

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

— * —

ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC
NGÀNH CÔNG NGHỆ THÔNG TIN

**ỨNG DỤNG CÁC GIẢI THUẬT
HEURISTIC ĐỂ THIẾT KẾ MẠNG CHỊU
LỖI**

Sinh viên thực hiện : **Nguyễn Thị Minh**
Lớp : **KHMT – K51**
Giáo viên hướng dẫn : **Ths Huỳnh Thị Thanh Bình**

HÀ NỘI 06-2011

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên

Họ và tên sinh viên: NGUYỄN THỊ MINH

Điện thoại liên lạc 0976 346 066

Email: ntminh1988@gmail.com

Lớp: KHMT – K51

Hệ đào tạo: Đại học chính quy

Đồ án tốt nghiệp được thực hiện tại: Bộ môn Khoa học máy tính, Viện CNTT&TT, Đại học Bách Khoa Hà Nội.

Thời gian làm ĐATN: Từ ngày 21/02/2011 đến 25/05/2011

2. Mục đích nội dung của ĐATN

- Nghiên cứu về bài toán thiết kế mạng chịu lỗi (SNDP), đề xuất giải thuật heuristic và meta-heuristic để giải bài toán.

3. Các nhiệm vụ cụ thể của ĐATN

- Nghiên cứu các bài toán thiết kế mạng chịu lỗi, tìm hiểu các heuristic và meta-heuristic có sẵn để xây dựng mô hình mạng chịu lỗi.
- Cài đặt giải thuật heuristic, meta-heuristic giải bài toán mạng chịu lỗi, cải tiến giải thuật sử dụng giải thuật cây khung nhỏ nhất và cấu trúc tập rời rạc (Disjoint Set)
- Tiến hành chạy thực nghiệm và so sánh kết quả.

4. Lời cam đoan của sinh viên:

Tôi – *Nguyễn Thị Minh* - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *Ths Huỳnh Thị Thanh Bình*.

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

Hà Nội, ngày tháng năm 2011

Tác giả ĐATN

Nguyễn Thị Minh

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:

Hà Nội, ngày tháng năm 2011

Giáo viên hướng dẫn

Ths Huỳnh Thị Thanh Bình

TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP

Hiện tại, trên thế giới đã có nhiều công trình nghiên cứu về giải thuật di truyền giải bài toán SNDP. Trong đó, hướng tiếp cận bài toán theo mô hình giải thuật heuristic đã đưa ra được kết quả tốt hơn cả.

Mục đích đề tài là nhằm nghiên cứu tổng quan về vấn đề thiết kế mạng chịu lỗi (Survivable Network Design Network Problem - SNDP), cài đặt và chạy thử nghiệm đưa ra kết quả cho một mô hình mạng chịu lỗi cụ thể. Trên cơ sở đó, đề xuất giải thuật heuristic và metaheuristic để giải quyết vấn đề.

Cấu trúc của đồ án gồm sáu chương với nội dung chính như sau:

Chương 1 trình bày các kiến thức cơ sở về đồ thị, cây Steiner và lớp bài toán NP-khó với mục đích làm nền tảng lý thuyết cho các chương tiếp theo.

Chương 2 trình bày về bài toán thiết kế mạng chịu lỗi mô hình hóa đồ thị, lịch sử phát triển, các ứng dụng.

Chương 3 trình bày về các giải thuật heuristic, meta-heuristic.

Chương 4 trình bày giải thuật heuristic để giải bài toán cây Steiner .

Chương 4 trình bày giải thuật cải tiến được thử nghiệm bởi nhiều tham số để so sánh đánh giá hiệu quả.

Chương 5 trình bày kết quả thu được của đồ án, đánh giá hiệu quả thuật toán. Các kết quả thống kê được trình bày dưới dạng bảng và biểu đồ.

ABSTRACT OF THESIS

Recently, with rapidly developing of digital information technology, network connecting of the people becomes popular. Therefore the building and designing an effective communication network is focused on field research, particularly Survivable Network Design Problem.

Faced on the explosive growth of the Internet, the network designers have some incentives to enhance the survivability in the network in case of facility failures. The rapid development of optical technology has heightened such incentives because the disorder of a single fiber can disrupt tens of thousand circuits. The main concern on network survivability design is to keep network remaining functional while some links or nodes are severed. For example, one typical survivable network design problem is selecting links so that the sum of their costs is minimized while satisfying the given requirements for the number of paths between every pair of nodes.

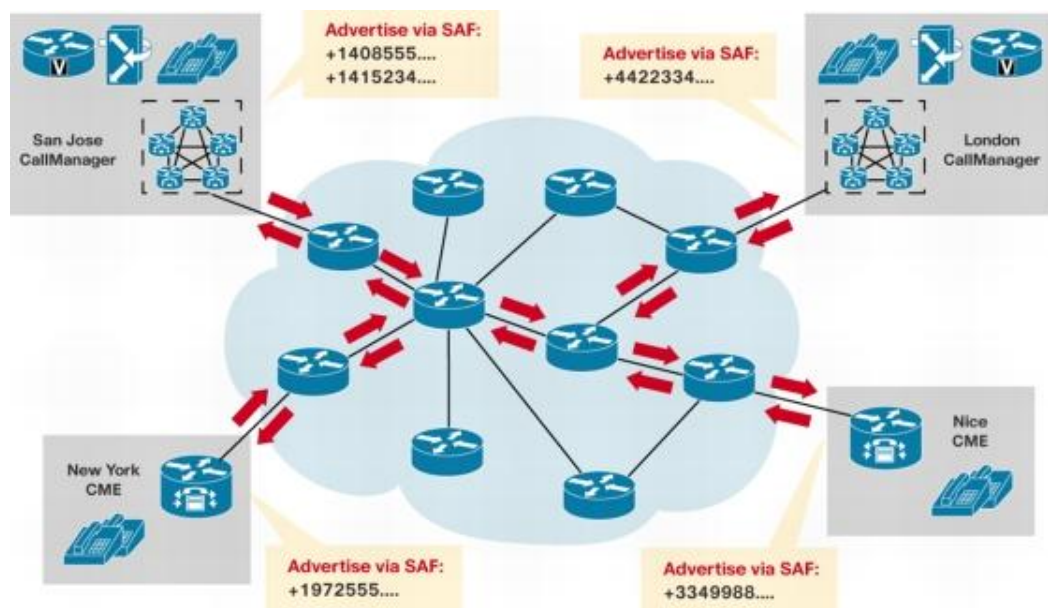
Survivable Network Design Problem (SNDP) is the real world communications networks design problem to ensure reliability.

The purpose of this thesis is proposed to make the algorithm optimal solution to the survivable network design problem.

This thesis researchs on SNDP and proposes the algorithms for solving this problem.

LỜI MỞ ĐẦU

Gần đây, cùng với sự tăng trưởng nhanh chóng của công nghệ thông tin nói chung và mạng cáp quang nói riêng, nhu cầu cung cấp mạng có khả năng sống sót cao cũng được tăng nhanh. Trong thực tế, việc thiết kế mạng – truyền thông, thiết kế các mạch VLSI (Very-Large-Scale Integration) và trong các hệ thống phục hồi thông tin hiện nay không những chỉ đòi hỏi về chi phí, giá thành thiết kế mà còn quan tâm đáng kể đến độ tin cậy của mạng, hay nói cách khác chính là vấn đề thiết kế mạng chịu lỗi (Survivable Network Design Problem – SNDP). Tuy nhiên, do SNDP là bài toán tối ưu hóa tổ hợp thuộc lớp bài toán NP – khó nên hiện chưa có một thuật toán chính xác nào có thể tìm được lời giải tối ưu khi kích thước của bài toán lớn trong thời gian đa thức. Vì vậy, giải thuật heuristic, meta-heuristic là phương pháp được quan tâm để giải bài toán này.



Hình 1: Mô hình mạng tổng quát

Hiện tại trên thế giới đã đưa ra rất nhiều mô hình cho bài toán thiết kế mạng chịu lỗi. Tùy thuộc vào thực tế, mỗi mô hình lại đại diện cho ứng dụng vào từng loại mạng. Đề án này tập trung nghiên cứu vào mô hình mạng truy cập (last mile). Đây là mô hình mạng kết nối đến cơ sở hạ tầng đã tồn tại. Đối với mô hình này, hướng tiếp cận bài toán theo giải thuật heuristic, meta-heuristic đã cho kết quả tốt hơn cả.

Đồ án này đã nghiên cứu lý thuyết tổng quan về bài toán thiết kế mạng chịu lỗi, cài đặt giải thuật heuristic, meta-heuristic giải quyết bài toán, thực hiện cải tiến giải thuật, so sánh và đánh giá kết quả. Qua đó, đưa ra hướng phát triển tiếp theo của bài toán thiết kế mạng chịu lỗi.

LỜI CẢM ƠN

Trước hết, em xin gửi lời cảm ơn chân thành tới cô Huỳnh Thị Thanh Bình, người đã tận tình dạy dỗ và hướng dẫn em trong quá trình hoàn thành đồ án cũng như trong học tập.

Đồng thời, em xin bày tỏ lòng biết ơn đến các thầy cô giáo trong Viện Công nghệ thông tin và Truyền thông – trường Đại học Bách Khoa Hà Nội, những người đã tận tình giảng dạy, truyền đạt cho chúng em những kiến thức cơ bản làm nền tảng cho việc thực hiện đồ án cũng như trong quá trình công tác sau này.

Em cũng xin gửi lời cảm ơn tới các anh chị tại trường Đại học Bách Khoa Hà Nội, các bạn, các em trong nhóm sinh viên nghiên cứu SNDP, những người luôn ở bên cạnh giúp đỡ, động viên em trong quá trình hoàn thành đồ án.

Cuối cùng, với tất cả sự kính trọng con xin bày tỏ lòng biết ơn sâu sắc tới bố mẹ và anh chị em trong gia đình đã luôn là chỗ dựa tinh thần vững chắc và tạo mọi điều kiện cho con ăn học nên người

Hà nội, ngày 25 tháng 05 năm 2011

Nguyễn Thị Minh

MỤC LỤC

| | |
|--|-----|
| PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP | i |
| TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP | ii |
| ABSTRACT OF THESIS | iii |
| LỜI MỞ ĐẦU | iv |
| LỜI CẢM ƠN | vi |
| MỤC LỤC | vii |
| DANH MỤC HÌNH VẼ | ix |
| DANH MỤC CÁC BẢNG | xi |
| DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ..... | xii |
| CHƯƠNG 1: CƠ SỞ LÝ THUYẾT VÀ LỚP BÀI TOÁN NP-KHÓ | 1 |
| 1. Các khái niệm cơ bản về đồ thị | 1 |
| 1.1. Định nghĩa đồ thị | 1 |
| 1.2. Đường đi và tính liên thông | 2 |
| 1.3. Cây và cây Steiner | 2 |
| 2. Độ phức tạp tính toán..... | 3 |
| 2.1. Một số khái niệm cơ bản..... | 3 |
| 2.2. Các ký hiệu tiệm cận..... | 4 |
| 2.3. Độ phức tạp tính toán của bài toán | 5 |
| 3. Lớp bài toán NP-khó..... | 5 |
| 3.1. Một số khái niệm cơ bản..... | 5 |
| 3.2. Lớp bài toán P, NP, và co-NP | 6 |
| 3.3. Khái niệm quy dẫn..... | 7 |
| 3.4. Lớp bài toán NP-đầy đủ và NP-khó | 8 |
| 4. Các giải thuật tìm đường đi ngắn nhất..... | 8 |
| 4.1. Giải thuật Dijkstra | 9 |
| 4.2. Giải thuật Kruskal..... | 10 |
| 4.3. Giải thuật Floyd-Warshall..... | 11 |
| CHƯƠNG 2: BÀI TOÁN THIẾT KẾ MẠNG CHỊU LỖI | 12 |
| 1. Phát biểu bài toán | 12 |
| 1.1. Bài toán thiết kế mạng chịu lỗi..... | 12 |
| 1.2. Sơ đồ quy dẫn..... | 12 |
| 2. Các ứng dụng của bài toán..... | 13 |
| 2.1. Thiết kế mạng truyền thông | 13 |
| 2.2. Thiết kế mạng lưới giao thông | 14 |
| CHƯƠNG 3: GIẢI THUẬT HEURISTIC VÀ META-HEURISTIC | 16 |
| 1. Lịch sử phát triển..... | 16 |
| 2. Các giải thuật heuristic cơ bản | 17 |

| | | |
|--|---|----|
| 2.1. | Giải thuật tìm kiếm cục bộ và luyện thép | 19 |
| 2.2. | Giải thuật gốc lân cận biến thiên (Variable Neighborhood Descent) và tìm kiếm lân cận biến thiên (Variable Neighborhood Search) | 19 |
| CHƯƠNG 4: ỨNG DỤNG GIẢI THUẬT HEURISTIC GIẢI BÀI TOÁN THIẾT KẾ MẠNG CHỊU LỖI | | 22 |
| 1. | Các hướng tiếp cận giải bài toán SNDP | 22 |
| 1.1. | Phát biểu bài toán SNDP..... | 22 |
| 1.2. | Tổng quan về bài toán SNDP..... | 22 |
| 2. | Giải thuật heuristic giải bài toán SNDP..... | 25 |
| 2.1. | Giải thuật tổng quát cho bài toán..... | 25 |
| 2.2. | Giải thuật xây dựng cây Steiner | 26 |
| 2.3. | Tỉa cây để loại bỏ các node không cần thiết trong cây Steiner. | 32 |
| 2.4. | Thêm kết nối dự phòng cho các đỉnh khách hàng loại hai (C2)..... | 33 |
| 2.5. | Quá trình cải tiến giải pháp sử dụng Local Search..... | 36 |
| 2.6. | Giải quyết ràng buộc không bắt chéo (Non-Crossing) trong mặt phẳng | 38 |
| CHƯƠNG 5: KẾT QUẢ THỰC NGHIỆM | | 40 |
| 1. | Dữ liệu thực nghiệm | 40 |
| 2. | Môi trường thử nghiệm..... | 40 |
| 3. | Tham số thực nghiệm | 40 |
| 4. | Kết quả thử nghiệm | 41 |
| 5. | So sánh kết quả..... | 50 |
| 5.1. | Đồ thị so sánh | 50 |
| 5.2. | Nhận xét | 57 |
| KẾT LUẬN..... | | 59 |
| PHỤ LỤC | | 61 |
| TÀI LIỆU THAM KHẢO | | 65 |

DANH MỤC HÌNH VẼ

| | |
|--|----|
| Hình 1: Mô hình mạng tổng quát | iv |
| Hình 2: Đồ thị vô hướng và đồ thị có hướng | 1 |
| Hình 3: Cây Steiner | 3 |
| Hình 4: Các lớp bài toán P, NP và co-NP | 7 |
| Hình 5: Sơ đồ quá trình quy dẫn | 7 |
| Hình 6: Phân lớp tạm thời các bài toán | 8 |
| Hình 7: Mô hình mạng chịu lỗi | 12 |
| Hình 8: Sơ đồ quy dẫn bài toán SNDP | 13 |
| Hình 9: Các kỹ thuật tìm kiếm | 17 |
| Hình 10: Lưu đồ giải thuật heuristic tổng quát | 18 |
| Hình 11: Ví dụ giải thuật SSSP | 26 |
| Hình 12: Cây Steiner và mạng tương ứng sinh bởi giải thuật APSP($n=100, m=65$) | 29 |
| Hình 13: Cây Steiner và Mạng tương ứng sinh bởi giải thuật APSP-NI($n=100, m=65$) | 30 |
| Hình 14: Hình minh họa cho ba phép di chuyển SMMove, SCMove và SDGMove | 37 |
| Hình 15: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=40, m=30$) | 50 |
| Hình 16: giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=80, m=60$) | 50 |
| Hình 17: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=100, m=150$) | 51 |
| Hình 18: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=200, m=100$) | 51 |
| Hình 19: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=200, m=200$) | 52 |
| Hình 20: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=300, m=500$) | 52 |
| Hình 21: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=450, m=750$) | 53 |
| Hình 22: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=600, m=1000$) | 53 |
| Hình 23: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=750, m=1250$) | 54 |
| Hình 24: Mối tương quan giữa chi phí xây dựng cây Steiner và chi phí mạng trong giải thuật APSP | 54 |
| Hình 25: Mối tương quan giữa chi phí xây dựng cây Steiner và chi phí mạng trong giải thuật APSP-I | 55 |
| Hình 26: Mối tương quan giữa chi phí xây dựng cây Steiner và chi phí mạng trong giải thuật APSP-N | 55 |
| Hình 27: Mối tương quan giữa chi phí xây dựng cây Steiner và chi phí mạng trong giải thuật APSP-NI | 56 |

| | |
|--|----|
| Hình 28: Mối tương quan giữa chi phí xây dựng cây Steiner và chi phí mạng trong giải thuật MST | 56 |
| Hình 29: Mối tương quan thời gian chạy ứng với sự biến thiên số lượng đỉnh của đồ thị.. | 57 |
| Hình 30: Giao diện chương trình demo với màn hình console..... | 61 |
| Hình 31: Chức năng nhập dữ liệu bằng tay | 61 |
| Hình 32: Chức năng nhập dữ liệu random..... | 62 |
| Hình 33: Chức năng nhập dữ liệu từ File | 62 |
| Hình 34: Chức năng hiển thị dữ liệu đầu vào | 63 |
| Hình 35: Chức năng chạy các giải thuật..... | 63 |
| Hình 36: Chức năng chạy các giải thuật đồng thời hiển thị kết quả bằng giao diện đồ thị. | 64 |
| Hình 37: Đồ thị kết quả khi chạy giải thuật..... | 64 |

DANH MỤC CÁC BẢNG

| | |
|---|----|
| Bảng 1: Tham số các bộ test | 40 |
| Bảng 2: Kết quả chạy thuật toán APSP | 42 |
| Bảng 3: Kết quả chạy thuật toán APSP-I | 43 |
| Bảng 4: Kết quả chạy thuật toán APSP-N | 45 |
| Bảng 5: Kết quả chạy thuật toán APSP-NI | 46 |
| Bảng 6: Kết quả chạy thuật toán MST | 48 |

DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ

| Chữ viết tắt | Viết đầy đủ | Ý nghĩa |
|--------------------|--|--|
| SNDP | Survivable Network Design Problem | Bài toán thiết kế mạng chịu lỗi |
| ACO | Ant Colony Optimization | Giải thuật tối ưu hóa đàn kiến |
| GA | Genetic Algorithm | Giải thuật di truyền |
| SA | Simulated Annealing | Giải thuật luyện thép |
| ILP | Integer Linear Program | Quy hoạch tuyến tính nguyên |
| MCF | Extended multi-commodity network flow | Mạng đa luồng mở rộng |
| OPT | Operative Planning Task | Lập kế hoạch thực tế |
| AUG _c | Augmentation | Tập cạnh tăng đường cho khách hàng loại 2 |
| SSSP | Single Source Shortest Path | Đường đi ngắn nhất từ một nguồn đơn |
| MST | Minimum Spanning Tree | Cây khung nhỏ nhất |
| APSP | All-Pairs-Shortest-Path | Giải thuật sử dụng đường đi ngắn nhất giữa tất cả cặp đỉnh |
| APSP _e | All-Pairs-Shortest-Path-extended | Giải thuật mở rộng sử dụng đường đi ngắn nhất giữa tất cả cặp đỉnh mở rộng |
| APSP _x | All-Pairs-Shortest-Path-adapting | Giải thuật sử dụng thích hợp đường đi ngắn nhất giữa tất cả cặp đỉnh |
| APSP-I | APSP Improvement | Giải thuật APSP cải tiến |
| APSP-N | APSP- New | Giải thuật APSP mới |
| APSP-NI | APSP-New Improvement | Giải thuật cải tiến APSP mới |
| AugSP | Augmentation by Shortest Path | Tăng luồng bởi đường đi ngắn nhất |
| AugSP _e | AugSP with Pseudo-Infrastructure Extension | Tăng luồng với cấu trúc cơ sở mạng mở rộng |
| SMMove | Single Migration Move | Di chuyển di trú đơn |
| SDGMove | Single Degree Move | Di chuyển bậc đơn |
| SCMove | Single Connecting Move | Di chuyển kết nối đơn |

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT VÀ LỚP BÀI TOÁN NP-KHÓ

1. Các khái niệm cơ bản về đồ thị

Đồ thị biểu diễn được rất nhiều cấu trúc. Nhiều bài toán thực tế có thể được biểu diễn bằng đồ thị. Trong toán học, lý thuyết đồ thị là một lĩnh vực đã xuất hiện từ lâu và có rất nhiều ứng dụng trong thực tế. Vào những năm đầu của thế kỷ 18, những tư tưởng cơ bản của lý thuyết đồ thị được đề xuất bởi nhà toán học lỗi lạc người Thụy Sĩ Leonhard Euler. Đặc biệt trong khoảng vài chục năm trở lại đây, với sự ra đời máy tính điện tử và sự phát triển nhanh chóng của công nghệ thông tin, lý thuyết đồ thị ngày càng được quan tâm nhiều hơn.

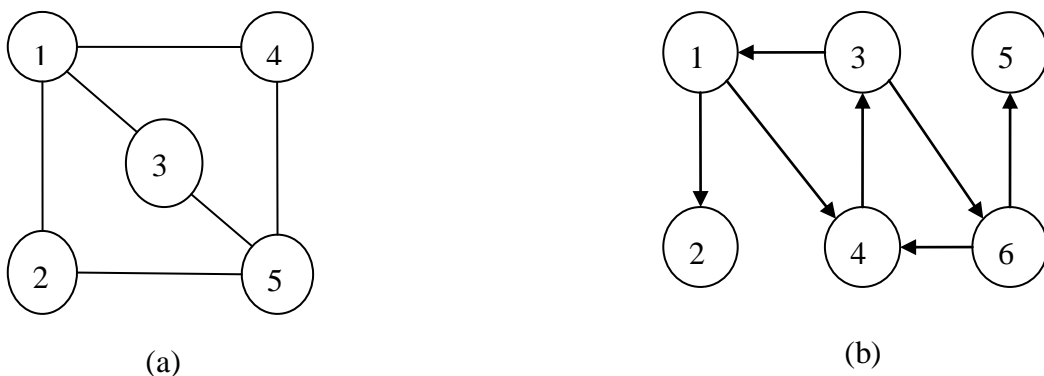
Sau đây, xin được trình bày qua các khái niệm cơ bản của đồ thị làm nền tảng cho việc áp dụng các giải thuật trong các lĩnh vực ứng dụng đồ thị.

1.1. Định nghĩa đồ thị

Đồ thị là một cấu trúc rời rạc gồm các đỉnh và cách cạnh nối các đỉnh này. Chúng ta phân biệt các loại đồ thị khác nhau bởi kiểu và số lượng cạnh nối các đỉnh này. Ta định nghĩa các loại đồ thị như sau.

Định nghĩa 1.1 Đơn đồ thị vô hướng $G = (V, E)$ bao gồm V là tập không rỗng chứa các đỉnh, và E là tập các cặp không có thứ tự gồm hai phần tử khác nhau của V gọi là cạnh [Hình 2.a].

Định nghĩa 1.2 Đơn đồ thị có hướng $G = (V, E)$ bao gồm tập các đỉnh V và tập các cạnh E là các cặp có thứ tự gồm hai phần tử khác nhau của V . Các cạnh của đồ thị có hướng còn được gọi là các cung. [Hình 2.b].



Hình 2: Đồ thị vô hướng và đồ thị có hướng

Để tiện sử dụng, chúng ta sẽ gọi đồ thị thay cho đơn đồ thị vô hướng mà không có chú thích gì thêm.

1.2. Đường đi và tính liên thông

Định nghĩa 1.3 Đường đi độ dài n từ đỉnh u đến đỉnh v , trong đó n là số nguyên dương trên đồ thị vô hướng $G = (V, E)$ là dãy $x_0, x_1, x_2, \dots, x_{n-1}, x_n$ trong đó $u = x_0, v = x_n, (x_i, x_{i+1}) \in E, i = 1..n$. Đỉnh u được gọi là đỉnh đầu, còn đỉnh v được gọi là đỉnh cuối của đường đi. Đường đi mà có đỉnh đầu trùng với đỉnh cuối được gọi là chu trình. Chu trình được gọi là đơn nếu như không có cạnh nào lặp lại.

Để xác định xem có luôn tồn tại đường đi giữa 2 cặp đỉnh trong đồ thị, chúng ta đưa ra khái niệm tính liên thông của đồ thị.

Định nghĩa 1.4 Đồ thị vô hướng $G = (V, E)$ được gọi là liên thông nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ của nó.

Định nghĩa 1.5 Đồ thị có hướng $G = (V, E)$ được gọi liên thông mạnh nếu luôn được đường đi giữa hai đỉnh bất kỳ của nó. Đồ thị có hướng $G = (V, E)$ được gọi là liên thông yếu nếu đồ thị vô hướng tương ứng với nó là liên thông.

1.3. Cây và cây Steiner

Định nghĩa 1.6 Cây là đồ thị vô hướng, liên thông và không chứa chu trình

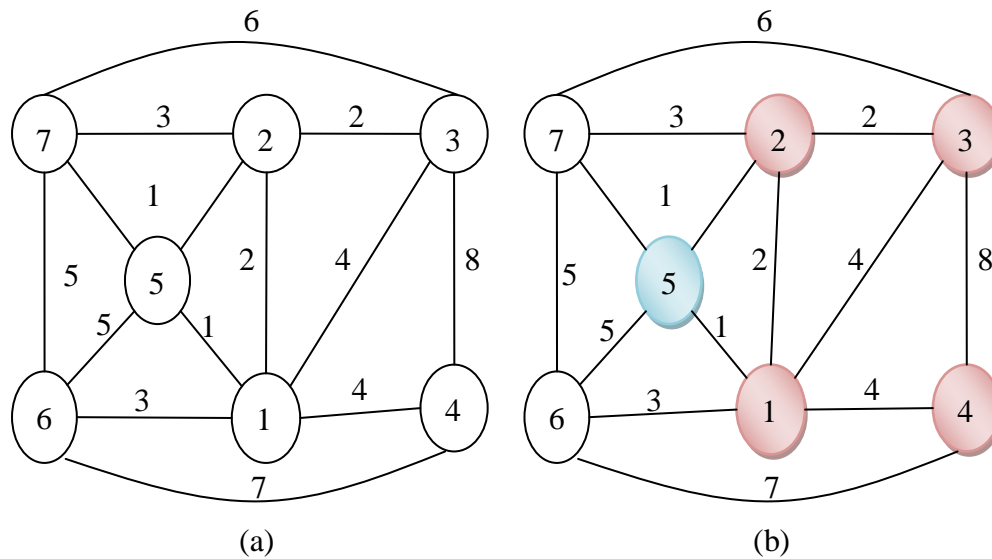
Định nghĩa 1.7 Cho đồ thị vô hướng $G = (V, E)$ và tập các đỉnh $W \subset V$. Cây $T = (W', F)$ được gọi là cây Steiner của W nếu nó là cây trong G bao trùm tất cả các đỉnh của W . Quy ước:

- Terminal node: là các đỉnh trong W
- Steiner node: là các đỉnh trong $W' - W$

Để thấy rằng một đồ thị vô hướng liên thông bất kỳ có thể có nhiều hơn một cây Steiner.

Trọng số của cây Steiner: nếu mỗi cạnh $e \in E$ của đồ thị G được gán một số thực $w(e)$ - gọi là trọng số của cạnh, thì trọng số của cây Steiner T là giá trị được tính bởi công thức $W(T) = \sum w(e)$

Cây Steiner nhỏ nhất: cây Steiner nhỏ nhất của G là cây Steiner T có $W(T)$ nhỏ nhất.



Hình 3: Cây Steiner

Ví dụ hình 3:

(a) là đồ thị ban đầu đã cho.

(b) Cây Steiner $T = (W', F)$ tương ứng của đồ thị (a) với $W = \{1, 2, 3, 4\}$

Các thông số của cây T:

- $W' = \{1, 2, 3, 4, 5\}$
- $F = \{(2, 3), (3, 4), (4, 1), (2, 5)\}$ có tổng trọng số là 11
- Terminal node: $W = \{1, 2, 3, 4, \}$
- Steiner node: $W' - W = \{5\}$

2. Độ phức tạp tính toán

Các vấn đề kỹ thuật thường được khái quát dưới dạng bài toán tính toán để tiện cho việc nghiên cứu và giải quyết. Bài toán tính toán là mối quan hệ giữa đầu vào (những yếu tố cho trước của bài toán) và đầu ra (những kết quả tính toán cần đạt được) của bài toán. Khái niệm độ phức tạp tính toán được xem như là chuẩn mực quan trọng để đánh giá hiệu quả của một bài toán tính toán.

2.1. Một số khái niệm cơ bản

Định nghĩa 2.1 Bài toán tính toán F là ánh xạ từ các xâu nhị phân độ dài hữu hạn vào tập các xâu nhị phân độ dài hữu hạn $F: \{0, 1\}^* \rightarrow \{0, 1\}^*$

Ở đây, các yếu tố đầu vào và đầu ra của bài toán được biểu diễn bằng xâu nhị phân. Mọi dạng dữ liệu (số, kí tự, xâu, mảng, tập hợp...) đều có thể mã hóa được bằng xâu nhị phân.

Bài toán chỉ ra mối quan hệ giữa đầu vào và đầu ra, nhưng để đạt được đầu ra từ đầu vào cho trước thì phải sử dụng các thuật toán để giải bài toán đó.

Định nghĩa 2.2 *Thuật toán giải bài toán đặt ra là một thủ tục xác định bao gồm hữu hạn các bước cần thực hiện để thu được đầu ra cho một đầu vào cho trước của bài toán*

Với mọi thuật toán, bên cạnh tính đúng đắn, thì độ phức tạp tính toán của thuật toán đó cũng là một yếu tố đáng được quan tâm.

Định nghĩa 2.3 *Độ phức tạp tính toán của một thuật toán là lượng tài nguyên tính toán mà thuật toán đó sử dụng để thực hiện công việc. Có hai loại tài nguyên cần quan tâm khi đánh giá độ phức tạp tính toán của thuật toán là bộ nhớ và thời gian.*

Ngày nay, do sự phát triển của công nghệ chế tạo bộ nhớ, vấn đề tài nguyên bộ nhớ cho thuật toán thường ít được tập trung hơn vấn đề về thời gian tính toán. Thời gian chạy thực tế của một thuật toán phụ thuộc vào nhiều yếu tố: cấu hình máy, ngôn ngữ cài đặt và cách thức cài đặt thuật toán, trình biên dịch và dữ liệu vào, trong đó dữ liệu vào là yếu tố quan trọng và đặc trưng nhất, được dùng để so sánh hiệu quả của thuật toán. Để tạo ra sự thống nhất trong cách đánh giá thời gian tính của thuật toán, chỉ xét đến yếu tố kích thước dữ liệu vào khi đánh giá.

2.2. Các ký hiệu tiệm cận:

Các ký hiệu tiệm cận thường hay sử dụng khi đánh giá độ phức tạp tính toán của thuật toán gồm có Θ , O , Ω và o , ω . Phần này sẽ nhắc lại định nghĩa và một số tính chất của các tiệm cận (bỏ qua hai ký hiệu o , ω)

Định nghĩa 2.4 *Cho các hàm $f(n)$ và $g(n)$ là các hàm số của số n nguyên dương*

- $\Theta(g(n)) = \{f(n) : \text{tồn tại các hằng số dương } c_1, c_2 \text{ và } n_0 \text{ sao cho } 0 \leq c_1 \leq f(n) \leq c_2 g(n), \text{ với mọi } n \geq n_0\}$. $g(n)$ được gọi là đánh giá tiệm cận đúng của $f(n)$ hay $f(n)$ có bậc là $g(n)$.
- $O(g(n)) = \{f(n) : \text{tồn tại các hằng số dương } c \text{ và } n_0 \text{ sao cho } f(n) \leq cg(n), \text{ với mọi } n \geq n_0\}$. $g(n)$ gọi là tiệm cận trên tiệm cận của $f(n)$ hay $f(n)$ có bậc không quá $g(n)$.
- $\Omega(g(n)) = \{f(n) : \text{tồn tại các hằng số dương } c \text{ và } n_0 \text{ sao cho } cg(n) \leq f(n), \text{ với mọi } n \geq n_0\}$. $g(n)$ được gọi là tiệm cận dưới tiệm cận của $f(n)$ hay $f(n)$ có bậc ít nhất là $g(n)$.

Để sử dụng các ký hiệu tiệm cận ở bên trong việc đánh giá thời gian tính của các thuật toán, các quy ước sau được sử dụng:

- Nếu thuật toán có thời gian tính trong tình huống nhanh nhất (tốt nhất) $t(n)$ với kích thước dữ liệu đầu vào n và $t(n) = \Omega(g(n))$ thì thời gian tính tốt nhất của thuật toán có bậc không nhỏ hơn $g(n)$ hay thời gian tính tốt nhất của thuật toán là $\Omega(g(n))$.
- Nếu thuật toán đòi hỏi thời gian tính trong tình huống chậm nhất (tồi nhất) $t(n)$ với kích thước dữ liệu đầu vào n và $t(n) = O(g(n))$ thì thời gian tính tốt nhất của thuật toán có bậc không nhỏ hơn $g(n)$ hay thời gian tính tốt nhất của thuật toán là $O(g(n))$.
- Nếu thuật toán đòi hỏi thời gian tính trong trung bình $t(n)$ với kích thước dữ liệu đầu vào n và $t(n) = \Theta(g(n))$ thì thời gian tính tốt nhất của thuật toán có bậc không nhỏ hơn $g(n)$ hay thời gian tính tốt nhất của thuật toán là $\Theta(g(n))$.

Thông thường khi nói thuật toán có thời gian tính là $O(f(n))$ thì hiểu là thời gian tính của thuật toán đánh giá trong tình huống tồi nhất là $O(f(n))$.

Còn khi nói thuật toán có thời gian tính là $\Omega(f(n))$ thì hiểu đánh giá thời gian tính của thuật toán trong tình huống tốt nhất là $\Omega(f(n))$.

2.3. Độ phức tạp tính toán của bài toán

Định nghĩa 2.5 *Độ phức tạp tính toán của một bài toán là thời gian tính (ở đây chỉ quan tâm đến đánh giá thời gian thực hiện, bỏ qua đánh giá về yêu cầu bộ nhớ) của thuật toán tốt nhất trong số tất cả các thuật toán giải bài toán đó.*

Với bài toán chắc chắn sẽ có những thuật toán chưa biết, vậy làm thế nào để biết được thời gian tính của thuật toán tốt nhất? Có 2 cách để giải quyết vấn đề này:

- Cách thứ nhất: Sử dụng các kỹ thuật đưa ra cận dưới cho độ phức tạp tính toán của bài toán.
- Cách thứ hai: Chỉ ra rằng bài toán đang xét có mức độ khó (tức là độ phức tạp tính toán) không thua kém gì bất kỳ một bài toán khó nào hiện biết

3. Lớp bài toán NP-khó

3.1. Một số khái niệm cơ bản

Định nghĩa 3.1 *Thuật toán có thời gian tính đa thức là thuật toán mà độ phức tạp thời gian của nó trong trường hợp xấu nhất được giới hạn trên bởi một hàm đa thức của kích thước dữ liệu đầu vào (kích thước dữ liệu đầu vào được tính bằng số bit*

cần thiết để biểu diễn nó). Tức là nếu n là kích thước dữ liệu đầu vào, thì luôn tồn tại một đa thức $p(n)$ sao cho

$$W(n) \in O(p(n))$$

Ví dụ:

Các thuật toán có độ phức tạp thời gian trong trường hợp xấu nhất sau đều có thời gian tính đa thức:

$$O(p(n)) = 2n ; 3n^3 + 4 ; 5n + n^{10} ; n \lg n$$

Các thuật toán có độ phức tạp thời gian trong trường hợp xấu nhất sau không có thời gian tính đa thức:

$$O(f(n)) = 2^n ; 2^{0.01n} ; 2^{\sqrt{n}} ; n!$$

Định nghĩa 3.2 Bài toán quyết định là bài toán mà đầu ra của nó chỉ có thể là “yes” hoặc “no” (0 hoặc 1, đúng hoặc sai...)

Định nghĩa 3.3 Xét bài toán tối ưu hóa $(P) \max\{f(x): x \in D\}$. Ta gọi bài toán dạng quyết định tương ứng với bài toán tối ưu (P) là bài toán quyết định sau:

(PD) “Cho giá trị K . Hỏi có tìm được $u \in D$ sao cho $f(u) \geq K$ hay không?”

Bài toán tối ưu và bài toán quyết định của nó có mối liên hệ được phát biểu trong định lý sau:

Định lý 3.1 Nếu bài toán quyết định tương ứng với một bài toán tối ưu có thể giải được hiệu quả (chẳng hạn bằng thuật toán có thời gian tính đa thức) thì bài toán tối ưu đó cũng giải được hiệu quả (bằng thuật toán thời gian tính đa thức).

Định nghĩa 3.4 Ta gọi bằng chứng ngắn gọn để kiểm tra xác nhận câu trả lời “yes” cho bộ dữ liệu vào “yes” của bài toán là một bằng chứng có độ dài bị chặn bởi một đa thức bậc cố định của độ dài dữ liệu đầu vào của bài toán, và việc kiểm tra nó là bằng chứng xác nhận câu trả lời “yes” đối với đầu vào đã cho của bài toán có thể thực hiện xong sau thời gian đa thức.

3.2. Lớp bài toán P, NP, và co-NP

Dưới đây là phân loại các lớp của bài toán

Định nghĩa 3.5 P là lớp bài toán quyết định có thể được giải quyết trong thời gian đa thức.

Hay nói cách khác, P là lớp các bài toán có thể được giải một cách nhanh chóng.

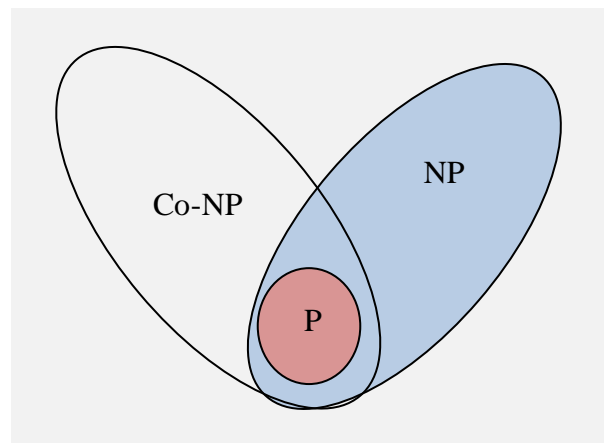
Định nghĩa 3.6 NP là lớp bài toán quyết định mà để xác nhận câu trả lời là “yes” của nó, có thể đưa ra bằng chứng ngắn gọn để kiểm tra.

Hay có thể nói NP là lớp bài toán mà có thể kiểm tra câu trả lời “yes” một cách nhanh chóng trong thời gian đa thức nếu đã có được lời giải.

Hiện nhiên ta có $P \subset NP$, tuy nhiên xác định xem $NP \subset P$ hay không hiện vẫn chưa có lời giải.

Định nghĩa 3.7 *co-NP là lớp bài toán mà để xác nhận câu trả lời “no” thì có thể đưa ra bằng chứng ngắn gọn để kiểm tra.*

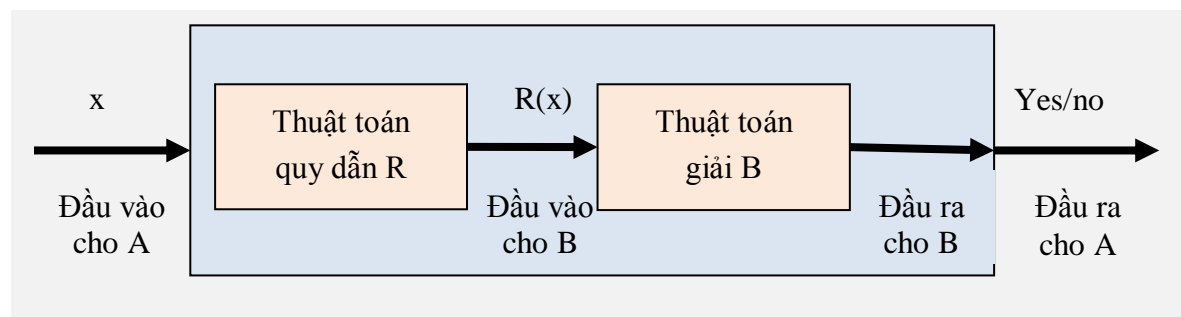
Như vậy có thể thấy co-NP là lớp bài toán hoàn toàn ngược với lớp NP. Có thể miêu tả mối quan hệ giữa ba lớp bài toán trên như trong hình dưới đây:



Hình 4: Các lớp bài toán P, NP và co-NP

3.3. Khái niệm quy dẫn

Định nghĩa 3.8 *Giả sử A và B là hai bài toán quyết định. Ta nói bài toán A có thể quy dẫn sau thời gian đa thức về bài toán B nếu tồn tại thuật toán thời gian đa thức R cho phép biến đổi bộ dữ liệu vào x của A thành bộ dữ liệu vào R(x) của B sao cho x là bộ dữ liệu “yes” của A khi và chỉ khi R(x) là bộ dữ liệu “yes” của B.*



Hình 5: Sơ đồ quá trình quy dẫn

Ký hiệu $A < B$ được dùng để chỉ bài toán A có thể quy dẫn về bài toán B. Phép quy dẫn thường dùng để so sánh độ khó của hai bài toán. Nếu A quy dẫn được về B thì

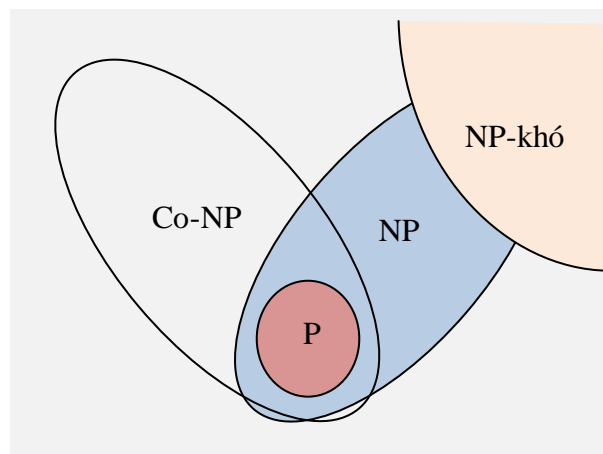
A không khó hơn B. Nếu A là khó (theo nghĩa chưa tìm được thuật toán thời gian tính đa thức để giải A) thì B cũng là khó, còn nếu B là dễ (nghĩa là đã có thuật toán thời gian tính đa thức giải B) thì A cũng là dễ.

3.4. Lớp bài toán NP-đầy đủ và NP-khó

Định nghĩa 3.9 Một bài toán quyết định A được gọi là NP-đầy đủ nếu như A là bài toán trong NP và mọi bài toán trong NP đều có thể quy dẫn về A.

Định nghĩa 3.10 Một bài toán A được gọi là NP-khó nếu sự tồn tại thuật toán đa thức để giải nó kéo theo sự tồn tại thuật toán đa thức để giải mọi bài toán trong NP.

Nói cách khác, nếu có thể giải một bài toán NP-khó nào đó một cách nhanh chóng, thì cũng có thể nhanh chóng giải quyết bất kỳ một bài toán nào khác. Bài toán NP-khó ít nhất là khó bằng bất cứ một bài toán nào trong NP. NP-đầy đủ là những bài toán khó nhất trong NP. Hình dưới đây biểu diễn cách phân lớp tạm thời các bài toán.



Hình 6: Phân lớp tạm thời các bài toán

4. Các giải thuật tìm đường đi ngắn nhất

Ở phần này, chúng ta sẽ tìm hiểu qua về bài toán tìm đường đi ngắn nhất từ một đỉnh hoặc giữa tất cả các cặp đỉnh trong đồ thị (bài toán này sẽ được áp dụng cụ thể trong chương 4).

Phát biểu bài toán tìm đường đi ngắn nhất: Cho trước đồ thị $G = (V, E)$, tìm đường đi ngắn nhất từ đỉnh nguồn $s \in V$ đến mỗi đỉnh $v \in V$. Có rất nhiều thuật toán giải quyết vấn đề này. Tiếp theo, sẽ nhắc lại một số thuật toán nổi tiếng và phân tích độ phức tạp tính toán của các thuật toán này.

4.1. Giải thuật Dijkstra

Thuật toán Dijkstra cho phép tìm đường đi ngắn nhất từ một đỉnh s đến các đỉnh còn lại của đồ thị có trọng số.

Phương pháp của thuật toán là xác định tuần tự đỉnh có chiều dài đến s theo thứ tự tăng dần.

Thuật toán được xây dựng dựa trên cơ sở gán cho mỗi đỉnh các nhãn tạm thời. Nhãn tạm thời của các đỉnh cho biết cận trên của chiều dài đường đi ngắn nhất từ s đến đỉnh đó. Nhãn của các đỉnh sẽ biến đổi trong các bước lặp, mà ở mỗi bước lặp sẽ có một nhãn tạm thời trở thành chính thức. Nếu nhãn của một đỉnh nào đó trở thành chính thức thì đó cũng chính là chiều dài ngắn nhất của đường đi từ s đến đỉnh đó.

Thuật toán Dijkstra

1. **for all** $v \in V$ **do**
 2. $d(v) = \infty$
 3. $color[u] = white$
 4. **end for**
 5. $d[s] = 0$
 6. $pred[s] = Null$
 7. $Q = V \setminus s$
 8. **while** $(Q \neq \emptyset)$ **do**
 9. $u = \text{đỉnh có } d[u] \text{ nhỏ nhất}$
 10. **for all** v là đỉnh kề của u **do**
 11. **if** $(d[u] + w(u, v) < d[v])$
 12. $d[v] = d[u] + w(u, v)$
 13. $pred[v] = u$
 14. **end if**
 15. **end for**
 16. $Color[u] = black$
 17. $Q = Q \setminus u$
 18. **end while**
-

Độ phức tạp tính toán:

Thời gian chạy của thuật toán Dijkstra phụ thuộc vào hàng đợi ưu tiên nhỏ nhất được cài đặt như thế nào. Xét trường hợp chúng ta duy trì hàng đợi ưu tiên nhỏ nhất bằng cách tận dụng các đỉnh được đánh số từ 1 đến $|V|$. Chúng ta đơn giản lưu trữ

$d[v]$ nó dưới dạng mảng. Mỗi thao tác chèn hay xóa cần $O(1)$ thời gian, và thao tác tìm đỉnh có $d[u]$ nhỏ nhất cần $O(V)$ thời gian (cần tìm kiếm trên toàn bộ phần tử của mảng). Như vậy toàn bộ thời gian cần thiết của thuật toán sẽ là $O(V^2+E) = O(V^2)$.

4.2. Giải thuật Kruskal

Thuật toán Kruskal dựa trên mô hình xây dựng cây khung bằng thuật toán hợp nhất, chỉ có điều thuật toán không phải xét các cạnh với thứ tự tùy ý mà xét các cạnh theo thứ tự đã sắp xếp: Với đồ thị vô hướng $G = (V, E)$ có n đỉnh. Khởi tạo cây T ban đầu không có cạnh nào. Xét tất cả các cạnh của đồ thị từ cạnh có trọng số nhỏ đến cạnh có trọng số lớn, nếu việc thêm cạnh đó vào T không tạo thành chu trình đơn trong T thì kết nạp thêm cạnh đó vào T . Lặp lại bước này cho đến khi:

- Đã kết nạp được $n-1$ cạnh vào trong T .
- Chưa kết nạp đủ nhưng xảy ra trường hợp cứ kết nạp thêm một cạnh bất kỳ trong số cạnh còn lại thì sẽ tạo ra chu trình đơn. Trường hợp này đồ thị G là không liên thông \rightarrow việc tìm kiếm cây khung bị thất bại.

Mô hình thuật toán Kruskal

Thuật toán Kruskal

1. **For all** $k \in V$ **do**
 2. $Lab[k] := -1$ // lab lưu số đỉnh của gốc cây k
 3. **For all** ($edge(u,v) \in E$ theo thứ tự từ cạnh trọng số nhỏ tới cạnh trọng số lớn) **do**
 4. $R1 := getRoot(u)$ // $r1$ là gốc của cây chứa đỉnh u
 5. $R2 := getRoot(v)$
 6. **If** $r1 \neq r2$ **then** // cạnh (u,v) nối hai cây khác nhau
 7. Kết nạp (u,v) vào cây, nếu đã đủ $n-1$ cạnh thì thuật toán dừng
 8. Union $(r1,r2)$ // hợp nhất hai cây thành một cây
 9. **End if**
 10. **End for**
 11. **End for**
-

Xét về độ phức tạp tính toán, ta có thể chứng minh được rằng thao tác GetRoot có độ phức tạp là $O(\lg n)$, còn thao tác Union là $O(1)$. Giả sử đã có danh sách cạnh đã sắp xếp rồi thì xét vòng lặp dựng cây khung, nó duyệt qua danh sách cạnh và với mỗi cạnh nó gọi 2 lần thao tác GetRoot, độ phức tạp là $O(m \lg n)$. Nếu đồ thị có cây khung ($m \geq n-1$), chi phí thời gian chủ yếu sẽ nằm ở thao tác sắp xếp danh sách cạnh bởi độ phức tạp của HeapSort là $O(m \lg m)$. Tóm lại độ phức tạp tính toán của

thuật toán là $O(m \lg m)$ trong trường hợp xấu nhất. Tuy nhiên, phải lưu ý rằng để xây dựng cây khung thì ít khi thuật toán phải duyệt toàn bộ danh sách cạnh mà chỉ một phần danh sách cạnh thôi.

4.3. Giải thuật Floyd-Warshall

Thuật toán giải quyết bài toán tìm đường đi ngắn nhất giữa các cặp đỉnh của đồ thị.

Ý tưởng: Thuật toán Floyd được thiết kế theo phương pháp quy hoạch động. Nguyên lý tối ưu được áp dụng cho bài toán này: “Nếu k là đỉnh nằm trên đường đi ngắn nhất từ i đến j thì đoạn đường từ i đến k và từ k đến j cũng phải ngắn nhất”

Thuật toán Floyd-Warshall

1. $N := \text{rows}(W)$
 2. $D^0 \leftarrow W$
 3. **For** $k \leftarrow 1$ **to** n **do**
 4. **For** $i \leftarrow 1$ **to** n **do**
 5. **For** $j \leftarrow 1$ **to** n **do**
 6. $D_{ij}^{(k)} \leftarrow \min(D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)})$
 7. **End for**
 8. **End for**
 9. **End for**
 10. **Return** $D^{(n)}$
-

Thời gian tính của thuật toán Floyd-Warshall được xác định bởi ba vòng lặp for lồng nhau. Vì mỗi lần thực hiện dòng ^[6] cần thời gian $O(1)$, nên thuật toán chạy trong thời gian $O(n^3)$

Như vậy, ở chương này đã trình bày cơ sở lý thuyết cơ bản về đồ thị và bài toán đường đi ngắn nhất. Chương 2 tiếp theo sẽ trình bày về bài toán thiết kế mạng chịu lỗi cũng như các ứng dụng của nó trong thực tế.

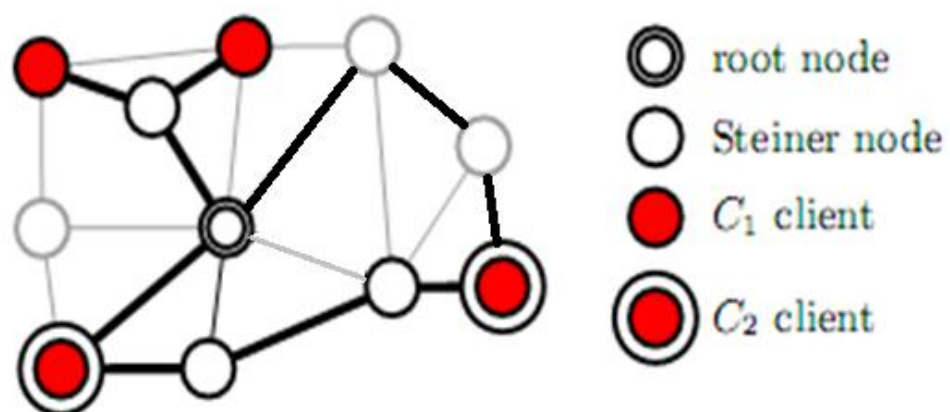
CHƯƠNG 2: BÀI TOÁN THIẾT KẾ MẠNG CHỊU LỖI

1. Phát biểu bài toán

1.1. Bài toán thiết kế mạng chịu lỗi

Bài toán thiết kế mạng chịu lỗi (SNDP - Survivable Network Design Problem) được phát biểu như sau:

“Cho một đơn đồ thị vô hướng kết nối $G = (V, E)$ với tập đỉnh là V , tập cạnh là E . Mỗi cạnh e trên đồ thị được gán với một giá trị không âm được gọi là trọng số $w(e)$. Trong đó tập đỉnh V bao gồm 4 loại nút: nút gốc (J), nút khách hàng loại 1 (C_1), nút khách hàng loại 2 (C_2) và nút trong không gian (S). Tìm cách kết nối các khách hàng $C = C_1 \cup C_2$ đến cơ sở hạ tầng đã tồn tại (J) sao cho tổng chi phí kết nối là nhỏ nhất, đồng thời thỏa mãn yêu cầu kết nối của từng loại khách hàng”



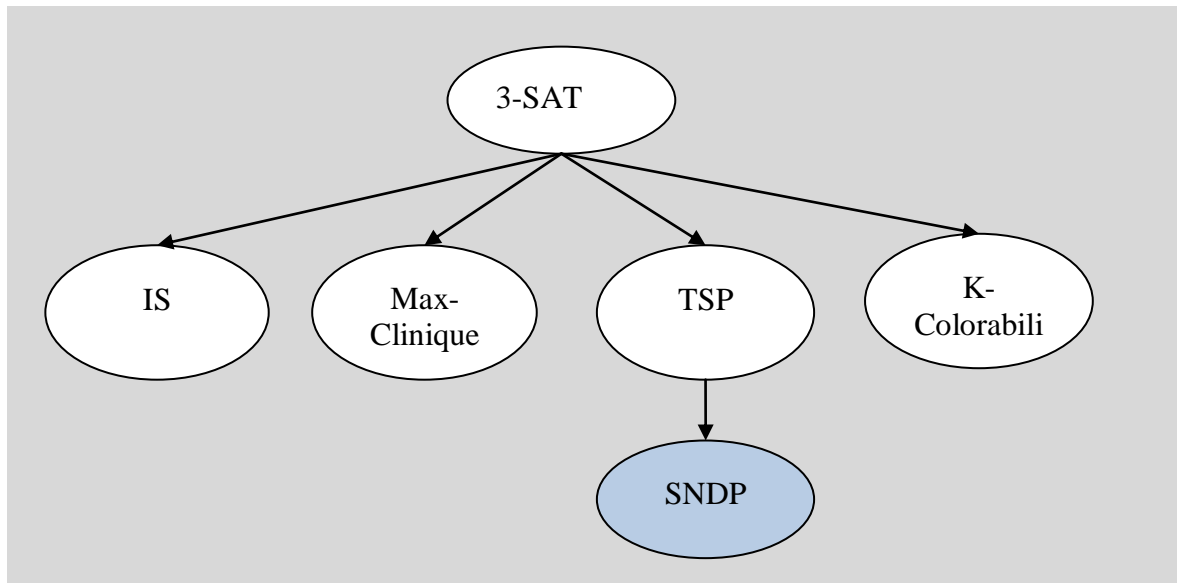
Hình 7: Mô hình mạng chịu lỗi

Để tiện cho việc trình bày, đề án xin đưa ra một số khái niệm liên quan.

1.2. Sơ đồ quy dẫn

Các nghiên cứu trước về bài toán SNDP cho thấy, bài toán SNDP thuộc vào lớp bài toán NP-khó. Dưới đây là sơ đồ quy dẫn của bài toán:

$$\text{SAT} \rightarrow \text{Chu kỳ Hamilton} \rightarrow \text{TSP}$$



Hình 8: Sơ đồ quy dẫn bài toán SNBP

Do bài toán SNBP thuộc lớp NP-khó nên không thể giải một cách chính xác khi kích thước dữ liệu đầu vào lớn. Đồ án này sẽ không hướng vào tìm kiếm lời giải chính xác tốt nhất cho bài toán mà tập trung tìm kiếm lời giải xấp xỉ. Trong trường hợp lý tưởng, lời giải xấp xỉ này sẽ rất gần với lời giải tốt nhất với chi phí thời gian chấp nhận được.

Một trong những phương pháp hay được sử dụng để tìm lời giải xấp xỉ cho bài toán NP-khó là sử dụng các giải thuật heuristic, meta-heuristic. Tổng quan về các giải thuật này sẽ được trình bày cụ thể ở chương 3.

2. Các ứng dụng của bài toán

SNBP có nhiều ứng dụng trong thực tế như: thiết kế mạng truyền thông, mạng phân tán khi quan tâm đến độ tin cậy của mạng,... Đồ án này giải quyết bài toán thiết kế mạng truy cập (last mile). Ngoài ra, cũng có thể sử dụng mô hình hóa một số bài toán ứng dụng thành đồ thị và áp dụng bài toán này trong các ứng dụng đó. Một số bài toán ứng dụng thường được quan tâm gắn liền với khái niệm sống sót (“Survivable”) như: ứng dụng thiết kế mạng lưới giao thông chống tắc nghẽn, ứng dụng trong hệ thống lưới điện, đường ống, đường nước và mạng truyền thông kiểm soát cơ sở hạ tầng giảm thiểu chi phí sửa chữa hệ thống khi đối mặt với các sự cố hỏng hóc thiết bị và đường dây... Phần tiếp theo giới thiệu hai ứng dụng cụ thể của bài toán SNBP: trong thiết kế mạng truyền thông và trong mạng lưới giao thông.

2.1. Thiết kế mạng truyền thông

Mạng truyền thông là một tập các máy tính độc lập và các phương tiện giao tiếp giữa chúng. Mạng truyền thông thường được sử dụng bởi nhiều người, có thể truy

cập tại bất cứ vị trí máy tính nào trong mạng. Vấn đề bảo mật của nó là vấn đề cần quan tâm khi quyết định thiết kế chúng. Giả thiết rằng chúng ta có thể phân tích tính chất bị tấn công của một cấu trúc mạng truyền thông dữ liệu. Mạng truyền thông có thể được đại diện bởi một đồ thị với trọng số bằng tham số chất lượng của dịch vụ (Quality of Service - QoS) ví dụ: tốc độ dữ liệu tối đa. Mục đích có thể đảm bảo tốc độ dữ liệu mong muốn, thậm chí khi có một vài link có thể bị thất bại. Đối với một mô hình mạng truyền thông, giả thiết rằng các hoạt động tấn công làm phá hủy liên kết truyền thông là điều khó tránh khỏi. Để đảm bảo tính tin cậy, tính an toàn của mạng, phụ thuộc vào đường đi giữa các máy tính trong mạng. Bài toán SNDP được dùng để đáp ứng được mức độ xảy ra lỗi hay hư hỏng đường truyền trong mạng, đáp ứng được độ tin cậy cần thiết cho việc thiết kế mạng truyền thông cơ bản.

Trong trường hợp, chỉ có một máy tính đảm nhận chức năng như máy chủ (server) và các máy tính khác kết nối đến máy tính này trong mạng, chúng ta mô hình hóa hệ thống này như một cây khung, gốc của nó chính là máy tính đã có (đảm nhận như chức năng của một server), các đỉnh là các máy tính (các khách hàng muốn kết nối đến mạng) hay các router, switch (tương ứng như là các đỉnh không cần thiết trong không gian), các cạnh là kết nối giữa hai máy. Như vậy, để đảm bảo được tính tin cậy của mạng, tùy thuộc vào chức năng của từng loại máy tính, chúng ta phải quyết định được số đường đi cần thiết đến máy tính gốc để đảm bảo khi gặp bất kỳ sự cố hỏng hóc hay bị tấn công nào, thì cũng không ảnh hưởng đến toàn bộ mạng truyền thông.

Trong trường hợp, có một cơ sở hạ tầng đã tồn tại (mạng truy cập), có nhiều máy tính đã sẵn có trong mạng, việc kết nối các máy tính mới cho mạng truyền thông này cần phải giải quyết qua bước tiền xử lý:

- Thu gọn cơ sở hạ tầng đã tồn tại ban đầu xem như nó chỉ được nhìn thấy trên mạng là một máy tính duy nhất.
- Việc kết nối thêm máy tính vào mạng tương tự như trường hợp ban đầu
- Kết thúc chúng ta lại phải xử lý đưa nó về tối ưu kết nối với một trong tập hợp các máy đã có.

Trong cả hai trường hợp vừa nêu, nếu dùng một cây Steiner và sau đó tăng thêm đường tùy thuộc vào nhu cầu kết nối, thì sẽ tiết kiệm chi phí và đáp ứng được các nhu cầu chịu lỗi của mạng truyền thông.

2.2. Thiết kế mạng lưới giao thông

Mạng lưới giao thông là một thành phần không thể thiếu trong mỗi thành phố, mỗi địa điểm dân cư. Cùng với sự gia tăng dân số ở các khu vực thành thị, mạng lưới

giao thông ngày càng trở nên phức tạp và được chú trọng nhiều hơn. Chẳng hạn, hiện nay tình trạng tắc nghẽn giao thông xảy ra thường xuyên trên địa bàn Hà Nội. Đã có nhiều phương án đã được đưa ra như phân luồng, chặn ngã ba, ngã tư ... Tuy nhiên, các phương pháp này cũng chỉ mới đạt được hiệu quả trong một thời gian ngắn. Tạo ra một cơ sở hạ tầng giao thông tối ưu, hạn chế tắc nghẽn là nhiệm vụ quan trọng của CIP (Critical Infrastructure) và mạng lưới truyền thông. Được biết tại báo cáo năm 1997 của Ủy ban Tổng thống về bảo vệ cơ sở hạ tầng quan trọng (PCCIP-Presidential Commission on Critical Infrastructure Protection) đã xác định các yếu tố cơ sở hạ tầng cần thiết cho quốc phòng, an ninh, kinh tế của Hoa Kỳ. Xác định cấu hình cơ sở hạ tầng tối thiểu là cần thiết để đáp ứng yêu cầu cơ bản. Quay lại với mô hình mạng lưới giao thông trong một thành phố, việc xác định các tình trạng nguy cơ dễ xảy ra tắc nghẽn ở một nút, đoạn đường giao thông nào đó là yếu tố quan trọng để quyết định xây dựng mô hình cơ sở hạ tầng giao thông cơ bản. Một phương thức thiết kế hiệu quả bắt đầu bằng việc sinh ra một cấu trúc đồ thị đầy đủ có trọng số các cạnh dựa trên khoảng cách Euclide, trong đó các đỉnh biểu diễn các nút giao thông, và trọng số cạnh biểu diễn khoảng cách Euclide giữa các nút giao thông này. Tiếp theo phương thức sử dụng đồ thị con để nhóm và kết nối các nút giao thông với nhau. Sau đó, tùy thuộc vào đặc điểm năng lực giao thông cần thiết, xác định số lượng các con đường phân chia thích hợp để kết nối giữa hai nút giao thông. Kết quả là một đồ thị con của bài toán SNDP sẽ hỗ trợ được quá trình thiết kế tối ưu cho mạng lưới giao thông.

Ngoài ra, các ứng dụng như hệ thống lưới điện, đường ống, đường nước cũng được xem xét như là vấn đề chịu lỗi của bài toán.

Qua hai phần trên, ta có thể thấy vai trò quan trọng của bài toán SNDP vào ứng dụng tin học cũng như các lĩnh vực khác trong đời sống. Với tính chất hữu dụng như thế nên không ít các nghiên cứu đã được tiến hành để đưa ra lời giải tốt cho bài toán. Phần sau xin trình bày lại bài toán SNDP và một số giải thuật heuristic để giải bài toán SNDP.

CHƯƠNG 3: GIẢI THUẬT HEURISTIC VÀ META-HEURISTIC

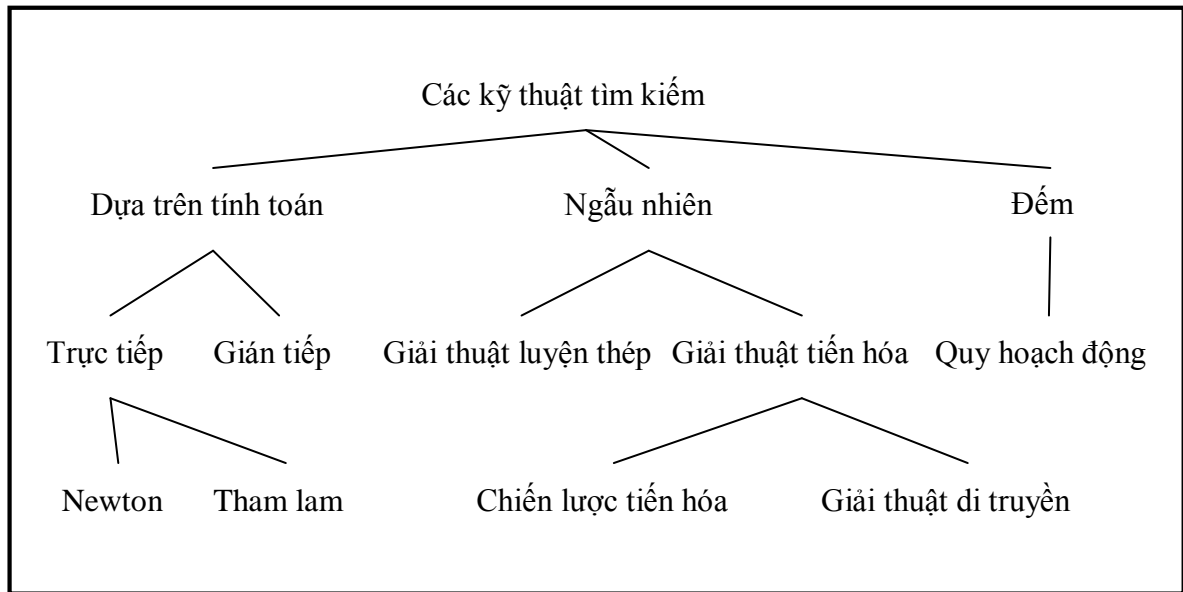
1. Lịch sử phát triển

Mục đích của bài toán tối ưu tổ hợp là tìm lời giải tốt nhất trong các lời giải có thể và không gian tìm kiếm lời giải của bài toán là rời rạc. Nhiều bài toán tổ hợp có độ phức tạp tính toán lớn và được phân loại vào lớp bài toán NP-khó (không tìm được lời giải tối ưu trong thời gian đa thức). Việc tìm ra lời giải tối ưu cho loại bài toán này trong các hệ thống song song lớn nhất cũng không thể thực hiện trong giới hạn thời gian cho phép, vì vậy các giải thuật heuristic, meta-heuristic giải các bài toán tổ hợp theo hướng xấp xỉ đã được phát triển để tìm ra lời giải gần tối ưu trong thời gian chấp nhận được.

Meta-heuristic là một cách gọi chung cho các giải thuật heuristic trong việc giải quyết các bài toán tổ hợp khó. Meta-heuristic bao gồm những chiến lược khác nhau trong việc khám phá không gian tìm kiếm bằng cách sử dụng những phương thức khác nhau và phải đạt được sự cân bằng giữa tính đa dạng và chuyên sâu của không gian tìm kiếm. Việc cài đặt thành công một giải thuật heuristic cho một bài toán tổ hợp đòi hỏi phải cân bằng giữa sự khai thác kinh nghiệm thu thập được trong quá trình tìm kiếm để xác định được những vùng với lời giải có chất lượng cao gần tối ưu. Các giải thuật heuristic và meta-heuristic nổi tiếng được biết đến như giải thuật luyện thép (SA- Simulated Annealing), giải thuật di truyền (GA - Genetic Algorithm), giải thuật tối ưu hóa đàn kiến (ACO – Ant Colony Optimization) ...

- **Giải thuật tối ưu hóa đàn kiến (ACO)** được đề xuất bởi Marco Dorigo năm 1992, là meta-heuristic dùng chiến lược của đàn kiến trong thế giới thực để giải bài toán tối ưu.
- **Giải thuật di truyền (SA)** do John Holland phát minh và được ông phát triển cùng với các đồng nghiệp và sinh viên. Giải thuật xuất phát từ phương thức xác suất dựa trên ý tưởng từ cơ chế di truyền trong sinh học và tiến trình tiến hóa trong cộng đồng cá thể của một loài.
- **Giải thuật luyện thép (SA)** là phương pháp xác suất được đề xuất bởi Kirkpatrick, Gelett và Vecchi (1983) và Cerny (1985), lại sử dụng tính chất thu được trạng thái năng lượng nhỏ nhất nhờ nóng chảy và ngưng tụ của kim loại trong tự nhiên để tìm ra giải pháp tốt nhất cho các loại bài toán tối ưu này.

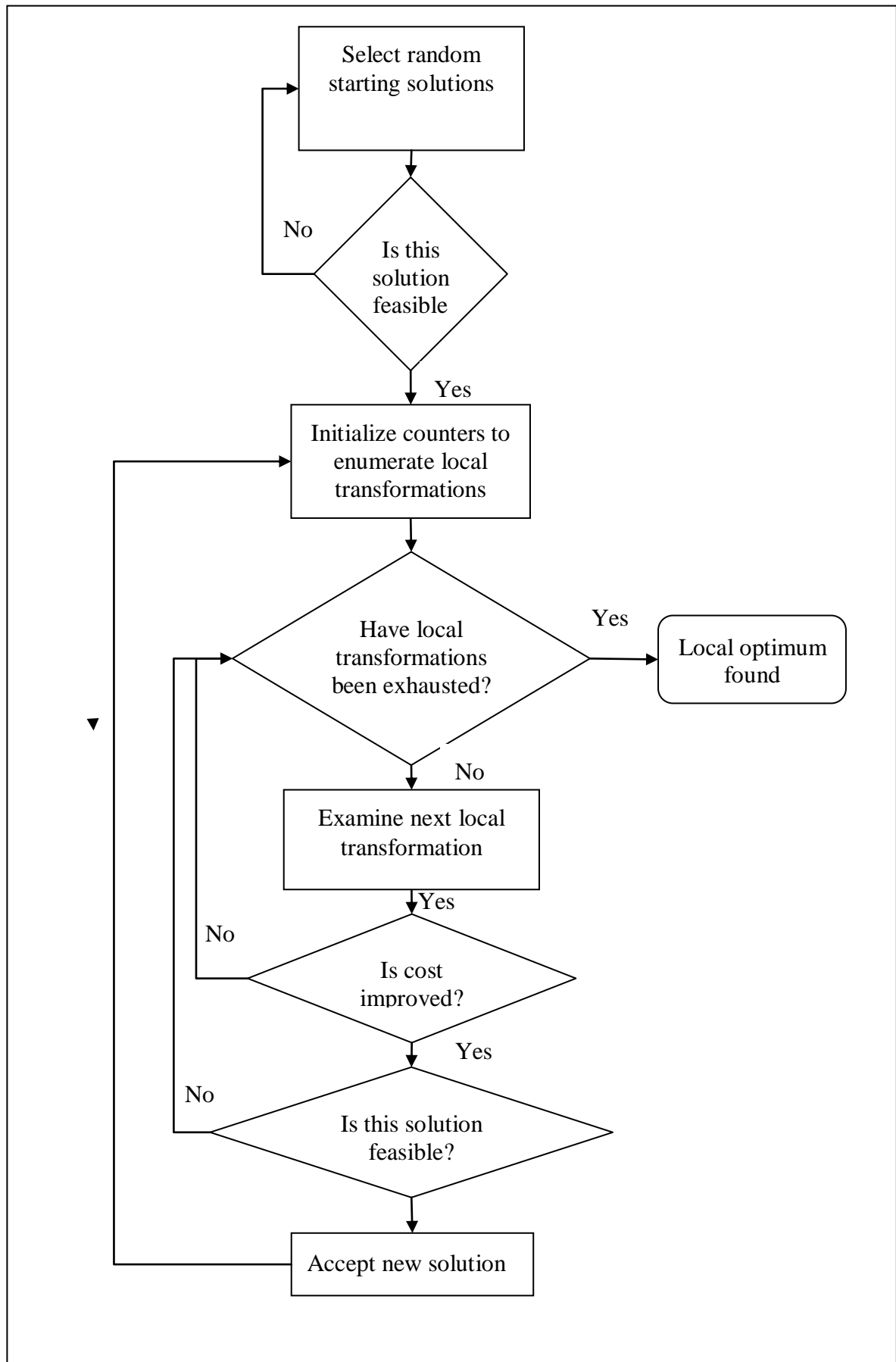
Hình sau đây cho ta thấy mối liên hệ giữa các kỹ thuật tìm kiếm



Hình 9: Các kỹ thuật tìm kiếm

2. Các giải thuật heuristic cơ bản

Mô hình giải thuật heuristic tổng quát như sau:



Hình 10: Lưu đồ giải thuật heuristic tổng quát

2.1. Giải thuật tìm kiếm cục bộ và luyện thép

2.1.1. Giải thuật tìm kiếm cục bộ

Tìm kiếm cục bộ đã được ứng dụng với nhiều bài toán tính toán phức tạp. Ý tưởng của phương pháp này là di chuyển từ một lời giải gần đúng ban đầu tới các lời giải láng giềng với mục tiêu tìm những lời giải tối ưu hơn.

Áp dụng vào đề tài này, lời giải được tạo nên trên đồ thị $G = (V, E, c)$ sử dụng cách tiếp cận heuristic, và sử dụng một trong ba phép toán di chuyển (sẽ nêu chi tiết trong chương 4). Một trong ba phép toán này được chọn ngẫu nhiên để áp dụng vào lời giải hiện tại. Nếu lời giải mới tìm được tối ưu hơn, thì nó trở thành lời giải hiện tại mới.

Giải thuật tìm kiếm cục bộ

1. $G_s = (V_s, E_s)$
 2. $G_s := \text{heuristic_solution_construction}(G)$
 3. $\text{candidate_solution} := G$
 4. **while** chưa đạt tới số lần thực hiện giải thuật (hoặc điều kiện kết thúc khác) **do**
 5. $\text{new_solution} := \text{di_chuyển_với_phép_toán_ngẫu_nhiên}(\text{candidate_solution})$
 6. **if** ($\text{cost}(\text{new_solution}) < \text{cost}(\text{candidate_solution})$) **then**
 7. $\text{candidate_solution} := \text{new_solution}$
 8. **end if**
 9. **end while**
-

2.1.2. Giải thuật luyện thép:

Giải thuật này mô phỏng quá trình luyện kim, nó lợi dụng tính chất của quá trình này: “Điều khiển “nhiệt độ” của khối kim loại dẫn tới sự cải thiện tính chất kim loại”.

Giải thuật luyện thép trong trường hợp này rất giống với tìm kiếm cục bộ, nhưng có một điểm khác biệt quan trọng: Thay vì chấp nhận giải pháp tốt hơn là giải pháp hiện tại, giải thuật luyện thép chấp nhận cả những lời giải tồi hơn với những điều kiện nhất định, gọi là nhiệt độ T . Đặc điểm này giúp giải thuật luyện thép vượt qua được giải pháp tối ưu cục bộ thường gặp phải ở giải thuật tìm kiếm cục bộ ở trên.

2.2. Giải thuật gốc lân cận biến thiên (Variable Neighborhood Descent) và tìm kiếm lân cận biến thiên (Variable Neighborhood Search)

2.2.1. Giải thuật Variable Neighborhood Descent

Đây là phương pháp sử dụng tìm kiếm hàng xóm khác nhau hướng tới sự cải thiện của giải pháp. Với bài toán này, lời giải khởi tạo được tạo ra trên đồ thị G sử dụng giải thuật heuristic đã được trình bày ở trên. Sau đó, hàng xóm đầu tiên của đồ thị

giải pháp G_s được xét tới, giải pháp nào cải thiện hơn giải pháp hiện tại sẽ trở thành giải pháp hiện tại. Một khi quá trình tìm kiếm hàng xóm này không đạt được cải thiện, sẽ tiếp tục tìm kiếm các hàng xóm khác một cách lần lượt. Cuối cùng, giải thuật sẽ đưa ra lời giải tốt nhất trong số các lân cận của giải pháp hiện thời.

Giải thuật Variable Neighborhood Descent

1. $G_s = (V_s, E_s)$
 2. $G_s := \text{get_heuristic_solution}(G=(V,E))$
 3. **while** chưa đạt tới số lần lặp mong muốn **do**
 4. **repeat**
 5. $G'_s =$ Đưa ra giải pháp tốt hơn từ hàng xóm lân cận
 6. **until** không có giải pháp nào tốt hơn giải pháp hiện tại
 7. $G'_s := \text{direct_path_replacement}(G_s) // \text{thay thế key path}$
 8. **if** $((G'_s < G_s))$ // nếu giải pháp không được cải thiện
 9. Break //stop
 10. **end if**
 11. **end while**
-

2.2.2. Giải thuật Variable Neighborhood Search

Giải thuật Variable Neighborhood Search là phương pháp meta-heuristic sử dụng thay đổi hàng xóm một cách có hệ thống trong giải thuật Local Search đã trình bày ở trên.

Trong bài toán cụ thể, giải thuật Variable Neighborhood sử dụng Variable Neighborhood Descent. Giải pháp khởi tạo $G_s=(V_s, E_s)$ cho đồ thị ban đầu $G = (V, E, c)$ được tạo ra từ phương pháp heuristic ở trên. Sau đó, áp dụng vào đồ thị con giải pháp. Một khi nó không tạo nên sự cải thiện, G_s được thay đổi 1 cách ngẫu nhiên sử dụng 1 trong các phép toán di chuyển sang lân cận, hoặc sử dụng multi-move. Hay còn gọi là “shaking”. Variable Neighborhood Descent được áp dụng một lần nữa. Quá trình “shaking” G_s áp dụng Variable Neighborhood Descent được lặp đi lặp lại, với sự thay đổi số lần “shaking” G_s mỗi lần được tăng dần, tới 1 giá trị cực đại. Khi đạt tới giá trị cực đại, giá trị “shaking” được thiết lập lại, và quá trình mới bắt đầu. Vòng lặp này được thực hiện tới khi đạt tới số lần thực hiện mong muốn.

Giải thuật Variable Neighborhood Search

1. $G_s = (V_s, E_s)$
-

2. $G_s = \text{get_heuristic_solution}(G(V, E))$
 3. $\text{num_shakes} = 1$
 4. **while** đạt tới số lần lặp tối đa **do**
 5. $G_s = \text{variable_neighborhood_descent}(G_s)$
 6. $\text{Shake}(G_s, \text{num_shakes})$
 7. $\text{Num_shakes} = (\text{num_shakes}) \bmod (\text{max_num_shakes}) + 1$
 8. **end while**
-

CHƯƠNG 4: ỨNG DỤNG GIẢI THUẬT HEURISTIC GIẢI BÀI TOÁN THIẾT KẾ MẠNG CHỊU LỖI

1. Các hướng tiếp cận giải bài toán SNDP

1.1. Phát biểu bài toán SNDP

Như đã trình bày ở chương 2, bài toán mạng chịu lỗi đã được mô hình hóa về dạng đồ thị : Cho một đồ thị vô hướng với chi phí trên các cạnh và các khách hàng có yêu cầu kết nối đến mạng cơ sở đã tồn tại (trong đó chia làm 2 loại khách hàng: loại 1 không yêu cầu độ dư thừa kết nối, loại 2 yêu cầu độ dư thừa.) Bài toán đặt ra là tìm một đồ thị con có chi phí thấp đảm bảo thỏa mãn kết nối của các khách hàng.

Điều kiện thỏa mãn kết nối được đưa ra chi tiết:

- Đối với khách hàng loại 1 (không yêu cầu độ dư thừa): Chỉ cần kết nối đơn (1 đường - path) đến nút gốc.
- Đối với khách hàng loại 2(yêu cầu độ dư thừa): chứa tối thiểu 2 đường rời rạc đến nút gốc. Rõ ràng, đối với khách hàng loại 2 này, khi một thành phần mạng (link or node) bị lỗi, vẫn còn một đường kết nối đến nút gốc.

Bài toán SNDP có thể xem như một dạng mở rộng của vấn đề cây Steiner, do đó nó cũng được biết đến như vấn đề mạng Steiner. Đương nhiên, SNDP thuộc lớp NP-khó, bởi vì SNDP bao gồm trường hợp đặc biệt của cây Steiner, thậm chí các dạng đơn giản nhất của vấn đề mạng Steiner cũng thuộc NP-khó. Vì vậy, việc tìm ra phương pháp xấp xỉ hiệu quả cho bài toán này được đặc biệt quan tâm. Như đã biết, bài toán tối ưu hóa tổ hợp là một lĩnh vực được quan tâm chủ yếu trong giới khoa học cũng như công nghiệp. Do vậy, một số thuật toán đã được phát triển để giải quyết chúng. Các giải thuật này có thể chia làm hai loại: giải thuật đúng, xấp xỉ và mẹo giải heuristic. Các giải thuật đúng luôn đảm bảo rằng sẽ tìm ra lời giải tối ưu cho bài toán. Tuy nhiên nó chỉ thích hợp với các bài toán kích thước hạn chế. Mẹo giải heuristic khắc phục được hạn chế này. Ý tưởng chung của mẹo giải heuristic là tìm lời giải gần đúng nhất trong một thời gian chấp nhận được. Đối với các loại bài toán NP-khó, khi chưa tìm được thuật toán có thời gian đa thức để giải quyết chúng, thì người ta hướng đến việc tìm ra lời giải xấp xỉ gần đúng sử dụng các giải thuật heuristic.

1.2. Tổng quan về bài toán SNDP

Vấn đề thiết kế mạng chịu lỗi nói chung đã được nghiên cứu từ những năm 60 của thế kỷ trước. Cho đến nay, đã có nhiều biến thể của bài toán chịu lỗi này. Mỗi một biến thể của bài toán được đi sâu nghiên cứu và đưa ra các giải thuật cụ thể áp dụng

cho mỗi mô hình. Tất cả các giải thuật tương ứng với mỗi hướng tiếp cận được mô tả chi tiết trong “Design of survivable network: A survey” năm 2005^[11].

Xét mô hình mạng chịu lỗi kinh điển được đưa ra đầu tiên. Trong mô hình này, ứng với một nút mạng u sẽ được gán giá trị không âm $r(u)$ đại diện cho độ quan trọng của nó trong mạng. Điều kiện mạng chịu lỗi được phát biểu: tồn tại tối thiểu $r(s, t) = \min\{r(s), r(t)\}$ đường rời rạc (không có nút chung và cạnh chung) giữa bất kỳ cặp nút s, t nào trong mạng. Nhiều giải thuật chính xác, xấp xỉ được đưa ra để giải quyết mô hình này. Trong đó, phải kể đến là giải thuật chính xác dựa trên phương pháp branch-and-cut và branch-and-bound của Chou and Frank (1992) và Hsu and Kao(1990), heuristic của Kenneth Steiglitz (1969). Tuy nhiên các giải thuật này cũng chỉ mới giải quyết với mạng nhỏ, số lượng nút ít. Gần đây, với sự phát triển của công nghệ thông tin và nhu cầu truy cập của con người cũng thay đổi, mô hình chịu lỗi này hầu như không còn phù hợp trong thực tế. Việc tồn tại độ dư thừa đường đi giữa các nút trong mạng quá nhiều sẽ gây lãng phí và không cần thiết. Hơn nữa, mạng truyền thông cục bộ thường có thêm thuộc tính đơn giản: kích cỡ mạng không có quá 100 nút, chúng gần như hai chiều, và bậc của các đỉnh thấp. Mặt khác, hầu hết các thành phần mạng hoạt động độc lập và khả năng lỗi khá thấp, chúng ta tiên liệu rằng rất ít thể hiện có khả năng chịu lỗi nhiều hơn 4, mà hầu hết các trường hợp, chúng ta thường đòi hỏi tồn tại lỗi không quá một hoặc hai thành phần mạng. Tóm lại, yêu cầu kết nối mạng được rút gọn như sau: “hầu hết các kết nối khai triển xung quanh các cluster cục bộ, và các kết nối non-local triển khai trung tâm nào đó của cluster”. Do đó, giải thuật cho lớp mạng truyền thông với đặc điểm đơn giản vừa trình bày ở mục 1.1 trong chương này được quan tâm đặc biệt và dần thay thế mô hình mạng chịu lỗi kinh điển ban đầu.

Đề án này tập trung giải quyết vấn đề thiết kế chịu lỗi cho mạng truy cập (last mile). Đây chính là bài toán giải quyết vấn đề kết nối thêm các khách hàng vào mạng dây cable có sẵn.

Hướng tiếp cận giải quyết bài toán thiết kế chịu lỗi cho mạng truy cập (last mile) liên quan đến vấn đề lập kế hoạch thực tế (OPT - Operative Planning Task), nó liên quan trực tiếp đến vấn đề cây Steiner và tăng độ kết nối cho một mạng cụ thể. Phần này, sẽ đưa ra các hướng đã giải quyết vấn đề này trong thời gian gần đây.

1.2.1. Hướng tiếp cận giải chính xác và xấp xỉ

Năm 2007, Wagner et al. [3] mô hình hóa bài toán về dạng quy hoạch tuyến tính nguyên (integer liner program - ILP) và giải quyết bằng phương pháp mạng đa luồng mở rộng (MCF- extended multi-commodity network flow). Phương pháp này đã giải quyết với mạng có kích thước 190 nút, 377 cạnh nhưng trong đó chỉ có 6

khách hàng được kết nối thêm vào mạng. Hướng tiếp cận này không thích hợp với các thể hiện mạng thực tế với số lượng node khách hàng lớn.

Sau đó Wagner cũng đưa ra công thức khác dựa trên các ràng buộc kết nối. Ông đã sử dụng giải thuật branch-and-cut và tìm được giải pháp tối ưu cho thể hiện gồm 190 nút, 377 cạnh và 13 nút khách hàng.

1.2.2. Hướng tiếp cận giải thuật heuristic

Hervé Kerivin ^[11] đã chỉ ra rằng, trừ khi $P=NP$, không có giải thuật thời gian đa thức nào đảm bảo tìm ra lời giải tối ưu hay bài toán SNDP là bài toán thuộc lớp NP-khó.

Các hướng tiếp cận giải chính xác bài toán SNDP như đã trình bày ở trên chỉ phù hợp với các thể hiện có kích thước nhỏ hoặc vừa phải (dưới 100 nút).

Để tiếp cận được với các thể hiện lớn hơn thì người ta đề xuất ra các heuristic. Heuristic bỏ qua việc đảm bảo luôn tìm thấy lời giải tối ưu để trả lời về một lời giải chấp nhận được trong thời gian cho phép. Thomas Bucsics ^[9] đưa ra hướng tiếp cận heuristic, meta-heuristic giải quyết vấn đề (năm 2007). Kết quả đã giải quyết được với thể hiện lên đến 400 nút.

Sau đó, năm 2008, Leiner ^[6] tiếp cận giải bài toán theo hướng kết hợp phân tích Lagrangian và giải thuật heuristic, meta-heuristic. Ông đã mô hình hóa về bài toán quy hoạch tuyến tính nguyên trừu tượng và áp dụng phân tích Lagrang để thu thập các giới hạn dưới như là giải pháp khả thi.

Tiếp theo, ở mục 2 sẽ trình bày chi tiết giải thuật đã có và giải thuật đề xuất để giải quyết bài toán mạng chịu lỗi.

Trước hết, chúng ta quy ước một số ký hiệu sẽ được sử dụng:

- V : tập đỉnh của đồ thị ban đầu
- E : tập cạnh của đồ thị ban đầu
- C : tập các đỉnh khách hàng
- J : tập các đỉnh trong mạng đã tồn tại (qua bước tiền xử lý được rút gọn thành 1 đỉnh duy nhất)
- C_1 : tập các đỉnh khách hàng loại 1 (loại khách hàng không yêu cầu độ dư thừa kết nối)
- C_2 : tập các đỉnh khách hàng loại 2 (loại khách hàng yêu cầu độ dư thừa kết nối)
- S : tập các đỉnh trong không gian có thể sử dụng để kết nối
- G_s : đồ thị giải pháp cho bài toán

- $G_T = (V_T, E_T)$: cây Steiner
- AUG_c : tập các cạnh trong đồ thị tạo thành đường đi dư thừa (kết nối) từ đỉnh khách hàng c đến đỉnh J

2. Giải thuật heuristic giải bài toán SNDP

2.1. Giải thuật tổng quát cho bài toán

Nhận thấy rằng, việc kết nối tất cả các khách hàng đến nút gốc ban đầu có thể được thỏa mãn bằng việc tìm cây Steiner trong đồ thị ban đầu $G = (V, E, c)$ với tập các đỉnh J và các đỉnh khách hàng. Nói cách khác, tất cả các đỉnh trừ các đỉnh S có trong đồ thị G ban đầu đều là các terminal node.

Sau khi cây Steiner $T = (V_T, E_T)$ được đưa ra, để thỏa mãn yêu cầu độ dư thừa kết nối cho các đỉnh khách hàng loại 2 (C_2), thì ngoài đường từ đỉnh $c \in C_2$ đến J đã có trong cây Steiner (gọi là đường đi đầu tiên của c), cần phải tìm đường đi dư thừa từ c đến J mà không sử dụng lại bất kỳ đỉnh hoặc cạnh nào trong đường đi đầu tiên của c . Chúng ta gọi quá trình này là “Augmentation”.

$$AUG_{edges} = \bigcup_{i \in C_2} AUG_i$$

Khi đó giải pháp khởi tạo thu được là $G_S = (V_T \cup AUG_{nodes}, E_T \cup AUG_{edges})$

Tóm lại, heuristic tổng quát sử dụng ý tưởng trước hết xây dựng cây Steiner, sau đó thực hiện tăng luồng (chính là đáp ứng yêu cầu tăng độ dư thừa kết nối cho các khách hàng loại 2), cuối cùng thực hiện tìm kiếm lân cận để thu được giải pháp tốt nhất cho bài toán.

Thuật toán heuristic tổng quát được mô tả như sau:

Thuật toán heuristic tổng quát

1. **Init** (khởi tạo):
 2. $Find_minimum_Steiner_tree()$; // Tìm Steiner tree có trọng số bé nhất
 3. $G_S = Update()$;
 4. // Bổ sung 1 kết nối dự phòng cho mọi đỉnh loại C_2
 5. // S là lời giải ban đầu nhận được từ 2 bước trên.
 6. **Process** (Dùng Local search để tìm kết quả tối ưu)
 7. **repeat**
 8. $Find(G_S')$; // Tìm lời giải G_S' nhận được từ G_S qua 1 phép biến đổi mà S' tốt hơn S
 9. **if** (G_S' không tồn tại) **then Exit**;
 10. $G_S = G_S'$;
 11. **until True**;
-

12. **Return** G_S ; // Trả về lời giải cho bài toán là G_S

Phần tiếp theo, trình bày chi tiết bước khởi tạo và cải tiến giải pháp theo thuật toán heuristic tổng quát trên.

2.2. Giải thuật xây dựng cây Steiner

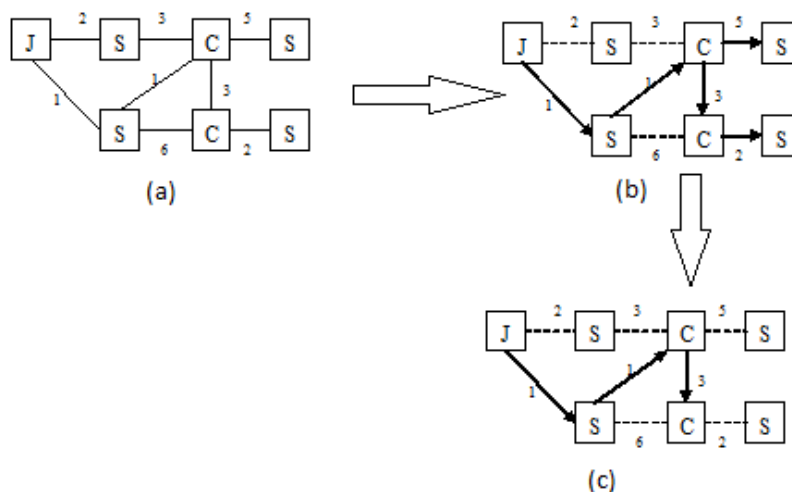
Như đã được trình bày ở trên, giải thuật heuristic xây dựng giải pháp sẽ tạo ra cây Steiner sử dụng các nút (J U C) là “terminal node”, và sau đó tăng đường đi dư thừa cho các đỉnh thuộc C_2 . Phần này trình bày một số giải thuật để xây dựng cây Steiner từ đồ thị ban đầu. Trong đó, giải thuật ở mục 2.2.1, 2.2.2, 2.2.3 là các giải thuật đưa ra bởi Thomas Busics^[9], giải thuật ở mục 2.1.4, 2.1.5 là hai giải thuật đề xuất.

Để tiện cho việc theo dõi giải thuật, chúng ta sẽ nhắc lại một số định nghĩa cũng như quy định các ký hiệu được sử dụng:

- “junction node”: Đỉnh gốc đại diện cho toàn bộ mạng đã có sẵn
- “terminal node”: tập các đỉnh gốc “junction node” và các đỉnh khách hàng C

2.2.1. Heuristic tìm đường đi ngắn nhất từ một nguồn đơn (SSSP: Single Source Shortest Path)

Heuristic này xây dựng cây Steiner $T = (V_T, E_T)$ bao gồm các đường đi ngắn nhất từ đỉnh “junction node” J đến tất cả các đỉnh khác trong đồ thị ban đầu $G = (V, E)$. Sau đó trong quá trình thanh lọc (purging) đồ thị giải pháp, ứng với mỗi nút $c \in C$, thêm các đường đi từ J đến c trong cây T vào đồ thị giải pháp G_S .



Hình 11: Ví dụ giải thuật SSSP

(a) Đồ thị ban đầu. (b) Cây đường đi ngắn nhất (c) Cây Steiner

Heuristic SSSP sử dụng giải thuật tìm đường đi ngắn nhất Dijkstra (đã được trình bày ở chương 1 mục 4.1) để xây dựng cây khung nhỏ nhất cho đồ thị ban đầu.

2.2.2. Heuristic tìm cây khung nhỏ nhất (MST: Minimum Spanning Tree)

Đây là heuristic tham lam dựa trên giải thuật tìm cây khung nhỏ nhất Kruskal (đã được trình bày ở Chương 1 mục 4.2). Giải thuật này tương tự như giải thuật heuristic SSSP. Điểm khác biệt so với SSSP là cây khung nhỏ nhất được sử dụng thay thế cây đường đi ngắn nhất, và tập các cạnh được tạo ra bởi giải thuật Kruskal phải được biến đổi thành cây với “junction node” là đỉnh nguồn. Điều này thực hiện bằng việc chạy giải thuật tìm kiếm theo chiều sâu trên đồ thị ban đầu, vượt qua chỉ các cạnh được đưa ra trong tập các cạnh trong cây khung nhỏ nhất. Quá trình thanh lọc giải pháp vào đồ thị giải pháp cũng tương tự như giải thuật SSSP.

Ta có thể nhận thấy rằng trong suốt quá trình xây dựng cây khung nhỏ nhất, có rất nhiều đỉnh kết nối đến đỉnh “junction node” mà không được sử dụng trong đồ thị giải pháp sau quá trình thanh lọc giải pháp. Có nhiều phương pháp để giải quyết vấn đề này. Hướng tiếp cận giải quyết hiệu quả vấn đề này có thể xem chi tiết trong ^[12].

2.2.3. Xây dựng cây khung nhỏ nhất sử dụng đường đi ngắn nhất giữa tất cả các cặp đỉnh (APSP: All-Pairs-Shortest-Path)

Giải thuật heuristic khác để xây dựng cây Steiner sử dụng đường đi ngắn nhất giữa 2 đỉnh. Điều này có thể thực hiện bởi thuật toán Floyd-Warshall (đã được trình bày ở chương 1 mục 4.3), nhưng bởi vì trong thực tế đối với vấn đề này, tỷ lệ giữa các terminal node và Steiner node (đỉnh không gian) là khá thấp, và không cần thiết phải tìm ra tất cả các đường đi ngắn nhất giữa các “terminal node” trong quá trình xây dựng cây Steiner. Vì vậy, ở đây sẽ sử dụng giải thuật Dijkstra (đã được trình bày ở chương 1 mục 4.1) để tìm đường đi ngắn nhất cho mỗi “terminal node”. Do đó, có thể truy nhập đến các đường đi ngắn nhất giữa các cặp đỉnh trong “terminal node”.

Phần tiếp theo sẽ trình bày ba biến thể của giải thuật xây dựng cây Steiner từ tập các đường đi ngắn nhất giữa các “terminal node”.

All-Pairs-Shortest-Path (APSP): Đầu tiên, sắp xếp khoảng cách giữa các “terminal node” theo đường đi ngắn nhất. Sau đó, tính toán cây khung nhỏ nhất (sử dụng giải thuật Kruskal) dựa trên các khoảng cách đó, cuối cùng trả lại các cạnh từ tất cả các đường đi ngắn nhất tương ứng với mỗi cạnh rút gọn đã được chọn vào bởi giải thuật cây khung nhỏ nhất. Tập cạnh này được biến đổi thành cây với đỉnh nguồn là “junction node”. Đây chính là cây Steiner thỏa mãn cần xây dựng.

All-Pairs-Shortest-Path-extended (APSPe): Giải thuật này là một biến thể khác cho việc sử dụng cây khung nhỏ nhất trên tập các cạnh đã được rút gọn trong

giải thuật APSP. Nó sử dụng tập các cạnh được chứa trong các đường đi ngắn nhất để quyết định cây khung nhỏ nhất cho đồ thị giải pháp.

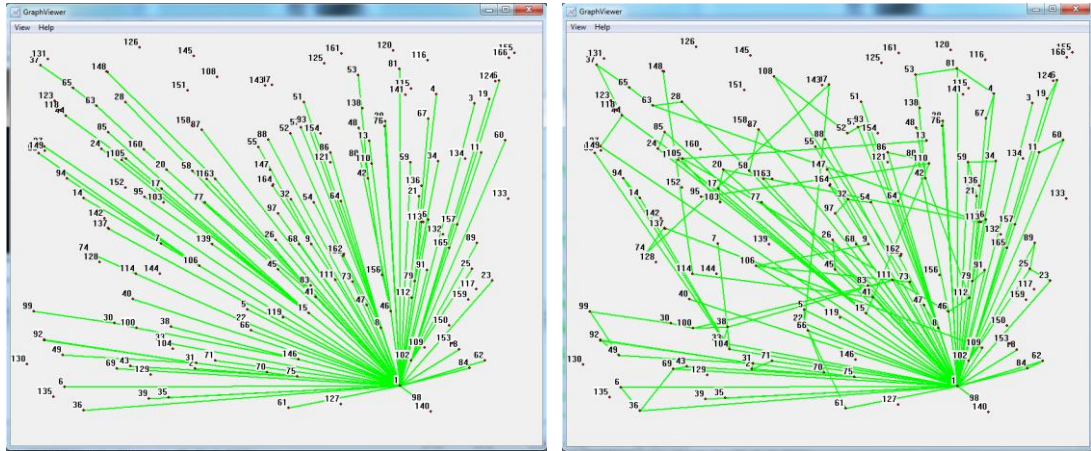
All-Pairs-Shortest-Path-adapting (APSPx): Giải thuật này hoàn toàn tương tự như APSP ngoại trừ trong thực tế các đường đi ngắn nhất sẽ xảy ra trường hợp chia sẻ cạnh. Khi đường đi P được thêm vào trong cây khung nhỏ nhất, cần phải kiểm tra các đường còn lại R chưa được thêm có hay không có các cạnh chia sẻ. Nếu có cạnh chia sẻ, giá trị khoảng cách ngắn nhất của R cần được cập nhật ngay. Thuật toán này được mô tả như sau:

Giải thuật APSPx

1. *Giải pháp khởi tạo $G_s = (V_s, E_s)$*
 2. $G_s = (\phi, \phi)$
 3. *Đưa ra cây đường đi ngắn nhất giữa các “terminal node” đến node gốc (J)*
 4. *Sắp xếp các cặp node theo khoảng cách giảm dần vào hàng đợi Q*
 5. **while** ($Q \neq \phi$) **do**
 6. $g :=$ đưa ra và xóa cặp điểm nào ngắn nhất trong Q
 7. $P :=$ đưa ra đường đi giữa cặp điểm này (g)
 8. **if** P chứa terminal node mà chưa được add vào bất kỳ đường nào khác **then**
 9. *Thêm P vào đồ thị giải pháp G_s*
 10. **for all** cặp h vẫn còn trong hàng đợi **do**
 11. $R =$ cặp có đường đi ngắn nhất trong h
 12. **for all** cạnh $e \in P \cap R$ chưa bị đánh dấu **do**
 13. *Thiết lập lại k/c mới(h) = k/c cũ (h) – $c(e)$*
 14. *Đánh dấu cạnh e*
 15. **end for**
 16. **end for**
 17. **end if**
 18. **end while**
 19. **return** G_s
-

Giải thuật giữ lại vết của các đường đi từ đỉnh gốc đến các đỉnh khách hàng và chi phí trên đường đi này. Mỗi lần thêm một đường đi vào đồ thị giải pháp, nói chung sẽ làm thay đổi chi phí trên các đường đi còn lại. Do vậy, chi phí thời gian cần để cập nhật cấu trúc dữ liệu, trong trường hợp này, tồi nhất là $O(n^2)$. Bước này được lặp lại $n-1$ lần (số các đường đi cần thêm vào), vì vậy thời gian chi phí là $O(n^3)$.

Chất lượng của cây Steiner phụ thuộc nhiều vào chi phí giữa các đỉnh khách hàng, và một phần đường đi trên nó đã được sử dụng. Tổng chi phí thời gian của toàn bộ quá trình là $O(n^3)$.



Hình 12: Cây Steiner và mạng tương ứng sinh bởi giải thuật APSP($n=100, m=65$)

Nhận xét: Khi APSP được áp dụng trong trường hợp bài toán có các tập đỉnh khách hàng có thể kết nối với nhau với chi phí nhỏ hơn, cây Steiner có thể có chi phí lớn bởi vì thuật toán này lãng phí đối với trường hợp có những đường đi đã được thêm trước vẫn có thể sử dụng là một phần để nối với đỉnh khách hàng chưa được thêm vào đồ thị giải pháp.

2.2.4. Giải thuật APSP đề xuất

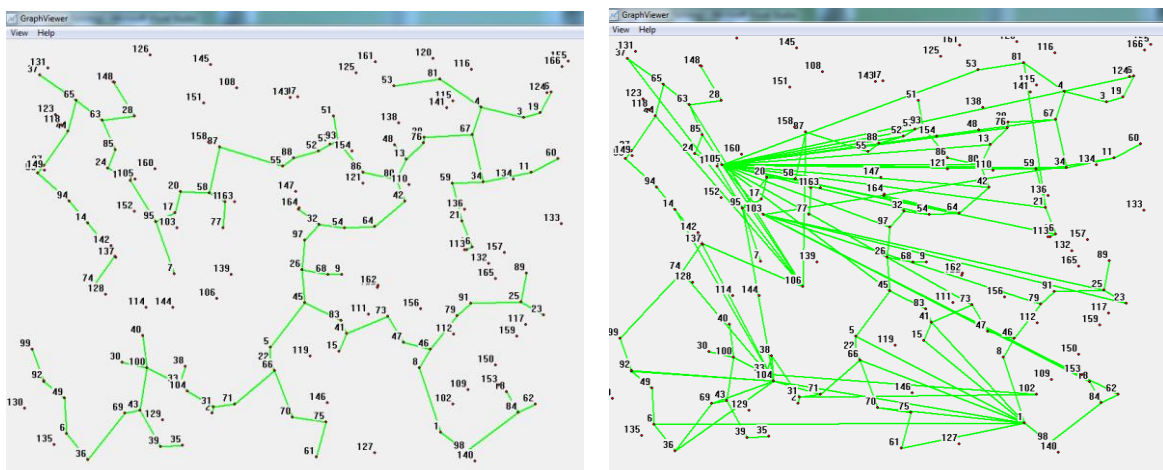
Chúng ta nhận thấy, với giải thuật APSP ở trên khi xây dựng cây Steiner sử dụng cây đường đi ngắn nhất giữa các “terminal node” đến node gốc J sẽ tạo ra cây Steiner với chi phí lớn khi không tận dụng được các cạnh trong đường đi ngắn nhất đã được thêm trước. Các đường đi ngắn nhất này là cố định không thay đổi khi thêm lần lượt nó vào đồ thị giải pháp. Vì vậy có một phương pháp để giảm thiểu chi phí xây dựng cây Steiner chính là sau khi chọn và thêm vào đồ thị giải pháp đường đi có khoảng cách ngắn nhất trong tất cả các đường đi ngắn nhất từ các đỉnh khách hàng đến đỉnh “junction node”, chúng ta sẽ cập nhật lại khoảng cách từ các đỉnh xuất hiện trong đường đi này đến đỉnh “junction node” (được gán bằng 0 bởi vì nếu sử dụng lại đường đi từ J đến đỉnh này thì không làm tăng chi phí). Chúng ta sẽ lặp lại cho đến khi không còn đỉnh khách hàng nào chưa được thêm vào giải pháp. Thuật toán cải tiến được mô tả như sau:

Giải thuật APSP-N

1. Giải pháp khởi tạo $G_s = (V_s, E_s)$
2. $G_s = (\phi, \phi)$

3. Đưa ra cây đường đi ngắn nhất giữa các “terminal node” đến node gốc (J)
4. Chọn u là đỉnh có khoảng cách ngắn nhất đến J
5. $P :=$ đưa ra đường đi giữa cặp điểm (u, J)
6. Add P vào G_s
7. $Q = (C_1 \cup C_2) \setminus u$
8. **while** ($Q \neq \emptyset$) **do**
9. **for all** $v_s \in P$
10. cập nhật khoảng cách (J, v_s) = 0
11. **end for**
12. Tìm cây đường đi ngắn nhất giữa các đỉnh trong Q đến J
13. $r =$ đỉnh có khoảng cách ngắn nhất đến J
14. $R =$ đường đi giữa (r, J)
15. add R vào G_s
16. $Q := Q \setminus r$
17. **end while**
18. **return** G_s

Thuật toán chạy cứ mỗi lần thêm một đường đi trong cây đường đi thì lại chạy lại thuật toán Dijkstra một lần. Như đã trình bày ở chương 1 mục 4.1, thuật toán Dijkstra có độ phức tạp tính toán là $O(n^2)$. Bước này cũng được lặp lại $(n-1)$ lần (chính là số các đường cần thêm vào đồ thị giải pháp). Vì thế tổng chi phí thời gian của toàn bộ quá trình là $O(n^3)$.



Hình 13: Cây Steiner và Mạng tương ứng sinh bởi giải thuật APSP-NI($n=100, m=65$)

2.2.5. Xây dựng cây khung nhỏ nhất sử dụng giải thuật kết nối các thành phần component

Đầu tiên ta chia đồ thị $G = (V, E, c)$ vào trong tập các thành phần liên thông với nhau, mỗi thành phần chứa 1 “terminal node”. Khởi tạo ban đầu có $|T|$ thành phần như vậy, mỗi lần lặp tìm hai thành phần có khoảng cách bé nhất và kết nối 2 thành phần này lại với nhau. Hai thành phần này sẽ hợp thành một thành phần lớn hơn. Lặp lại quá trình này cho đến khi mọi thành phần được hợp thành một thành phần duy nhất chứa tất cả các nút trong T .

Giải thuật MST

1. *Random()* // Chia các đỉnh vào T component
 2. **repeat**
 3. **if** (chỉ còn duy nhất một component) **then**
 4. *exit*
 5. **end if**
 6. *Find_component()* // tìm hai component gần nhau nhất
 7. *Union_component()* // hợp nhất hai component tìm được thành một component
 8. **until true**
-

Để hợp nhất hai component thành một component duy nhất, ta sẽ sử dụng cấu trúc dữ liệu “Disjoin set”

“Disjoin set” là một cấu trúc dữ liệu thường được lưu trữ dưới dạng một rừng, tức là gồm nhiều cây, mỗi cây đại diện cho một tập hợp. Gốc của cây là một nút bất kỳ đại diện cho cả tập hợp. Các phần tử khác của tập hợp được liên kết với phần tử đại diện thông qua một số liên kết. Như vậy:

- Để tìm tập hợp của một phần tử chỉ cần tìm về nút gốc của cây chứa nó $\text{Findset}(i)$
- Để hợp nhất hai tập hợp chỉ cần hợp nhất hai nút gốc của hai cây đại diện cho hai tập $\text{Union}(i, j)$. Ta có thể lưu trữ rừng cây này bằng danh sách liên kết động hoặc mảng. Nhưng để đơn giản ta sẽ sử dụng mảng $\text{dad}[i]$ lưu trữ cha trực tiếp của đỉnh i . Riêng với nút gốc J thì cha của nó là chính nó. Chúng ta sẽ lợi dụng điều này làm điều kiện kết thúc truy vấn ngược từ một nút con lên nút cha).
- Các function cho cấu trúc dữ liệu này:

Function $\text{Find}(i: \text{integer})$ // tìm component chứa i

1. **while** ($i \neq \text{Set}[i]$) **do**
2. $i = \text{Set}[i]$
3. **end while**
4. **return** i

Câu lệnh 2 của hàm này thực hiện lặp lại n lần. Độ phức tạp tính toán của hàm $\text{Find}(i)$ là $O(n)$

Function $\text{Union}(i, j: \text{integer})$ // kết nối 2 component i và j với nhau

1. $\text{int } u, v$
2. $u = \text{Find}(i)$
3. $v = \text{Find}(j)$
4. **if** ($u < v$) **then**
5. $\text{Set}[v] = u$
6. **else**
7. $\text{Set}[u] = v$
8. **end if**

Câu lệnh 2, 3 ở trong hàm $\text{Union}(i, j)$ thực hiện với chi phí tối nhất $O(n)$. Toàn bộ chi phí để thực hiện hàm này là $O(n)$.

Function $\text{visit_tree}(i: \text{integer})$ // duyệt cây để xác định lại các quan hệ cha con

1. **for all** $v \in \text{đỉnh kề}(i)$ **do**
2. $\text{dad}(v) := i$
3. $\text{NChild}(i)++$
4. **end for**

Chi phí để thực hiện hàm $\text{visit_tree}(i)$ là $O(n)$

Tóm lại, để chạy giải thuật MST xây dựng cây Steiner ban đầu sử dụng cấu trúc dữ liệu Disjoin set, chúng ta cần chi phí tính toán tổng cộng là $O(n^2)$

2.3. Tỉa cây để loại bỏ các node không cần thiết trong cây Steiner.

Sau bước xây dựng cây Steiner, chúng ta thu được đồ thị kết nối đỉnh gốc “junction node” và các đỉnh khách hàng C . Nếu để ý đối với giải thuật MST, cây Steiner thu được sẽ tồn tại các nút lá không quan trọng (không phải terminal node). Như vậy, việc loại bỏ chúng đi có thể giảm được chi phí mà ràng buộc kết nối của mạng

không bị ảnh hưởng. Vì thế cần có thuật toán để loại bỏ các nút này khỏi cây Steiner để thu được giải pháp có chi phí nhỏ hơn.

Giải thuật tĩa cây (Pruned tree) – Visit_tree()

1. *repeat*
 2. *cut_node()*
 3. *if* (các node lá đều là “terminal node”) *then*
 4. *exit*
 5. *else*
 6. *Visit_tree ()*
 7. *end if*
 8. *until true*
-

Thuật toán tĩa cây trên sử dụng hàm của cấu trúc dữ liệu “Disjoin set” là visit_tree() như đã nói ở mục 2.2.5. Giải thuật tĩa cây này cũng cần thời gian là $O(n^2)$.

2.4. Thêm kết nối dự phòng cho các đỉnh khách hàng loại hai (C2)

Sau khi xây dựng cây Steiner $T = (V_T, E_T)$ từ đồ thị ban đầu $G = (V, E, c)$, để thu được giải pháp thỏa mãn các ràng buộc kết nối, bước tiếp theo cần thiết lập độ dư thừa kết nối cho tất cả các đỉnh khách hàng loại hai $c_2 \in C_2$. Phần này sẽ trình bày hai hướng tiếp cận, cả hai hướng này đều sử dụng giải thuật tìm đường đi ngắn nhất Dijkstra để tạo ra đường kết nối mới, riêng biệt từ các đỉnh khách hàng loại hai đến nút gốc.

2.4.1. Hướng tiếp cận thứ nhất(AugSP: Augmentation by Shortest Path)

Phương thức đơn giản nhất để thêm kết nối dự phòng cho mỗi đỉnh khách hàng loại hai là tìm kiếm đường đi ngắn nhất từ “junction node” J đến c_2 , sử dụng giải thuật Dijkstra và bỏ qua tất cả các nút thuộc V_P trừ J và C_2 (Trong đó $P = (V_P, E_P)$ là đồ thị con đại diện cho đường đi hiện tại trong T mà kết nối C_2 đến “junction node”). Sau đó, đường này được thêm vào giải pháp S.

Tuy nhiên hướng tiếp cận này bỏ qua thực tế rằng T đã chứa các cạnh kề với các nút chứa trong P mà có thể được sử dụng cho kết nối dư thừa đến C_2 . Việc sử dụng các cạnh này không gây ra việc thêm chi phí kết nối, và có thể tạo ra đường dư thừa từ J đến C_2

2.4.2. Hướng tiếp cận thứ hai(AugSPe: Augmentation by Shortest Path with Pseudo-Infrastructure Extension)

Đây là một biến thể của AugSP. Với mỗi nút c_2 , đưa ra một tập các cạnh E_A kết nối tất cả các nút $V_S \setminus V_P \cup V_{\text{indirect}}$ đến J và có chi phí là 0. Trong đó, V_{indirect} là tất cả các nút kết nối đến infrasture thông qua một đường trong T có chứa một nút trong V_P . **Các node trong V_{indirect} phải không được kết nối với cạnh “free”** bởi vì thực tế các cạnh này sẽ đưa ra giải thuật xây dựng độ dư thừa có các nút tương tự được lặp lại 2 lần. Sau đó, sử dụng giải thuật Dijkstra để tìm ra một đường mới và thêm nó vào đồ thị giải pháp S .

Giải thuật AugSPe // thêm kết nối dự phòng cho các đỉnh khách hàng loại hai

1. $G_S = (V_S, E_S) := T(V_T, E_T)$
 2. **for all** $c_2 \in C_2$ **do**
 3. $P = (V_P, E_P)$
 4. $P := \text{get_path}(c_2, J, T)$ // đưa ra đường đi hiện tại từ c_2 đến “junction node” J
 5. **for all** $v_S \in (V_S \setminus (V_P \cup V_{\text{indirect}}))$ **do**
 6. $E_A := E_A \cup (j, v_S, 0)$ // thêm cạnh “free” từ J đến v_S
 7. **end for**
 8. $G_S := G_S \cup \text{find_shortest_path}(c_2, j, (G \cup E_A) \setminus (P \setminus \{c_2, j\}))$ // tìm đường mới sử dụng cạnh trong E_A , đồng thời không sử dụng các đỉnh trong đường đi hiện tại
 9. **end for**
-

Câu lệnh 8 gọi đến thuật toán Dijkstra(có thời gian tính toán là $O(n^2)$) và được thực hiện r (= số lượng khách hàng loại 2) lần. Như vậy giải thuật tăng đường AugSPe có thời gian tính tổng cộng xấp xỉ là $O(n^3)$.

2.4.3. Đề xuất cải tiến giải thuật tăng đường

Ở hướng tiếp cận trên, ta có nhận xét rằng:

Khi một khách hàng C_2 đã được tăng luồng thì ta vẫn có thể sử dụng đường đi này để thực hiện kết nối với khách hàng tiếp theo được xét mà không làm tăng chi phí lúc tăng đường.

Từ nhận xét này, giải thuật tăng đường cải tiến được đề xuất như sau

Giải thuật AugSPe-I

1. $G_S = (V_S, E_S) := T(V_T, E_T)$
2. **for all** $c_2 \in C_2$ **do**
3. **for all** c_2 đã đánh dấu **do**

4. $EA := EA \cup (j, c_2, 0)$ // thêm cạnh “free” từ J đến khách hàng đã được xét
 5. **end for**
 6. $P := (V_P, E_P)$
 7. $P := \text{get_path}(c_2, J, T)$ // đưa ra đường đi hiện tại từ c_2 đến “junction node” J
 8. **for all** $v_s \in (V_S \setminus (V_P \cup V_{\text{indirect}}))$ **do**
 9. $E_A := E_A \cup (j, v_s, 0)$ // thêm cạnh “free” từ J đến v_s
 10. **end for**
 11. $G_S := G_S \cup \text{find_shortest_path}(c_2, j, (G \cup E_A) \setminus (P \setminus \{c_2, j\}))$ // tìm đường mới sử dụng cạnh trong E_A , đồng thời không sử dụng các đỉnh trong đường đi hiện tại
 12. Đánh dấu C_2
 13. **end for**
-

Bước cải tiến này sẽ ảnh hưởng rất lớn đến chi phí của mạng khởi tạo nếu cây Steiner (xây dựng trong các giải thuật ở mục 2.3) có chiều sâu lớn.

Như vậy, tóm lại ta đưa ra được độ phức tạp tính toán cho bước xây dựng giải pháp khởi tạo như sau:

- Giải thuật APSP:
 - Chi phí xây dựng cây Steiner (APSPe) : $O(n^3)$
 - Chi phí tăng đường (AugSPe): $O(n^3)$
 - Chi phí tổng cộng: $O(n^3)$
- Giải thuật APSP-I:
 - Chi phí xây dựng cây Steiner(APSPe): $O(n^3)$
 - Chi phí tăng đường (AugSPe-I)
- Giải thuật APSP-N:
 - Chi phí xây dựng cây Steiner (APSP-N) : $O(n^3)$
 - Chi phí tăng đường(AugSPe): $O(n^3)$
 - Chi phí tổng cộng: $O(n^3)$
- Giải thuật APSP-NI:
 - Chi phí xây dựng cây Steiner (APSP-N): $O(n^3)$
 - Chi phí tăng đường(AugSPe-I): $O(n^3)$
 - Chi phí tổng cộng: $O(n^3)$
- Giải thuật MST:

- Chi phí xây dựng cây Steiner: $O(n^2)$
- Chi phí tăng đường: $O(n^3)$
- Chi phí tổng cộng: $O(n^3)$

2.5. Quá trình cải tiến giải pháp sử dụng Local Search

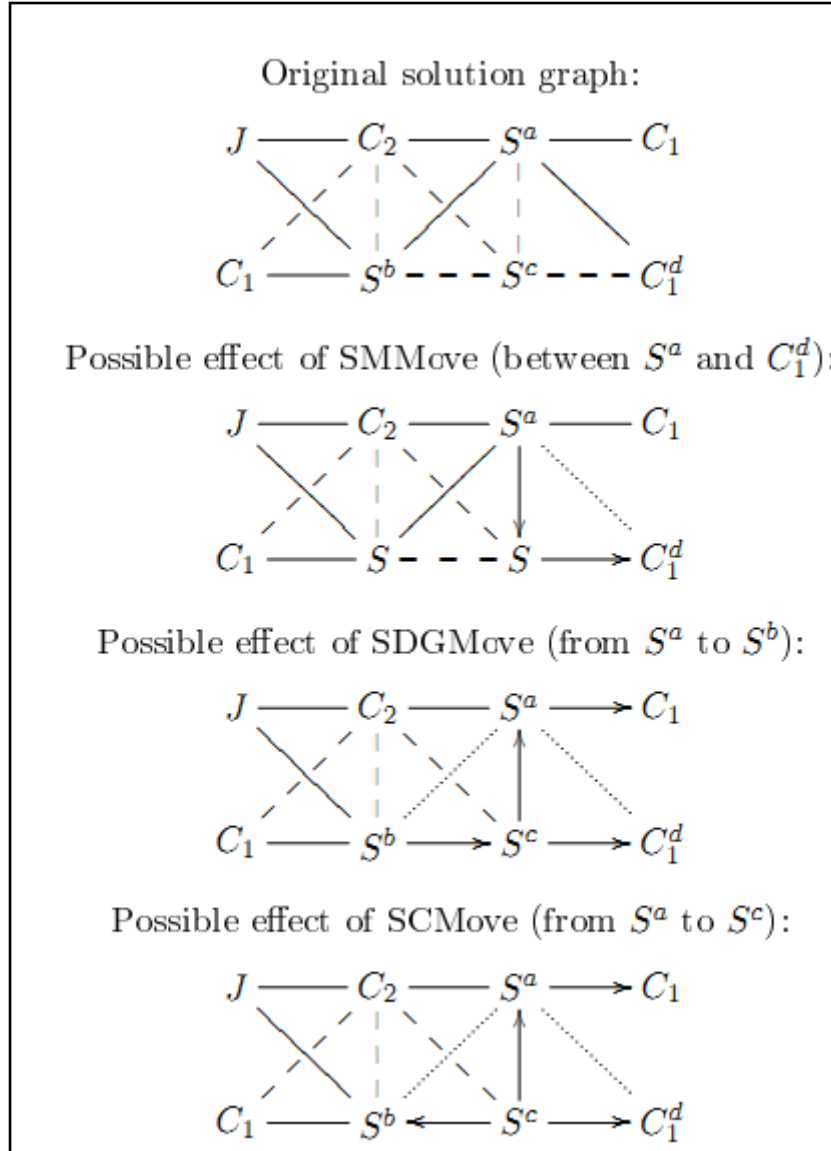
Để áp dụng tìm kiếm Local Search ở trên, chúng ta cần định nghĩa các loại hàng xóm để thay đổi lời giải mà không vi phạm tính chịu lỗi, đồng thời có thể cải thiện chi phí xây dựng. Với mục đích này, ta sẽ đưa ra ba giải thuật giúp cải thiện giải pháp khởi tạo trên:

- SMMove (Single Migration Move)
- SDGMove (Single Degree Move)
- SCMove (Single Connecting Move)

Mỗi giải thuật sử dụng một đồ thị giải pháp khởi tạo $G_S=(V_S, E_S)$ trên đồ thị đầy đủ G và biến đổi nó bằng những cách thức như nhau, nhưng ngẫu nhiên thay đổi giải pháp. Chúng được thiết kế để bổ sung lẫn nhau, với một trong số giải thuật này sẽ được sử dụng trong phép toán tìm kiếm cục bộ hoặc luyện thép. Chúng đều dựa trên giải thuật Dijkstra, và đều cố gắng tạo ra những lời giải khác nhau, sử dụng một trong ba phép biến đổi.

- SMMove: phép toán di chuyển này tìm đường đi $P = (V_P, E_P)$ từ một đỉnh ngẫu nhiên trên đồ thị G_S , sau đó thay thế nó, bởi đường đi ngắn nhất sử dụng giải thuật Dijkstra. Mục đích phép toán này dùng để cải thiện đường đi trong lời giải, nhưng không ảnh hưởng đến các đỉnh trong G_S có bậc trong giải pháp khởi tạo lớn hơn 2.
- SDGMove: Với phép toán di chuyển này, sử dụng một đỉnh ngẫu nhiên thuộc tập $\{C_1 \cup \{v_s \in V_S | \text{bậc}(v_s) > 2\}\}$, sau đó tìm trên mọi lân cận trực tiếp của N , trong đó $n \neq \{C_2, J\}$. Giải thuật sử dụng cây tìm kiếm theo chiều sâu, bắt đầu tại n và dừng lại khi gặp đỉnh J , đánh dấu tất cả các đỉnh $R \in V$ đi trước khi gặp đỉnh C_2 . Giải thuật loại bỏ tất cả các đường tới những đỉnh $n \in \{N \setminus R\}$ có nguồn gốc từ v , và gán n vào đỉnh ngẫu nhiên $w \in G_S$ với $w \neq \{J, C_2\}$, đồng thời nối v bởi một đường đi nằm trong G_S (là đường đi ngắn nhất có thể, bỏ qua tất cả các đỉnh đã có trong giải pháp). Sự thu gọn R là rất quan trọng trong việc bảo toàn độ dư thừa cho đường đi đến các đỉnh C_2 : một đường tới mỗi đỉnh C_2 luôn được bảo toàn còn đường khác thì có thể thay đổi. Thuật toán dừng khi $v \in C$ hoặc các kết nối không thể di chuyển. Lúc này, v đã gắn với w bởi giải thuật. Phép toán này nhằm mục đích di chuyển các đỉnh “junction node” trong đồ thị giải pháp G_S

- **SCMove:** Phép toán di chuyển này hoàn toàn tương tự với **SDGMove**. Nó cũng chọn ngẫu nhiên đỉnh $v \in \{C_1 \cup \{v_s \in V_s | \text{bậc}(v_s) > 2\}\}$, đánh dấu tất cả những đỉnh trên đường đi đến đỉnh C_2 , tìm kiếm tất cả các đỉnh dẫn đến đỉnh N với $N \neq \{C_2, J, R\}$ và được xuất phát từ v . Sau đó, thực hiện loại bỏ chúng. Tuy nhiên, nó kết nối đỉnh lân cận trực tiếp với đỉnh lân cận $w \in G$, trong đó $w \neq \{J, G_s\}$, điều này có nghĩa là giải thuật này chọn w là các đỉnh nằm bên ngoài giải pháp đã có. Phép toán này nhằm mở rộng phạm vi của bài toán tới những đỉnh không nằm trong lời giải khởi tạo.



Hình 14: Hình minh họa cho ba phép di chuyển **SMMove**, **SCMove** và **SDGMove**

Hai phép biến đổi **SCMove** và **SDGMove** có một ý tưởng chung là đa dạng hóa giải pháp bằng cách thay đổi những tập đỉnh nằm trong giải pháp cũ bằng cách sử dụng cách tìm kiếm cục bộ được giới thiệu ở trên. Các tiếp cận trong đồ án này không xây dựng heuristic cho toàn bộ bài toán, thay vào đó là tính toán trực tiếp trên đồ thị

lời giải G_S . Lý do chọn cách tiếp cận này bởi vì ở đây không chỉ cố gắng để giải bài toán cây Steiner, mà còn cố gắng tăng cường và tìm kiếm lời giải. Chính vì vậy, kết quả thể hiện trên những bài toán phức tạp hơn với giải thuật xây dựng heuristic có thời gian chạy dài hơn thời gian cần thiết để xây dựng cây khung nhỏ nhất.

Mục đích trong các phương pháp tìm kiếm lân cận này không phải là thêm vào hay xóa đi các đỉnh, ở đây quan tâm đến việc di chuyển các đỉnh. Tuy vẫn có thể thêm hoặc xóa các đỉnh: Khi một đỉnh di chuyển bởi một trong hai phép toán, các đỉnh gắn với nó được di chuyển tới một đỉnh khác đã là đỉnh được chọn trong đồ thị giải pháp. Vì vậy, kết quả thu được chứa ít hơn một đỉnh so với giải pháp khởi tạo ban đầu. Cũng có thể thêm vào một đỉnh bên ngoài khi các phép toán không có khả năng kết nối tất cả các đỉnh trong giải pháp hiện tại tới đỉnh mục tiêu, hoặc việc thêm đỉnh sẽ làm giảm chi phí. Lúc này, một số đỉnh có sẵn được giữ lại, còn một số đỉnh khác được di chuyển đến một đỉnh mới.

Những phép toán di chuyển có những tính chất rất đặc thù. Vì vậy, ở đây chúng ta sẽ sử dụng cùng một lúc nhiều phép toán (hay còn gọi là multi-move), có thể lựa chọn ngẫu nhiên thứ tự thực hiện các phép toán.

Void SMMove() //di chuyển để thay thế key-path

1. $Boundary_node := C \cup \{v_s \in V_S | bậc(v_s) > 2\} \cup J$
 2. **For all** đỉnh $v \in boundary_node$ **do**
 3. **For all** đỉnh chưa được đánh dấu $w \in boundary_node$ mà được kết nối trực tiếp trong G_S thông qua $P(V_P, E_P)$ **do**
 4. Xóa tất cả các đỉnh trong $V_P \setminus \{w, v\}$ và tất cả cạnh kề trong G_S
 5. $G_S = G_S \cup find_shortest_path(w, v, G \setminus \{w, v\})$
 6. **End for**
 7. Đánh dấu v
 8. **End for**
-

2.6. Giải quyết ràng buộc không bắt chéo (Non-Crossing) trong mặt phẳng

Để ý rằng, tất cả các giải thuật được đưa ra ở trên, chưa đề cập đến trường hợp giải quyết vấn đề trên mặt phẳng (tức là với các bộ dữ liệu Euclidean). Vì vậy, ở phần này sẽ đưa ra một số công việc cần xử lý khi áp dụng bài toán đối với bộ dữ liệu Euclidean.

Ràng buộc Non-Crossing được phát biểu như sau: Các kết nối không thể vượt qua (cắt chéo) các kết nối khác trên mặt phẳng.

Tuy nhiên, ràng buộc này sẽ không được xem xét trong các hướng tiếp cận meta-heuristic.

Như vậy để giải quyết ràng buộc này, các giải thuật tìm đường đi ngắn nhất Dijkstra và tìm cây khung nhỏ nhất Kruskal được sửa chữa để tương thích với ràng buộc “Non-crossing”. Có 4 cách để giải quyết khi gặp trường hợp bắt chéo (hai kết nối cắt nhau trên mặt phẳng) như sau:

- Cạnh có chi phí nhỏ hơn (Cheapest edge): Nếu hai cạnh chéo nhau, chỉ cạnh có chi phí nhỏ hơn được thêm vào đồ thị giải pháp.
- Cạnh có chi phí lớn hơn (Most expensive edge): Nếu hai cạnh chéo nhau, chỉ cạnh có chi phí lớn hơn được thêm vào đồ thị giải pháp.
- Cạnh ngẫu nhiên (Random edge): Nếu hai cạnh chéo nhau, lấy cạnh ngẫu nhiên trong hai cạnh này để thêm vào đồ thị giải pháp
- Cạnh đầu tiên (First edge): Khi một cạnh được thêm vào đồ thị giải pháp, tất cả cạnh đã được thêm vào mà bắt chéo cạnh này sẽ trở thành không khả thi để chọn lựa. Trong trường hợp này, tất cả các cạnh trở thành không khả thi bị gây ra bởi cạnh này bị xóa khỏi giải pháp, đồng thời cạnh đó được thêm vào đồ thị giải pháp.

Trong đồ án này, áp dụng loại sửa chữa cuối cùng “first edge” để giải quyết ràng buộc không bắt chéo. Loại sửa chữa này ảnh hưởng đến các giải thuật ở bước tăng đường cho các khách hàng C2(khi thứ tự các đỉnh khách hàng C2 khi thực hiện tăng đường thay đổi, thì giải pháp tìm thấy khác nhau).

CHƯƠNG 5: KẾT QUẢ THỰC NGHIỆM

1. Dữ liệu thực nghiệm

Để tiến hành các giải thuật đã trình bày, qua các tài liệu liên quan, chưa có bộ test chuẩn cho vấn đề này. Vì vậy đồ án sử dụng bộ test dữ liệu tự sinh (dựa vào các tham số random đã được đề cập trong tài liệu ^[9]). Bộ test này được sinh ra với bộ dữ liệu là các đồ thị đầy đủ, các đỉnh là các điểm trên mặt phẳng và trọng số của các cạnh nối giữa hai đỉnh được tính bằng khoảng cách Non-Euclidean giữa chúng. Đồ án tiến hành thực nghiệm trên các đồ thị có tổng số đỉnh là 70, 140, 250, 300, 400, 800, 1200, 1600, 2000 với các tỷ lệ khách hàng với các node không gian 1:1, 4:3, 2:3, 3:5. Ứng với mỗi kích thước đồ thị, ta sử dụng năm bộ test.

| STT | No.Test | Total Vertices | Instance | |
|-----|---------|----------------|----------|------|
| | | | n | m |
| 1 | 5 | 70 | 40 | 30 |
| 2 | 5 | 140 | 80 | 60 |
| 3 | 5 | 200 | 100 | 100 |
| 4 | 5 | 250 | 150 | 100 |
| 5 | 5 | 300 | 200 | 100 |
| 6 | 5 | 400 | 200 | 200 |
| 7 | 5 | 800 | 300 | 500 |
| 8 | 5 | 1200 | 450 | 750 |
| 9 | 5 | 1600 | 600 | 1000 |
| 10 | 5 | 2000 | 750 | 1250 |

Bảng 1: Tham số các bộ test

Trong đó:

- Number Test: Số lượng các test
- Number Vertice: Tổng số lượng đỉnh trong đồ thị ban đầu
- n: Số lượng các đỉnh khách hàng($C = C_1 \cup C_2$)
- m: Số lượng các đỉnh trong không gian (S)

2. Môi trường thử nghiệm

Chương trình thực hiện các giải thuật ở trên được cài đặt trên ngôn ngữ lập trình VC++, và chạy trên môi trường windows.

3. Tham số thực nghiệm

Đồ án tiến hành thực nghiệm hai giải thuật đề xuất và giải thuật được trình bày trong tài liệu [9] trên bộ dữ liệu test tự sinh.

Mỗi bài test được chạy 10 lần trên 5 bộ dữ liệu ứng với số đỉnh, ghi lại trọng số mạng tốt nhất, trọng số mạng trung bình, chi phí xây dựng cây Steiner. Các kết quả được so sánh với kết quả thực hiện theo giải thuật trong tài liệu [9] trên cùng bộ dữ liệu

4. Kết quả thử nghiệm

Để thuận tiện cho việc tổng hợp và xử lý, kết quả được ghi vào các file .txt. Định dạng file kết quả như sau:

outputFile.txt

Các ký hiệu dùng trong bảng kết quả:

- N: số đỉnh khách hàng (gồm cả loại 1 và loại 2)
- M: số đỉnh trong không gian
- N0.: số thứ tự bộ test
- Best: giá trị mạng nhỏ nhất tìm được
- Avg: giá trị trung bình các mạng tìm được
- Cost_Tree: chi phí xây dựng cây Steiner
- Time(s): thời gian chạy giải thuật.
- APSP: giải thuật xây dựng cây đường đi ngắn nhất giữa “terminal node” và các đỉnh khách hàng của Bucsics và Raidl (2007)
- APSP-I: giải thuật cải tiến của giải thuật APSP
- APSP-N: giải thuật đề xuất xây dựng cây Steiner sử dụng chạy nhiều lần giải thuật Dijkstra để tạo cây Steiner có trọng số nhỏ
- APSP-NI: giải thuật đề xuất cải tiến APSP-N
- MST: giải thuật đề xuất xây dựng cây Steiner dựa trên việc xây dựng cây khung nhỏ nhất từ các component.

Bảng 2: Kết quả chạy thuật toán APSP

| Test | No. | Instance | | Thuật toán APSP | | | | |
|------|-----|----------|-----|-----------------|---------|-----------|----------|-------|
| | | | | Cost | Time(s) | Cost_Tree | Avg_Aug | Best |
| | | n | m | | | | | |
| 1 | 1 | 40 | 30 | 20027 | 0.45 | 8506 | 11130.09 | 12974 |
| | 2 | 40 | 30 | 15181 | 0.45 | 9203 | 14762 | 15108 |
| | 3 | 40 | 30 | 17993 | 0.45 | 9565 | 17993 | 17993 |
| | 4 | 40 | 30 | 17624 | 0.48 | 9871 | 17624 | 17624 |
| | 5 | 40 | 30 | 14451 | 0.47 | 8673 | 14451 | 14451 |
| 2 | 1 | 80 | 60 | 19457 | 0.98 | 9667 | 19457 | 19457 |
| | 2 | 80 | 60 | 15900 | 0.95 | 8227 | 15900 | 15900 |
| | 3 | 80 | 60 | 12763 | 0.92 | 8026 | 12750 | 12739 |
| | 4 | 80 | 60 | 17530 | 0.95 | 11476 | 17530 | 17530 |
| | 5 | 80 | 60 | 15064 | 0.95 | 10412 | 15064 | 15064 |
| 3 | 1 | 100 | 100 | 12495 | 0.78 | 7865 | 12519.1 | 12495 |
| 4 | 1 | 100 | 150 | 12264 | 1.47 | 7752 | 12264 | 12264 |
| | 2 | 100 | 150 | 12591 | 1.47 | 6614 | 12591 | 12591 |
| | 3 | 100 | 150 | 14632 | 1.48 | 8135 | 14632 | 14632 |
| | 4 | 100 | 150 | 13102 | 1.50 | 7540 | 13102 | 13102 |
| | 5 | 100 | 150 | 14150 | 1.50 | 7019 | 14150 | 14150 |
| 5 | 1 | 200 | 100 | 14477 | 5.83 | 8838 | 14477 | 14477 |
| | 2 | 200 | 100 | 17065 | 5.84 | 9503 | 17065 | 17065 |
| | 3 | 200 | 100 | 25436 | 5.91 | 10646 | 25436 | 25436 |
| | 4 | 200 | 100 | 19958 | 5.87 | 10869 | 19958 | 19958 |
| | 5 | 200 | 100 | 20027 | 5.94 | 10136 | 20027 | 20027 |
| 6 | 1 | 200 | 200 | 16821 | 6.22 | 8646 | 16821 | 16821 |
| | 2 | 200 | 200 | 15322 | 6.12 | 10062 | 15322 | 15322 |
| | 3 | 200 | 200 | 14788 | 6.18 | 8753 | 14788 | 14788 |
| | 4 | 200 | 200 | 23536 | 6.19 | 9439 | 23536 | 23536 |
| | 5 | 200 | 200 | 15222 | 6.15 | 9231 | 15222 | 15222 |
| 7 | 1 | 300 | 500 | 11499 | 18.88 | 7574 | 11499 | 11499 |
| | 2 | 300 | 500 | 11739 | 19.00 | 7768 | 11739 | 11739 |
| | 3 | 300 | 500 | 13351 | 18.92 | 8527 | 13351 | 13351 |
| | 4 | 300 | 500 | 12810 | 18.91 | 7426 | 12810 | 12810 |

| | | | | | | | | |
|----|---|-----|------|-------|--------|------|-------|-------|
| | 5 | 300 | 500 | 12637 | 18.74 | 8068 | 12637 | 12637 |
| 8 | 1 | 450 | 750 | 11638 | 55.40 | 7586 | 11638 | 11638 |
| | 2 | 450 | 750 | 12766 | 56.07 | 7676 | 12766 | 12766 |
| | 3 | 450 | 750 | 12166 | 55.71 | 7506 | 12166 | 12166 |
| | 4 | 450 | 750 | 12167 | 55.88 | 7709 | 12167 | 12167 |
| | 5 | 450 | 750 | 12561 | 55.71 | 7979 | 12561 | 12522 |
| 9 | 1 | 600 | 1000 | 12627 | 124.11 | 8201 | 12627 | 12627 |
| | 2 | 600 | 1000 | 19547 | 124.07 | 7770 | 19547 | 19547 |
| | 3 | 600 | 1000 | 14081 | 125.05 | 8701 | 14081 | 14081 |
| | 4 | 600 | 1000 | 13236 | 123.99 | 8286 | 13236 | 13234 |
| | 5 | 600 | 1000 | 12874 | 124.89 | 7950 | 12874 | 12874 |
| 10 | 1 | 750 | 1250 | 13111 | 232.91 | 8131 | 13111 | 13111 |
| | 2 | 750 | 1250 | 12764 | 231.85 | 8216 | 12764 | 12764 |
| | 3 | 750 | 1250 | 12648 | 231.24 | 8106 | 12648 | 12641 |
| | 4 | 750 | 1250 | 13343 | 233.02 | 8281 | 13343 | 13343 |
| | 5 | 750 | 1250 | 12952 | 232.07 | 8193 | 12952 | 12949 |

Bảng 3: Kết quả chạy thuật toán APSP-I

| Test | No. | Instance | | Thuật toán APSP-I | | | | |
|------|-----|----------|-----|-------------------|---------|-----------|---------|-------|
| | | | | Cost | time(s) | Cost_tree | Avg | Best |
| | | n | m | | | | | |
| 1 | 1 | 40 | 30 | 15034 | 0.47 | 8506 | 11158 | 15034 |
| | 2 | 40 | 30 | 17218 | 0.47 | 9203 | 15329 | 17218 |
| | 3 | 40 | 30 | 20720 | 0.45 | 9565 | 20720 | 20720 |
| | 4 | 40 | 30 | 18880 | 0.47 | 9871 | 18880 | 18880 |
| | 5 | 40 | 30 | 17307 | 0.47 | 8673 | 17303 | 17307 |
| 2 | 1 | 80 | 60 | 20559 | 0.94 | 9667 | 20559 | 20559 |
| | 2 | 80 | 60 | 17716 | 0.92 | 8227 | 17716 | 17716 |
| | 3 | 80 | 60 | 15440 | 0.94 | 8026 | 15440 | 15440 |
| | 4 | 80 | 60 | 19690 | 1.04 | 11476 | 19690 | 19690 |
| | 5 | 80 | 60 | 18352 | 1.06 | 10412 | 18352 | 18352 |
| 3 | 1 | 100 | 100 | 12525 | 1.00 | 7865 | 12519.2 | 12467 |
| 4 | 1 | 100 | 150 | 12560 | 1.44 | 7752 | 12560 | 12560 |

| | | | | | | | | |
|----|---|-----|------|-------|--------|-------|-------|-------|
| | 2 | 100 | 150 | 13128 | 1.44 | 6614 | 13128 | 13128 |
| | 3 | 100 | 150 | 14910 | 1.58 | 8135 | 14900 | 14880 |
| | 4 | 100 | 150 | 14417 | 1.61 | 7540 | 14417 | 14417 |
| | 5 | 100 | 150 | 12933 | 1.58 | 7019 | 12933 | 12933 |
| 5 | 1 | 200 | 100 | 17668 | 5.74 | 8838 | 17668 | 17668 |
| | 2 | 200 | 100 | 18540 | 5.76 | 9503 | 18540 | 18540 |
| | 3 | 200 | 100 | 20668 | 5.79 | 10646 | 20668 | 20688 |
| | 4 | 200 | 100 | 20939 | 5.77 | 10869 | 20939 | 20939 |
| | 5 | 200 | 100 | 18193 | 5.74 | 10136 | 18193 | 18193 |
| 6 | 1 | 200 | 200 | 17185 | 6.08 | 8646 | 17185 | 17185 |
| | 2 | 200 | 200 | 16884 | 6.21 | 10062 | 16884 | 16884 |
| | 3 | 200 | 200 | 16467 | 6.07 | 8753 | 16467 | 16467 |
| | 4 | 200 | 200 | 15643 | 6.04 | 9439 | 15643 | 15643 |
| | 5 | 200 | 200 | 16599 | 6.05 | 9231 | 16599 | 16599 |
| 7 | 1 | 300 | 500 | 12997 | 18.85 | 7574 | 12997 | 12997 |
| | 2 | 300 | 500 | 12702 | 18.58 | 7768 | 12702 | 12702 |
| | 3 | 300 | 500 | 15030 | 18.75 | 8527 | 15030 | 15030 |
| | 4 | 300 | 500 | 13170 | 18.64 | 7426 | 13170 | 13170 |
| | 5 | 300 | 500 | 13514 | 18.28 | 8086 | 13514 | 13514 |
| 8 | 1 | 450 | 750 | 12790 | 54.88 | 7586 | 12790 | 12790 |
| | 2 | 450 | 750 | 14226 | 55.40 | 7676 | 14226 | 14226 |
| | 3 | 450 | 750 | 13229 | 55.11 | 7506 | 13229 | 13229 |
| | 4 | 450 | 750 | 13380 | 55.47 | 7709 | 13380 | 13380 |
| | 5 | 450 | 750 | 13462 | 55.40 | 7979 | 13462 | 13462 |
| 9 | 1 | 600 | 1000 | 13654 | 122.23 | 8201 | 13654 | 13654 |
| | 2 | 600 | 1000 | 13527 | 122.41 | 7770 | 13527 | 13527 |
| | 3 | 600 | 1000 | 14571 | 123.41 | 8701 | 14571 | 14571 |
| | 4 | 600 | 1000 | 14627 | 122.60 | 8286 | 14627 | 14614 |
| | 5 | 600 | 1000 | 14187 | 123.24 | 7950 | 14187 | 14187 |
| 10 | 1 | 750 | 1250 | 14028 | 229.76 | 8131 | 14028 | 14028 |
| | 2 | 750 | 1250 | 14415 | 228.99 | 8216 | 14415 | 14415 |
| | 3 | 750 | 1250 | 13810 | 228.82 | 8106 | 13810 | 13810 |
| | 4 | 750 | 1250 | 14232 | 229.13 | 8281 | 14232 | 14229 |
| | 5 | 750 | 1250 | 14051 | 228.88 | 8193 | 14051 | 14051 |

Bảng 4: Kết quả chạy thuật toán APSP-N

| Test | No. | Instance | | Thuật toán APSP-N | | | | |
|------|-----|----------|-----|-------------------|---------|-----------|--------|--------|
| | | | | Cost | time(s) | Cost_tree | Avg | Best |
| | | n | m | | | | | |
| 1 | 1 | 40 | 30 | 18953 | 0.21 | 5711 | 14337 | 18860 |
| | 2 | 40 | 30 | 49739 | 0.25 | 6583 | 15750 | 49739 |
| | 3 | 40 | 30 | 21859 | 0.23 | 7056 | 21589 | 21589 |
| | 4 | 40 | 30 | 19276 | 0.22 | 8271 | 19201 | 19196 |
| | 5 | 40 | 30 | 27464 | 0.25 | 6466 | 27464 | 27464 |
| 2 | 1 | 80 | 60 | 41433 | 0.31 | 7138 | 41433 | 41433 |
| | 2 | 80 | 60 | 20526 | 0.32 | 6439 | 20526 | 20526 |
| | 3 | 80 | 60 | 50263 | 0.31 | 5822 | 50263 | 50263 |
| | 4 | 80 | 60 | 23504 | 0.31 | 8609 | 23500 | 23485 |
| | 5 | 80 | 60 | 28675 | 0.31 | 7151 | 28675 | 28675 |
| 3 | 1 | 100 | 100 | 14428 | 0.98 | 5271 | 14417 | 14336 |
| 4 | 1 | 100 | 150 | 40923 | 0.48 | 5011 | 40923 | 40923 |
| | 2 | 100 | 150 | 37833 | 0.53 | 5535 | 37833 | 37833 |
| | 3 | 100 | 150 | 19964 | 0.50 | 6268 | 19964 | 19964 |
| | 4 | 100 | 150 | 135527 | 0.70 | 5468 | 135527 | 135527 |
| | 5 | 100 | 150 | 25888 | 0.44 | 5620 | 25870 | 25835 |
| 5 | 1 | 200 | 100 | 27323 | 0.76 | 7089 | 27323 | 27323 |
| | 2 | 200 | 100 | 16709 | 0.72 | 7109 | 16709 | 16709 |
| | 3 | 200 | 100 | 22730 | 0.75 | 8048 | 22730 | 22730 |
| | 4 | 200 | 100 | 122928 | 1.05 | 7768 | 122928 | 122928 |
| | 5 | 200 | 100 | 52499 | 0.87 | 7671 | 52499 | 52499 |
| 6 | 1 | 200 | 200 | 88894 | 1.47 | 6638 | 88894 | 88894 |
| | 2 | 200 | 200 | 109802 | 1.30 | 6691 | 109802 | 109802 |
| | 3 | 200 | 200 | 58648 | 1.20 | 6333 | 58648 | 58648 |
| | 4 | 200 | 200 | 34868 | 1.12 | 6228 | 34868 | 34868 |
| | 5 | 200 | 200 | 70956 | 1.19 | 6728 | 70956 | 70956 |
| 7 | 1 | 300 | 500 | 42567 | 5.12 | 5574 | 42567 | 42567 |
| | 2 | 300 | 500 | 145641 | 4.76 | 5416 | 145641 | 145641 |
| | 3 | 300 | 500 | 178370 | 5.57 | 5801 | 178370 | 178370 |
| | 4 | 300 | 500 | 26826 | 5.23 | 5411 | 26826 | 26826 |
| | 5 | 300 | 500 | 143640 | 5.49 | 5608 | 143640 | 143640 |

| | | | | | | | | |
|----|---|-----|------|--------|--------|------|--------|--------|
| 8 | 1 | 450 | 750 | 139670 | 18.97 | 5305 | 139670 | 139670 |
| | 2 | 450 | 750 | 69145 | 19.17 | 5865 | 69145 | 69145 |
| | 3 | 450 | 750 | 55549 | 18.41 | 5295 | 55549 | 55549 |
| | 4 | 450 | 750 | 73143 | 19.39 | 5524 | 73143 | 73143 |
| | 5 | 450 | 750 | 141759 | 19.56 | 5607 | 141759 | 141759 |
| 9 | 1 | 600 | 1000 | 312580 | 49.00 | 6028 | 312580 | 312580 |
| | 2 | 600 | 1000 | 75738 | 48.47 | 5774 | 75738 | 75738 |
| | 3 | 600 | 1000 | 984301 | 51.65 | 6002 | 984301 | 984301 |
| | 4 | 600 | 1000 | 14443 | 48.31 | 6269 | 14443 | 14443 |
| | 5 | 600 | 1000 | 34955 | 47.63 | 5846 | 34955 | 34955 |
| 10 | 1 | 750 | 1250 | 871895 | 101.51 | 5641 | 871895 | 871895 |
| | 2 | 750 | 1250 | 144365 | 97.38 | 5961 | 144365 | 144365 |
| | 3 | 750 | 1250 | 290246 | 100.06 | 5950 | 290246 | 290246 |
| | 4 | 750 | 1250 | 365399 | 100.82 | 6031 | 365399 | 365399 |
| | 5 | 750 | 1250 | 158373 | 98.30 | 5841 | 158373 | 158373 |

Bảng 5: Kết quả chạy thuật toán APSP-NI

| Test | No. | Instance | | Thuật toán APSP-NI | | | | |
|------|-----|----------|-----|--------------------|---------|-----------|----------|-------|
| | | | | Cost | time(s) | Cost_tree | Avg | Best |
| | | n | m | | | | | |
| 1 | 1 | 40 | 30 | 13969 | 0.22 | 5711 | 13957.09 | 13876 |
| | 2 | 40 | 30 | 14480 | 0.23 | 6583 | 14480 | 14480 |
| | 3 | 40 | 30 | 18911 | 0.23 | 7056 | 18911 | 18911 |
| | 4 | 40 | 30 | 19399 | 0.22 | 8271 | 19399 | 19399 |
| | 5 | 40 | 30 | 16689 | 0.22 | 6466 | 16580 | 16428 |
| 2 | 1 | 80 | 60 | 19637 | 0.75 | 7138 | 19632.4 | 19591 |
| | 2 | 80 | 60 | 16170 | 0.27 | 6439 | 16170 | 16170 |
| | 3 | 80 | 60 | 14671 | 0.25 | 5822 | 14671 | 14671 |
| | 4 | 80 | 60 | 18459 | 0.30 | 8609 | 18459 | 18459 |
| | 5 | 80 | 60 | 15313 | 0.28 | 7151 | 15313 | 15313 |
| 3 | 1 | 100 | 100 | 13207 | 0.71 | 5271 | 13195.3 | 13090 |
| 4 | 1 | 100 | 150 | 11126 | 0.44 | 5011 | 11126 | 11126 |
| | 2 | 100 | 150 | 13173 | 0.39 | 5535 | 13173 | 13173 |

| | | | | | | | | |
|----|---|-----|------|-------|-------|------|-------|-------|
| | 3 | 100 | 150 | 14396 | 0.50 | 6268 | 14369 | 14369 |
| | 4 | 100 | 150 | 12833 | 0.44 | 5468 | 12833 | 12833 |
| | 5 | 100 | 150 | 11650 | 0.41 | 5620 | 11650 | 11650 |
| 5 | 1 | 200 | 100 | 16706 | 0.61 | 7089 | 16706 | 16706 |
| | 2 | 200 | 100 | 17432 | 0.64 | 7109 | 17432 | 17432 |
| | 3 | 200 | 100 | 18779 | 0.61 | 8048 | 18779 | 18779 |
| | 4 | 200 | 100 | 18470 | 0.72 | 7768 | 18470 | 18470 |
| | 5 | 200 | 100 | 17542 | 0.59 | 7671 | 17542 | 17542 |
| 6 | 1 | 200 | 200 | 15637 | 0.97 | 6638 | 15637 | 15637 |
| | 2 | 200 | 200 | 14920 | 0.94 | 6691 | 14920 | 14920 |
| | 3 | 200 | 200 | 15507 | 0.94 | 6333 | 15507 | 15507 |
| | 4 | 200 | 200 | 14170 | 0.94 | 6228 | 14170 | 14170 |
| | 5 | 200 | 200 | 15374 | 0.92 | 6728 | 15374 | 15374 |
| 7 | 1 | 300 | 500 | 11517 | 4.59 | 5574 | 11517 | 11517 |
| | 2 | 300 | 500 | 11038 | 4.57 | 5416 | 11038 | 11038 |
| | 3 | 300 | 500 | 12853 | 4.57 | 5801 | 12853 | 12853 |
| | 4 | 300 | 500 | 11358 | 4.48 | 5411 | 11358 | 11358 |
| | 5 | 300 | 500 | 12228 | 4.57 | 5608 | 12228 | 12228 |
| 8 | 1 | 450 | 750 | 11401 | 17.39 | 5305 | 11401 | 11401 |
| | 2 | 450 | 750 | 13196 | 17.89 | 5865 | 13196 | 13196 |
| | 3 | 450 | 750 | 12010 | 17.66 | 5295 | 12010 | 12005 |
| | 4 | 450 | 750 | 11954 | 17.82 | 5524 | 11954 | 11954 |
| | 5 | 450 | 750 | 11830 | 18.25 | 5607 | 11830 | 11830 |
| 9 | 1 | 600 | 1000 | 11936 | 45.80 | 6028 | 11936 | 11936 |
| | 2 | 600 | 1000 | 11922 | 46.22 | 5774 | 11922 | 11922 |
| | 3 | 600 | 1000 | 12812 | 46.52 | 6002 | 12812 | 12812 |
| | 4 | 600 | 1000 | 13483 | 46.21 | 6269 | 13483 | 13483 |
| | 5 | 600 | 1000 | 12185 | 45.91 | 5846 | 12185 | 12164 |
| 10 | 1 | 750 | 1250 | 12753 | 96.74 | 5641 | 12753 | 12753 |
| | 2 | 750 | 1250 | 12304 | 95.08 | 5961 | 12304 | 12304 |
| | 3 | 750 | 1250 | 12145 | 94.05 | 5950 | 12145 | 12139 |
| | 4 | 750 | 1250 | 12711 | 94.86 | 6031 | 12711 | 12711 |
| | 5 | 750 | 1250 | 12370 | 95.11 | 5841 | 12370 | 12370 |

Bảng 6: Kết quả chạy thuật toán MST

| Test | No. | Instance | | Thuật toán MST | | | | |
|------|-----|----------|------|----------------|---------|-----------|----------|---------|
| | | | | Cost | time(s) | Cost_tree | MST-Mean | MST-Min |
| | | n | m | | | | | |
| 1 | 1 | 40 | 30 | 10304 | 0.298 | 5908 | 10304 | 10304 |
| | 2 | 40 | 30 | 11916 | 0.276 | 6817 | 11916 | 11916 |
| | 3 | 40 | 30 | 13507 | 0.363 | 7420 | 13507 | 13507 |
| | 4 | 40 | 30 | 14934 | 0.347 | 8706 | 14934 | 14934 |
| | 5 | 40 | 30 | 10957 | 0.207 | 6833 | 10957 | 10957 |
| 2 | 1 | 80 | 60 | 14199 | 0.453 | 7537 | 14199 | 14199 |
| | 2 | 80 | 60 | 12004 | 0.729 | 6609 | 12004 | 12004 |
| | 3 | 80 | 60 | 10435 | 0.619 | 6049 | 10435 | 10435 |
| | 4 | 80 | 60 | 14842 | 0.709 | 7179 | 14842 | 14842 |
| | 5 | 80 | 60 | 12478 | 0.472 | 7626 | 12478 | 13478 |
| 3 | 1 | 100 | 100 | 9895 | 0.943 | 5762 | 9896 | 9895 |
| 4 | 1 | 100 | 150 | 8973 | 0.234 | 5291 | 8973 | 8973 |
| | 2 | 100 | 150 | 11115 | 0.296 | 5941 | 11115 | 11115 |
| | 3 | 100 | 150 | 11735 | 0.218 | 6824 | 11735 | 11735 |
| | 4 | 100 | 150 | 9767 | 0.28 | 5932 | 9767 | 9767 |
| | 5 | 100 | 150 | 10664 | 0.219 | 6104 | 10664 | 10664 |
| 5 | 1 | 200 | 100 | 12575 | 0.784 | 7375 | 12575 | 12575 |
| | 2 | 200 | 100 | 12150 | 0.468 | 7291 | 12150 | 12150 |
| | 3 | 200 | 100 | 13517 | 0.359 | 8346 | 13517 | 13517 |
| | 4 | 200 | 100 | 13463 | 0.405 | 8058 | 13463 | 13463 |
| | 5 | 200 | 100 | 12290 | 0.405 | 7871 | 12290 | 12290 |
| 6 | 1 | 200 | 200 | 12266 | 1.03 | 7030 | 12266 | 12266 |
| | 2 | 200 | 200 | 11341 | 1.076 | 7098 | 11341 | 11341 |
| | 3 | 200 | 200 | 11031 | 0.764 | 6614 | 11031 | 11031 |
| | 4 | 200 | 200 | 11118 | 0.889 | 6417 | 11118 | 11118 |
| | 5 | 200 | 200 | 12251 | 0.904 | 7172 | 12251 | 12251 |
| 7 | 1 | 300 | 500 | 9821 | 1.368 | 5901 | 9821 | 9821 |
| | 2 | 300 | 500 | 9520 | 1.344 | 5819 | 9520 | 9520 |
| | 3 | 300 | 500 | 10487 | 1.507 | 6336 | 10487 | 10487 |
| | 4 | 300 | 500 | 9388 | 1.316 | 5830 | 9388 | 9388 |
| | 5 | 300 | 500 | 9927 | 1.429 | 6153 | 9927 | 9927 |
| 8 | 1 | 450 | 750 | 9501 | 4.106 | 5651 | 9501 | 9501 |
| | 2 | 450 | 750 | 10640 | 4.749 | 6442 | 10640 | 10640 |
| | 3 | 450 | 750 | 9846 | 4.5 | 5785 | 9846 | 9846 |
| | 4 | 450 | 750 | 9817 | 4.306 | 5814 | 9817 | 9817 |
| | 5 | 450 | 750 | 9735 | 4.584 | 6067 | 9735 | 9735 |
| 9 | 1 | 600 | 1000 | 14192 | 44.023 | 10975 | 14192 | 14192 |
| | 2 | 600 | 1000 | 13690 | 46.41 | 10711 | 13690 | 13690 |
| | 3 | 600 | 1000 | 14480 | 48.142 | 10981 | 14480 | 14480 |

| | | | | | | | | |
|----|---|-----|------|-------|--------|-------|-------|-------|
| | 4 | 600 | 1000 | 14869 | 45.24 | 11111 | 14869 | 14869 |
| | 5 | 600 | 1000 | 14162 | 45.911 | 10776 | 14162 | 14162 |
| 10 | 1 | 750 | 1250 | 14037 | 94.01 | 10599 | 14037 | 14037 |
| | 2 | 750 | 1250 | 14360 | 81.869 | 11088 | 14360 | 14360 |
| | 3 | 750 | 1250 | 14340 | 80.667 | 10973 | 14340 | 14340 |
| | 4 | 750 | 1250 | 14228 | 82.665 | 10701 | 14228 | 14228 |
| | 5 | 750 | 1250 | 14033 | 81.556 | 10803 | 14033 | 14033 |

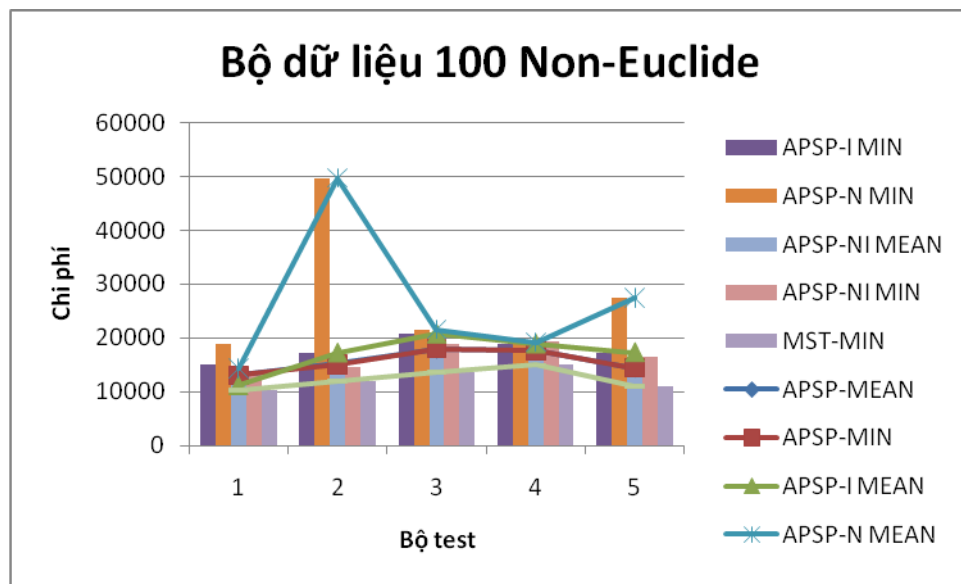
5. So sánh kết quả

5.1. Đồ thị so sánh

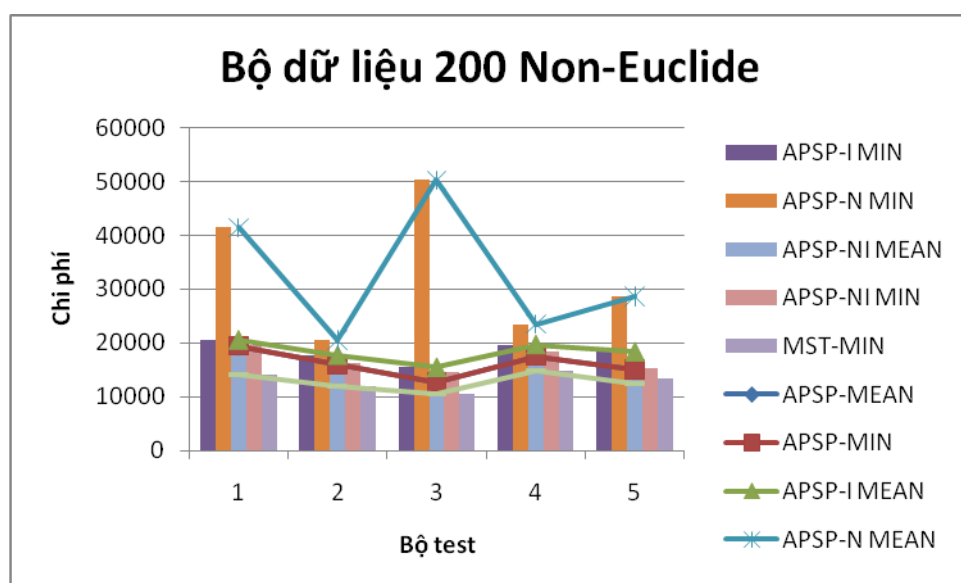
Để đánh giá hiệu quả của thuật toán, kết quả thu được sẽ so sánh với giá trị thu được từ thuật toán đã có của bài toán.

Dưới đây là kết quả được biểu diễn dưới dạng đồ thị của thuật toán đề xuất APSP-I, APSP-N, APSP-NI, MST so với thuật toán có sẵn APSP

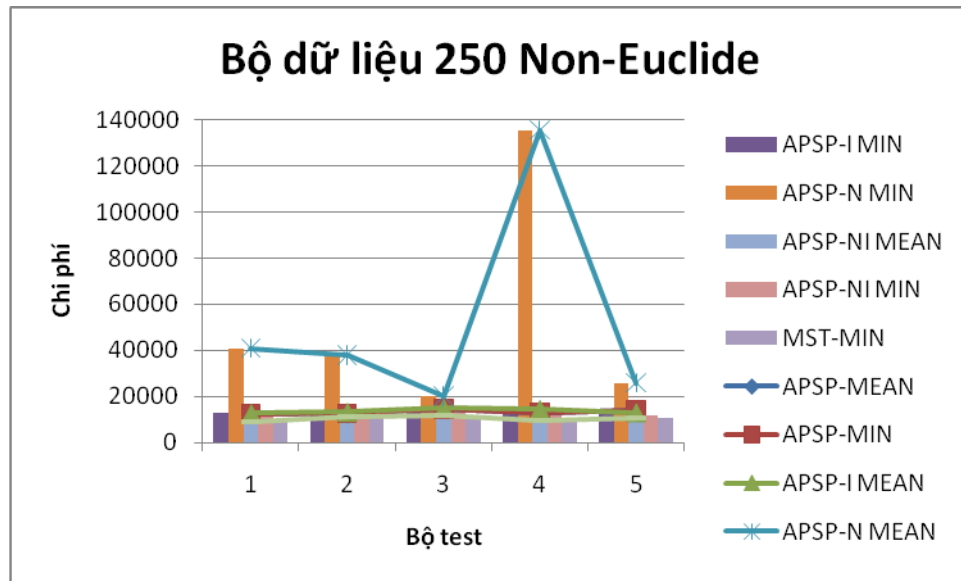
5.1.1. Giá trị tối ưu mạng trong cả năm giải thuật



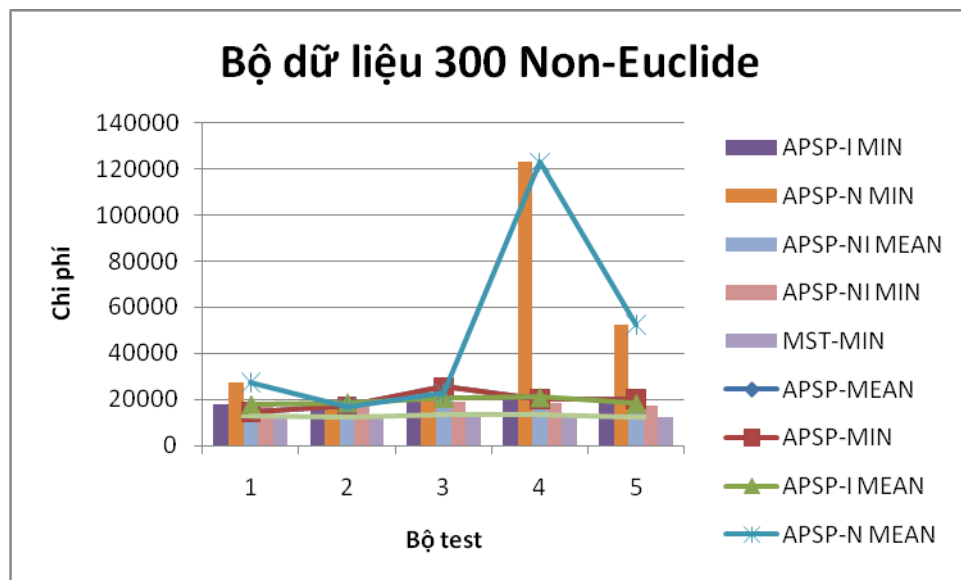
Hình 15: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=40, m=30$)



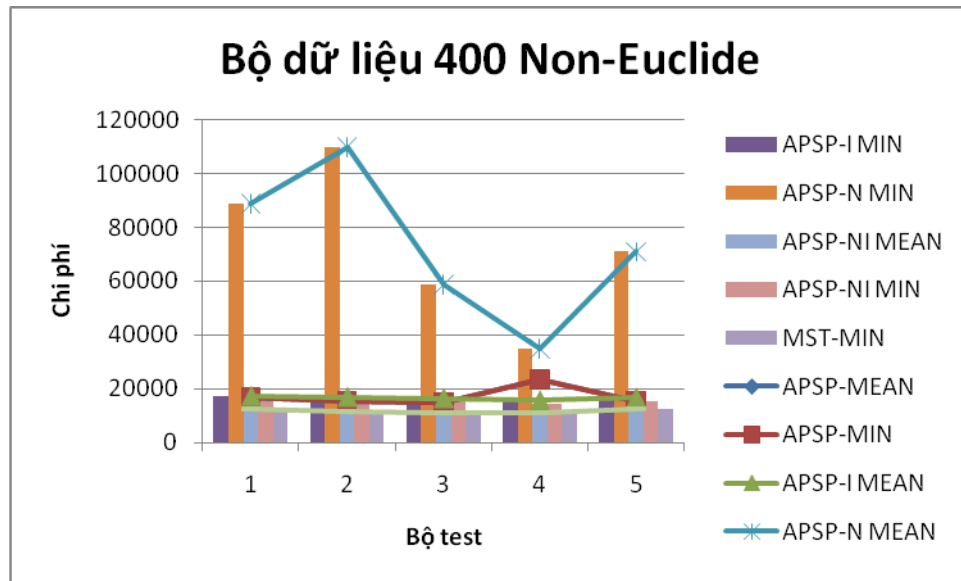
Hình 16: giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=80, m=60$)



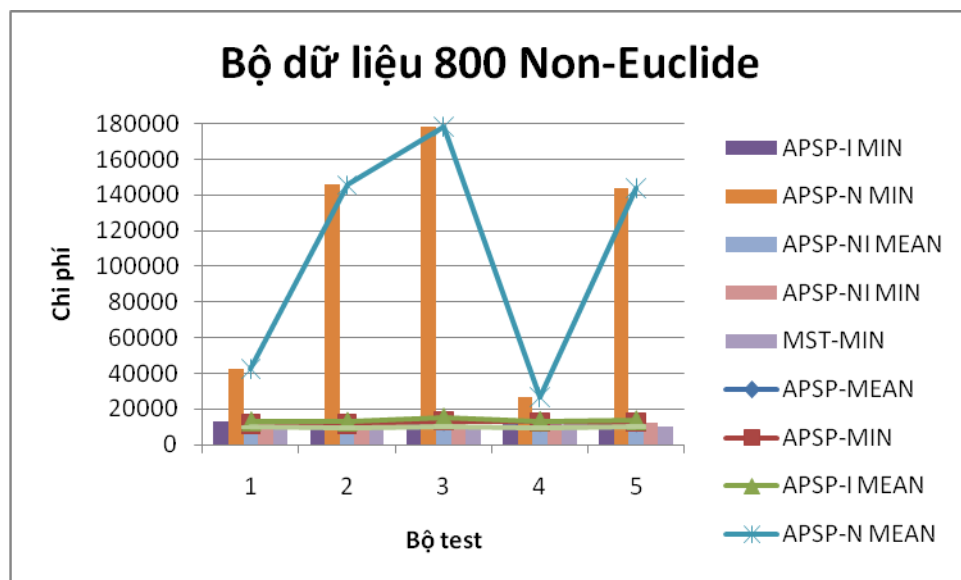
Hình 17: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=100, m=150$)



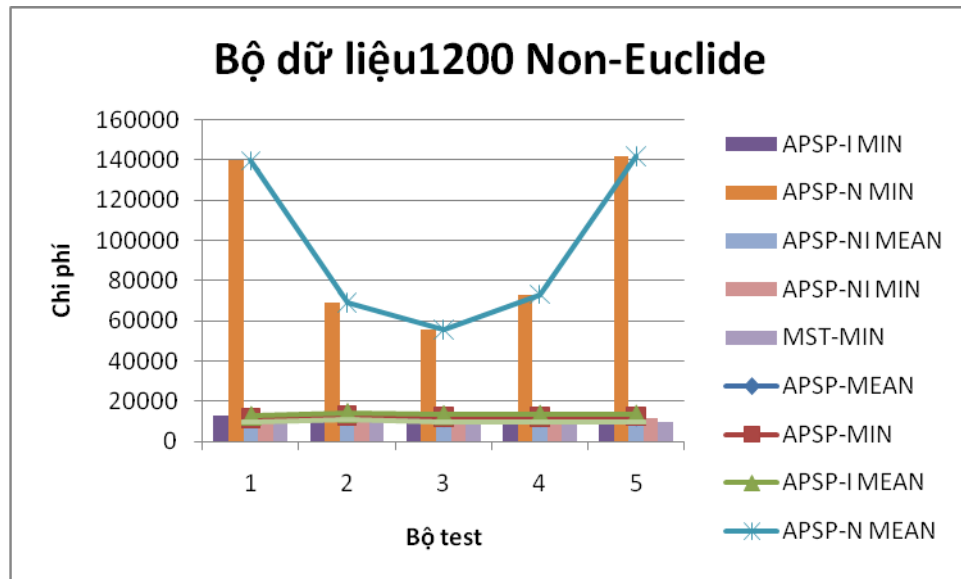
Hình 18: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=200, m=100$)



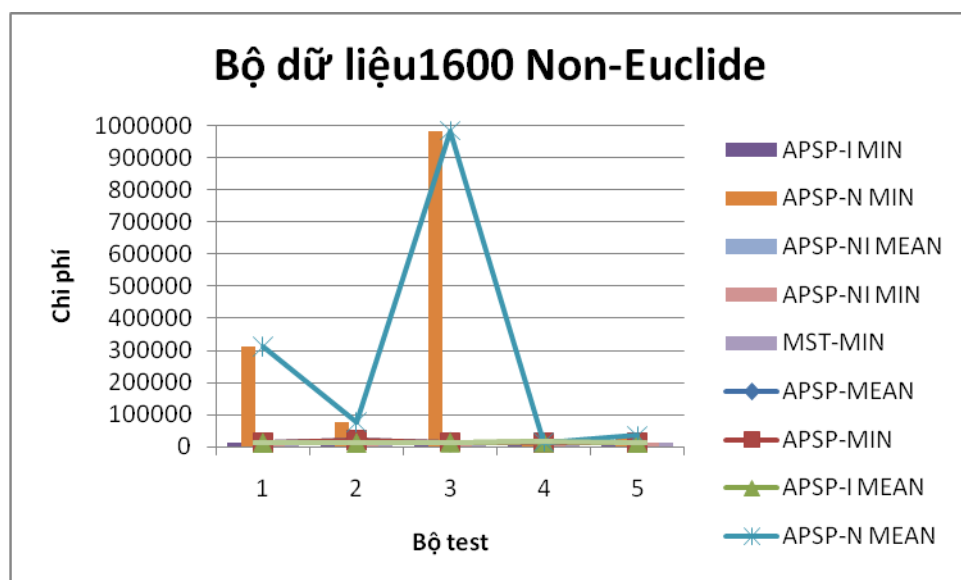
Hình 19: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=200, m=200$)



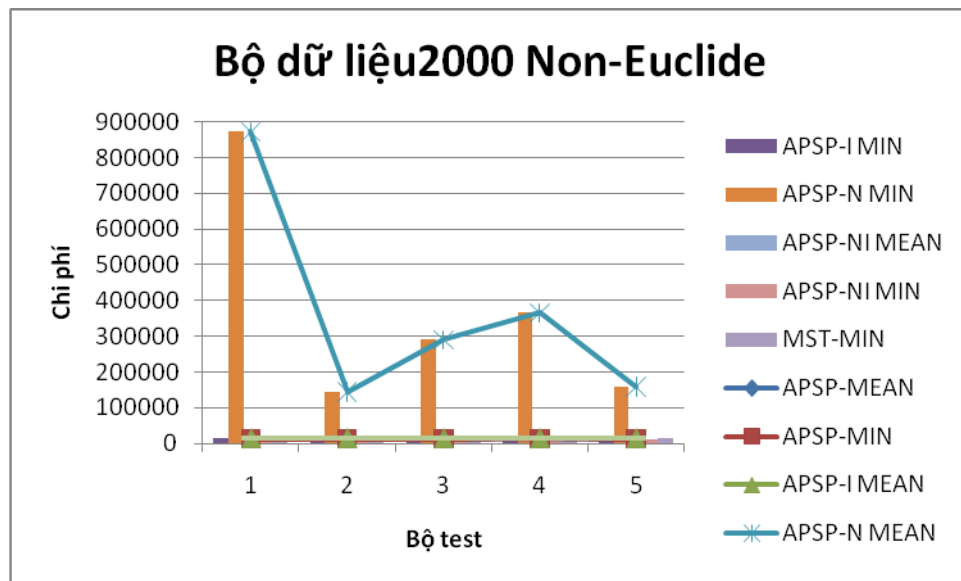
Hình 20: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=300, m=500$)



Hình 21: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=450, m=750$)

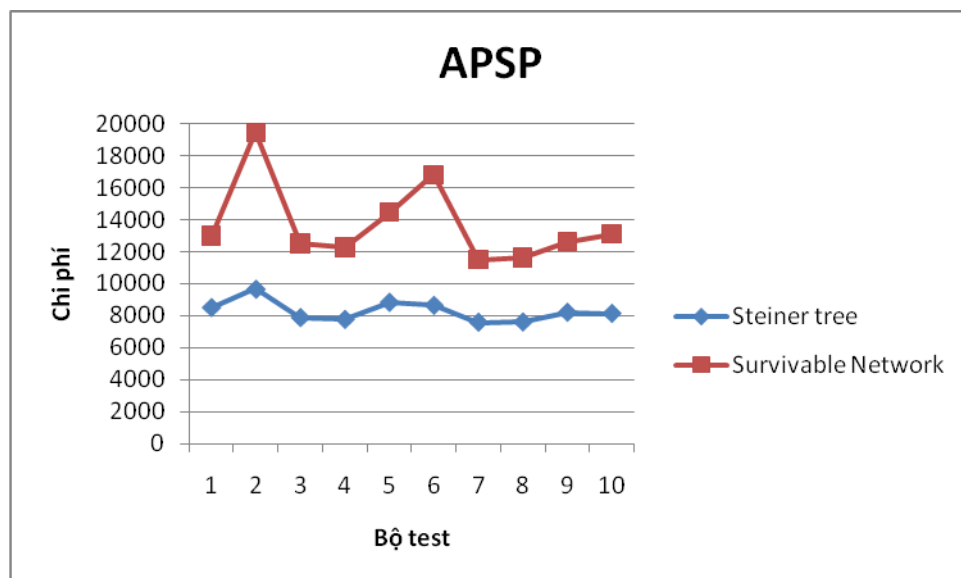


Hình 22: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=600, m=1000$)

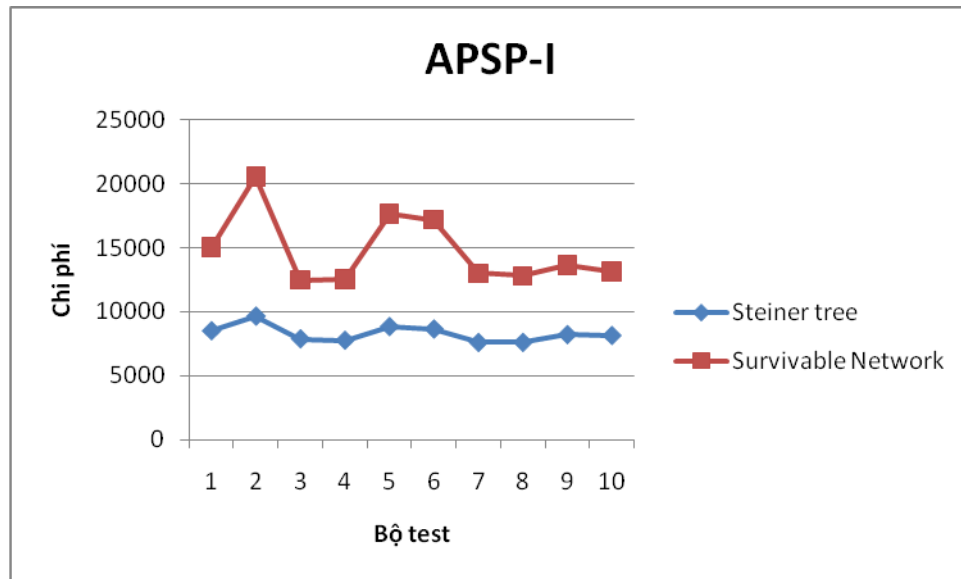


Hình 23: Giá trị tối ưu mạng của APSP, APSP-I, APSP-N, APSP-NI, MST ($n=750, m=1250$)

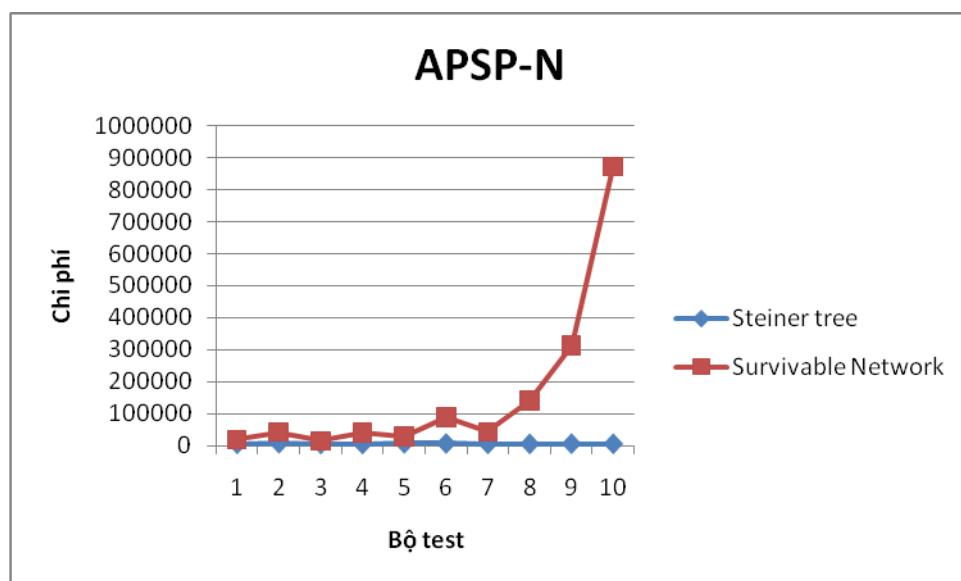
5.1.2. So sánh giữa chi phí xây dựng cây Steiner và chi phí toàn bộ mạng cần thiết trong mỗi giải thuật



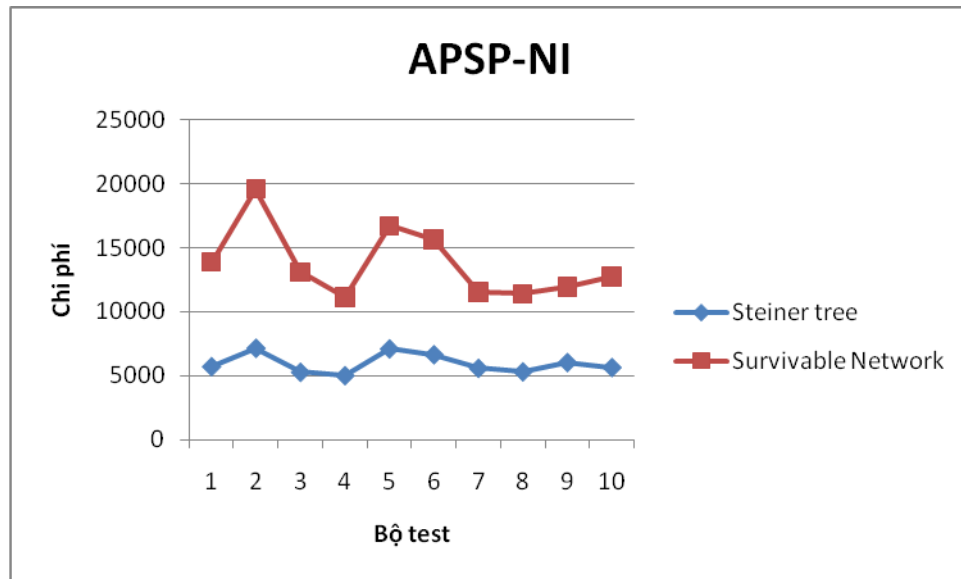
Hình 24: Mối tương quan giữa chi phí xây dựng cây Steiner và chi phí mạng trong giải thuật APSP



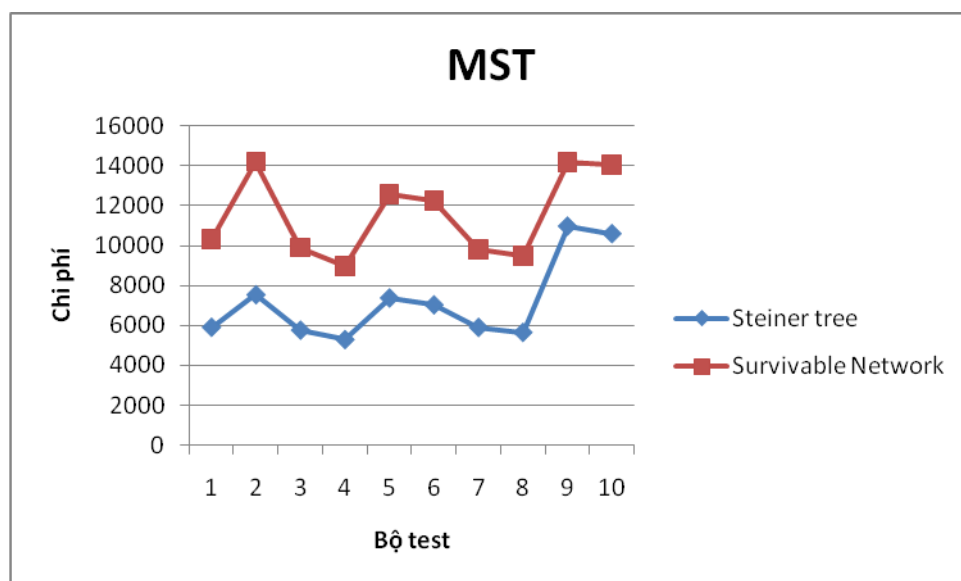
Hình 25: Mối tương quan giữa chi phí xây dựng cây Steiner và chi phí mạng trong giải thuật APSP-I



Hình 26: Mối tương quan giữa chi phí xây dựng cây Steiner và chi phí mạng trong giải thuật APSP-N

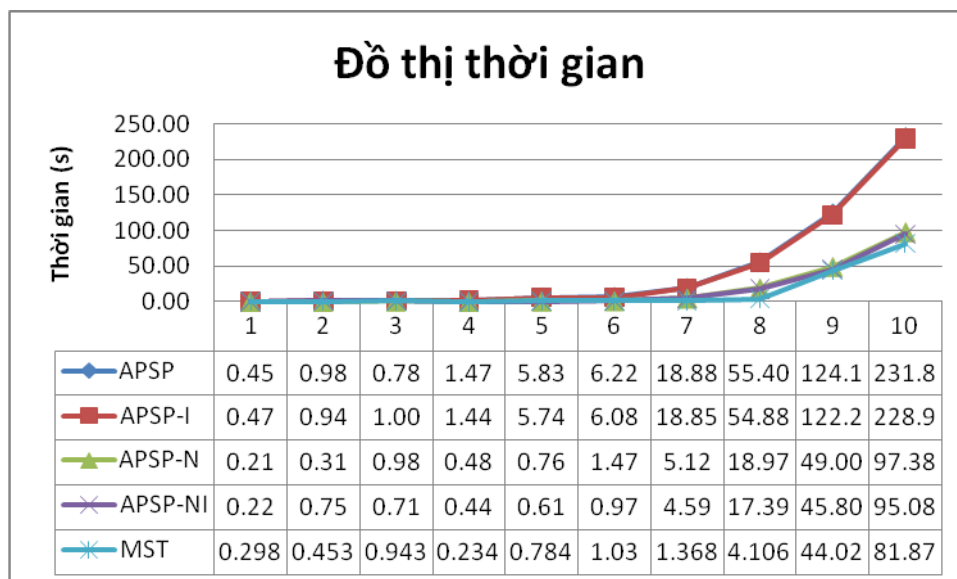


Hình 27: Mối tương quan giữa chi phí xây dựng cây Steiner và chi phí mạng trong giải thuật APSP-NI



Hình 28: Mối tương quan giữa chi phí xây dựng cây Steiner và chi phí mạng trong giải thuật MST

5.1.3. So sánh thời gian chạy của các thuật toán theo sự biến thiên của số lượng đỉnh



Hình 29: Mỗi tương quan thời gian chạy ứng với sự biến thiên số lượng đỉnh của đồ thị

5.2. Nhận xét

5.2.1. Về mặt kết quả

Trong 5 giải thuật đã được cài đặt, giải thuật APSP được cài đặt lại theo thuật toán được đưa ra trong ^[9], các giải thuật APSP-I, APSP-N, APSP-NI, và MST là các giải thuật tự đề xuất.

Quan sát các đồ thị ở mục 5.1, có thể rút ra một số nhận xét như sau:

- Đối với các bộ dữ liệu test tự sinh ngẫu nhiên ở trên, so với giải thuật cài đặt lại giải thuật được đưa ra ở bài báo ^[9] nói riêng và trong cả 5 giải thuật nói chung, thì giải thuật MST được đề xuất hầu hết cho kết quả tốt hơn hẳn (Chỉ có 2/30=6% cho kết quả kém hơn). Hơn nữa, giải thuật này có tính ổn định cao (Gần như các lần chạy đều cho cùng một kết quả), điều này giúp ta có thể tiên đoán, kết quả của giải thuật thu được đã rất gần với kết quả tối ưu.
- Hai giải thuật đề xuất APSP-N và APSP-NI hiện có kết quả không tốt bằng MST(Đặc biệt giải thuật APSP-N có kết quả rất tồi trong trường hợp số đỉnh đồ thị lớn). Tuy nhiên, có thể nhận thấy rằng khi phân tích đặc tính của cây Steiner được tạo ra trước khi tăng đường, và đề xuất cải thiện tính chất cho cây Steiner thì chúng ta có thể thu được kết quả tối ưu hơn. Ở hai giải thuật này, đã tìm ra được cây Steiner có chi phí nhỏ hơn nhiều so với các giải thuật còn lại (APSP, APSP-I, và ngay cả MST). Vì thế, có thể đưa ra nhận xét

rằng, giải thuật để xây dựng cây Steiner (hay nói cách khác tính chất cây Steiner thu được) ảnh hưởng rất lớn đến toàn bộ chi phí thiết kế mạng chịu lỗi. Đây sẽ là một hướng khả quan cho các giải thuật cải tiến nghiên cứu về sau.

Như vậy, để đưa ra chi phí tốt nhất cho bài toán, chúng ta cần xem xét kỹ các đặc điểm cây Steiner có lợi thế cho bước tăng luồng và đề xuất giải thuật phù hợp.

5.2.2. Về mặt thời gian chạy

Quan sát đồ thị thời gian chạy của cả năm thuật toán [Hình 29], ta có nhận xét:

- Khi số lượng đỉnh của đồ thị ban đầu nhỏ hơn 500, thì hầu như cả 5 giải thuật chênh lệch nhau không lớn lắm 0.1 đến 0.5 giây.
- Khi số lượng đỉnh tăng lên đến 800, 1200, 1600, 2000 thì có sự phân rõ đặc biệt giữa năm giải thuật: Thời gian chạy của hai giải thuật APSP và APSP-I lớn hơn hẳn so với ba giải thuật còn lại (APSP-N, APSP-NI, MST)
- Với số lượng đỉnh được sinh ra trong bộ test, thì giải thuật MST có thời gian chạy nhỏ nhất trong cả năm giải thuật.

KẾT LUẬN

1. Các kết quả đạt được

Về mặt lý thuyết, đồ án đã trình bày được các nội dung sau:

- Các khái niệm cơ bản về đồ thị, phân lớp bài toán, qua đó phát biểu bài toán thiết kế mạng chịu lỗi (SNDP) bằng mô hình đồ thị cùng các ứng dụng
- Tổng quan các giải thuật tìm đường đi ngắn nhất giữa các cặp đỉnh trên đồ thị
- Tổng quan giải thuật heuristic, meta-heuristic
- Đề xuất giải thuật heuristic, meta-heuristic để giải bài toán SNDP

Về mặt thực nghiệm, đồ án đã thu được một số kết quả:

- Đã cài đặt thành công 5 giải thuật (trong đó 1 giải thuật APSP được cài đặt lại theo giải thuật được đưa ra trong bài báo [[9]], bốn giải thuật APSP-I, APSP-N, APSP-NI, MST là các giải thuật tự đề xuất).
- So với giải thuật APSP đã được đưa ra bởi Thomas Bucsics [[9]], giải thuật MST đề xuất cho kết quả tốt hơn và tốt nhất trong cả năm giải thuật.
- Độ ổn định của giải thuật MST đề xuất tốt hơn giải thuật APSP đã có.

2. Hạn chế

Như vậy, kết quả thực nghiệm là khá phù hợp với những gì trình bày trong lý thuyết. Tuy vậy, do còn nhiều hạn chế về kiến thức, kinh nghiệm của bản thân, cũng như thời gian thực hiện, đồ án còn các hạn chế:

- Chưa cài đặt thử nghiệm các giải thuật meta-heuristic để so sánh với giải thuật heuristic đã có và giải thuật đề xuất.
- Trong bốn giải thuật đề xuất, chỉ có giải thuật MST đưa ra kết quả tốt hơn, còn ba giải thuật còn lại APSP-I, APSP-N, APSP-NI chưa thật sự mang lại kết quả tốt. Nguyên nhân là chưa tìm hiểu được đặc điểm cây Steiner phù hợp cho bài toán SNDP để tối ưu chi phí lúc tăng đường cho mạng chịu lỗi từ cây Steiner đã được xây dựng.

3. Hướng phát triển của đồ án

Trong quá trình thực hiện đề tài, chúng tôi nhận thấy vẫn còn nhiều vấn đề khác liên quan đến lĩnh vực Thiết kế mạng chịu lỗi này cần được quan tâm. Từ những hạn chế đã gặp phải lúc thực hiện đề tài, chúng tôi đề xuất hướng phát triển của đề tài như sau:

- Nghiên cứu vấn đề thiết kế mạng chịu lỗi với nhiều mô hình khác nhau.
- Đề xuất và cài đặt thử nghiệm các giải thuật meta-heuristic để so sánh kết quả với giải thuật heuristic đã có và các giải thuật đề xuất trong đề tài này
- Nghiên cứu tìm hiểu đặc điểm cây Steiner phù hợp cho bài toán SNDP. Từ đó, đề xuất phương pháp để cải tiến các giải thuật APSP-I, APSP-N, APSP-NI đã được thực hiện trong đề tài này.

PHỤ LỤC

1. Giới thiệu chung

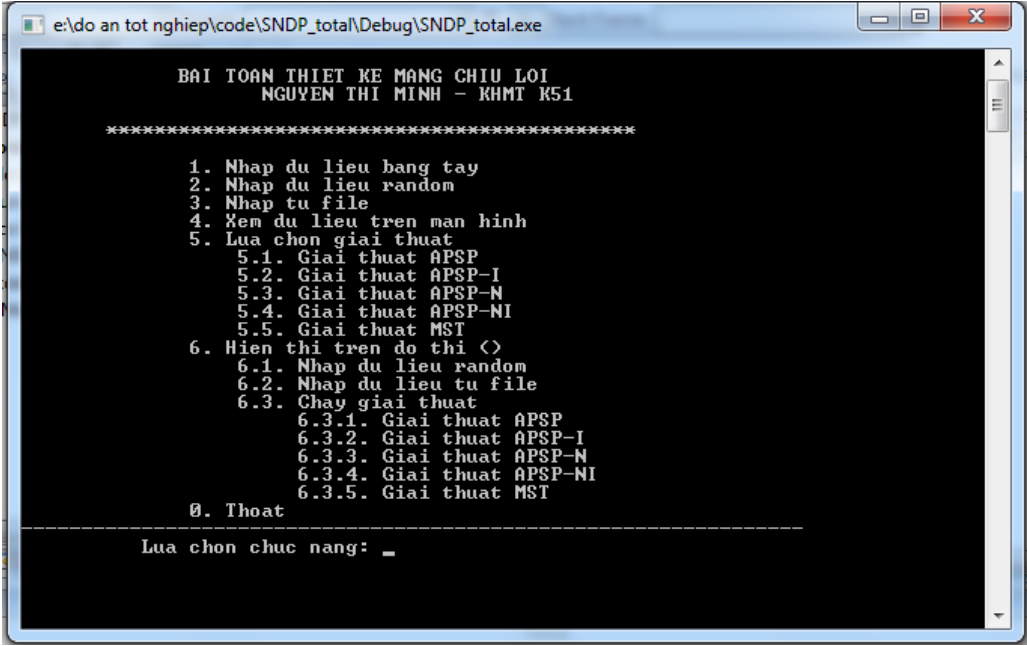
Phụ lục này là hướng dẫn sử dụng chương trình demo:

- Chạy các giải thuật với màn hình console.

2. Hướng dẫn sử dụng chương trình demo với màn hình console

2.1. Chạy File SNDP.exe

2.2. Giao diện menu ban đầu:



```
e:\do an tot nghiep\code\SNDP_total\Debug\SNDP_total.exe

BAI TOAN THIET KE MANG CHIU LOI
NGUYEN THI MINH - KHMT K51

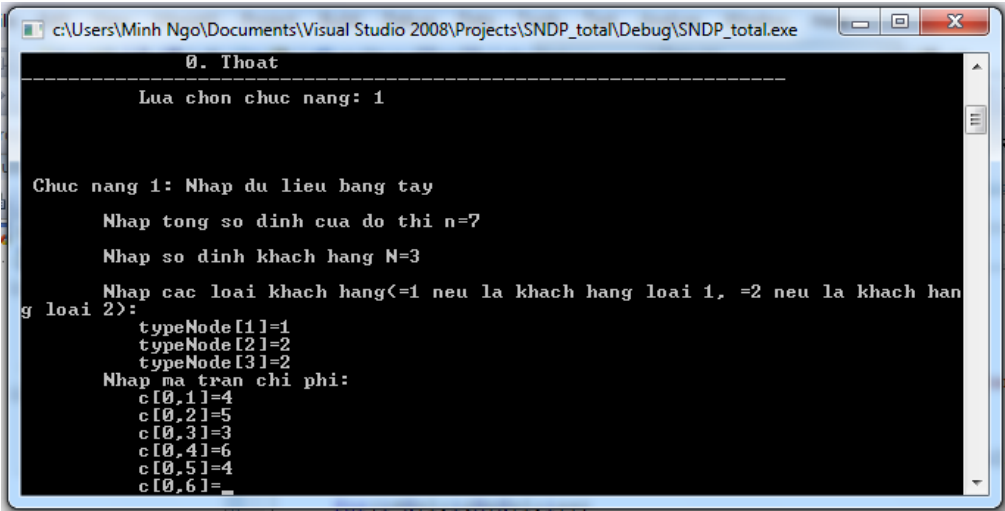
*****

1. Nhap du lieu bang tay
2. Nhap du lieu random
3. Nhap tu file
4. Xem du lieu tren man hinh
5. Lua chon giai thuat
  5.1. Giai thuat APSP
  5.2. Giai thuat APSP-I
  5.3. Giai thuat APSP-N
  5.4. Giai thuat APSP-NI
  5.5. Giai thuat MST
6. Hien thi tren do thi <>
  6.1. Nhap du lieu random
  6.2. Nhap du lieu tu file
  6.3. Chay giai thuat
    6.3.1. Giai thuat APSP
    6.3.2. Giai thuat APSP-I
    6.3.3. Giai thuat APSP-N
    6.3.4. Giai thuat APSP-NI
    6.3.5. Giai thuat MST
0. Thoat

-----
Lua chon chuc nang: _
```

Hình 30: Giao diện chương trình demo với màn hình console

2.3. Chức năng 1: Nhập dữ liệu bằng tay



```
c:\Users\Minh Ngo\Documents\Visual Studio 2008\Projects\SNDP_total\Debug\SNDP_total.exe

0. Thoat
-----
Lua chon chuc nang: 1

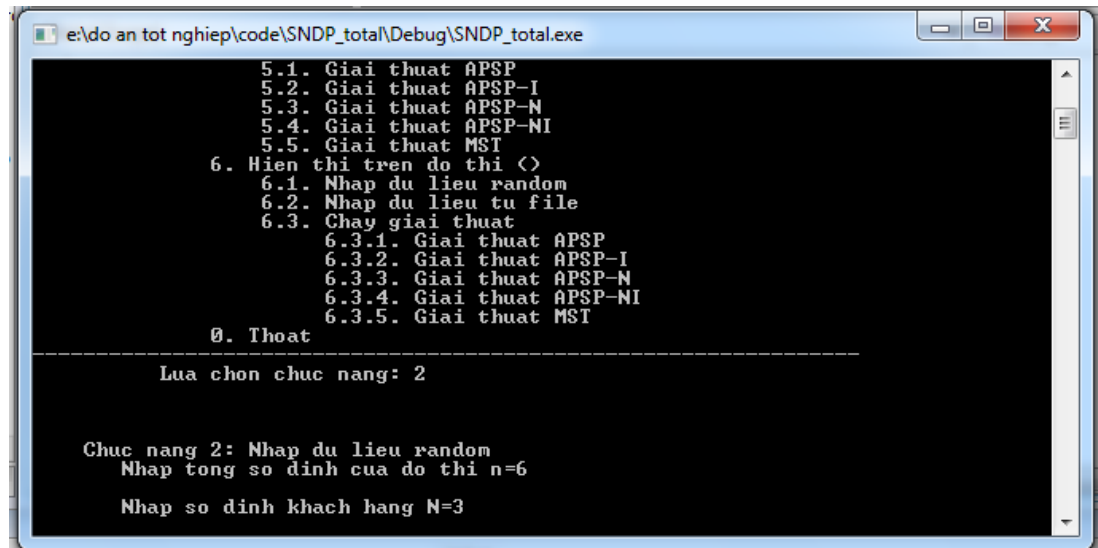
Chuc nang 1: Nhap du lieu bang tay
Nhap tong so dinh cua do thi n=?
Nhap so dinh khach hang N=3
Nhap cac loai khach hang(<=1 neu la khach hang loai 1, =2 neu la khach hang loai 2):
  typeNode[1]=1
  typeNode[2]=2
  typeNode[3]=2
Nhap ma tran chi phi:
  c[0,1]=4
  c[0,2]=5
  c[0,3]=3
  c[0,4]=6
  c[0,5]=4
  c[0,6]=_
```

Hình 31: Chức năng nhập dữ liệu bằng tay

Yêu cầu:

- Nhập tổng số đỉnh của đồ thị ban đầu
- Nhập số lượng khách hàng cần kết nối
- Nhập loại khách hàng: Chỉ được nhập 1 nếu là khách hàng loại 1 và 2 nếu là khách hàng loại 2

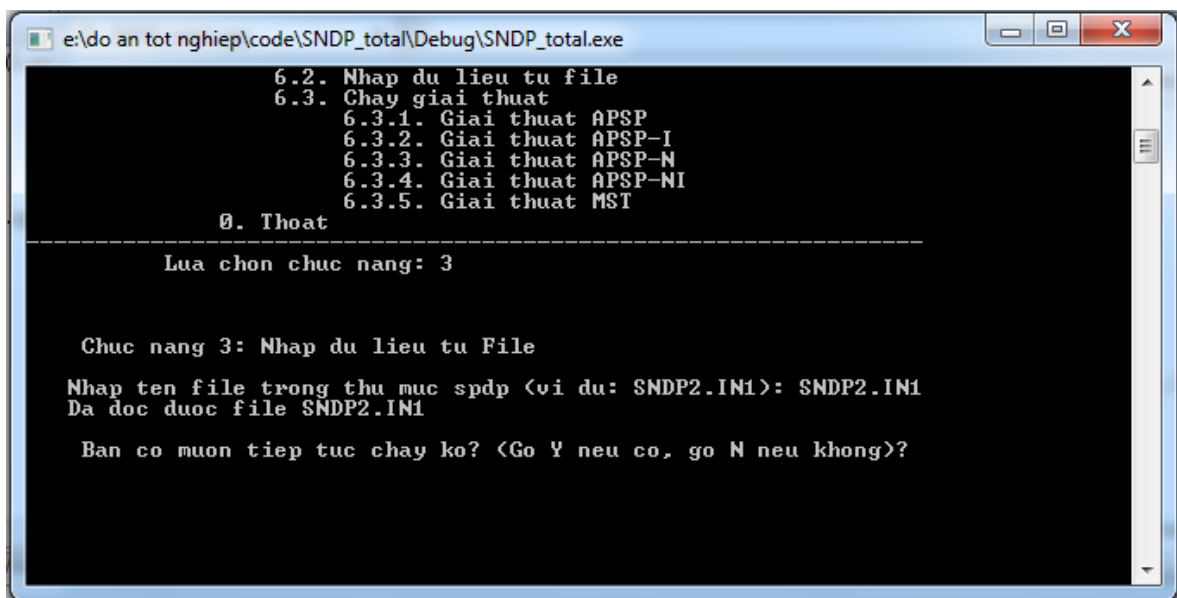
2.4. Chức năng 2: Nhập dữ liệu random



Hình 32: Chức năng nhập dữ liệu random

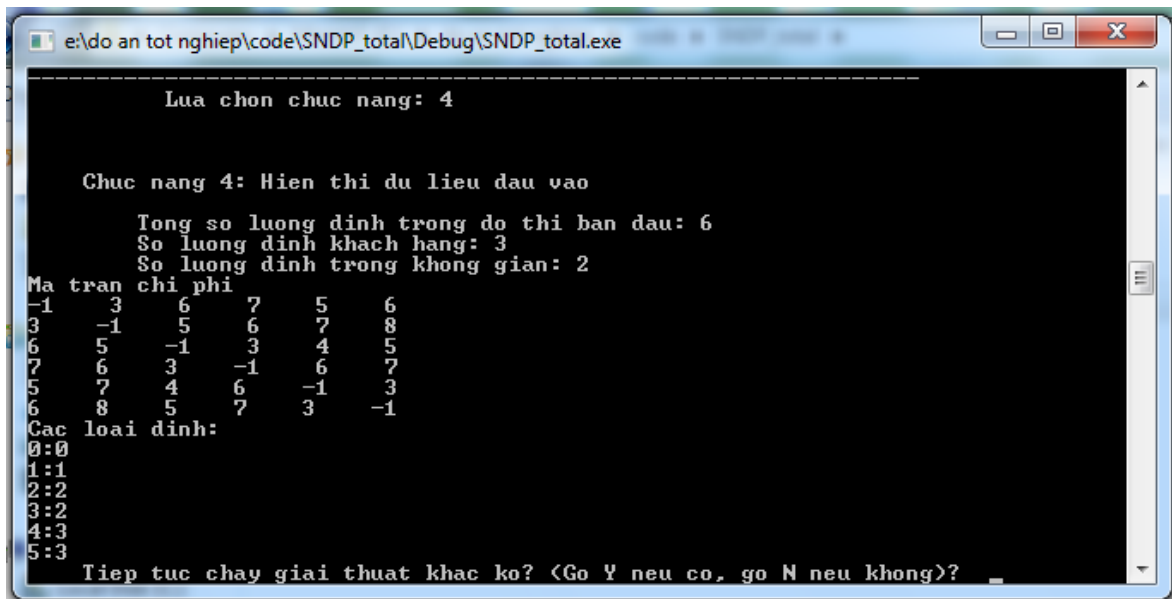
2.5. Chức năng 3: Nhập dữ liệu từ File

Nhập tên file : **SNDP2.IN1**



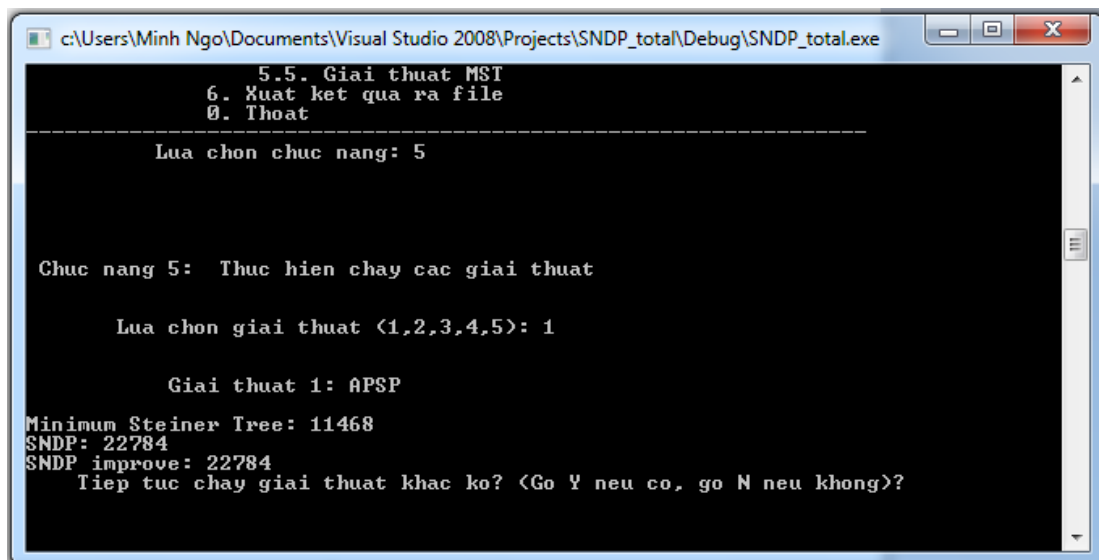
Hình 33: Chức năng nhập dữ liệu từ File

2.6. Chức năng 4: Hiện thị dữ liệu đầu vào



Hình 34: Chức năng hiển thị dữ liệu đầu vào

2.7. Chức năng 5: Thực hiện chạy các giải thuật

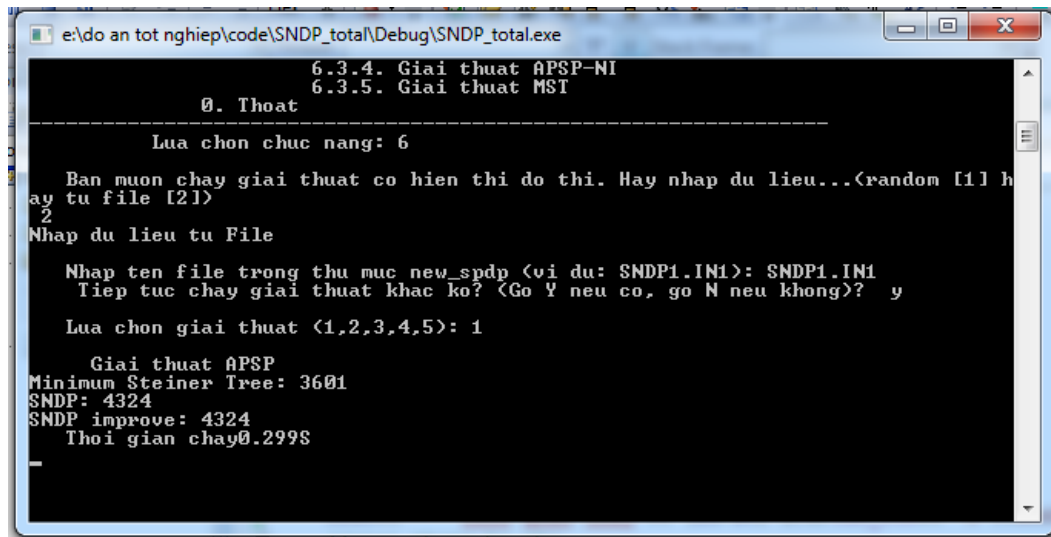


Hình 35: Chức năng chạy các giải thuật

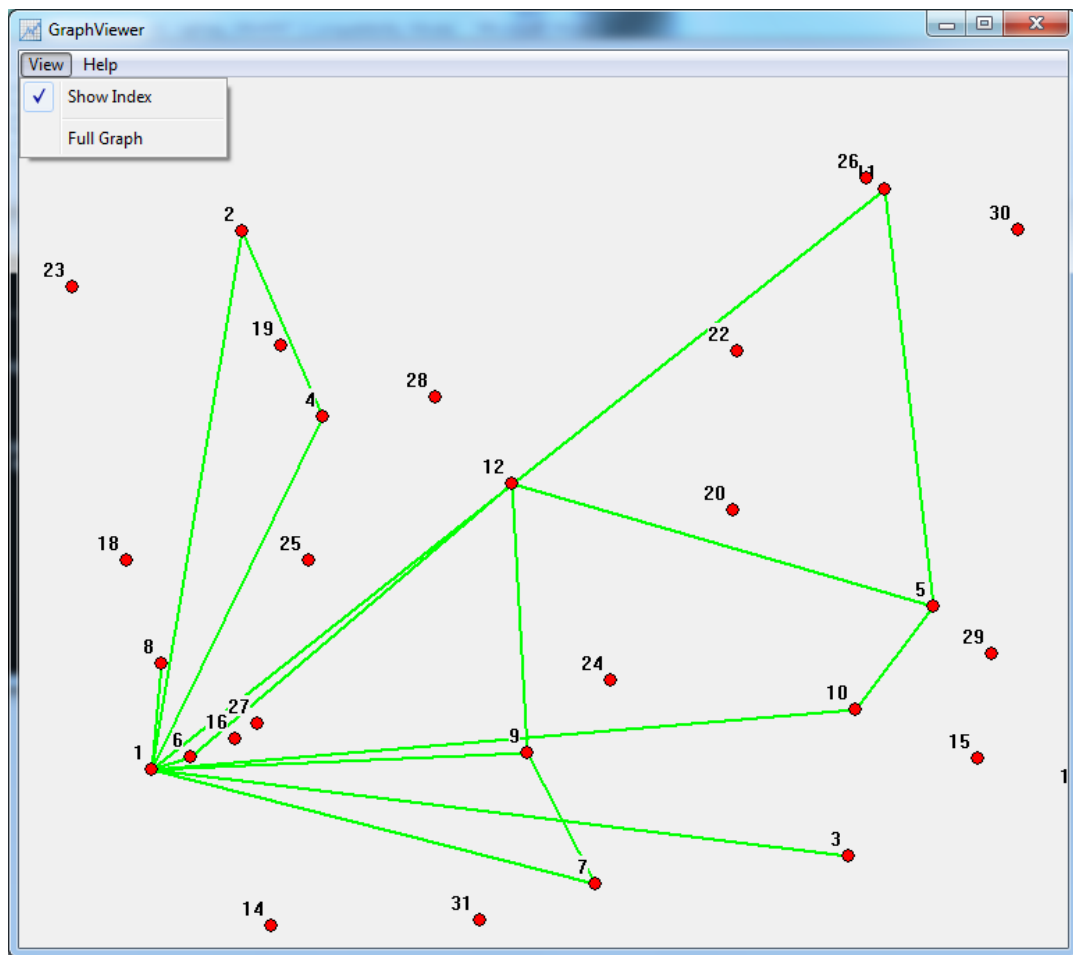
2.8. Chức năng 6: Chạy các giải thuật có hiển thị đồ thị minh họa

Chức năng này chỉ thêm vào để có cái nhìn trực quan cho kết quả chạy của giải thuật.

Nhập tên File: **SNDP1.IN1**



Hình 36: Chức năng chạy các giải thuật đồng thời hiển thị kết quả bằng giao diện đồ thị
Đồng thời đồ thị kết quả sẽ được hiện trên màn hình:



Hình 37: Đồ thị kết quả khi chạy giải thuật

TÀI LIỆU THAM KHẢO

- [1] I. Ljubić, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Programming, Series B*, 105(2-3):427-449, 2006
- [2] D. Wagner, G. R. Raidl, P. Mutzel, and P. Bachhiesl. A multi-commodity flow approach for the design of the last mile in real-world fiber optic networks. In K.-H. Waldmann and U.M. Stocker, editors, *Operations Research Proceedings 2006*, pages 197-202, Springer, 2007
- [3] D. Wagner, U. Pferschy, P. Mutzel, G. R. Raidl, and P. Bachhiesl. A directed cut model for the design of the last mile in real-world fiber optic networks. In B. Fortz, editor Spa, *Proceedings of the International Network Optimization Conference 2007*, pages 103/1-6, Spa, Belgium, 2007
- [4] Jakub Gładysz, Krzysztof Walkowiak. Tabu Search Algorithm for Survivable Network Design Problem with Simultaneous Unicast and anycast Flow, 2010
- [5] L. Bahiense, F. Barahona, and O. Porto. Solving Steiner tree problems in graphs with Lagrangian relaxation. *Journal of Combinatorial Optimization*, 7(3): 259-282, 2003
- [6] M. Leitner and G. R. Raidl. Lagrangian decomposition, metaheuristics, and hybrid approaches for the design of the last mile in fiber optic networks. In M. J. Blesa et al., *Hybrid Metaheuristics 2008*, volume 5296 of LNCS, pages 158-174, Springer, 2008.
- [7] S. A. Canuto, M. G. C. Resende, and C. C. Ribeiro. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 38:50-58, 2001
- [8] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3):207-232, 1998
- [9] Thomas Bucsics, Günther Raidl. Metaheuristic Approaches for Designing Survivable Fiber-Optic Networks. *Institute for Computer Graphics and Algorithms of the Vienna University of Technology*, 2007

- [10] Kenneth Steiglitz, member, IEEE, Petter Weiner, member, and D. J. Kleitman. The Design of Minimum-Cost Survivable Networks. *IEEE transactions on circuit theory*, vol. CT-16, No.4, November 1969
- [11] Hervé Kerivin, A.Ridha Mahjoub. Design of Survivable Networks : A Survey. *Research Report LIMOS/RR-05-04*, 3 mars 2005
- [12] C. C. Ribeiro, E. Uchoa, and R. F. Werneck. A hybrid grasp with perturbations for the steiner problem in graphs. *INFORMS J. on Computing*, 14(3):228–246, 2002
- [13] W. Krings, A. Azadmanesh. Graph Based Model for Survivability Applications, *National Institute of Standards and Technology, U.S. Dept. of Commerce*, 2004
- [14] Qian Lv, Anjing Wang. ILP Applications in Survivable Network Design. 2005