ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

NGUYỄN HỮU ĐÔNG

NGHIÊN CỨU THUẬT TOÁN TABU SEARCH VÀ ỨNG DỤNG VÀO BÀI TOÁN NGƯỜI DU LỊCH

LUẬN VĂN THẠC SĨ KHOA HỌC MÁY TÍNH

THÁI NGUYÊN - NĂM 2014

LỜI CẢM ƠN

Để hoàn thành luận văn tốt nghiệp "*Nghiên cứu thuật toán Tabu Search và ứng dụng vào bài toán người du lịch*" lời đầu tiên tôi xin gửi cám ơn sâu sắc nhất tới GS.TS. Vũ Đức Thi đã hướng dẫn và chỉ bảo tôi tận tình trong suốt thời gian làm khóa luận.

Tôi xin chân thành cảm ơn các thầy cô giáo Trường Đại học Công nghệ thông tin và Truyền thông Thái Nguyên, các giảng viên đã truyền đạt những kiến thức, kỹ năng, kinh nghiệm nghề nghiệp..

Tôi xin chân thành cám ơn Ban giám hiệu, tập thể giáo viên khoa Điện tử - Tin học Trường Cao đẳng nghề Cơ điện Phú Thọ, gia đình cùng các bạn trong lớp cao học Khoa học máy tính khóa 2012-2014 đã tạo mọi điều kiện giúp đỡ, động viên, chia sẻ để tôi hoàn thành bản luận văn này.

Bản luận văn chắc còn nhiều thiết sót, rất mong được các thầy cô giáo trong hội đồng chấm luận văn xem xét, góp ý kiến để luận văn được hoàn thiện hơn.

Tôi xin chân thành cảm ơn!

Thái Nguyên, tháng 9 năm 2014

HỌC VIÊN

Nguyễn Hữu Đông

LỜI CAM ĐOAN

Với mục đích học tập, nghiên cứu để nâng cao trình độ chuyên môn nên tôi đã làm luận văn này một cách nghiêm túc và hoàn toàn trung thực.

Trong luận văn, tôi có sử dụng tài liệu tham khảo của một số tác giả, tôi đã nêu trong phần tài liệu tham khảo ở cuối luận văn.

Tôi xin cam đoan và chịu trách nhiệm về nội dung, sự trung thực trong luận văn tốt nghiệp Thạc sĩ của mình.

Thái Nguyên, tháng 09 năm 2014

HỌC VIÊN

Nguyễn Hữu Đông

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU, CÁC TỪ VIẾT TẮT

Từ viết tắt	Từ đầy đủ	Giải thích
AI	**A**rtificial **I**ntelligent	Trí tuệ nhân tạo
BFS	**B**readth **F**irst **S**earch	Tìm kiếm theo chiều rộng
DFS	**D**epth **F**irst **S**earch	Tìm kiếm theo chiều sâu
CNTT	**C**ông **n**ghệ **T**hông **t**in	Công nghệ Thông tin
CNPM	**C**ông **n**ghệ **P**hần **m**ềm	Công nghệ Phần mềm
GA	**G**enetic **A**lgorithms	Giải thuật Di truyền
LNS	**L**arge **N**eighborhood **S**earch	Tìm kiếm Lân cận lớn
LS	**L**ocal **S**earch	Tìm kiếm Cục bộ
LTM	**L**ong **T**erm **M**emory	Bộ nhớ dài hạn
SA	**S**imulated **A**nnealing	Luyện thép
STM	**S**hort **T**erm **M**emory	Bộ nhớ ngắn hạn
TS	**T**abu **S**earch	Tìm kiếm Tabu
TTNT	**T**rí **t**uệ **N**hân **t**ạo	Trí tuệ Nhân tạo
TSP	**T**ravelling **S**alesman **P**roblem	Bài toán người du lịch
OR	**O**peration **R**esarch	Nghiên cứu tối ưu

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

MỞ ĐẦU

1. Lý do chọn đề tài

Lớp các bài toán tối ưu hóa tổ hợp xuất hiện trong nhiều lĩnh vực quan trọng trong cuộc sống: Tin học, tài chính, lập lịch, sản xuất...và lớp bài toán có nhiều ứng dụng trên thực tế, một số bài toán kinh điển trong các bài toán này: Bài toán người du lịch, bài toán n – queens, bài toán tô màu đồ thị, bài toán xếp lịch trực y tá, bài toán tìm tập phủ đỉnh của đồ thị.....

Lớp các bài toán tối ưu tổ hợp thường các tập không gian trạng thái lớn mà không thể sử dụng các phương pháp tìm kiếm thông thường để xem xét tất cả không gian trạng thái. Tìm kiếm cục bộ được thiết kế cho bài toán tìm kiếm với không gian trạng thái rất lớn và cho phép tìm kiếm trạng thái tương đối tốt với thời gian tìm kiếm chấp nhận được. Tuy nhiên phương pháp tìm kiếm cục bộ vẫn còn một số nhược điểm: Thời gian giải quyết các bài toán có thể vẫn còn dài, thuật toán có thể không tìm ra lời giải tốt nhất trong một lần chạy...

Thuật toán tìm kiếm Tabu được cải tiến từ phương pháp tìm kiếm cục bộ. Bằng kết quả thực nghiệm đã cho thấy kỹ thuật tìm kiếm Tabu có thể giải quyết hiệu quả các bài toán tối ưu.

Trong khuôn khổ của khóa luận, đề tài tập trung tìm hiểu các nguyên lý chung và nền tảng của tìm kiến Tabu, áp dụng giải thuận này để giải quyết bài toán người du lịch, từ đó đánh giá hiệu quả của giải thuật này so với một số giải thuật khác.

2. Mục tiêu nghiên cứu

- Tìm hiểu các giải thuật tìm kiếm cục bộ cho các bài toán tối ưu hóa tổ hợp

- Nghiên cứu giải thuật tìm kiếm Tabu: Nguyên lý chung của tìm kiếm Tabu, cách sử dụng bộ nhớ, nền tảng của tìm kiếm Tabu.
- Sử dụng phương pháp tìm kiếm Tabu để giải quyết bài toán người du lịch, đánh giá được hiệu quả của giải thuật này so với một số giải thuật tìm kiếm khác

3. Đối tượng và phạm vi nghiên cứu

Nghiêm cứu tìm hiểu lý thuyết và thuật toán Tabu Search từ đó sử dụng thuật toán này để giải quyết bài toán người du lịch, sau đó đánh giá được hiệu quả của thuật toán này đem lại so với một số thuật toán tìm kiếm khác.

4. Hướng nghiên cứu của đề tài

- Tìm hiểu các thuật toán tìm kiếm cục bộ cho các bài toán tối ưu hóa tổ hợp
- Nghiên cứu thuật toán Tabu Search: Nguyên lý chung của tìm kiếm Tabu, cách sử dụng bộ nhớ, nền tảng của tìm kiếm Tabu.
- Sử dụng phương pháp tìm kiếm Tabu để giải quyết bài toán người du lịch, đánh giá được hiệu quả của thuật toán này so với một số thuật toán tìm kiếm khác.

5. Ý nghĩa khoa học của đề tài

Nghiên cứu thuật toán tìm kiếm Tabu: Nguyên lý chung của tìm kiếm Tabu, cách sử dụng bộ nhớ, nền tảng của tìm kiếm Tabu.

Kết quả đạt được của đề tài có thể được ứng dụng để giải quyết vào bài toán người du lịch.

6. Phương pháp nghiên cứu

- Nghiên cứu tài liệu khoa học về tổng quan các thuật toán tìm kiếm cục bộ.
- Nghiên cứu tài liệu khoa học về các phương pháp tìm kiếm cục bộ.
- Nghiên cứu lý thuyết về thuật toán tìm kiếm Tabu.

- Sử dụng thuật toán tìm kiếm Tabu cài đặt cho bài toán người du lịch.
- Đánh giá hiệu quả của thuật toán này so với một số thuật toán khác.

CHƯƠNG1: TỔNG QUAN VỀ TÌM KIẾM

1.1. Giải quyết vấn đề bằng tìm kiếm

Tìm kiếm là một trong những hướng nghiên cứu quan trọng trong CNTT. Trong thực tế, nhiều bài toán có thể đưa về bài toán tìm kiếm, ví dụ:

+ Trò chơi: Nhiều trò chơi, ví dụ cờ vua, thực chất là quá trình tìm kiếm nước đi của các bên trong số những nước mà luật chơi cho phép, để giành lấy ưu thế cho mình.

+ Lập thời khóa biểu: Lập thời khóa biểu là lựa chọn thứ tự, thời gian, tài nguyên (máy móc, địa điểm, con người) thỏa mãn một tiêu chí nào đó. Như vậy, lập thời khóa biểu có thể coi như quá trình tìm tiếm trong số tổ hợp phương án sắp xếp phương án đáp ứng yêu cầu đề ra.

+ Tìm đường đi: Trong số những đường đi, lựa chọn đường đi tới đích, có thể thỏa mãn một số tiêu chí nào đó như tiêu chí tối ưu về độ dài, thời gian, giá thành….

+ Lập kế hoạch: Là lựa chọn chuỗi hành động cơ sở cho phép đạt mục tiêu đề ra đồng thời thỏa mãn các yêu cầu phụ.

Sự phổ biến của các vấn đề có tích chất tìm kiếm dẫn tới yêu cầu phát biểu bài toán tìm kiếm một cách tổng quát, đồng thời xây dựng phương pháp giải bài toán tìm kiếm sao cho hiệu quả, thuận lợi. Do vậy, tìm kiếm đã được nghiên cứu trong khuôn khổ toán rời rạc, lý thuyết thuật toán. Trong TTNT, tìm kiếm được đặc biệt quan tâm từ khía cạnh xây dựng phương pháp cho

phép tìm ra kết quả trong trường hợp không gian tìm kiếm có kích thước lớn khiến cho những phương pháp truyền thống gặp khó khăn.

1.2. Bài toán tìm kiếm trong không gian trạng thái

Một bài toán có thể giải quyết thông qua tìm kiếm bằng cách xác định tập hợp tất cả các phương án, đối tượng hay trạng thái liên quan gọi chung là không gian trạng thái. Thủ tục tìm kiếm sau đó sẽ khảo sát không gian trạng thái theo một cách nào đó để tìm ra lời giải cho vấn đề. Tùy vào cách thức khảo sát không gian trạng thái cụ thể, ta sẽ có những phương pháp tìm kiếm khác nhau.

Để có thể khảo sát không gian trạng thái, thuật toán tìm kiếm bắt đầu từ một trạng thái xuất phát nào đó, sau đó sử dụng những phép biến đổi trạng thái để nhận biết và chuyển sang trạng thái khác. Quá trình tìm kiếm kết thúc khi tìm ra lời giải, tức là khi đạt tới trạng thái đích.

Bài toán tìm kiếm về cơ bản có thể phát biểu thông qua năm thành phần chính sau [2]:

+ Tập các trạng thái *Q* đây chính là không gian trạng thái của bài toán.

+ Tập (không rỗng) các trạng thái xuất phát *S* ($S \subseteq Q$). Thuật toán tìm kiếm sẽ xuất phát từ một trong những trạng thái này để khảo sát không gian tìm kiếm.

+ Tập (không rỗng) các trạng thái đích *G* ($G \subseteq Q$). Trạng thái đích có thể được cho một cách tường minh, tức là chỉ ra cụ thể đó là trạng thái nào hoặc không tường minh. Trong trường hợp sau, thay về trạng thái cụ thể bài toán sẽ quy định một số điều kiện mà trạng thái đích cần thỏa mãn. Ví dụ, khi chơi cờ vua, thay vì chỉ ra vị trí cụ thể quân cờ, ta chỉ ra quy tắc cho biết trạng thái chiếu hết.

+ Các toán tử còn gọi là hành động hay chuyển động, thực chất đây là ánh xạ *P: Q →Q*, cho phép chuyển từ trạng thái hiện thời sang các trạng thái khác. Với mỗi trạng thái *n, P(n)* là tập các trạng thái được sinh ra khi áp dụng toán tử hay chuyển động *P*.

+ Giá thành *c: Q x Q → R*. Trong một số trường hợp, quá trình tìm kiếm cần quan tâm tới giá thành đường đi. Giá thành để di chuyển từ nút *x* tới nút hàng xóm *y* được cho dưới dạng số dương *c(x,y)*.

Hiệu quả của việc tìm kiếm thể hiện qua việc đánh giá theo 4 tiêu chuẩn:

+ Độ phức tạp tính toán: Được xác định bằng khối lượng tính toán cần thực hiện để tìm ra lới giải. Thông thường, khối lượng tính toán được xác định bằng số lượng trạng thái cần xem xét trong suốt quá trình tìm ra lời giải.

+ Bộ nhớ: Được xác định bằng số lượng trạng thái cần lưu trữ khi thực hiện thuật toán.

+ Tính đầy đủ: Nếu bài toán có lời giải thì thuật toán có khả năng tìm ra lời giải đó không? Nếu có, ta nói rằng thuật toán có tính đầy đủ, trong trường hợp ngược lại ta nói thuật toán không có tính đầy đủ.

+ Tính tối ưu: Nếu bài toán có nhiều lời giải thì thuật toán có cho phép tìm ra lời giải tốt nhất không? Nếu không, ta nói lời giải đảm bảo tính tối ưu.

1.3. Các kĩ thuật tìm kiếm cơ bản

Ý tưởng của thuật toán tìm kiếm: Xem xét trạng thái, sử dụng các hàm biến đổi trạng thái để di chuyển trong không gian trạng thái cho tới khi đạt đến trạng thái mong muốn.

Thuật toán tìm kiếm tổng phát sinh ra một cây tìm kiếm, trong đó mỗi trạng thái tương ứng với một nút trên cây. Trạng thái xuất phát tương ứng với gốc cây, những trạng thái được mở rộng tạo thành các nút thế hệ tiếp theo.

Trên thực tế, việc di chuyển trong không gian trạng thái sẽ dẫn tới những nút đã duyệt qua và tạo thành vòng lặp. Trong trường hợp như vậy, cây tìm kiếm có thể là vô tận và cần có cách kiểm tra để không xem xét lại nút đã duyệt.

Các kỹ thuật tìm kiếm được áp dụng rộng rãi hiện nay:

- Tìm kiếm không có thông tin
- Tìm kiếm có thông tin

1.3.1. Tìm kiếm không có thông tin

Tìm kiếm không có thông tin (hay tìm kiếm mù) là tìm kiếm không có hiểu biết gì về các đối tượng để hướng dẫn tìm kiếm, không có sự hướng dẫn nào cho tìm kiếm, chỉ phát triển các trạng thái từ trạng thái ban đầu cho tới khi gặp một trạng thái đích nào đó, nhược điểm của các giải thuật này là phần lớn các không gian tìm kiếm có kích thước cực kì lớn và một quá trình tìm kiếm (đặc biệt tìm kiếm theo cây) sẽ cần một khoảng thời gian đáng kể cho các ví dụ nhỏ. Một số dạng tìm kiếm không có thông tin nổi bật ứng với các cách tổ chức dữ liệu:

1.3.1.1. Tìm kiếm trên danh sách

Các giải thuật tìm kiếm trên danh sách [1] là loại giải thuật tìm kiếm cơ bản nhất. Mục đích là tìm trong một tập hợp một phần tử chứa một khóa nào đó. Các giải thuật tìm kiếm tiêu biểu nhất trên danh sách là: Tìm kiếm tuần tự (hay tìm kiếm tuyến tính), tìm kiếm nhị phân.

Tìm kiếm tuần tự kiểm tra từng phần tử trong danh sách theo thứ tự của danh sách đó. Nó có thời gian chạy khá lớn: *O(n)*, trong đó *n* là số phần tử trong danh sách, nhưng có thể sử dụng cho một danh sách bất kỳ mà không cần tiền xử lý.

Tìm kiếm nhị phân là một thuật toán cao cấp hơn so với thuật toán tìm kiếm tuần tự với thời gian chạy là *O(logn)*. Đối với các danh sách lớn, thuật toán này tốt hơn hẳn tìm kiếm tuyến tính nhưng nó đòi hỏi danh sách phải được sắp xếp từ trước và đòi hỏi khả năng truy cập ngẫu nhiên. Thuật toán tìm kiếm nội suy tốt hơn so với thuật toán tìm kiếm nhị phân đối với danh sách rất lớn và phân bổ gần đều.

Ngoài ra bảng băm (Hash Table) cũng được dùng cho tìm kiếm trên danh sách. Nó đòi hỏi thời gian hằng số trong trường hợp trung bình, nhưng lại cần nhiều chi phí về không gian bộ nhớ và thời gian chạy *O(n)* cho trường hợp xấu nhất. Một phương pháp tìm kiếm khác dựa trên các cấu trúc dữ liệu chuyên biệt sử dụng cây tìm kiếm nhị phân cân bằng và đòi hỏi thời gian chạy *O(log n)*. Các giải thuật loại này có thể coi là mở rộng của tư tưởng chính về tìm kiếm nhị phân để cho phép chèn và xóa nhanh.

1.3.1.2 Tìm kiếm trên cây

Tìm kiếm trên cây [1] là trung tâm các kỹ thuật tìm kiếm. Các kỹ thuật này tìm kiếm trên các cây gồm các nút, cây này có thể là cây tường minh hoặc được xây dựng dần trong quá trình tìm kiếm.

Nguyên lý cơ bản là: Một nút được lấy ra từ một cấu trúc dữ liệu, các nút con của nó được xem xét và bổ sung vào cấu trúc dưc liệu đó. Bằng cách thao tác trên cấu trúc dữ liệu này, cây tìm kiếm được duyệt theo các thứ tự khác nhau, chẳng hạn theo từng mức (tìm kiếm theo chiều rộng) hoặc đi tới một nút lá trước rồi quay lui (tìm kiếm theo chiều sâu).

Tìm kiếm theo chiều rộng (BFS)

Tìm kiếm theo chiều rộng mang hình ảnh của vết dầu loang. Từ trạng thái ban đầu, ta xây dựng tập hợp *S* bao gồm các trạng thái kế tiếp (mà từ trạng thái ban đầu có thể biến đổi thành) sau đó ứng với mỗi trạng thái T_k trong tập *S*, ta xây dựng tập S_k bao gồm các trạng thái kế tiếp của T_k rồi lần lượt bổ sung các S_k vào *S*. Quá trình này cứ lặp lại cho đến lúc *S* có chứa trạng thái kết thúc hoặc *S* không thay đổi sau khi đã bổ sung tất cả S_k.

Tìm kiếm theo chiều sâu

Trong tìm kiếm theo chiều sâu, tại trạng thái (đỉnh) hiện hành, ta cho chọn một trạng thái kế tiếp (trong tập các trạng thái có thể biến đổi thành từ trạng thái hiện hành) làm trạng thái hiện hành cho đến lúc trạng thái hiện hành là trạng thái đích. Trong trường hợp trạng thái hiện hành ta không thể biến đổi thành trạng thái kế tiếp thì ta quay lui (Backtracking) lại trạng thái hiện hành (trạng thái biến đổi thành trạng thái hiện hành) để chọn đường khác. Nếu ở trạng thái trước này mà cũng không thể biến đổi được nữa thì ta quay lui lại trạng thái trước nữa và cứ thế. Nếu ta quay lui đến trạng thái khởi đầu mà vẫn thất bại thì kết luận là không có lời giải.

Tìm kiếm theo chiều sâu và tìm kiếm theo chiều rộng đều là các phương pháp tìm kiếm có hệ thống và chắc chắn tìm ra lời giải. Tuy nhiên, do bản chất là vét cạn nên với những bài toán có không gian lớn thì ta không thể dùng hai chiến lược này được. Hơn nữa, hai chiến lược này đều có tính chất "mù" vì chúng không chú ý đến những thông tin (tri thức) ở trạng thái hiện thời và thông tin về đích cần đạt tới cùng mối quan hệ giữa chúng. Các tri thức này vô cùng quan trọng và rất có ý nghĩa để thiết kế các giải thuật hiệu quả hơn. Do đó, hai chiến lược trên được cải tiến thành một số thuật toán tìm kiếm mới trên cây bao gồm: Tìm kiếm lặp sâu dần, tìm kiếm chiều sâu giới hạn, tìm kiếm hai chiều và tìm kiếm chi phí đều.

1.3.1.3. Tìm kiếm trên đồ thị

Nhiều dạng bài toán tìm kiếm cụ thể trên đồ thị như: Tìm đường ngắn nhất, tìm cây bao trùm nhỏ nhất, tìm bao đóng bắc cầu,… Tuy nhiên ứng với mỗi dạng bài toán có một số giải thuật tìm kiếm thích hợp để giải quyết. Chẳng hạn thuật toán Dijkstra, thuật toán Kruskal, giải thuật láng giềng gần nhất và giải thuật Prim [1]. Các thuật toán này có thể được coi là các mở rộng

của các thuật toán tìm kiếm trên cây: Tìm kiếm theo chiều sâu, tìm kiếm theo chiều rộng.

Thuật toán Dijkstra: Là một thuật toán giải quyết bài toán đường đi ngắn nhất nuồn đơn trong một đồ thị có hướng không có cạnh mang trọng số âm. Thuật toán này có thể tính toán tất cả các đường đi ngắn nhất từ một đỉnh xuất phát cho trước tới mọi đỉnh khác mà không làm tăng thời gian chạy.

Thuật toán Kruskal: Là thuật toán xây dựng cây bao trùm ngắn nhất bằng cách chọn thêm dần các cung vào cây.

Thuật toán Prim: Là thuật toán nhằm xây dựng cây bao trùm ngắn nhất. Tư tưởng của thuật giải Prim là chọn đưa dần vào cây T các đỉnh kề "tốt nhất" trong số các đỉnh còn lại. Thời gian thực hiện giải thuật Prim là $O(n^2)$.

1.3.2. Tìm kiếm có thông tin

Các kỹ thuật tìm kiếm không có thông tin trong một số trường hợp rất kém hiệu quả và thậm chí không áp dụng được. Để tăng tốc độ của các quá trình tìm kiếm ta có thể dùng các giải thuật tìm kiếm có thông tin. Một số chiến lược tìm kiếm có thông tin hay còn gọi là chiến lược tìm kiếm *Heuristic* (Tìm kiếm kinh nghiệm), đó là các phương pháp sử dụng hàm đánh giá để hướng dẫn sự tìm kiếm.

Trong nhiều vấn đề, ta có thể sử dụng kinh nghiệm, tri thức của chúng ta về vấn đề để đánh giá các trạng thái của vấn đề. Với mỗi trạng thái *u* ta xác định một giá trị số *h(u)*, số này đánh giá "sự gần đích" của trạng thái *u*. Hàm *h(u)* được gọi là hàm đánh giá. Trong tìm kiếm có thông tin người ta sử dụng hàm đánh giá này như một đánh giá *Heuristic* đặc thù cho bài toán cần giải quyết với vai trò hướng dẫn cho quá trình tìm kiếm. Một cách đánh giá *Heuristic* tốt sẽ làm cho quá trình tìm kiếm thông tin hoạt động hiệu quả hơn

hẳn một phương pháp tìm kiếm không có thông tin bất kỳ. Trong quá trình tìm kiếm, tại mỗi bước ta sẽ chọn trạng thái có giá trị hàm đánh giá là nhỏ nhất, trạng thái này được xem là trạng thái có nhiều hứa hẹn nhất hướng tới đích.

Các kỹ thuật tìm kiếm sử dụng hàm đánh giá để hướng dẫn sự tìm kiếm được gọi chung là các kỹ thuật tìm kiếm có thông tin hay tìm kiếm kinh nghiệm (tìm kiếm Heuristic). Các giai đoạn cơ bản để giải quyết vần đề bằng tìm kiếm *Heuristic* như sau:

- Tìm biểu diễn thích hợp mô tả các trạng thái và các toán tử hay phép chuyển của vấn đề.
- Xây dựng hàm đánh giá.
- Thiết kế chiến lược chọn trạng thái để phát triển ở mỗi bước.

1.4. Bài toán tối ưu hóa tổ hợp

Bài toán tối ưu hóa tổ hợp (Combinatorial Optimizatoin) liên quan đến việc tìm giá trị cho các biến số rời rạc như lời giải tối ưu mà có lưu ý tới hàm đánh giá cho trước. Bài toán có thể là bài toán tìm cực đại hoặc tìm cực tiểu. Một cách thông thường, bài toán tối ưu hóa tổ hợp được cho dưới dạng bộ ba (S,f, Ω). Trong đó:

- S là tập các lời giải ứng cử viên.
- F là hàm đánh giá (hàm này gán giá trị *f(s)* sao cho mỗi lời giải ứng viên $s \in S$).
- Ω là tập các ràng buộc của bài toán.

Các lời giải thuộc tập $S^* \subseteq S$ thỏa mãn các ràng buộc Ω gọi là lời giải khả thi. Mục tiêu bài toán là tìm ra một lời giải s^* với giá nhỏ nhất, nghĩa là $f(s^*) \leq f(s)$ với mọi lời gải $s \subseteq S$. Ngược lại bài toán tối ưu hóa cực đại là tìm

lời giải s^* với giá lớn nhất, nghĩa là $f(s^*) \geq f(s)$ với mọi lời giải $s \subseteq S$. Bài toán tối ưu hóa tổ hợp có thể chia hai loại: Bài toán tĩnh và bài toán động.

Bài toán tối ưu hóa tổ hợp tĩnh (Static Combinatorial Optimization)

Là bài toán tối ưu hóa tổ hợp trong đó cấu trúc (Topology) và giá (Cost) không thay đổi khi bài toán đang được giải quyết. Ví dụ bài toán người du lịch (TSP). Khi thực hiện thuật toán để giải quyết bài toán vị trí các thành phố, khoảng cách giữa các thành phố là không thay đổi.

Bài toán tối ưu hóa tổ hợp động (Dynamic Combinatorial Optimization)

Là bài toán tối ưu hóa tổ hợp trong đó cấu trúc và giá có thể thay đổi khi bài toán đang được giải quyết. Ví dụ bài toán định hướng trong mạng viễn thông, trong đó mô hình mạng và dung lượng yêu cầu luôn thay đổi.

Lớp bài toán tối ưu hóa tổ hợp có những đặc điểm sau:

- Tìm trạng thái tối ưu hóa cực đại hóa hoặc cực tiểu hóa hàm mục tiêu. Không quan tâm tới đường đi.
- Không gian trạng thái lớn.
- Không thể sử dụng các phương pháp tìm kiếm thông thường để xem xét tất cả không gian trạng thái.
- Thuật toán cho phép tìm lời giải tốt nhất với độ phức tạp tính toán nhỏ. Thuật toán cũng chấp nhận lời giải tương đối tốt.

Tối ưu hóa tổ hợp là lớp bài toán có nhiều ứng dụng trên thực tế, một số bài toán kinh điển trong lớp bài toán này là: Bài toán người du lịch, bàn toán n – queens, bài toán tô màu đồ thị, bài toán xếp lịch y tá…

1.5. Giải thuật tìm kiếm cục bộ

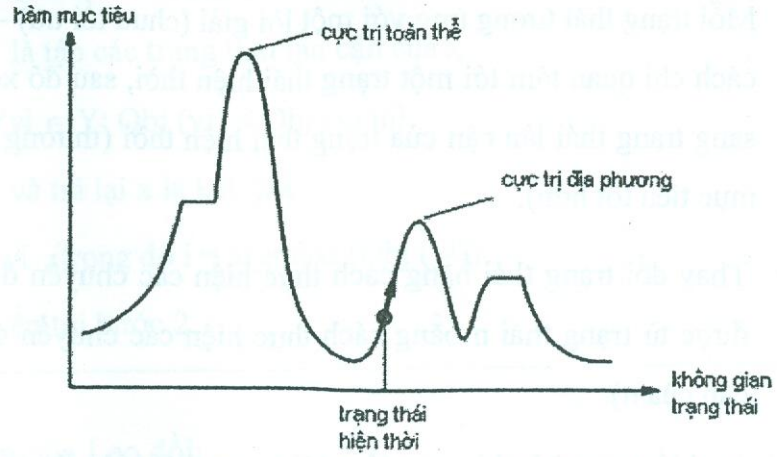
Giải thuật tìm kiếm cục bộ là giải pháp *Metaheuristic* [11] cho việc giải các bài toán tối ưu hóa tổ hợp hoặc tối ưu hóa rời rạc trên máy tính, tức là

những bài toán trong đó cần tìm trạng thái tối ưu hoặc tổ hợp tối ưu trong không gian rời rạc các trạng thái và không quan tâm tới đường đi dẫn tới trạng thái đó. Giải thuật này có thể áp dụng cho các bài toán tìm kiếm lời giải gần đúng tối ưu trong một loạt các lời giải ứng viên. Phương pháp tìm kiếm sẽ duyệt qua các lời giải trong không gian tìm kiếm cho đến khi tìm ra lời giải được cho là tối ưu hoặc vượt quá thời gian tìm kiếm cho phép.

Tìm kiếm cục bộ được thiết kế cho bài toán tìm kiếm với không gian trạng thái rất lớn và cho phép tìm kiếm trạng thái tương đối tốt với thời gian tìm kiếm chấp nhận được.

Ý tưởng chung của tìm kiếm cục bộ:

- Chỉ quan tâm đến trạng thái đích, không quan tâm đến đường đi.



Hình 1.1. Bài toán tìm kiếm cục bộ với không gian trạng thái và hàm mục tiêu

1.6. Một số thuật toán tìm kiếm cục bộ cơ bản

1.6.1. Thuật toán Leo đồi

Leo đồi (*Hill climbing*) [2] là tên chung để chỉ một họ các thuật toán. Thuật toán thực hiện bằng cách tạo ra lân cận cho trạng thái hiện thời và di chuyển sang lân cận có hàm mục tiêu tốt hơn, tức là di chuyển lên cao đối với

trường hợp cần cực đại hóa hàm mục tiêu. Thuật toán dựng lại khi đạt tới một đỉnh của đồ thị hàm mục tiêu, tương ứng với trạng thái không có lân cận nào tốt hơn. Đỉnh này có thể là đỉnh cao nhất hoặc cũng là đỉnh thấp hơn (Hình 1.1). Trong trường hợp thấp nhất, thuật toán tìm được giá trị cực trị, trong trường hợp thứ hai thuật toán chỉ tìm được cực trị địa phương. Thuật toán Leo đồi không lưu lại những trạng thái đã qua, đồng thời không nhìn xa hơn lân cận của trạng thái hiện thời.

1.6.1.1. Di chuyển sang trạng thái tốt nhất

Có nhiều phiên bản khác nhau của thuật toán Leo đồi. Một trong những phiên bản thông dụng nhất có tên là Leo đồi di chuyển sang trạng thái tốt nhất (*Best Improvement Hill climbing*). Phiên bản này của Leo đồi lựa chọn trong số lân cận hiện thời lân cận có hàm mục tiêu tốt nhất. Nếu lân cận đó tốt hơn trạng thái hiện thời thì di chuyển sang lân cận đó. Nếu ngược lại thì kết thúc và trả về trạng thái hiện thời. Thuật toán đầy đủ được thể hiện ở dưới:

Đầu vào: Bài toán tối ưu hóa **Đầu ra**: Trạng thái với hàm mục tiêu lớn nhất (hoặc cực đại địa phương)

- Mỗi trạng thái tương ứng với một lời giải (chưa tối ưu) → cải thiện dần bằng cách chỉ quan tâm tới một trạng thái hiện thời, sau đó xem xét để di chuyển sang trạng thái lân cận của trạng thái hiện thời (thường là trạng thái có hàm mục tiêu tốt hơn).
- Thay đổi trạng thái bằng cách thực hiện các chuyển động (trạng thái nhận được từ trạng thái *n* bằng cách thực hiện các chuyển động được gọi là lân cận của *n*)

Do tìm kiếm cục bộ chỉ quan tâm tới trạng thái hiện thời và lân cận nên cần ít bộ nhớ hơn nhiều so với các phương pháp tìm kiếm thông thường. Tìm

kiếm cục bộ thường cho phép tìm được lời giải chấp nhận được kể cả khi bài toán lớn đến mức không dùng được những phương pháp tìm kiếm thông thường.

Phát biểu bài toán: Bài toán tìm kiếm cục bộ [2] được cho bởi những thành phần sau:

- Không gian trạng thái *X*.
- Tập chuyển động để sinh ra lân cận.
- Hàm mục tiêu *Obj:* $X \rightarrow R$.
- Yêu cầu: Tìm trạng thái X^* sao cho *Obj* (X^*) là min hoặc max.

Có thể minh họa bài toán tìm kiếm cục bộ như Hình 1.1.

- Trục hoành trên hình vẽ thể hiện không gian các trạng thái (để cho đơn giản, không gian trạng thái ở đây được thể hiện trong không gian một chiều dưới dạng các điểm trên trục hoành).
- Trục tung là độ lớn của hàm mục tiêu.

Yêu cầu bài toán tối ưu hóa tổ hợp là tìm được trạng thái (điểm trên trục hoành) có hàm mục tiêu lớn nhất. Hình vẽ minh họa trường hợp cần tìm trạng thái với hàm mục tiêu lớn nhất, tuy nhiên trong một số bài toán khác có thể yêu cầu tìm trạng thái với hàm mục tiêu nhỏ nhất.

1. Chọn ngẫu nhiên trạng thái x
2. Gọi Y là tập các trạng thái lân cận của x
3. Nếu $\forall$ yi $\in$ Y: Obj(yi)< Obj(x) thì kết thúc và trả lại x là kết quả
1. x$\leftarrow$yi, trong đó i = argmaxi (Obj(yi))
2. Chuyển tới bước 2

Đặc điểm của leo đồi:

- Đơn giản, dễ lập trình, không tốn bộ nhớ do không phải lưu lại bất kỳ thứ gì, chỉ lưu lại trạng thái tạm thời và các lân cận.
- Dễ bị lời giải tối ưu cục bộ (cực trị địa phương) tương ứng với đỉnh các "đồi" thấp trong hình 1.1. Để khắc phục vấn đề này, thuật toán được thực hiện nhiều lần, mỗi lần sử dụng một trạng thái xuất phát sinh ngẫu nhiên khác với trạng thái xuất phát trong những lần trước đó.

Khi thiết kế thuật toán leo đồi, việc lựa chọn chuyển động rất quan trọng. Nếu nhiều chuyển động sẽ sinh ra nhiều lân cận do vậy việc chọn ra lân cận tốt nhất đòi hỏi nhiều thời gian do phải tính hàm mục tiêu cho tất cả lân cận. Ngược lại, nếu sinh ra tập lân cận nhỏ sẽ dễ dẫn tới cực trị địa phương do không vượt qua được những "hố" nhỏ trên đường đi.

1.6.1.2. Leo đồi ngẫu nhiên

Leo đồi ngẫu nhiên (*Stochastic Hill Climbing*) là một phiên bản khác của leo đồi. Thay vì tìm ra lân cận tốt nhất, phiên bản này lựa chọn ngẫu nhiên một lân cận. Nếu lân cận đó tốt hơn trạng thái hiện thời, lân cận đó sẽ được chọn làm trạng thái hiện thời và thuật toán lặp lại. Ngược lại, nếu lân cận được chọn không tốt hơn, thuật toán sẽ chọn ngẫu nhiên một lân cận khác và so sánh. Thuật toán kết thúc và trả lại trạng thái hiện thời khi đã quá thời gian. Thông thường, quá thời gian được cho bằng số lượng tối đa lân cận mà thuật toán xem xét trong mỗi bước lặp hoặc trong toàn bộ thuật toán.

Đầu vào: Bài toán tối ưu hóa **Đầu ra**: Trạng thái với hàm mục tiêu lớn nhất (hoặc cực đại địa phương)

1. Chọn ngẫu nhiên trạng thái x 2. Chọn Y là tập các trạng thái lân cận của x

3. Chọn ngẫu nhiên yi ∈ Y
4. Nếu Obj (yi)>Obj(x) thì x ← yi
5. Chuyển tới bước 2 nếu chưa hết kiên nhẫn

Các nghiên cứu cho thấy, trong một số trường hợp leo đồi ngẫu nhiên cho kết quả nhanh hơn và có thể tránh được một số cực trị địa phương.

1.6.2. Thuật toán Luyện thép

Một vấn đề lớn với thuật toán leo đồi là thuật toán không có khả năng ‘‘đi xuống’’ do vậy không thoát khỏi được cực trị địa phương khi đã rơi vào. Ngược lại, cách di chuyển hoàn toàn ngẫu nhiên (*Random walk*) có thể khảo sát toàn bộ không gian trạng thái nhưng hiệu quả. Thuật toán Luyện thép (*Simulated Annealing*) [2] là một phương pháp tìm kiếm cục bộ cho phép giải quyết phần nào vấn đề cực trị địa phương một cách tương đối hiệu quả.

Có thể coi Luyện thép là phiên bản của thuật toán leo đồi ngẫu nhiên, trong đó thuật toán chấp nhận cả những trạng thái kém hơn trạng thái hiện thời với một xác suất *p* nào đó. Cụ thể là khi lựa chọn ngẫu nhiên một lân cận, nếu lân cận đó kém hơn trạng thái hiện thời, thuật toán có thể quyết định di chuyển sang đó với một xác suất *p*.

Theo thời gian, giá trị của *p* phải giảm dần. Ý nghĩa của việc giảm *p* theo thời gian là do mới bắt đầu, thuật toán chưa ở vào vùng trạng thái tốt và do vậy chấp nhận thay đổi lớn. Theo thời gian, thuật toán sẽ chuyển sang trạng thái tốt hơn và do vậy cần hạn chế thay đổi.

Vấn đề quan trọng với thuật toán là lựa chọn xác suất *p* thế nào. Nguyên tắc chung là không chọn *p* cố định, giá trị của *p* được xác định dựa trên hai yếu tố sau:

- Nếu trạng thái mới kém hơn nhiều so với trạng thái hiện thời thì *p* phải giảm đi. Có nghĩa là xác suất chấp nhận trạng thái tỷ lệ nghịch với độ kém của trạng thái đó. Gọi *Δ(x,y) = Obj(y)* trong đó *x* là trạng thái hiện thời, ta cần chọn *p* tỷ lệ nghịch với *Δ(x,y)*.
- Theo thời gian, giá trị của *p* phải giảm dần. Ý nghĩa của việc giảm *p* theo thời gian là do khi mới bắt đầu, thuật toán chưa ở vào vùng trạng thái tốt và do vậy chấp nhận thay đổi lớn. Theo thời gian, thuật toán sẽ chuyển sang vùng trạng thái tốt hơn và do vậy cần hạn chế thay đổi.

SA(X, Obj, N, m, x, C) //Obj càng nhỏ càng tốt
Đầu vào : Số bước lặp m
Trạng thái bắt đầu x (chọn ngẫu nhiên)
Sơ đồ làm lạnh C
Đầu ra : Trạng thái tốt nhất x^*
Khởi tạo : $x^* = x$
For i = 1 to m
1. Chọn ngẫu nhiên $y \in N(x)$
$\Delta(x,y) = Obj(y) - Obj(x)$
If $\Delta(x,y)<0$ then p = 1
Else $p = e - \Delta(x,y)/T$
    If rand[0,1] < p then $x \leftarrow y$
    If $Obj(x) < Obj(x^*)$ then $x^* \leftarrow x$
2. Giảm T theo sơ đồ C
Return x^* // x^* là trạng thái tốt nhất trong số những trạng thái đã xem xét

Thuật toán Luyện thép vừa trình bày dựa trên một hiện tượng cơ học là quá trình làm lạnh kim loại để tạo ra cấu trúc tinh thể bền vững. Hàm mục

tiêu khi đó được đo bằng độ vững chắc của cấu trúc tinh thể. Khi còn nóng, mức năng lượng trong kim loại cao, các nguyên tử kim loại có khả năng di chuyển linh động hơn. Khi nhiệt độ giảm xuống, tinh thể dần chuyển tới trạng thái ổn định và tạo ra mạng tinh thể. Bằng cách thay đổi nhiệt độ hợp lý, có thể tạo ra những mạng tinh thể rất rắn chắc.

Chính vì sự tương tự với cách tôi kim loại như vậy nên thuật toán xác suất *p* giảm theo thời gian dựa vào một công thức gọi là sơ đồ làm lạnh C. Có nhiều dạng sơ đồ làm lạnh khác nhau. Sau đây là ví dụ một sơ đồ làm lạnh:

$T_{t+1}=T_0*\alpha^{t*k}$

Trong đó:

$T_0 > 0$, α thuộc (0,1), $1<k<m$

t càng tăng $\rightarrow \alpha$ càng nhỏ $\rightarrow$ T càng nhỏ

Khi $T\rightarrow\infty$: p = 1 với $\Delta(x,y)\rightarrow$ tương đương với chuyển động ngẫu nhiên

Khi $T\rightarrow 0$: p = 0 với $\Delta(x,y)\rightarrow$đưa về trường hợp leo đồi ngẫu nhiên

Việc lựa chọn các tham số cho sơ đồ làm lạnh thường được thực hiện bằng phương pháp thực nghiệm với từng bài toán cụ thể.

1.6.3. Một số thuật toán tìm kiếm cục bộ khác

1.6.3.1. Giải thuật tìm kiếm Lân cận lớn

Giải thuật tìm kiếm Lân cận lớn (*Large Neighborhood Search - LNS*) là một giải thuật tìm kiếm cục bộ thuộc nhóm các giải thuật Very Large Scale Neighborhood Search (gọi tắt là VLSN – các giải thuật tìm kiếm cục bộ với các miền lân cận có kích thước rất lớn và biến động). Với các giải thuật VLSN, các miền láng giềng được xem tại mỗi bước lặp thường có kích thước rất lớn, điều này sẽ giúp quá trình tìm kiếm có thể vượt ra khỏi những điểm tối ưu cục bộ, nhờ đó có thể tìm ra những lời giải gần với tối ưu toàn cục. Tuy

nhiên, chính vì kích thước miền láng giềng lớn mà các giải thuật VLSN khi chạy thường tốt rất nhiều thời gian. Nhiều kĩ thuật đã được đề nghị để khắc phục vấn đề này, từ đó tạo nên nhiều biến thể VLSN khác nhau. Giải thuật LNS chính là một trong những biến thể này. Trong giải thuật LNS, việc duyệt miền láng giềng của lời giải hiện tại được hiện thông qua hai bước chính : Bước *phá hủy (Destroy)* và bước *chỉnh sửa (Repair)*. Tại bước phá hủy, một số phần tử của lời giải hiện tại sẽ bị loại ra tạo nên một *thành phần chưa đầy đủ (Partial Solution),* sau đó tại bước chỉnh sửa, các phần tử vừa bị loại sẽ lần lượt được thêm trở lại vào lời giải thành phần của bước trước, để tạo lại một lời giải hoàn chỉnh. Như vậy, miền lân cận tại mỗi bước lặp chính là tập các lời giải đầy đủ mới thu được sau khi áp dụng hai bước phá hủy và chỉnh sửa lên lời giải hiện tại. Phương pháp cụ thể được chọn để thực hiện bước phá hủy và bước chỉnh sửa sẽ quyết định lượng thời gian cần tiêu tốn cho một bước lặp. Phương pháp đơn giản nhất là chọn ngẫu nhiên tại bước phá hủy và thực hiện thêm các phần tử vào bước chỉnh sửa bằng một giải thuật tham lam đơn giản.

```
Input: Giải thuật khả thi x
x^b = x;
Repeat
   x^t = r (d(x));
   if accept(x^t,x) then
      x=x^t
   end if
   if c(x^t)<c(x^b) then x^b=x^t;
   end if
```

until khi gặp điều kiện dừng

return x^b;

Trong mã giả của giải thuật LNS ở trên, đầu vào của giải thuật là một *lời giải hợp lệ (Feasible Solution) x*, nghĩa là một lời giải thỏa tất cả các ràng buộc cứng, x^b là *lời giải tốt nhất hiện tại, d(.)* là hàm phá hủy, *r(.)* là hàm chỉnh sửa. Ở đây, tại mỗi bước lặp, thay vì quét toàn bộ miền láng giềng và chọn ra lời giải tốt nhất giải thuật chỉ chọn một lời giải duy nhất *x'* thuộc tập láng giềng *N(x)* (có thể phát sinh ngẫu nhiên hoặc chọn lời giải tốt nhất thuộc một tập con rất nhỏ của tập láng giềng *N(x)*). Tiêu chuẩn *accept (x', x)* sẽ quyết định xem có chọn lời giải *x'* thay thế cho lời giải hiện tại hay không, hai tiêu chuẩn *accept (x', x)* phổ biến nhất là:

- Chỉ chấp nhận các lời giải tốt hơn lời giải hiện tại: Nếu x'<x thì *accept (x', x) = true,* ngược lại *accept (x', x)=false.*
- Chấp nhận theo tiêu chuẩn của giải thuật Luyện thép:

Nếu r< $e^{\frac{c(x')-c(x)}{T}}$ thì *accept (x', x)=true,* ngược lại *accept (x', x)= false*,

với r là một số ngẫu nhiên thuộc đoạn [0,1].

Việc chọn phương pháp phá hủy và phương pháp chỉnh sửa sao cho hợp lý và các vấn đề rất quan trọng, quyết định tính hiệu quả của giải thuật LNS.

- Quá trình phá hủy: Việc chọn kích thước tập các phần tử bị loại ra khỏi lời giải hiện tại rất quan trọng, nếu con số này quá nhỏ, quá trình tìm kiếm sẽ khó thoát ra khỏi tối ưu cục bộ, nhưng nếu con số này quá lớn, việc tìm kiếm sẽ mất rất nhiều thời gian. Hơn nữa, chiến lược chọn tập các phần tử bị loại cũng phải được chọn sao cho quá trình tìm kiếm có thể bao quát hết không gian lời giải hoặc hướng được đến nhiều vùng không gian lời giải có khả năng chứa tối ưu

toàn cục, tránh việc "lẩn quẩn" quanh một miền không gian con duy nhất. Do đó, các chiến lược phá hủy thường có chứa yếu tố mang tính xác suất.

- Quá trình chỉnh sửa: Quá trình chỉnh sửa có thể thực hiện bằng phương pháp *heuristic* hoặc bằng một thuật toán chính xác. Nếu sử dụng các phương pháp chính xác để giải, trong một số trường hợp sẽ giúp đạt đến tối ưu toàn cục nhanh hơn, tuy nhiên trong một số trường hợp có thể làm giảm tính đa dạng hóa (diversification) của quá trình tìm kiếm. Do đó, cần cân nhắc và chọn lựa phương pháp giải quyết phù hợp tùy từng trường hợp cụ thể.

1.6.3.2. Giải thuật tìm kiếm Tabu

Giải thuật tìm kiếm Tabu (Tabu Search - TS) được đưa ra đầu tiên bởi Glover vào năm 1986. Ý tưởng này cũng được đề nghị bởi Hansen (1986), sau đó đã có nhiều nghiên cứu đề ra các kỹ thuật thêm vào để mang lại hiệu quả cao hơn cho tìm kiếm Tabu: Werra và Hertz (1989), Glover (1989,1990), Fox (1993)… Hiện nay, bằng kết quả thực nghiệm cho thấy kỹ thuật tìm kiếm Tabu có thể giải quyết hiệu quả các bài toán tối ưu.

Ý tưởng chính của kỹ thuật tìm kiếm Tabu là sử dụng một danh sách lưu trữ các lời giải đã đi qua để đảm bảo rằng chúng ta sẽ *không viếng thăm một lời giải hai lần*. Trong giải thuật tìm kiếm Tabu, những bước chuyển có chi phí thấp hơn với lời giải hiện tại cũng vẫn được chấp nhận chỉ cần nó không có trong "danh sách cấm" (*Tabu List*), là danh sách các lời giải đã được viếng thăm. So với giải thuật leo đồi, giải thuật tìm kiếm Tabu tiếp tục tìm kiếm trong trường hợp không thể tìm ra được lời giải tốt hơn lời giải hiện tại với hi vọng sẽ vượt qua được những lời giải tối ưu cục bộ.

Trong giải thuật tìm kiếm Tabu, kích thước của Tabu List hết sức quan trọng. Chiều dài Tabu List phải đủ lớn để đảm bảo thoát ra khỏi được vùng tối

ưu cục bộ nhưng cũng phải đủ ngắn để tránh bỏ qua các lời giải có khả năng. Đây là tham số ảnh hưởng đến hiệu quả của việc tìm kiếm và được thiết lập bởi người lập trình, vì vậy đối với mỗi bài toán cụ thể chúng ta cần trải qua thực nghiệm mới có thể đưa ra thông số tốt nhất cho giải thuật.

CHƯƠNG 2: TÌM KIẾM TABU

2.1. Nguyên lý chung của tìm kiếm Tabu

Tabu được viết lại từ chữ *Taboo*, *Taboo* mang ý nghĩa chỉ sự cấm kỵ trong tiếng Anh. *Ý tưởng chính của kỹ thuật tìm kiếm Tabu là sử dụng một danh sách lưu trữ các lời giải đã đi qua (Tabu List) để đảm bảo rằng chúng ta sẽ không viếng thăm một lời giải hai lần*. Trong giải thuật tìm kiếm Tabu, những bước chuyển có chi phí thấp hơn với lời giải hiện tại cũng vẫn chấp nhận chỉ cần nó không có trong "danh sách cấm" là danh sách các lời giải đã được viếng thăm. So với giải thuật leo đồi, giải thuật tìm kiếm Tabu tiếp tục tìm kiếm trong trường hợp không thể tìm ra được lời giải tốt hơn lời giải hiện tại với hi vọng sẽ vượt qua được những lời giải tối ưu cục bộ.

Tìm kiếm Tabu dựa trên giả thuyết vấn đề đã được giải, kết hợp chặt chẽ *bộ nhớ thích nghi* và *thăm dò phản ứng (Responsve Exploration*). Giống như việc leo núi, người leo núi phải nhớ có chọn lọc các thành phần của quãng đường đi qua (sử dụng *Adaptive Memory*) và lập ra các lựa chọn chiến lược trên đường (sử dụng *Responsve Exploration*). Bộ nhớ thích nghi này cho phép việc tìm kiếm trong không gian lời giải một cách tiết kiệm và hiệu quả.

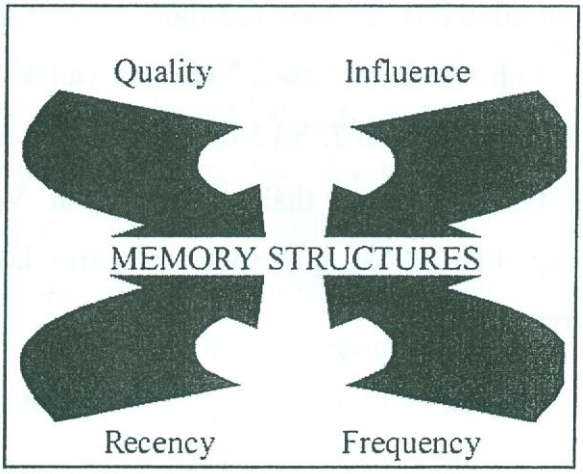
Việc nhấn mạnh vào đặc điểm *thăm dò phản ứng* của tìm kiếm Tabu được hiểu rằng, dù một lựa chọn chiến lược kém thì vẫn cung cấp nhiều thông tin hơn một lựa chọn ngẫu nhiên tốt (trong hệ thống sử dụng bộ nhớ, một lựa chọn kém nhưng dựa trên chiến lược có thể cung cấp nhiều thông tin hơn về cách mà chiến lược đã thay đổi thuận lợi như thế nào).

Thăm dò phản ứng tích hợp các nguyên lý cơ bản của *tìm kiếm thông minh* (khai thác những đặc điểm lời giải tốt trong khi vẫn tìm kiếm những vùng có tiềm năng khác). Tìm kiếm Tabu được phối hợp với việc tìm kiếm những con đường mới và hiệu quả hơn trong việc kết hợp những điểm mạnh của những kỹ thuật có liên quan đến cả *bộ nhớ thích nghi* và *thăm dò phản ứng*.

2.2. Cách sử dụng bộ nhớ

Cấu trúc bộ nhớ trong TS[5] hoạt động bằng việc tham khảo bốn chiều chính sau:

- Tính chất *mới xẩy ra (Recency).*
- Tính chất *thường xuyên (Frequency).*
- Tính chất *chất lượng (Quality).*
- Tính chất *ảnh hưởng (Infuence).*



Hình 2.1: Cấu trúc bộ nhớ tìm kiếm Tabu

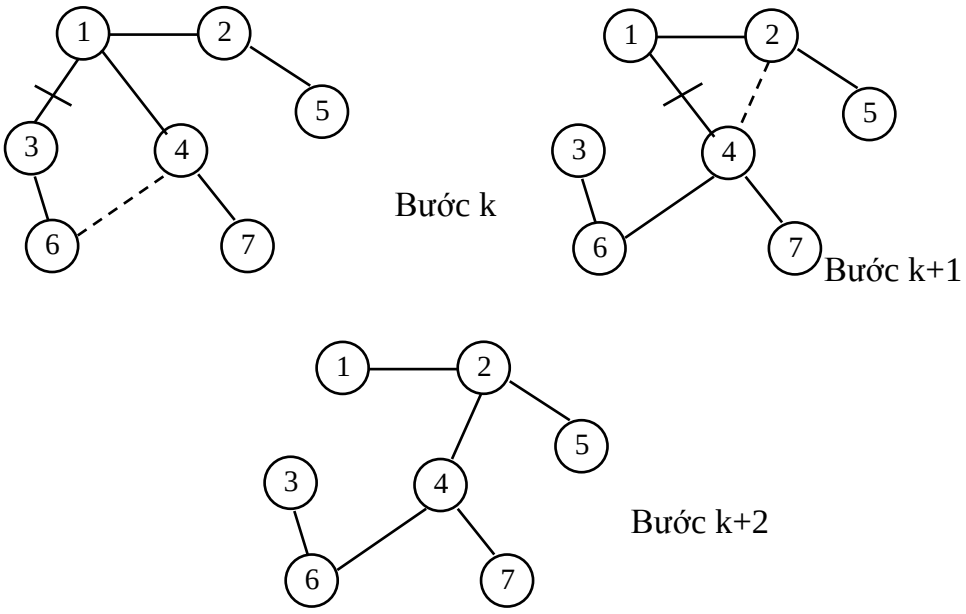
Bộ nhớ *Recency – based* và *Frequency– based* hỗ trợ lẫn nhau.

Chiều *chất lượng* thể hiện khả năng phân biệt chất lượng của các lời giải được tìm thấy trong quá trình tìm kiếm. Trong ngữ cảnh này, bộ nhớ có thể sử dụng để nhận dạng các thành phần hoặc các con đường dẫn tới lời giải tốt. Tính chất lượng hướng tới việc tạo ra các *thành phần khích lệ* để cung cấp các hướng dẫn đến lời giải tốt và các *thành phần vi phạm (Penalty)* để ngăn chặn các hướng dẫn đến lời giải kém. Khái niệm chất lượng được sử dụng trong tìm kiếm Tabu rộng hơn so với cái được sử dụng trong phương pháp tối ưu chuẩn. Chiều thứ tư là tính ảnh hưởng xem việc ảnh hưởng của các lựa chọn được tạo trong quá trình tìm kiếm không nhưng trên chất lượng mà còn trên cấu trúc (có thể xem tính chất chất lượng là một dạng đặc biệt của tính chất ảnh hưởng).

Bộ nhớ sử dụng trong tìm kiếm Tabu là *bộ nhớ hiện (Explicit Memory)* và *bộ nhớ thuộc tính (Attributive Memory)*. Bộ nhớ hiện ghi nhận toàn bộ lời giải, thường là chứa các lời giải tốt trong quá trình tìm kiếm. Những lời giải tốt đã ghi nhận sẽ được dùng để mở rộng tìm kiếm cục bộ. Những ứng dụng của loại bộ nhớ này giới hạn ở chỗ, vì cần phải thiết kế cấu trúc dữ liệu để không tốn quá nhiều bộ nhớ. Thay vào đó, tìm kiếm Tabu sử dụng bộ nhớ thuộc tính. Loại bộ nhớ này lưu lại thông tin về các thuộc tính của lời giải khi có thay đổi từ lới giải này sang lời giải khác. Ví dụ, trong đồ thị hay mạng, các thuộc tính có thể bao gồm các nút hoặc các cung được thêm vào hoặc bớt đi bởi phép chuyển.

Ví dụ minh họa:

Đây là một minh họa cho thấy các thuộc tính có thể được sử dụng như thế nào trong cấu trúc bộ nhớ *Recency – based*. Xem một vấn đề tìm cây tối ưu trên đồ thị với các nút được đánh số từ 1 đến 7. Và ba đồ thị thể hiện cho đồ thị ở bước *k, k+1, k+2* trong quá trình tạo ra các trạng thái để tìm lời giải. Với quy định là khi một phép chuyển được thực hiện sẽ bao gồm một cạnh được lấy ra và một cạnh được thêm vào để giữ nguyên cây.



Hình 2.2: Minh họa bài toán cây tối ưu

Phép chuyển áp dụng ở bước *k* để tạo ra một cây ở bước *k+1* bao gồm việc gỡ cạnh (1,3) và thêm vào cạnh (4,6). Được thể hiện ở hình 2.2, là cạnh được đánh dấu và cạnh đứt nét. Ở bước *k*, cạnh (1,3) và cạnh ảo (4,6) trong cây được xem là hai thuộc tính lời giải (Solution attribute) khác nhau và cạnh được gọi là (1,3) trong và cạnh (4,6) ngoài. Các thuộc tính này được xem là *Tabu – active*, được dùng để định nghĩa ra trạng thái Tabu (Tabu status) của các phép chuyển ở các bước tiếp theo.

Ví dụ trong bộ nhớ *Recency – based* ta có thể chọn cạnh (1,3) trong là *Tabu – active* trong 3 bước, để giữ cho cạnh (1,3) không bị thêm lại vào cây hiện tại trong những bước này. Do đó cạnh (1,3) chỉ có thể thêm vào cây ở bước gần nhất là bước *k+4*. Tương tự, nếu cạnh (4,6) ngoài *Tabu – active*, trong 1 bước để giữ cho cạnh (4,6) khỏi bị gỡ ra trong 1 bước. Những điều kiện này giúp cho việc thực hiện các phép chuyển tránh lặp lại những lần chuyển ở những bước trước.

Số bước mà một cạnh trong trạng thái *Tabu – active* được gọi là *Tabu tenure*. Nếu giá trị *Tabu tenure* quá lớn sẽ làm cho chất lượng của lời giải không được cải thiện. Nhưng nếu *Tabu tenure* quá nhỏ sẽ làm cho hàm mục tiêu lặp lại theo chu kỳ.

Trong ví vụ trên, khi *bộ nhớ hiện* mở rộng các lời giải lân cận trong khi tìm kiếm cục bộ (bằng cách nhớ các lời giải tốt) thì bộ nhớ thuộc tính làm giảm các lời giải đó (bằng cách giữ hoặc cấm các phép chuyển có chọn lọc).

2.3. Lập trình với bộ nhớ thích nghi

Thực tế tìm kiếm Tabu có hai phiên bản: *Đơn giản và Phức tạp*. Phương pháp đơn giản kết hợp chặt chẽ các phần giới hạn của thiết kế tìm

kiếm Tabu và đôi khi sử dụng trong các phân tích sơ bộ để kiểm tra hiệu suất của các thành phần của một tập con giới hạn và thường chỉ sử dụng *bộ nhớ ngắn hạn*.

Phương pháp phức tạp sử dụng *bộ nhớ dài hạn* cùng với các chiến lược tăng cường, đa dạng liên quan. Cách tiếp cận này chú trọng vào việc khai thác tập hợp của các thành phần trong bộ nhớ, và đôi khi được gọi là *lập trình với bộ nhớ thích nghi (Adaptive Memory Programming)*.

Bộ nhớ thích nghi dựa trên thuộc tính và phụ thuộc vào bốn nhân tố: Vừa xẩy ra, thường xuyên, chất lượng và ảnh hưởng. Bốn nhân tố này che đi các khả năng đầy bất ngờ, mà sẽ lộ rõ ra khi chúng được kết hợp và phân biệt cho các lớp thuộc tính qua các loại khác nhau và các thời điểm khác nhau.

2.4. Làm việc với bộ nhớ dài hạn

Trong nhiều ứng dụng, các thành phần bộ nhớ ngắn hạn của tìm kiếm Tabu cũng đủ để tạo ra các lời giải có chất lượng rất cao. Tuy nhiên, tìm kiếm Tabu sẽ trở trên mạng mẽ hơn bằng việc bao gồm cả bộ nhớ dài hạn và những chiến lược có liên quan của bộ nhớ này. Trong các chiến lược tìm kiếm Tabu dàn hạn, tập lời giải có liên quan được tạo ra bằng tìm kiếm Tabu có thể chứa các lời giải không có trong tập gốc và thường bao gồm cả các lời giải tốt được chọn (tối ưu cục bộ chất lượng cao) đã gặp tại các thời điểm khác nhau trong quá trình tìm kiếm lời giải. Các lời giải tốt này được xác định như các phần tử của một vùng cục bộ trong *chiến lược tăng cường (Intensification Strategy)* và như các phần tử của các vùng khác nhau trong *chiến lược đa dạng (DiversificationStrategy)*.

Không cần phải chạy một lời giải lớn để thấy được hiệu quả của bộ nhớ dàn hạn. Thường hiệu quả của bộ nhớ này bắt đầu trở nên rõ ràng trong một

khoảng thời gian vừa phải và có thể cho phép việc tìm lời giải có thể kết thúc trong khoảng thời gian giới hạn, dựa trên việc tìm kiếm các lời giải với chất lượng rất cao trong một khoảng thời gian vừa phải. Nói cách khác, việc có cơ hội tìm ra được các giải pháp tốt hơn trong thời gian tiếp theo – khi chưa tìm thấy lời giải tối ưu – được tăng cường bằng việc sử dụng bộ nhớ dài hạn ngoài việc chỉ sử dụng bộ nhớ ngắn hạn.

2.5. Tiếp cận dựa trên tần số

*Bộ nhớ tần số (Frequency - based)*cung cấp một loại thông tin bổ sung cho các thông tin do bộ nhớ *Recency based* cung cấp, mở rộng cơ sở đối với việc chọn ra những phép chuyển mong muốn. Cũng như tính vừa mới xảy ra, tính thường xuyên thường được đánh trọng hay phân ra thành các lớp con bằng việc ghi nhận các chiều của chất lượng lời giải và độ ảnh hưởng của phép chuyển. Ngoài ra, tính thường xuyên có thể được tích hợp với tính vừa xảy ra để xây dựng một cấu trúc tổng hợp cho việc tạo ra vi phạm và khích lệ dùng để điều chỉnh các ước lượng phép.

Ta xem tần số gồm các tỉ lệ, trong đó:

- Tử số đại diện bằng hai loại độ đo:
    - *Độ đo chuyển đổi (Transition Measure):* Số lượng bước mà một thuộc tính thay đổi (được đưa vào hay gõ ra khỏi) lời giải được tìm đến trên đường tìm kiếm.
    - *Độ đo cư trú (Residence Measure)*: Số lượng bước mà một thuộc tính thuộc về các lời giải đã đến thăm trên một con đường tìm kiếm cụ thể hoặc là số lần mà một thuộc tính xuất hiện trong các lời giải thuộc một tập con cụ thể.
- Mẫu số thể hiện bằng một trong ba loại số lượng:

- (1) Số lượng tổng cộng các biến cố của tất cả các sự kiện được thể hiện bởi tử số (như số lượng bước có liên quan).
- (2) Tổng cộng (hoặc trung bình cộng) của các tử số.
- (3) Giá trị tử số lớn nhất. Trong trường hợp mà tử số thể hiện trọng số, vài tử số trong số đó có thể bị âm, thì mẫu số loại (3) thể hiện bằng giá trị tuyệt đối và mẫu số loại (2) thể hiện bằng tổng các giá trị tuyệt đối (có khả năng cộng thêm một hằng số nhỏ vào để tránh trường hợp mẫu số bằng 0).

Các tỉ lệ tạo ra *tần số chuyển đổi, tần số cư trú* để theo dõi độ thường xuyên mà các thuộc tính là thành phần của các lời giải được tạo ra. Ngoài ra, liên quan đến các tần số này, các ngưỡng dựa trên các tử số có thể hữu dụng cho việc định ra khi nào các pha của việc *đa dạng hóa lớn hơn (Greater Diversification)* là thích hợp (ngưỡng của các thuộc tính có thể dịch chuyển sau khi một *pha đa dạng hóa* (*Diversification Phase*) được thực hiện).

Tần số cư trú và tần số chuyển đổi đôi khi chứa các thông tin có liên quan đến nhau, nhưng nói chung, chúng mang các ngụ ý khác nhau. Sự khác biệt quan trọng là đại lượng cư trú, ngược lại với đại lượng chuyển đổi, không liên quan đến các tính chất của thuộc tính lời giải cụ thể hoặc nó có là thuộc tính mà thay đổi trong việc chuyển từ một lời giải này sang lời giải khác hay không. Ví dụ trong bài toán *Cây – k tối tiểu (Min k - Tree)*, đại lượng cư trú có thể đếm số lần cạnh *(i,j)* tham gia vào lười giải trong khi đại lượng chuyển đổi có thể đếm số lần cạnh *(i,j)* được thêm vào lời giải cũng có nhiều loại đại lượng phức tạp hơn, như là số lần cạnh *(i,j)* hợp vào lời giải bằng cạnh *(k,l)*, hoặc xóa khỏi lời giải vì cạnh *(k,l)* được thêm vào lời giải, cũng có thể được tạo ra một cách chọn lọc. Nhưng tấn số này liên hệ đến các vấn đề tạo ra các thuộc tính phức tạp hơn từ các loại đơn giản.

Một tần số cư trú cao có thể chỉ ra rằng một thuộc tính rất hấp dẫn nếu vùng đó chứa các lời giải tốt hoặc ngược lại nếu vùng đó chứa các lời giải xấu. Nói cách khác, tần số cư trú cao (hoặc thấp) khi vùng được chọn để lấy cả lời giải tốt lẫn xấu có thể dẫn đến một thuộc tính bị *giữ kỹ* (hoặc bị loại trừ) gây ra việc không gian tìm kiếm bị giới hạn và cần phải được bỏ đi (hoặc thêm vào) để làm độ đa dạng cao hơn. Ví dụ, một thuộc tính bị *giữ kỹ* có thể là một công việc được phân vào vùng vị trí trong một loạt các bước mà bao gồm cả giá trị ước lượng chất lượng tốt và xấu của hàm mục tiêu.

Một tần số chuyển đổi cao, ngược với tần số cư trú cao, có thể chỉ ra thuộc tính có liên quan là một *thuộc tính đầy (Crack Filler),* là loại thuộc tính chuyển vào và chuyển ra khỏi lời giải để thực hiện một chức năng điều chỉnh tốt. Trong ngữ cảnh như vậy thì tần số chuyển đổi được xem như là đại lượng của sự không ổn định.

Vài tập con của những phần tử này cũng có thể là một phần của lời giải tối ưu. Tập con này có thể được xác định mà ít nhất khó khăn hơn khi các phần tử khác cũng ở trong vùng. Nói cách khác, một lời giải (đầy đủ hoặc một phần) có thể chứa các *thuộc tính lấp đầy* thật nhưng lại đưa ra đầu mối để xác định các thuộc tính còn lại có thể chuyển lời giải thành lời giải tối ưu.

Bảng 2.1: Ví dụ về độ đo tần số

Bài toán	**Độ đo cư trú**	**Độ đo tần số**
Sắp công việc	Số lượng công việc *j* giữ vị trí $\pi(j)$	Số lần công việc *i* thay đổi vị trí việc *j*
	Tổng cộng thời gian làm việc của công việc *j* khi công việc này giữ vị trí $\pi(j)$	Số lần công việc *j* được chuyển tới vị trí trước đó trong dãy

Cây – k tối tiểu	Số lần cạnh *(i,j)* tham gia vào lời giải hiện tại.	Số lần cạnh *(i,j)* bị gỡ khỏi lời giải hiện tại khi cạnh *(k,l)* được thêm vào.
	Tổng trọng số khi cạnh *(i,j)* tham gia vào lời giải.	Số lần cạnh *(i,j)* được thêm vào trong quá trình phát triển lời giải

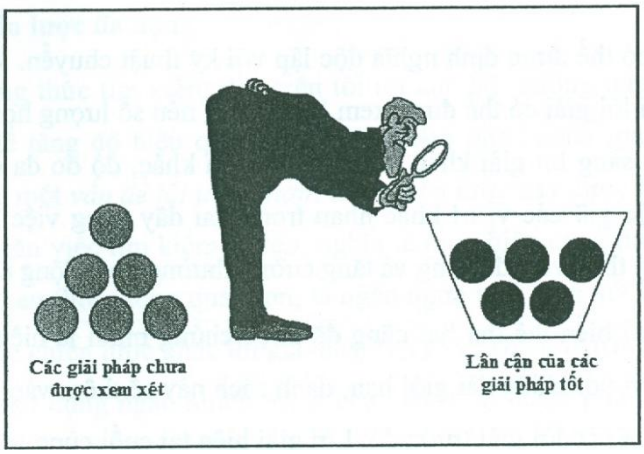
Các thuộc tính có độ đo tần số cao thì cũng là những thuộc tính có độ đo vừa xảy ra cao (có nghĩa là xảy ra trong những lời giải hoặc các phép chuyển gần với hiện tại), có thể *lật (Trigger)* một luật đánh Tabu nếu những thuộc tính đó dựa trên nhiều lời giải liên tiếp nhau và kết thúc bằng lời giải hiện tại. Tuy nhiên, bộ nhớ Frequency – based thường sử dụng trong chiến lược bộ nhớ dài hạn *(Longer Term Strategy)*, nó tận dụng việc khuyến khích cũng như giới hạn để quyết định những phép chuyển nào được lựa chọn. Trong chiến lược này, luật đánh Tabu được chuyển thành ước lượng vi phạm và khích lệ trở thành ước lượng tăng cường, để thay đổi nền tảng cho việc đánh giá phép chuyển là hấp dẫn hay không.

Để mô tả việc này, một thuộc tính như là công việc *j* với một tần số cư trú cao trong vị trí $\pi(j)$ có thể được gán một khích lệ mạnh để xem như là một thuộc tính hoán chuyển, do đó kết quả là trong lựa chọn của một phép chuyển cung cấp một dãy mới Π' với $\pi'(j) \neq \pi(j)$. Khích lệ này đặc biệt liên quan trong trường hợp giá trị *TabuEnd* của công việc *j* nhỏ hơn so với bước hiện tại, vì giá trị này cho biết bước cuối cùng mà công việc *j* là một *thuộc tính hoán chuyển (swap attribute)*. Và do đó cho thấy công việc *j* nằm ở vị trí $\pi(j)$ trong các lời giải từ đó trở đi.

Do đó bộ nhớ *Frequency – based* thường được áp dụng bằng việc đưa ra các trạng thái Tabu đã được đánh dấu như là cơ sở cho việc định nghĩa giá trị vi phạm và khích lệ để điều chỉnh các ước lượng của phép chuyển. Nếu *Tenure* của một thuộc tính trong bộ nhớ *Recency – based* được hình thành như là ngưỡng điều kiện cho việc áp dụng vi phạm lớn, sau đó phân loại Tabu được tạo ra từ bộ nhớ này có thể được chuyển thành kết quả của việc ước lượng mà trở thành cạnh rất kém khi vi phạm được kích hoạt. Ngưỡng điều kiện cũng có liên quan trong việc quyết định giá trị của vi phạm và khích lệ trong các chiến lược dài hạn. Tuy nhiên hầu hết các ứng dụng hiện giờ đều sử dụng một *tuyến tính đa (Linear Multiple)* của độ đo tần số đơn giản để tạo ra cân bằng giữa vi phạm hay khích lệ và hệ số chi phí (hay thuận lợi) của hàm mục tiêu.

2.6. Chiến lược Tăng cường và chiến lược Đa dạng

Hai thành phần rất quan trọng của tìm kiếm Tabu là chiến lược *Tăng cường (Intensication)* và chiến lược *Đa dạng (Diversication)*. Chiến lược tăng cường dựa trên việc thay đổi các luật lựa chọn để tăng cường việc kết hợp các phép chuyển và các đặc tính của lời giải tốt trong quá trình tìm kiếm. Vì các lời giải tốt được ghi nhận lại để đánh giá các lân cận kề chúng, nên bộ nhớ hiện có liên hệ đến các cài đặt cho chiến lược tăng cường này.



Hình 2.3: Tăng cường và đa dạng

Khác biệt chính giữa chiến lược tăng cường và đa dạng là trong chiến lược tăng cường, việc tìm kiếm chú trọng đến việc đánh giá lân cận của các lời giải tốt. Ngược lại, chiến lược đa dạng, thúc đẩy quá trình tìm kiếm khảo sát các vùng chưa được đến thăm và tạo ra các lời giải khác nhau bằng nhiều cách khác nhau từ những lời giải đã có từ trước.

Ở đây từ *lân cận* mang ý nghĩa rộng hơn là trong ngữ cảnh thường dùng của từ *tìm kiếm lân cận*. Có nghĩa là, ngoài xem các lời giải gần với các lời giải tốt bằng những phương thức chuyển bình thường thì chiến lược tăng cường còn tạo ra các *lân cận* bằng cách ghép các thành phần của các lời giải tốt hoặc bằng cách sử dụng các ước lượng đề nghị các thành phần tốt cho lời giải hiện tại.

2.6.1. Các chiến lược tăng cường

Các chiến lược tăng cường được dựa trên việc thay đổi các luật để khuyến khích việc kết hợp các phép chuyển và đặc tính lời giải tốt trong quá trình tìm kiếm. Các chiến lược này có thể khởi tạo một sự đáp lại đến các vùng hấp hẫn để tìm kiếm các lời giải đó kỹ hơn.

Các biến thể của chiến lược này đã chứng minh được là khá thành công. Một loại đưa ra độ đo đa dạng để đảm bảo rằng các lời giải được ghi nhận khác với những cái khác bằng một độ mong muốn, và sau đó xóa tất cả bộ nhớ ngắn hạn trước khi trở lại từ lời giải tốt nhất của các lời giải đã ghi nhận. Độ đo đa dạng có thể liên hệ tới số lượng các phép chuyển cần thiết để chuyển từ lời giải này sang lời giải khác. Hoặc độ đo này có thể được định nghĩa độc lập với kỹ thuật chuyển. Ví dụ, trong bài toán sắp việc, hai lời giải có thể được xem là đa dạng nếu số lượng hoán đổi cần chuyển từ lời giải này sang lời giải khác là lớn. Nói cách khác, độ đo đa dạng có thể là số lượng công việc giữa các vị trí khác nhau trong hai dãy công việc được so sánh (điều này cũng cho thấy việc đa dạng và tăng cường thường hoạt động chung với nhau).

Loại biến thể thứ hai cũng đã được chứng minh là hiệu quả. Loại này giữ một danh sách với chiều dài giới hạn, danh sách này sẽ thêm vào lời giải mới nếu lời giải đó tốt hơn các lời giải trước đó. Lời giải hiện tại cuối cùng trong danh sách luôn luôn là lời giải được chọn (và bị xóa khỏi danh sách) khi bắt đầu lại quá trình tìm kiếm. Tuy nhiên trong tìm kiếm Tabu thì bộ nhớ ngắn hạn thực hiện lời giải này được giữ nguyên và phép chuyển đầu tiên nếu là phép chuyển trước đó của lời giải này sẽ bị cấm, do đó một con đường đi tìm lời giải tiếp theo sẽ hoàn thành mới. Một phiên bản đơn giản của loại hai của chiến lược tăng cường thể hiện ở dưới.

```
Áp dụng một chiến lược lựa chọn các lời giải tốt
Do {
    Chọn ra một trong tập các lời giải tốt
    Chạy lại TS từ lời giải được chọn
    Thêm các lời giải tốt khác vào danh sách các lời giải tốt nếu có thể.
```

} **While** ((các bước < giới hạn) và (danh sách không rỗng))

Và các biến thể thứ ba của tiếp cận này có liên quan đến chiến lược khởi *động lại quá trình tìm kiếm từ các lời giải có liên quan nhưng chưa được xét đến (Unvisited Neighbors)* của các lời giải được tạo ra trước đó. Chiến lược này theo dõi chất lượng của những lời giải liên quan này để chọn ra một tập các lời giải tốt, và giới hạn chú trọng vào các lời giải cụ thể nào đó, như là các lời giải liên quan của tối ưu cục bộ, hoặc là các lời giải liên quan của các lời giải đã được xét lại những bước kề cận trước khi đạt được tối ưu cục bộ. loại chiến lược của các *lời giải chưa xét* này ít khi được thực hiện. Tuy nhiên hiệu quả của nó mang lại không bằng hai loại biến thể trước.

2.6.2. Các chiến lược đa dạng

Các phương thức tìm kiếm dựa trên tối ưu cục bộ thường dựa trên các chiến lược đa dạng để tăng độ hiệu quả trong việc khám phá không gian lời giải được định nghĩa bằng một *vấn đề tối ưu tổ hợp*. Các chiến lược này được thiết kế với mục đích chính là ngăn việc tìm kiếm bị *lặp*, nghĩa là thực hiện cùng một dãy các phép chuyển giống nhau (hoặc tổng quát hơn, là ngăn ngừa việc xem xét lại cùng một tập các lời giải). Các chiến lược khác thì gia tăng khả năng cho quy trình tìm kiếm. Giải thuật Di truyền sử dụng ngẫu nhiên trong các tiến trình thành phần như là việc kết hợp các thành phần quần thể và áp dụng lai ghép (cũng như đột biến), do đó tạo ra một hiệu ứng đa dạng thích hợp. Giải thuật Luyện thép cũng giống như việc kết hợp ngẫu nhiên để tạo cho đa dạng một hàm của nhiệt độ, hàm này giảm dần dần các biến đổi có hướng trong con đường làm việc mục tiêu của các lời giải được chọn.

Trong tìm kiếm Tabu, độ đa dạng được tạo ra cho vài thứ mở rộng bằng các chức năng của bộ nhớ ngắn hạn, nhưng được củng cố cụ thể bằng các thể

của bộ nhớ dài hạn. Các chiến lược tìm kiếm Tabu đa dạng hóa, được thiết kế để hướng quy trình tìm kiếm và các vùng tìm kiếm mới. Thường thì chúng dựa vào việc điều chỉnh các luật lựa chọn để đem các thuộc tính ít khi được dùng vào lời giải. Ngoài ra, các chiến lược này có thể đưa ra các thuộc tính bằng việc áp dụng các phương thức mà gộp các tập con của các thuộc tính này một cách định kỳ vào các giải pháp ứng cửa để tiếp tục việc tìm kiếm.

2.6.2.1. Thay đổi các luật lựa chọn

Xem xét giải thuật tìm kiếm Tabu để giải quyết bài toán Phân vùng đồ thị *(Graph Partitioning)* sử dụng phép hoán chuyển toàn phần và một phần để tìm kiếm các lời giải cục bộ có liên quan. Mục đích của bài toán này là phân vùng các nút của đồ thị thành các phần bằng nhau để tổng trọng số của các cạnh được tạo từ các nút của phần này và các nút của phần kia là nhỏ nhất. Phép hoán chuyển toàn phần chuyển hai nút nằm trong hai phần khác nhau. Phép hoán chuyển một phần chỉ chuyển một nút từ phần này sang phần khác. Do đó hoán chuyển toàn phần không làm thay đổi số nút trong hai phần con. Do đó, một tiếp cận để ứng dụng tính đa dạng là không cho phép sử dụng các phép hoán chuyển toàn phần không hiệu quả trong một bước được chọn. Các phép chuyển một phần tất nhiên phải được kết hợp để cho phép tính khả thi được hồi phục sau khi đạt được các cấp độ của việc không thể thực hiện được. Được cài đặt một cách thích hợp, chiến lược này có hiệu quả trong việc xáo trộn thông minh lời giải hiện tại, trong khi thoát khỏi giới hạn cục bộ, để mở rộng quy trình tìm kiếm sang các vùng khác chưa được biết đến. Thực tế đã chứng minh, chiến lược này hoạt động khá hiệu quả.

Việc kết hợp các phép hoán chuyển một phần trong các phép hoán chuyển toàn phần trong ví dụ ở trên có thể được điều chỉnh bằng cách sử dụng hàm đánh giá vi phạm:

$$MoveValue' = MoveValue+d*Penalty$$

Loại tiếp cận đánh vi phạm này thường được sử dụng trong tìm kiếm Tabu, trong đó giá trị *Penalty* thường là một hàm của đại lượng tần số và *d* là một tham số độ đa dạng có thể chuyển đổi được. Giá trị *d* lớn thể hiện cho việc mong muốn đa dạng hơn (có nghĩa là các nút thay đổi các tập hợp bị đánh vi phạm thường xuyên hơn và nặng hơn để khuyến khích chọn ra các phép chuyển có liên quan đến các nút khác. Đánh vi phạm *âm* hay khích lệ, có thể được sử dụng để thúc đẩy các thành phần kém thường xuyên). Vi phạm có thể áp dụng cho các loại phép hoán chuyển cũng như là các thuộc tính của phép chuyển. Do đó, trong một pha có các phép hoán chuyển toàn phần đã bị loại khỏi lời giải, tất cả các phép chuyển này bị đánh một giá trị vi phạm lớn (với một giá trị *d* là vô cực).

Trong nhiều ứng dụng *d* được dùng để hạn chế các cạnh có khả năng bị lặp các phép chuyển. Giá trị tham số nhỏ thì cũng cho việc ràng buộc vi phạm, giá trị tham số lớn cho phép tìm kiếm qua các vùng không thể làm được. Tham số này có thể được dùng để điều khiển lượng ngẫu nhiên trong các phiên bản xác suất của tìm kiếm Tabu.

2.6.2.2. Khởi động lại

Thông tin tần số có thể được sử dụng trong nhiều cách để thiết kế kỹ thuật khởi động lại trong tìm kiếm Tabu. Ví dụ trong bài toán sắp công việc, tần số chung của các công việc chiến các vị trí có thể được sử sụng để tạo một quy trình tìm kiếm và tạo ra điểm khởi tạo mới. Giả sử luật ngày phải hoàn thành công việc sớm nhất được sử dụng để khởi tạo một lời giải ban đầu. Luật này gán độ ưu tiên cao nhất cho công việc có ngày phải hoàn thành sớm nhất.

Ví dụ ta có các công việc với thời điểm phải hoàn thành và thời gian thực hiện việc các công việc như sau:

Bảng 2.2: Bài toán sắp công việc

Nội dung	**Dãy**
Công việc	(1, 2, 3, 4, 5, 6)
Thời gian thực hiện	(6, 4, 8, 2, 10, 3)
Ngày phải hoàn thành	(9, 12, 15, 8, 20, 22)

Lời giải ban đầu là (4, 1, 2, 3,5, 6). Một dãy công việc khởi động lại có thể được tạo ra bằng việc sử dụng thông tin tần số để điều chỉnh chỉ số ưu tiên của mỗi công việc. Trong trường hợp này, các chỉ số ưu tiên đơn giản chính là ngày phải hoàn thành công việc và có thể được điều chỉnh bằng tham số Tần số thích hợp.

$$PrioryIndex' = PrioryIndex + d * FrequencyMeasure$$

Quá trình thực hiện được thay đổi qua giá trị *PrioryIndex'*. Với bài toán sắp công việc trên, giá trị *FrequencyMeasure* cho một công việc tại bước *Iter* có thể là phần trăm thời gian mà công việc có thể được thực hiện đúng hẹn, xét với tất cả các lời giải từ lần khởi động cuối cùng đến điểm *Iter* hiện tại. với mục đích đa dạng hóa, ta muốn giảm độ ưu tiên của các công việc mà có

giá trị *FrequencyMeasure* cao và tăng độ ưu tiên cho các công việc có giá trị *FrequencyMeasure* thấp. Giả sử các giá trị *FrequencyMeasure* cho các công việc trong ví dụ này trong một pha khởi động lại được mô tả trong bảng sau:

Bảng 2.3 : Khởi động lại bài toán sắp việc

Công việc	**Chỉ số ưu tiên (PrioryIndex)**	**Giá trị tần số (FrequencyMeasure)**	**Chỉ số ưu tiên (PrioryIndex')**
1	9	0.01	9.1
2	12	0.23	14.3
3	15	0.83	23.3
4	8	0.13	9.3
5	20	0.31	23.1
6	22	0.93	31.3
Giá trị *d*=10			

Khi đó dãy công việc được khởi động lại là (1, 4, 2, 5, 3, 6)

Loại quy trình này có thể được kết hợp nhiều loại ứng dụng và thấy được tính hữu dụng lớn với các thông tin về tần số được ghi lại dễ dàng và khá nhiều các bước được thực hiện.

Cũng có thể có các dạng khác của chức năng bộ nhớ khi mà các kỹ thuật khởi động lại được cài đặt. Như một dạng cơ bản mà ta đã xem qua với *bộ nhớ sự kiện then chốt (Critical Event Memory)*. Việc sử dụng loại bộ nhớ này rất quan trọng trong các thiết kế tìm kiếm Tabu, vì trong các trường hợp này điểm bắt đầu thường đưa ra cùng hướng tìm kiếm.

Thường việc đa dạng hóa thực hiện giống với ngẫu nhiên. Chắc chắn việc giới thiệu một yếu tố ngẫu nhiên để đạt được một hiệu ứng đa dạng hóa là một nền tảng mở rộng giữa các quy trình tìm kiếm, và là hoạt động cơ bản

của giải thuật Luyện thép và giải thuật Di truyền. Từ khía cạnh trừu tượng thì không có gì sai với việc coi ngẫu nhiên và đa dạng hóa là như nhau, nhưng với việc mở rộng ra, việc đa dạng hóa bao hàm các sự khác nhau giữa các thành phần trong một tập hợp, và việc mở rộng dùng để xây dựng lên các khác biệt có liên quan đến chiến lược tìm kiếm, thì việc sử dụng phổ biến của ngẫu nhiên là một vỏ bọc thuận lợi cho những thứ khá khác biệt.

Ví dụ, khi ngẫu nhiên được dùng như một phần của kỹ thuật khởi động lại, các thông tin tần số được dùng để ước lượng xác suất phân phối để thay đổi quy trình khởi động. Theo cách này thì ngẫu nhiên không hẳn là ngẫu nhiên mà được dẫn đường bởi quy trình tìm kiếm.

2.7. Dao động chiến lược

Dao động chiến lược *(Strategic Oscillation)* có các liên kết gần với các phiên bản gốc của tìm kiếm Tabu và cung cấp một phương tiện để đạt được một ảnh hưởng lẫn nhau hiệu quả giữa *tăng cường* và *đa dạng* trên bộ nhớ từ trung bình tới dài hạn.

Dao động chiến lược thực hiện bằng việc hướng các phép chuyển có liên quan với nhau tới một mức then chốt, khi được xác định bởi một giai đoạn của việc khởi tạo hay một khoảng được chọn giữa các giá trị. Mức then chốt hay *‘‘biên dao động’’ (Oscillation Boundary)* này thường được thể hiện bằng một điểm mà thường phương thức tìm kiếm dừng lại. Tuy nhiên, thay vì dừng khi đạt tới biên, các luật cho việc lựa chọn các phép chuyển được thay đổi, để cho phép vùng được định ra bởi mức then chốt có thể được sử dụng. Tiếp theo tiếp cận này sẽ xử lý đối với chiều sâu bên ngoài biên dao động và quay ngược lại. Biên dao động được đạt tới và vượt qua, nhưng lúc này là với hướng ngược lại và phương thức xử lý điểm mới vừa chuyển này.

Quy trình này lặp đi lặp lại việc đạt tới rồi vượt qua giới hạn theo mức then chốt từ các hướng khác nhau tạo ra một hành vi dao động. Việc điều khiển trên hành vi này được xây dựng bằng cách tạo ra các giá trị ước lượng và luật của phép chuyển, dựa trên các vùng được xem xét và chiều của việc tìm kiếm. Khả năng của việc quay lui bị bỏ qua do kỹ thuật tìm kiếm Tabu chuẩn, giống như những gì được xây dựng trên các chức năng của bộ nhớ *Recency – based* và *Frequency – based*.

Một ứng dụng của chiến lược này được áp dụng vào bài toán lý thuyết đồ thị khi mà các mức then chốt thể hiện một hình thức của cấu trúc đồ thị, có khả năng được tạo ra bởi việc thêm vào liên tục các thành phần cơ bản, như cạnh, nút hay đồ thị con. Chiến lược dao động khi áp dụng vào bài toán này đưa ra kết quả bằng một quá trình khởi tạo với việc đưa ra các thành phần cho đến khi đạt cấp then chốt, và sau đó cung cấp các thành phần để vượt qua giới hạn mà mức then chốt đã tạo ra. Lời giải hiện tại có thể thay đổi cấu trúc một lần khi đạt tới giới hạn, và do đó một tập các lời giải liên quan cũng phải được tạo ra, dựa trên các luật đã thay đổi trong việc chọn các phép chuyển. Các luật cũng thay đổi để xử lý theo hướng ngược lại, gỡ các thành phần cho đến khi tạo lại cấu trúc tại mức then chốt.

Dao động chiến lược có thể được xem là sự kết hợp của hai tập hợp quy định, một là ở cấp vĩ mô và một ở cấp vi mô.

Bảng 2.4 : Các quyết định dao động chiến lược

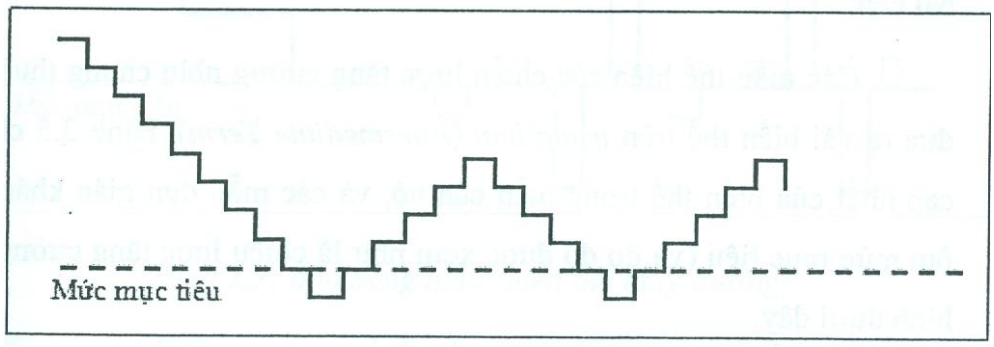
Cấp độ quyết định vĩ mô
1. Chọn các hàm hướng dẫn dao động.
2. Chọn một mức mục tiêu cho hàm này.
3. Chọn một mẫu cho việc dao động.
Cấp độ quyết định vi mô

1. Chọn một tỉ lệ mục tiêu của việc thay đổi (cho việc di chuyển tới hoặc lui từ mức mục tiêu).
2. Chọn một nhóm mục tiêu thay đổi.
3. Định ra chuẩn tham vọng để vượt qua giới hạn mục tiêu.

Quyết định mức vĩ mô:

Hàm hướng dẫn dao động của quyết định mức vi mô thứ nhất tương ứng với thành phần được điều khiển, có nghĩa là giới hạn Tabu, vi phạm không có khả năng hoặc hàm mục tiêu (dễ thấy hơn, hàm này cung cấp một độ đo của thành phần này cho phép quyền điều khiển được dựng nên). Trong quyết định mức vi mô thứ hai, một biến thể của các mức mục tiêu là có khả năng, và dạng của chúng thường được quyết định bởi các thiết lập của bài toán. Ví dụ, trong một quy trình của việc thêm hoặc xóa một cạnh trong đồ thị, thì hàm mục tiêu có thể là giai đoạn mà tập hợp hiện tại các cạnh tạo ra một *cây bao trùm (Spanning tree)*. Tương tự trong một giai đoạn của việc thêm hoặc bỏ một công việc, mục tiêu có thể là giai đoạn thực hiện xong việc gán công việc. Mỗi loại trong số đó có thể được chuyển dạng tùy theo các trường hợp đặc biệt của hoàn cảnh mà mục tiêu dựng lên một biên của khả năng, được tiếp cận từ bên trong hay bên ngoài.

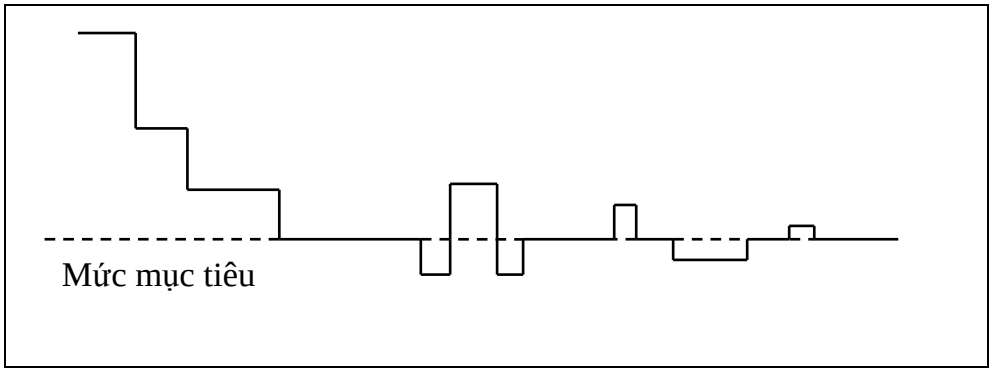
Mẫu của dao động ở mức vĩ mô có các đặc tính như độ sâu mà quá trình tìm kiếm vượt qua theo hướng đưa ra, và tổng quát hơn con đường đó có độ sâu thay đổi theo thời gian. Một ví dụ của loại dao động đơn giản này được thể hiện trong hình dưới đây, lúc này quá trình tìm kiếm bắt đầu bằng việc chạm mục tiêu *từ trên xuống* và các dao động với biên độ cố định từ đó về sau.



Hình 2.4: Dao động chiến lược đơn giản

Mẫu trong hình 2.4 được thể hiện như các đường gãy khúc hơn là một đường cong, để chỉ ra rằng việc tìm kiếm không biến đổi liên tục từ cấp này đến cấp tiếp theo, như có thể giữ trong một bước tại một mức nào đó. Đường đứt nét đại diện cho mức mục tiêu có thể được xem như là chiều thời gian. Tương tự, mỗi đoạn của đường dao động cũng có thể được hình thành *hẹp* hay *rộng*.

Một loại của mẫu dao động thường thực hiện trong các chiến lược tìm kiếm Tabu *ngắn hạn* tiến sát đến mục tiêu, trong nhiều trường hợp, làm chậm tỉ lệ của việc tiếp cận và mất thêm một thời gian tại các cấp nằm gần vùng lân cận của mục tiêu. Khi kết hợp với mục tiêu một khi nó đạt được, mẫu này là một phiên bản của một *chiến lược tăng cường*. Như hình dưới đây.



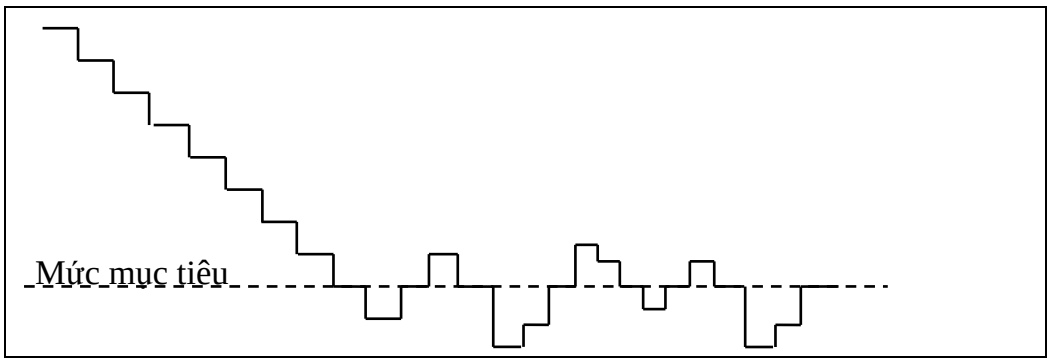
Hình 2.5 : Dao động mẫu (tăng cường)

Trong cả hai hình 2.4 và 2.5, các dao động xảy ra trên và dưới mức mục tiêu có thể bị thay thế bởi các dao động chỉ ở một bên của mức tiêu thôi.

Ví dụ, với các khoảng thời gian dài hơn, mẫu này có thể chủ yếu được tập trung vào một bên và sau đó là bên còn lại, hoặc là vài bước tập trung ở một bên rồi vài bước tập trung cả hai bên.

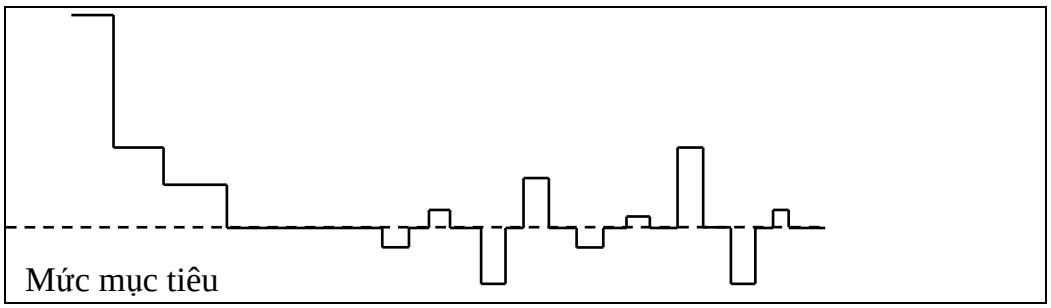
Các mẫu thể hiện các chiến lược tăng cường nhìn chung thuận lợi bằng việc đưa ra vài biến thể trên *trung hạn (Intermediate Term)*. Hình 2.5 chứa một cấp độ cao nhất của biến thể trong mẫu của nó, và các mẫu đơn giản khác, cũng hầu như ôm mức mục tiêu (và do đó được xem như là chiến lược tăng cường) như trong hai hình dưới đây.

Các mẫu tăng cường với độ biến thiên cao có thể dễ dàng được tạo ra.



Hình 2.6 : Dao động mẫu (biến thể tăng cường)

Các loại tăng cường được mô tả trong các đồ thị trước không yêu cầu phụ thuộc vào bộ nhớ, ngoại trừ việc theo dõi pha hiện tại của mẫu đang được thực thi. Tuy nhiên, việc sử dụng các cấu trúc bộ nhớ tìm kiếm Tabu tăng cường khả năng tìm kiếm và tập trung trên các vùng quan trọng chiến lược.



Hình 2.7: Dao động mẫu (biến thể tăng cường)

Các quy trình tăng cường có thể được nhúng vào trong dao động chiến lược bằng việc biến đổi các lựa chọn luật để khuyến khích việc hợp nhất của các thuộc tính cụ thể, bằng cách khóa các thuộc tính này vào trong một lời giải trong một thời gian. Các quy trình này có thể được xem như là các thiết kế cho việc sử dụng các *biến được quyết định mạnh và biến vững chắc (Strongly Determined và Consistent Variables).* Một *biến quyết định mạnh (Strongly Determined Variables)* là cái không thể thay đổi giá trị của nó trong một lời giải chất lượng cao không làm giảm giá trị nghiêm trọng hoặc tính khả thi, trong khi một *biến bền vững(Consistent Variables)* là cái thường xuyên giữ các giá trị khác nhau (hoặc là một khoảng giới hạn cao của giá trị) trong các lời giải tốt. Sự phát triển của các thước đo của ‘‘sức mạnh’’ và ‘‘vững chắc’’ là then chốt để sử dụng những ghi nhận này, thông thường bằng sự tính toán cho việc cân bằng được quyết định bởi ngữ cảnh. Tuy nhiên, việc sử dụng bộ nhớ Frequency – based để theo dõi độ vững chắc, đôi khi được đánh trọng bởi thành phần của chất lượng và ảnh hưởng, cung cấp các phương thức với hiệu suất tốt.

Các quy trình dài hạn *(Longer Term)*, có thể đưa ra các chiến lược đa dạng vào trong mẫu dao động. Khi dao động được dựa trên quá trình khởi động và hủy, ứng dụng được lặp lại của các pha khởi tạo chứa một loại dao động mà tương tự như phương thức khởi động lại. Trong các thể hiện này thì điểm khởi động lại luôn luôn giống nhau (có nghĩa là một trạng thái null) thay vì chứa các lời giải khởi tạo khác nhau, và do đó điều quan trọng là sử dụng luật được chọn các biến thể để đảm bảo đa dạng hóa thích hợp.

Một kết nối có thể được nhận ra giữa một siêu phiên bản dao động chiến lược và nột lớp các quy trình được biết như hướng tiếp cận *làm xáo trộn (Pertubation).*

Các phương pháp làm xáo trộn có thể được xem như các quy trình có cấu trúc lỏng lẻo cho việc khích lệ dao động, mà không tham khảo đến việc tăng cường và đa dạng hóa và những chiến lược có liên quan. Tương tự, các phương thức xáo trộn không được thiết kế để sử dụng việc cân bằng với các biến thể giới hạn trong các thành phần như là các loại khác nhau của tính khả thi, độ đo của sự đổi chỗ từ các phía khác nhau của các biên. Tuy nhiên, tại cấp đầu tiên của xấp xỉ, các phương thức xáo trộn tìm kiếm mục tiêu tương tự với những gì được theo đuổi bởi dao động chiến lược.

Con đường tìm kiếm được theo bởi dao động chiến lược không được mô tả hoàn hảo trong hình 2.7, cho tới đó con đường tìm kiếm sẽ không tuân theo các cấp giai đoạn của chức năng một cách tổng quát, mà thường sẽ nằm trong các vùng bị chồng lên nhau một phần. Hơn nữa trong các tiếp cận thông thường, các pha đa dạng hóa được liên kết với các pha tăng cường, các mẫu được mô tả về việc thoáng qua (*hover*) mức mục tiêu đôi khi được kết hợp bằng việc cũng thoáng qua các mức khác, để sử dụng một khái niệm gọi là*Nguyên lý tối ưu xấp xỉ (Proximate Optimately Principle - POP).* Theo khái niệm này, các lời giải tốt tại một mức có thể thường được tìm thấy gần với các lời giải tốt tại một mức sát ngay đó.

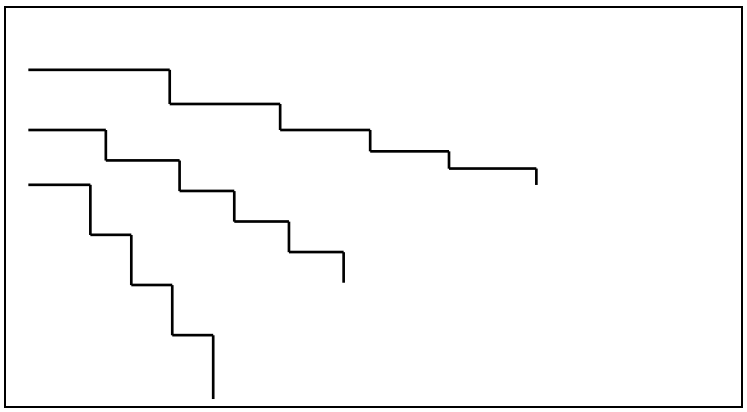
Khái niệm POP thúc đẩy một chiến lược tìm kiếm Tabu gọi là *nối lại đường (Path Relinking)*, cung cấp một phương thức quan trọng cho việc cải thiện kết quả của dao động chiến lược. Nối lại đường đưa ra một cách để cải thiện dao động chiến lược bằng việc chọn ra các lời giải tốt từ các mức khác

nhau, và kết hợp chúng lại bằng các con đường mới đã đi qua mức mục tiêu. Các lời giải tốt từ chính mức mục tiêu cũng cung cấp một nền tảng cho việc xử lý nối lại đường để bắt đầu những cuộc tìm kiếm mới tại mức này.

Khái niệm POP gợi ý một hình thức của việc kết nối cho không gian tìm kiếm mà có thể được sử dụng hữu ích bằng tiếp cận này. Có nghĩa là, các con đường nối lại đường, được chúng hướng dẫn bởi các lời giải tốt, dù là đoán trước hay xác suất, cũng gần như đi qua các vùng mà các lời giải tốt ở đó, dùng tập lời giải lân cận thích hợp đã cung cấp.

Quyết định mức Vi mô:

Quyết định lựa chọn một tỉ lệ mục tiêu của sự thay đổi cho dao động chiến lược được đặt ở mức vi mô vì nó hiển nhiên liên quan đến tính biến thiên của một hình thức cục bộ cụ thể. Hình 2.8 mô tả tỉ lệ thay đổi khác nhau, nơi mà hướng hiện tại là *giảm dần*. Một lần nữa, những thay đổi được biểu diễn bằng các đường gãy khúc, nhưng chú ý rằng chúng không luôn luôn chạy mượt hoặc đều. Vấn đề của việc sử dụng các tỉ lệ mục tiêu như các thành phần của chiến lược tìm kiếm có thể xem như là hiển nhiên, nhưng nó quan trọng là thường bị bỏ qua các loại Framework tìm kiếm khác.



Hình 2.8 : Tỉ lệ mục tiêu của sự thay đổi

Khi tham số dao động có liên quan đến các giá trị của hàm mục tiêu, tìm kiếm Tabu thường quy định các thay đổi chủ động, tìm kiếm sự cải thiện nhiều nhất hoặc ít đi xuống nhất có khả năng. Tiếp cận này áp dụng chủ yếu vào các pha tăng cường và có thể được làm dịu đi hay thậm chí bị đảo ngược trong pha đa dạng hóa. Rõ ràng hơn, như việc nhất mạnh trước đó, khái niệm của việc tìm kiếm Tabu bằng việc định ra ý nghĩa của các biến thể tốt nhất trong các hiệu chỉnh và các pha tìm kiếm khác, nơi mà các tỉ lệ thay đổi tạo thành một trong các thành phần của biến thể này.

Kết hợp tỉ lệ mục tiêu của thay đổi là quyết định mức vi mô của việc chọn một ban mục tiêu của thay đổi, cái mà quy định các biên độ trên các mức lệnh từ tỉ lệ mục tiêu. Các đoạn thẳng trong hình 2.8 được xem là các đoạn bằng nhau, và ban mục tiêu của thay đổi được đưa ra để điều khiển chiều rộng này.

Các tỉ lệ mục tiêu và các ban không được quyết định độc lập với sự hiểu biết về các lời giải có thể sử dụng được, nhưng dựa trên việc khám phá các lời giải có thể liên quan hiện tại để quyết định các khả năng có được (do đó các chỗ ngoặc của hình 2.8 được tạo ra do quyết định từ các khả năng này).

Cuối cùng, tiêu chuẩn tham vọng tại mức vi mô cho phép sự điều khiển chỉ ra trước đó bị bỏ qua nếu một cái khác thay thế hiệu quả hơn xuất hiện, như là một phép chuyển dẫn đến lời giải mới tốt, hoặc tới một lời giải tốt nhất tại mức dao động hiện tại.

2.8. Nối lại đường

Một tích hợp hữu ích của chiến lược tăng cường và đa dạng xảy ra trong tiếp cận được gọi là *nối lại đường*. Tiếp cận này tạo ra các lời giải mới bằng cách sử dụng các con đường đến các lời giải tốt – bằng cách bắt đầu từ

một trong các lời giải đó, được gọi là *lời giải khởi tạo (Initial Solution)*, và tạo ra con đường trong không gian các lời giải có liên quan để dẫn đến các lời giải khác gọi là *lời giải dẫn đường (Guilding Solution)*. Điều này được thực hiện bằng việc chọn các phép chuyển cung cấp các thuộc tính trong các lời giải dẫn đường.

Tiếp cận này được xem như là một siêu thể hiện (độ tập trung cao) của một chiến lược tìm kiếm để kết hợp các thuộc tính của các lời giải chất lượng, bằng việc tạo ra các khích lệ để thu hút các thuộc tính trong các phép chuyển được lựa chọn. Tuy nhiên, thay thế việc sử dụng một sự khích lệ hiếm khi khuyến khích tập các thuộc tính này, tiếp cận nối lại đường phụ thuộc vào tất cả xem xét tới mục tiêu của việc lựa chọn các phép chuyển để đưa ra các thuộc tính của các lời giải dẫn đường (guilding solution), để tạo ra một *thuộc tính tổng hợp* tốt trong lời giải hiện tại. Thuộc tính tổng hợp này tại mỗi bước được quyết định bởi việc lựa chọn phép chuyển tốt, sử dụng các tiêu chuẩn lựa chọn thông thường, từ các tập bị giới hạn của các phép chuyển kết hợp với một số tối đa (hay một giá trị trọng số tối đa) của các thuộc tính trong các lời giải dẫn đường. Như trong các ứng dụng khác của tìm kiếm Tabu, tiêu chuẩn mong đợi *(Aspiration Criteria)* có thể vượt qua giới hạn này để cho phép các phép chuyển khác với chất lượng cao hơn được xem xét.

Đặc biệt, trên việc xác định một tập hợp của một hoặc nhiều các lời giải để hướng dẫn con đường của một lời giải được chọn, những thuộc tính của các *lời giải hướng dẫn* được gán trọng số ưu tiên khi các khích lệ được lựa chọn. Các trong số lớn hơn được gán vào các thuộc tính xẩy ra với số lần lớn hơn số lời giải hướng dẫn, cho phép xu hướng để tăng độ chú ý đến các lời

giải với chất lượng cao hơn hoặc với các đặc biệt (có nghĩa là bổ sung cho những thuộc tính của lời giải được khởi tạo con đường mới).

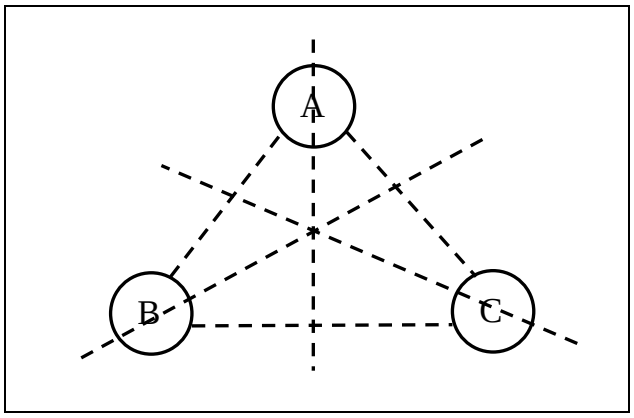
Tổng quát hơn, không cần thiết để một thuộc tính xảy ra trong một lời giải hướng dẫn để mà có một tình trạng tốt. Trong vài thuộc tính điều chỉnh có thể chia sẻ độ giống nhau và trong trường hợp này nó có thể hữu ích để xem một lời giải vector như việc ‘‘ủng hộ’’ để khuyến khích các thuộc tính cụ thể. Thường các hình thức mạnh mẽ nhất của tiêu chuẩn tham vọng được tin tưởng để vượt qua loại này của luật lựa chọn.

Trong một tập hợp các lời giải tốt, vai trò của việc khởi tạo lời giải và dẫn đường các lời giải có thể được thay thế qua lại. Sự khác biệt giữa khởi tạo lời giải và dẫn đường lời giải có thể loại bỏ hiệu quả trong các trường hợp cụ thể. Ví dụ, một tập hợp của lời giải hiện tại có thể được tạo ra đồng thời, mở rộng ra nhiều con đường khác nhau và cho phép một lời giải khởi tạo được thay thế (bằng một lời giải dẫn đường cho những lời giải khác) bất cứ khi nào lời giải hiện tại có liên quan thỏa mãn được một tiêu chuẩn tham vọng đủ mạnh.

Vì các vai trò có thể thay đổi lẫn nhau, các lời giải khởi tạo và các lời giải hướng dẫn được gọi chung là các *lời giải tham khảo (Reference Solutions*). Nhưng lời giải tham khảo này có thể có các thể hiện khác nhau dựa trên Framework lời giải. Các điểm tham khảo có thể được tạo ra bởi bất kỳ số nào của các *heuristic* khác nhau mà kết quả là các lời giải chất lượng cao.

Một hình thức của quy trình được thể hiện trong hình sau. Tập hợp được chọn của các lời giải tham khảo bao gồm ba thành viên A, B, C. Các con đường được tạo ra bằng cách phép mỗi cái đáp ứng thư như lời giải khởi tạo. Các con đường thể hiện bằng đường thẳng dĩ nhiên là quá đơn giản, từ khi chọn giữa các phép chuyển có khả năng trong các lời giải có liên quan sẽ tạo ra một con

đường phức tạp hơn. Tăng cường có thể đạt được bằng cách tạo ra các đường từ các lời giải tương tự, trong khi đa dạng hóa được dùng để tạo ra các con đường từ những lời giải không giống nhau. Tiêu chuẩn tham vọng thích hợp cho phép độ chênh lệch từ các con đường ở các lời giải lân cận hấp dẫn.

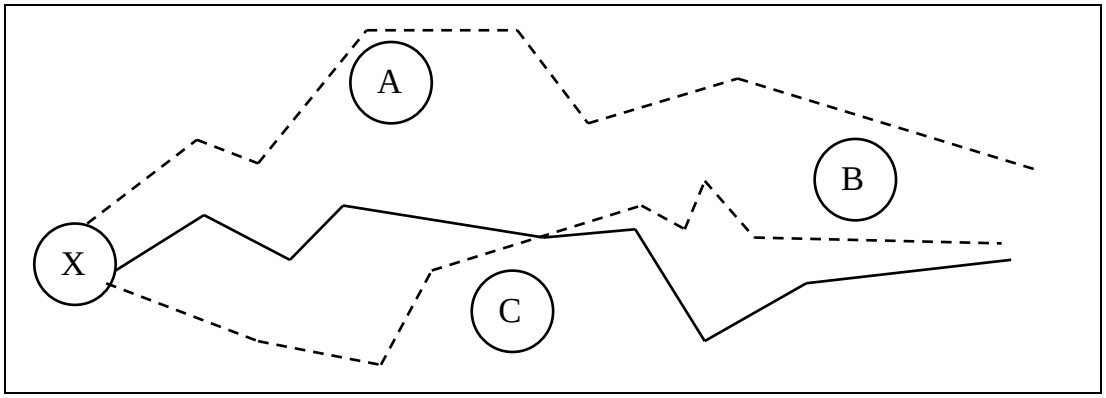


Hình 2.9 : Nối lại đường trong không gian các lời giải liên quan

Như hình 2.9 mô tả, ít nhất một con đường tiếp tục được cho phép vượt qua mỗi lời giải khởi tạo/dẫn đường. Sự tiếp tục này có thể được thực hiện bằng việc đánh giá vi phạm các tập hợp thuộc tính bị gỡ ra trên con đường tìm kiếm, bao gồm các thuộc tính của lời giải dẫn đường mà có thể bị bắt buộc phải gỡ để mà có thể tiếp tục con đường (một lời giải khởi tạo có thể được đẩy ra khỏi các lời giải dẫn đường bằng việc đánh giá vi phạm tập hợp các thuộc tính của chúng từ lúc ban đầu).

Các vùng có triển vọng được tìm kiếm kỹ hơn trong kỹ thuật nối lại đường (Path relinking) bằng việc thay đổi trọng số gắn trên các thuộc tính của các lời giải dẫn đường và bằng việc thay đổi xu hướng liên quan với chất lượng lời giải và các đặc tính của lời giải được chọn. Hình 2.10 mô tả loại của biến thể có thể là kết quả, nơi mà các điểm X thể hiện là một lời giải khởi tạo, các đường đứt nét, chấm chấm và liền nét là các con đường tìm kiếm khác nhau. Các biến thể của loại này trong một vùng có triển vọng được thúc đẩy

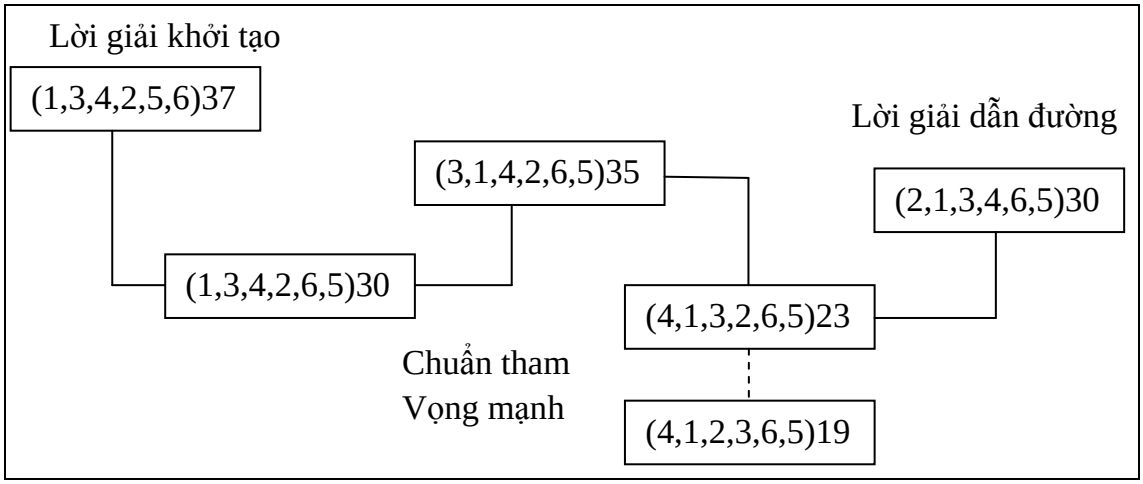
bởi nguyên lý tối ưu xấp xỉ trong kết nối với dao động chiến lược. Việc các lựa chọn thích hợp của các điểm tham khảo (và các lời giải liên quan để tạo ra các con đường từ chúng), nguyên lý này đề nghị những điểm tốt có thể được tìm trong những vùng mà các con đường đi qua, sau đó bắt đầu tìm kiếm các điểm chất lượng cao trên các đường này.



Hình 2.10: Nối lại đường bằng thuộc tính thu hút

Để mô tả các khái niệm có liên quan với nối lại đường, chúng ta sẽ lấy lại ví dụ bài toán sắp công việc từ các phần trước. Lời giải khởi tạo với tổng trọng số công việc phải làm là 37, được dùng để bắt đầu một quy trình nối lại đường đến lời giải dẫn đường (2, 1, 2, 4, 6, 5). Các phép hoán đổi được chọn để tạo ra một con đường ngắn từ một lời giải này đến một lời giải khác. Trong trường hợp này, thông thường các nhóm nghiên cứu tìm kiếm Tabu định nghĩa một khái niệm *điểm số (point)* đơn giản cho lời giải hiện tại bằng cách đếm số lượng công việc đang ở trong cùng vị trí tuyệt đối so với lời giải hướng dẫn. Điểm số của lời giải khởi tạo lúc đó bằng 0. Sau phép chuyển đầu tiên, công việc 5 và 6 được đặt vào các chỗ chính xác nên điểm số lời giải thứ hai là 2. Điểm số của hai lời giải sau theo thứ tự có điểm là 3, 4. Tại thời điểm này, như trong hình dưới, một lời giải với một giá trị hàm mục tiêu tốt hơn các lời giải trước đó được tìm thấy. Do đó sử dụng một *tiêu – chuẩn – tham –*

vọng – cải – thiện – tốt – nhất (Improved Best Aspiration Criterion) có thể cho phép con đường lệch khỏi mục tiêu chính. Trên việc ước lượng và lưu trữ lời giải, quy trình có thể tiếp tục tìm những lời giải tốt trên con đường hay hướng con đường của nó tới lời giải dẫn đường, trước khi quay trở lại tìm kỹ hơn trong vùng có các lời giải tiềm năng trên con đường tìm kiếm trước đó.



Hình 2.11 : Ví dụ nối lại đường

Các kỹ thuật nối lại đường có thể phát triển cho các loại ứng dụng khác bằng việc tạo ra các quy trình hướng dẫn phù hợp.

2.8.1. Vai trò của tăng cường và đa dạng hóa

Nối lại đường thường đi với dao động chiến lược, đưa đến một nền tảng tự nhiên cho việc phát triển chiến lược tăng cường và đa dạng hóa. Các chiến lược tăng cường trong hoàn cảnh này lựa chọn các lời giải tham khảo làm lời giải tốt, các lời giải đó nằm trong vùng phổ biến hoặc chia sẻ các đặc tính phổ biến. Tương tự các chiến lược đa dạng dựa trên kỹ thuật nối lại đường về bản chất chọn các lời giải tham khảo từ các vùng khác nhau hoặc cho thấy các đặc tính trái ngược. Các chiến lược đa dạng hóa cũng thường chú ý trên các con đường vượt qua các điểm tham khảo. Tập hợp của các điểm tham khảo chứa

trong đó các điều kiện có thể quyết định hữu ích bằng việc phân vùng và các phương thức phân tích điều kiện.

2.8.2. Kết hợp các lời giải liên quan

Các chiến lược nối lại đường trong tìm kiếm Tabu có thể tốt bằng việc tận dụng tập lời giải liên quan khác nhau và các định nghĩa thuộc tính lơn là những cái sử dụng bằng *heuristic* cho việc tạo ra các lời giải tham khảo. Ví dụ, đôi khi có thuận lợi khi sử dụng một khởi tạo tập lời giải có liên quan cho việc nối lại đường, có nghĩa là các lời giải liên quan nào đó cho phép một lời giải được tạo ra trong dãy các bước khởi tạo. Trong trường hợp này, quy trình khởi tạo có thể được sử dụng để đưa ra một phần khởi tạo bắt đầu, bằng việc xác định các thuộc tính cụ thể như là nền tảng cho các bước khởi tạo còn lại. Tương tự, nối lại đường có thể sử dụng các lời giải có liên quan bị hủy, nơi mà các lời giải khởi tạo bị ‘‘quá tải’’ với các thuộc tính được đưa ra bởi các lời giải hướng dẫn, và các thuộc tính này bị thay đổi cho đến khi đạt đến một tập hợp với một kết cấu thích hợp.

Khi nối lại đường được dựa trên các *lời giải khởi tạo có liên quan (Contructive Neighborhood)*, các lời giải dẫn đường cung cấp cung cấp thuộc tính quan hệ đưa ra các lựa chọn cho các giai đoạn tiếp theo sau của việc khởi tạo. Hơn nữa, một khởi tạo đầy đủ có thể được tạo ra bằng việc tạo một lời giải khởi tạo là lời giải null *(Null Solution)*. Việc hủy cao nhất bắt đầu từ một ‘‘tập hoàn chỉnh’’ của các phần tử trong lời giải. Các tiết cận khởi tạo hoặc hủy khác với tiếp cận chuyển đổi bằng việc tạo ra chỉ một lời giải mới thay vì một dãy các lời giải, trên mỗi con đường từ lời giải khởi tạo đến các lời giải bên ngoài nếu một *tập lời giải chuyển tiếp liên quan (Transition Neighborhood)* không được sử dụng mở rộng *tập lời giải khởi tạo có liên quan (Contructive Neighborhood)*.

Tập lời giải khởi tạo có liên quan có thể được xem như là trường hợp đặc biệt của khả năng tái tạo lại các lời giải liên quan (Restoring Neighborhood), từ lời giải null hoặc là lời giải đã xây dựng một phần không thỏa mãn tất cả các điều kiện để đánh giá là khả thi. Tương tự *tập lời giải hủy có liên quan* có thể cũng đại diện một thể hiện của một hàm hồi phục tính khả thi, khi quá nhiều các phần tử có thể vi phạm ràng buộc.

Tóm tắt từng bước của kỹ thuật *nối lại đường*:

- B1: Xác định cấu trúc của các lời giải có liên quan và các thuộc tính lời giải có liên quan cho việc nối lại đường (*Path Relinking*) có thể khác nhau tùy vào chiến lược tìm kiếm Tabu áp dụng vào các bài toán.
- B2: Xác định một tập hợp của hai hay nhiều các lời giải, và định ra thành viên nào sẽ được xem như là lời giải khởi tạo và các lời giải dẫn đường (các lời giải tham khảo có thể không khả thi, như là các thành phần lời giải không hoàn chỉnh hay quá tải được đối xử như các lời giải khởi tạo hay hủy có liên quan).

Phép chuyển từ lời giải khởi tạo đến hay vượt qua các lời giải dẫn đường tạo ra một hay nhiều các lời giải trung gian như các ứng viên để khởi tạo các nỗ lực giải quyết vấn đề về sau (nếu pha đầu tiên của bước này tạo ra một giải pháp không khả thi, áp dụng pha thứ hai có liên quan với một *tập – lời – giải – khôi – phục – khả - thi – có – liên – quan (Feasibility Restoring Neighborhood)).*

CHƯƠNG 3: ỨNG DỤNG THUẬT TOÁN TABU SEARCH VÀO BÀI TOÁN NGƯỜI DU LỊCH

3.1. Lịch sử bài toán người du lịch

Bài toán người du lịch (hay có tên khác là bài toán người bán hàng) (tiếng Anh: Travelling Salesman Problem - TSP) là một bài toán NP-Hardness thuộc thể loại tối ưu tổ hợp được nghiên cứu trong lý thuyết khoa học máy tính. Nội dung bài toán có thể hiểu khái quát như sau: Cho trước một danh sách các thành phố và khoảng cách giữa chúng, tìm chu trình ngắn nhất đi qua tất cả các thành phố đúng 1 lần.

Bài toán được nêu ra lần đầu tiên năm 1930 và là một trong những bài toán được nghiên cứu sâu nhất trong tối ưu hóa. Nó thường được dùng làm thước đo cho nhiều phương pháp tối ưu hóa. Mặc dù bài toán rất khó giải trong trường hợp tổng quát, có nhiều phương pháp giải chính xác cũng như heuristic đã được tìm ra để giải quyết một số trường hợp có tới hàng chục nghìn thành phố.

Bài toán người du lịch được định nghĩa trong thế kỉ 19 bởi nhà toán học Ireland William Rowan Hamilton và nhà toán học Anh Thomas Kirkman. Trường hợp tổng quát của TSP có thể được nghiên cứu lần đầu tiên bởi các nhà toán học ở Vienna và Harvard trong những năm 1930. Hassler Whitney ở đại học Princeton đưa ra tên bài toán người du lịch ngay sau đó. Trong những năm 1950 và 1960, bài toán trở nên phổ biến trong giới nghiên cứu khoa học ở Châu Âu và Mỹ. George Dantzig, Delbert Ray Fulkerson và Selmer M. Johnson ở công ty RAND tại Santa Monica đã có đóng góp quan trọng cho bài toán này, biểu diễn bài toán dưới dạng quy hoạch nguyên và đưa ra phương pháp mặt phẳng cắt để tìm ra lời giải. Với phương pháp mới này, họ đã giải được tối ưu một trường hợp có 49 thành phố bằng cách xây dựng một chu trình và chứng minh rằng không có chu trình nào ngắn hơn. Trong những

thập niên tiếp theo, bài toán được nghiên cứu bởi nhiều nhà nghiên cứu trong các lĩnh vực toán học, khoa học máy tính, hóa học, vật lý, và các ngành khác.

Năm 1972, Richard M. Karp chứng minh rằng bài toán chu trình Hamilton là NP-đầy đủ, kéo theo bài toán TSP cũng là NP-đầy đủ. Đây là một lý giải toán học cho sự khó khăn trong việc tìm kiếm chu trình ngắn nhất. Một bước tiến lớn được thực hiện cuối thập niên 1970 và 1980 khi Grötschel, Padberg, Rinaldi và cộng sự đã giải được những trường hợp lên tới 2392 thành phố, sử dụng phương pháp mặt phẳng cắt và nhánh cận.

Trong thập niên 1990, Applegate, Bixby, Chvátal, và Cook phát triển một chương trình mang tên Concorde giải được nhiều trường hợp có độ lớn kỉ lục hiện nay. Gerhard Reinelt xuất bản một bộ dữ liệu các trường hợp có độ khó khác nhau mang tên TSPLIB năm 1991, và nó đã được sử dụng bởi nhiều nhóm nghiên cứu để so sánh kết quả. Năm 2005, Cook và cộng sự đã giải được một trường hợp có 33810 thành phố, xuất phát từ một bài toán thiết kế vi mạch. Đây là trường hợp lớn nhất đã được giải trong TSPLIB.

Đến nay bài toán TSP vẫn được tiếp tục nghiên cứu tìm ra lời giải cho các bộ dữ liệu lớn hơn. Chẳng hạn bộ dữ liệu của nước Mĩ với 115,475 thành phố người giải ra chu trình tối ưu được trao thưởng 500$ (thông tin chi tiết tại http://www.tsp.gatech.edu/).

3.2. Phân tích bài toán

Có một người thương gia cần đi du lịch tại n thành phố. Anh ta xuất phát từ một thành phố nào đó, đi qua các thành phố khác để tham quan và trở về thành phố ban đầu. Mỗi thành phố chỉ đến một lần, và khoảng cách từ một thành phố đến các thành phố khác đã được biết trước. Hãy tìm một chu trình (một đường đi khép kín thỏa mãn điều kiện trên) sao cho tổng độ dài các cạnh là nhỏ nhất.

Dưới dạng đồ thị: Bài toán người du lịch có thể được mô hình hoá như một đồ thị vô hướng có trọng số, trong đó mỗi thành phố là một đỉnh của đồ thị còn đường đi giữa các thành phố là mỗi cách. Khoảng cách giữa hai thành phố là độ dài cạnh. Đây là vấn đề cực tiểu hoá với điểm đầu và điểm cuối là cùng một đỉnh sau khi thăm hết các đỉnh còn lại đúng một lần. Mô hình này thường là một đồ thị đầy đủ (giữa mỗi cặp đỉnh đều có cạnh). Nếu không có đường giữa hai thành phố thì có thể thêm một cạnh với độ dài đủ lớn vào đồ thị mà không ảnh hưởng đến kết quả tối ưu sau cùng.

Cho đồ thị đầy đủ n đỉnh vô hướng, có trọng số G = (V, E). Tìm chu trình v1 → v2 → … → vn → v1 với vi ∈ V, i = $\overline{1,n}$ sao cho tổng trọng số hành trình trên các cạnh (vi, vi+1) và (vn, v1) là nhỏ nhất.

Bài toán TSP thuộc lớp bài toán NP-Khó (lớp các bài toán không có giải thuật trong thời gian đa thức). Việc thực hiện liệt kê hết tất cả các chu trình là điều gần như không thể với số đỉnh lớn (đồ thị n đỉnh phải duyệt n! chu trình). Số chu trình phải duyệt tăng rất nhanh khi số đỉnh n càng lớn. Ngay với một đồ thị 100 đỉnh, việc duyệt toàn bộ cũng là điều rất khó thực hiện.

3.3. Xây dựng ứng dụng giải quyết bài toán

3.3.1. Cấu trúc dữ liệu đầu vào

Chương trình biểu diễn các thành phố bằng tọa độ của chúng, không mất tính tổng quát, ta giả sử tọa độ các đỉnh là các số nguyên.

Ví dụ biểu diễn tọa độ của 6 thành phố

1 193 255

2 214 320

3 432 72

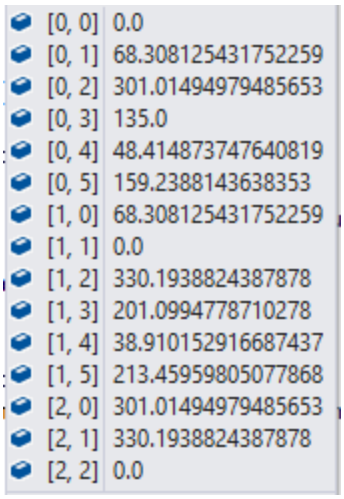
4 193 120

5 231 285

6 52 181

Trong đó mỗi dòng biểu diễn tọa độ của một thành phố, trong mỗi dòng số đầu tiên là số thứ tự, số thứ 2 là biểu diễn hoành độ và số thứ 3 biểu diễn tung độ tương ứng với tọa độ của thành phố đó.

Sau khi được xử lý, chương trình sẽ thay đổi dữ liệu đầu vào thành dạng ma trận biểu diễn khoảng cách tương ứng giữa các cặp đỉnh – được gọi là ma trận khoảng cách, ma trận này sẽ là đầu vào cho cho thuật toán Tabu Search.



Hình 3.1. Biểu diễn ma trận khoảng cách

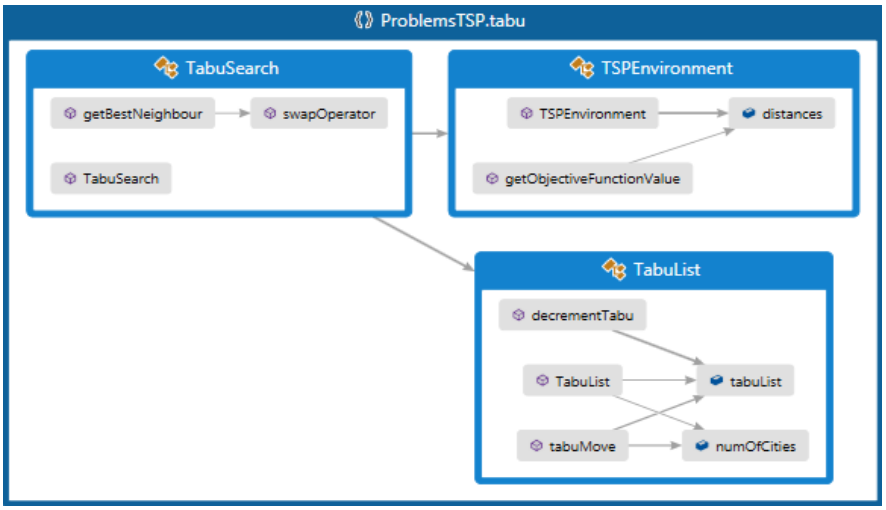
3.3.2. Cấu trúc chương trình và mối quan hệ giữa các lớp chính

Chương trình cài đặt giải quyết bài toán người du lịch sử dụng 3 thuật toán: Tìm kiếm Tabu, Luyện thép và Quay lui. Cấu trúc chi tiết các lớp chương trình như sau:

3.3.2.1. Tìm kiếm Tabu

- Lớp TSPEnvironment: Xử lý đọc dữ liệu từ tệp *txt* và chuyển dữ liệu thành dạng ma trận khoảng cách. Lớp này là lớp dùng chung cho cả ba thuật toán.

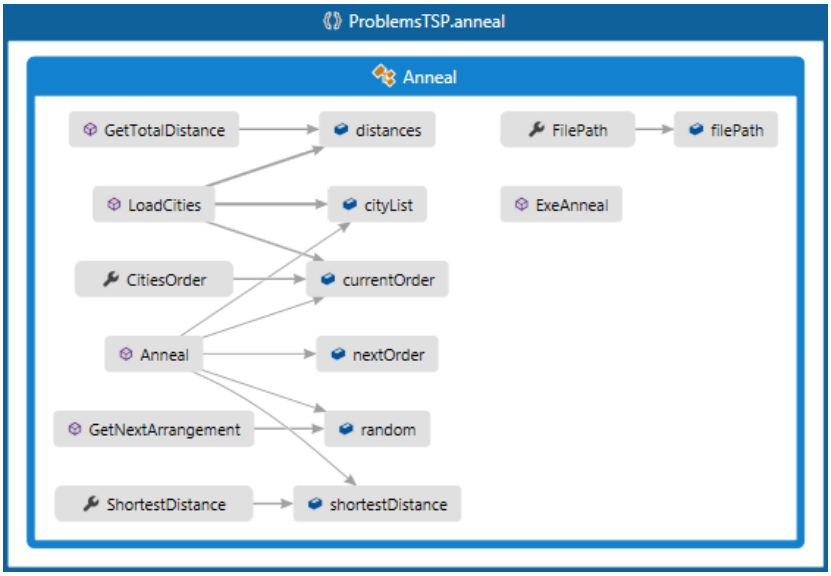
- Lớp TabuList: Lưu trữ một danh sách cấm, chứa các kết quả đã duyệt qua.
- Lớp TabuSearch: Đây là lớp chính chịu trách nhiệm xử lý phép tìm lặp ở mỗi giai đoạn của thuật toán.



Hình 3.2. Cấu trúc lớp chương trình Tabu

3.3.2.2. Luyện thép

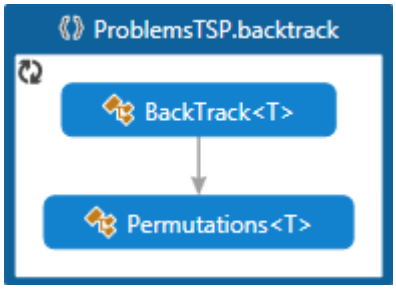
- Lớp Anneal: Lớp chính xử lý thuật toán, tại lớp này sẽ khai báo các tham số quan trọng trong giải thuật luyện thép: nhiệt độ cao nhất, tỷ lệ làm nguội, nhiệt độ thấp nhất.



Hình 3.3. Cấu trúc lớp chương trình giải thuật Luyện thép

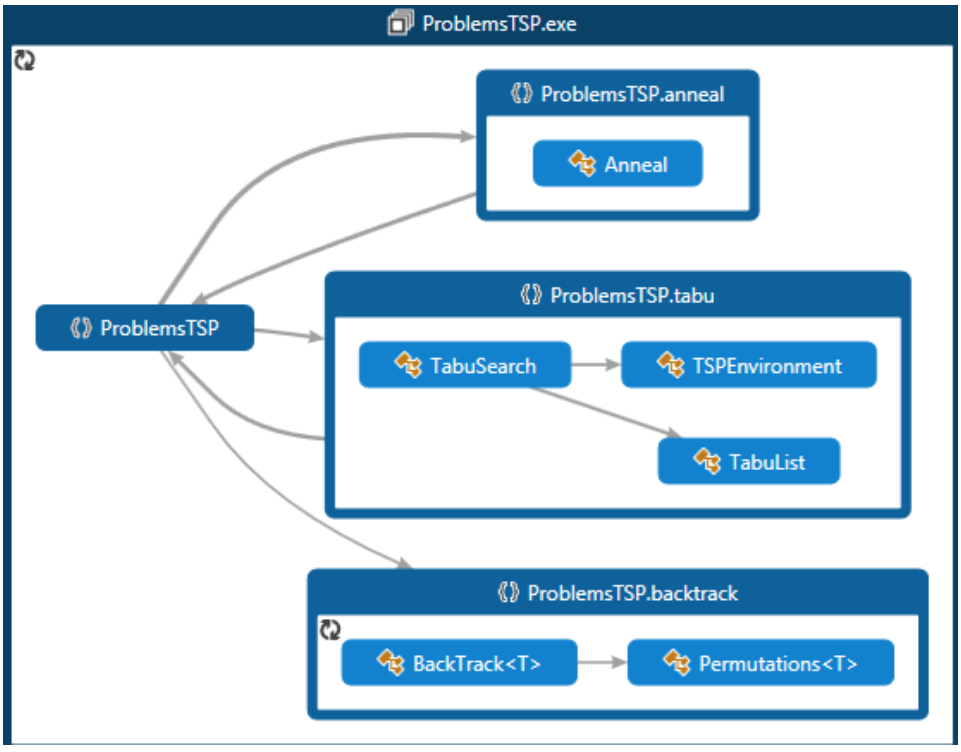
3.3.2.3. Quay lui

- Lớp Permutations: Lưu vết các đường đi tìm được.
- Lớp BackTrack: Lớp chính xử lý lời gọi đệ quy cho mỗi đáp án con trong cây đáp án.



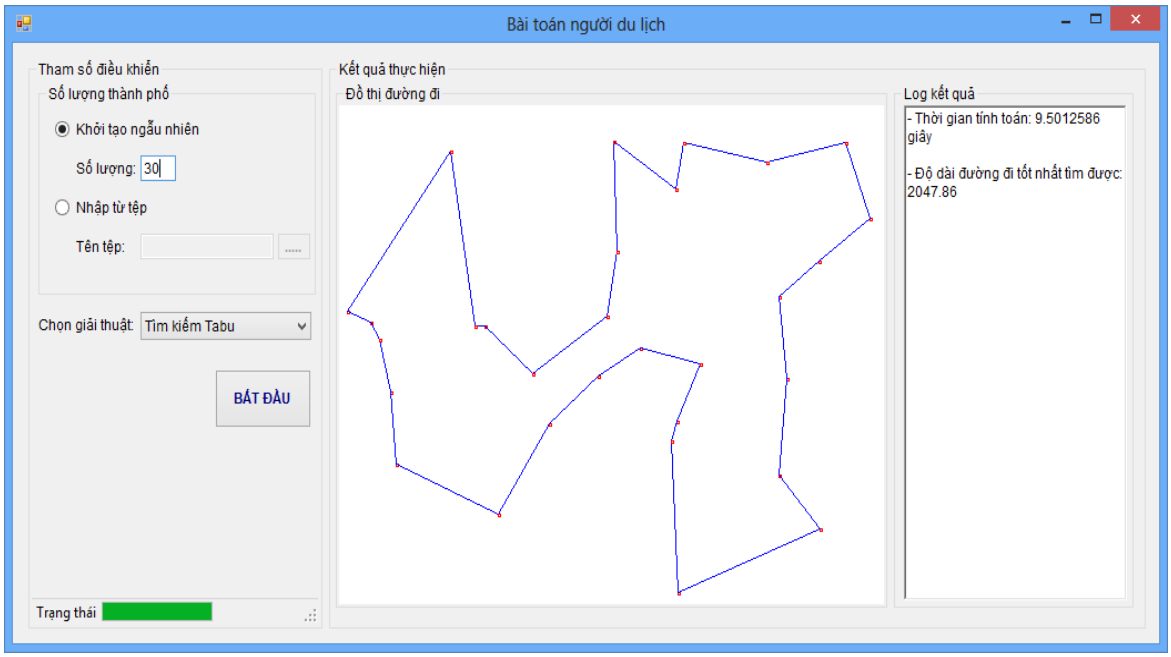
Hình 3.4. Cấu trúc lớp chương trình giải thuật Quay lui

Mối quan hệ tổng quan giữa các lớp được thể hiện như sau:

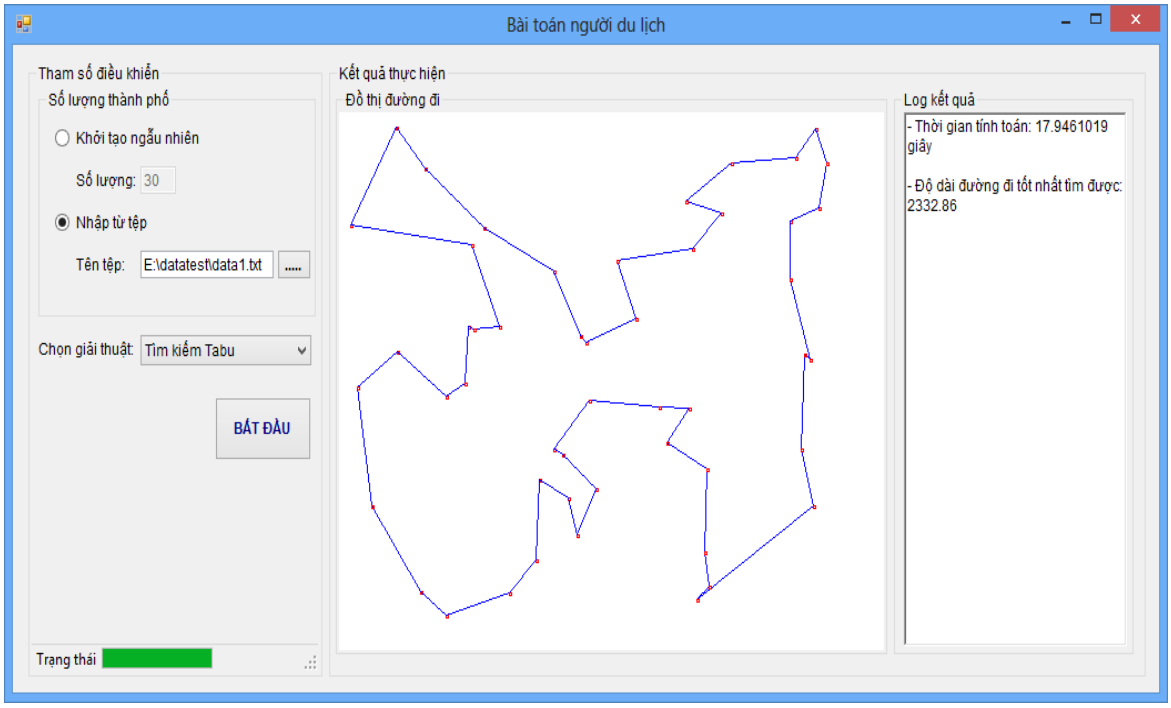


3.3.3. Kết quả khi chạy chương trình

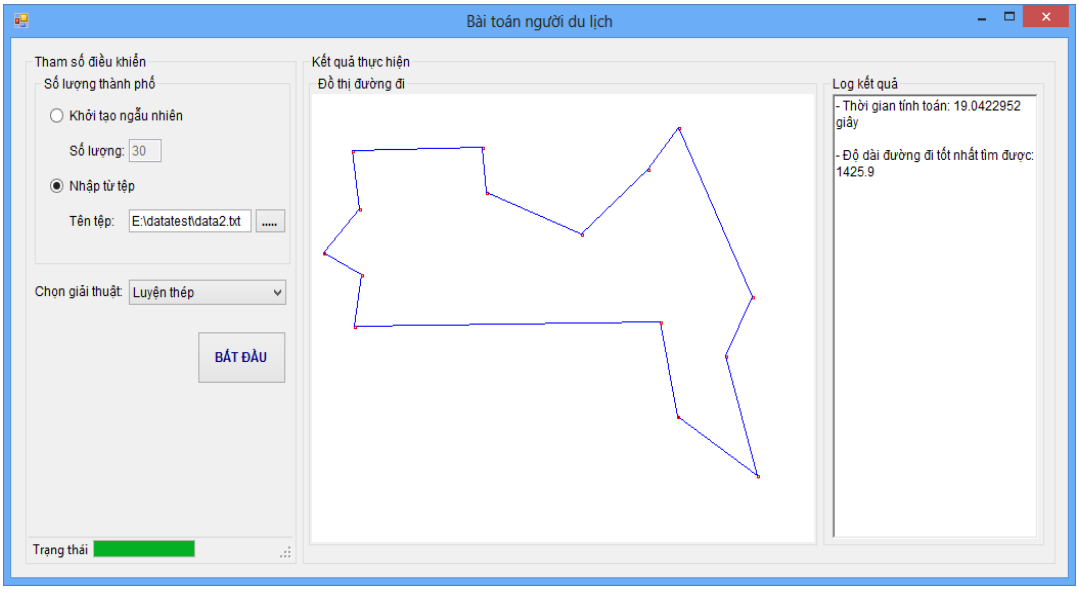
Chương trình cho phép lựa chọn kiểu dữ liệu đầu vào theo dạng khởi tạo ngẫu nhiên hoặc nhập từ tệp dữ liệu *txt*, tệp dữ liệu này phải có cấu trúc như được mô tả trong phần 3.3.1.



Hình 3.5. Kết quả chương trình bằng giải thuật Tabu với 30 thành phố khởi tạo ngẫu nhiên



Hình 3.6. Kết quả chương trình bằng giải thuật Tabu với 50 thành phố đọc dữ liệu từ tệp



Hình 3.7. Kết quả chương trình bằng giải thuật Luyện thép với 15 thành phố đọc dữ liệu từ tệp

3.4. Đánh giá hiệu quả của giải thuật tìm kiếm Tabu Search

Để kiểm tra tính hiệu quả của giải thuật tìm kiếm Tabu, đề tài thực hiện kiểm tra trên kịch bản như sau:

Thực hiện việc ghi chép về thời gian xử lý (tính bằng mili giây) mà mỗi giải thuật cần để tính toán được kết quả.

Mỗi lần tính toán sẽ được tiến hành 3 lần, tính giá trị trung bình và lấy một số phần thập phân ở kết quả cuối cùng. Kết quả đo đạc được ghi lại dưới bảng sau:

Bảng 3.1. Kết quả tính toán bằng giải thuật quay lui

STT	Số đỉnh	Lần đo 1	Lần đo 2	Lần đo 3	Trung bình
1	5	46.5	47.2	45.8	46.5
2	8	274.2	278.3	280.6	277.7
3	10	29481.5	29601.8	29596.3	29599.9
4	20	_	_	_	_
5	30	_	_	_	_

Bảng 3.2. Kết quả tính toán bằng giải thuật Luyện thép

STT	Số đỉnh	Lần đo 1	Lần đo 2	Lần đo 3	Trung bình
1	5	2042.0	2156.6	1986.5	2061.7
2	8	2756.5	2896.4	2801.2	2818.0
3	10	5963.2	6025.4	6100.2	6029.6
4	20	12032.5	13023.5	12636.4	12564.1
5	30	23632.6	23589.5	23587.1	23603.1
6	50	42568.5	43124.4	41986.2	42559.7

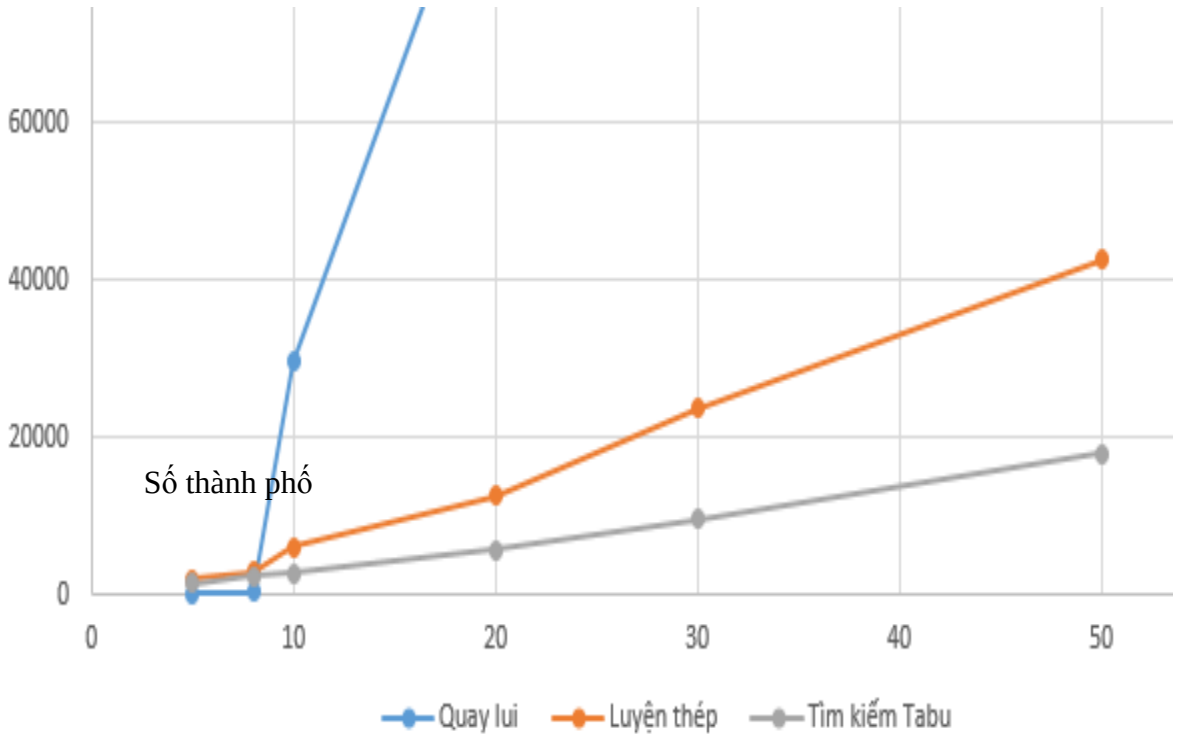
Bảng 3.3. Kết quả tính toán bằng giải thuật Tìm kiếm Tabu

STT	Số đỉnh	Lần đo 1	Lần đo 2	Lần đo 3	Trung bình

1	5	1423.5	1420.0	1430.8	1424.8
2	8	2269.1	2250.4	2260.5	2260.0
3	10	2783.6	2788.0	2802.8	2791.5
4	20	5761.2	5788.2	5766.5	5772.0
5	30	9560.6	9620.3	9520.2	9567.0
6	50	17661.2	18002.5	17862.2	17842.0

Bảng 3.4. Tổng hợp kết quả tính toán của ba giải thuật

STT	**Số đỉnh**	**Quay lui**	**Luyện thép**	**Tìm kiếm Tabu**
1	5	46.5	2061.7	1424.8
2	8	277.7	2818.0	2260.0
3	10	29599.9	6029.6	2791.5
4	20	_	12564.1	5772.0
5	30	_	23603.1	9567.0
6	50	_	42559.7	17842.0



Hình 3.8. Đồ thị biểu diễn thời gian chạy của 3 giải thuật

Với ba giải thuật tìm kiếm cục bộ: Giải thuật tìm kiếm Tabu hiệu quả hơn so với giải thuật Luyện thép và giải thuật Quay lui khi xử lý với số lượng đỉnh lớn. Dựa trên kết quả ở hình 3.8 ta có thể thấy thời gian để tìm kiếm theo giải thuật Tabu tăng không quá nhanh khi số lượng đỉnh tăng.

KẾT LUẬN

1. Kết quả đạt được của đề tài

Sau thời gian nghiên cứu thực hiện, đề tài đã hoàn thành các nhiệm vụ cơ bản ban đầu đặt ra, với các kết quả đạt được như sau:

Thứ nhất: Đề tài đã trình bày được tổng quan về thuật toán tìm kiếm cục bộ, cũng như trình bày được một số giải thuật tìm kiếm cục bộ hiện nay: Leo đồi, Luyện thép.

Thứ hai: Đề tài đã tìm hiểu và trình bày được các vấn đề cơ bản của thuật toán tìm kiếm Tabu: nguyên lý chung của tìm kiếm Tabu, cách sử dụng bộ nhớ, nền tảng của giải thuật, các chiến lược trong tìm kiếm Tabu…

Thứ ba: Đề tài đã xây dựng chương trình ứng dụng mô phỏng giải quyết bài toán người du lịch bằng thuật toán tìm kiếm Tabu. Đề tài cũng áp dụng một số giải thuật tìm kiếm để cài đặt mô phỏng bài toán người du lịch, từ đó đánh giá và so sánh được hiệu quả của các giải thuật đem lại.

Các kết quả thực nghiệm thu được của đề tài đã phản ánh trung thực với phần cơ sở lý thuyết được trình bày ở trên.

2. Hạn chế của đề tài

Chương trình cài đặt đã sử dụng một số quy tắc như nhóm các đỉnh gần nhau để tạo ra chu trình hoặc nhóm các chu trình có đường đi ngắn nhất để tạo ra các chu trình mới tối ưu hơn trong phép lặp ở bước khởi tạo chu trình, quy trình này giúp giảm thiểu sự ngẫu nhiên của dữ liệu, đồng thời khả năng tạo ra kết quả mới tối ưu là cao hơn.

Tuy vậy chương trình còn một số hạn chế trong các phép tính xử lý các các giá trị trung gian như ma trận khoảng cách, mảng động lưu kết quả chu

trình tìm được, các giá trị trung gian này sẽ tăng nhanh khi số đỉnh đầu vào là lớn gây tốn kém về mặt bộ nhớ.

Các kết quả thực nghiệm chạy trên chương trình của đề tài xây dựng cho bài toán người du lịch mới chỉ thực hiện với số lượng thành phố nhỏ, chưa đo được thời gian chạy giải thuật với số thành phố đủ lớn.

Các kết quả đánh giá mới chỉ thực hiện trên bài toán người du lịch, chưa thực hiện trên nhiều bài toán để đánh giá được hiệu quả của giải thuật tìm kiếm Tabu một cách chính xác và tổng quát hơn

3. Hướng phát triển của đề tài

Việc giải quyết những hạn chế trên cũng là hướng phát triển của đề tài trong tương lai. Tác giả sẽ sớm triển khai thực nghiệm trên nhiều bài toán (bên cạnh bài toán người du lịch) để từ đó đánh giá được hiệu quả của giải thuật tìm kiếm Tabu mang lại.

TÀI LIỆU THAM KHẢO

[1]. Nguyễn Đức Nghĩa – Nguyễn Tô Thành, *Toán rời rạc*, NXB Giáo dục, 1997

[2]. Từ Minh Phương, *Bài giảng Nhập môn trí tuệ nhân tạo*, 2010;

[3]. Costa, "*A Tabu Search Algorithm for Computing an Operational Timetable*", European Journal of Operational Research, 1994, 98 – 110;

[4]. Fred Glover, "*Tabu Search: A tutorial*", Interfaces, 20, 4 (1990) 74 – 94;

[5]. Fred Glover, Manuel Laguna, "*Tabu Search*", Kluwer Academic Publishers, United States of America, 1998;

[6]. Holger H. Hoos, Thomas Stuzle, " *Stochastic Local Search Foundation and Applications*", Morgan Kaufmann – USA;

[7]. Manuel Laguna, " *A Guide to Implementinh Tabu Search*", 1994, 5 – 25;

[8]. Ivica Martinjak, Marin Golub, Comparison of Heuristic Algorithms for the N – Queen Problem, Proceedings of the ITI Cavtat, Croatia, 760-764, 2007;

[9]. Wikipedia, http://en.wikipedia.org/wiki/Local_search

[10]. Wikipedia, http://en.wikipedia.org/wiki/Tabu_search

[11]. Wikipedia, http://en.wikipedia.org/wiki/Metaheuristic