kiếm cục bộ, kiến vẫn có thể tìm được lời giải tốt ngay trong giai đoạn đầu, không cần sử dụng thông tin heuristic nào cả, mặc dù có làm cho quá trình tìm kiếm chậm hơn.

2.3.4 Quy tắc cập nhật vết mùi

Quy tắc cập nhật mùi thể hiện chiến lược học của thuật toán, với mỗi một quy tắc cập nhật mùi khác nhau thì tương ứng là các thuật toán khác nhau. Dưới đây sẽ giới thiệu một vài quy tắc cập nhật mùi khác nhau theo thứ tự thời gian xuất hiện.

2.3.4.1 Thuật toán AS

Đây là thuật toán ACO đầu tiên được Dorigo đề xuất năm 1991[12,13]. Vết mùi khởi tạo: $\tau_{ij} = \tau_0 = \frac{m}{C^{nn}}$ với m là số lượng kiến, C^{nn} độ dài lời giải tìm được của thuật toán heuristic .

Ban đầu tất cả các cạnh sẽ bị mất đi một lượng mùi do bay hơi:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} \qquad (2.3)$$

Sau đó những cạnh nào có kiến đi qua sẽ được cộng thêm một lượng mùi mà kiến để lại:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{m'=1}^{m} \Delta_{ij}^{m'} \qquad (2.4)$$

Trong đó:

$$\Delta_{ij}^{m'} \begin{cases} \frac{1}{C^{m'}} \ nếu\ cạnh\ (i,j) thuộc\ T^{m'} \\ 0\ ngược\ lại \end{cases} \qquad (2.5)$$

Trong đó: $C^{m'}$ là độ dài hành trình $T^{m'}$ do kiến m' xây dựng.

2.3.4.2 Thuật toán ACS

Thuật toán ACS (Ant Colony System) do Dorigo và Gambardella đề xuất năm 1997[11]. Trong thuật toán ACS gồm có hai quy tắc cập nhật mùi.

Cập nhật mùi toàn cục

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta_{ij} \qquad (2.6)$$

Trong đó:

$$\Delta_{ij} = \begin{cases} C_{gb}^{-1}, nếu\ (i,j) \in\ lời\ giải\ G-b \\ 0 \qquad\qquad\quad \overline{\in}\ lời\ giải\ G-b \end{cases} \qquad (2.7)$$

Cập nhật mùi cục bộ

$$\tau_{ij} \leftarrow (1-\gamma)\tau_{ij} + \gamma\Delta_{ij} \qquad (2.8)$$

Trong đó: $0 < \gamma < 1$ và $\Delta_{ij} = \tau_0 = (n.C_{nn})^{-1}$, n là số thành phố, L_{nn} là độ dài hành trình theo thuật toán tham ăn. Thuật toán ACS chỉ có duy nhất một kiến tìm được lời giải $G-b$ được phép để lại mùi sau mỗi lần lặp.

2.3.4.3 Thuật toán Max-Min

Thuật toán Max - Min (Max _ Min Ant System) được kí hiệu là MMAS được Stutzle và Hoos đề xuất năm 2000[3], với hàm cập nhật vết mùi như sau:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta_{ij}^{best} \qquad (2.9)$$

Trong đó: $\Delta_{ij}^{best} = 1/f(C^{best})$, với $f(C^{best}) = C^{gb}\ hoặc\ C^{ib}$

Vết mùi bị giới hạn trong đoạn $[\tau_{min}, \tau_{max}]$:

$$\tau_{ij} = \begin{cases} \tau_{max} & nếu\ \tau_{ij} > \tau_{max} \\ \tau_{ij} & \tau_{ij} \in [\tau_{min}, \tau_{max}] \\ \tau_{min} & nếu\ \tau_{ij} < \tau_{min} \end{cases} \qquad (2.10)$$

Trong MMAS việc sử dụng C^{gb} và C^{ib} ảnh hưởng tới hướng tìm kiếm và không xảy ra đồng thời mà thay phiên nhau. Khi luôn cập nhật bằng C^{gb} thì việc tìm kiếm định hướng xảy ra nhanh chóng. Khi cập nhật bằng C^{ib} thì việc tìm kiếm định hướng giảm do số lượng cạnh được cập nhật mùi nhiều.

Giới hạn vết mùi

Trong thuật toán MMAS sử dụng giới hạn trên τ_{max} và giới hạn dưới τ_{min} đối với vết mùi trên tất cả các cạnh nhằm mục đích cho thuật toán tránh khỏi

tình trạng tắc nghẽn. Giới hạn của vết mùi sẽ ảnh hưởng tới giới hạn xác suất p_{ij} trong việc lựa chọn đỉnh đi tiếp theo và bị giới hạn trong khoảng $[p_{min}, p_{max}]$.

2.3.4.4 Thuật toán Max- Min trơn

Thuật toán Max-Min trơn (Smooth _ Max Min Ant System) kí hiệu là SMMAS được Đỗ Đức Đông và Hoàng Xuân Huấn đề xuất năm 2012[2].

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta_{ij}$$

Với:

$$\Delta_{ij} = \begin{cases} \rho\tau_{min} \; nếu \; (i,j) \notin w(t) \\ \rho\tau_{max} \;\; nếu \; (i,j) \in w(t) \end{cases} \quad (2.11)$$

Chọn $\tau_0 = \tau_{max}$. Trong đó kí hiệu $w(t)$ là lời giải tốt nhất các kiến tìm được cho tới lần lặp thứ t. Nếu kí hiệu $w^i(t)$ hiểu là lời giải tốt nhất trong bước lặp thứ t. Nếu $w^i(t)$ không tốt hơn $w(t-1)$ ta có $w(t) = w(t-1)$, lúc này sẽ quan tâm tới lời giải gần đúng $w(t)$.

2.3.5 ACO kết hợp với tìm kiếm địa phương

Môt số tài liệu chỉ ra rằng với các phương pháp metaheuristic, một cách tiếp cận đầy hứa hẹn cho phép nhận được lời giải có chất lượng cao là kết hợp với thuật toán tìm kiếm địa phương.

Mô hình ACO có thể bao gồm cả tìm kiếm địa phương. Sau khi kiến xây dựng xong lời giải, có thể áp dụng tìm kiếm địa phương để nhận được lời giải tối ưu địa phương.Việc cập nhật mùi được thực hiện trên các cạnh thuộc lời giải tối ưu địa phương này. Kết hợp xây dựng lời giải với tìm kiếm địa phương sẽ là một cách tiếp cận có triển vọng, là do trên thực tế, cách xây dựng lời giải của ACO có sử dụng lân cận khác với tìm kiếm địa phương. Thực nghiệm cho thấy khả năng kết hợp tìm kiếm địa phương cải tiến được lời giải là khá cao.

2.3.6 Số lượng kiến

Như đã nói ở trên, nếu không sử dụng tìm kiếm địa phương và thông tin heuristic ít (hoặc không có), trong giai đoạn đầu vết mùi không thể giúp kiến tìm đường đi dẫn tới các lời giải tốt. Nếu sử dụng số lượng kiến ít, trong giai đoạn đầu sẽ không tìm được lời giải tốt và như vậy, việc cập nhật mùi được cập nhật dựa trên các lời giải không tốt. Khi đó, sẽ hướng việc tìm kiếm xung quanh lời giải không tốt và do đó thuật toán sẽ không hiệu quả. Có thể khắc phục phần nào

nhược điểm này bằng cách tăng số kiến, để tăng khả năng tìm được lời giải tốt ở mỗi vòng lặp. Khi có sử dụng tìm kiếm địa phương hoặc thông tin heuristic mạnh, sử dụng nhiều kiến là lãng phí.

2.3.7 Tham số bay hơi

Ở mỗi vòng lặp, khi xây dựng được lời giải tốt (sử dụng tìm kiếm địa phương hoặc thông tin heuristic mạnh), tham số bay hơi sẽ được xác lập có giá trị lớn, điều này giúp kiến quên đi những lời giải đã xây dựng, tập trung công việc tìm kiếm xung quanh lời giải tốt mới được xây dựng. Trong trường hợp ngược lại, ở mỗi vòng lặp, khả năng kiến tìm được lời giải tốt không cao thì tham số bay hơi phải được thiết lập với giá trị nhỏ.

CHƯƠNG 3: THUẬT TOÁN ĐỀ XUẤT

Ở chương 1, luận văn đã trình bày về bài toán (ℓ,d)-motif và một số cách tiếp cận để giải quyết bài toán, trong chương 3 luận văn đề xuất phương pháp tối ưu đàn kiến (ACO) để giải quyết bài toán.

3.1 Thuật toán tối ưu đàn kiến

Tối ưu đàn kiến (ACO) là một phương pháp metaheuristic dựa trên ý tưởng mô phỏng cách tìm đường đi từ tổ tới nguồn thức ăn của các con kiến tự nhiên. Phương pháp này đến nay đã có nhiều cải tiến và ứng dụng. Trong chương này luận văn giới thiệu thuật toán tối ưu đàn kiến (ACO) để giải quyết bài toán (ℓ,d)-motif. Trên cơ sở, cải tiến thuật toán tối ưu đàn kiến ACOMotif để áp dụng giải bài toán (ℓ,d) – motif

Thuật toán ACO như đã trình bày trong chương II gồm các công việc như sau:

+) Xây dựng đồ thị cấu trúc

+) Xây dựng lời giải tuần tự

+) Xác định thông tin Heuristic

+) Chọn quy tắc cập nhật mùi

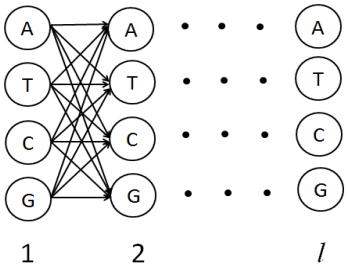
Trong đó việc xây đồ thị cấu trúc và xây dựng lời giải cho kiến thì tùy vào đặc thù của từng bài toán. Việc xác định thông tin Heuristic giúp làm tăng hiệu quả của thuật toán. Việc chọn quy tắc cập nhật mùi là áp dụng chung cho các bài toán, có nhiều quy tắc cập nhật mùi đã được đề xuất. Kỹ thuật tìm kiếm địa phương được áp dụng sau khi kiến xây dựng xong lời giải, để nhận được lời giải tối ưu địa phương. Trong luận văn này, chúng tôi đề xuất thuật toán (được đặt tên là F-ACOMotif) áp dụng phương pháp tối ưu đàn kiến để giải quyết bài toán (ℓ,d)-motif chúng tôi sử dụng đồ thị cấu trúc như MFACO[15], quy tắc cập nhật mùi như SMMAS, thông tin heuristic, lời giải tuần tự, kỹ thuật tìm kiếm địa phương như ACOMotif[19].

F-ACOMotif hoạt động theo lược đồ được mô tả trong hình 3.1, cho đầu ra là tập các motif có độ dài ℓ và các xâu con cùng độ dài trên các chuỗi DNA có khoảng cách hamming từ motif tới các chuỗi DNA nhỏ hơn hoặc bằng d (d là tham số cho trước) làm các instance. Các thành tố của F-ACOMotif như sau:

3.2. Xây dựng đồ thị cấu trúc

Đồ thị cấu trúc G(V, E) được dùng như MFACO [18]. Để tìm motif có độ dài ℓ, đồ thị có $4*\ell$ đỉnh được xếp thành 4 hàng và ℓ cột. Mỗi đỉnh tại vị trí (u, j) được gán nhãn của một loại nucleotide tương ứng như trong hình 2, nhãn của các đỉnh trong mỗi hàng cũng được dùng để chỉ hàng. Từ trái sang phải các cạnh sẽ nối từ đỉnh ở cột trước tới các đỉnh của cột sau. Ta ký hiệu $e_j(u, v)$ là cạnh nối đỉnh (u, j) với $(v, j+1)$.

Vết mùi và thông tin heuristics để ở các đỉnh của cột đầu và các cạnh.



Hình 3.1: Đồ thị cấu trúc tìm motif độ dài ℓ

3.3. Thông tin heuristic

Như đã trình bày ở mục 2.3.3, thông tin heuristic sẽ rất cần thiết để có được lời giải tốt. Trên thực tế, ở giai đoạn đầu vết mùi được khởi tạo như nhau. Khi đó vết mùi không thể giúp kiến tìm đường đi dẫn tới các lời giải tốt, vì chưa khác nhau nhiều. Vai trò chính của thông tin heuristic là để khắc phục điều này giúp kiến có thể xây dựng được các hành trình tốt ngay trong giai đoạn đầu.

Thông tin này gồm hai loại: ở các đỉnh của cột đầu và ở các cạnh.

- Ở các đỉnh của cột đầu, thông tin heuristics là tần số (frequency) xuất hiện nucleotide tương ứng trong tập dữ liệu *S*.
- Thông tin heuristics ở các cạnh $e_j(u, v)$ là tần số xuất hiện thành phần *uv* trong tập *S*. Chúng chỉ gồm 16 đại lượng $\eta_{u,v}$, $(u,v) \in \sum x \sum$

3.4. Xây dựng lời giải tuần tự

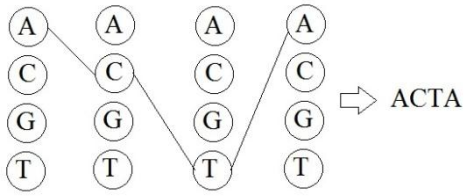
Trong mỗi lần lặp, mỗi con kiến chọn ngẫu nhiên một nút xuất phát *u* ở cột đầu với xác suất P_u^1

$$P_u^1 = \frac{\tau_u^1 * \eta_u}{\sum_{v \in \{A,C,G,T\}} \tau_v^1 * \eta_v} \quad (3.1)$$

Trong đó, η_u là thông tin heuristic được tính theo tần số của nucleotide *u* trong dữ liệu và τ_u^1 là vết mùi đã được cập nhật tại đỉnh. Ngoài ra, một con kiến di chuyền từ đỉnh *(u, j)* tới đỉnh *(v, j+1)* theo xác xuất sau:

$$P_{uv}^j = \frac{\tau_{uv}^j * \eta_{u,v}}{\sum_{r \in \{A,C,G,T\}} \tau_{ur}^j * \eta_{u,r}} \quad (3.2)$$

Trong đó, $\eta_{v,j}$ là thông tin heuristic của canh e_j(u, v). Chúng tôi đếm số lượng từng loại cặp base trong dữ liệu, từ đó tính ra được xác xuất xuất hiện của nó. Khi kiến di chuyên giữa 2 cột trong đồ thị, xác xuất đi từ đỉnh *(u, j)* tới *(v, j+1)* là xác xuất xuất hiện của cặp base (u,v) trong dữ liệu trong đó *u, v* $\in \{A, T, C, G\}$. Sau khi mỗi con kiến xây dựng xong đường đi và trước khi cập nhật mùi, kết quả sẽ được cải thiện bằng cách áp dụng thuật toán tìm kiếm địa phương.



Hình 3.2: Cách xây dựng đường đi của kiến

3.5. Quy tắc cập nhật mùi (pheromone update rule)

Sử dụng công thức cập nhật mùi mới SMMAS-Smooth Max-Min Ant System (H. Hoang Xuan 2012).

Các vết mùi τ_u^1 trên mỗi đỉnh *u* ở cột đầu và $\tau_{u,v}^j$ trên các cạnh $e_j(u, v)$ ban đầu được khởi tạo bằng τ_{max} cho trước. Sau mỗi vòng lặp, vết mùi τ_u^1 ở mỗi đỉnh u của cột đầu được cập nhật mùi theo Eq (3.3):

$$\tau_u^1 \leftarrow (1 - \rho)\tau_u^1 + \Delta_u^1, \quad (3.3)$$

Trong đó: $\Delta_u^1 = \begin{cases} \rho\tau_{max} & u \in \text{giải pháp tốt nhất} \\ \rho\tau_{min} & \text{giải pháp khác} \end{cases}$,

Trong đó τ_{max}, τ_{min} và ρ là các tham số chọn trước.

Vết mùi ở các cạnh $e_j(u, v)$ được cập nhật theo Eq (3.4)

$$\tau_{u,v}^j \leftarrow (1-\rho)\tau_{u,v}^j + \Delta_{u,v}^j, \quad (3.4)$$

Trong đó: $\Delta_{u,v}^j = \begin{cases} \rho\tau_{max} & uv \in \text{giải pháp tốt nhất} \\ \rho\tau_{min} & \text{giải pháp khác} \end{cases}$

3.6. Tìm kiếm địa phương (local search)

Sau khi các con kiến tìm được lời giải trong vòng lặp, các lời giải có hàm mục tiêu $\sum_{i=1}^{N} d_H(m, S_i)$ nhỏ nhất được áp dụng tìm kiếm địa phương bởi thủ tục lặp.

Thuật toán tìm kiếm địa phương được áp dụng trong chương trình là giải thuật leo đồi. Ta sẽ áp dụng kỹ thuật leo đồi như sau để tìm kiếm tăng cường.. Với mỗi motif tiềm năng (potemtial motif) S_m, dùng tập Q(S_m) để chứa kết quả tìm kiếm (), và thủ tục lặp này thực hiện như sau:

Bước 1: khởi tạo Q(S_m) = {S_m};

Bước 2. Thực hiện lặp:

For mỗi *i=1,…,l* thực hiện:

2.1. Thay ký tự (letter) ở vị trí thứ *i* của S_m lần lươt bởi một trong ba ký tự còn lại trong tập $\sum$ để có S_p;

2.2. Tính $H_d(S_p)$;

2.3. Nếu $H_d(S_p) \leq H_d(S_m)$ thì $S_m \leftarrow S_p$ và Q(S_m) = {S_p};

Until khi không thể cải thiện được hàm mục tiêu nữa.

Trong đó: $H_d(S_p)$ *là khoảng cách hamming của* S_p

Sau khi áp dụng tìm kiếm địa phương cho các motif tiềm năng trong mỗi lần lặp, các tập Q(S_m) có hàm mục tiêu nhỏ nhất hoặc gần nhỏ nhất được hợp lại thành tập Q các lời giải được xem là tốt nhất sau khi lọc các lời giải có cùng vị trí liên kết (chỉ giữ lại một motif). Dựa trên tập Q, các vết mùi trên đồ thị được cập nhật theo các Eq(3.3) và (3.4) để dùng cho vòng lặp kế tiếp.

Sau khi có tập Q là tập các motif có điểm khoảng cách hamming nhỏ nhất,

ta tiến hành kiểm tra các motif có $d_H(m,S_i)$ <=d thì ta in ra motif (ℓ,d).

Thuật toán dừng khi thực hiện xong số vòng lặp chọn trước. Các vị trí liên kết ứng với các motif trong Q cho ta xác định được instance của mottif. $d_H(m,S_i) = \min\{ d_H(m,m_i)$: m_i độ $dài\ \ell$ là $một\ chuỗi\ con\ của\ S_i$ }

Các xâu m_i cực tiểu (minimize) sẽ là instance của m và vị trí của nó trong S_i tương ứng sẽ là vị trí liên kết.

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM, SO SÁNH VÀ ĐÁNH GIÁ KẾT QUẢ

4.1 Bộ dữ liệu chuẩn

Để chạy thực nghiệm, luận văn sử dụng 13 bộ dữ liệu: trong đó 4 bộ dữ liệu là dữ liệu sinh học đã được công bố, được lấy từ bài báo [20]. Đây là bộ dữ liệu mà tác giả bài báo [16] sử dụng để chạy chương trình.

4.2 Tiến hành chạy thực nghiệm trên hệ điều hành ubuntu

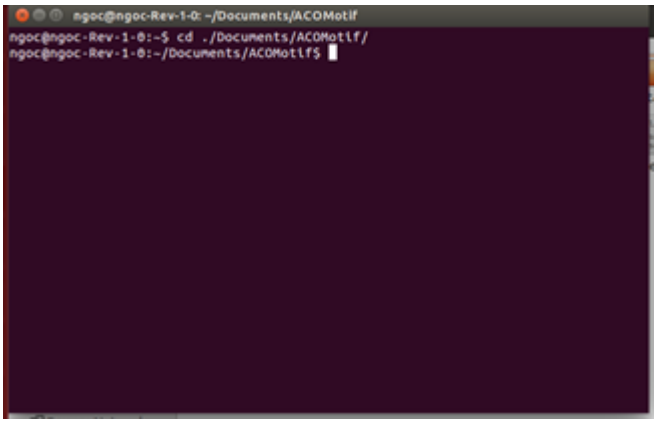
Chương trình được viết bằng ngôn ngữ Perl chạy trên máy Desktop cấu hình CPU intel core i5 2.5Ghz Ram 8GB, sử dụng hệ điều hành Ubuntu 12.04. Thực nghiệm so sánh hiệu quả thuật toán với Pairmotif+, MEME trên cùng các bộ dữ liệu, số kiến được dùng là 10.

Vì ACO là lớp thuật toán meta-heuristic nên mỗi lần chạy sẽ cho các kết quả cụ thể là khác nhau, nên tôi tiến hành chạy chương trình 10 lần trên mỗi bộ dữ liệu, thống kê kết quả thu được, so sánh thông qua tiêu trí là kết quả tốt nhất

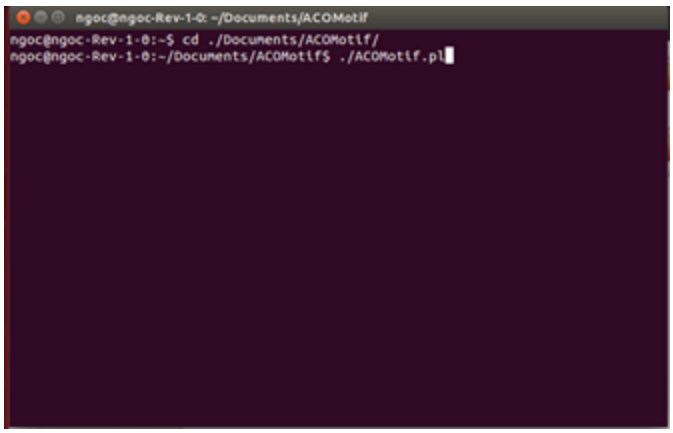
Hiệu quả nổi trội của F-ACOMotif được đánh giá bằng thực nghiệm trên cùng bộ dữ liệu đã được công bố của 2 thuật toán trên. Kết quả thực nghiệm cho thấy F-ACOMotif hiệu quả trong việc tìm ra Motif thực chính xác hơn so với 2 thuật toán Pairmotif+ và MEME.

Chạy chương trình F-ACOMotif:

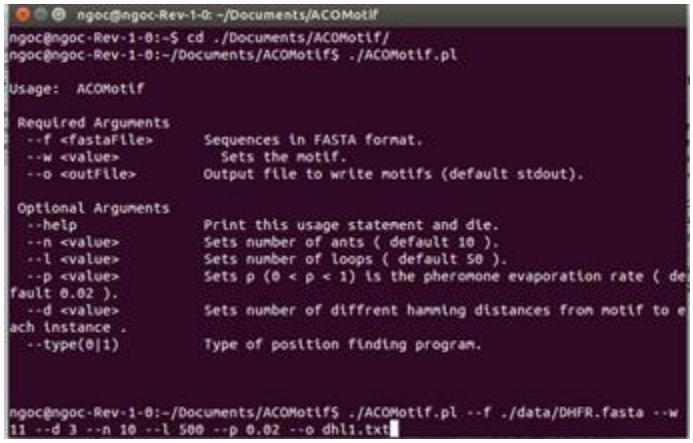
Bước 1: Mở hộp thoại Terminal (hoặc gõ Ctrl+Alt+T). Đưa con trỏ đến thư mục ACOMotif bằng cách: gõ lệnh **cd ./Docucment/ACOMotif**



Bước 2: gõ lệnh: ./ACOMotif.pl được kết quả như sau:



Bước 3: Chạy chương trình bằng cách gõ lệnh: ./ACOMotif.pl –f ./data/bộ dữ liệu –w (độ dài motif) –d (khoảng cách hamming) – n (số kiến) – l (số vòng lặp) – p (hệ số bay hơi) – o (output)



Bước 4: Kết quả chạy chương trình hiển thị trên file *.txt

4. 3 Kết quả chạy thực nghiệm và đánh giá

4.3.1 Kết quả thực nghiệm

Thực nghiệm F-ACOMotif trên tập dữ liệu Tompa. Dữ liệu Tompa được tải về theo địa chỉ sau: http://bio.cs.washington.edu/assessment/download.html

Dữ liệu Tompa bao gồm 52 tập dữ liệu từ cơ sở dữ liệu TRANSFAC và liên quan đến 4 loài: human (hm), mouse (ms), Drosophila melanogaster (dm) và Sacharomyces cerevisiae (yst). Luận văn tiến hành chạy thực nghiệm trên 4 tập dữ liệu, 2 tập dữ liệu trên loài mouse và 2 tập dữ liệu trên loài human. Luận văn chỉ lựa chọn những tập dữ liệu có số lượng chuỗi từ 4 đến 6 chuỗi, và có độ dài từ 501 nucleotide đến 1501 nucleotide.

Cụ thể, tập dữ liệu mus05 gồm 4 chuỗi, độ dài 501 nucleotide. Tập dữ liệu mus07 gồm 4 chuỗi, độ dài 1501 nucleotide. Tập dữ liệu hm19 gồm 5 chuỗi, độ dài 501 nucleotide. Tập dữ liệu hm22 gồm 6 chuỗi, độ dài 501 nucleotide.

Thực nghiệm chạy với 2 tham số $\ell = 21$ và $d = 8$ (các tham số ℓ, d được

lựa chọn theo dữ liệu thực), $\tau_{max} = 1.0$; $\tau_{min} = {}^{1}/_{4^l}$

Các tham số khác như sau:

n (số kiến)	(vòng lặp)	ρ(tham số bay hơi)
10	500	0.02

Bảng 4. 1: Các tham số chạy F-ACOMotif cho thực nghiệm

Mus 05	Position: 360 281 141 414 Motif : AGAGGTAAAAAAAAAGGAGAG Position: 360 281 141 414 Motif : AGAGGTAAAAAAAAAGGGGAG
Mus07	Position: 1402 1455 1343 336 Motif : CCCCCCCCCCAACACCTGCTG Position: 1239 701 99 647 Motif : TACACACACACACCCACACAC Position: 94 101 891 850 Motif : CTATGAGTCCAAAGCCAGCCT Position: 1239 701 99 647 Motif : TACAGACACACACACACACAC Position: 1402 1455 1343 336 Motif : CCACCCCCCCAACACCTGCTG
hm19	Position: 377 447 358 282 113 Motif : AGGGCGGGGCAGTGTGATGGG Position: 389 234 425 30 142 Motif : TGGGATGGGGCCGGGCGGGGG Position: 423 366 131 71 63 Motif : CTCTCCTCCCACCACCCACAG Position: 378 448 359 283 114 Motif : GGGCGGGGCACTGTGATGGGA Position: 389 234 425 30 142 Motif : TGGGATGCGGCCGGGTGGGGG Position: 389 234 425 30 142 Motif : TGGGATGCGGCCGGGCGGGGG Position: 389 234 425 30 142 Motif : TGGGATGGGGCGGGGCGGGGG Position: 377 447 358 282 113 Motif : AGGGCGGGGCACTGTGATGGG Position: 423 366 131 71 63 Motif : CTCTCCTCCCCCCACCCACAG Position: 389 234 425 30 142 Motif : TGGGATGCGGCGGGGTGGGGG Position: 389 234 425 30 142

	Motif : TGGGATGCGGCGGGGCGGGGG Position: 174 364 129 76 61 Motif : CCCCCTCCTCCCACCACCCAC Position: 174 364 129 76 61 Motif : CCCTCTCCTCCCACCACCCAC Position: 378 448 359 283 114 Motif : GGGCGGGGCAGTGTGATGGGA
hm22	Position: 20 83 306 199 384 131 Motif : GACAGAGGGCGGGTCCCTCCC Position: 370 404 77 473 159 54 Motif : AGGCAGGAAGGAGAAGGGAGG Position: 371 405 78 474 160 55 Motif : GGCAGGAAGGAGAAGGGAGGG Position: 370 404 77 473 159 54 Motif : AGGCAGGAATGAGAAGGGAGG Position: 121 184 124 186 34 122 Motif : GGGACACTGCAGAGCCTGGGG Position: 122 185 125 366 35 123 Motif : GGGCACGGCAGAGCCTGGGGA Position: 371 405 78 474 160 55 Motif : GGCAGGAATGAGAAGGGAGGG Position: 122 185 125 366 35 123 Motif : GGACACGGCAGAGCCTGGGGA Position: 122 185 125 366 35 123 Motif : GGCCACGGCAGAGCCTGGGGA Position: 121 184 124 186 34 122 Motif : TGGACACTGCAGAGCCTGGGG Position: 121 184 124 186 34 122 Motif : AGGACACTGCAGAGCCTGGGG

Bảng 4. 2: Kết quả thực nghiệm trên cơ sở dữ liệu TRANSFAC

Nhận xét:

Từ kết quả thực nghiệm cho thấy, F-ACOMotif cho kết quả là một tập các motif và một tập vị trí các thể hiện của motif. Ở đây luận văn không in ra danh sách các thể hiện mà chỉ in ra vị trí của các thể hiện, vì quá nhiều thể hiện, nếu in ra các thể hiện sẽ rất rối.

4.3.2 So sánh và đánh giá

4.3.2.1 So sánh với MEME

MEME là một chương trình mã nguồn mở, MEME tìm ra motif trong khoảng thời gian rất ngắn, ở đây luận văn chỉ so sánh F-ACOMotif với MEME

về độ chính xác khi tìm ra motif.

Thực nghiệm trên tập dữ liệu lấy từ [20] gồm 5 tập dữ liệu với số lượng các tham số d lần lượt là 2, 4, 5, 6, 7. Độ dài motif lần lượt là 9, 15, 18, 21, 24. Với số lượng các chuỗi là 20, mỗi chuỗi có độ dài là 601 nucleotide. Tập dữ liệu này chưa có công bố motif thực được thực nghiệm trong sinh học. Do đó, thực nghiệm chỉ so sánh kết quả.

Mỗi tập dữ liệu con được đặt tên tương ứng với 2 tham số ℓ và d lần lượt là (9,2); (15,4); (18,5); (21,6); (24,7) được lựa chọn theo dữ liệu đã cho, mỗi tập dữ liệu con được chạy trên MEME và F-ACOMotif 10 lần. Các tham số chạy F-ACOMotif lần lượt như sau:

n (số kiến)	(vòng lặp)	ρ(tham số bay hơi)
10	500	0.004

Bảng 4.3: Tham số chạy F-ACOMotif

$\tau_{max} = 1.0;\ \tau_{min} = {}^{1}/_{4^{l}}$

MEME được tải về theo địa chỉ:
"http://meme-suite.org/doc/download.html?man_type=web"
Tiến hành chạy thực nghiệm trên MEME với các tham số (ℓ,d) tương tự như chạy trên F-ACOMotif.

Ta nhận được kết quả như sau:

(ℓ,d)	MEME	F-ACOMotif
(9,2)	GTTCAGCGT	GTTCAGCGT
(15,4)	AGCGAGCCTTTACAA	ATCGAGCTTTGACAA
(18,5)	AGTGAAAGACTTGTACCT	AGTGAAAGACTTGTACCT
(21,6)	GCGCGACGGACTTACGTCTTC	GCGCGACGGACTTACGTCTTC
(24,7)	AATTACTTTTCGATAAAGTGGATC	AATTACTTTCCGATAAAGTGGATC

Bảng 4.4: Kết quả so sánh F-ACOMotif với thuật toán MEME

Nhận xét:

Từ bảng so sánh kết quả, ta nhận thấy rằng với các tham số (ℓ,d) lần lượt là: (9,2); (18,5); (21,6); (24,7) thì F-ACOMotif và MEME kết quả gần giống nhau chỉ khác kết quả duy nhất ở 1 tham số là (15,4) tuy nhiên không lớn lắm.

Do đó, ta có thể kết luận F-ACOMotif tìm được motif chính xác tương đương MEME.

4.3.2.2 Kết quả so sánh F-ACOMotif với Pairmotif+ và MEME trên tập dữ liệu thực

Trong phần này thực nghiệm so sánh F-ACOMotif với 2 thuật toán PairMotif+ và MEME trên tập dữ liệu thực sử dụng rộng dãi, bao gồm DHFR với số lượng chuỗi là 4 chuỗi, chuỗi ngắn nhất có độ dài 1345 nucleotide, chuỗi dài nhất có độ dài 1955 nucleotide. Preproinsulin với số lượng chuỗi là 4, chuỗi dài nhất có độ dài 2853 nucleotide, chuỗi ngắn nhất có độ dài 1473 nucleotide. Metallothionein với số lượng chuỗi là 4 chuỗi, chuỗi ngắn nhất có độ dài 1148 nucleotide, chuỗi dài nhất có độ dài 8352 nucleotide. Và Yeast ECB với số lượng chuỗi là 5 chuỗi, chuỗi dài nhất có độ dài 1051 nucleotide, chuỗi ngắn nhất có độ dài 251 nucleotide.

Mỗi tập dữ liệu này tương ứng với một (ℓ, *d*), bởi vì mỗi chuỗi chứa một instance motif và tất cả các instances motif có cùng độ dài. Mục đích thực nghiệm trên các tập dữ liệu này là để kiểm tra xem các thuật toán đề xuất có thể tìm thấy vị trí liên kết các yếu tố phiên mã biết cách sử dụng (ℓ,d) cụ thể, ℓ là chiều dài của motif công bố và d là khoảng cách hamming.

Data	(ℓ,d)	Pairmotif+	MEME	F-ACOMotif	Motif công bố
DHFR	(11, 3)	GCGCCAAACTT	-	ATTTCGCGCCA	ATTTCGCGCCA
Preproinsulin	(15, 4)	TGCAACCTCAGCCCC	-	CAGACCCAGCACCAG	CAGCCTCAGCCCCCA
Metallothionein	(15, 4)	CTCTGCACCCGGCCC	-	CTCTGCACCCGGCCC	CTCTGCACRCCGCCC
Yeast ECB	(16, 5)	TTACCCAGTAAGGAAA	TTTCCCGTTTAGGAAA	TTTCCCGTTTAGGAAA	TTTCCCNNTNAGGAAA

Bảng 4.5: Kết quả so sánh F-ACOMotif với MEME và PairMotif+

Nhận xét:

Từ bảng kết quả so sánh F-ACOMotif với MEME và PairMotif+ ta nhận thấy: MEME tìm ra motif với thời gian rất ngắn. Nhưng hạn chế của MEME là với những chuỗi đầu vào có độ dài quá lớn, MEME không tìm được motif.

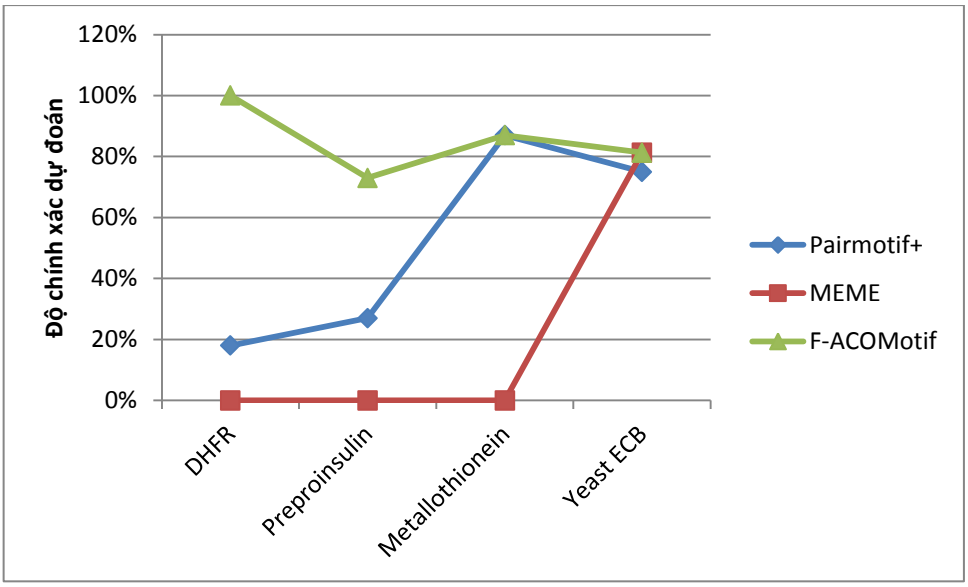
Từ bảng kết quả so sánh F-ACOMotif với MEME và PairMotif+ ta tiến hành lập bảng so sánh độ chính xác của motif dự đoán:

Data	(ℓ,d)	Pairmotif+	MEME	F-ACOMotif
DHFR	(11, 3)	18%	0%	100%
Preproinsulin	(15, 4)	27%	0%	73%
Metallothionein	(15, 4)	87%	0%	87%
Yeast ECB	(16, 5)	75%	81.25%	81.25%

Bảng 4.6: So sánh độ chính xác của motif dự đoán

Nhận xét:

Từ bảng so sánh độ chính xác của motif dự đoán ta nhận thấy rằng F-ACOMotif dự đoán motif chính xác hơn so với MEME và Pairmotif+.



Hình 4.1: Đồ thị so sánh độ chính xác của F-ACOMotif so với PairMotif+ và MEME

Nhận xét:

Từ kết quả so sánh F-ACOMotif với MEME và PairMotif+ có thể thấy rằng F-ACOMotif hiệu quả hơn thuật toán MEME và PairMotif+ về độ chính xác khi tìm ra Motif so với motif thực.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

KẾT LUẬN

Bài toán tìm kiếm (ℓ,d) motif là một bài toán có ý nghĩa trong tin sinh học, nó đóng vai trò quan trọng trong việc xác định vị trí liên kết trong quá trình phiên mã trong chuỗi DNA. Xác định được các Motif và các thể hiện tương ứng của nó có ý nghĩa rất quan trọng, từ đó các nhà nghiên cứu sinh học có thể phát hiện ra các tương tác giữa DNA và Protein, điều hòa gen cũng như sự phát triển và tương tác trong một tế bào.

Trong luận văn này, chúng tôi đã dựa trên ý tưởng của thuật toán ACOMotif đề xuất thuật toán mới là F-ACOMotif để giải quyết bài toán (ℓ,d) motif.

So sánh thực nghiệm với thuật toán MEME và PairMotif+, cho thấy thuật toán F-ACOMotif cho kết quả tốt hơn khi tìm ra motif với độ chính xác cao so với motif thực được công bố trong thực nghiệm sinh học.

HƯỚNG PHÁT TRIỂN

Luận văn đề xuất thuật toán ACO để giải quyết bài toán tìm kiếm (ℓ,d) motif và cho lời giải tốt. Tuy nhiên, thời gian chạy thuật toán để cho lời giải tốt còn chậm. Và F-ACOMotif chỉ cho hiệu quả đối với các tập dữ liệu với số chuỗi đầu vào nhỏ hơn 10. Trong tương lai sẽ nghiên cứu cải tiến bài toán tìm kiếm (ℓ,d) motif với thời gian thực hiện ngắn và độ chính xác so với motif thực sẽ cao hơn.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Hồ Tú Bảo, Phạm Thọ Hoàn, School of Knowledge Science Japan Advanced Institute of Science and technology, Tin sinh học khái niệm và bài toán cơ bản Một vài kết quả nghiên cứu, 2005

[2] Đỗ Đức Đông (2012), Phương pháp tối ưu đàn kiến và ứng dụng, Luận án tiến sĩ công nghệ thông tin ĐHCN – ĐHQGHN.

Tiếng Anh

[3] [SH00] T. St¨utzle and H.H. Hoos. "*MAX-MIN Ant System"*. Journal of FutureGeneration Computer Systems, special issue on Ant Algorithms, 16:889–914, 2000.

[4] Bailey TL, Williams N, Misleh C. *et al*. "*MEME: discovering and analyzing DNA and protein sequence motifs". Nucleic Acids Res.* 2006; 34: 369-73

[5] Buhler J, Tompa M. "*Finding motifs using random projections***"**. *J* Comput Biol. 2002; 9:225-42

[6] Cheng-Hong Yang, Member, IAENG, Yu-Tang Liu, and Li-Yeh Chuang (2011) "*DNA Motif Discovery Based on Ant Colony Optimization and Expectation Maximization".* Proceedings of the International MultiConference of Engineer, and Computer Scientists 2011 Vol I, IMECS 2011, March 16 – 18, 2011, Hong Kong.

[7] E. Alpaydın (2010), "*Introduction to Machine Learning",* Massachusetts Institute of Technology, Second Edition.

[8] H. Dinh, S. Rajasekaran, and J. Davila. *qPMS7: "A Fast Algorithm for Finding (ℓ, d)-Motifs in DNA and Protein Sequences",* PloS one , Vol.7. No. 7 (2012): e41425.

[9] J. Liu, A. Neuwald, and C. Lawrence. "*Bayesian models for multiple local sequence alignment and Gibbs sampling strategies*." Journal of the American Statistical Association, 90(432):1156–1170, 1995.

[10] M. Dorigo (1992), "*Optimization, learning and natural algorithms"*, PhD. dissertation, Milan Polytechnique, Italy.

[11] M. Dorigo and L.M. Gambardella (1997)*, "*Ant colony system: A cooperative learning approach to the traveling salesman problem", *IEEE Trans. on evolutionary computation*, Vol 1 (1), pp. 53-66.

[12] M. Dorigo, and T.Stützle (2004), "*Ant Colony Optimization*," The MIT

Press, Cambridge, Masachusetts.

[13] M. Dorigo, V. Maniezzo and A. Colorni (1991), "*The Ant System: An autocatalytic optimizing process"*, Technical Report 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Milano, Italy.

[14] Neil C. Jones, and Pavel A. Pevzner "*An introduction to bioinformatics algorithms"*. MIT press, 2004.

[15] Pradhan, Medha, "*Motif Discovery in Biological Sequences*" (2008). Master's Projects

[16] Qiang Yu, Hongwei Huo, Yipu Zhang, Hongzhi Guo, and Haitao Guo. "PairMotif+*: a fast and effective algorithm for de novo motif discovery in DNA sequences*." International journal of biological sciences 9, no. 4 (2013): 412.

[17] Rajasekaran S. "*Computational techniques for motif search"*. Frontiers in Bioscience. 2009;14:5052–5065. doi: 10.2741/3586

[18] S. Bouamama, A. Boukerram, and A.F. Al Badarneh: "*Motif Finding Using Ant Colony Optimization",* ANTS'10 Proc. of the 7th int. conf. on Swarm intelligence (2010), LNCS vol.6234, 464-471.

[19] Xuan-Huan Hoang and T.A. Tuyet Duong and T.T. Ha Doan and T. Hung Nguyen (2014) "*An Efficient Ant Colony Algorithm for DNA Motif Finding"*. In: 2014: The 6th International Conference on Knowledge and Systems Engineering (KSE 2014), 9-11 October 2014, Hanoi, Vietnam.

[20] Yu Q, Huo H, Zhang Y, Guo H (2012) *"PairMotif: A New Pattern-Driven Algorithm for Planted (l,d) DNA Motif Search."* PLoS ONE 7(10): e48442.doi:10.1371/journal.pone.0048442