**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**TRẦN VIỆT CHƯƠNG**

# NGHIÊN CỨU PHÁT TRIỂN THUẬT TOÁN METAHEURISTIC GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT ĐỊNH HƯỚNG ỨNG DỤNG CHO THIẾT KẾ HỆ THỐNG MẠNG

**LUẬN ÁN TIẾN SĨ KỸ THUẬT**

HÀ NỘI - 2023

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**TRẦN VIỆT CHƯƠNG**

# NGHIÊN CỨU PHÁT TRIỂN THUẬT TOÁN METAHEURISTIC GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT ĐỊNH HƯỚNG ỨNG DỤNG CHO THIẾT KẾ HỆ THỐNG MẠNG

LUẬN ÁN TIẾN SĨ KỸ THUẬT
CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN
MÃ SỐ: 9.48.01.04

**NGƯỜI HƯỚNG DẪN KHOA HỌC:**
**PGS.TS. HÀ HẢI NAM**
**TS. PHAN TẤN QUỐC**

HÀ NỘI - 2023

# LỜI CAM ĐOAN

Nghiên cứu sinh cam đoan nội dung luận án này là kết quả nghiên cứu của bản thân dưới sự hướng dẫn chính của PGS.TS. Hà Hải Nam và hướng dẫn phụ của TS. Phan Tấn Quốc. Các kết quả và số liệu trình bày trong luận án là trung thực, một phần đã được công bố trong các công trình của nghiên cứu sinh và chưa được công bố trong công trình khoa học của tác giả khác. Tất cả nội dung tham khảo từ những nghiên cứu liên quan đều được nêu rõ ràng trong danh mục tài liệu tham khảo ở phía sau luận án.

*Hà Nội, ngày 09 tháng 5 năm 2023*

**Tác giả**

# LỜI CẢM ƠN

Để hoàn thành luận án này, đầu tiên nghiên cứu sinh chân thành cảm ơn sự hướng dẫn khoa học và tận tình giúp đỡ của PGS.TS. Hà Hải Nam và TS. Phan Tấn Quốc. Nghiên cứu sinh trân trọng cảm ơn quý thầy cô trong Ban Giám đốc Học viện Công nghệ Bưu chính Viễn thông, Hội đồng Tiến sĩ, Khoa Đào tạo Sau Đại học, Khoa Công nghệ thông tin 1 đã tạo điều kiện thuận lợi cho nghiên cứu sinh thực hiện và hoàn thành chương trình nghiên cứu. Xin trân trọng cảm ơn quý Thầy, Cô đã đọc và đóng góp ý kiến hoàn thiện luận án.

Nghiên cứu sinh trân trọng cảm ơn lãnh đạo UBND tỉnh Cà Mau, Ban Giám đốc Sở Thông tin và Truyền thông, Sở Nội vụ tỉnh Cà Mau đã tạo điều kiện công tác thuận lợi và hỗ trợ kinh phí để nghiên cứu sinh tham gia và hoàn thành khóa đào tạo trong hoàn cảnh dịch bệnh Covid-19 diễn ra phức tạp.

Cuối cùng, nghiên cứu sinh xin trân trọng ghi nhận những tình cảm và bày tỏ lòng biết ơn sâu sắc đến cha mẹ, gia đình, người thân, đồng nghiệp, những người đã luôn bên cạnh, động viên và ủng hộ nghiên cứu sinh trong suốt quá trình học tập nghiên cứu.

*Hà Nội, ngày 09 tháng 5 năm 2023*

**Tác giả**

# MỤC LỤC

# DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT

| Thuật ngữ/Từ viết tắt | Nghĩa Tiếng Anh | Nghĩa Tiếng Việt |
|---|---|---|
| Bees-Steiner | Bees-Steiner | Thuật toán Bees-Steiner |
| *HCSMT* | Hill Climbing Search Steiner Minimal Tree | Thuật toán tìm kiếm leo đồi giải bài toán *SMT* |
| *Heu* | Heuristic | Thuật toán *Heu* [77] |
| InitPopulation | Init Population | Khởi tạo quần thể |
| *i-PD*-Steiner | improve *PD*-Steiner | Thuật toán *i-PD*-Steiner |
| *ISP* | Internet Service Provider | Nhà cung cấp dịch vụ Internet |
| *i-SPT*-Steiner | improve *SPT*-Steiner | Thuật toán *i-SPT*-Steiner |
| *LAN* | Local Area Network | Mạng cục bộ |
| LikePrim | Like Prim | Thuật toán tựa Prim |
| *LS* | Local Search | Thuật toán tìm kiếm cục bộ |
| *MST* | Minimum Spanning Tree | Cây khung nhỏ nhất |
| *MST*-Steiner | Minimum Spanning Tree Steiner | Thuật toán *MST*-Steiner |
| *NB* | Node-Based | Thuật toán Node-Based |
| NeighSearch | Neigh Search | Tìm kiếm lân cận |
| *NP* | Nondeterministic Polynomial time | Lớp *NP* |
| *NP-Hard* | Nondeterministic Polynomial time Hard | Lớp *NP-Khó* |
| Opt | Optimal | Giá trị tối ưu |

| | | |
|---|---|---|
| *PB* | Path-Based | Thuật toán Path-Based |
| *PD* | Prim Dijkstra | Thuật toán Prim Dijkstra |
| *PD*-Steiner | Prim Dijkstra Steiner | Thuật toán *PD*-Steiner |
| *PGA* | Parallel Genetic Algorithm | Thuật toán di truyền song song |
| *PGA*-Steiner | *PGA*-Steiner | Thuật toán *PGA*-Steiner |
| RandSearch | Rand Search | Tìm kiếm ngẫu nhiên |
| *SMT* | Steiner Minimal Tree | Cây Steiner nhỏ nhất |
| SortPopulation | Sort Population | Sắp xếp quần thể |
| *SPH* | Shortest Path Heuristic | Thuật toán *SPH* [15] |
| *SPT* | Shortest Path Tree | Cây đường đi ngắn nhất |
| *SPT*-Steiner | Shortest Path Tree Steiner | Thuật toán *SPT*-Steiner |
| Tabu-Steiner | Tabu-Steiner | Thuật toán Tabu-Steiner |
| *TS* | Tabu Search | Thuật toán Tabu Search |
| *URL* | Uniform Resource Locator | Bộ định vị tài nguyên hợp nhất |
| *VLSI* | Very Large Scale Integrated | Mạch tích hợp mật độ cao |
| *VNS* | Variable Neighborhood Search | Thuật toán tìm kiếm lân cận biến đổi |
| *VPN* | Virtual Private Network | Mạng riêng ảo |
| *WLAN* | Wide Local Area Network | Mạng cục bộ mở rộng |

# DANH MỤC CÁC KÝ HIỆU

| Ký hiệu | Ý nghĩa |
|---|---|
| $C(T)$ | Chi phí của cây $T$ |
| $e$ | Cạnh $e$ |
| $E(G)$ | Tập cạnh của đồ thị $G$ |
| $E(T)$ | Tập cạnh của cây $T$ |
| $e_{uv}$ | Cạnh cầu Steiner của $G$ |
| $f(n)$ | Hàm xác định thời gian thực hiện thuật toán |
| $F(s)$ | Hàm mục tiêu |
| $G$ | Đồ thị $G$ |
| $G'$ | Đồ thị rút gọn Steiner của $G$ |
| $k$ | Số gen |
| $L$ | Tập đỉnh Terminal |
| $LS_i$ | Thuật toán tìm kiếm lân cận thứ $i$ |
| $m$ | Số cạnh của đồ thị |
| $n$ | Số đỉnh của đồ thị |
| $N(s)$ | Tập lời giải lân cận |
| $O$ | Khái niệm $O$ lớn xác định độ phức tạp thời gian thực hiện thuật toán |
| $OXY$ | Hệ trục tọa độ $OXY$ |
| $p$ | Số đỉnh của Cây Steiner $T$ |

| | |
|---|---|
| $P$ | Đường đi ngắn nhất từ đỉnh $u$ đến đỉnh $v$ |
| $S$ | Không gian lời giải bài toán |
| $s$ | Lời giải/giải pháp |
| $s$’ | Lời giải lận cận với $s$ |
| $SPT_i$ | Cây đường đi ngắn nhất thứ $i$ |
| $T$ | Cây $T$ |
| $u, v$ | Đỉnh $u, v$ |
| $V(G)$ | Tập đỉnh của đồ thị $G$ |
| $V(T)$ | Tập đỉnh của cây $T$ |
| $w(e)$ | Trọng số cạnh $e$ |
| $\alpha$ | Cận tỉ lệ $\alpha$ |

# DANH MỤC CÁC BẢNG

# DANH MỤC CÁC HÌNH VẼ

# MỞ ĐẦU

## 1. Tính cấp thiết của đề tài

Việc kết nối một tập điểm cho trước với chi phí tối thiểu được xem như một trong những bài toán quan trọng nhất của thiết kế mạng truyền thông. Mạng truyền thông có thể được mô hình hóa bởi đồ thị vô hướng, liên thông và có trọng số. Do vậy, từ những năm 70 của thế kỷ trước, bài toán Cây Steiner (Steiner Tree Problem - STP) trong đồ thị hay bài toán Cây Steiner nhỏ nhất (Steiner Minimal Trees Problem - SMT) là bài toán tối ưu tổ hợp, đã được các nhà khoa học quan tâm nghiên cứu để áp dụng cho thiết kế hệ thống mạng và nhiều ứng dụng quan trọng khác trong khoa học và kỹ thuật. Phần lớn các bài toán tối ưu là bài toán thuộc lớp *NP-hard*, không thể giải trong thời gian đa thức. Chỉ với bài toán quy mô nhỏ thì có thể giải bằng các phương pháp toán chính xác. Các bài toán khác được giải quyết bằng phương pháp xấp xỉ để tạo ra một giải pháp đủ tốt trong một thời gian hợp lý, đó là phương pháp heuristic và metaheuristic.

Ứng dụng bài toán Cây Steiner trong khoa học kỹ thuật nói chung đã được nghiên cứu và công bố trong nhiều công trình [9][17][20][27][37][59][64]. Tuy nhiên, do bản chất đây là bài toán tối ưu thuộc lớp *NP-hard* nên cho đến nay, bài toán vẫn tiếp tục được nghiên cứu nhằm tìm lời giải tối ưu hơn cho các ứng dụng thực tế, đặc biệt là ứng dụng trong thiết kế hệ thống mạng.

Mô hình toán học của bài toán Cây Steiner nhỏ nhất có thể phát biểu như sau [14]:

Cho $G = (V(G), E(G))$ là một đơn đồ thị vô hướng liên thông, có trọng số không âm trên cạnh, trong đó $V(G)$ là tập gồm $n$ đỉnh, $E(G)$ là tập gồm $m$ cạnh, $w(e)$ là trọng số của cạnh $e$ với $e \in E(G)$. Cho $L$ là tập con các đỉnh của $V(G)$, cây $T$ đi qua tất cả các đỉnh trong $L$ được gọi là Cây Steiner của $L$. Chi phí của cây $T$, ký hiệu

là $C(T)$, là tổng trọng số của các cạnh thuộc cây $T$. Bài toán tìm Cây Steiner có chi phí nhỏ nhất được gọi là bài toán Cây Steiner nhỏ nhất.

Trong trường hợp tổng quát, bài toán *SMT* đã được chứng minh thuộc lớp bài toán *NP-hard* [14][52][80]. Bài toán *SMT* có nhiều ứng dụng quan trọng trong một số lĩnh vực khoa học và kỹ thuật, cụ thể như: Bài toán thiết kế mạng truyền thông [13], bài toán thiết kế vi mạch cỡ cực lớn *VLSI* (Very large scale integrated), tin sinh học [19][26], các bài toán liên quan đến hệ thống mạng với chi phí nhỏ nhất [13][19][32][34],…

Bài toán *SMT* vẫn thu hút được sự nghiên cứu của nhiều nhà khoa học trong hàng chục năm qua. Hiện nay, đã có hàng loạt thuật toán giải bài toán *SMT* được đề xuất và có thể chia chúng thành các hướng tiếp cận sau: Các thuật toán rút gọn đồ thị, các thuật toán cận tỉ lệ, các thuật toán tìm lời giải đúng, các thuật toán heuristic và metaheuristic. Trong khi thuật toán heuristic được áp dụng hiệu quả cho bài toán cụ thể, thì thuật toán metaheuristic được xem như khung thuật toán tổng quát có thể áp dụng cho nhiều bài toán tối ưu. Cho đến hiện tại, hướng tiếp cận metaheuristic cho chất lượng lời giải tốt nhất trong số các thuật toán giải gần đúng.

## 2. Đối tượng và phạm vi nghiên cứu

**- Đối tượng nghiên cứu**

Bài toán Cây Steiner nhỏ nhất, các thuật toán heuristic và metaheuristic, các công trình công bố liên quan đến thuật toán heuristic, metaheuristic giải bài toán Cây Steiner nhỏ nhất và các ứng dụng của bài toán Cây Steiner nhỏ nhất.

**- Phạm vi nghiên cứu**

Bài toán Cây Steiner nhỏ nhất hiện được nghiên cứu có ba dạng sau: bài toán Cây Steiner với khoảng cách Euclide, bài toán Cây Steiner với khoảng cách chữ nhật và bài toán Cây Steiner với khoảng cách giữa các đỉnh là trọng số của các cạnh (giá trị khoảng cách ngẫu nhiên).

Luận án này chỉ giới hạn nghiên cứu về các thuật toán heuristic, metaheuristic giải bài toán Cây Steiner với khoảng cách giữa các đỉnh là trọng số của các cạnh (giá trị khoảng cách ngẫu nhiên).

## 3. Mục tiêu nghiên cứu

Mục tiêu của luận án này là nghiên cứu phát triển một số thuật toán gần đúng dạng heuristic, metaheuristic hiệu quả nhằm giải bài toán *SMT* cho chất lượng lời giải tốt hơn so với các thuật toán có cùng cỡ thời gian tính *hoặc* đòi hỏi thời gian tính ít hơn khi so sánh với các thuật toán có chất lượng lời giải tương đương *hoặc* đưa ra lời giải mới tốt nhất trên một số bộ dữ liệu thực nghiệm chuẩn, mở rộng và định hướng ứng dụng cho thiết kế hệ thống mạng.

## 4. Phương pháp nghiên cứu

Luận án kết hợp phương pháp nghiên cứu lý thuyết và thực nghiệm. Trên cơ sở khảo sát các công trình khoa học liên quan đến bài toán Cây Stener nhỏ nhất và ứng dụng trong thiết kế mạng. Mô hình cải thiện chất lượng lời giải bài toán được đề xuất với các thuật toán mới và cải tiến. Các thuật toán đề xuất mới hoặc cải tiến được thực nghiệm, đánh giá dựa trên hệ thống dữ liệu thực nghiệm chuẩn và mở rộng. Việc đánh giá so sánh hiệu năng các thuật toán được dựa trên độ phức tạp thuật toán, chất lượng lời giải và thời gian chạy thực nghiệm.

## 5. Nội dung nghiên cứu

Đề xuất một số thuật toán heuristic, metaheuristic mới hoặc cải tiến, dựa trên ý tưởng tìm cây đường đi ngắn nhất, cây khung nhỏ nhất, các chiến lược tìm kiếm lân cận và lược đồ các metaheuristic cơ bản để giải bài toán Cây Steiner nhỏ nhất và định hướng ứng dụng kết quả nghiên cứu cho thiết kế hệ thống mạng.

## 6. Những đóng góp chính của luận án

Sau đây là những đóng góp chính của luận án:

- Đề xuất hai thuật toán heuristic mới: *SPT*-Steiner và *PD*-Steiner giải bài toán *SMT*; các thuật toán này được cài đặt thực nghiệm trên 98 bộ dữ liệu (gồm có 78 bộ dữ liệu là các đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn và 20 bộ dữ liệu mở rộng là các đồ thị thưa kích thước lớn lên đến 10000 đỉnh - *steinf*). Từ kết quả thực nghiệm, luận án tiến hành so sánh, đánh giá chi tiết hiệu quả của hai thuật toán heuristic đề xuất mới với thuật toán heuristic *MST*-Steiner [14] đã được công bố trước đó. Hai thuật toán đề xuất bởi luận án cho chất lượng lời giải tốt hơn thuật toán *MST*-Steiner [14] trên một số bộ dữ liệu. Thời gian chạy của các thuật toán *SPT*-Steiner và *PD*-Steiner chậm hơn so với thuật toán *MST*-Steiner.

- Đề xuất hai thuật toán heuristic cải tiến: *i-SPT*-Steiner và *i-PD*-Steiner giải bài toán *SMT* trong trường hợp đồ thị thưa kích thước lớn. Hai thuật toán heuristic cải tiến *i-SPT*-Steiner và *i-PD*-Steiner được cài đặt thực nghiệm và so sánh, đánh giá tính hiệu quả trên 80 bộ dữ liệu là các đồ thị thưa kích thước lớn lên đến 100000 đỉnh. Hai thuật toán heuristic cải tiến cho chất lượng lời giải tốt hơn hoặc tương đương thuật toán *MST*-Steiner [14] trên một số bộ dữ liệu. Thời gian chạy của thuật toán *i-PD*-Steiner nhanh hơn so với thuật toán *MST*-Steiner và thuật toán *i-SPT*-Steiner. Thời gian chạy của thuật toán *i-SPT*-Steiner chậm hơn so với thuật toán *MST*-Steiner và thuật toán *i-PD*-Steiner.

- Đề xuất mới ba thuật toán metaheuristic dạng cá thể, quần thể giải bài toán *SMT* đó là: thuật toán *Bees*-Steiner, thuật toán tìm kiếm lân cận biến đổi *VNS* và thuật toán tìm kiếm leo đồi Hill climbing search *(HCSMT)*. Ngoài ra, luận án cũng đề xuất 2 chiến lược tìm kiếm lân cận: Tham lam và có xác suất; đồng thời sử dụng chúng trong lược đồ thuật toán tìm kiếm lân cận biến đổi, nhằm nâng cao hơn nữa chất lượng cho các thuật toán metaheuristic.

- Các thuật toán metaheuristic đề xuất mới này được cài đặt thực nghiệm trên hệ thống dữ liệu thực nghiệm chuẩn và so sánh hiệu quả với các thuật toán

metaheuristic khác hiện biết. Kết quả so sánh cho thấy các thuật toán metaheuristic đề xuất mới cho chất lượng lời giải tốt hơn hoặc bằng các thuật toán metaheuristic công bố trước đó trên một số bộ dữ liệu. Chất lượng của các thuật toán metaheuristic phụ thuộc chủ yếu vào các chiến lược tìm kiếm lân cận.

## 7. Ý nghĩa khoa học và thực tiễn

Việc đề xuất thuật toán dạng heuristic, metaheuristic giải bài toán *SMT* có ý nghĩa quan trọng; một mặt, nhằm giải quyết những bài toán ứng dụng đặt ra trong thực tiễn như vừa nêu; mặt khác, còn là cơ sở để giải quyết một số bài toán tối ưu thuộc lớp *NP-hard* khác.

## 8. Bố cục luận án

Bố cục của luận án được tổ chức thành 3 chương và phần kết luận, cụ thể như sau:

- ***Chương 1:*** Trình bày tổng quan về cơ sở lý thuyết bài toán Cây Steiner nhỏ nhất với các nội dung: Một số định nghĩa, định lý cơ bản liên quan; các dạng của bài toán Cây Steiner nhỏ nhất; sơ lược một số hướng tiếp cận. Tiếp theo, khảo sát một số thuật toán metaheuristic giải bài toán Cây Steiner nhỏ nhất, cụ thể như: Giới thiệu sơ đồ một số thuật toán metaheuristic thường gặp như: thuật toán local search, thuật toán leo đồi, thuật toán tìm kiếm lân cận biến đổi, thuật toán bầy ong; các tiêu chí đánh giá chất lượng thuật toán metaheuristic; khảo sát kết quả một số thuật toán heuristic, metaheuristic hiện biết giải bài toán Cây Steiner nhỏ nhất; định hướng ứng dụng bài toán Cây Steiner nhỏ nhất cho thiết kế hệ thống mạng và cuối cùng là giới thiệu hệ thống dữ liệu thực nghiệm chuẩn và mở rộng cho bài toán.

- ***Chương 2:*** Đề xuất 2 thuật toán heuristic mới *SPT*-Steiner, *PD*-Steiner và 2 thuật toán heuristic cải tiến *i-SPT*-Steiner, *i-PD*-Steiner giải bài toán Cây Steiner nhỏ nhất. Thuật toán heuristic *SPT*-Steiner dựa trên ý tưởng cơ bản của bài toán tìm cây đường đi ngắn nhất và thuật toán heuristic *PD*-Steiner là sự kết hợp ý tưởng chính của thuật toán Prim và Dijkstra. Hai thuật toán heuristic cải tiến *i-SPT*-Steiner

và *i-PD*-Steiner dựa trên ý tưởng thay thế thuật toán Dijkstra bằng thuật toán Dial (một biến thể của thuật toán Dijkstra) trong việc tìm đường đi ngắn nhất. Các thuật toán đề xuất đã được cài đặt thực nghiệm trên hệ thống dữ liệu thực nghiệm chuẩn và mở rộng là các đồ thị thưa có kích thước lên đến 100000 đỉnh. Kết quả thực nghiệm cho thấy, thuật toán đề xuất hiệu quả hơn một số thuật toán giải gần đúng đã được công bố trên một số bộ dữ liệu, nhất là đối với các đồ thị thưa có kích thước lớn.

- ***Chương 3:*** Đề xuất 3 thuật toán metaheuristic giải bài toán Cây Steiner nhỏ nhất; các thuật toán này lần lượt dựa trên khung thuật toán metaheuristic gồm: Thuật toán Bees, Hill climbing search và Variable neighborhood search. Luận án cũng đề xuất một số chiến lược tìm kiếm lân cận cho bài toán Cây Steiner nhỏ nhất. Các thuật toán đề xuất này đã được cài đặt thực nghiệm trên hệ thống dữ liệu thực nghiệm chuẩn; kết quả thực nghiệm cho thấy thuật toán đề xuất cho lời giải với chất lượng tốt hơn một số thuật toán heuristic và metaheuristic hiện biết trên một số bộ dữ liệu. Luận án cũng phân tích ưu nhược điểm của từng thuật toán cụ thể và qua đó định hướng phạm vi áp dụng vào thực tế cho từng thuật toán đề xuất.

Trong phần Kết luận, luận án trình bày những kết quả đạt được và định hướng phát triển cho nghiên cứu trong tương lai khi áp dụng kết quả luận án vào thực tiễn.

# Chương 1. TỔNG QUAN VỀ BÀI TOÁN CÂY STEINER NHỎ NHẤT VÀ ĐỊNH HƯỚNG ỨNG DỤNG CHO THIẾT KẾ HỆ THỐNG MẠNG

*Chương này trình bày tổng quan những vấn đề nghiên cứu của luận án. Thứ nhất tổng quan về bài toán Cây Steiner nhỏ nhất. Thứ hai tiếp cận thuật toán metaheuristic giải bài toán Cây Steiner nhỏ nhất. Thứ ba khảo sát một số thuật toán tiêu biểu giải bài toán Cây Steiner nhỏ nhất. Thứ tư định hướng ứng dụng bài toán Cây Steiner nhỏ nhất cho thiết kế hệ thống mạng và cuối cùng là đề xuất lựa chọn dữ liệu thực nghiệm. Từ kết quả nghiên cứu tổng quan và khảo sát phân tích, đánh giá một số thuật toán tiêu biểu giải bài toán Cây Steiner nhỏ nhất, cho thấy hướng tiếp cận bằng thuật toán metaheuristic là rất khả thi và hiệu quả. Chương này được tổng hợp từ các công trình [CT1], [CT6] và [CT8] trong danh mục các công trình nghiên cứu của tác giả.*

## 1.1. CƠ SỞ LÝ THUYẾT

### 1.1.1. Một số định nghĩa

**Định nghĩa 1.1.** ***Cây Steiner***

Cho $G = (V(G), E(G))$ là một đơn đồ thị vô hướng liên thông, có trọng số không âm trên cạnh, trong đó $V(G)$ là tập gồm $n$ đỉnh, $E(G)$ là tập gồm $m$ cạnh, $w(e)$ là trọng số của cạnh $e$ với $e \in E(G)$. Cho $L$ là tập con các đỉnh của $V(G)$, cây $T$ đi qua tất cả các đỉnh trong $L$ được gọi là Cây Steiner ứng với tập $L$ trên đồ thị $G$. Tập $L$ được gọi là tập *terminal*, các đỉnh thuộc tập $L$ được gọi là đỉnh *terminal*. Đỉnh thuộc cây $T$ mà không thuộc tập $L$ được gọi là đỉnh *Steiner* [14].

**Định nghĩa 1.2.** ***Chi phí Cây Steiner***

Cho $T = (V(T), E(T))$ là một Cây Steiner của đồ thị $G$. Chi phí của cây $T$, ký hiệu là $C(T)$, là tổng trọng số của các cạnh thuộc cây $T$, tức là $C(T) = \sum_{e \in E(T)} w(e)$ [14].

**Định nghĩa 1.3.** ***Cây Steiner nhỏ nhất***

Cho đồ thị $G$ được mô tả như trên, bài toán tìm Cây Steiner có chi phí nhỏ nhất được gọi là bài toán Cây Steiner nhỏ nhất (Steiner minimal trees problem – *SMT*); hoặc được gọi ngắn gọn là bài toán Cây Steiner (Steiner trees problem) [14][16].

*SMT* là bài toán lý thuyết đồ thị thuộc dạng tối ưu tổ hợp [5][3]. Trong trường hợp tổng quát, *SMT* đã được chứng minh thuộc lớp *NP-hard* [14][52].

Khác với bài toán cây khung nhỏ nhất (Minimum spanning tree problem); Cây Steiner chỉ cần đi qua tất cả các đỉnh thuộc tập *terminal L* và có thể thêm một số đỉnh khác nữa thuộc tập $V(G)$ chứ không nhất thiết phải đi qua tất cả các đỉnh của đồ thị $G$ [14].

**Định lý 1.1.** ***Định lý về số đỉnh Steiner***

Cho đồ thị $G$ và tập *terminal L*, Cây Steiner $T$ của $L$ có $p$ đỉnh thì số đỉnh Steiner của $T$ không vượt quá $p - 2$ [1].

**Định nghĩa 1.4.** ***Cạnh cầu Steiner***

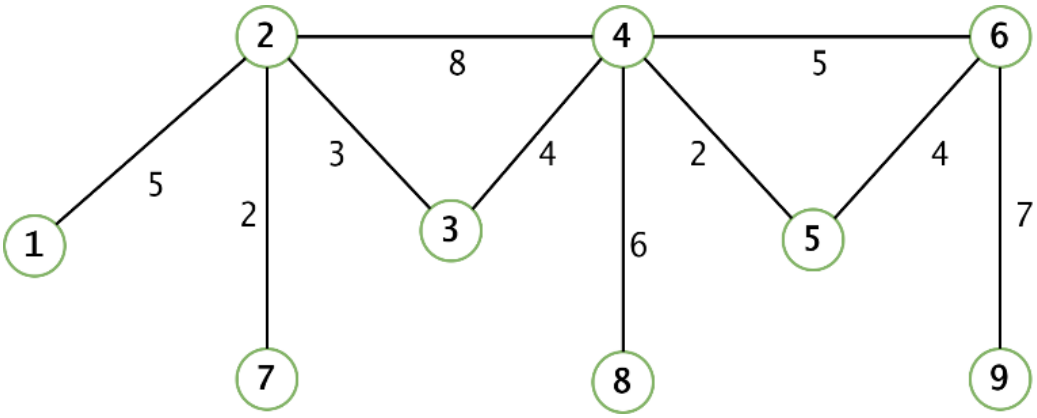
Cho đồ thị $G$ và tập *terminal L,* cạnh $e_{uv}$ được gọi là cạnh cầu *Steiner* của $G$ nếu khi loại cạnh $e_{uv}$ thì tập các đỉnh *terminal L* không cùng thuộc về một thành phần liên thông [1].

**Định nghĩa 1.5.** ***Đồ thị rút gọn Steiner***

Cho đồ thị $G$ và tập *terminal L, G'* được gọi là đồ thị rút gọn *Steiner* của $G$ nếu số đỉnh và số cạnh của $G$' nhỏ hơn hoặc bằng số đỉnh và số cạnh của $G$ và trong $G'$ tồn tại ít nhất một Cây Steiner nhỏ nhất ứng với tập *terminal L* của đồ thị $G$.

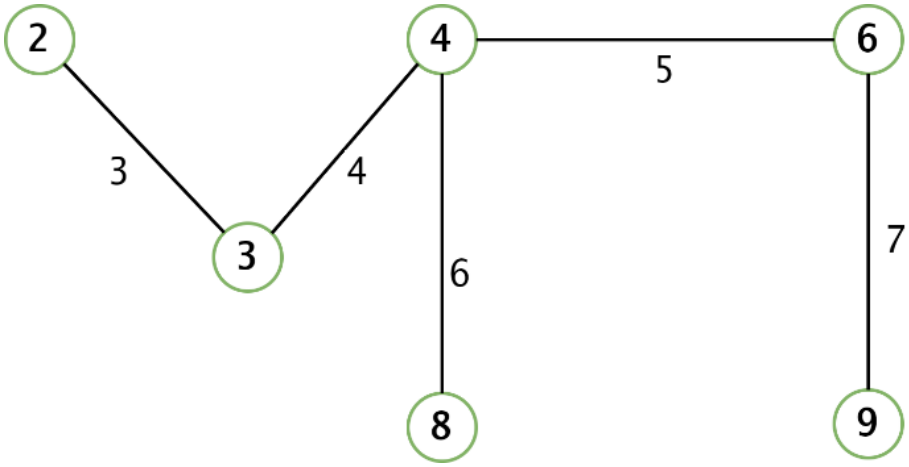
Để ngắn gọn, trong định nghĩa này từ *đồ thị* được hiểu là đơn đồ thị, vô hướng, liên thông và có trọng số không âm.

**Ví dụ 1.1.** Cho một đồ thị $G$ có 9 đỉnh và 10 cạnh như Hình 1.1 và tập *terminal* $L = \{2, 8, 9\}$.




Hình 1.1. Minh họa một đồ thị $G$ vô hướng liên thông có trọng số

Khi đó, Cây Steiner nhỏ nhất tìm được ứng với tập *terminal* $L$ trên đồ thị $G$ là $T$ có $V(T) = \{2, 3, 4, 6, 8, 9\}$ và $E(T) = \{(2, 3), (3, 4), (4, 6), (4, 8), (6, 9)\}$ như được minh họa ở Hình 1.2; cây $T$ có tập đỉnh Steiner là $\{3, 4, 6\}$ và có chi phí là 25.




Hình 1.2. Cây Steiner nhỏ nhất ứng với tập terminal $L$ của đồ thị $G$

### 1.1.2. Một số dạng của bài toán Cây Steiner nhỏ nhất

Bài toán Cây Steiner nhỏ nhất hiện được nghiên cứu ở các dạng sau đây:

- *Thứ nhất* là bài toán Cây Steiner với khoảng cách Euclide**.** Trong mặt phẳng tọa độ OXY, cho đồ thị $G = (V(G), E(G))$ và tập đỉnh $L \subseteq V$. Cần tìm cây đi qua tất

cả các đỉnh thuộc tập $L$ và có tổng độ dài là nhỏ nhất, cho phép thêm vào một số điểm phụ (điểm Steiner) được lấy từ các đỉnh thuộc $V$. Khoảng cách Euclide giữa hai điểm $P_1$ ($x_1$, $y_1$) và $P_2$ ($x_2$, $y_2$) là độ dài đoạn thẳng nối hai đỉnh $P_1$, $P_2$ [11][16][33][41][63].

- *Thứ hai* là bài toán Cây Steiner với khoảng cách chữ nhật**.** Trong mặt phẳng tọa độ OXY cho đồ thị $G = (V(G), E(G))$ và tập đỉnh $L \subseteq V$. Hãy tìm cây đi qua tất cả các đỉnh thuộc tập $L$ và có tổng độ dài là nhỏ nhất, cho phép thêm vào một số điểm phụ (điểm Steiner) được lấy từ các đỉnh thuộc $V$. Khoảng cách chữ nhật (rectilinear distance) giữa hai điểm $P_1$ ($x_1$, $y_1$) và $P_2$ ($x_2$, $y_2$) là $d$ ($P_1$, $P_2$) = $|x_1 - x_2| + |y_1 - y_2|$ [32][35][87].

- *Thứ ba* là bài toán Cây Steiner với khoảng cách ngẫu nhiên. Khoảng cách giữa các đỉnh là trọng số của các cạnh; và đây là phạm vi nghiên cứu về bài toán Cây Steiner nhỏ nhất của tác giả trong đề tài này. Có hai trường hợp đặc biệt đối với bài toán *SMT* là giải được trong thời gian đa thức; đó là khi $L = V(G)$ và khi $|L| = 2$. Khi $L = V(G)$ thì bài toán *SMT* có thể giải bằng các thuật toán tìm cây khung nhỏ nhất: chẳng hạn như thuật toán Prim hoặc thuật toán Kruskal; khi $|L| = 2$ thì bài toán *SMT* có thể giải được bằng các thuật toán tìm đường đi ngắn nhất giữa hai đỉnh, chẳng hạn như thuật toán Dijkstra [38][67]**.**

### 1.1.3. Tổng quan các nghiên cứu liên quan đến bài toán Cây Steiner nhỏ nhất

Hiện tại, trong nước đã có một số công trình nghiên cứu về bài toán Cây Steiner nhỏ nhất như của tác giả Vũ Đình Hòa [1], Trần Lê Thủy [7],...

Trên thế giới đã có nhiều công bố liên quan đến bài toán Cây Steiner nhỏ nhất; trong số đó có luận án của tác giả Martin Zachariasen [53], luận án của tác giả Tobias Polzin [78], luận án của tác giả Pieter Oloff De Wet [61], luận án của tác giả Xinhui Wang [84], luận án của tác giả Jon William Van Laarhoven [46], luận án của tác giả Zhiliu Zhang [87],…

Hiện tại, ở Việt Nam chưa có luận án tiến sĩ kỹ thuật chuyên ngành nghiên cứu về bài toán này.

Có nhiều hướng tiếp cận giải bài toán Cây Steiner nhỏ nhất như các thuật toán rút gọn đồ thị, các thuật toán tìm lời giải đúng, các thuật toán tìm lời giải gần đúng cận tỉ lệ, các thuật toán heuristic và các thuật toán metaheuristic [8][30][45][51][54] [86],...

***- Các thuật toán rút gọn đồ thị***

Một số công trình về rút gọn đồ thị cho bài toán Cây Steiner nghiên cứu các kỹ thuật nhằm giảm thiểu kích thước của đồ thị như công trình của Jeffrey H.Kingston và Nicholas Paul Sheppard [44], công trình của Thorsten Koch và Alexander Martin [77], công trình của C. C. Ribeiro và M.C. Souza [15],…

Ý tưởng chung của các thuật toán rút gọn đồ thị là nhằm đến hai mục tiêu: *Thứ nhất* là gia tăng số lượng các đỉnh thuộc tập *terminal*; *Thứ hai* là loại bỏ các đỉnh của đồ thị mà nó chắc chắn không thuộc về Cây Steiner nhỏ nhất cần tìm.

Chất lượng các thuật toán giải bài toán *SMT* phụ thuộc vào độ lớn của hệ số $n - |L|$; do vậy mục đích của các thuật toán rút gọn đồ thị là làm giảm thiểu tối đa hệ số $n - |L|$.

Các thuật toán rút gọn đồ thị được xem là bước tiền xử lý dữ liệu quan trọng để nâng cao chất lượng lời giải bài toán *SMT*; và công đoạn này càng cần thiết đối với các thuật toán tìm lời giải đúng như quy hoạch động hoặc nhánh cận.

***- Các thuật toán tìm lời giải đúng***

Các nghiên cứu tìm lời giải đúng cho bài toán *SMT* như thuật toán quy hoạch động của Dreyfus và Wagner [68], thuật toán dựa trên phép nới lỏng Lagrange của Beasley [39], các thuật toán nhánh cận của Koch và Martin [77], Xinhui Wang [84],…

Ưu điểm của hướng tiếp cận này là tìm được lời giải chính xác, nhược điểm của hướng tiếp cận này là chỉ giải được các bài toán có kích thước nhỏ; nên khả

năng ứng dụng của chúng không cao. Việc giải đúng bài toán *SMT* thực sự là một thách thức trong lý thuyết tối ưu tổ hợp [14][46][52][58].

Hướng tiếp cận này là cơ sở quan trọng có thể được sử dụng để đánh giá chất lượng lời giải của các thuật toán giải gần đúng khác khi giải bài toán *SMT*.

***- Các thuật toán gần đúng cận tỉ lệ***

Thuật toán gần đúng cận tỉ lệ $\alpha$, nghĩa là lời giải tìm được gần đúng một cận tỉ lệ $\alpha$ so với lời giải tối ưu [14].

Ưu điểm của thuật toán gần đúng cận tỉ lệ là có sự đảm bảo về mặt toán học theo nghĩa cận tỉ lệ trên, nhược điểm của thuật toán dạng này là cận tỉ lệ tìm được trong thực tế thường kém hơn nhiều so với chất lượng lời giải tìm được bởi nhiều thuật toán gần đúng khác dựa trên thực nghiệm. Thuật toán *MST-Steiner* của Bang Ye Wu và Kun-Mao Chao có cận tỉ lệ 2 [14], thuật toán *Zelikovsky-Steiner* có cận tỉ lệ 11/6 [14]; cận tỉ lệ tốt nhất tìm được hiện tại cho bài toán *SMT* là 1.39 [18][50] [65][66][76].

***- Các thuật toán heuristic***

Thuật toán heuristic chỉ những kinh nghiệm riêng biệt để tìm kiếm lời giải cho một bài toán tối ưu cụ thể. Thuật toán heuristic thường tìm được lời giải có thể chấp nhận được trong thời gian cho phép nhưng không chắc đó là lời giải chính xác. Các thuật toán heuristic cũng không chắc hiệu quả trên mọi loại dữ liệu đối với một bài toán cụ thể. Chúng ta có thể bắt gặp các thuật toán heuristic cho lời giải gần đúng trong nhiều bài toán kinh điển như: bài toán người bán hàng, bài toán tô màu, bài toán lập lịch. Bên cạnh đó, chúng ta cũng có thể tìm thấy các thuật toán heuristic cho lời giải đúng như trong việc giải bài toán tìm cây khung nhỏ nhất, bài toán lập lịch hai máy,...

Các thuật toán heuristic điển hình cho bài toán *SMT* như: *MST-Steiner* [14], *SPH* [15], *Heu* [77], distance network heuristic của Kou, Markowsky,..[47][74][75].

***- Các thuật toán metaheuristic***

Thuật toán metaheuristic sử dụng nhiều heuristic kết hợp với các kỹ thuật phụ trợ nhằm khai phá không gian tìm kiếm; metaheuristic thuộc lớp các thuật toán tìm kiếm tối ưu.

Hiện đã có nhiều công trình sử dụng thuật toán metaheuristic giải bài toán *SMT*; chẳng hạn như thuật toán local search [29][43], thuật toán tìm kiếm với lân cận biến đổi [43], thuật toán di truyền [22], thuật toán tabu search [31][72], thuật toán di truyền song song [57],…

## 1.2. TIẾP CẬN THUẬT TOÁN METAHEURISTIC GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

### 1.2.1. Thuật toán heuristic

Thuật toán heuristic là một phương pháp xấp xỉ để tạo ra một lời giải đủ tốt trong một thời gian hợp lý. Nó thể hiện cách giải bài toán với các đặc tính sau [4]:

- Thường tìm được lời giải tốt (nhưng không chắc là lời giải tốt nhất)

- Giải bài toán theo thuật toán heuristic thường dễ dàng và nhanh chóng đưa ra kết quả hơn so với giải thuật tối ưu, vì vậy chi phí thấp hơn.

- Thuật toán heuristic thường thể hiện khá tự nhiên, gần gửi với cách suy nghĩ và hành động của con người.

Có nhiều phương pháp để xây dựng một thuật toán heuristic, trong đó người ta thường dựa vào một số nguyên lý cơ bản sau [4]:

- ***Nguyên lý vét cạn thông minh:*** Trong một bài toán tìm kiếm nào đó, khi không gian tìm kiếm lớn, ta thường tìm cách giới hạn lại không gian tìm kiếm hoặc thực hiện một kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu.

- ***Nguyên lý tham lam (Greedy):*** Lấy tiêu chuẩn tối ưu (trên phạm vi toàn cục) của bài toán để làm tiêu chuẩn chọn lựa hành động cho phạm vi cục bộ của từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải.

- ***Nguyên lý thứ tự:*** Thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát nhằm nhanh chóng đạt được một lời giải tốt.

- ***Hàm Heuristic:*** Trong việc xây dựng các thuật toán heuristic, người ta thường dùng các hàm heuristic. Đó là các hàm đánh giá thô, giá trị của hàm phụ thuộc vào trạng thái hiện tại của bài toán tại mỗi bước giải. Nhờ giá trị này, ta có thể chọn được cách hành động tương đối hợp lý trong từng bước của thuật toán.

### 1.2.2. Thuật toán metaheuristic

Thuật toán metaheuristic được xem như khung thuật toán tổng quát có thể áp dụng cho nhiều bài toán tối ưu. Thuật toán này sử dụng nhiều heuristic kết hợp với các kỹ thuật phụ trợ nhằm khai phá không gian tìm kiếm lời giải của bài toán; thuật toán metaheuristic thuộc lớp các thuật toán tìm kiếm tối ưu.

Thuật ngữ "*metaheuristic*" thường đề cập đến các thuật toán dạng cá thể như: local search, simulated annealing, tabu search, variable neighborhood search,... hoặc các thuật toán dạng quần thể như: genetic algorithm [12][36][70], particle sarm optimization, harmony search, ant algorithm, bee algorithm, bat algorithm, cuckoo search algorithm [82][83],...

Khó để xác định thuật toán metaheuristic dạng cá thể hay thuật toán metaheuristic dạng quần thể, dạng nào là hiệu quả hơn; vì cả hai dạng này đều có thể hiệu quả dưới điều kiện/bộ tham số thực nghiệm thích hợp. Tuy vậy, nhìn chung các thuật toán dạng quần thể có thể có hiệu quả hơn trên các bài toán tối ưu đa mục tiêu vì nhiều phép tìm kiếm được thực hiện cùng lúc; điều này có thể đúng theo quan sát thực nghiệm, nhưng chưa đủ cơ sở lý thuyết để khẳng định.

### 1.2.3. Tính tăng cường và tính đa dạng

Hai yếu tố quan trọng nhất của một thuật toán metaheuristic là tính tăng cường (intensification) và tính đa dạng (diversification). Một thuật toán metaheuristic được đánh giá là hiệu quả nếu nó có khả năng sinh ra một dãy các lời giải đa dạng nhằm

khai phá hiệu quả không gian tìm kiếm lời giải của bài toán. Song song đó, nó cũng cần có khả năng tăng cường tìm kiếm các lân cận xung quanh các lời giải tốt một cách hiệu quả. Tính tăng cường và tính đa dạng là hai thành phần có phần đối lập nhau, việc tận dụng tốt sự phối hợp giữa chúng là chìa khóa thành công của mọi thuật toán metaheuristic.

Tăng cường hóa nhằm mục đích khai thác sâu hơn các vùng không gian lời giải tiềm năng để hy vọng tìm được lời giải có chất lượng tốt hơn. Tăng cường hóa thường được thực hiện nhờ áp dụng các tiếp cận tham lam - tìm kiếm được hướng đến phương án lân cận tốt nhất. Nếu tính tăng cường quá cao, nó có thể nảy sinh vấn đề hội tụ sớm; các lời giải tối ưu cục bộ tìm được có độ lệch cao hoặc vô nghĩa. Ngược lại, nếu tính tăng cường quá thấp, quá trình hội tụ sẽ chậm.

Đa dạng hóa nhằm khai phá những vùng không gian tìm kiếm mới, tránh việc tìm kiếm rơi vào bẫy tối ưu cục bộ. Đa dạng hóa thường được thực hiện qua việc sử dụng tính ngẫu nhiên và có thể kết hợp với một số heuristic riêng biệt cho từng bài toán. Nếu tính đa dạng quá cao thì sẽ có nhiều vùng không gian lời giải được khai phá ngẫu nhiên, theo đó sẽ làm chậm quá trình hội tụ của thuật toán. Nếu tính đa dạng quá thấp thì không gian lời giải cần khai phá sẽ bị giới hạn, các lời giải tìm được có khuynh hướng hội tụ cục bộ hoặc vô nghĩa.

Chất lượng của các thuật toán metaheuristic phụ thuộc vào nhiều yếu tố như: cách thức tạo lời giải/quần thể lời giải ban đầu, điều kiện dừng, cách thức mã hóa và giải mã lời giải [71],... nhưng quan trọng nhất là chiến lược tăng cường hóa, chiến lược đa dạng hóa và sự phối hợp giữa chúng. Chiến lược tăng cường hóa được sử dụng xuyên suốt trong quá trình thuật toán metaheuristic diễn ra, còn chiến lược đa dạng hóa thường được sử dụng khi chất lượng của lời giải hiện tại không được cải thiện sau một số lần lặp xác định.

### 1.2.4. Tiêu chí đánh giá chất lượng thuật toán metaheuristic

Chất lượng của một thuật toán được đánh giá qua chất lượng lời giải và thời gian tính.

Đánh giá độ phức tạp của thuật toán là đánh giá lượng tài nguyên các loại mà thuật toán đòi hỏi sử dụng. Có hai loại tài nguyên quan trọng nhất là bộ nhớ và thời gian. Hiện nay, khi giải quyết nhiều vấn đề ứng dụng thực tiễn quan trọng, người sử dụng quan tâm nhiều hơn đến yếu tố thời gian.

Đối với các thuật toán gần đúng để giải các bài toán *NP-hard* trong tối ưu hóa, vì chúng không đảm bảo đưa ra được lời giải tối ưu cần tìm, nên một tiêu chí không kém phần quan trọng để đánh giá thuật toán gần đúng đó là chất lượng lời giải mà thuật toán đảm bảo.

Đối với lớp những thuật toán gần đúng cận tỷ lệ, có thể đánh giá đóng góp mới thông qua cận tỷ lệ của thuật toán. Tuy nhiên, đối với phần lớn các thuật toán gần đúng hiện nay, việc đánh giá tiên nghiệm chất lượng của lời giải mà chúng đưa ra là không thể thực hiện được. Trong tình huống này, các nhà khoa học chấp nhận giải pháp là đánh giá qua thực nghiệm.

Đối với nhiều lớp bài toán khó có nhiều ứng dụng thực tiễn, cộng đồng các nhà khoa học đã cùng nhau xây dựng bộ dữ liệu thực nghiệm chuẩn (benchmark test). Để chứng tỏ thuật toán đề xuất của mình có những đóng góp mới để giải bài toán đặt ra, các nhà khoa học cần tiến hành thực nghiệm trên các bộ dữ liệu này để chỉ ra thuật toán của mình đề xuất so với những thuật toán hiện biết có những điểm tốt hơn hoặc ở tiêu chí thời gian, hoặc ở tiêu chí chất lượng lời giải.

Khi lựa chọn thuật toán để giải quyết vấn đề ứng dụng thực tiễn, người sử dụng luôn căn cứ vào cả hai tiêu chí thời gian tính và chất lượng lời giải để quyết định dùng thuật toán nào là phù hợp với yêu cầu của thực tiễn ứng dụng. Vì thế, có những vấn đề ứng dụng thực tiễn mà một thuật toán có chất lượng lời giải tốt hơn lại không được lựa chọn sử dụng vì đòi hỏi thời gian không đáp ứng yêu cầu. Tương tự, một thuật toán đòi hỏi ít thời gian nhưng cho chất lượng lời giải không đảm bảo yêu cầu cũng có thể không được lựa chọn sử dụng.

Khi phải lựa chọn thuật toán để giải quyết bài toán ứng dụng thực tiễn, một trong những tiêu chí quan trọng để lựa chọn thuật toán là thời gian tính của thuật toán. Thời gian tính của thuật toán phụ thuộc nhiều yếu tố, chẳng hạn cấu hình máy

tính (cả về phần cứng lẫn phần mềm) thực hiện thuật toán, ngôn ngữ lập trình sử dụng để cài đặt thuật toán, trình độ của lập trình viên cài đặt thuật toán,…

Mặc dù biết rằng, sai khác thời gian tính của thuật toán trên các cấu hình máy tính khác nhau, hoặc các ngôn ngữ lập trình khác nhau, hoặc trình độ lập trình viên khác nhau là không quá hằng số, nhưng người sử dụng rất cần có những thông tin chi tiết để lựa chọn thuật toán thích hợp vào việc giải quyết vấn đề cụ thể của mình. Chẳng hạn nếu đó là vấn đề ứng dụng trong vận hành hệ thống, thì cần lựa chọn thuật toán cho lời giải chất lượng tốt nhất trong số các thuật toán đáp ứng được yêu cầu hạn chế về thời gian. Còn nếu đó là vấn đề phát sinh trong tính toán thiết kế hệ thống, thì các thuật toán đưa ra lời giải với chất lượng cao hơn lại đặc biệt được quan tâm. Trong lý thuyết phân tích độ phức tạp tính toán của thuật toán, các nhà khoa học đã đưa ra tiêu chí khách quan (không phụ thuộc vào các yếu tố mang tính chủ quan như đã nêu trên) để đánh giá thời gian tính của thuật toán: Đó là đánh giá thời gian tính của thuật toán giải bài toán bởi một hàm của kích thước dữ liệu đầu vào của bài toán, được ghi nhận dưới dạng ký pháp tiệm cận (asymptotic notation), trong đó ký hiệu $O$ được sử dụng để ghi nhận đánh giá tiệm cận trên. Tuy nhiên, một nhược điểm của việc sử dụng ký hiệu tiệm cận chính là kết quả so sánh tốc độ tăng của các hàm chỉ đúng khi đối số "đủ lớn". Vì thế, khi đối số chưa đủ lớn thì kết quả so sánh có thể là không đúng. Chẳng hạn, một thuật toán có đánh giá thời gian tính là $f(n) = 1000n^2 \in O(n^2)$ là nhanh hơn thuật toán có đánh giá $g(n) = 2n^3 \in O(n^3)$ khi $n$ đủ lớn. Nhưng khi $n < 100$, dễ thấy là đòi hỏi $1000n^2$ ($= 10^7$, khi $n = 100$) là lớn hơn đòi hỏi $2n^3$ ($= 2*10^6$, khi $n = 100$) đến 5 lần.

Mặt khác, người sử dụng rất cần thông tin chi tiết về thời gian mà thuật toán đòi hỏi để đưa ra lời giải có chất lượng đáp ứng yêu cầu đặt ra đối với những kích thước cụ thể tương ứng với kích thước bài toán ứng dụng mà họ cần lựa chọn thuật toán giải. Để đáp ứng yêu cầu này, bên cạnh việc đưa ra đánh giá thời gian tính lý thuyết của thuật toán trong ký pháp tiệm cận, các nhà khoa học khi phát triển thuật toán thường đưa ra thông tin về thời gian tính thực nghiệm của thuật toán.

Khi so sánh thời gian tính của các thuật toán khác nhau dựa trên thực nghiệm, lại phát sinh yêu cầu là, khi so sánh các thuật toán cần chạy trên một hạ tầng thông

tin; mà để đáp ứng yêu cầu này, khi tiến hành thực nghiệm các tác giả không những phải cài đặt thuật toán của mình mà còn phải cài đặt lại các thuật toán của các tác giả khác trên cùng một ngôn ngữ lập trình và chạy trên cùng một cấu hình máy tính để giải cùng bộ dữ liệu chuẩn. Đây là một thách thức với cộng đồng những nhà khoa học trong lĩnh vực phát triển thuật toán. Vấn đề này đã được giải quyết từ cuối những năm 1980 của thế kỷ trước; đó là dựa vào thông tin đánh giá tốc độ xử lý của các máy tính được các chuyên gia máy tính đưa ra. Hiện nay, trong công trình [42] đã trình bày cách đánh giá hiệu quả của các hệ thống máy tính khác nhau bằng việc sử dụng các phần mềm tính toán giải hệ phương trình tuyến tính. Công trình này sử dụng đơn vị đo Mflop/s (Million Floating-point Operations Per Second – triệu phép tính dấu phẩy động trên giây) để đánh giá hiệu suất của các máy tính. Đơn vị đo này đặc biệt có ý nghĩa trong lĩnh vực tính toán khoa học sử dụng nhiều tính toán thập phân. Kết quả của công trình nghiên cứu là bảng số liệu về tốc độ đo bởi Mflop/s cho mỗi một cấu hình máy tính. Sử dụng bảng thông tin này, các tác giả không cần cài đặt lại thuật toán của người khác, mà để so sánh tương đối thời gian tính của các thuật toán có thể đưa ra các thông tin sau: Thời gian tính được công bố bởi chính tác giả thuật toán; thông tin về cấu hình máy tính thực hiện thuật toán; qui đổi thời gian tính dựa trên thông tin về tốc độ máy tính lấy từ công trình của Dongarra [42].

Việc so sánh thời gian tính các thuật toán trên các máy tính có cấu hình khác nhau chỉ mang tính tương đối; tuy nhiên các thông tin so sánh về thời gian chạy cũng cho ta một cái nhìn tham khảo về thời gian tính của các thuật toán.

### 1.2.5. Sơ đồ chung của thuật toán metaheuristic

Sơ đồ chung của một thuật toán metaheuristic [82][83] giải bài toán tối ưu có thể được biểu diễn như sau:

**Algorithm 1.1: Thuật toán metaheuristic**

1. Khởi tạo lời giải ban đầu; //lời giải có thể là một cá thể hoặc một quần thể;
2. Định nghĩa các chiến lược tăng cường hóa;

```
3. Định nghĩa các chiến lược đa dạng hóa;
4. Tính độ thích nghi cho mỗi lời giải;
5. while (điều kiện dừng chưa thỏa) do
6.     Thực hiện các chiến lược tăng cường hóa;
7.     Thực hiện các chiến lược đa dạng hóa;
8.     Cập nhật lời giải tốt nhất cho đến thời điểm hiện tại;
9. end while
10. Trả về lời giải tốt nhất tìm được;
```

### 1.2.6. Phân tích các thành phần của một thuật toán metaheuristic

***- Khởi tạo lời giải ban đầu***

Trong các thuật toán metaheuristic, lời giải ban đầu được khởi tạo ngẫu nhiên hoặc theo một hoặc một số heuristic riêng của bài toán.

Thường thì lời giải ban đầu được khởi tạo ngẫu nhiên, nhất là khi số lần thực nghiệm trên mỗi bộ dữ liệu là đủ lớn để kết quả thực nghiệm là khách quan.

***- Chiến lược tăng cường hóa lời giải***

Tính tăng cường hóa cho lời giải thường được thực hiện dựa vào các chiến lược tìm kiếm lân cận.

Tùy theo đặc tính của bài toán mà có thể áp dụng các cách thức tìm kiếm lân cận khác nhau; trong đó thường là: Thay một gen ngẫu nhiên trong lời giải (cá thể) bằng một gen ngẫu nhiên khác, hoặc thay một gen ngẫu nhiên trong lời giải bằng một gen tốt nhất trong tập (hoặc tập con) các gen ứng viên.

***- Chiến lược đa dạng hóa lời giải***

Một lời giải khi nó được cho thực hiện liên tiếp việc tìm kiếm lân cận theo một cách thức nào đó, thì sau một số lần lặp, chất lượng của lời giải đó sẽ không còn cải thiện được nữa (lời giải được gọi là đã khai thác cạn); nếu tiếp tục tìm kiếm lân cận theo hướng đó thì sẽ bế tắc.

Có hai chiến lược đa dạng hóa lời giải thường được sử dụng: Thay thế ngẫu nhiên *k* gen của lời giải bằng *k* gen ngẫu nhiên khác mà không quan tâm đến chất lượng lời giải sau phép thay thế, hoặc tạo lời giải hoàn toàn mới dựa vào xác suất của mỗi gen (phụ thuộc vào đặc tính riêng của từng bài toán) để thay thế cho lời giải đã được khai thác cạn.

***- Điều kiện dừng của thuật toán metaheuristic***

Có nhiều điều kiện dừng đã được áp dụng trong các thuật toán metaheuristic, sau đây là ba điều kiện dừng thường được sử dụng:

+ *Thứ nhất*, lời giải tốt nhất tìm được của bài toán không được cải thiện sau *một số lần lặp định trước*;

+ *Thứ hai*, số lần lặp của thuật toán đạt đến một *giá trị định trước;*

+ *Thứ ba*, thuật toán chạy hết một lượng *thời gian định trước* (các tham số này được đề xuất từ quá trình thực nghiệm tham số).

Tiếp theo, nghiên cứu sinh sẽ trình bày sơ đồ cơ bản của một số thuật toán metaheuristic liên quan đến các thuật toán đề cập trong luận án này.

### 1.2.7. Thuật toán Local Search

Từ một giải pháp hiện tại được chọn là *s*, thuật toán local search sẽ tìm một giải pháp lân cận *s*’ của *s*; nếu *s*’ tốt hơn *s* thì *s*’ sẽ trở thành giải pháp hiện tại. Quá trình này sẽ dừng khi thuật toán lặp đến một số lần định trước hoặc khi giải pháp tốt nhất không được cải thiện qua một số lần lặp xác định.

Local search là thuật toán quan trọng của dạng thuật toán metaheuristic [29][81]; là nền tảng của nhiều thuật toán metaheuristic khác.

**Algorithm 1.2: Thuật toán Local Search**

1. Gọi *S* là toàn bộ (hoặc một phần) không gian lời giải của bài toán;
2. Khởi tạo ngẫu nhiên giải pháp *s* thuộc *S*;

3. **while** (*điều kiện dừng chưa được thỏa*) **do**

4. Tìm một giải pháp lân cận *s'* của *s* với *s'* thuộc *S*;

5. Nếu *s'* tốt hơn *s* thì *s'* sẽ trở thành giải pháp hiện tại, tức đặt *s = s'*;

6. **end while**

7. *s* là lời giải tìm được của bài toán;

Trong đó điều kiện dừng có thể sử dụng là: Khi thuật toán lặp đến một số lần định trước hoặc khi giải pháp tốt nhất không được cải thiện qua một số lần lặp xác định.

### 1.2.8. Thuật toán Hill Climbing Search

Thuật toán Hill climbing search [10] là một trong những kỹ thuật dùng để tìm kiếm tối ưu cục bộ cho một bài toán tối ưu. Thuật toán Hill climbing search liên tục thực hiện việc di chuyển từ một lời giải *s* đến một lời giải mới *s'* trong một cấu trúc lân cận xác định trước theo sơ đồ sau:

**Algorithm 1.3: Thuật toán Hill climbing search**

**1.** Khởi tạo: Chọn lời giải xuất phát *s*, tính giá trị hàm mục tiêu *F*(*s*).

**2.** Sinh lân cận: Chọn tập lân cận *N*(*s*) và tìm lời giải *s'* trong tập lân cận này với giá trị hàm mục tiêu *F*(*s'*).

**3.** Test chấp nhận: Kiểm tra xem có chấp nhận di chuyển từ *s* sang *s'*. Nếu chấp nhận thì thay *s* bằng *s'*; ngược lại giữ nguyên *s* là lời giải hiện tại.

**4.** Test điều kiện dừng: Nếu điều kiện dừng thỏa mãn thì kết thúc thuật toán và đưa ra lời giải tốt nhất tìm được; ngược lại quay lại bước 2;

Vấn đề lớn nhất mà thuật toán Hill climbing search gặp phải là nó dễ rơi vào bẫy tối ưu cục bộ, đó là lúc chúng ta leo lên một đỉnh mà chúng ta không thể tìm

được một lân cận nào tốt hơn được nữa nhưng đỉnh này lại không phải là đỉnh cao nhất. Để giải quyết vấn đề này, khi leo đến một đỉnh tối ưu cục bộ, để tìm được lời giải tốt hơn nữa, ta có thể lặp lại thuật toán Hill climbing search với nhiều điểm xuất phát khác nhau được chọn ngẫu nhiên và lưu lại kết quả tốt nhất ở mỗi lần lặp. Nếu số lần lặp đủ lớn thì ta có thể tìm đến được đỉnh tối ưu toàn cục. Tuy nhiên, với những bài toán mà không gian tìm kiếm lớn, thì việc tìm được lời giải tối ưu toàn cục là một thách thức lớn. Để giải quyết vấn đề tối ưu cục bộ, trong chương này luận án đề xuất việc kết hợp thuật toán Hill climbing search với chiến lược tìm kiếm lân cận ngẫu nhiên với hy vọng nâng cao chất lượng lời giải của thuật toán.

### 1.2.9. Thuật toán tìm kiếm lân cận biến đổi

Ý tưởng của thuật toán tìm kiếm lân cận biến đổi (Variable neighborhood search - *VNS*) [60] là thực hiện lần lượt từng thuật toán lân cận, hết thuật toán lân cận này đến thuật toán lân cận khác. Trong quá trình thực hiện thuật toán *VNS*, ta luôn ghi nhận lời giải tốt nhất (kỷ lục). Khi thực hiện một thuật toán lân cận, nếu tìm được kỉ lục mới thì ta quay trở lại thực hiện thuật toán *VNS* từ đầu; ngược lại, ta chuyển sang thuật toán lân cận tiếp theo. Quá trình tìm kiếm được tiếp tục cho đến khi thực hiện hết tất cả các thuật toán lân cận mà lời giải tốt nhất không được cải thiện.

**Algorithm 1.4: Thuật toán Variable Neighborhood Search**

1. Gọi $S$ là toàn bộ (hoặc một phần) không gian lời giải của bài toán;
2. Khởi tạo ngẫu nhiên giải pháp $s$ thuộc $S$;
3. Gọi $LS_1$, $LS_2$, …, $LS_k$ là các thuật toán tìm lân cận của bài toán đang xét;
4. **while** (điều kiện dừng chưa được thỏa) **do**
5. **for** (int $i = 1$; $i <= k$; $i$++) **do**
6. Gọi $s$’ là lận cận của $s$ với $s$’ được tìm thấy bằng $LS_i$;

7. Nếu *s'* tốt hơn *s* thì break; // thoát khỏi vòng lặp for;

8. **end for**

9. **end while**

10. *s* là lời giải tìm được của bài toán;

Trong đó điều kiện dừng có thể sử dụng là: Khi thực hiện hết tất cả các thuật toán lân cận mà lời giải tốt nhất không được cải thiện.

### 1.2.10. Thuật toán Bees cơ bản

Các thuật toán mô phỏng theo quá trình tìm kiếm thức ăn của loài ong đã được biết đến trong các công trình của các nhóm tác giả Dusan Teodorovic (2010) [25][28], D.T. Pham (2005) [21], Xin-She Yang (2010) [83],…

Các thuật toán Bees sử dụng đồng thời ba chiến lược tìm kiếm sau: *tìm kiếm lân cận*, *tìm kiếm ngẫu nhiên* và *tìm kiếm sâu hơn ở những vùng tiềm năng*. Các thuật toán Bees được đánh giá là thích hợp để giải các bài toán tối ưu tổ hợp khó so với nhiều metaheuristic phổ biến trước đó [4][6][83].

**Algorithm 1.5: Thuật toán Bees**

1. Khởi tạo quần thể ban đầu với *N* cá thể ong; các cá thể này được tạo ngẫu nhiên hoặc bằng một heuristic đặc thù tùy theo bài toán.

2. Đánh giá độ thích nghi cho mỗi cá thể của quần thể.

3. **while** (*điều kiện dừng chưa thoả*) **do**

4. Trong *N* cá thể ong, chọn ngẫu nhiên ra *p* cá thể để thực hiện tìm kiếm lân cận (*p* < *N*). Trong số *p* cá thể được chọn này, chọn ra *h* cá thể có độ thích nghi cao nhất (phân bổ các thể vào ba nhóm khác nhau để thực hiện việc tìm kiếm).

5. Cử thêm *nep* ong để thực hiện việc tìm kiếm lân cận cho *h* cá thể tốt nhất trên và *nsp* ong để thực hiện việc tìm kiếm lân cận cho *p* – *h* cá thể được chọn còn lại. Đánh giá độ thích

nghi cho các cá thể vừa được bổ sung (*nsp < nep, nsp và nep* là các số nguyên dương, thực hiện việc tìm kiếm lân cận).

6. Sau khi thực hiện tìm kiếm lân cận, mỗi cá thể (gốc) có thể sinh thêm một hoặc một số lân cận. Trong số cá thể gốc và các cá thể lân cận của nó (có thể gọi là một vùng các cá thể) chọn ra một cá thể có độ thích nghi cao nhất đại diện cho vùng cá thể đó ở lần tiến hóa tiếp theo.

7. Mỗi cá thể trong số $N - p$ cá thể không được chọn sẽ được thay thế bằng một cá thể ngẫu nhiên (thực hiện việc tìm kiếm ngẫu nhiên). Đánh giá độ thích nghi cho các cá thể được bổ sung.

8. **end while**

9. **return** cá thể tốt nhất trong quần thể ở thế hệ cuối cùng;

Thuật toán Bees cơ bản trên sử dụng các tham số sau: $N$ là số lượng cá thể ong được khởi tạo; $p$ là số lượng cá thể được chọn trong số $N$ cá thể; $h$ là số lượng cá thể có chất lượng tốt nhất trong số $p$ cá thể được chọn; $p - h$ là số lượng cá thể được chọn còn lại; $N - p$ là số lượng cá thể không được chọn; *nep* là số lượng ong cử đến $h$ cá thể tốt nhất; *nsp* là số lượng ong cử đến $p - h$ cá thể được chọn còn lại.

## 1.3. KHẢO SÁT MỘT SỐ THUẬT TOÁN TIÊU BIỂU GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

Kết quả khảo sát thực nghiệm một số thuật toán heuristic, metaheuristic tiêu biểu được ghi nhận lại trong các Bảng 1.1, 1.2, 1.3. Các bảng có cấu trúc như sau: Cột *Test* là tên các bộ dữ liệu trong hệ thống dữ liệu thực nghiệm; mỗi cột tiếp theo ghi nhận giá trị chi phí Cây Steiner tương ứng với mỗi thuật toán. Cột *Opt* ghi nhận giá trị ứng với thuật toán giải chính xác của J. E. Beasley [39]; cột *SPH* [15], *Heu* [77] ghi nhận giá trị của các thuật toán heuristic; các cột *NB* [69], *PB* [69], *TS* [15], *PGA* [57] ghi nhận giá trị của một số thuật toán metaheuristic điển hình hiện biết.

Bảng 1.1. Kết quả thực nghiệm một số thuật toán trên nhóm đồ thị *steinc*

| **Test** | ***Opt*** | ***SPH*** | ***Heu*** | ***NB*** | ***PB*** | ***TS*** | ***PGA*** |
|---|---|---|---|---|---|---|---|
| steinc1 | 85 | 105 | 85 | 85 | 85 | 85 | 85 |
| steinc2 | 144 | 165 | 144 | 144 | 144 | 144 | 144 |
| steinc3 | 754 | 776 | 755 | 754 | 754 | 754 | 754 |
| steinc4 | 1079 | 1092 | 1080 | 1079 | 1079 | 1079 | 1079 |
| steinc5 | 1579 | 1593 | 1579 | 1579 | 1579 | 1579 | 1579 |
| steinc6 | 55 | 70 | 55 | 55 | 55 | 55 | 55 |
| steinc7 | 102 | 113 | 102 | 102 | 103 | 102 | 102 |
| steinc8 | 509 | 527 | 510 | 509 | 509 | 509 | 509 |
| steinc9 | 707 | 723 | 715 | 707 | 707 | 707 | 707 |
| steinc10 | 1093 | 1109 | 1093 | 1093 | 1093 | 1093 | 1093 |
| steinc11 | 32 | 42 | 32 | 32 | 33 | 32 | 32 |
| steinc12 | 46 | 55 | 46 | 46 | 46 | 46 | 46 |
| steinc13 | 258 | 272 | 262 | 258 | 258 | 258 | 258 |
| steinc14 | 323 | 335 | 324 | 323 | 323 | 324 | 323 |
| steinc15 | 556 | 563 | 557 | 556 | 556 | 556 | 556 |
| steinc16 | 11 | 15 | 11 | 11 | 11 | 11 | 11 |
| steinc17 | 18 | 20 | 19 | 18 | 18 | 18 | 18 |
| steinc18 | 113 | 125 | 120 | 116 | 116 | 117 | 113 |
| steinc19 | 146 | 159 | 150 | 147 | 147 | 148 | 146 |
| steinc20 | 267 | 269 | 268 | 267 | 268 | 267 | 267 |

Bảng 1.2. Kết quả thực nghiệm một số thuật toán trên nhóm đồ thị *steind*

| **Test** | ***Opt*** | ***SPH*** | ***Heu*** | ***NB*** | ***PB*** | ***TS*** | ***PGA*** |
|---|---|---|---|---|---|---|---|
| steind1 | 106 | 109 | 106 | 106 | 106 | 106 | 106 |
| steind2 | 220 | 243 | 220 | 220 | 220 | 220 | 220 |
| steind3 | 1565 | 1606 | 1570 | 1565 | 1565 | 1567 | 1565 |
| steind4 | 1935 | 1975 | 1936 | 1935 | 1935 | 1935 | 1935 |
| steind5 | 3250 | 3286 | 3252 | 3250 | 3254 | 3250 | 3250 |
| steind6 | 67 | 85 | 70 | 68 | 70 | 70 | 67 |
| steind7 | 103 | 105 | 103 | 103 | 103 | 103 | 103 |
| steind8 | 1072 | 1106 | 1092 | 1072 | 1077 | 1078 | 1072 |
| steind9 | 1448 | 1504 | 1462 | 1448 | 1449 | 1450 | 1448 |
| steind10 | 2110 | 2148 | 2113 | 2110 | 2111 | 2112 | 2110 |
| steind11 | 29 | 38 | 29 | 29 | 29 | 30 | 29 |
| steind12 | 42 | 49 | 42 | 42 | 42 | 42 | 42 |
| steind13 | 500 | 523 | 510 | 501 | 502 | 502 | 500 |
| steind14 | 667 | 699 | 675 | 669 | 667 | 667 | 669 |
| steind15 | 1116 | 1137 | 1120 | 1117 | 1120 | 1117 | 1116 |
| steind16 | 13 | 14 | 13 | 13 | 13 | 13 | 13 |
| steind17 | 23 | 26 | 23 | 23 | 23 | 23 | 23 |
| steind18 | 223 | 250 | 238 | 228 | 228 | 230 | 225 |
| steind19 | 310 | 336 | 325 | 313 | 317 | 315 | 312 |
| steind20 | 537 | 541 | 539 | 537 | 539 | 538 | 537 |

Bảng 1.3. Kết quả thực nghiệm một số thuật toán trên nhóm đồ thị *steine*

| **Test** | ***Opt*** | ***SPH*** | ***Heu*** | ***NB*** | ***PB*** | ***TS*** | ***PGA*** |
|---|---|---|---|---|---|---|---|
| steine1 | 111 | 152 | 111 | 111 | 111 | 111 | 111 |
| steine2 | 214 | 255 | 214 | 214 | 214 | 216 | 214 |
| steine3 | 4013 | 4121 | 4052 | 4016 | 4018 | 4018 | 4015 |
| steine4 | 5101 | 5215 | 5114 | 5101 | 5101 | 5105 | 5101 |
| steine5 | 8128 | 8249 | 8130 | 8128 | 8128 | 8128 | 8128 |
| steine6 | 73 | 96 | 73 | 73 | 73 | 73 | 73 |
| steine7 | 145 | 168 | 149 | 145 | 145 | 149 | 145 |
| steine8 | 2640 | 2745 | 2686 | 2648 | 2648 | 2649 | 2645 |
| steine9 | 3604 | 3734 | 3656 | 3608 | 3608 | 3605 | 3607 |
| steine10 | 5600 | 5682 | 5614 | 5600 | 5602 | 5602 | 5600 |
| steine11 | 34 | 43 | 34 | 34 | 34 | 34 | 34 |
| steine12 | 67 | 73 | 68 | 67 | 67 | 68 | 67 |
| steine13 | 1280 | 1342 | 1312 | 1292 | 1292 | 1299 | 1286 |
| steine14 | 1732 | 1785 | 1752 | 1735 | 1735 | 1740 | 1733 |
| steine15 | 2784 | 2826 | 2792 | 2784 | 2784 | 2784 | 2784 |
| steine16 | 15 | 19 | 15 | 15 | 15 | 15 | 15 |
| steine17 | 25 | 31 | 26 | 25 | 25 | 25 | 25 |
| steine18 | 564 | 627 | 608 | 584 | 584 | 595 | 572 |
| steine19 | 758 | 801 | 788 | 770 | 778 | 778 | 761 |
| steine20 | 1342 | 1359 | 1349 | 1343 | 1343 | 1352 | 1342 |

Trong số các thuật toán trên, thuật toán metaheuristic dạng quần thể cho lời giải chất lượng tốt hơn so với các thuật toán heuristic và các thuật toán

metaheuristic dạng cá thể. Tuy nhiên, khi thực nghiệm với các đồ thị có kích thước lớn thì các thuật toán metaheuristic dạng cá thể cho thời gian chạy nhanh hơn so với các thuật toán metaheuristic dạng quần thể.

## 1.4. ĐỊNH HƯỚNG ỨNG DỤNG BÀI TOÁN CÂY STEINER NHỎ NHẤT CHO THIẾT KẾ HỆ THỐNG MẠNG

### 1.4.1. Giới thiệu bài toán quy hoạch mạng

Theo [2], vấn đề thiết kế hệ thống mạng có thể đặt ra dưới dạng một trong hai bài toán quy hoạch mạng sau đây.

- ***Bài toán thứ nhất:*** Là bài toán nâng cấp, mở rộng hệ thống mạng đã có. Nội dung của bài toán này bao gồm việc xác định cấu hình, kích cỡ, danh mục chi tiết các thiết bị cần đầu tư, các công trình cần xây dựng nhằm đáp ứng các yêu cầu mở rộng sản xuất và kinh doanh của tổ chức, doanh nghiệp trong một khoảng thời gian theo kế hoạch đã định trước.

- ***Bài toán thứ hai:*** Là bài toán xây dựng mới hệ thống mạng. Bài toán này được đặt ra cho các khu vực đô thị được xây mới hay các khu công nghiệp, ….

Trong thực tế, hai bài toán quy hoạch mạng nêu trên thường được giải quyết theo các bước sau đây.

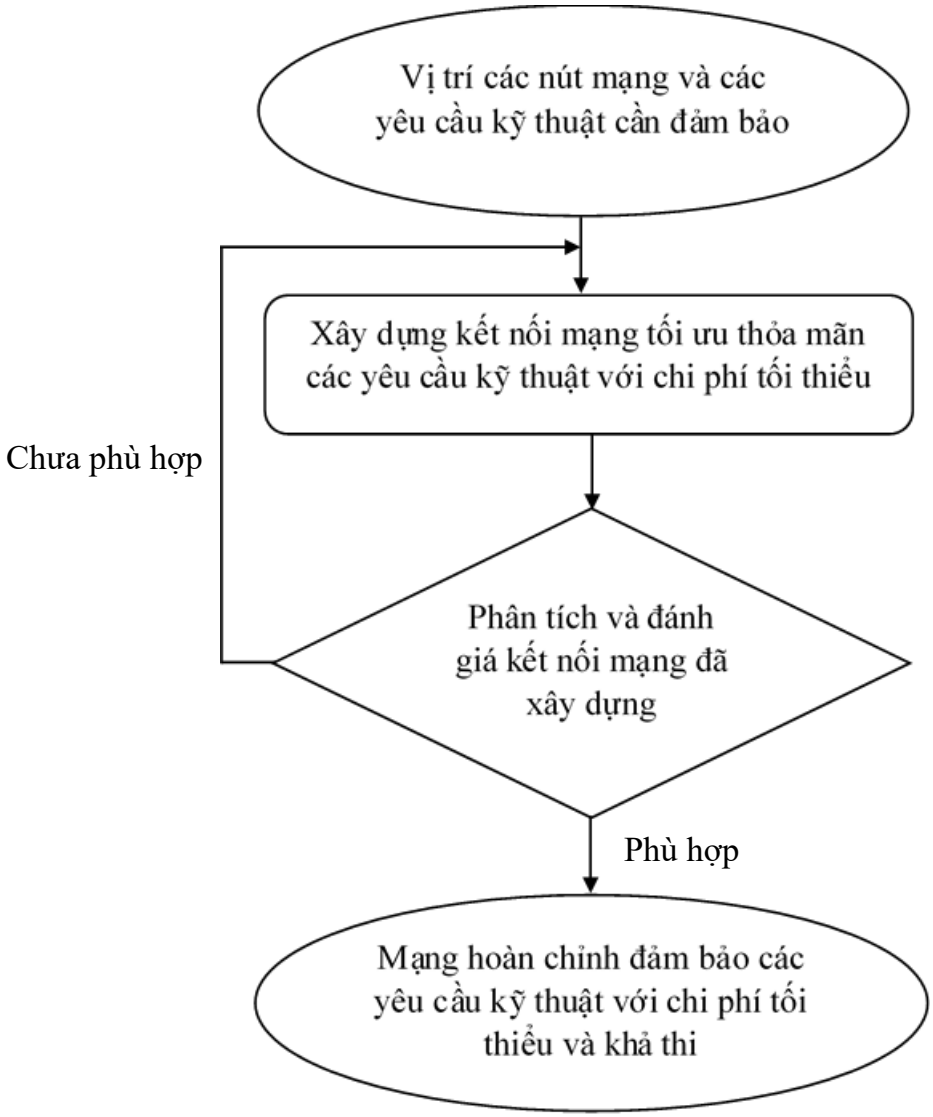
- ***Bước thứ nhất:*** Trong quy trình quy hoạch mạng là khảo sát thực địa vùng mạng cần xây dựng để xác định được các vị trí nút mạng và các yêu cầu cầu kỹ thuật cần thiết bao gồm lưu lượng đường truyền giữa các nút mạng, các ràng buộc chất lượng mạng như băng thông, độ trễ, khả năng hồi phục, ... Các kết quả ở bước này cũng chính là dữ liệu đầu vào cho bài toán quy hoạch mạng.

- ***Bước thứ hai:*** Trong quy trình quy hoạch mạng sẽ căn cứ dữ liệu đầu vào, tiến hành xây dựng kết nối mạng tối ưu thỏa mãn các yêu cầu kỹ thuật. Nội dung của bước này là xây dựng và sử dụng các thuật toán thiết kế kết nối (topology)

mạng tối ưu. Kết quả của bước này đưa ra kết nối mạng đảm bảo các yêu cầu kỹ thuật với chi phí tối thiểu.

- ***Bước thứ ba:*** Trong quy trình quy hoạch mạng là thực hiện phân tích và đánh giá kết quả kết nối mạng đưa ra trong bước thứ hai. Trên cơ sở phân tích và so sánh các giải pháp thiết kế mạng để lựa chọn giải pháp tốt nhất và khả thi. Tùy thuộc vào kết quả phân tích, có thể tiến hành lại bước thứ hai để tìm ra một giải pháp khả thi hơn. Đó cũng chính là kết quả đầu ra cho bài toán quy hoạch mạng.

Quy trình giải quyết bài toán quy hoạch mạng có thể biểu diễn dưới dạng sơ đồ khối như trong Hình 1.3.




Hình 1.3. Sơ đồ khối giải quyết bài toán quy hoạch mạng

Trong quá trình giải quyết bài toán quy hoạch mạng, quá trình xây dựng kết nối mạng tối ưu có vai trò quan trọng nhất. Quá trình này liên quan đến một loạt các thuật toán, các quá trình lặp lại nhằm tìm ra một kết nối mạng phù hợp nhất với các yêu cầu kỹ thuật, chi phí tối thiểu và khả thi trong triển khai thực tế.

### 1.4.2. Định hướng ứng dụng các thuật toán tìm Cây Steiner nhỏ nhất trong thiết kế hệ thống mạng

Hệ thống mạng bao gồm mạng máy tính nói chung và mạng viễn thông nói riêng có thể được mô hình hóa bởi đồ thị vô hướng, có trọng số *G = (V(G), E(G), W(E))*. Trong đó, *V(G)* là tập hợp các nút mạng, *E(G)* là tập hợp các kết nối giữa các nút mạng và *W(E)* là tập hợp trọng số trên các kết nối được tổng hợp từ các yêu cầu kỹ thuật cần đảm bảo của hệ thống mạng [2][32][48].

Gọi *L* là tập các đỉnh thuộc *G* bểu diễn cho các nút mạng mới bổ sung để mở rộng mạng. Bài toán quy hoạch, thiết kế mạng trở thành bài toán lựa chọn các kết nối giữa các đỉnh thuộc *L* và các kết nối giữa các đỉnh của *L* với các đỉnh không thuộc *L* sao cho chi phí nhỏ nhất. Trong trường hợp cần thiết kế hệ thống mạng mới thì tập hợp *L* trùng với *V(G)*.

Như vậy, bài toán quy hoạch, thiết kế hệ thống mạng có thể chuyển về bài toán tìm Cây Steiner nhỏ nhất trên tập *L* của đồ thị *G*. Đây là bài toán thuộc lớp *NP-hard* và là bài toán thường được sử dụng trong thiết kế hệ thống mạng có thể triển khai trong thực tiễn [9][14][23][24][48].

Khi xem xét các ứng dụng của bài toán *SMT* trong thiết kế hệ thống mạng, cần xác định rõ ta đang quan tâm đến góc độ của bài toán thiết kế hay góc độ của bài toán thực thi: Nếu ở góc độ của bài toán thiết kế thì ưu tiên về chất lượng lời giải hơn; còn nếu ở góc độ của bài toán thực thi thì ưu tiên về thời gian chạy hơn.

Sau đây là một số mô hình định hướng ứng dụng bài toán *SMT* trong việc thiết kế hệ thống mạng [34][73][85][62].

- **Mô hình 1:** ***Thiết kế hệ thống mạng cục bộ***

Xét một hệ thống mạng cục bộ có dây truyền thống trong một cao ốc. Bộ khung vị trí các điểm phát mới cho WLAN có thể tận dụng hệ thống LAN sẵn có, nhưng không nhất thiết phải thực hiện tương tự. Giả sử đã biết tập các điểm phát được nối với nhau qua LAN và chi phí để kết nối một điểm phát đến một điểm kết nối định nghĩa bởi $k_{ij}$.

Trong trường hợp này, tập $V$ gồm tất cả các điểm phát và mọi điểm kết nối đến hệ thống LAN sẵn có. Vì chỉ có các điểm phát mới giao tiếp với nhau, nên $S$ chỉ gồm những điểm phát và $H$ là một đồ thị đầy đủ. Chi phí kết nối giữa hai điểm kết nối $i, j \in V \setminus S$ được định nghĩa là $k_{i,j} = 0$. Ứng dụng được biết đến như là bài toán Cây Steiner có chi phí nhỏ nhất. Tập con $U \subset V$ được gọi là tập các đỉnh cuối (terminal) liên thông với nhau, trong khi số đỉnh còn lại $V \setminus U$ được xem như là các tải trung tâm (hub) giúp giảm chi phí kết nối.

**- Mô hình 2:** ***Cung cấp dịch vụ mạng riêng ảo***

Một nhà cung ứng dịch vụ Internet (ISP) cung cấp dịch vụ mạng ảo cá nhân (*VPN*) cho các khách hàng. Mỗi *VPN* là một mạng máy tính sử dụng những mạch ảo tham gia vào một mạng lớn hơn. Với mỗi khách hàng, một *VPN* được thiết lập cho phép giao tiếp mọi nơi mà khách hàng đó đã đăng kí. Nhà cung cấp Internet thường thuê băng thông truyền từ nhà mạng viễn thông. Chi phí tiết kiệm có được bằng cách khai thác thông tin tổ hợp yêu cầu từ nhiều khách hàng để có kế hoạch triển khai tối ưu.

Đây là trường hợp suy rộng của bài toán Cây Steiner. Nếu tất cả các đỉnh trong $U$ không liên thông thì cấu trúc tôpô không nhất thiết phải là cây khung $U$. Với mỗi thành phần $H$, cấu trúc tôpô tối ưu sẽ liên thông, nhưng không cần phải nối các đỉnh ở các thành phần khác nhau. Tuy nhiên, trong vài trường hợp, việc nối một đỉnh của thành phần này với một đỉnh của thành phần khác lại tỏ ra hiệu quả hơn. Trong thực tế, cấu trúc tôpo kết quả thường sẽ là một rừng (forest) với số lượng phần tử nhiều nhất (các cây).

**- Mô hình 3:** ***Bài toán truyền thông đa hướng***

Cho một mạng máy tính được biểu diễn bởi một đồ thị vô hướng $G = (V(G), E(G))$, một tập con $T$ các đỉnh của $V(G)$ được gọi là nhóm truyền thông và một điểm $s$ được gọi là một máy nguồn.

Bài toán yêu cầu tìm một topology nối tất cả các máy trong nhóm truyền thông $T \cup s$ sao cho tổng số các kênh nối (trong một số trường hợp có thể là tổng độ dài các kênh nối) cần sử dụng là nhỏ nhất mà vẫn thỏa mãn các ràng buộc của mạng như tốc độ đường truyền, dung lượng băng thông, .....

- **Mô hình 4:** ***Thiết kế vi mạch VLSI***

Trong bước thiết kế vật lý của vi mạch VLSI, pha đi dây bao gồm việc xác định sơ đồ nối dây cho các điểm nối của từng khối hay từng cổng trên Chip. Một mạng trên Chip là tập hợp các điểm cần phải nối với nhau, thường bao gồm một điểm nguồn và nhiều điểm đích. Trong việc đi dây, sơ đồ kết nối được đặt tả cho một số lượng lớn các mạng. Kết nối thường được thực hiện trên một mạng cùng lúc. Mỗi phương án kết nối cho một mạng đơn là một Cây Steiner phủ các điểm nối. Nhiều yêu cầu tối ưu đặt ra cho việc nối mạng phụ thuộc vào chức năng của từng mạng. Việc thiết kế vi mạch VLSI, các đường nối chỉ là đường ngang dọc trên bảng mạch. Bài toán tối ưu chiều dài dây nối được đưa về một biến thể của bài toán Cây Steiner, đó là bài toán Cây Steiner thẳng góc trên đồ thị lưới.

- **Mô hình 5:** ***Bài toán kết nối nhóm***

Cho một đồ thị vô hướng $G = (V(G), E(G))$ và tập các đỉnh $T_1, T_2, ...., T_n \subset V$. Bài toán yêu cầu tìm một đồ thị con $G'$ của $G$ thỏa mãn hai điều kiện sau:

- Mỗi tập con $T_i$ , $i = 1..n$ phải có ít nhất một điểm trong $G$.

- Độ dài của đồ thị $G'$ phải được cực tiểu.

- **Mô hình 6:** ***Bài toán Steiner nhiều pha***

Hơi khác so với bài toán kết nối nhóm, bài toán Steiner nhiều pha yêu cầu phải chứa được tất cả các tập đỉnh $T_1, T_2, ...., T_n$.

Cho một đồ thị $G = (V(G), E(G))$ và tập các đỉnh $T_1, T_2, ...., T_n \subset V$. Gọi $S_1, S_2, ...., S_n$ là các đồ thị con liên thông của $T_1, T_2, ...., T_n$. Bài toán đặt ra yêu cầu tìm một đồ thị có giá trị cực tiểu và chứa tất cả các đồ thị liên thông.

- **Mô hình 7:** ***Bài toán mạng Steiner nhiều pha***

Cho một đồ thị $G = (V(G), E(G))$, và các tập con của $V$: $X_1, Y_1, X_2, Y_2, ....., X_n, Y_n$ với $X_i \cap Y_i = \varnothing$.

Bài toán yêu cầu tìm một đồ thị con $G'$ của $G$ thỏa mãn hai điều kiện:

- Với mỗi $i = 1..n$, $G'$ chứa một Cây Steiner $S_i$ của $X_i$.
- Các điểm Steiner của $S_i$ chỉ được lấy từ $Y_i$.

- **Mô hình 8:** ***Bài toán phát sinh loài***

Sự phát sinh loài được biểu diễn bằng một cây tiến hóa. Nghiên cứu xây dựng cây phát sinh loài là một trong những bài toán tính toán cơ bản trong sinh học. Ta quan tâm ở đây tới sự phân loài được mô tả bởi các tính trạng mà chúng thể hiện ra từ tập các đặc tính. Dữ liệu trên được mô tả dưới dạng bảng nhị phân, phần tử $(i, j)$ nhận giá trị 0 hoặc 1 tương ứng với loài $i$ không có tính trạng $j$ hoặc có tính trạng $j$. Sự khác biệt giữa hai loài được đo bằng khoảng cách Hamming giữa hai vector tính trạng.

Bài toán đặt ra là xây dựng một cây tiến hóa sao cho sự khác biệt giữa các loài đạt giá trị nhỏ nhất chính là giải bài toán Cây Steiner nhỏ nhất. Các điểm kết thúc là các vector tính trạng của loài, các nút trung gian bản thân cũng là các vector nhị phân, chúng nằm trên cạnh nối từ gốc đến lá. Trọng số của các cạnh đo bằng khoảng cách Hamming.

## 1.5. LỰA CHỌN DỮ LIỆU THỰC NGHIỆM

Để đánh giá so sánh các thuật toán được phát triển bởi luận án với các thuật toán khác, cần phải lựa chọn các bộ dữ liệu tiêu chuẩn đã được sử dụng và công nhận rộng rãi trong cộng đồng nghiên cứu.

Hầu hết các đồ thị gặp trong thực tế ứng dụng là các đồ thị thưa (hệ thống mạng máy tính, hệ thống mạng truyền thông, hệ thống mạng đường giao thông, hệ thống mạng điện, mạng đường ống cung cấp nước, hệ thống mạng thủy lợi,…). Vì vậy, luận án sử dụng 78 bộ dữ liệu là các đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn dùng cho bài toán Cây Steiner tại địa chỉ URL: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html [40]; các đồ thị này có tối đa 2500 đỉnh (các bộ dữ liệu chuẩn này được trình bày chi tiết trong Phụ lục 1, phía sau).

Do cách tiếp cận bài toán *SMT* bằng các thuật toán heuristic có ưu điểm hơn cách tiếp cận bằng các thuật toán metaheuristic đối với các đồ thị có kích thước lớn, ít nhất cũng về mặt thời gian tính; nên luận án đề xuất thêm 80 bộ dữ liệu là các đồ thị thưa kích thước lớn lên đến 100000 đỉnh và đặt tên là: *steinf, steing, steinh, steini* (các bộ dữ liệu mở rộng kích thước lớn được trình bày chi tiết trong Phụ lục 2, phía sau).

Các đồ thị thưa kích thước lớn được sinh trước hết từ một cây khung ngẫu nhiên đi qua tất cả *n* đỉnh của đồ thị; sau đó sinh ngẫu nhiên thêm một số cạnh đến khi đồ thị có đủ *m* cạnh; trọng số các cạnh cũng được sinh ngẫu nhiên.

Trong luận án, toàn bộ hệ thống dữ liệu thực nghiệm chuẩn và mở rộng được đề cập và sử dụng trong Chương 2 và Chương 3.

## 1.6. KẾT LUẬN CHƯƠNG 1

Trên cơ sở tìm hiểu tổng quan về các vấn đề nghiên cứu, Chương 1 của luận án đã đạt được một số kết quả như sau:

- Nghiên cứu tổng quan về bài toán Cây Steiner nhỏ nhất và các vấn đề liên quan.

- Tìm hiểu hướng tiếp cận bằng các thuật toán heuristic, metaheuristic giải bài toán Cây Steiner nhỏ nhất; hệ thống dữ liệu thực nghiệm chuẩn cho bài toán Cây Steiner nhỏ nhất và đề xuất thêm các bộ dữ liệu mở rộng là các đồ thị thưa có kích thước lớn.

- Khảo sát đánh giá một số thuật toán tiêu biểu giải bài toán Cây Steiner nhỏ nhất. Qua đó cho thấy, hướng tiếp cận giải gần đúng bài toán Cây Steiner nhỏ nhất đạt hiệu quả khả quan là các thuật toán metaheuristic; định hướng ứng dụng kết quả nghiên cứu bài toán Cây Steiner nhỏ nhất cho việc thiết kế hệ thống mạng.

- Từ kết quả Chương 1, luận án đặt ra hai vấn đề cần giải quyết trong các chương tiếp theo.

- ***Vấn đề 1:*** Đề xuất phát triển mới, cải tiến thuật toán heuristic giải bài toán Cây Steiner nhỏ nhất trong trường hợp đồ thị thưa và đồ thị thưa kích thước lớn (*trình bày trong Chương 2*).

- ***Vấn đề 2:*** Đề xuất phát triển thuật toán metaheuristic giải bài toán Cây Steiner nhỏ nhất trong trường hợp đồ thị thưa (*trình bày trong Chương 3*).

# Chương 2. ĐỀ XUẤT THUẬT TOÁN HEURISTIC GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

*Chương 2 nghiên cứu đề xuất hai thuật toán heuristic mới giải bài toán Cây Steiner nhỏ nhất gồm: Thuật toán SPT-Steiner (dựa trên ý tưởng bài toán tìm cây đường đi ngắn nhất) và thuật toán PD-Steiner (dựa trên ý tưởng kết hợp tìm cây khung nhỏ nhất với tìm đường đi ngắn nhất); đồng thời đề xuất hai thuật toán heuristic cải tiến: i-SPT-Steiner và i-PD-Steiner (dựa trên ý tưởng thay thế thuật toán Dijkstra bằng thuật toán Dial – một biến thể của thuật toán Dijkstra trong việc tìm đường đi ngắn nhất) để giải bài toán Cây Steiner nhỏ nhất trong trường hợp đồ thị thưa kích thước lớn. Chương này được tổng hợp từ các công trình [CT5] và [CT6] trong danh mục các công trình nghiên cứu của tác giả.*

## 2.1. GIỚI THIỆU HƯỚNG TIẾP CẬN HEURISTIC GIẢI BÀI TOÁN SMT

Trong các hướng tiếp cận giải bài toán *SMT* thì các thuật toán heuristic có ưu điểm là cho chất lượng lời giải chấp nhận được với thời gian chạy nhanh hơn so với cách tiếp cận theo hướng các thuật toán metaheuristic. Đây là cách tiếp cận khả quan nhất hiện nay khi giải quyết bài toán *SMT* trong trường hợp đồ thị có kích thước lớn.

Một số thuật toán heuristic điển hình giải bài toán *SMT* như: Thuật toán Distance network heuristic của Kou, Markowsky và Berman; thuật toán *MST*-Steiner của Bang Ye Wu và Kun-Mao Chao,...

Tiếp theo trong chương này, luận án đề xuất hai thuật toán heuristic mới dựa trên ý tưởng của thuật toán tìm đường đi ngắn nhất và tìm cây khung nhỏ nhất để giải bài toán *SMT*. Đồng thời, luận án cũng đề xuất hai heuristic cải tiến giải bài toán *SMT* trong trường hợp đồ thị thưa kích thước lớn lên đến 100000 đỉnh. Các thuật toán heuristic đề xuất mới và cải tiến được cài đặt thực nghiệm và so sánh hiệu quả với thuật toán heuristic tốt hiện biết *MST*-Steiner [14].

## 2.2. THUẬT TOÁN MST-STEINER

Thuật toán *MST*-Steiner được nhóm tác giả Bang Ye Wu và Kun-Mao Chao trình bày chi tiết trong công trình [14].

Trong mục này, luận án trình bày lại thuật toán heuristic *MST*-Steiner của Bang Ye Wu và Kun-Mao Chao. Thuật toán này có dữ liệu đầu vào là đồ thị $G = (V(G), E(G))$, tập đỉnh terminal $L \subseteq V(G)$; và dữ liệu đầu ra là chi phí của Cây Steiner *T*.

**Algorithm 2.1: *MST*-Steiner**

**Input**: Đồ thị $G = (V(G), E(G))$, tập đỉnh terminal $L \subseteq V(G)$;
**Output**: Cây Steiner nhỏ nhất $T$;

1. Xây dựng đồ thị đầy đủ $G_L$ của tập *terminal* $L$; trong đó trọng số của các cạnh là chiều dài của đường đi ngắn nhất nối hai đầu mút của cạnh đó;
2. Tìm một cây khung nhỏ nhất $T_L$ của $G_L$ (chẳng hạn sử dụng thuật toán Kruskal hoặc Prim);
3. $T = \emptyset$;
4. **for** (với mỗi cạnh $e = (u, v) \in E(T_L)$) **do**
5. Tìm một đường đi ngắn nhất $P$ từ đỉnh $u$ đến đỉnh $v$ trên đồ thị $G$;
6. **if** $P$ chứa ít hơn hai đỉnh trong $T$ **then**
7. Chèn $P$ vào $T$;
8. **else**
9. Cho $p_i$ và $p_j$ là đỉnh đầu, đỉnh cuối đã có trong $T$; chèn đường đi từ $u$ đến $p_i$ và từ $p_j$ đến $v$ vào $T$;
10. **end for**
11. **return** $T$;

Đánh giá thuật toán *MST*-Steiner có độ phức tạp thời gian tính là: $O(n|L|^2)$ [14].

## 2.3. THUẬT TOÁN SPT-STEINER

Tiếp theo, luận án trình bày một số khái niệm liên quan đến bài toán cây đường đi ngắn nhất (Shortest Path Tree - *SPT*) và đề xuất thuật toán heuristic giải bài toán *SMT* dựa trên ý tưởng tìm cây đường đi ngắn nhất và đặt tên cho thuật toán này là *SPT*-Steiner.

**Định nghĩa 2.1.** Cây đường đi ngắn nhất

Cho đồ thị *G*, cây đường đi ngắn nhất có gốc tại đỉnh *s* là cây khung của *G* với tập cạnh là các cạnh trên các đường đi ngắn nhất xuất phát từ đỉnh *s* đến các đỉnh còn lại của *G* [14].

Cây đường đi ngắn nhất có gốc tại đỉnh *s* của đồ thị *G* có thể tìm được nhờ áp dụng thuật toán Dijkstra để tìm đường đi ngắn nhất từ đỉnh *s* đến tất cả các đỉnh còn lại.

---

**Algorithm 2.2: *SPT*-Steiner (Shortest Path Tree-Steiner)**

---

**Input**: Đồ thị $G = (V(G), E(G))$, tập đỉnh terminal $L \subseteq V(G)$;

**Output**: Cây Steiner nhỏ nhất $T$;

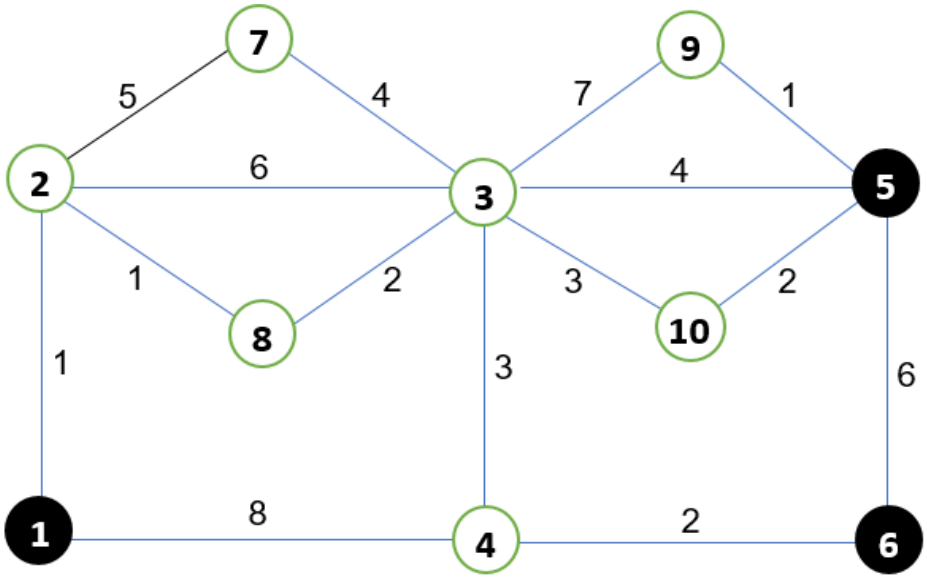
1. Tìm các cây đường đi ngắn nhất xuất phát tại các đỉnh thuộc tập $V(G)$: $SPT_1$, $SPT_2$, …, $SPT_n$
2. **for** ($T \in \{SPT_1, SPT_2, \ldots, SPT_n\}$) **do**
3. Với mỗi đỉnh treo $u \in T$, nếu $u \notin L$ thì xóa cạnh chứa đỉnh $u$ khỏi $E(T)$, xóa đỉnh $u$ trong $V(T)$ và cập nhật bậc của đỉnh kề với đỉnh $u$ trong $T$.
4. Lặp lại bước 3 đến khi $T$ không còn thay đổi (bước này gọi là bước xóa các cạnh dư thừa); sau bước này thu được các $SPT_i$, tương ứng với các $SPT_i$.
5. **end for**
6. Tìm một cây $SPT_i$, có tổng trọng số các cạnh nhỏ nhất là $SPT_{min}$;
7. **return** $SPT_{min}$;

Đánh giá độ phức tạp thời gian tính cho thuật toán *SPT*-Steiner

Do thuật toán Dijkstra sử dụng cấu trúc dữ liệu heap có độ phức tạp thời gian tính là $O(m\log n)$ nên bước 1 có độ phức tạp $O(mn\log n)$; bước 2 có độ phức tạp $O(n^2)$; bước 3 có độ phức tạp $O(n)$. Vậy thuật toán *SPT*-Steiner có độ phức tạp thời gian tính là: $O(mn\log n)$.

**Ví dụ 2.1**: Minh họa từng bước thực hiện thuật toán *SPT*-Steiner

Cho đồ thị $G$ có 10 đỉnh và 15 cạnh như Hình 2.1; trong đó tập $L = \{1, 5, 6\}$.

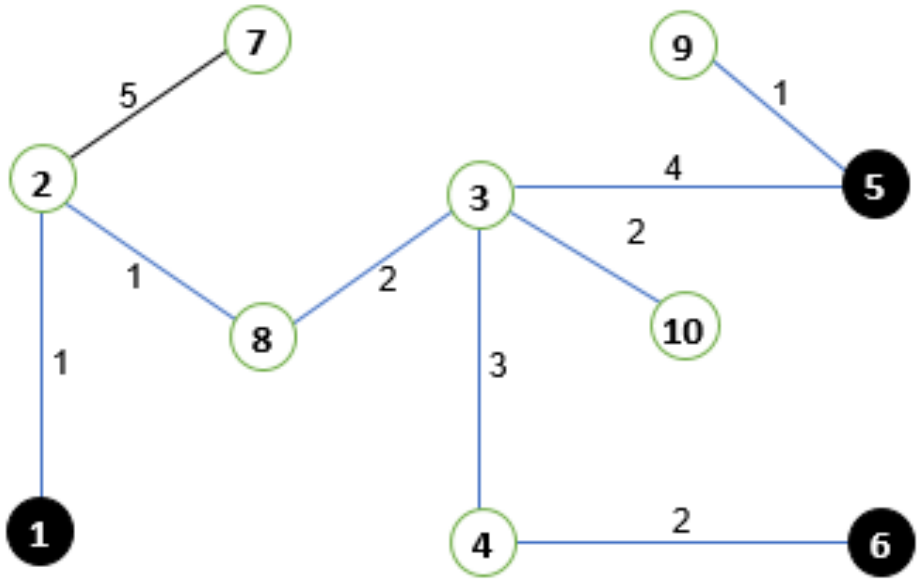



Hình 2.1. Đồ thị vô hướng liên thông có trọng số $G$

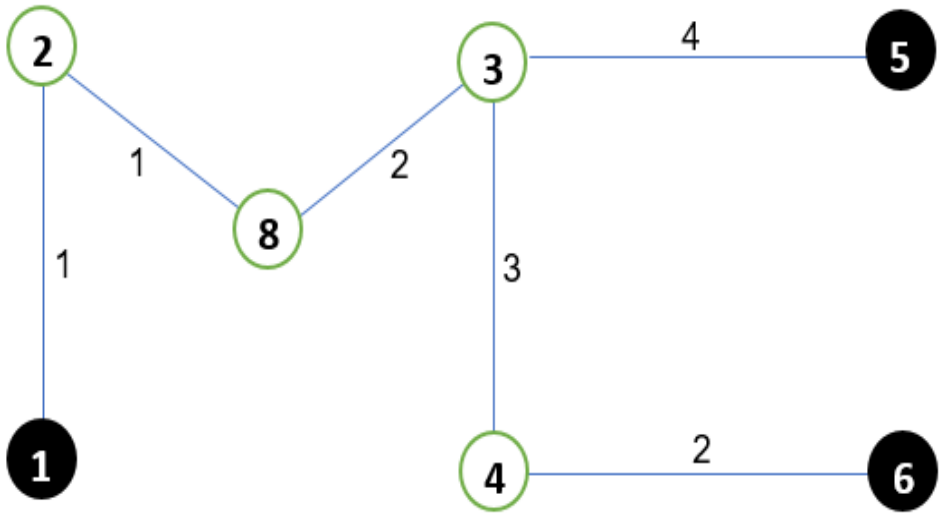
Theo định nghĩa 2.1 và thuật toán *SPT*-Steiner, ta xây dựng các cây đường đi ngắn nhất có gốc tại các đỉnh **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**. Sau khi xóa các cạnh dư thừa, ta có các Cây Steiner có chi phí lần lượt là: 13, 13, 13, 13, 14, 15, 19, 13, 14, 15.

Để ngắn gọn, luận án chỉ minh họa việc xây dựng các Cây Steiner với gốc xuất phát thuộc tập *terminal L*.

Ta xét Hình 2.1, đầu tiên ta tìm cây đường đi ngắn nhất có gốc tại đỉnh **1**; ta thu được cây đường đi ngắn nhất như Hình 2.2; tiếp theo ta loại bỏ các cạnh dư thừa (**2**, **7**), (**5**, **9**), (**3**, **10**). Kết quả thu được Cây Steiner như Hình 2.3 với chi phí là 13.

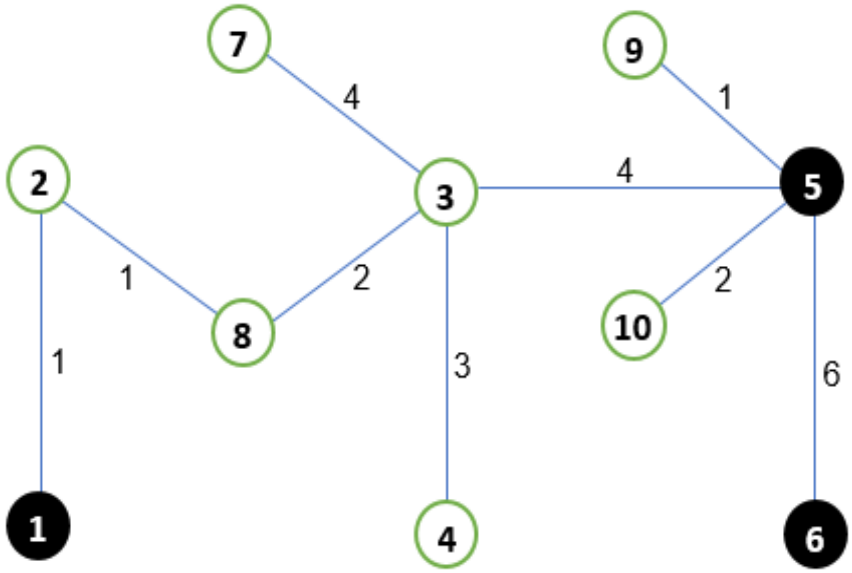



Hình 2.2. Cây đường đi ngắn nhất có gốc tại đỉnh 1

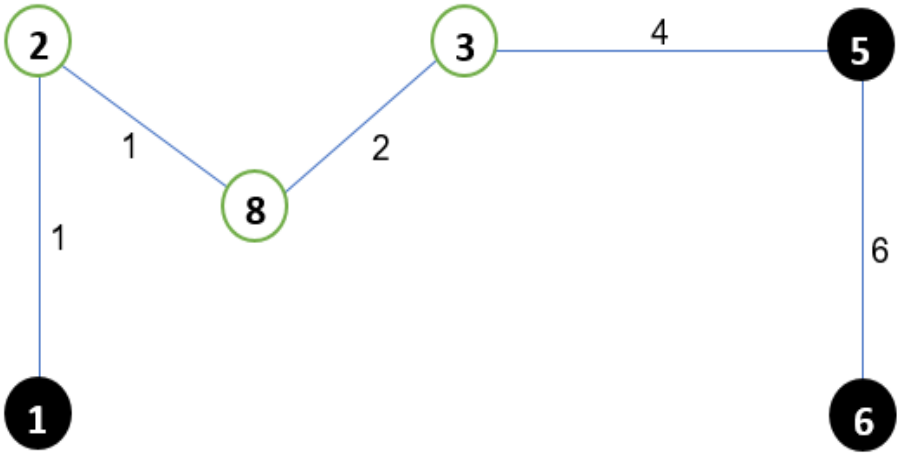



Hình 2.3. Cây Steiner sau khi đã xóa cạnh dư thừa

Tương tự, từ Hình 2.1 ta tìm cây đường đi ngắn nhất có gốc tại đỉnh **5**; ta thu được cây đường đi ngắn nhất như Hình 2.4; tiếp theo ta loại bỏ các cạnh dư thừa (**3**, **7**), (**5**, **9**), (**5**, **10**), (**3**, **4**). Kết quả thu được Cây Steiner như Hình 2.5 với chi phí là 14.

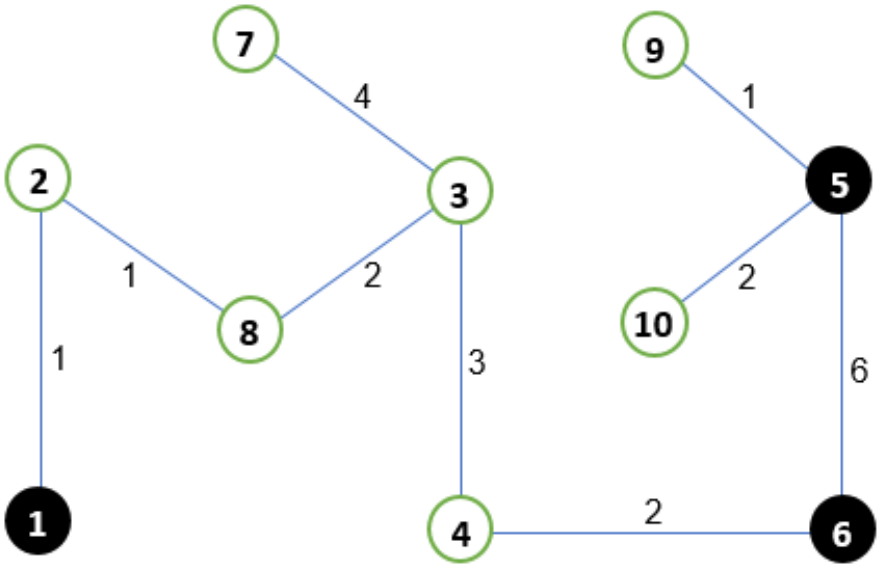



Hình 2.4. Cây đường đi ngắn nhất có gốc tại đỉnh 5

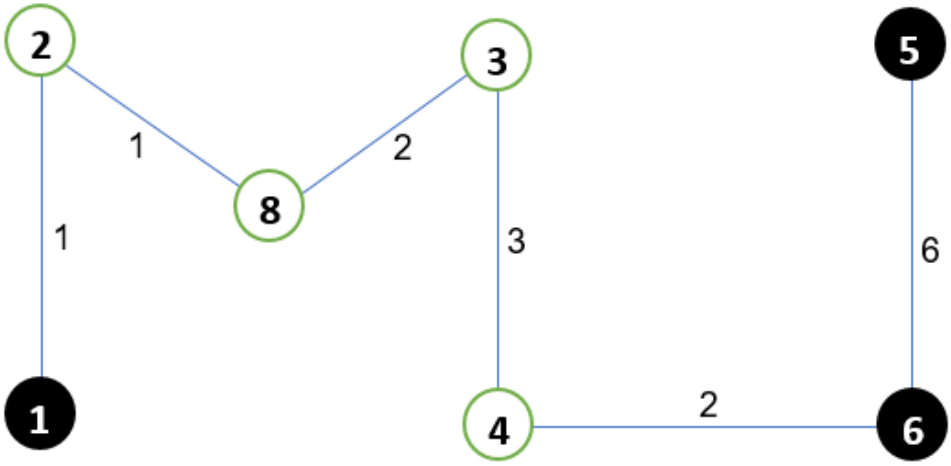



Hình 2.5. Cây Steiner sau khi đã xóa cạnh dư thừa

Tương tự, từ Hình 2.1 ta tìm cây đường đi ngắn nhất có gốc tại đỉnh **6**; ta thu được cây đường đi ngắn nhất ở Hình 2.6; tiếp theo ta loại bỏ các cạnh dư thừa (**3**,**7**), (**5**, **9**), (**5**, **10**). Kết quả thu được Cây Steiner như Hình 2.7 với chi phí là 15.




Hình 2.6. Cây đường đi ngắn nhất có gốc tại đỉnh 6




Hình 2.7. Cây Steiner sau khi đã xóa cạnh dư thừa

Và theo thuật toán *SPT*-Steiner trên, Cây Steiner kết quả tìm được có chi phí là 13, như Hình 2.3.

## 2.4. THUẬT TOÁN PD-STEINER

Mục này, luận án trình bày đề xuất thuật toán heuristic để giải bài toán *SMT* dựa trên ý tưởng tìm cây khung nhỏ nhất kết hợp với tìm đường đi ngắn nhất và đặt tên cho thuật toán này là *PD*-Steiner.

**Algorithm 2.3: *PD*-Steiner (Prim + Dijkstra-Steiner)**

**Input**: Đồ thị *G = (V(G), E(G))*, tập đỉnh terminal $L \subseteq V(G)$;
**Output**: Cây Steiner nhỏ nhất *T;*

1. Tìm các đường đi ngắn nhất từ các đỉnh thuộc tập *L* đến các đỉnh thuộc tập *V*;
2. Chọn một đỉnh $u \in L$; đặt $T = \{u\}$;
3. **while** (*T* chưa chứa tất cả các đỉnh thuộc tập *L*) **do**
4. Từ mỗi đỉnh $v \in L$ và *v* chưa thuộc *T*, tìm các đường đi ngắn nhất từ đỉnh *v* đến các đỉnh $z \in T$;
5. Chọn ra một đường đi ngắn nhất *P*;
6. Kết nạp các đỉnh và các cạnh trên đường đi *P* vào cây *T*;
7. **end while**
8. **return** *T*;

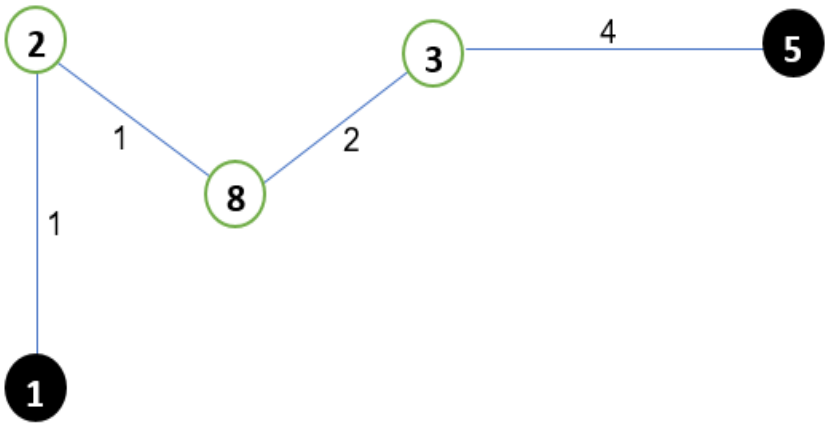
Đánh giá độ phức tạp thời gian tính cho thuật toán *PD*-Steiner

Bước 1 có độ phức tạp $O(|L|m\log n)$; bước 2 có độ phức tạp $O(1)$; vòng lặp ở bước 3 có độ phức tạp $O(|L|n^2)$. Vậy thuật toán *PD*-Steiner có độ phức tạp thời gian tính là: $O(|L|n^2)$.

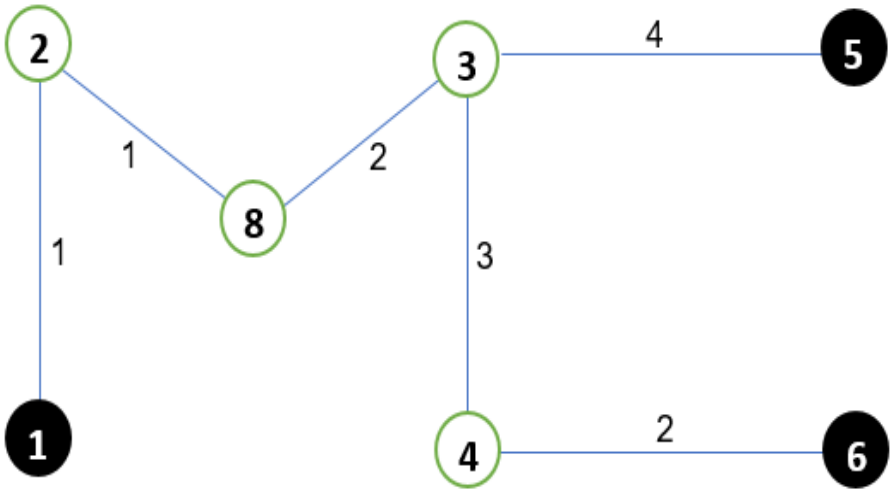
**Ví dụ 2.2**: Minh họa từng bước thực hiện thuật toán *PD*-Steiner

Ta xét Hình 2.1, khoảng cách ngắn nhất từ các đỉnh thuộc tập *L* = **{1, 5, 6}** đến các đỉnh thuộc tập *V(G)* lần lượt là:

Đầu tiên chọn *T* = {**1**}, đường đi ngắn nhất từ đỉnh **5** đến đỉnh **1** như Hình 2.8; tiếp theo từ việc tìm đường đi ngắn nhất từ đỉnh **6** đến các đỉnh **1, 2, 3, 5, 8** và ta chọn đường đi ngắn nhất từ đỉnh **6** đến đỉnh **3**; kết quả thu được Cây Steiner như Hình 2.9 với chi phí là 13.




Hình 2.8. Đường đi ngắn nhất từ đỉnh **5** đến đỉnh **1**




Hình 2.9. Đường đi ngắn nhất từ đỉnh **6** đến đỉnh **3**

Và theo thuật toán *PD*-Steiner trên, Cây Steiner kết quả tìm được có chi phí là 13, như Hình 2.9.

## 2.5. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Trong phần này, luận án mô tả chi tiết việc thực nghiệm các thuật toán *MST*-Steiner [14], *SPT*-Steiner và *PD*-Steiner; đồng thời đưa ra một số đánh giá về kết quả đạt được. Hệ thống dữ liệu thực nghiệm cho các thuật toán đã được đề cập trong Chương 1, luận án sử dụng 78 bộ dữ liệu là các đồ thị thưa trong hệ thống dữ

liệu thực nghiệm chuẩn dùng cho bài toán cây Steiner tại địa chỉ URL http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html [40]; trong đó: Nhóm *steinb* có 18 đồ thị, nhóm *steinc* có 20 đồ thị, nhóm *steind* có 20 đồ thị, nhóm *steine* có 20 đồ thị (các đồ thị này có tối đa 2500 đỉnh). Ngoài ra, thực nghiệm đề xuất sử dụng thêm 20 bộ dữ liệu là các đồ thị nhóm *steinf*, đây là các đồ thị thưa kích thước lớn 10000 đỉnh. Tổng cộng hệ thống dữ liệu thực nghiệm cho các thuật toán này là 98 bộ dữ liệu.

### 2.5.1. Môi trường thực nghiệm

Các thuật toán đề xuất được cài đặt bằng ngôn ngữ C++, sử dụng môi trường DEV C++ 5.9.2; được thực nghiệm trên một máy chủ ảo, Hệ điều hành Windows server 2008 R2 Enterprise, 64bit, Intel(R) Xeon (R) CPU E5-2660 0 @ 2.20 GHz, RAM 4GB.

### 2.5.2. Kết quả thực nghiệm

Kết quả thực nghiệm các thuật toán được ghi nhận ở Bảng 2.1, 2.2, 2.3, 2.4, 2.5. Các bảng có cấu trúc như sau: Cột đầu tiên (Test) là tên các bộ dữ liệu trong hệ thống dữ liệu thực nghiệm; số đỉnh ($n$), số cạnh ($m$) và số đỉnh thuộc tập *terminal* ($|L|$) của từng đồ thị; các cột tiếp theo ghi nhận giá trị chi phí Cây Steiner tương ứng với từng thuật toán: *MST*-Steiner [14], *SPT*-Steiner và *PD*-Steiner.

Bảng 2.1. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steinb*

| **Test** | ***n*** | ***m*** | **\|*L*\|** | ***MST*-Steiner** | ***SPT*-Steiner** | ***PD*-Steiner** |
|---|---|---|---|---|---|---|
| steinb1.txt | 50 | 63 | 9 | 82 | 82 | 82 |
| steinb2.txt | 50 | 63 | 13 | 90 | 84 | 84 |
| steinb3.txt | 50 | 63 | 25 | 140 | 147 | 138 |
| steinb4.txt | 50 | 100 | 9 | 64 | 59 | 62 |
| steinb5.txt | 50 | 100 | 13 | 64 | 62 | 61 |
| steinb6.txt | 50 | 100 | 25 | 128 | 134 | 126 |
| steinb7.txt | 75 | 94 | 13 | 111 | 111 | 111 |
| steinb8.txt | 75 | 94 | 19 | 104 | 113 | 104 |
| steinb9.txt | 75 | 94 | 38 | 222 | 222 | 220 |
| steinb10.txt | 75 | 150 | 13 | 98 | 90 | 90 |
| steinb11.txt | 75 | 150 | 19 | 91 | 93 | 90 |
| steinb12.txt | 75 | 150 | 38 | 174 | 192 | 174 |
| steinb13.txt | 100 | 125 | 17 | 175 | 172 | 175 |
| steinb14.txt | 100 | 125 | 25 | 237 | 253 | 235 |
| steinb15.txt | 100 | 125 | 50 | 323 | 335 | 318 |
| steinb16.txt | 100 | 200 | 17 | 137 | 138 | 133 |
| steinb17.txt | 100 | 200 | 25 | 134 | 139 | 132 |
| steinb18.txt | 100 | 200 | 50 | 222 | 250 | 222 |

Bảng 2.2. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steinc*

| **Test** | ***n*** | ***m*** | **\|*L*\|** | ***MST*-Steiner** | ***SPT*-Steiner** | ***PD*-Steiner** |
|---|---|---|---|---|---|---|
| steinc1.txt | 500 | 625 | 5 | 88 | 86 | 85 |
| steinc2.txt | 500 | 625 | 10 | 144 | 158 | 144 |
| steinc3.txt | 500 | 625 | 83 | 779 | 843 | 762 |
| steinc4.txt | 500 | 625 | 125 | 1114 | 1193 | 1085 |
| steinc5.txt | 500 | 625 | 250 | 1599 | 1706 | 1583 |
| steinc6.txt | 500 | 1000 | 5 | 60 | 56 | 55 |
| steinc7.txt | 500 | 1000 | 10 | 115 | 103 | 102 |
| steinc8.txt | 500 | 1000 | 83 | 531 | 597 | 516 |
| steinc9.txt | 500 | 1000 | 125 | 728 | 865 | 718 |
| steinc10.txt | 500 | 1000 | 250 | 1117 | 1327 | 1107 |
| steinc11.txt | 500 | 2500 | 5 | 37 | 32 | 34 |
| steinc12.txt | 500 | 2500 | 10 | 49 | 46 | 48 |
| steinc13.txt | 500 | 2500 | 83 | 274 | 322 | 268 |
| steinc14.txt | 500 | 2500 | 125 | 337 | 417 | 332 |
| steinc15.txt | 500 | 2500 | 250 | 571 | 703 | 562 |
| steinc16.txt | 500 | 12500 | 5 | 13 | 12 | 12 |
| steinc17.txt | 500 | 12500 | 10 | 19 | 19 | 20 |
| steinc18.txt | 500 | 12500 | 83 | 125 | 146 | 123 |
| steinc19.txt | 500 | 12500 | 125 | 158 | 195 | 159 |
| steinc20.txt | 500 | 12500 | 250 | 269 | 339 | 268 |

Bảng 2.3. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steind*

| **Test** | ***n*** | ***m*** | **\|*L*\|** | ***MST*-Steiner** | ***SPT*-Steiner** | ***PD*-Steiner** |
|---|---|---|---|---|---|---|
| steind1.txt | 1000 | 1250 | 5 | 107 | 107 | 107 |
| steind2.txt | 1000 | 1250 | 10 | 237 | 228 | 232 |
| steind3.txt | 1000 | 1250 | 167 | 1636 | 1771 | 1593 |
| steind4.txt | 1000 | 1250 | 250 | 2012 | 2174 | 1957 |
| steind5.txt | 1000 | 1250 | 500 | 3310 | 3511 | 3270 |
| steind6.txt | 1000 | 2000 | 5 | 74 | 70 | 75 |
| steind7.txt | 1000 | 2000 | 10 | 105 | 111 | 103 |
| steind8.txt | 1000 | 2000 | 167 | 1138 | 1287 | 1104 |
| steind9.txt | 1000 | 2000 | 250 | 1540 | 1773 | 1500 |
| steind10.txt | 1000 | 2000 | 500 | 2163 | 2550 | 2141 |
| steind11.txt | 1000 | 5000 | 5 | 31 | 29 | 31 |
| steind12.txt | 1000 | 5000 | 10 | 43 | 44 | 42 |
| steind13.txt | 1000 | 5000 | 167 | 531 | 643 | 518 |
| steind14.txt | 1000 | 5000 | 250 | 702 | 851 | 691 |
| steind15.txt | 1000 | 5000 | 500 | 1151 | 1437 | 1134 |
| steind16.txt | 1000 | 25000 | 5 | 15 | 13 | 14 |
| steind17.txt | 1000 | 25000 | 10 | 25 | 25 | 23 |
| steind18.txt | 1000 | 25000 | 167 | 251 | 301 | 246 |
| steind19.txt | 1000 | 25000 | 250 | 344 | 424 | 334 |
| steind20.txt | 1000 | 25000 | 500 | 544 | 691 | 542 |

Bảng 2.4. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steine*

| **Test** | ***n*** | ***m*** | **\|*L*\|** | ***MST*-Steiner** | ***SPT*-Steiner** | ***PD*-Steiner** |
|---|---|---|---|---|---|---|
| steine1.txt | 2500 | 3125 | 5 | 125 | 111 | 115 |
| steine2.txt | 2500 | 3125 | 10 | 244 | 214 | 227 |
| steine3.txt | 2500 | 3125 | 417 | 4232 | 4570 | 4118 |
| steine4.txt | 2500 | 3125 | 625 | 5316 | 5675 | 5201 |
| steine5.txt | 2500 | 3125 | 1250 | 8313 | 8976 | 8226 |
| steine6.txt | 2500 | 5000 | 5 | 86 | 73 | 78 |
| steine7.txt | 2500 | 5000 | 10 | 165 | 150 | 159 |
| steine8.txt | 2500 | 5000 | 417 | 2809 | 3254 | 2726 |
| steine9.txt | 2500 | 5000 | 625 | 3809 | 4474 | 3727 |
| steine10.txt | 2500 | 5000 | 1250 | 5745 | 6847 | 5673 |
| steine11.txt | 2500 | 12500 | 5 | 39 | 34 | 38 |
| steine12.txt | 2500 | 12500 | 10 | 73 | 68 | 69 |
| steine13.txt | 2500 | 12500 | 417 | 1370 | 1704 | 1332 |
| steine14.txt | 2500 | 12500 | 625 | 1814 | 2304 | 1778 |
| steine15.txt | 2500 | 12500 | 1250 | 2856 | 3626 | 2819 |
| steine16.txt | 2500 | 62500 | 5 | 17 | 15 | 15 |
| steine17.txt | 2500 | 62500 | 10 | 27 | 27 | 26 |
| steine18.txt | 2500 | 62500 | 417 | 646 | 804 | 639 |
| steine19.txt | 2500 | 62500 | 625 | 809 | 1059 | 806 |
| steine20.txt | 2500 | 62500 | 1250 | 1358 | 1753 | 1357 |

Bảng 2.5. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steinf*

| **Test** | ***n*** | ***m*** | **\|*L*\|** | ***MST*-Steiner** | ***SPT*-Steiner** | ***PD*-Steiner** |
|---|---|---|---|---|---|---|
| steinf1.txt | 10000 | 93750 | 10 | 62 | 58 | 59 |
| steinf2.txt | 10000 | 93750 | 20 | 107 | 109 | 105 |
| steinf3.txt | 10000 | 93750 | 834 | 2292 | 2912 | 2214 |
| steinf4.txt | 10000 | 93750 | 1250 | 3022 | 3865 | 2921 |
| steinf5.txt | 10000 | 93750 | 2500 | 4979 | 6481 | 4841 |
| steinf6.txt | 10000 | 125000 | 10 | 54 | 50 | 50 |
| steinf7.txt | 10000 | 125000 | 20 | 92 | 94 | 86 |
| steinf8.txt | 10000 | 125000 | 834 | 1993 | 2499 | 1907 |
| steinf9.txt | 10000 | 125000 | 1250 | 2729 | 3500 | 2624 |
| steinf10.txt | 10000 | 125000 | 2500 | 4328 | 5641 | 4228 |
| steinf11.txt | 10000 | 156250 | 10 | 37 | 38 | 37 |
| steinf12.txt | 10000 | 156250 | 20 | 82 | 84 | 76 |
| steinf13.txt | 10000 | 156250 | 834 | 1816 | 2313 | 1717 |
| steinf14.txt | 10000 | 156250 | 1250 | 2429 | 3118 | 2339 |
| steinf15.txt | 10000 | 156250 | 2500 | 3962 | 5124 | 3888 |
| steinf16.txt | 10000 | 187500 | 10 | 39 | 38 | 38 |
| steinf17.txt | 10000 | 187500 | 20 | 79 | 75 | 75 |
| steinf18.txt | 10000 | 187500 | 834 | 1660 | 2140 | 1587 |
| steinf19.txt | 10000 | 187500 | 1250 | 2227 | 2909 | 2161 |
| steinf20.txt | 10000 | 187500 | 2500 | 3709 | 4808 | 3612 |

### 2.5.3. Đánh giá kết quả thực nghiệm

***- Chất lượng lời giải***

Phần này nhằm so sánh chất lượng lời giải của các thuật toán *SPT*-Steiner, *PD*-Steiner với thuật toán *MST*-Steiner. Kết quả so sánh được ghi nhận ở Bảng 2.6; Bảng này được tổng hợp từ các Bảng 2.1, 2.2, 2.3, 2.4, 2.5. Nội dung của Bảng 2.6 cho biết số lượng (***SL***) và phần trăm (***%**: tỷ lệ phần trăm tốt hơn, bằng hoặc kém hơn trên tổng số bộ dữ liệu*) tương ứng với số bộ dữ liệu cho chất lượng lời giải tốt hơn (*ghi nhận bởi ký hiệu "<"*) hoặc cho chất lượng lời giải bằng nhau (*ghi nhận bởi ký hiệu "="*) hoặc cho chất lượng lời giải kém hơn (*ghi nhận bởi ký hiệu ">"*) khi so sánh mỗi thuật toán *SPT*-Steiner và *PD*-Steiner với thuật toán *MST*-Steiner [14].

Bảng 2.6. So sánh chất lượng lời giải các thuật toán *SPT*-Steiner và *PD*-Steiner với thuật toán *MST*-Steiner

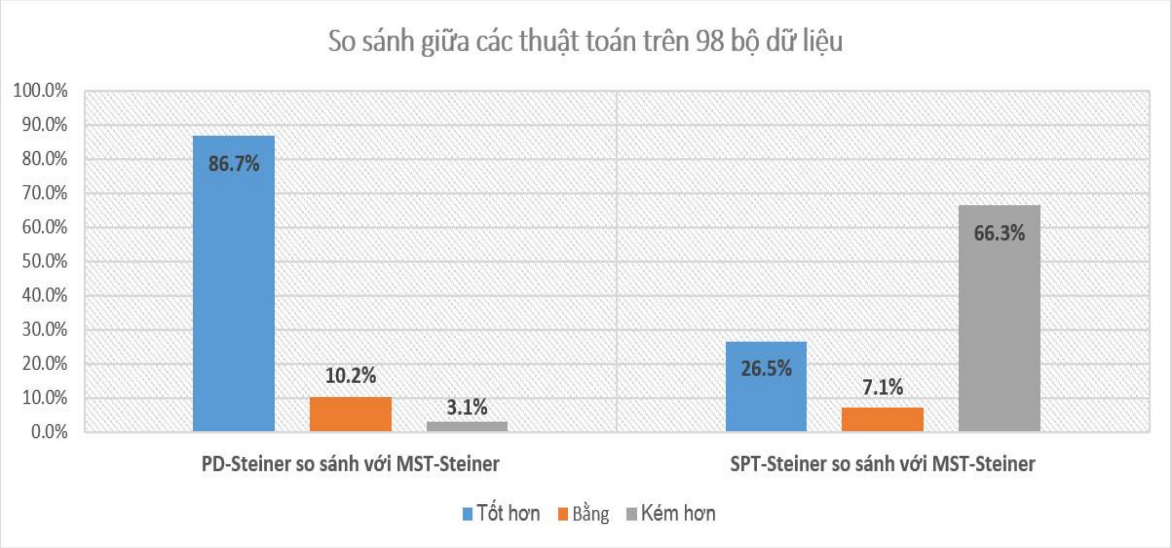
| **Nhóm đồ thị** | ***SPT* < *MST*** | | ***SPT* = *MST*** | | ***SPT* > *MST*** | | ***PD* < *MST*** | | ***PD* = *MST*** | | ***PD* > *MST*** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***SL*** | ***%*** | ***SL*** | ***%*** | ***SL*** | ***%*** | ***SL*** | ***%*** | ***SL*** | ***%*** | ***SL*** | ***%*** |
| *steinb* | 5 | 27.8 | 3 | 16.7 | 10 | 55.6 | 12 | 66.7 | 6 | 33.3 | 0 | 0.0 |
| *steinc* | 6 | 30.0 | 1 | 5.0 | 13 | 65.0 | 17 | 85.0 | 1 | 5.0 | 2 | 10.0 |
| *steind* | 4 | 20.0 | 2 | 10.0 | 14 | 70.0 | 17 | 85.0 | 2 | 10.0 | 1 | 5.0 |
| *steine* | 7 | 35.0 | 1 | 5.0 | 12 | 60.0 | 20 | 100.0 | 0 | 0.0 | 0 | 0.0 |
| *steinf* | 4 | 20.0 | 0 | 0.0 | 16 | 80.0 | 19 | 95.0 | 1 | 5.0 | 0 | 0.0 |
| Tổng cộng: | 26 | 26.5 | 7 | 7.1 | 65 | 66.3 | 85 | 86.7 | 10 | 10.2 | 3 | 3.1 |

Với 18 bộ dữ liệu nhóm *steinb*; thuật toán *SPT*-Steiner cho chất lượng lời giải tốt hơn, bằng, kém hơn thuật toán *MST*-Steiner lần lượt là: (27.8%, 16.7%, 55.6%); thuật toán *PD*-Steiner cho chất lượng lời giải tốt hơn, bằng, kém hơn thuật toán *MST*-Steiner lần lượt là: (66.7%, 33.3%, 0.0%). Kết quả so sánh các thuật toán *SPT*-Steiner

và *PD*-Steiner với thuật toán *MST*-Steiner trên mỗi nhóm dữ liệu *steinc, steind, steine, steinf* còn lại cũng được thể hiện chi tiết ở Bảng 2.6.

Đánh giá chung trên toàn bộ 98 bộ dữ liệu, thuật toán *SPT*-Steiner cho chất lượng lời giải tốt hơn, bằng, kém hơn so với thuật toán *MST*-Steiner lần lượt là: (26.5%, 7.1%, 66.3%) số bộ dữ liệu; và thuật toán *PD*-Steiner cho chất lượng lời giải tốt hơn, bằng, kém hơn thuật toán *MST*-Steiner lần lượt là: (86.7%, 10.2%, 3.1%) số bộ dữ liệu.

Kết quả so sánh giữa thuật toán *SPT*-Steiner và *PD*-Steiner với thuật toán *MST*-Steiner trên tổng số 98 bộ dữ liệu được minh họa bằng biểu đồ như Hình 2.10.




Hình 2.10. So sánh giữa các thuật toán trên 98 bộ dữ liệu

***- Thời gian chạy của các thuật toán MST-Steiner, SPT-Steiner và PD-Steiner***

Tiếp theo là ghi nhận thời gian chạy trung bình của ba thuật toán *MST*-Steiner, *SPT*-Steiner và *PD*-Steiner qua các test có số thứ tự 13, 14, 15 của mỗi nhóm dữ liệu (*vì các test này có cùng số đỉnh và số cạnh*).

Thời gian chạy ngoài việc phụ thuộc vào độ phức tạp thời gian tính của thuật toán; còn phụ thuộc vào môi trường thực nghiệm…Tuy nhiên, ghi nhận về thời gian chạy các thuật toán được liệt kê ở Bảng 2.7 cung cấp thêm thông tin tham khảo sâu sát hơn về ba thuật toán này.

Bảng 2.7. Thời gian tính trung bình của các thuật toán theo mỗi nhóm dữ liệu

| **Nhóm đồ thị** | ***MST*-Steiner** | ***SPT*-Steiner** | ***PD*-Steiner** |
|---|---|---|---|
| *steinb* | 0.001 | 0.003 | 0.001 |
| *steinc* | 0.032 | 0.298 | 0.023 |
| *steind* | 0.136 | 1.752 | 0.139 |
| *steine* | 0.939 | 23.334 | 1.402 |
| *steinf* | 19.594 | 1241.592 | 23.393 |

***Ghi chú**: Thời gian chạy trung bình của thuật toán trên một nhóm đồ thị bằng trung bình cộng thời gian chạy của các testcase trong nhóm đồ thị đó và được tính bằng đơn vị giây (s).*

## 2.6. CẢI TIẾN THUẬT TOÁN HEURISTIC GIẢI BÀI TOÁN *SMT* TRONG TRƯỜNG HỢP ĐỒ THỊ THƯA KÍCH THƯỚC LỚN

Trong phần này, luận án đề xuất cải tiến hai thuật toán heuristic *PD*-Steiner và *SPT*-Steiner dựa trên sự kết hợp ý tưởng của bài toán tìm đường đi ngắn nhất và tìm cây khung nhỏ nhất của đồ thị, đồng thời sử dụng thuật toán Dial [59] (một biến thể của thuật toán Dijkstra, được khuyến khích dùng trong trường hợp đồ thị có trọng số cạnh nhỏ) thay cho thuật toán Dijkstra trong tìm đường đi ngắn nhất, để giải bài toán *SMT* trong trường hợp đồ thị thưa kích thước lớn và đặt tên cho hai thuật toán này là: *i-PD*-Steiner và *i-SPT*-Steiner (*i: improving*).

Hai thuật toán heuristic cải tiến này được cài đặt thực nghiệm trên hệ thống dữ liệu mở rộng gồm 80 đồ thị thưa kích thước lớn lên đến 100000 đỉnh; đồng thời phân tích, đánh giá và so sánh chất lượng lời giải của thuật toán *i-PD*-Steiner và *i-SPT*-Steiner với thuật toán *MST*-Steiner [14]. Vấn đề này càng có ý nghĩa xét về gốc độ bài toán thực thi, khi áp dụng bài toán Cây Steiner vào vấn đề thực tế.

### 2.6.1. Thuật toán *i-SPT*-Steiner

Mục này, trình bày thuật toán heuristic cải tiến *i-SPT*-Steiner giải bài toán *SMT* trong trường hợp đồ thị thưa kích thước lớn.

Cây đường đi ngắn nhất có gốc tại đỉnh *s* của đồ thị *G* có thể tìm được nhờ áp dụng thuật toán Dial [59] để tìm đường đi ngắn nhất từ đỉnh *s* đến tất cả các đỉnh còn lại. Độ phức tạp của thuật toán Dial là $O(m + nC)$ [59] với *C* là giá trị trọng số lớn nhất của đồ thị.

**Algorithm 2.4: *i-SPT*-Steiner (i-Shortest Path Tree-Steiner)**

**Input**: Đồ thị $G = (V(G), E(G))$, tập đỉnh terminal $L \subseteq V(G)$;

**Output**: Cây Steiner nhỏ nhất $T$;

1. Sử dụng Thuật toán **Dial** tìm các cây đường đi ngắn nhất xuất phát tại các đỉnh thuộc tập terminal $L$: $SPT_1$, $SPT_2$, …, $SPT_L$
2. **for** (T $\in$ {$SPT_1$, $SPT_2$, …, $SPT_L$}) **do**
3. Với mỗi đỉnh treo $u \in T$, nếu $u \notin L$ thì xóa cạnh chứa đỉnh $u$ khỏi $E(T)$, xóa đỉnh $u$ trong $V(T)$ và cập nhật bậc của đỉnh kề với đỉnh $u$ trong $T$.
4. Lặp lại bước 3 đến khi $T$ không còn thay đổi (bước này gọi là bước xóa các cạnh dư thừa); sau bước này, thu được các cây $SPT_i$, tương ứng với các cây $SPT_i$.
5. **end for**
6. Tìm một cây $SPT_i$, có tổng trọng số các cạnh nhỏ nhất là $SPT_{min}$;
7. **return** $SPT_{min}$;

*Độ phức tạp thời gian tính của thuật toán i-SPT-Steiner*

Do thuật toán Dial sử dụng cấu trúc dữ liệu hàng đợi ưu tiên có độ phức tạp thời gian tính là $O(m + nC)$ [59] nên bước 1 có độ phức tạp $O(|L|(m + nC))$; bước 2 có độ phức tạp $O(n|L|)$; bước 3 có độ phức tạp $O(n)$. Vậy, thuật toán *i-SPT*-Steiner có độ phức tạp thời gian tính là: $O(|L|(m + nC))$.

### 2.6.2. Thuật toán *i-PD*-Steiner

Mục này, trình bày thuật toán heuristic cải tiến *i-PD*-Steiner giải bài toán *SMT* trong trường hợp đồ thị thưa kích thước lớn.

**Algorithm 2.5: *i-PD*-Steiner (i-Prim+Dijkstra-Steiner)**

**Input**: Đồ thị $G = (V(G), E(G))$, tập đỉnh terminal $L \subseteq V(G)$;

**Output**: Cây Steiner nhỏ nhất $T$;

**1.** Chọn một đỉnh $u \in L$, đặt $V(T) = \{u\}$;

**2. for** mỗi đỉnh $v \in L$ **do**

**3.** Tìm cây đường đi ngắn nhất xuất phát từ đỉnh $v$;

(Ở bước này sử dụng Thuật toán **Dial** [59] để tìm các cây đường đi ngắn nhất)

**4. end for**

**5. while** ($T$ chưa chứa tất cả các đỉnh thuộc tập $L$) **do**

**6.** Từ mỗi đỉnh $v \in L$ và $v \notin V(T)$, Chọn đường đi ngắn nhất $P$ xuất phát từ đỉnh $v$ đến đỉnh $z \in V(T)$, với $z$ là đỉnh kết thúc của đường đi ngắn nhất trong tất cả các đường đi ngắn nhất xuất phát từ $v$ đến các đỉnh của $V(T)$;

**7.** Kết nạp các đỉnh và các cạnh trên đường đi $P$ vào cây $T$;

**8. end while**

**9. return** $T$;

*Độ phức tạp thời gian tính của thuật toán i-PD-Steiner*

Dòng 1 có độ phức tạp là $O(1)$.

Dòng 3 sử dụng thuật toán Dial để tìm đường đi ngắn nhất có độ phức tạp là $O(m + nC)$. Vì vậy, vòng lặp ở bước 2 có độ phức tạp là $O(|L|(m + nC))$.

Dòng 6 có độ phức tạp là $O(|L||V(T)|)$, nên vòng lặp ở bước 5 có độ phức tạp là $O(|L|^2 |V(T)|)$.

Tóm lại, thuật toán *i-PD*-Steiner có độ phức tạp là: $O(|L|max(m + nC, |L||V(T)|))$.

## 2.7. THỰC NGHIỆM VÀ ĐÁNH GIÁ

### 2.7.1. Dữ liệu thực nghiệm

Luận án đề xuất chọn 80 bộ dữ liệu là các đồ thị thưa kích thước lớn, trong đó có: 20 bộ test là các đồ thị thưa kích thước lớn 10000 đỉnh *steinf*; 20 bộ test là các đồ thị thưa kích thước lớn 20000 đỉnh *steing*; 20 bộ test là các đồ thị thưa kích thước lớn 50000 đỉnh *steinh* và 20 bộ test là các đồ thị thưa kích thước lớn 100000 đỉnh *steini*. Các đồ thị này cũng như trọng số cạnh được sinh ngẫu nhiên.

### 2.7.2. Môi trường thực nghiệm

Các thuật toán *MST*-Steiner, *i-SPT*-Steiner và *i-PD*-Steiner được cài đặt bằng ngôn ngữ C++, sử dụng môi trường Code::Blocks 17.12 và trình biên dịch GNU GCC ver 9.3.0; được thực nghiệm trên một máy chủ ảo, Hệ điều hành Ubuntu 20.04.1 LTS (Focal Fossa), 64bit, Intel(R) Xeon (R) CPU Platinum 8160 @ 2.8 GHz, Cache 33MB, RAM 64GB. Các cấu trúc dữ liệu được sử dụng trong cài đặt đều được chuyển sang cấu trúc dữ liệu động và tối ưu.

### 2.7.3. Kết quả thực nghiệm

Kết quả thực nghiệm các thuật toán được ghi nhận ở Bảng 2.8, Bảng 2.9, Bảng 2.10 và Bảng 2.11. Các bảng này có cấu trúc như sau: Cột đầu tiên (Test) là tên các bộ dữ liệu trong hệ thống dữ liệu thực nghiệm; số đỉnh ($n$), số cạnh ($m$) và số đỉnh thuộc tập terminal ($|L|$) của từng đồ thị; các cột tiếp theo ghi nhận giá trị chi phí Cây Steiner tương ứng với từng thuật toán.

Bảng 2.8. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steinf*

| **Test** | ***n*** | ***m*** | **\|*L*\|** | ***MST*-Steiner** | ***i-SPT*-Steiner** | ***i-PD*-Steiner** |
|---|---|---|---|---|---|---|
| steinf1.txt | 10000 | 93750 | 10 | 58 | 49 | 51 |
| steinf2.txt | 10000 | 93750 | 20 | 97 | 95 | 94 |
| steinf3.txt | 10000 | 93750 | 83 | 2119 | 2551 | 2027 |
| steinf4.txt | 10000 | 93750 | 1250 | 2801 | 3532 | 2691 |
| steinf5.txt | 10000 | 93750 | 2500 | 4957 | 6567 | 4826 |
| steinf6.txt | 10000 | 125000 | 10 | 40 | 40 | 39 |
| steinf7.txt | 10000 | 125000 | 20 | 71 | 67 | 66 |
| steinf8.txt | 10000 | 125000 | 834 | 1961 | 2399 | 1838 |
| steinf9.txt | 10000 | 125000 | 1250 | 2576 | 3303 | 2444 |
| steinf10.txt | 10000 | 125000 | 2500 | 4282 | 5823 | 4124 |
| steinf11.txt | 10000 | 156250 | 10 | 35 | 35 | 35 |
| steinf12.txt | 10000 | 156250 | 20 | 77 | 74 | 70 |
| steinf13.txt | 10000 | 156250 | 834 | 1727 | 2199 | 1627 |
| steinf14.txt | 10000 | 156250 | 1250 | 2335 | 3049 | 2243 |
| steinf15.txt | 10000 | 156250 | 2500 | 3933 | 5398 | 3797 |
| steinf16.txt | 10000 | 187500 | 10 | 43 | 38 | 38 |
| steinf17.txt | 10000 | 187500 | 20 | 75 | 69 | 66 |
| steinf18.txt | 10000 | 187500 | 834 | 1597 | 2050 | 1513 |
| steinf19.txt | 10000 | 187500 | 1250 | 2244 | 2910 | 2153 |
| steinf20.txt | 10000 | 187500 | 2500 | 3764 | 5122 | 3624 |

Dựa vào Bảng 2.8, đánh giá kết quả thực nghiệm thuật toán đối với 20 bộ dữ liệu là các đồ thị thuộc nhóm *steinf;* thuật toán *i-PD*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner tương ứng là: (95.0%, 5.0%, 0.0%) số bộ dữ liệu; thuật toán *i-SPT*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner tương ứng là: (30.0%, 10.0%, 60.0%) số bộ dữ liệu; thuật toán *i-PD*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *i-SPT*-Steiner tương ứng là: (85.0%, 10.0%, 5.0%) số bộ dữ liệu.

Bảng 2.9. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steing*

| **Test** | ***n*** | ***m*** | **\|*L*\|** | ***MST*-Steiner** | ***i-SPT*-Steiner** | ***i-PD*-Steiner** |
|---|---|---|---|---|---|---|
| steing1.txt | 20000 | 215000 | 15 | 79 | 75 | 75 |
| steing2.txt | 20000 | 215000 | 25 | 110 | 102 | 105 |
| steing3.txt | 20000 | 215000 | 950 | 2426 | 3081 | 2348 |
| steing4.txt | 20000 | 215000 | 1750 | 4039 | 5085 | 3838 |
| steing5.txt | 20000 | 215000 | 3780 | 7318 | 9689 | 7099 |
| steing6.txt | 20000 | 275000 | 15 | 71 | 66 | 64 |
| steing7.txt | 20000 | 275000 | 25 | 112 | 111 | 100 |
| steing8.txt | 20000 | 275000 | 950 | 2245 | 2830 | 2129 |
| steing9.txt | 20000 | 275000 | 1750 | 3643 | 4709 | 3489 |
| steing10.txt | 20000 | 275000 | 3780 | 6511 | 8773 | 6319 |
| steing11.txt | 20000 | 385000 | 15 | 65 | 60 | 59 |
| steing12.txt | 20000 | 385000 | 25 | 98 | 94 | 90 |
| steing13.txt | 20000 | 385000 | 950 | 2041 | 2577 | 1919 |
| steing14.txt | 20000 | 385000 | 1750 | 3211 | 4253 | 3081 |
| steing15.txt | 20000 | 385000 | 3780 | 5788 | 7981 | 5641 |

| | | | | | | |
|---|---|---|---|---|---|---|
| steing16.txt | 20000 | 447500 | 15 | 61 | 55 | 54 |
| steing17.txt | 20000 | 447500 | 25 | 95 | 93 | 87 |
| steing18.txt | 20000 | 447500 | 950 | 1954 | 2485 | 1845 |
| steing19.txt | 20000 | 447500 | 1750 | 3079 | 4063 | 2970 |
| steing20.txt | 20000 | 447500 | 3780 | 5534 | 7647 | 5380 |

Dựa vào Bảng 2.9, đánh giá kết quả thực nghiệm thuật toán đối với 20 bộ dữ liệu là các đồ thị thuộc nhóm *steing*; thuật toán *i-PD*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner tương ứng là: (100%, 0%, 0%) số bộ dữ liệu; thuật toán *i-SPT*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner tương ứng là: (40%, 0%, 60%) số bộ dữ liệu; thuật toán *i-PD*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *i-SPT*-Steiner tương ứng là: (90%, 5%, 5%) số bộ dữ liệu.

Bảng 2.10. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steinh*

| **Test** | ***n*** | ***m*** | **\|*L*\|** | ***MST*-Steiner** | ***i-SPT*-Steiner** | ***i-PD*-Steiner** |
|---|---|---|---|---|---|---|
| steinh1.txt | 50000 | 425000 | 15 | 76 | 71 | 71 |
| steinh2.txt | 50000 | 425000 | 25 | 129 | 116 | 118 |
| steinh3.txt | 50000 | 425000 | 950 | 2707 | 3296 | 2587 |
| steinh4.txt | 50000 | 425000 | 1750 | 4344 | 5466 | 4125 |
| steinh5.txt | 50000 | 425000 | 3780 | 7947 | 10348 | 7563 |
| steinh6.txt | 50000 | 475000 | 15 | 71 | 70 | 66 |
| steinh7.txt | 50000 | 475000 | 25 | 99 | 105 | 96 |
| steinh8.txt | 50000 | 475000 | 950 | 2531 | 3129 | 2402 |
| steinh9.txt | 50000 | 475000 | 1750 | 4198 | 5299 | 3951 |

| | | | | | | |
|---|---|---|---|---|---|---|
| steinh10.txt | 50000 | 475000 | 3780 | 7718 | 10099 | 7336 |
| steinh11.txt | 50000 | 528000 | 15 | 80 | 72 | 74 |
| steinh12.txt | 50000 | 528000 | 25 | 112 | 110 | 106 |
| steinh13.txt | 50000 | 528000 | 950 | 2498 | 3082 | 2378 |
| steinh14.txt | 50000 | 528000 | 1750 | 4026 | 5132 | 3784 |
| steinh15.txt | 50000 | 528000 | 3780 | 7275 | 9702 | 6939 |
| steinh16.txt | 50000 | 587500 | 15 | 70 | 67 | 63 |
| steinh17.txt | 50000 | 587500 | 25 | 123 | 109 | 109 |
| steinh18.txt | 50000 | 587500 | 950 | 2408 | 2962 | 2262 |
| steinh19.txt | 50000 | 587500 | 1750 | 3886 | 4987 | 3643 |
| steinh20.txt | 50000 | 587500 | 3780 | 7127 | 9489 | 6797 |

Dựa vào Bảng 2.10, đánh giá kết quả thực nghiệm thuật toán đối với 20 bộ dữ liệu là các đồ thị thuộc nhóm *steinh*; thuật toán *i-PD*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner tương ứng là: (100.0%, 0.0%, 0.0%) số bộ dữ liệu; thuật toán *i-SPT*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner tương ứng là: (35.0%, 0.0%, 65.0%) số bộ dữ liệu; thuật toán *i-PD*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *i-SPT*-Steiner tương ứng là: (80.0%, 10.0%, 10.0%) số bộ dữ liệu.

Bảng 2.11. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steini*

| **Test** | ***n*** | ***m*** | **\|*L*\|** | ***MST*-Steiner** | ***i-SPT*-Steiner** | ***i-PD*-Steiner** |
|---|---|---|---|---|---|---|
| steini1.txt | 100000 | 125000 | 25 | 188 | 190 | 180 |
| steini2.txt | 100000 | 125000 | 45 | 369 | 364 | 358 |
| steini3.txt | 100000 | 125000 | 1250 | 6390 | 6677 | 6363 |

| | | | | | | |
|---|---|---|---|---|---|---|
| steini4.txt | 100000 | 125000 | 2450 | 12209 | 12810 | 12178 |
| steini5.txt | 100000 | 125000 | 4500 | 21754 | 22868 | 21664 |
| steini6.txt | 100000 | 200000 | 25 | 148 | 144 | 142 |
| steini7.txt | 100000 | 200000 | 45 | 264 | 273 | 253 |
| steini8.txt | 100000 | 200000 | 1250 | 4978 | 5510 | 4902 |
| steini9.txt | 100000 | 200000 | 2450 | 9066 | 10376 | 8969 |
| steini10.txt | 100000 | 200000 | 4500 | 15126 | 17456 | 14879 |
| steini11.txt | 100000 | 500000 | 25 | 111 | 110 | 104 |
| steini12.txt | 100000 | 500000 | 45 | 184 | 191 | 173 |
| steini13.txt | 100000 | 500000 | 1250 | 3133 | 3849 | 2974 |
| steini14.txt | 100000 | 500000 | 2450 | 5487 | 6889 | 5207 |
| steini15.txt | 100000 | 500000 | 4500 | 8951 | 11629 | 8532 |
| steini16.txt | 100000 | 2500000 | 25 | 66 | 72 | 66 |
| steini17.txt | 100000 | 2500000 | 45 | 120 | 128 | 111 |
| steini18.txt | 100000 | 2500000 | 1250 | 2031 | 2556 | 1951 |
| steini19.txt | 100000 | 2500000 | 2450 | 3366 | 4529 | 3268 |
| steini20.txt | 100000 | 2500000 | 4500 | 5167 | 7591 | 5118 |

Dựa vào Bảng 2.11, đánh giá kết quả thực nghiệm thuật toán đối với 20 bộ dữ liệu là các đồ thị nhóm *steini*; thuật toán *i-PD*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner tương ứng là: (95.0%, 5.0%, 0.0%) số bộ dữ liệu; thuật toán *i-SPT*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner tương ứng là: (15.0%, 0.0%, 85.0%) số bộ dữ liệu; thuật toán *i-PD*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *i-SPT*-Steiner tương ứng là (100.0%, 0.0%, 0.0%) số bộ dữ liệu.

### 2.7.4. Đánh giá kết quả thực nghiệm

***- Đánh giá chất lượng của các thuật toán i-SPT-Steiner và i-PD-Steiner***

Đánh giá trên tổng số 80 bộ dữ liệu, thuật toán *i-PD*-Steiner cho lời giải có chất lượng tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner là: (97.5%, 2.5%, 0.0%) số bộ dữ liệu; thuật toán *i-SPT*-Steiner cho lời giải có chất lượng tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner là: (30.0%, 2.5%, 67.5%) số bộ dữ liệu; thuật toán *i-PD*-Steiner cho lời giải có chất lượng tốt hơn, tương đương, kém hơn thuật toán *i-SPT*-Steiner là: (88.75%, 6.25%, 5.0%) số bộ dữ liệu;

***- Đánh giá thời gian chạy của các thuật toán i-SPT-Steiner và i-PD-Steiner***

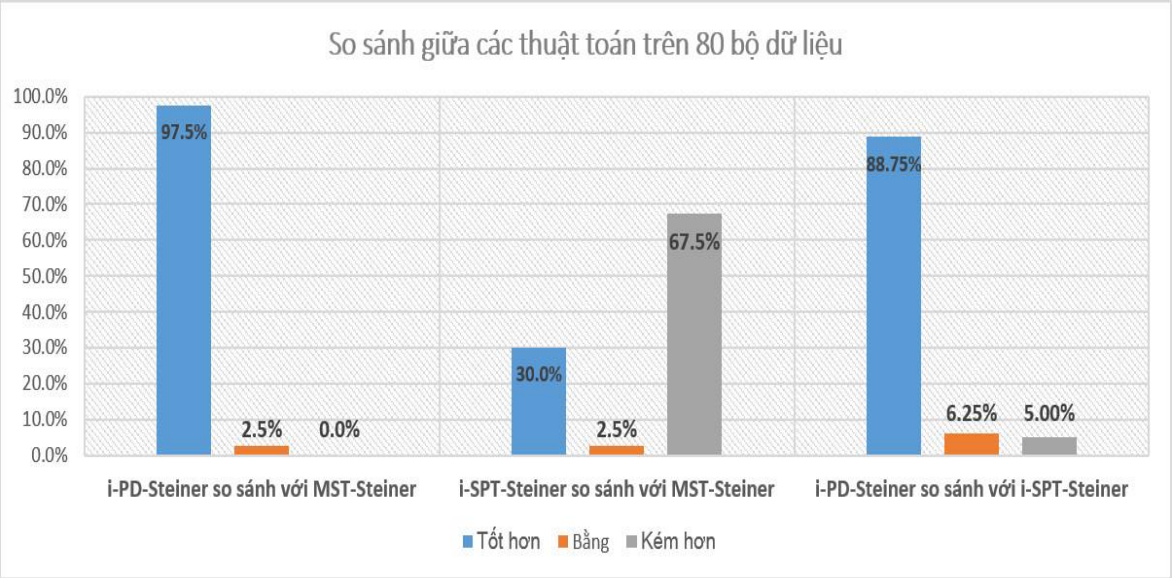
Thời gian chạy trung bình trên các bộ dữ liệu của ba thuật toán: *MST*-Steiner, *i-SPT*-Steiner và *i-PD*-Steiner được ghi nhận như trong Bảng 2.12. Theo Bảng 2.12, thuật toán *i-PD*-Steiner có thời gian chạy nhanh hơn so với thuật toán *MST*-Steiner và thuật toán *i-SPT*-Steiner. Thuật toán *i-SPT*-Steiner có thời gian chạy chậm hơn so với thuật toán *MST*-Steiner. Kết quả ghi nhận về thời gian thực hiện của ba thuật toán trên cung cấp thêm thông tin tham khảo hữu ích về các thuật toán này (Cấu trúc dữ liệu hàng đợi ưu tiên được lựa chọn để cài đặt các thuật toán trên).

Bảng 2.12. Thời gian chạy trung bình của các thuật toán

| **Nhóm đồ thị** | ***i-PD*-Steiner** | ***MST*-Steiner** | ***i-SPT*-Steiner** |
|---|---|---|---|
| *steinf* | 83,758 | 137,538 | 928,68 |
| *steing* | 280,356 | 421,402 | 2374,775 |
| *steinh* | 416,057 | 766,585 | 30416,253 |
| *steini* | 1243,415 | 1561,921 | 93943,235 |

***Ghi chú**: Thời gian chạy trung bình của thuật toán trên một nhóm đồ thị bằng trung bình cộng thời gian chạy của các testcase trong nhóm đồ thị đó và được tính bằng đơn vị giây (s).*

Kết quả so sánh giữa thuật toán *i-PD*-Steiner và *i-SPT*-Steiner với thuật toán *MST*-Steiner trên tổng số 80 bộ dữ liệu được minh họa bằng biểu đồ như Hình 2.11.




Hình 2.11. So sánh giữa các thuật toán trên 80 bộ dữ liệu

## 2.8. ĐÁNH GIÁ CÁC THUẬT TOÁN THÔNG QUA ĐỘ PHỨC TẠP

Bảng 2.13 ghi nhận kết quả độ phức tạp thời gian của các thuật toán heuristic đề xuất. Thông qua bảng này cho thấy: Thuật toán *SPT*-Steiner có độ phức tạp thời gian lớn nhất, thuật toán *i-PD*-Steiner có độ phức tạp thời gian nhỏ nhất, qua đó cung cấp thêm thông tin về tính hiệu quả của các thuật toán này.

Bảng 2.13. Độ phức tạp thời gian của các thuật toán

| **Thuật toán** | **Độ phức tạp thời gian** |
|---|---|
| *MST*-Steiner | $O(n\|L\|^2)$ |
| *SPT*-Steiner | $O(mnlog\ n)$ |
| *PD*-Steiner | $O(\|L\|n^2)$ |
| *i-SPT*-Steiner | $O(\|L\|(m + nC))$ |
| *i-PD*-Steiner | $O(\|L\|max(m + nC, \|L\|\|V(T)\|))$ |
| Các thuật toán được xếp theo độ phức tạp thời gian giảm dần: | ***SPT*-Steiner, *i-SPT*-Steiner, *PD*-Steiner, *MST*-Steiner, *i-PD*-Steiner** |

***Chú thích:***
*n:* số đỉnh của đồ thị *G*; *m:* số cạnh của đồ thị *G; |V(T)|:* số đỉnh của cây *T*; *|L|:* số đỉnh của tập *terminal*; *C:* trọng số tối đa của cạnh trong đồ thị *G*.

## 2.9. KẾT LUẬN CHƯƠNG 2

Trong chương này, luận án đề xuất hai thuật toán heuristic mới: *SPT*-Steiner và *PD*-Steiner giải bài toán *SMT*; các thuật toán này được cài đặt thực nghiệm trên 98 bộ dữ liệu (gồm có 78 bộ dữ liệu là các đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn và 20 bộ dữ liệu mở rộng là các đồ thị thưa kích thước lớn lên đến 10000 đỉnh - *steinf*). Từ kết quả thực nghiệm, luận án đã tiến hành so sánh, đánh giá chi tiết hiệu quả của hai thuật toán heuristic đề xuất với thuật toán heuristic *MST*-Steiner [14] đã được công bố trước đó. Hai thuật toán đề xuất bởi luận án cho chất lượng lời giải tốt hơn thuật toán *MST*-Steiner [14] trên một số bộ dữ liệu. Thời gian chạy của các thuật toán *SPT*-Steiner và *PD*-Steiner chậm hơn so với thuật toán *MST*-Steiner.

Ngoài ra, luận án cũng đề xuất hai thuật toán heuristic cải tiến: *i-SPT*-Steiner và *i-PD*-Steiner để giải bài toán *SMT* trong trường hợp đồ thị thưa kích thước lớn. Hai thuật toán heuristic cải tiến *i-SPT*-Steiner và *i-PD*-Steiner được cài đặt thực nghiệm và so sánh, đánh giá hiệu quả trên 80 bộ dữ liệu là các đồ thị thưa kích thước lớn lên đến 100000 đỉnh. Hai thuật toán heuristic cải tiến cho chất lượng lời giải tốt hơn hoặc tương đương thuật toán *MST*-Steiner [14] trên một số bộ dữ liệu. Thời gian chạy của thuật toán *i-PD*-Steiner nhanh hơn so với thuật toán *MST*-Steiner và thuật toán *i-SPT*-Steiner. Thời gian chạy của thuật toán *i-SPT*-Steiner chậm hơn so với thuật *MST*-Steiner và thuật toán *i-PD*-Steiner.

Các thuật toán heuristic đề xuất mới và cải tiến, cùng kết quả thực nghiệm thuật toán trong chương này là những thông tin hữu ích cho việc nghiên cứu phát triển thuật toán metaheuristic giải bài toán *SMT* sẽ được đề cập tiếp theo trong Chương 3.

# Chương 3. ĐỀ XUẤT THUẬT TOÁN METAHEURISTIC GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

*Từ kết quả nghiên cứu đạt được trong Chương 1 và Chương 2. Tiếp theo trong chương cuối này, luận án đề xuất ba thuật toán metaheuristic mới dạng cá thể, quần thể dựa trên ý tưởng kết hợp lược đồ cơ bản của các thuật toán tiến hóa với các chiến lược tìm kiếm lận cận để giải bài toán SMT. Đồng thời luận án cũng đề xuất hai chiến lược tìm kiếm lận mới cho bài toán SMT, các chiến lược tìm kiếm lân cận đề xuất có thể sử dụng trong các lược đồ thuật toán metaheuristic nhằm nâng cao chất lượng lời giải cho các thuật toán. Chương này được tổng hợp từ các công trình [CT1], [CT2], [CT3], và [CT4] trong danh mục các công trình nghiên cứu của tác giả.*

## 3.1. GIỚI THIỆU HƯỚNG TIẾP CẬN METAHEURISTIC GIẢI BÀI TOÁN SMT

Thuật toán metaheuristic sử dụng nhiều heuristic kết hợp với các kỹ thuật phụ trợ nhằm khai phá không gian tìm kiếm; metaheuristic thuộc lớp các thuật toán tìm kiếm tối ưu [8][56][82].

Hiện đã có các công trình sử dụng thuật toán metaheuristic giải bài toán *SMT*, chẳng hạn như: Thuật toán local search [29][43], thuật toán tìm kiếm với lân cận biến đổi [43], thuật toán di truyền [22], thuật toán tabu search [31][72], thuật toán di truyền song song [57],…

## 3.2. KHỞI TẠO LỜI GIẢI BAN ĐẦU

Chất lượng lời giải của các thuật toán metaheuristic chịu ảnh hưởng bởi cách thức khởi tạo lời giải ban đầu. Trong mục này, sẽ phân tích chi tiết một số cách thức khởi tạo lời giải ban đầu cho bài toán Cây Steiner nhỏ nhất.

### 3.2.1. Khởi tạo Cây Steiner theo một heuristic

**- Sử dụng cây đường đi ngắn nhất**

Tìm cây đường đi ngắn nhất $T$ xuất phát tại một đỉnh của đồ thị. Sau đó duyệt các đỉnh treo $u \in T$, nếu $u \notin L$ thì xóa cạnh chứa đỉnh $u$ khỏi $E(T)$, xóa đỉnh $u$ trong $V(T)$ và cập nhật bậc của đỉnh kề với đỉnh $u$ trong $T$. Lặp lại bước này đến khi $T$ không còn thay đổi (bước này gọi là bước xóa các cạnh dư thừa). Sau bước này ta thu được một Cây Steiner.

**- Sử dụng ý tưởng của thuật toán Prim**

Chọn đỉnh $u \in L$, đặt $T = \{u\}$; từ mỗi đỉnh $v \in L$ và $v$ chưa thuộc $T$, tìm đường đi ngắn nhất $P$ từ đỉnh $v$ đến các đỉnh $z \in T$; kết nạp các đỉnh và các cạnh trên đường đi $P$ vào cây $T$; lặp lại đến khi $T$ chứa tất cả các đỉnh thuộc tập $L$.

Cách làm trên có thể mở rộng để tạo một quần thể Cây Steiner ban đầu theo cách sau: Nếu sử dụng cây đường đi ngắn nhất thì có thể tạo cây có gốc tại các đỉnh khác nhau của đồ thị. Nếu sử dụng ý tưởng thuật toán Prim có thể chọn ngẫu nhiên các đỉnh $v \in L$ trong giai đoạn tìm đường đi ngắn nhất $P$.

Ta cũng có thể tạo Cây Steiner bằng cách áp dụng thuật toán Kruskal hoặc thuật toán Prim để tìm cây khung nhỏ nhất, sau đó xóa các cạnh sao cho các cạnh còn lại vẫn là một Cây Steiner.

### 3.2.2. Khởi tạo Cây Steiner ngẫu nhiên

Bắt đầu từ một đỉnh nào đó thuộc tập terminal $L$, tiếp theo trong mỗi bước lặp, trong số các đỉnh chưa được chọn để tham gia vào cây, ta chọn một đỉnh kề với ít nhất một đỉnh nằm trong cây đang được xây dựng mà không quan tâm đến trọng số của cạnh. Đỉnh được chọn và cạnh nối nó với đỉnh của cây đang được xây dựng sẽ được bổ sung vào cây; thuật toán dừng khi tất cả các đỉnh thuộc tập terminal $L$ đều đã được chọn. Sau đó, xóa các cạnh dư thừa trong cây $T$ sao cho $T$ vẫn là một Cây

Steiner. Đây là cách tạo lời giải ban đầu mà luận án sử dụng xuyên suốt trong tất cả các thuật toán được đề cập trong Chương 3.

Ưu điểm của việc tạo Cây Steiner ban đầu theo cách ngẫu nhiên so với cách sử dụng các thuật toán heuristic chính là sự đa dạng các cạnh trong Cây Steiner được hình thành. Chất lượng của quần thể ban đầu được tạo theo cách ngẫu nhiên dù không tốt bằng cách sử dụng các heuristic đặc thù của bài toán *SMT*; tuy nhiên sau quá trình tiến hóa, quần thể ban đầu được khởi tạo bằng cách ngẫu nhiên sẽ cho chất lượng lời giải tốt hơn.

**Algorithm 3.1: LikePrim algorithm**

**Input:** Đồ thị $G = (V(G), E(G))$;

**Output:** Trả lại cây khung ngẫu nhiên $T = (V(T), E(T))$;

1. Chọn ngẫu nhiên đỉnh $u \in V(G)$;
2. $V(T) = \{u\}$;
3. $E(T) = \emptyset$;
4. **while** ($|V(T)| < n$) **do**
5. Chọn ngẫu nhiên đỉnh $v \in V(G) - V(T)$ sao cho $v$ có kề với một đỉnh $z \in V(T)$;
6. $V(T) = V(T) \cup \{v\}$;
7. $E(T) = E(T) \cup \{(v, z)\}$;
8. **end while**
9. **return** cây khung $T$;

### 3.2.3. Khởi tạo Cây Steiner dựa vào xác suất

Mỗi cạnh $e$ của đồ thị được gán một xác suất $p(e)$, xác suất này tỉ lệ với trọng số của cạnh. Khi đó ta chọn ngẫu nhiên các cạnh dựa vào xác suất cho đến khi được Cây Steiner thì dừng. Tiếp theo ta xóa các cạnh dư thừa.

## 3.3. CÁC CHIẾN LƯỢC TÌM KIẾM CÂY STEINER LÂN CẬN

### 3.3.1. Định nghĩa Cây Steiner lân cận

**Định nghĩa 1.** *1-lân cận của Cây Steiner T*

Cho đồ thị *G* và *T* là một Cây Steiner của *G*. Ta gọi *1-lân cận* của Cây Steiner *T* là tập tất cả các Cây Steiner của đồ thị *G* sai khác với *T* đúng một cạnh. Nếu *T*' là một Cây Steiner thuộc *1-lân cận* của *T* thì ta nói *T* và *T*' là *1-lân cận* với nhau.

Trong một số trường hợp chúng ta còn sử dụng những lân cận rộng hơn so với *1-lân cận*. Định nghĩa *k-lân cận* dưới đây là mở rộng trực tiếp của định nghĩa *1-lân cận*.

**Định nghĩa 2.** *k-lân cận của Cây Steiner T*

Cho đồ thị *G* và *T* là một Cây Steiner của nó. Ta gọi *k-lân cận* của Cây Steiner *T* là tập tất cả các Cây Steiner của đồ thị *G* sai khác với *T* không quá *k* cạnh. Nếu *T*' là một Cây Steiner thuộc *k-lân cận* của *T* thì ta nói *T* và *T*' là *k-lân cận* với nhau.

**Định nghĩa 3.** *Lân cận tất định và lân cận ngẫu nhiên*

Nếu các Cây Steiner trong lân cận được xác định không phụ thuộc vào yếu tố ngẫu nhiên thì ta nói về lân cận tất định, còn nếu ngược lại, ta nói về lân cận ngẫu nhiên.

Tiếp theo, luận án sẽ trình bày một số chiến lược tìm kiếm Cây Steiner lân cận hiện biết.

### 3.3.2. Chiến lược chèn cạnh - xóa cạnh

Cho đồ thị vô hướng liên thông có trọng số *G*. Bắt đầu từ Cây Steiner *T* của *G* được khởi tạo ngẫu nhiên, chèn lần lượt từng cạnh $e \in E(G) - E(T)$ vào Cây Steiner *T*. Nếu Cây Steiner *T* không chứa chu trình thì cạnh *e* không cần được xem xét; nếu $E(T) \cup \{e\}$ chứa chu trình thì tìm một cạnh *e*' trên chu trình này sao cho việc loại nó

dẫn đến Cây Steiner *T'* có chi phí là nhỏ nhất. Tiếp theo, nếu $C(T') < C(T)$ thì thay $T$ bằng *T'*.

Thao tác tìm chu trình trong Cây Steiner $T$ sau khi chèn thêm một cạnh $e$ được tiến hành như sau: Khi chèn cạnh $e = (u, v)$ vào $T$, duyệt Cây Steiner $T$ theo chiều sâu bắt đầu từ $u$, lưu vết trên đường đi bằng mảng $p$ (đỉnh trước của một đỉnh trong phép duyệt). Tiếp theo, bắt đầu từ đỉnh $v$, truy vết theo mảng $p$ đến khi gặp $u$ thì kết thúc, các cạnh trên đường truy vết chính là các cạnh trong chu trình cần tìm.

### 3.3.3. Chiến lược tìm lân cận tốt hơn

Thủ tục tìm kiếm lân cận bắt đầu từ Cây Steiner $T$ được tiến hành như sau: loại ngẫu nhiên khỏi $T$ một cạnh $e$, chọn ngẫu nhiên cạnh $e$' từ tập $E(G) - E(T)$, nếu tập cạnh $E(T) - \{e\} \cup \{e'\}$ cho ta Cây Steiner $T'$ có chi phí tốt hơn $T$ thì ghi nhận Cây Steiner này. Thủ tục trên sẽ được lặp lại $k$ lần đối với Cây Steiner $T$, trong số các Cây Steiner được ghi nhận, chọn ra $T^*$ là Cây Steiner tốt nhất, nếu Cây Steiner $T$ có chi phí tốt hơn $T^*$ thì đặt $T = T^*$. Như vậy, quần thể $P$ được cập nhật sau khi thực hiện xong thủ tục tìm kiếm lân cận. Hàm tìm kiếm lân cận vừa mô tả được đặt tên là ***NeighSearch***($T$, $k$).

Thuật toán sau đây cho phép tìm kiếm Cây Steiner lân cận.

**Algorithm 3.2: NeighSearch algorithm**

***NeighSearch***($T$, $k$)

**Input:** Cây Steiner $T = (V(T), E(T))$ và số nguyên dương $k$

**Output:** Thay thế Cây Steiner $T$ bởi Cây Steiner tốt nhất trong số $k$ Cây Steiner *lân cận* được tạo ngẫu nhiên.

```
1. T* = T;// T* là Cây Steiner tốt nhất trong lân cận của cây T
2. for i = 1..k do
3.     Loại ngẫu nhiên một cạnh e ∈ E(T);
4.     Chọn ngẫu nhiên cạnh e'∈ E(G) – E(T);
```

5. **if** ($T'$ = ($V$, $E(T)$ – $\{e\} \cup \{e'\}$) là Cây Steiner) *và* ($C(T') < C(T^*)$)

6. $T^* = T'$;

7. **end for**

8. **if** $C(T^*) < C(T)$

9. $P = P - \{T\} \cup \{T^*\}$; //thay Cây Steiner $T$ bởi Cây Steiner $T^*$ trong quần thể $P$.

### 3.3.4. Chiến lược tìm lân cận ngẫu nhiên

Tìm kiếm ngẫu nhiên cho Cây Steiner $T$ được tiến hành tương tự như tìm kiếm lân cận nhưng không quan tâm đến chi phí trên cạnh: Loại ngẫu nhiên cạnh $e$ trong $T$, tìm ngẫu nhiên một cạnh $e'$ từ tập $E - E(T)$ sao cho $E(T) - \{e\} \cup \{e'\}$ cho ta Cây Steiner $T'$ (không quan tâm đến việc $T'$ có tốt hơn $T$ hay không). Cập nhật $T = T'$ và lặp lại $k$ lần thao tác trên.

Thuật toán sau đây cho phép tìm kiếm Cây Steiner ngẫu nhiên.

---

**Algorithm 3.3: Randsearch algorithm**

---

***Randsearch*(*T*, *k*)**

**Input:** Cây Steiner $T = (V(T), E(T))$ của đồ thị $G$, $k$ là số cạnh cần thay thế ngẫu nhiên.

**Output:** Cây Steiner $T$ sau khi đã thay thế $k$ cạnh ngẫu nhiên

1. **for** i = 1..$k$ **do**

2. Loại ngẫu nhiên cạnh $e \in T$;

3. Tìm ngẫu nhiên cạnh $e' \in E(G) - E(T)$ thỏa $T - \{e\} \cup \{e'\}$ là một Cây Steiner;

4. $T = T - \{e\} \cup \{e'\}$;

5. **end for**

6. **return** $T$;

### 3.3.5. Chiến lược tìm lân cận Node-based

Cho Cây Steiner *T*. Xét một đỉnh ngẫu nhiên $u \in T$ mà không thuộc tập đỉnh terminal *L*. Xóa đỉnh *u* và các cạnh kề đỉnh *u* trong *T*; khi đó *T* được phân rã thành một số thành phần liên thông; gọi đây là đồ thị *H*. Lần lượt bổ sung các cạnh theo thứ tự trọng số tăng dần của *G* - *H* vào đồ thị *H* cho đến khi nào *H* trở thành một cây. Xóa các cạnh dư thừa trong *H* để *H* trở thành một Cây Steiner *T'*. Nếu cây *T'* có chi phí tốt hơn *T* thì cập nhật *T* bằng *T'*; ngược lại thì ta giữ nguyên *T* [69].

**Algorithm 3.4: Node-based search algorithm**

**Input:** Cho đồ thị vô hướng $G = (V(G), E(G))$; $V(G)$ là tập đỉnh; $E(G)$ là tập cạnh; $L \subseteq V(G)$ là tập đỉnh terminal.

**Output:** Cây Steiner *T* nhỏ nhất

1. Sử dụng thuật toán tựa Prim để tìm cây khung của đồ thị, gọi là cây *T*;
2. Xóa các cạnh dư thừa của *T*, khi đó *T* là một Cây Steiner. Việc xóa các cạnh dư thừa được thực hiện như sau: Với mỗi Cây Steiner *T*, duyệt các đỉnh treo $u \in T$, nếu $u \notin L$ thì xóa cạnh chứa đỉnh *u* khỏi $E(T)$, xóa đỉnh *u* trong $V(T)$ và cập nhật bậc của đỉnh kề với đỉnh *u* trong *T*. Lặp lại bước này đến khi *T* không còn thay đổi.
3. **while** (điều kiện dừng chưa thỏa) **do**
4. Đặt $T_1 = T$;
5. Chọn đỉnh ngẫu nhiên $u \in T_1$; đỉnh *u* không thuộc tập đỉnh terminal *L*; sau đó loại bỏ các cạnh liên quan đến đỉnh *u* trong $T_1$; khi đó $T_1$ được phân rã thành nhiều thành phần liên thông; gọi đây là đồ thị $T_2$.
6. Sắp xếp các cạnh của đồ thị *G* theo trọng số tăng dần, lần lượt bổ sung các cạnh theo thứ tự đã được sắp của *G* vào đồ thị $T_2$ cho đến khi $T_2$ là một cây;

7. Xóa các cạnh dư thừa trong $T_2$;

8. Nếu cây $T_2$ có chi phí tốt hơn $T$ thì cập nhật $T$ bằng $T_2$; ngược lại hoặc là không tạo được cây $T_2$ thì đặt $T$ bằng $T_1$;

9. **end while**

10. **return** $T$;

### 3.3.6. Chiến lược tìm lân cận Path-based

Một *key-node* là một đỉnh của Cây Steiner có bậc ít nhất bằng 3.

Một *key-path* là một đường đi với tất cả các đỉnh trung gian (không là đỉnh *terminal*) đều có bậc bằng 2 và hai đỉnh đầu cuối của đường đi đó hoặc thuộc tập *terminal* hoặc là đỉnh *key-node*.

Việc tìm *key-path* ngẫu nhiên được thực hiện như sau: Chọn ngẫu nhiên một cạnh trong *T*; nếu hai đỉnh đầu cuối của *key-path* có bậc bằng 2 và là đỉnh Steiner thì thêm cạnh tiếp theo kề của đỉnh đó cho đến khi nào 2 đỉnh đầu cuối không phải bậc 2 và không phải Steiner thì dừng lại, và kiểm tra đường đi đó có phải là một *key-path* hay không. Nếu trong lúc thực hiện không thỏa mãn thì dừng.

*T* là một Cây Steiner. Giả sử *p* một *key-path* ngẫu nhiên trên cây *T*. Tiến hành loại bỏ *p*; khi đó *T* được tách thành hai thành phần liên thông $T_1$ và $T_2$. Chọn cạnh ngắn nhất có thể nối hai thành phần liên thông $T_1$ và $T_2$ với nhau; giả sử khi đó ta được cây mới là *T'*. Nếu cây *T'* có chi phí tốt hơn *T* thì cập nhật *T* bằng *T'*; ngược lại thì giữ nguyên *T* [69].

---

**Algorithm 3.5: Path-based search algorithm**

---

**Input:** Cho đồ thị vô hướng $G = (V(G), E(G))$; $V(G)$ là tập đỉnh; $E(G)$ là tập cạnh; $L \subseteq V(G)$ là tập đỉnh terminal.

**Output:** Cây Steiner nhỏ nhất *T*

1. Sử dụng thuật toán tựa Prim để tìm cây khung của đồ thị, gọi là cây *T*;

2. Xóa các cạnh dư thừa của $T$, khi đó $T$ là một Cây Steiner;

3. **while** (điều kiện dừng chưa thỏa) **do**

4. Đặt $T_1 = T$;

5. Giả sử $p$ một key path ngẫu nhiên nào đó; tiến hành loại bỏ $p$; khi đó $T$ được tách thành hai thành phần liên thông $T_a$ và $T_b$;

6. Chọn cạnh ngắn nhất có thể nối hai thành phần liên thông $T_a$ và $T_b$ với nhau; giả sử khi đó ta được cây mới là $T_2$.

7. Nếu cây $T_2$ có chi phí tốt hơn $T$ thì cập nhật $T$ bằng $T_2$; ngược lại hoặc nếu không tồn tại $T_2$ thì đặt $T$ bằng $T_1$;

8. **end while**

9. **return** $T$;

### 3.3.7. Chiến lược tìm kiếm lân cận tham lam

Cho Cây Steiner *T*. Loại ngẫu nhiên khỏi *T* một cạnh *e*, chọn ngẫu nhiên cạnh *e'* từ tập *E(G) - E(T)*, nhưng cạnh *e'* thêm vào cây *T* thì ta lấy ngẫu nhiên từ tập những cạnh kề với tập đỉnh của cây *T* trong tập *E(G) – E(T)* trên tiêu chí là chi phí càng nhỏ thì tỉ lệ chọn càng cao. Sau khi loại cạnh *e* và thêm cạnh *e'* ta được Cây Steiner *T'* có chi phí nhỏ hơn thì ghi nhận *T = T'*. Lặp lại *k* lần thủ tục này, ta được cây *T* có chi phí tốt nhất.

### 3.3.8. Chiến lược tìm kiếm lân cận có xác suất

Cho Cây Steiner *T*, mỗi cạnh *e* thuộc *T* được gán cho một xác suất chọn *p(e)*; nếu trọng số của cạnh *e* càng bé thì xác suất chọn *p(e)* càng cao. Loại ngẫu nhiên khỏi *T* một cạnh *e*, *e* là cạnh được chọn ngẫu nhiên với việc ưu tiên chọn cạnh có chi phí lớn (chi phí cạnh càng lớn thì tỉ lệ chọn càng cao). Sau đó tìm ngẫu nhiên một cạnh *e'* từ tập *E - E(T)*, với *e'* là cạnh chọn ngẫu nhiên với việc ưu tiên chọn cạnh có chi phí nhỏ (chi phí cạnh càng nhỏ thì tỉ lệ chọn càng cao) sao cho *E(T)* -

*{e}* ∪ *{e'}* cho ta Cây Steiner *T'* có chi phí tốt hơn *T* thì ghi nhận Cây Steiner này, *T* = *T'*. Lặp lại *k* lần thủ tục này, ta thu được cây *T* có chi phí tốt nhất.

Hai chiến lược tìm kiếm lân cận tham lam và tìm kiếm lân cận có xác suất, cũng như việc sử dụng chúng trong lược đồ thuật toán tìm kiếm lận cận biến đổi được công bố tại Công trình [CT4].

## 3.4. THUẬT TOÁN BEES GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

Phần này trình bày chi tiết các bước của thuật toán Bees giải bài toán *SMT*; thuật toán này được đặt tên là *Bees*-Steiner.

### 3.4.1. Điều kiện dừng của thuật toán Bees-Steiner

Thuật toán *Bees*-Steiner sử dụng điều kiện dừng sau: Lời giải tốt nhất tìm được (kỷ lục) của bài toán không được cải thiện sau một số lần lặp định trước (thường là một hàm theo kích cỡ của dữ liệu đầu vào).

### 3.4.2. Phân nhóm các cá thể

Quần thể *P* có *N* cá thể, sắp xếp các cá thể trong quần thể *P* theo chiều tăng dần chi phí của các cá thể. Sau khi sắp xếp, ta phân bố các cá thể vào ba nhóm: Nhóm 1 gồm *h* cá thể tốt nhất, nhóm 2 gồm *p* - *h* cá thể tốt tiếp theo và nhóm 3 gồm *N* - *p* cá thể còn lại.

Hàm sắp xếp quần thể và phân bố các cá thể vào các nhóm được đặt tên là ***SortPopulation*** (*P, N, h, p*).

Thuật toán sau đây cho phép phân nhóm các cá thể.

***SortPopulation* (*P, N, h, p*)**

**Input:** Quần thể *P* gồm *N* cây $T_1, T_2, \ldots, T_N$ của đồ thị *G*. Các tham số *h, p.*

**Output:** Xếp các cá thể trong *P* vào ba nhóm.

1. Sắp xếp các cá thể của quần thể theo chi phí tăng dần;

2. Xếp $h$ cá thể đầu tiên $T_1$, $T_2$,..., $T_h$ vào nhóm 1;

3. Xếp $p$ - $h$ cá thể tiếp theo $T_{h+1}$, $T_{h+2}$,..., $T_p$ vào nhóm 2;

4. Xếp $N$ - $p$ cá thể còn lại $T_{p+1}$, $T_{p+2}$,..., $T_N$ vào nhóm 3.

### 3.4.3. Sơ đồ Thuật toán Bees-Steiner

Thuật toán *Bees*-Steiner trước hết tạo quần thể ban đầu $P$, sau đó lặp lại các thao tác: Sắp xếp các cá thể thuộc quần thể $P$ và phân bố các cá thể vào các nhóm; mỗi cá thể thuộc nhóm 1 cho tìm kiếm lân cận $k_1$ lần, mỗi cá thể thuộc nhóm 2 cho tìm kiếm lân cận $k_2$ lần. Trong mỗi bước lặp, quần thể $P$ đã được cập nhật thông qua các thao tác tìm kiếm lân cận tốt hơn và tìm kiếm lân cận ngẫu nhiên. Khi thuật toán dừng, cá thể tốt nhất tìm được trong quá trình thực hiện thuật toán được công bố làm lời giải cần tìm.

Thuật toán ***Bees*-Steiner** được mô tả như sau:

**Algorithm 3.6: *Bees*-Steiner algorithm**

**Input:** Đồ thị $G = (V(G), E(G))$;

**Output:** Cây Steiner có chi phí nhỏ nhất tìm được $T_{best}$;

1. ***InitPopulation*** ($V(G)$, $E(G)$, $N$); // Tạo quần thể $P$ gồm các Cây Steiner $T_1$, $T_2$,...,$T_N$
2. **While** (*điều kiện dừng chưa thỏa*) **do**
3. ***SortPopulation*** ($P$, $N$, $h$, $p$);
4. Cập nhật $T_{best} = T_1$;
5. **For** i = $1..h$ **do**
6. ***NeighSearch*** ($T_i$, $k_1$);
7. **For** i = $h + 1..p$ **do**
8. ***NeighSearch*** ($T_i$, $k_2$);
9. **For** i = $p + 1..N$ **do**

10. ***RandSearch*** ($T_i$, $k_3$);

11. **end while**

12. Cập nhật cá thể $T_{best}$;

13. **return** $T_{best}$;

Độ phức tạp của thuật toán Bees-Steiner là: $O(N(Nlog(N) + V(hk_1 + (p - h)k_2 + (N - p)k_3)))$.

## 3.5. THUẬT TOÁN TÌM KIẾM LÂN CẬN BIẾN ĐỔI GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

Vấn đề lớn nhất mà các thuật toán metaheuristic gặp phải là nó dễ rơi vào bẫy tối ưu cục bộ. Để giải quyết vấn đề này, luận án đề xuất việc kết hợp thuật toán tìm kiếm lân cận biến đổi (Variable neighborhood search algorithm - *VNS*) giải bài toán *SMT* với hai chiến lược tìm kiếm lân cận Node-base [69] và Path-based [69] nhằm nâng cao chất lượng lời giải của thuật toán.

Về điều kiện dừng: Lời giải tốt nhất tìm được bởi thuật toán không được cải thiện qua một số lần lặp định trước.

Về tạo lời giải ngẫu nhiên ban đầu: Cây Steiner ban đầu được khởi tạo bằng thuật toán *LikePrim*.

**Algorithm 3.7: *VNS* algorithm**

**Input:** Đồ thị $G = (V(G), E(G))$;

**Output:** Cây Steiner có chi phí nhỏ nhất;

1. $T$ là cây khung ngẫu nhiên được tạo bởi thuật toán tựa Prim;
2. Xóa các cạnh dư thừa trong $T$ để thu được Cây Steiner;
3. **while** (điều kiện dừng chưa thỏa) **do**
4. Lần lượt thực hiện hai chiến lược tìm kiếm lân cận Node-based và Path-based;

5. Trong quá trình thực hiện các chiến lược tìm kiếm lân cận trên, ghi nhận lại lời giải tốt nhất;
6. Khi thực hiện một chiến lược tìm kiếm lân cận, nếu tìm được kỉ lục mới thì quay trở lại thực hiện thuật toán *VNS* từ đầu (sau vòng lặp while); ngược lại, chuyển sang chiến lược tìm kiếm lân cận tiếp theo;
7. Thuật toán *VNS* kết thúc khi điều kiện dừng được thỏa; điều kiện dừng được lựa chọn trong thuật toán này là 10**n*, với *n* là số đỉnh của đồ thị.
8. **end while**
9. Trả về lời giải tốt nhất tìm được;

Độ phức tạp của thuật toán *VNS* là: $O(N^2(Elog(E)))$.

## 3.6. THUẬT TOÁN HILL CLIMBING SEARCH GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

### 3.6.1. Ý tưởng thuật toán

Cho đồ thị vô hướng liên thông có trọng số $G$. Bắt đầu từ Cây Steiner $T$ của $G$ được khởi tạo ngẫu nhiên, chèn lần lượt từng cạnh $e \in E(G) - E(T)$ vào Cây Steiner $T$. Nếu Cây Steiner $T$ không chứa chu trình thì cạnh $e$ không cần được xem xét; nếu $E(T) \cup \{e\}$ chứa một chu trình thì tìm một cạnh $e$' trên chu trình này sao cho việc loại nó dẫn đến Cây Steiner $T$' có chi phí là nhỏ nhất. Tiếp theo, nếu $C(T') < C(T)$ thì thay $T$ bằng $T'$. Thuật toán dừng nếu trong một lần duyệt qua tất cả các cạnh $e \in E(G) - E(T)$ mà không cải thiện được chi phí của Cây Steiner $T$.

Tác giả đặt tên thuật toán Hill climbing search giải bài toán *SMT* là *HCSMT*. Thuật toán *HCSMT* ngoài việc lời giải ban đầu được khởi tạo ngẫu nhiên thì các Cây Steiner lân cận tìm được trong quá trình tìm kiếm là kiểu *1-lân cận* tất định. Hiệu quả của thuật toán *HCSMT* có thể được cải thiện khi ta thay đổi thứ tự các cạnh được

duyệt trong tập $E(G) - E(T)$; nghĩa là ta sẽ duyệt tập cạnh này theo một hoán vị được sinh ngẫu nhiên chứ không theo một thứ tự cố định ở tất cả các lần duyệt.

***Kết hợp với tìm kiếm ngẫu nhiên***

Thuật toán *HCSMT* chủ yếu sử dụng tính tăng cường, thể hiện qua các chiến lược tìm kiếm Cây Steiner lân cận. Tính đa dạng được sử dụng vào các thời điểm sau: thứ nhất là khi khởi tạo lời giải ban đầu, thứ hai là thay đổi thứ tự duyệt của các cạnh trong tập cạnh ứng viên (như đã đề cập ở đoạn trên), thứ ba là khi việc tìm kiếm lân cận không cải thiện qua một số lần lặp thì tiến hành chọn ngẫu nhiên một số cạnh của cây để bắt đầu trở lại việc tìm kiếm lân cận.

### 3.6.2. Thuật toán HCSMT

**Algorithm 3.8: *HCSMT* algorithm**

**Input:** Đồ thị $G = (V(G), E(G))$;

**Output:** Cây Steiner có chi phí nhỏ nhất tìm được $T_{best}$;

1. $stop$ = false;
2. $d$ = 0; //$d$ là số lần tăng cường
3. $T$ là Cây Steiner được khởi tạo ngẫu nhiên;
4. $T_{best}$ = $T$; //lưu lại Cây Steiner tốt nhất trước khi thực hiện chiến lược đa dạng hóa
5. **while** ($d$ < $N$) **do** //$N$ là số lần lặp định trước
6. $stop$ = true;
7. $S = E - E(T)$;
8. **for** (với mỗi cạnh $e \in S$ ) **do**
9. min = +∞;
10. **if** (cạnh $e$ có 2 đỉnh $(u, v) \in T$) **then**
11. $F = T \cup \{e\}$;

12. Xác định chu trình Cycle trong $F$ chứa $e$;
13. **for** (với mỗi cạnh $e' \in$ Cycle) **do**
14. **if** ($C(F - e') <$ min) **then**
15. $min = C(F - e')$;
16. Ghi nhận $T' = F - e'$;
17. **end if**
18. **end for**
19. **if** ($min < C(T)$) **then**
20. $T = T'$;
21. $stop$ = false;
22. **end if**
23. **end if**
24. **end for**
25. **if** ($stop$) **then**
26. $d$++; //tăng số biến đếm tăng cường
27. **if** ($C(T) < C(T_{best})$) $T_{best} = T$;
28. **while** (*Thỏa điều kiện đa dạng hóa*) **do**
29. Loại bỏ một số cạnh ngẫu nhiên của Cây Steiner đang xét;
30. Chọn ngẫu nhiên một số cạnh trong các cạnh còn lại của đồ thị $G$ cho đến khi cây $T$ liên thông trở lại (điều kiện cạnh thêm vào là: Phải có đỉnh thuộc $T$ mới được thêm vào để tránh trường hợp thêm vào quá nhiều lần nhưng $T$ không liên thông trở lại). Sử dụng định nghĩa *k-lân cận* ở trên;
31. Nếu thêm quá nhiều lần mà không làm cho $T$ liên thông trở lại, thì tiếp tục đa dạng hóa lại với các cạnh khác;

```
        end while
    end if
end while
return T_best;
```

Độ phức tạp của thuật toán HCSMT là: *O(N(Elog (E) + EV)).*

## 3.7. THỰC NGHIỆM VÀ ĐÁNH GIÁ CÁC THUẬT TOÁN METAHEURISTIC GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

### 3.7.1. Thuật toán Bees-Steiner

Phần này, luận án trình bày kết quả thực nghiệm thuật toán *Bees*-Steiner giải bài toán *SMT* trong trường hợp đồ thị thưa và đưa ra một số phân tích đánh giá về kết quả đạt được. Kết quả thực nghiệm thuật toán *Bees*-Steiner được so sánh với hai heuristic giải bài toán *SMT* gồm: Heuristic *MST*-Steiner của Bang Ye Wu và Kun-Mao Chao, heuristic dựa vào cây đường đi ngắn nhất (*SPT*-Steiner) và hai thuật toán metaheuristic giải bài toán *SMT* gồm: Thuật toán di truyền song song (*PGA*-Steiner) và thuật toán tìm kiếm tabu (Tabu-Steiner).

***- Môi trường thực nghiệm***

Thuật toán *Bees-Steiner* được cài đặt bằng ngôn ngữ C++, sử dụng môi trường DEV C++ 5.9.2; được thực nghiệm trên một máy chủ ảo, Hệ điều hành Windows server 2008 R2 Enterprise, 64bit, Intel(R) Xeon (R) CPU E5-2660 0 @ 2.20 GHz, RAM 4GB.

***- Tham số thực nghiệm***

Qua thực nghiệm, thuật toán *Bees-Steiner* sử dụng bộ tham số sau: Số bước lặp $Imax = 300$, $N = 75$, $h = 0.35 \times N$, $p = 0.85 \times N$, $k_1 = 0.50 \times n$, $k_2 = 0.25 \times n$, $k_3 = 0.01 \times n$. Thuật toán *Bees-Steiner* thực hiện 30 lần cho mỗi bộ dữ liệu.

***- Chất lượng lời giải***

Kết quả thực nghiệm các thuật toán được ghi nhận ở Bảng 3.1 và Bảng 3.2. Theo đó, kết quả thực nghiệm của thuật toán *PGA*-Steiner và Tabu-Steiner được tác giả ghi nhận lại từ các công trình có liên quan đã công bố; kết quả của các thuật toán *MST*-Steiner, *SPT*-Steiner và Bees-Steiner là do tác giả cài đặt; với mỗi đồ thị, kết quả của thuật toán Bees-Steiner là kết quả tốt nhất (có chi phí nhỏ nhất) sau 30 lần chạy.

Bảng 3.1. Kết quả thực nghiệm thuật toán trên các đồ thị thuộc nhóm *steinb*

| **Test** | **MST-Steiner** | **SPT-Steiner** | **PGA-Steiner** | **Bees-Steiner** |
|---|---|---|---|---|
| steinb1 | 82 | 82 | 82 | 82 |
| steinb2 | 90 | 84 | 83 | 83 |
| steinb3 | 140 | 147 | 138 | 138 |
| steinb4 | 64 | 59 | 59 | 59 |
| steinb5 | 64 | 62 | 61 | 61 |
| steinb6 | 128 | 134 | 122 | 122 |
| steinb7 | 111 | 111 | 111 | 111 |
| steinb8 | 104 | 113 | 104 | 104 |
| steinb9 | 222 | 222 | 220 | 220 |
| steinb10 | 98 | 90 | 86 | 86 |
| steinb11 | 91 | 93 | 88 | 88 |
| steinb12 | 174 | 192 | 174 | 174 |
| steinb13 | 175 | 172 | 165 | 165 |
| steinb14 | 237 | 253 | 235 | 235 |
| steinb15 | 323 | 335 | 318 | 318 |
| steinb16 | 137 | 138 | 127 | 127 |
| steinb17 | 134 | 139 | 131 | 132 |
| steinb18 | 222 | 250 | 218 | 219 |

Với các đồ thị nhóm *steinb*, chất lượng lời giải tốt hơn, bằng và kém hơn của thuật toán Bees-Steiner so với các thuật toán: *MST*-Steiner, *SPT*-Steiner, *PGA*-Steiner lần lượt là: (77.8%; 22.2%; 0.0%), (83.3%; 16.7%; 0.0%), (0.0%; 88.9%; 11.1%) số bộ dữ liệu.

Bảng 3.2. Kết quả thực nghiệm thuật toán trên các đồ thị thuộc nhóm *steinc*

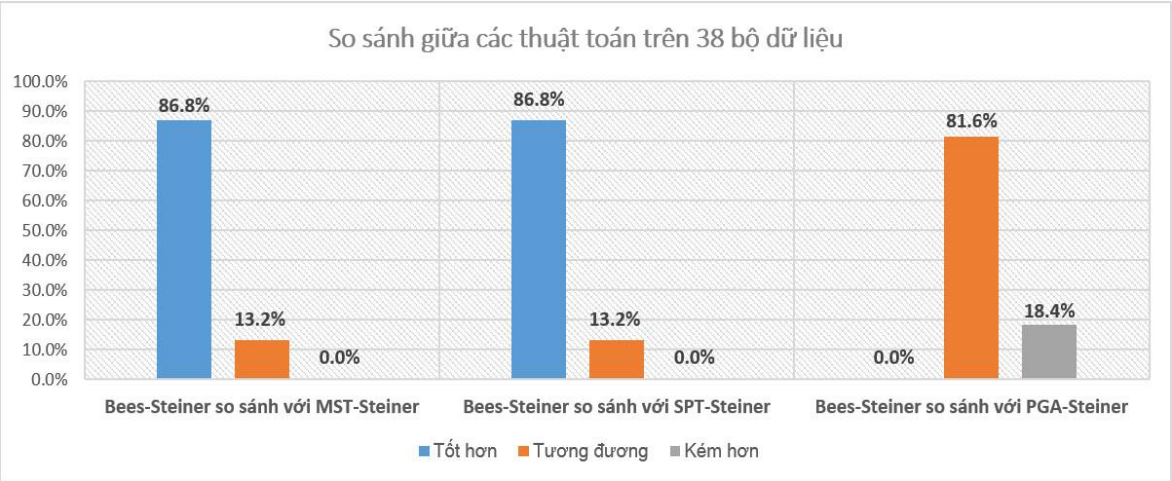
| **Test** | ***MST*-Steiner** | ***SPT*-Steiner** | **Tabu-Steiner** | ***PGA*-Steiner** | **Bees-Steiner** |
|---|---|---|---|---|---|
| steinc01 | 88 | 86 | 85 | 85 | 85 |
| steinc02 | 144 | 158 | 144 | 144 | 144 |
| steinc03 | 779 | 843 | 755 | 754 | 754 |
| steinc04 | 1114 | 1193 | 1081 | 1079 | 1079 |
| steinc05 | 1599 | 1706 | 1579 | 1579 | 1579 |
| steinc06 | 60 | 56 | 55 | 55 | 55 |
| steinc07 | 115 | 103 | 102 | 102 | 102 |
| steinc08 | 531 | 597 | 509 | 509 | 509 |
| steinc09 | 728 | 865 | 708 | 707 | 707 |
| steinc10 | 1117 | 1327 | 1093 | 1093 | 1094 |
| steinc11 | 37 | 32 | 32 | 32 | 32 |
| steinc12 | 49 | 46 | 46 | 46 | 46 |
| steinc13 | 274 | 322 | 258 | 258 | 260 |
| steinc14 | 337 | 417 | 324 | 323 | 324 |
| steinc15 | 571 | 703 | 556 | 556 | 556 |
| steinc16 | 13 | 12 | 11 | 11 | 11 |
| steinc17 | 19 | 19 | 18 | 18 | 18 |

| | | | | | |
|---|---|---|---|---|---|
| steinc18 | 125 | 146 | 117 | 113 | 116 |
| steinc19 | 158 | 195 | 149 | 146 | 149 |
| steinc20 | 269 | 339 | 267 | 267 | 267 |

Với các đồ thị nhóm *steinc*, chất lượng lời giải tốt hơn, bằng và kém hơn của thuật toán Bees-Steiner so với các thuật toán: *MST*-Steiner, *SPT*-Steiner, Tabu-Steiner và *PGA*-Steiner lần lượt là: (95.0%; 5.0%; 0.0%), (90.0%; 10.0%; 0.0%), (20.0%; 70.0%; 10.0%), (0.0%; 75.0%; 25.0%) số bộ dữ liệu.

Đánh giá chung trên tổng số 38 bộ dữ liệu, thuật toán Bees-Steiner cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *MST*-Steiner lần lượt là: (86.8%, 13.2%, 0.0%) số bộ dữ liệu; thuật toán Bees-Steiner cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *SPT*-Steiner lần lượt là: (86.8%, 13.2%, 0.0%) số bộ dữ liệu; thuật toán Bees-Steiner cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *PGA*-Steiner lần lượt là: (0.0%, 81.6%, 18.4%) số bộ dữ liệu.

Kết quả so sánh giữa thuật toán Bees-Steiner với các thuật toán: *MST*-Steiner, *SPT*-Steiner và *PGA*-Steiner trên tổng số 38 bộ dữ liệu được minh họa bằng biểu đồ như Hình 3.1.




Hình 3.1. So sánh giữa các thuật toán trên 38 bộ dữ liệu

Các kết quả chính của đề xuất này được công bố tại Công trình [CT1].

### 3.7.2. Thuật toán tìm kiếm lân cận biến đổi

Thuật toán tìm kiếm lân cận biến đổi (Variable neighborhood search algorithm - *VNS*) được cài đặt bằng ngôn ngữ C++, sử dụng môi trường DEV C++ 5.9.2 và được thực nghiệm trên một máy chủ ảo, Hệ điều hành Windows server 2008 R2 Enterprise, 64bit, Intel(R) Xeon (R) CPU E5-2660 0 @ 2.20 GHz, RAM 4GB.

Kết quả thực nghiệm của thuật toán VNS được ghi nhận ở Bảng 3.3.

Bảng 3.3. Kết quả thực nghiệm thuật toán *VNS*

| **Test** | ***VNS*** | **Test** | ***VNS*** | **Test** | ***VNS*** | **Test** | ***VNS*** |
|---|---|---|---|---|---|---|---|
| steinb01 | 82 | steinc01 | 85 | steind01 | 106 | steine01 | 111 |
| steinb02 | 83 | steinc02 | 144 | steind02 | 220 | steine02 | 214 |
| steinb03 | 138 | steinc03 | 754 | steind03 | 1565 | steine03 | 4015 |
| steinb04 | 59 | steinc04 | 1079 | steind04 | 1935 | steine04 | 5101 |
| steinb05 | 61 | steinc05 | 1579 | steind05 | 3250 | steine05 | 8128 |
| steinb06 | 122 | steinc06 | 55 | steind06 | 67 | steine06 | 73 |
| steinb07 | 111 | steinc07 | 102 | steind07 | 103 | steine07 | 145 |
| steinb08 | 104 | steinc08 | 509 | steind08 | 1073 | steine08 | 2648 |
| steinb09 | 220 | steinc09 | 707 | steind09 | 1448 | steine09 | 3608 |
| steinb10 | 86 | steinc10 | 1093 | steind10 | 2111 | steine10 | 5600 |
| stein0b11 | 88 | steinc11 | 33 | steind11 | 29 | steine11 | 34 |
| stein0b12 | 174 | steinc12 | 46 | steind12 | 42 | steine12 | 67 |
| stein0b13 | 165 | steinc13 | 258 | steind13 | 502 | steine13 | 1292 |
| stein0b14 | 236 | steinc14 | 323 | steind14 | 671 | steine14 | 1735 |
| stein0b15 | 318 | steinc15 | 556 | steind15 | 1116 | steine15 | 2784 |
| stein0b16 | 127 | steinc16 | 11 | steind16 | 13 | steine16 | 15 |
| stein0b17 | 131 | steinc17 | 18 | steind17 | 23 | steine17 | 25 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| stein0b18 | 218 | steinc18 | 115 | steind18 | 228 | steine18 | 583 |
| | | steinc19 | 148 | steind19 | 318 | steine19 | 768 |
| | | steinc20 | 268 | steind20 | 538 | steine20 | 1342 |

Luận án so sánh chất lượng lời giải của thuật toán *VNS* với nhóm các thuật toán heuristic: *MST*-Steiner [14], *SPT*-Steiner [CT6], *PD*-Steiner [CT6] và nhóm các thuật toán metaheuristic: Node-based [69] và Path-based [69].

Với 18 bộ dữ liệu nhóm *steinb*, thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *MST*-Steiner lần lượt là: (77.8%, 22.2%, 0.0%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *SPT*-Steiner lần lượt là: (83.3%, 16.7%, 0.0%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *PD*-Steiner lần lượt là: (50.0%, 44.4%, 5.6%) số bộ dữ liệu.

Với 20 bộ dữ liệu nhóm *steinc*, thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán Node-based lần lượt là: (5%, 80%, 15%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán Path-based lần lượt là: (10%, 85%, 5%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *MST*-Steiner lần lượt là: (95.0%, 5.0%, 0.0%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *SPT*-Steiner lần lượt là: (90.0%, 5.0%, 5.0%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *PD*-Steiner lần lượt là: (75.0%, 25.0%, 0.0%) số bộ dữ liệu.

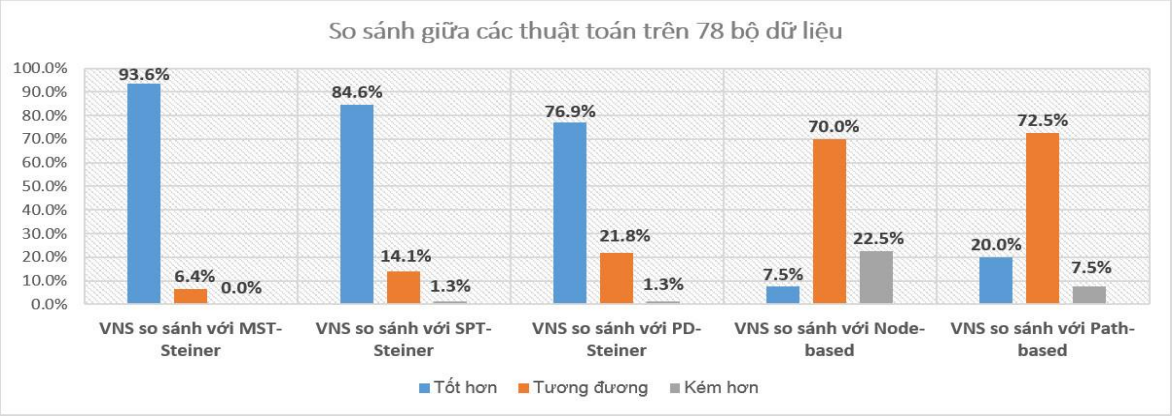
Với 20 bộ dữ liệu nhóm *steind*, thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán Node-based lần lượt là: (10%, 60%, 30%) số bộ dữ liệu; thuật toán VNS cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán Path-based lần lượt là: (30%, 60%, 10%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *MST*-Steiner lần lượt là: (100.0%, 0.0%, 0.0%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *SPT*-Steiner lần lượt là: (90.0%, 10.0%,

0.0%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *PD*-Steiner lần lượt là: (85.0%, 15.0%, 0.0%) số bộ dữ liệu.

Với 20 bộ dữ liệu nhóm *steine*, thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *MST*-Steiner lần lượt là: (100.0%, 0.0%, 0.0%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *SPT*-Steiner lần lượt là: (75.0%, 25.0%, 0.0%) số bộ dữ liệu; thuật toán *VNS* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *PD*-Steiner lần lượt là: (95.0%, 5.0%, 0.0%) số bộ dữ liệu.

Đánh giá chung trên 78 bộ dữ liệu là các đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn; thuật toán *VNS* cho lời giải chất lượng tốt hơn, tương đương và kém hơn thuật toán *MST*-Steiner lần lượt là: (93.6%, 6.4%, 0.0%) số bộ dữ liệu; thuật toán *VNS* cho lời giải chất lượng tốt hơn, tương đương và kém hơn thuật toán *SPT*-Steiner lần lượt là: (84.6%, 14.1%, 1.3%) số bộ dữ liệu; thuật toán *VNS* cho lời giải chất lượng tốt hơn, tương đương và kém hơn thuật toán *PD*-Steiner lần lượt là: (76.9%, 21.8%, 1.3%) số bộ dữ liệu; thuật toán *VNS* cho lời giải chất lượng tốt hơn, tương đương và kém hơn thuật toán Node-based lần lượt là: (7.5%, 70.0%, 22.5%) số bộ dữ liệu; thuật toán *VNS* cho lời giải chất lượng tốt hơn, tương đương và kém hơn thuật toán Path-based lần lượt là: (20.0%. 72.5%, 7.5%) số bộ dữ liệu.

Kết quả so sánh giữa thuật toán *VNS* với các thuật toán: *MST*-Steiner, *SPT*-Steiner, *PD*-Steiner, Node-based và Path-based trên tổng số 78 bộ dữ liệu được minh họa bằng biểu đồ như Hình 3.2.




Hình 3.2. So sánh giữa các thuật toán trên 78 bộ dữ liệu

Kết quả nghiên cứu chính của đề xuất này được công bố tại Công trình [CT2].

### 3.7.3. Thuật toán Hill Climbing Search

Trong phần này, luận án mô tả việc thực nghiệm thuật toán Hill climbing search - *HCSMT* và đưa ra một số so sánh, đánh giá về các kết quả đạt được của thuật toán *HCSMT* so với các thuật toán: *Heu* [77], *PD* [CT6], *VNS* [CT2], *TS* [15].

Kết quả thực nghiệm của thuật toán *HCSMT* được ghi nhận ở các Bảng 3.4, 3.5, 3.6. Các bảng này có cấu trúc như sau: Cột đầu tiên (Test) là tên các bộ dữ liệu trong hệ thống dữ liệu thực nghiệm của từng đồ thị; các cột tiếp theo ghi nhận giá trị chi phí Cây Steiner lần lượt ứng với các thuật toán: *Heu* [77], *PD* [CT6], *VNS* [CT2], *TS* [15] và thuật toán *HCSMT*.

Thuật toán *HCSMT* được cài đặt bằng ngôn ngữ C++, sử dụng môi trường DEV C++ 5.9.2; được thực nghiệm trên một máy chủ ảo, Hệ điều hành Windows server 2008 R2 Enterprise, 64bit, Intel(R) Xeon (R) CPU E5-2660 0 @ 2.20 GHz, RAM 4GB.

Bộ tham số được xác định như sau qua thực nghiệm: Số lần chạy mỗi bộ dữ liệu là 30, số lần tăng cường là 50, số cạnh loại bỏ ngẫu nhiên của mỗi lần tăng cường là $0.05 \times |E(T)|$

Bảng 3.4. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steinc*

| **Test** | ***Heu*** | ***PD*** | ***VNS*** | ***TS*** | ***HCSMT*** |
|---|---|---|---|---|---|
| steinc01 | 85 | 85 | 85 | 85 | 85 |
| steinc02 | 144 | 144 | 144 | 144 | 144 |
| steinc03 | 755 | 762 | 754 | 754 | 754 |
| steinc04 | 1080 | 1085 | 1079 | 1079 | 1080 |
| steinc05 | 1579 | 1583 | 1579 | 1579 | 1579 |
| steinc06 | 55 | 55 | 55 | 55 | 55 |
| steinc07 | 102 | 102 | 102 | 102 | 102 |

| | | | | | |
|---|---|---|---|---|---|
| steinc08 | 510 | 516 | 509 | 509 | 509 |
| steinc09 | 715 | 718 | 707 | 707 | 707 |
| steinc10 | 1093 | 1107 | 1093 | 1093 | 1093 |
| steinc11 | 32 | 34 | 33 | 32 | 32 |
| steinc12 | 46 | 48 | 46 | 46 | 46 |
| steinc13 | 262 | 268 | 258 | 258 | 258 |
| steinc14 | 324 | 332 | 323 | 324 | 324 |
| steinc15 | 557 | 562 | 556 | 556 | 557 |
| steinc16 | 11 | 12 | 11 | 11 | 11 |
| steinc17 | 19 | 20 | 18 | 18 | 18 |
| steinc18 | 120 | 123 | 115 | 117 | 115 |
| steinc19 | 150 | 159 | 148 | 148 | 149 |
| steinc20 | 268 | 268 | 268 | 267 | 268 |

Bảng 3.5. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steind*

| **Test** | ***Heu*** | ***PD*** | ***VNS*** | ***TS*** | ***HCSMT*** |
|---|---|---|---|---|---|
| steind01 | 106 | 107 | 106 | 106 | 106 |
| steind02 | 220 | 228 | 220 | 220 | 220 |
| steind03 | 1570 | 1771 | 1565 | 1567 | 1565 |
| steind04 | 1936 | 2174 | 1935 | 1935 | 1936 |
| steind05 | 3252 | 3511 | 3250 | 3250 | 3250 |
| steind06 | 70 | 70 | 67 | 70 | 67 |
| steind07 | 103 | 111 | 103 | 103 | 103 |
| steind08 | 1092 | 1287 | 1073 | 1078 | 1073 |

| | | | | | |
|---|---|---|---|---|---|
| steind09 | 1462 | 1773 | 1448 | 1450 | 1448 |
| steind10 | 2113 | 2550 | 2111 | 2112 | 2113 |
| steind11 | 29 | 29 | 29 | 30 | 29 |
| steind12 | 42 | 44 | 42 | 42 | 42 |
| steind13 | 510 | 643 | 502 | 502 | 507 |
| steind14 | 675 | 851 | 671 | 667 | 674 |
| steind15 | 1120 | 1437 | 1116 | 1117 | 1118 |
| steind16 | 13 | 13 | 13 | 13 | 13 |
| steind17 | 23 | 25 | 23 | 23 | 23 |
| steind18 | 238 | 301 | 228 | 230 | 231 |
| steind19 | 325 | 424 | 318 | 315 | 321 |
| steind20 | 539 | 691 | 538 | 538 | 539 |

Bảng 3.6. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steine*

| **Test** | ***Heu*** | ***PD*** | ***VNS*** | ***TS*** | ***HCSMT*** |
|---|---|---|---|---|---|
| steine01 | 111 | 111 | 111 | 111 | 111 |
| steine02 | 214 | 214 | 214 | 216 | 214 |
| steine03 | 4052 | 4570 | 4015 | 4018 | 4015 |
| steine04 | 5114 | 5675 | 5101 | 5105 | 5101 |
| steine05 | 8130 | 8976 | 8128 | 8128 | 8130 |
| steine06 | 73 | 73 | 73 | 73 | 73 |
| steine07 | 149 | 150 | 145 | 149 | 145 |
| steine08 | 2686 | 3254 | 2648 | 2649 | 2648 |
| steine09 | 3656 | 4474 | 3608 | 3605 | 3608 |

| | | | | | |
|---|---|---|---|---|---|
| steine10 | 5614 | 6847 | 5600 | 5602 | 5600 |
| steine11 | 34 | 34 | 34 | 34 | 34 |
| steine12 | 68 | 68 | 67 | 68 | 68 |
| steine13 | 1312 | 1704 | 1292 | 1299 | 1312 |
| steine14 | 1752 | 2304 | 1735 | 1740 | 1735 |
| steine15 | 2792 | 3626 | 2784 | 2784 | 2799 |
| steine16 | 15 | 15 | 15 | 15 | 15 |
| steine17 | 26 | 27 | 25 | 25 | 25 |
| steine18 | 608 | 804 | 583 | 595 | 594 |
| steine19 | 788 | 1059 | 768 | 778 | 768 |
| steine20 | 1349 | 1753 | 1342 | 1352 | 1342 |

Nội dung của Bảng 3.7 cho biết số lượng (***SL***) và phần trăm (***%:*** *tỷ lệ phần trăm trên tổng số bộ dữ liệu*) tương ứng với số lượng bộ dữ liệu cho chất lượng lời giải tốt hơn (*ghi nhận bởi ký hiệu "<"*) hoặc cho chất lượng lời giải tương đương (*ghi nhận bởi ký hiệu "="*) hoặc cho chất lượng lời giải kém hơn (*ghi nhận bởi ký hiệu ">"*) khi so sánh thuật toán *HCSMT* với các thuật toán: *Heu*, *PD*, *VNS* và *TS*.

Với 20 bộ dữ liệu nhóm *steinc*; thuật toán *HCSMT* cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *Heu* lần lượt là: (35.0%, 65.0%, 0.0%) số bộ dữ liệu. Kết quả so sánh thuật toán *HCSMT* với thuật toán *Heu* trên các nhóm dữ liệu *steind, steine* cũng được thể hiện chi tiết ở Bảng 3.7.

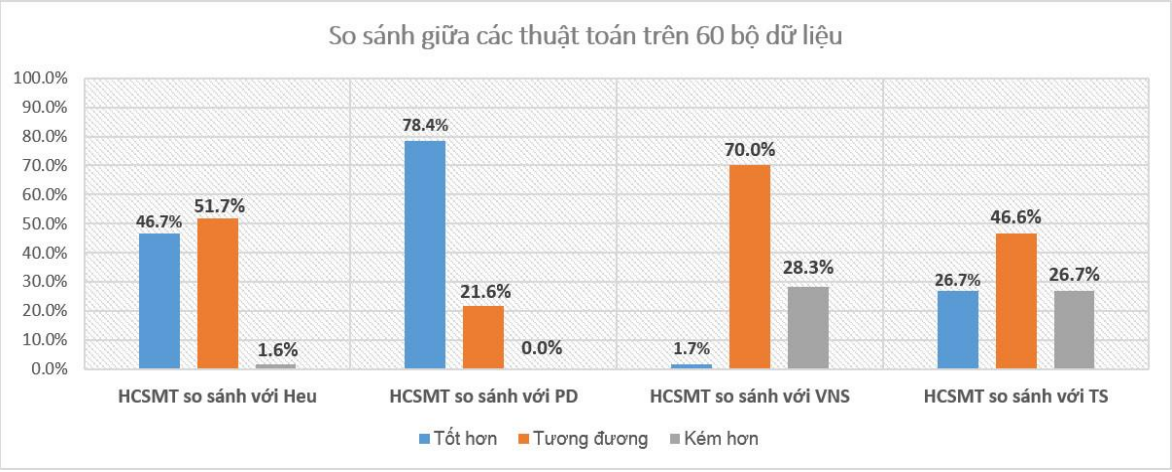
Bảng 3.7. So sánh kết quả thuật toán *HCSMT* với các thuật toán: *Heu*, *PD*, *VNS* và *TS*

| **So sánh chất lượng các thuật toán** | **steinc** | | **steind** | | **steine** | | **Tổng cộng** | |
|---|---|---|---|---|---|---|---|---|
| | ***SL*** | **%** | ***SL*** | **%** | ***SL*** | **%** | ***SL*** | **%** |
| *HCSMT < Heu* | 7 | 35% | 10 | 50% | 11 | 55% | 28 | 46.7% |
| *HCSMT = Heu* | 13 | 65% | 10 | 50% | 8 | 40% | 31 | 51.7% |
| *HCSMT > Heu* | 0 | 0% | 0 | 0% | 1 | 5% | 1 | 1.6% |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *HCSMT < PD* | 15 | 75% | 18 | 90% | 14 | 70% | 47 | 78.4% |
| *HCSMT = PD* | 5 | 25% | 2 | 10% | 6 | 30% | 13 | 21.6% |
| *HCSMT > PD* | 0 | 0% | 0 | 0% | 0 | 0% | 0 | 0% |
| *HCSMT < VNS* | 1 | 5% | 0 | 0% | 0 | 0% | 1 | 1.7% |
| *HCSMT = VNS* | 15 | 75% | 12 | 60% | 15 | 75% | 42 | 70.0% |
| *HCSMT > VNS* | 4 | 20% | 8 | 40% | 5 | 25% | 17 | 28.3% |
| *HCSMT < TS* | 1 | 5% | 5 | 25% | 10 | 50% | 16 | 26.7% |
| *HCSMT = TS* | 15 | 75% | 7 | 35% | 6 | 30% | 28 | 46.6% |
| *HCSMT > TS* | 4 | 20% | 8 | 40% | 4 | 20% | 16 | 26.7% |

Đánh giá chung trên tổng số 60 bộ dữ liệu; thuật toán *HCSMT* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *Heu* lần lượt là: (46.7%, 51.7%, 1.6%) số bộ dữ liệu; thuật toán *HCSMT* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *PD* lần lượt là: (78.4%, 21.6%, 0.0%) số bộ dữ liệu; thuật toán *HCSMT* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *VNS* lần lượt là: (1.7%, 70%, 28.3%) số bộ dữ liệu; thuật toán *HCSMT* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *TS* lần lượt là: (26.7%, 46.6%, 26.7%) số bộ dữ liệu.

Kết quả so sánh giữa thuật toán *HCSMT* với các thuật toán: *Heu*, *PD*-Steiner, *VNS* và *TS* trên tổng số 60 bộ dữ liệu được minh họa bằng biểu đồ như Hình 3.3.




Hình 3.3. So sánh giữa các thuật toán trên 60 bộ dữ liệu

Thời gian tính trung bình của thuật toán *HCSMT* trên các đồ thị ứng với nhóm đồ thị *steinc* là 2.54 giây, tương tự với nhóm đồ thị *steind* là 16.78 giây, với nhóm đồ thị *steine* là 214.81 giây. Thời gian chạy của thuật toán *HCSMT* nêu trên, cũng cung cấp thêm thông tin tham khảo cần thiết về thuật toán này.

Đóng góp chính của luận án là đã đề xuất cách thức tìm kiếm lân cận kết hợp với tìm kiếm lân cận ngẫu nhiên nhằm nâng cao chất lượng của thuật toán. Luận án đã thực nghiệm thuật toán đề xuất trên hệ thống gồm 60 bộ dữ liệu thực nghiệm chuẩn. Kết quả thực nghiệm cho thấy rằng, thuật toán *HCSMT* cho chất lượng lời giải tốt hơn hoặc bằng hai thuật toán dạng heuristic và cho chất lượng lời giải tốt hơn, bằng, kém hơn hai thuật toán dạng metaheuristic tốt hiện biết trên một số bộ dữ liệu thực nghiệm chuẩn.

Kết quả nghiên cứu chính thuật toán tìm kiếm leo đồi được nghiên cứu sinh công bố tại Công trình [CT3].

## 3.8. ĐÁNH GIÁ CÁC THUẬT TOÁN THÔNG QUA ĐỘ PHỨC TẠP

Bảng 3.8 ghi nhận kết quả độ phức tạp thời gian của các thuật toán metaheuristic đề xuất. Thông qua bảng này ta thấy: Thuật toán *Bees*-Steiner có độ phức tạp thời gian lớn nhất, thuật toán *HCSMT* có độ phức tạp thời gian nhỏ nhất, qua đó cung cấp thêm thông tin về tính hiệu quả của các thuật toán đề xuất.

Bảng 3.8. Độ phức tạp thời gian của các thuật toán

| **Thuật toán** | **Độ phức tạp thời gian** |
|---|---|
| Bees-Steiner | $O(N(Nlog(N) + V(hk_1 + (p - h)k_2 + (N - p)k_3))))$ |
| VNS | $O(N^2(Elog(E)))$ |
| HCSMT | $O(N(Elog\ (E) + EV))$ |
| Các thuật toán được xếp theo độ phức tạp thời gian giảm dần: | ***Bees-Steiner, VNS, HCSMT*** |

***Chú thích**:*

*V*: số đỉnh của đồ thị *G*; *E*: số cạnh của đồ thị *G*; *h*: số cá thể của nhóm 1; *p*: tổng số cá thể *nhóm 1* và *nhóm 2*; *k1, k2, k3*: số lần thực hiện tìm kiếm lân cận của mỗi cá thể trong các *nhóm 1, 2, 3*; *N* (thuật toán Bees-Steiner): số cá thể trong quần thể; *N* (trong thuật toán VNS và HCSMT): số lần lặp định trước, dùng làm điều kiện dừng.

## 3.9. KẾT LUẬN CHƯƠNG 3

Trong chương cuối này, luận án đã tập trung giải quyết các vấn đề sau:

- Đề xuất được ba thuật toán metaheuristic giải bài toán *SMT* đó là: thuật toán *Bees*-Steiner, thuật toán tìm kiếm lân cận biến đổi *VNS* và thuật toán tìm kiếm leo đồi Hill climbing search *(HCSMT)*. Ngoài ra, luận án cũng đề xuất thêm 2 chiến lược tìm kiếm lân cận: Tham lam và có xác suất, đồng thời sử dụng chúng trong lược đồ thuật toán tìm kiếm lân cận biến đổi, nhằm nâng cao hơn nữa chất lượng lời giải cho các thuật toán metaheuristic.

- Các thuật toán metaheuristic đề xuất mới được cài đặt thực nghiệm trên hệ thống dữ liệu thực nghiệm chuẩn và so sánh hiệu quả với các thuật toán metaheuristic khác hiện biết. Kết quả cho thấy các thuật toán metaheuristic đề xuất mới cho chất lượng lời giải tốt hơn các thuật toán metaheuristic công bố trước đó trên một số bộ dữ liệu. Chất lượng thuật toán metaheuristic phụ thuộc chủ yếu vào các chiến lược tìm kiếm lân cận.

Các thuật toán metaheuristic mà luận án nghiên cứu phát triển trong chương cuối này, giúp làm rõ hơn về các trường hợp bài toán Cây Steiner nhỏ nhất có thể giải được một cách hữu hiệu. Từ đó, giúp nâng cao khả năng ứng dụng của bài toán *SMT* trong thực tiễn nói chung và trong thiết kế hệ thống mạng nói riêng.

# KẾT LUẬN

Bài toán Cây Steiner nhỏ nhất là bài toán tối ưu tổ hợp trên đồ thị đã được chứng minh thuộc lớp *NP-hard* và không thể giải trong thời gian đa thức. Vì vậy, bài toán *SMT* luôn là thách thức lớn đối với các nhà khoa học, nhằm nghiên cứu tìm ra lời giải hiệu quả nhất. Do là bài toán tối ưu nên luôn mang tính cấp thiết và có giá trị thực tiễn cao, có rất nhiều ứng dụng trong khoa học kỹ thuật nói chung, trong đó có thiết kế hệ thống mạng nói riêng. Hiện tại, có rất nhiều hướng tiếp cận giải bài toán *SMT*, có thể chia thành hai nhóm như sau: Giải chính xác, đối với những bài toán có kích thước nhỏ; và giải gần đúng đối với những bài toán có kích thước lớn, trong đó có phát triển các thuật toán heuristic và metaheuristic để giải bài toán này. Vì vậy bài toán *SMT* đòi hỏi phải được nghiên cứu một cách đa chiều và toàn diện. Từ các nghiên cứu đó, phạm vi ứng dụng của bài toán sẽ được chỉ ra một cách rõ ràng và cụ thể hơn trong thực tế.

Dựa trên phương pháp nghiên cứu được sử dụng như thông qua một số cơ sở lý thuyết toán học, xây dựng phát triển thuật toán và cài đặt thực nghiệm để phân tích, so sánh đánh giá, kết hợp với các công cụ thống kê. Các kết quả chính của luận án được trình bày trong Chương 2 và Chương 3

## 1. Các đóng góp chính của luận án

*1.1. Đề xuất hai thuật toán heuristic mới giải bài toán Cây Steiner nhỏ nhất trong trường hợp đồ thị thưa, cụ thể là:*

- ***Đóng góp thứ nhất***: Đề xuất thuật toán heuristic *PD*-Steiner giải bài toán *SMT* trong trường hợp đồ thị thưa. Thuật toán *PD*-Steiner được cài đặt thực nghiệm và đánh giá hiệu quả so với heuristic *MST*-Steiner [14] trên 98 bộ dữ liệu, kết quả cho thấy: Thuật toán *PD*-Steiner cho chất lượng lời giải tốt hơn, bằng, kém hơn thuật toán *MST*-Steiner lần lượt là: (86.7%, 10.2%, 3.1%) số bộ dữ liệu.

- ***Đóng góp thứ hai***: Đề xuất thuật toán heuristic *SPT*-Steiner giải bài toán *SMT* trong trường hợp đồ thị thưa. Thuật toán *SPT*-Steiner được cài đặt thực nghiệm và đánh giá hiệu quả so với heuristic *MST*-Steiner [14] trên 98 bộ dữ liệu, kết quả cho thấy: Thuật toán *SPT*-Steiner cho chất lượng lời giải tốt hơn, bằng, kém hơn so với thuật toán *MST*-Steiner lần lượt là: (26.5%, 7.1%, 66.3%) số bộ dữ liệu.

*1.2. Đề xuất hai thuật toán heuristic cải tiến giải bài toán Cây Steiner nhỏ nhất trong trường hợp đồ thị thưa kích thước lớn lên đến 100000 đỉnh, cụ thể là:*

- ***Đóng góp thứ ba***: Đề xuất thuật toán heuristic cải tiến *i-SPT*-Steiner giải bài toán *SMT* trong trường hợp đồ thị thưa kích thước lớn. Thuật toán *i-SPT*-Steiner được cài đặt thực nghiệm và đánh giá hiệu quả so với heuristic *MST*-Steiner [14] trên 80 bộ dữ liệu, kết quả cho thấy: Thuật toán *i-SPT*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner là: (30.0%, 2.5%, 67.5%) số bộ dữ liệu.

- ***Đóng góp thứ tư***: Đề xuất thuật toán heuristic cải tiến *i-PD*-Steiner giải bài toán *SMT* trong trường hợp đồ thị thưa kích thước lớn. Thuật toán *i-PD*-Steiner được cài đặt thực nghiệm và đánh giá hiệu quả so với heuristic *MST*-Steiner [14] trên 80 bộ dữ liệu, kết quả cho thấy: Thuật toán *i-PD*-Steiner cho chất lượng lời giải tốt hơn, tương đương, kém hơn thuật toán *MST*-Steiner là: (97.5%, 2.5%, 0.0%) số bộ dữ liệu.

*1.3. Đề xuất ba thuật toán metaheuristic dạng cá thể, quần thể giải bài toán Cây Steiner nhỏ nhất trong trường hợp đồ thị thưa, cụ thể là:*

- ***Đóng góp thứ năm***: Đề xuất thuật toán metaheuristic dạng quần thể *Bees*-Steiner giải bài toán *SMT* trong trường hợp đồ thị thưa. Thuật toán *Bees*-Steiner được cài đặt thực nghiệm và đánh giá hiệu quả so với các heuristic, metaheuristic khác hiện biết trên 38 bộ dữ liệu, kết quả cho thấy: Thuật toán *Bees*-Steiner cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *MST*-Steiner [14], *SPT*-Steiner, *PGA*-Steiner [57] lần lượt là: (86.8%, 13.2%, 0.0%), (86.8%, 13.2%, 0.0%), (0.0%, 81.6%, 18.4%) số bộ dữ liệu.

- ***Đóng góp thứ sáu***: Đề xuất thuật toán metaheuristic dạng cá thể *VNS* giải bài toán *SMT* trong trường hợp đồ thị thưa. Thuật toán *VNS* được cài đặt thực nghiệm và đánh giá hiệu quả so với các heuristic, metaheuristic khác hiện biết trên 78 bộ dữ liệu, kết quả cho thấy: Thuật toán *VNS* cho lời giải chất lượng tốt hơn, tương đương và kém hơn thuật toán *MST*-Steiner [14], *SPT*-Steiner, *PD*-Steiner, Node-based [69], Path-based [69] lần lượt là: (93.6%, 6.4%, 0.0%), (84.6%, 14.1%, 1.3%), (76.9%, 21.8%, 1.3%), (7.5%, 70.0%, 22.5%), (20.0%. 72.5%, 7.5%) số bộ dữ liệu.

- ***Đóng góp thứ bảy***: Đề xuất thuật toán metaheuristic dạng cá thể Hill Climbing Search (*HCSMT*) giải bài toán *SMT* trong trường hợp đồ thị thưa. Thuật toán *HCSMT* được cài đặt thực nghiệm và đánh giá hiệu quả so với các heuristic, metaheuristic khác hiện biết trên 60 bộ dữ liệu, kết quả cho thấy: Thuật toán *HCSMT* cho chất lượng lời giải tốt hơn, tương đương và kém hơn thuật toán *Heu [77]*, *PD*-Steiner, *VNS*, *TS [15]* lần lượt là: (46.7%, 51.7%, 1.6%), (78.4%, 21.6%, 0.0%), (1.7%, 70%, 28.3%), (26.7%, 46.6%, 26.7%) số bộ dữ liệu.

*1.4. Đề xuất các chiến lược tìm kiếm lân cận sử dụng trong lược đồ thuật toán metaheuristic nhằm nâng cao hơn nữa hiệu quả của thuật toán metaheuristic trong việc giải bài toán Cây Steiner nhỏ nhất.*

- ***Đóng góp thứ tám***: Luận án nghiên cứu chi tiết đặc tính của các thuật toán metaheuristic trong việc giải bài toán Cây Steiner nhỏ nhất. Đóng góp mới của luận án là sự tổng hợp các thuật toán trên với các chiến lược tìm kiếm lân cận trên cơ sở đặc tính của bài toán Cây Steiner nhỏ nhất. Hiện tại, luận án là công trình đầu tiên phát triển thuật toán metaheuristic dựa trên lược đồ các thuật toán tiến hóa như: Thuật toán bầy ong cơ bản, thuật toán tìm kiếm leo đồi cơ bản để giải bài toán Cây Steiner nhỏ nhất.

- ***Đóng góp thứ chín***: Dựa trên cơ sở kết quả thực nghiệm các thuật toán được phát triển, luận án đề xuất hướng áp dụng từng thuật toán vào các loại đồ thị, cụ thể như sau: Đối với loại đồ thị thưa kích thước nhỏ, nên chọn các thuật toán metaheuristic dạng cá thể; với các đồ thị thưa kích thước lớn nên chọn thuật toán metaheuristic dạng quần thể, hoặc các thuật toán heuristic. Qua đó làm sâu sắc

thêm, phong phú hơn việc giải hiệu quả bài toán Cây Steiner nhỏ nhất và định hướng ứng dụng cho thiết kế hệ thống mạng.

## 2. Những nội dung nghiên cứu tiếp theo

Hiện nay, các hướng tiếp cận giải chính xác hoặc giải gần đúng bài toán Cây Steiner nhỏ nhất nói chung đang được các nhà khoa học, các tập toàn công nghệ lớn quan tâm, đầu tư nghiên cứu. Do là bài toán tối ưu thuộc lớp *NP-hard*, nên mỗi phương án mới đưa ra có nhiều ưu điểm và ngày càng hiệu quả hơn.

Theo hướng này, trong thời gian tiếp tới luận án sẽ tiếp tục phát triển các nội dung sau:

- Tiếp tục phát triển mới các chiến lược tăng cường hóa, đa dạng hóa lời giải (các chiến lược tìm kiếm lân cận), áp dụng vào sơ đồ các thuật toán đề xuất, nhằm nâng cao hơn nữa chất lượng lời giải cho các thuật toán.

- Tiếp tục cải tiến cấu trúc dữ liệu, kỹ thuật lập trình khi cài đặt thực nghiệm thuật toán với hy vọng rút ngắn thời gian chạy các thuật toán khi làm việc với các đồ thị thưa kích thước lớn (cỡ 100 000 đỉnh).

- Dựa vào sơ đồ các thuật toán đề xuất và các thuật toán hiện biết của tác giả khác, phát triển một thuật toán mới dạng metaheuristic giải quyết một số bài toán cây khung truyền thông tối ưu, hoặc các bài toán định tuyến trên đồ thị thuộc lớp *NP-hard* khác.

- Dựa vào các kết quả nghiên cứu đạt được, xây dựng một phần mềm ứng dụng giải bài toán Cây Steiner nhỏ nhất và áp dụng kết quả đạt được vào thiết kế một hệ thống mạng cụ thể trong thực tế.

# CÁC CÔNG TRÌNH KHOA HỌC ĐÃ CÔNG BỐ

## TẠP CHÍ KHOA HỌC

[CT1] **Trần Việt Chương**, Phan Tấn Quốc, Hà Hải Nam (2017), Thuật toán Bees giải bài toán Cây Steiner nhỏ nhất trong trường hợp đồ thị thưa, *Tạp chí Khoa học Công nghệ thông tin và Truyền thông*, Học viện Công nghệ Bưu chính Viễn thông, Bộ Thông tin và Truyền thông, ISSN: 2525-2224, Số 02 & 03 (CS.01) 2017, trang 24-28.

[CT2] **Tran Viet Chuong**, Ha Hai Nam (2018), A variable neighborhood search algorithm for solving the Steiner minimal tree problem in sparse graphs, *EAI Endorsed Transactions on Context-aware Systems and Applications*, ISSN: 2409-0026, Issue 15, Vol 5, EAI 2018 (https://eudl.eu/issue/casa/5/15), pp. 1-6.

[CT3] **Trần Việt Chương**, Phan Tấn Quốc, Hà Hải Nam (2020), Thuật toán tìm kiếm Hill climbing giải bài toán Cây Steiner nhỏ nhất, *Chuyên san Các công trình nghiên cứu, phát triển và ứng dụng Công nghệ thông tin và Truyền thông, Tạp chí Thông tin và Truyền thông, Bộ Thông tin và Truyền thông*, Tập 2020, Số 1, tháng 6, trang 1-8.

[CT4] **Trần Việt Chương**, Phan Tấn Quốc, Hà Hải Nam (2021), Đề xuất chiến lược tìm kiếm lân cận cho bài toán Cây Steiner nhỏ nhất, *Tạp chí Khoa học Công nghệ thông tin và Truyền thông*, Học viện Công nghệ Bưu chính Viễn thông, Bộ Thông tin và Truyền thông, ISSN: 2525-2224, Số 01 (CS.01) 2021, trang 90-97.

[CT5] **Trần Việt Chương**, Phan Tấn Quốc, Hà Hải Nam (2022), Improving the Heuristic Algorithms to Solve a Steiner-minimal-tree Problem in Large Size Sparse Graphs, *Chuyên san các Công trình nghiên cứu, phát triển và ứng dụng Công nghệ thông tin và Truyền thông, Tạp chí Thông tin và Truyền thông, Bộ Thông tin và Truyền thông*, Tập 2022, Số 1, tháng 3, pp. 43-52.

**HỘI NGHỊ KHOA HỌC**

[CT6] **Trần Việt Chương**, Phan Tấn Quốc, Hà Hải Nam (2017), Đề xuất một số thuật toán Heuristic giải bài toán Cây Steiner nhỏ nhất, *Kỷ yếu Hội nghị Khoa học Công nghệ Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR)*; Trường Đại học Đà Nẵng, ngày 17-18/08/2017, ISBN: 978-604-913-614-6, NXB Khoa học và Kỹ thuật, trang 138-147.

[CT7] **Tran Viet Chuong**, Ha Hai Nam (2018), A Variable Neighborhood Search Algorithm for Solving the Steiner Minimal Tree Problem, *Kỷ yếu Hội nghị khoa học Quốc tế EAI lần thứ 7 - ICCASA và lần thứ 4 - ICTCC 2018*, chủ đề Context-Aware Systems and Applications, and Nature of Computation and Communication, LNICST 266, Springer, pp. 218-225.

[CT8] **Trần Việt Chương**, Phan Tấn Quốc, Hà Hải Nam (2019), Khảo sát một số thuật toán giải bài toán Cây Steiner nhỏ nhất trong trường hợp đồ thị thưa, *Kỷ yếu Hội thảo Quốc gia lần thứ XXII về Một số vấn đề chọn lọc của Công nghệ thông tin và Truyền thông*, chủ đề Chuyển đổi số điều hành Kinh tế - Xã hội trong cách mạng công nghiệp 4.0, do Viện CNTT và Trường Đại học Thái Bình tổ chức ngày 28-29/6/2019 tại TP Thái Bình, NXB Khoa học và Kỹ thuật, trang 132-137.

# TÀI LIỆU THAM KHẢO


**Tiếng Việt**

[1]. Vũ Đình Hòa, "Bài toán Steiner", http://math.ac.vn.

[2]. Phạm Quốc Huy (2009), "Nghiên cứu xây dựng thuật toán thiết kế topology mạng áp dụng cho mạng thế hệ sau NGN", Luận án tiến sĩ, Học viện Công nghệ Bưu chính Viễn thông.

[3]. Phan Quốc Khánh (2006), "Vận Trù Học", Nhà xuất bản Giáo Dục.

[4]. Hoàng Kiếm (2005), "Giải một bài toán trên máy tính như thế nào", Tập 1, 3, Nhà xuất bản Giáo Dục.

[5]. Nguyễn Đức Nghĩa (1999), "Tối ưu hóa - Quy hoạch tuyến tính và rời rạc", Nhà xuất bản Giáo Dục.

[6]. Phan Tấn Quốc, Nguyễn Đức Nghĩa (2013), "Thuật toán bầy ong giải bài toán cây khung với chi phí định tuyến nhỏ nhất". Tạp chí Tin học và điều khiển học, Viện Công nghệ thông tin, Viện Hàn lâm Khoa học và Công nghệ Việt Nam, T.29, S3, pp.265-276.

[7]. Trần Lê Thủy (2014), "Về bài toán Steiner", Viện Toán học, Viện Hàn lâm Khoa học và Công nghệ Việt Nam, (Luận văn Thạc sĩ).

**Tiếng Anh**

[8]. A. Glenstrup (2002), "Optimised Design and Analysis of All-Optical Networks", Ph.D. thesis, Research Center COM, Technical University of Denmark.

[9]. Aaron Kershenbaum (1993), "Telecommunications Network Design Algorithms", McGraw-Hill.

[10]. Alan W. Johnson (1996), "Generalized Hill Climbing Algorithms for Discrete Optimization Problems". Doctor of Philosophy, Industrial and Systems Engineering, Blacksburg, Virginia, pp.1-119.

[11]. Andrew W. Moore (2001), "K-means and Hierarchical Clustering", School of Computer Science Carnegie Mellon University.

[12]. Ankit Anand, Shruti, Kunwar Ambarish Singh (2015), “An efficient approach for Steiner tree problem by genetic algorithm”, International Journal of Computer Science and Engineering (SSRG-IJCSE), vol.2, pp.233-237.

[13]. Arie M.C.A. Koster, Xavier Munoz (Eds) (2010), “Graphs and algorithms in communication networks”, Texts in Theoretical Computer Science. An EATCS Series, Springer, pp.1-177.

[14]. Bang Ye Wu, Kun-Mao Chao (2004), “Spanning trees and optimization problems”, Discrete Mathematics and Its Applications, Chapman&Hall/CRC, pp.13-139.

[15]. C. C. Ribeiro, M.C. Souza (2000), “Tabu Search for the Steiner problem in graphs”, Networks, 36, pp.138-146.

[16]. Cédric Bentz, Marie-Christine Costa, Alain Hertz (2020), “On the edge capacitated Steiner tree problem”, Discrete Optimization, Volume 38, 100607.

[17]. Cheng, X., Du, D.-Z. (eds.) (2001), “Steiner Trees in Industry”, Combinatorial Optimization, vol. 11. Kluwer, Dordrecht.

[18]. Chi-Yeh Chen (2018), “An efficient approximation algorithm for the Steiner tree problem”, National Cheng Kung University, Taiwan.

[19]. Chin Lung Lu, Chuan Yi Tang, Richard Chia-Tung Lee (2003), “The full Steiner tree problem”, Theoretical Computer Science, Volume 306, issue 1-3, Elsevier, pp.55-67.

[20]. Cieslik, D. (1998), “Steiner minimal trees”. Nonconvex Optimization and Its Applications, volume 23, Kluwer, Dordrecht.

[21]. D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi (2006), “The Bees algorithm - A novel tool for complex optimisation problems”,

Intelligent Production Machines and Systems, ELSEVIER, Proceedings of IPROMS 2006 Conference, Cardiff University, pp.454–461.

[22]. Daniel Markus Rehfeldt (2015), “Generic Approach to Solving the Steiner Tree Problem and Variants”, Fachbereich Mathematik der Technischen University at Berlin.

[23]. Davide Bilò (University of Sassari) (2018), “New algorithms for Steiner tree reoptimization”, ICALP.

[24]. Demetres D. Kouvatsos (Edited) (2011), “Network Performance Engineering”, Lecture Notes in Computer Science, volume 5233, Springer-Verlag Berlin Heidelberg, pp.1-795.

[25]. Dervis Karaboga, Bahriye Basturk (2007). “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm”. Journal of Global Optimization, volume 39, Springer, pp.459-471.

[26]. Ding-Zhu Du, J. M. Smith, J.H. Rubinstein (2000), “Advances in Steiner trees”, Combinatorial Optimization, volume 6, pp.1-322.

[27]. Du, D.-Z., Smith, J.M., Rubinstein, J.H. (eds.) (2000): “Advances in Steiner Trees”, Combinatorial Optimization, volume 6, Kluwer, Dordrecht.

[28]. Dusan Teodorovic (2010), “Bee colony optimization”, Belgrade, Serbia, pp.1-26.

[29]. Eduardo Uchoa, Renato F. Werneck (2010), “Fast local search for Steiner trees in graphs”, SIAM.

[30]. Fichte Johannes K., Hecher Markus, and Schidler André (2020), “Solving the Steiner Tree Problem with few Terminals”, University of Potsdam, Germany.

[31]. Fred Glover, Manuel Laguna (1998), “Tabu search”, Kluwer Academic Publishers, Boston, pp.1-209.

[32]. G. Dahl and M. Stoer (1998), "A cutting plane algorithm for multi-commodity survivable network design problems", INFORMS Journal on Computing, Vol. 10, pp. 1–11.

[33]. Geoffrey Ross Grimwood (1994), “The Euclidean Steiner Tree Problem: Simulated Annealing and Other Heuristics”, Operations Research, volume 9, Victoria University of Wellington.

[34]. Guy Even, Guy Kortsarz, Wolfgang Slany (2005), "On network design problems: fixed cost flows and the covering steiner problem", ACM Transactions on Algorithms, Volume 1, Issue 1, pp 74–101.

[35]. Hadrien Cambazard, Nicolas Catusse (2019), “Fixed-parameter algorithms for rectilinear steiner tree and rectilinear traveling salesman problem in the plane”, European Journal of Operational Research, volume 270, issue 2, Univ. Grenoble Alpes, France, pp.419-429.

[36]. Huỳnh Thị Thanh Bình (2011), “Genetic algorithms for solving bounded diameter minimum spanning tree problem”, Luận án tiến sĩ, Trường Đại Học Bách Khoa Hà Nội.

[37]. Hwang, F.K., Richards, D.S., Winter, P. (1992), “The Steiner Tree Problem”, Annals of Discrete Mathematics, vol. 53. North-Holland, Amsterdam.

[38]. Ivana Ljubic (2020), “Solving Steiner Trees – Recent Advances, Challenges and Perspectives”, Networks, volume 77, issue 2, pp.177-204.

[39]. J. E. Beasley (1989), “An SST-Based algorithm for the Steiner problem in graphs”, Networks, volume 19, issue 1, pp.1-16.

[40]. J. E. Beasley, OR-Library: URL http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html

[41]. Jack Holby (2017), “Variations on the Euclidean Steiner Tree Problem and Algorithms”, Rose–Hulman Undergraduate Mathematics Journal, volume 18, issue 1, article 7, St. Lawrence University, pp.124-155.

[42]. Jack J. Dongarra (2014), “Performance of various computers using standard linear equations software”. University of Manchester, pp.1-109 (http://netlib.org/benchmark/performance.pdf).

[43]. Jason Brownlee (2011), "Clever Algorithms: Nature-Inspired Programming Recipes".

[44]. Jeffrey H. Kingston, Nicholas Paul Sheppard (2006), “On reductions for the Steiner problem in graphs”, Basser Department of Computer Science the University of Sydney, Australia, pp.1-10.

[45]. Job Kwakernaak (2020), “Pipeline network optimization using Steiner nodes”, Wageningen University and Research Centre, The Netherland.

[46]. Jon William Van Laarhoven (2010), “Exact and heuristic algorithms for the euclidean Steiner tree problem”, University of Iowa, (Doctoral thesis).

[47]. L. Kou, G. Markowsky, L. Berman (1981), “A Fast Algorithm for Steiner Trees”, acta informatica, Vol.15, pp.141-145.

[48]. Lei Wang, Huayang Feng, Li Lin, Li Du (2018) – “Design of a Heuristic Topology Generation Algorithm in Multi-Domain Optical Networks”, Communications and Network, 2018, 10, 65-77.

[49]. Leitner and et al (2014), “New Real-world Instances for the Steiner Tree Problem in Graphs”.

[50]. M. Andrews and L. Zhang (2002), "Approximation algorithms for access network design", Algorithmica, Vol 34, pp.197-215.

[51]. M. Hauptmann, M. Karpinski (Eds) (2015), “A compendium on Steiner tree problems”, pp.1-36.

[52]. Marcello Caleffi, Ian F. Akyildiz, Luigi Paura (2015), “On the solution of the Steiner tree np-hard problem via physarum bionetwork”, IEEE/ACM Transactions on Networking, volume 23, issue 4, IEEE, pp.1092-1106.

[53]. Martin Zachariasen (1998), “Algorithms for Plane Steiner Tree Problems”, University of Copenhagen, (PH.D. Thesis).

[54]. Matjaz Kovse (2016), “Vertex decomposition of steiner wiener index and Steiner betweenness centrality”, University of Wroc law, Poland.

[55]. Mauricio G. C. Resende, Panos M. Pardalos (2006), "Handbook of Optimization in Telecommunications".

[56]. Mohamed Abdel-Basset, Laila Abdel-Fatah, Arun Kumar Sangaiah (2018), "Metaheuristic Algorithms: A Comprehensive Review", Computational

Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, pp.185-231.

[57]. Nguyen Viet Huy, Nguyen Duc Nghia (2008), “Solving graphical Steiner tree problem using parallel genetic algorithm”, RIVF.

[58]. Ondra Suchý (2014), “Exact algorithms for Steiner tree”, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic.

[59]. P Prof. James Orlin (2010), 15.082J Network Optimization MIT Course.

[60]. Pierre Hansen, Nenad Mladenovic (2004), “Variable neighborhood search”, Search Methodologies, GERAD and HEC Montreal, Canada, pp.211-238.

[61]. Pieter Oloff De Wet (2008), “Geometric steiner minimal trees”, university of South Africa, (Thesis).

[62]. Poompat Saengudomlert (2011), “Optimization for Communications and Networks”. Taylor and Francis Group, LLC, pp.1-201.

[63]. Prosenjit Bose, Anthony D'Angelo, Stephane Durocher (2020), “On the Restricted 1-Steiner Tree Problem”, Computing and Combinatorics, Funded in part by the Natural Sciences and Engineering Research Council of Canada, pp.448-459.

[64]. Prömel, H.-J., Steger, A. (2002), “The Steiner Tree Problem: A Tour through Graphs, Algorithms and Complexity”, Advanced Lectures in Mathematics, Vieweg+Teubner Verlag.

[65]. Radek Hušek, Dušan Knop, Tomáš Masarík (2020), “Approximation Algorithms for Steiner Tree Based on Star Contractions: A Unified View”, International Symposium on Parameterized and Exact Computation (IPEC 2020), ACM.

[66]. Reyan Ahmed and et al (2018), “Multi-level Steiner trees”, ACM J. Exp. Algor., 1(1).

[67]. Robert Sedgewick, Kevin Wayne (2011), “Algorithms”, Fourth edition, Addsion-Wesley, pp.518-700.

[68]. S. E. Dreyfus, R. A. Wagner (1971), “The Steiner Problem in Graphs”, Networks, Vol.1, issue 3, pp.195-207.

[69]. S.L. Martins, M.G.C. Resende, C.C. Ribeiro, and P.M. Pardalos (2000), “A parallel grasp for the Steiner tree problem in graphs using a hybrid local search strategy”, Journal of Global Optimization, volume 17, pp.267-283.

[70]. S.N. Sivanandam, S.N. Deepa (2008), “Introduction to Genetic Algorithms”, Springer.

[71]. Sally Picciotto (1999), “How to encode a tree”, PhD thesis, University of California, San Diego, pp.1-123.

[72]. Sebastián Urrutia, Irene Loiseau (2001), "A New Metaheuristic and its Application to the Steiner Problems in Graphs", SCCC 2001. 21st International Conference of the Chilean Computer Science Society.

[73]. Siavash Vahdati Daneshmand (2003), “Algorithmic Approaches to the Steiner Problem in Networks”, Mannheim.

[74]. Stefan Hougardy, Jannik Silvanus, Jens Vygen (2015), “Dijkstra meets Steiner: a fast-exact goal-oriented Steiner tree algorithm”, Research Institute for Discrete Mathematics, University of Bonn, pp.1-59.

[75]. Thomas Bosman (2015), “A solution merging heuristic for the Steiner problem in graphs using tree decompositions”, Lecture Notes in Computer Science, volume 9125, VU University Amsterdam, The Netherlands, 2015, pp.1-12.

[76]. Thomas Pajor, Eduardo Uchoa, Renato F. Werneck (2018), “A robust and scalable algorithm for the Steiner problem in graphs”, Mathematical Programming Computation, volume 10, Springer, pp.69-118.

[77]. Thorsten Koch, Alexander Martin (1996), “Solving Steiner tree problems in graphs to optimality”, Germany, pp.1-31.

[78]. Tobias Polzin (2003), “Algorithms for the Steiner Problem in Networks”, Universit¨at des Saarlandes, (Thesis).

[79]. Volker Gerd Fischer (2002), "Evolutionary Design of Corporate Networks under Uncertainty", Institut für Informatik der Technischen Universität München Lehrstuhl für Informatik VIII, Rechnerkommunikation und maschinelle Deduktion.

[80]. Voß, S. (1990), “Steiner-Probleme in Graphen”, Hain-Verlag, Frankfurt/M.

[81]. Wassim Jaziri (Edited) (2008), “Local Search Techniques: Focus on Tabu Search”. In-teh, Vienna, Austria, pp.1-201.

[82]. Xin-She Yang (2010), “Engineering optimization”, WILEY, pp.21-137.

[83]. Xin-She Yang (2010), “Nature-inspired metaheuristic algorithms”. LUNIVER Press, pp.53–62.

[84]. Xinhui Wang (2008), “Exact algorithms for the Steiner tree problem”, doctoral thesis, University of Twente, ISSN 1381-3617.

[85]. Xiuzhen Cheng, Ding-Zhu Du (2004), “Steiner trees in industry”, Handbook of Combinatorial Optimization, vol.5, Kluwer Academic Publishers, pp.193-216.

[86]. Yahui Sun (2019), “Solving the Steiner Tree Problem in Graphs using Physarum-inspired Algorithms”.

[87]. Zhiliu Zhang (2016), “Rectilinear Steiner Tree Construction”, Lincoln, Nebraska, USA, (Thesis).

# PHỤ LỤC 1. HỆ THỐNG DỮ LIỆU CHUẨN

*(Gồm 78 đồ thị trong hệ thống dữ liệu thực nghiệm chuẩn)*

Phụ lục 1.1. Bảng nhóm các đồ thị *steinb*

| **Test** | ***n*** | ***m*** | **\|*L*\|** |
|---|---|---|---|
| steinb1.txt | 50 | 63 | 9 |
| steinb2.txt | 50 | 63 | 13 |
| steinb3.txt | 50 | 63 | 25 |
| steinb4.txt | 50 | 100 | 9 |
| steinb5.txt | 50 | 100 | 13 |
| steinb6.txt | 50 | 100 | 25 |
| steinb7.txt | 75 | 94 | 13 |
| steinb8.txt | 75 | 94 | 19 |
| steinb9.txt | 75 | 94 | 38 |
| steinb10.txt | 75 | 150 | 13 |
| steinb11.txt | 75 | 150 | 19 |
| steinb12.txt | 75 | 150 | 38 |
| steinb13.txt | 100 | 125 | 17 |
| steinb14.txt | 100 | 125 | 25 |
| steinb15.txt | 100 | 125 | 50 |
| steinb16.txt | 100 | 200 | 17 |
| steinb17.txt | 100 | 200 | 25 |
| steinb18.txt | 100 | 200 | 50 |

Phụ lục 1.2. Bảng nhóm các đồ thị *steinc*

| **Test** | ***n*** | ***m*** | **$\|L\|$** |
|---|---|---|---|
| steinc1.txt | 500 | 625 | 5 |
| steinc2.txt | 500 | 625 | 10 |
| steinc3.txt | 500 | 625 | 83 |
| steinc4.txt | 500 | 625 | 125 |
| steinc5.txt | 500 | 625 | 250 |
| steinc6.txt | 500 | 1000 | 5 |
| steinc7.txt | 500 | 1000 | 10 |
| steinc8.txt | 500 | 1000 | 83 |
| steinc9.txt | 500 | 1000 | 125 |
| steinc10.txt | 500 | 1000 | 250 |
| steinc11.txt | 500 | 2500 | 5 |
| steinc12.txt | 500 | 2500 | 10 |
| steinc13.txt | 500 | 2500 | 83 |
| steinc14.txt | 500 | 2500 | 125 |
| steinc15.txt | 500 | 2500 | 250 |
| steinc16.txt | 500 | 12500 | 5 |
| steinc17.txt | 500 | 12500 | 10 |
| steinc18.txt | 500 | 12500 | 83 |
| steinc19.txt | 500 | 12500 | 125 |
| steinc20.txt | 500 | 12500 | 250 |

Phụ lục 1.3. Bảng nhóm các đồ thị *steind*

| **Test** | ***n*** | ***m*** | **$\|L\|$** |
|---|---|---|---|
| steind1.txt | 1000 | 1250 | 5 |
| steind2.txt | 1000 | 1250 | 10 |
| steind3.txt | 1000 | 1250 | 167 |
| steind4.txt | 1000 | 1250 | 250 |
| steind5.txt | 1000 | 1250 | 500 |
| steind6.txt | 1000 | 2000 | 5 |
| steind7.txt | 1000 | 2000 | 10 |
| steind8.txt | 1000 | 2000 | 167 |
| steind9.txt | 1000 | 2000 | 250 |
| steind10.txt | 1000 | 2000 | 500 |
| steind11.txt | 1000 | 5000 | 5 |
| steind12.txt | 1000 | 5000 | 10 |
| steind13.txt | 1000 | 5000 | 167 |
| steind14.txt | 1000 | 5000 | 250 |
| steind15.txt | 1000 | 5000 | 500 |
| steind16.txt | 1000 | 25000 | 5 |
| steind17.txt | 1000 | 25000 | 10 |
| steind18.txt | 1000 | 25000 | 167 |
| steind19.txt | 1000 | 25000 | 250 |
| steind20.txt | 1000 | 25000 | 500 |

Phụ lục 1.4. Bảng nhóm các đồ thị *steine*

| **Test** | ***n*** | ***m*** | **$\|L\|$** |
|---|---|---|---|
| steine1.txt | 2500 | 3125 | 5 |
| steine2.txt | 2500 | 3125 | 10 |
| steine3.txt | 2500 | 3125 | 417 |
| steine4.txt | 2500 | 3125 | 625 |
| steine5.txt | 2500 | 3125 | 1250 |
| steine6.txt | 2500 | 5000 | 5 |
| steine7.txt | 2500 | 5000 | 10 |
| steine8.txt | 2500 | 5000 | 417 |
| steine9.txt | 2500 | 5000 | 625 |
| steine10.txt | 2500 | 5000 | 1250 |
| steine11.txt | 2500 | 12500 | 5 |
| steine12.txt | 2500 | 12500 | 10 |
| steine13.txt | 2500 | 12500 | 417 |
| steine14.txt | 2500 | 12500 | 625 |
| steine15.txt | 2500 | 12500 | 1250 |
| steine16.txt | 2500 | 62500 | 5 |
| steine17.txt | 2500 | 62500 | 10 |
| steine18.txt | 2500 | 62500 | 417 |
| steine19.txt | 2500 | 62500 | 625 |
| steine20.txt | 2500 | 62500 | 1250 |

# PHỤ LỤC 2. HỆ THỐNG DỮ LIỆU MỞ RỘNG

*(Gồm 80 đồ thị thưa kích thước lớn)*

Phụ lục 2.1. Bảng nhóm các đồ thị *steinf*

| **Test** | ***n*** | ***m*** | **$\|L\|$** |
|---|---|---|---|
| steinf1.txt | 10000 | 93750 | 10 |
| steinf2.txt | 10000 | 93750 | 20 |
| steinf3.txt | 10000 | 93750 | 834 |
| steinf4.txt | 10000 | 93750 | 1250 |
| steinf5.txt | 10000 | 93750 | 2500 |
| steinf6.txt | 10000 | 125000 | 10 |
| steinf7.txt | 10000 | 125000 | 20 |
| steinf8.txt | 10000 | 125000 | 834 |
| steinf9.txt | 10000 | 125000 | 1250 |
| steinf10.txt | 10000 | 125000 | 2500 |
| steinf11.txt | 10000 | 156250 | 10 |
| steinf12.txt | 10000 | 156250 | 20 |
| steinf13.txt | 10000 | 156250 | 834 |
| steinf14.txt | 10000 | 156250 | 1250 |
| steinf15.txt | 10000 | 156250 | 2500 |
| steinf16.txt | 10000 | 187500 | 10 |
| steinf17.txt | 10000 | 187500 | 20 |
| steinf18.txt | 10000 | 187500 | 834 |
| steinf19.txt | 10000 | 187500 | 1250 |
| steinf20.txt | 10000 | 187500 | 2500 |

Phụ lục 2.2. Bảng nhóm các đồ thị *steing*

| **Test** | ***n*** | ***m*** | **$\|L\|$** |
|---|---|---|---|
| steing1.txt | 20000 | 215000 | 15 |
| steing2.txt | 20000 | 215000 | 25 |
| steing3.txt | 20000 | 215000 | 950 |
| steing4.txt | 20000 | 215000 | 1750 |
| steing5.txt | 20000 | 215000 | 3780 |
| steing6.txt | 20000 | 275000 | 15 |
| steing7.txt | 20000 | 275000 | 25 |
| steing8.txt | 20000 | 275000 | 950 |
| steing9.txt | 20000 | 275000 | 1750 |
| steing10.txt | 20000 | 275000 | 3780 |
| steing11.txt | 20000 | 385000 | 15 |
| steing12.txt | 20000 | 385000 | 25 |
| steing13.txt | 20000 | 385000 | 950 |
| steing14.txt | 20000 | 385000 | 1750 |
| steing15.txt | 20000 | 385000 | 3780 |
| steing16.txt | 20000 | 447500 | 15 |
| steing17.txt | 20000 | 447500 | 25 |
| steing18.txt | 20000 | 447500 | 950 |
| steing19.txt | 20000 | 447500 | 1750 |
| steing20.txt | 20000 | 447500 | 3780 |

Phụ lục 2.3. Bảng nhóm các đồ thị *steinh*

| **Test** | ***n*** | ***m*** | **\|*L*\|** |
|---|---|---|---|
| steinh1.txt | 50000 | 425000 | 15 |
| steinh2.txt | 50000 | 425000 | 25 |
| steinh3.txt | 50000 | 425000 | 950 |
| steinh4.txt | 50000 | 425000 | 1750 |
| steinh5.txt | 50000 | 425000 | 3780 |
| steinh6.txt | 50000 | 475000 | 15 |
| steinh7.txt | 50000 | 475000 | 25 |
| steinh8.txt | 50000 | 475000 | 950 |
| steinh9.txt | 50000 | 475000 | 1750 |
| steinh10.txt | 50000 | 475000 | 3780 |
| steinh11.txt | 50000 | 528000 | 15 |
| steinh12.txt | 50000 | 528000 | 25 |
| steinh13.txt | 50000 | 528000 | 950 |
| steinh14.txt | 50000 | 528000 | 1750 |
| steinh15.txt | 50000 | 528000 | 3780 |
| steinh16.txt | 50000 | 587500 | 15 |
| steinh17.txt | 50000 | 587500 | 25 |
| steinh18.txt | 50000 | 587500 | 950 |
| steinh19.txt | 50000 | 587500 | 1750 |
| steinh20.txt | 50000 | 587500 | 3780 |

Phụ lục 2.4. Bảng nhóm các đồ thị *steini*

| **Test** | ***n*** | ***m*** | **\|*L*\|** |
|---|---|---|---|
| steini1.txt | 100000 | 125000 | 25 |
| steini2.txt | 100000 | 125000 | 45 |
| steini3.txt | 100000 | 125000 | 1250 |
| steini4.txt | 100000 | 125000 | 2450 |
| steini5.txt | 100000 | 125000 | 4500 |
| steini6.txt | 100000 | 200000 | 25 |
| steini7.txt | 100000 | 200000 | 45 |
| steini8.txt | 100000 | 200000 | 1250 |
| steini9.txt | 100000 | 200000 | 2450 |
| steini10.txt | 100000 | 200000 | 4500 |
| steini11.txt | 100000 | 500000 | 25 |
| steini12.txt | 100000 | 500000 | 45 |
| steini13.txt | 100000 | 500000 | 1250 |
| steini14.txt | 100000 | 500000 | 2450 |
| steini15.txt | 100000 | 500000 | 4500 |
| steini16.txt | 100000 | 2500000 | 25 |
| steini17.txt | 100000 | 2500000 | 45 |
| steini18.txt | 100000 | 2500000 | 1250 |
| steini19.txt | 100000 | 2500000 | 2450 |
| steini20.txt | 100000 | 2500000 | 4500 |