**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

# ĐỒ ÁN TỐT NGHIỆP

## Tối ưu sắp đặt chức năng mạng và định tuyến mạng ảo hoá theo tiến hoá đa mục tiêu

**Dương Duy Đông**
dong.dd183497@sis.hust.edu.vn

**Ngành: Khoa Học Máy Tính**

**Giảng viên hướng dẫn:** PGS.TS. Nguyễn Khanh Văn _____

Chữ kí GVHD

**Khoa:** Khoa Học Máy Tính

**Trường:** Công nghệ thông tin và Truyền thông

**HÀ NỘI, 08/2023**

# LỜI CAM KẾT

Họ và tên sinh viên: Dương Duy Đông.
Điện thoại liên lạc: 0866883212.
Email: dong.dd183497@sis.hust.edu.vn.
Lớp: Khoa học máy tính 03K63.
Hệ đào tạo: Chính quy.

Tôi – *Dương Duy Đông* – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *PGS.Nguyễn Khanh Văn* giảng viên bộ môn Khoa học máy tính, Trường CNTT, Đại học Bách Khoa Hà Nội và *TS.Nguyễn Thị Tâm* nghiên cứu sinh phòng thí nghiệm Mô hình hóa, mô phỏng và tối ưu hóa. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

*Hà Nội, ngày tháng năm*

Tác giả ĐATN

*Họ và tên sinh viên*

# LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn chân thành và sâu sắc nhất tới thầy Nguyễn Khanh Văn giảng viên bộ môn Khoa học máy tính, Trường CNTT, Đại học Bách Khoa Hà Nội và cô Nguyễn Thị Tâm nghiên cứu sinh phòng thí nghiệm Mô hình hóa, mô phỏng và tối ưu hóa đã giúp đỡ và đã luôn tận tình chỉ bảo, hướng dẫn để em có thể hoàn thành tốt đồ án tốt nghiệp này.

Em xin chân thành cảm ơn các thầy giáo, cô giáo – Khoa học máy tính – Đại học Bách khoa Hà Nội, những người đã tận tình truyền đạt các kiến thức cho em trong suốt thời gian em học tập và nghiên cứu tại trường. Em xin cảm ơn các bạn học cùng lớp KHMT – 03 K63 đã giúp đỡ và động viên em trong suốt quá tình học tập và thực hiện đồ án tốt nghiệp. Bên cạnh đó, xin gửi lời cảm ơn tới cô Huỳnh Thị Thanh Bình đã gửi tặng em những lời khuyên và góp ý để đồ án được hoàn thiện trọn vẹn.

Cuối cùng, em cũng xin gửi lời cảm ơn tới gia đình đã ủng hộ, động viên em trong suốt quá trình học tập vừa qua.

Do trong quá trình nghiên cứu, tìm hiểu và thực nghiệm đồ án chắc chắn không thể tránh khỏi những sai sót nhất định, em rất mong nhận được sự góp ý của thầy, cô giáo và các bạn để đồ án được hoàn chỉnh hơn. Xin trân trọng cảm ơn!

# MỤC LỤC

# DANH MỤC HÌNH VẼ

# DANH MỤC BẢNG BIỂU

# DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

| Thuật ngữ | Ý nghĩa |
|---|---|
| Dominates | Trội (dominates) |
| gen | Một đơn vị vật chất di truyền có chức năng nhỏ bé nhất (gen) |
| MOEA/D | Thuật toán tiến hóa đa mục tiêu dựa trên sự phân rã (Multi-Objective Evolutionary Algorithm Based on Decomposition) |
| MOTLBO | Tối ưu hóa dạy-học dựa trên đa mục tiêu (Multi-Objective Teaching-Learning-Based Optimization) |
| NSGA2 | Giải thuật di truyền sắp xếp không trội (Non-dominated Sorting Genetic Algorithm II) |
| NV | Mạng ảo hoá (Network virtualization) |
| SFC | Chuỗi các chức năng dịch vụ (Service Function Chain) |
| Switch | Thiết bị chuyển mạch trong hệ thống mạng (Switch) |
| VM | Máy ảo (VM - Virtual Machine) |
| VNF | Chức năng mạng ảo (Virtual Network Functions) |

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## 1.1 Đặt vấn đề

Trong thời đại hiện nay, Internet đã trở thành một phần không thể thiếu trong mọi lĩnh vực. Nó đóng vai trò vô cùng quan trọng và đa dạng trong xã hội và nền kinh tế hiện đại. Internet mang lại sự tiện lợi và hiệu quả trong việc giao tiếp, là nguồn tài liệu vô cùng quý giá và đa dạng. Nó đã tạo ra môi trường thương mại điện tử để ta có thể mua bán một cách trực tuyến, cung cấp các thông tin giải trí, chơi game, tin tức. Là nguồn gốc của công nghệ mới, mang lại nhiều sự phát triển về kinh tế. Tóm lại, vai trò của Internet không chỉ giới hạn trong việc kết nối con người với nhau, mà còn mở ra nhiều cơ hội mới, thay đổi cách thức làm việc và cách sống của chúng ta. Tuy nhiên, để có thể tận dụng tài nguyên mạng một cách tối đa, tăng tính linh hoạt, hiệu suất và độ tin cậy cao, giảm chi phí vận hành thì ta cần đến Mạng ảo hoá.

Mạng ảo hóa (NV - Network virtualization) là phương pháp kết hợp các tài nguyên sẵn có trong mạng bằng cách chia băng thông sẵn có thành các kênh, mỗi kênh độc lập với các kênh khác và mỗi kênh có thể được gán (hoặc gán lại) cho một máy ảo (VM - Virtual Machine) hoặc thiết bị cụ thể theo thời gian thực. Ý tưởng chính của NV là sự tạo ra các mạng logic độc lập trên cơ sở hạ tầng mạng vật lý. NV cho phép tài nguyên mạng được phân chia và chia sẻ hiệu quả giữa nhiều ứng dụng và hệ thống. Thay vì phải dành riêng một mạng vật lý cho mỗi ứng dụng, NV cho phép sử dụng lại tài nguyên mạng, giảm bớt lãng phí và tăng khả năng tận dụng tài nguyên. NV cho phép tạo ra các mạng logic độc lập, có thể được cấu hình và quản lý theo nhu cầu, điều này giúp tăng tính linh hoạt trong việc triển khai và quản lý mạng, cho phép điều chỉnh và thay đổi cấu trúc mạng một cách nhanh chóng và dễ dàng. NV cho phép cách ly các mạng logic khác nhau, giúp tránh sự can thiệp và ảnh hưởng đến nhau. Ngoài ra NV cung cấp khả năng chuyển đổi và chuyển tiếp dữ liệu một cách linh hoạt, tối ưu hóa đường truyền và tăng hiệu suất mạng. Đồng thời, NV cũng cung cấp khả năng sao lưu và khôi phục dữ liệu, giúp đảm bảo độ tin cậy và khả năng phục hồi sau sự cố. NV cho phép quản lý và vận hành mạng một cách tự động và trung tâm hơn. Quản trị mạng trở nên đơn giản hơn, giảm thiểu sự phụ thuộc vào phần cứng và giảm công sức và chi phí trong việc cấu hình, quản lý và bảo trì mạng. NV cung cấp nền tảng cơ sở để triển khai và quản lý các mô hình điện toán đám mây và ảo hóa hạ tầng. Nó cho phép tổ chức tận dụng các ưu điểm của việc sử dụng tài nguyên chia sẻ và linh hoạt, tăng cường khả năng mở rộng và quản lý hiệu quả các tài nguyên ảo trên mạng. Tóm lại, NV mang lại nhiều lợi ích

cho việc quản lý và vận hành mạng, bao gồm tối ưu hóa tài nguyên, tăng tính linh hoạt, cải thiện hiệu suất và độ tin cậy, giảm chi phí và hỗ trợ các mô hình điện toán đám mây và ảo hóa hạ tầng. Vì vậy, NV đã trở thành một phần quan trọng trong quản lý và phát triển hạ tầng mạng hiện đại. Mặc dù NV mang lại nhiều lợi ích, nhưng trong quá trình triển khai và sử dụng có một số vấn đề đã và đang được giải quyết như: khả năng mở rộng, bảo mật, và đặc biệt về việc quản lý tài nguyên.

## 1.2 Vấn đề còn tồn tại trong mạng ảo hoá

NV mang đến khả năng mở rộng linh hoạt, bảo mật và quản lý tài nguyên hiệu quả. Tuy nhiên, nhiều vấn đề còn tồn tại, nó là một trong những thách thức mà các nhà quản lý hệ thống và quản trị viên mạng phải đối mặt.

Vấn đề hiệu suất: Khi triển khai NV vấn đề xảy ra khi nhiều máy ảo (VM) hoặc VNF yêu cầu sử dụng tài nguyên mạng hoặc băng thông cao đồng thời, vượt quá khả năng chịu đựng của hạ tầng mạng vật lý hoặc ảo hóa gây ra hiện tượng giảm hiệu suất mạng, độ trễ tăng cao, gây ảnh hưởng đến chất lượng dịch vụ và trải nghiệm người dùng.

Vấn đề về bảo mật: NV tạo ra một môi trường phức tạp hơn cho việc quản lý bảo mật. Các máy ảo và VNF cần được bảo vệ khỏi các mối đe dọa mạng như tấn công từ bên ngoài, xâm nhập và đánh cắp dữ liệu. Đồng thời, cần xác định và quản lý quyền truy cập vào các tài nguyên NV.

Vấn đề về tương thích: Trong môi trường NV, các hệ thống, ứng dụng và dịch vụ phải tương thích với nhau và hoạt động trơn tru để đảm bảo tính hợp nhất và khả năng chuyển đổi giữa các môi trường ảo.

Vấn đề về mở rộng: NV cần có khả năng mở rộng để đáp ứng nhu cầu tăng cao về tài nguyên mạng và số lượng máy ảo hoặc VNF. Việc mở rộng NV đòi hỏi sự linh hoạt và khả năng mở rộng của hạ tầng mạng vật lý để có thể thêm hoặc giảm các tài nguyên mạng một cách dễ dàng và không gây gián đoạn đến hoạt động hiện tại.

Vấn đề quản lý, phân bổ tài nguyên: NV đòi hỏi quản lý và phân chia tài nguyên mạng một cách hiệu quả để đảm bảo rằng các máy ảo và VNF có đủ tài nguyên để hoạt động một cách ổn định. Trong NV, việc phân bổ tài nguyên là quá trình chia sẻ và cấp phát tài nguyên mạng cho các mạng logic ảo. Để hỗ trợ các hoạt động Mạng logic ảo cần nhiều tài nguyên khác nhau chẳng hạn như: Băng thông mạng, Địa chỉ IP, CPU, RAM, Bộ nhớ lưu trữ, các chức năng dịch vụ mạng (tường lửa, cân bằng tải, chuyển tiếp gói, quản lý băng thông). Quá trình phân bổ tài nguyên trong NV cần dựa trên các yêu cầu và ưu tiên của từng mạng logic ảo. Điều này đòi

hỏi việc đánh giá và hiểu rõ nhu cầu sử dụng tài nguyên của từng ứng dụng, đảm bảo tài nguyên được phân bổ một cách tối ưu và hiệu quả.

Trước khi có NV, các chức năng mạng như tường lửa, định tuyến, NAT (Network Address Translation), hay cân bằng tải thường được triển khai trên các thiết bị vật lý riêng biệt. Tuy nhiên, với sự phát triển của công nghệ ảo hóa chức năng mạng, các chức năng này có thể được ảo hóa và triển khai trên cơ sở hạ tầng mạng ảo. Chức năng mạng ảo (VNF - Virtualized Network Function) là một phiên bản ảo của chức năng mạng truyền thống, được triển khai và quản lý trên một nền tảng ảo hóa. Thay vì cần cài đặt và cấu hình riêng lẻ trên các thiết bị vật lý, các VNF có thể được triển khai dưới dạng máy ảo trên hạ tầng ảo hóa, chia sẻ tài nguyên vật lý và quản lý thông qua một trung tâm điều khiển ảo. Với VNF, các chức năng mạng truyền thống trở thành các ứng dụng phần mềm linh hoạt và có thể mở rộng. Các VNF có thể được triển khai, cấu hình và quản lý từ xa, cho phép điều chỉnh linh hoạt theo nhu cầu và tải công việc của mạng. Điều này mang lại tính linh hoạt, khả năng mở rộng và tiết kiệm chi phí trong việc triển khai và quản lý các chức năng mạng.

Trong một mạng truyền thống, các chức năng mạng như tường lửa, cân bằng tải, định tuyến, kiểm soát băng thông, hay phân giải tên miền (DNS) thường được triển khai riêng lẻ trên các thiết bị vật lý. Chuỗi chức năng dịch vụ (SFC - Service Function Chaining) cho phép triển khai và kết hợp các chức năng này theo một chuỗi hoặc luồng dữ liệu nhất định, tạo ra một dịch vụ mạng toàn diện. Trong NV cũng vậy SFC xác định luồng dữ liệu và định tuyến nó qua các VNF một cách tuần tự. Mỗi VNF có thể thực hiện các chức năng mạng cụ thể như kiểm tra bảo mật, biến đổi giao thức, phân loại lưu lượng, hoặc áp dụng quy tắc chính sách.

Nhận thấy tầm quan trọng trong quản lý và phát triển hạ tầng mạng hiện đại, việc giải quyết vấn đề phân bổ tài nguyên cho mạng ảo hóa một cách tối ưu mà vẫn đảm bảo hiệu năng trong mạng là việc rất cần thiết, có ý nghĩa rất lớn trong mạng ảo hoá. Do đó đồ án nghiên cứu này xin trình bày phương pháp giải quyết vấn đề cài đặt các VNF nào trên các server trong mạng và các SFC được định tuyến như thế nào để tối ưu ba mục tiêu chi phí kích hoạt server, chi phí cài đặt VNF trên server và độ trễ cho các yêu cầu SFC, mà vẫn phải thoả mãn các ràng buộc về tài nguyên tối đa băng thông, bộ nhớ, CPU (mô hình toán học của bài toán sẽ được trình bày trong Chương 2).

## 1.3 Cách tiếp cận

Có nhiều cách giải quyết bài toán khác nhau, tùy thuộc vào mạng cần cần giải quyết. Tuy nhiên, một bài toán tối ưu sẽ có hai hướng giải quyết là lời chính xác và

lời giải chấp nhận được. Mỗi cách sẽ có ưu nhược điểm riêng, như trong thời gian ngắn lời giải chính xác sẽ phù hợp với mạng đơn giản còn với mạng phức tạp thì lời giải chính xác có thể sẽ không đưa ra được lời giải. Với hướng giải quyết theo lời giải chấp nhận được sẽ phù hợp với mạng phức tạp hơn.

Trong đồ án này sẽ trình bày cách giải quyết bài toán sử dụng thuật toán tiến hoá đa mục tiêu giải bài toán cài đặt chức năng mạng và định tuyến trong mạng ảo hoá. Lời giải tìm được là tập các giải pháp trong một khoảng thời gian.

## 1.4 Bố cục đồ án

Ngoài phần giới thiệu đề tài và kết luận, cấu trúc của đồ án được tổ chức như sau:

**Chương 2:** Đồ án trình bày về bài toán bao gồm đầu vào, đầu ra, các yêu cầu, ràng buộc, mục tiêu.

**Chương 3:** Cơ sở lý thuyết bao gồm bài toán tối ưu, bài toán tối ưu đa mục tiêu, thuật toán tiến hoá.

**Chương 4:** Đồ án sẽ trình bày đề xuất cách mã hóa, định tuyến và các phương pháp giải bài toán đa mục tiêu.

**Chương 5:** Đồ án trình bày về cài đặt thực nghiệm, các bộ dữ liệu, các tiêu chí đánh giá, kết quả của thực nghiệm.

# CHƯƠNG 2. BÀI TOÁN ĐẶT CHỨC NĂNG MẠNG VÀ ĐỊNH TUYẾN TRONG MẠNG ẢO HOÁ

Trong chương này, đồ án sẽ trình bày một tổng hợp các nghiên cứu liên quan đã được thực hiện trước đó, đặc biệt là những nghiên cứu liên quan đến bài toán đặt chức năng và định tuyến trong mạng ảo hoá. Các nghiên cứu này cung cấp một cơ sở quan trọng để hiểu rõ hơn về vấn đề và tìm kiếm các giải pháp tiềm năng. Tiếp theo trong chương đồ án sẽ cung cấp một cái nhìn tổng quan về bài toán đặt chức năng mạng ảo và định tuyến trong mạng ảo hoá. Các yếu tố chính cần được xem xét, bao gồm đầu vào, đầu ra, ràng buộc và mục tiêu của bài toán sẽ được xác định rõ. Điều này giúp định hình cụ thể các yêu cầu và hạn chế khi giải quyết bài toán, từ đó đưa ra các phương pháp và thuật toán giải để tối ưu hóa quá trình đặt chức năng và định tuyến trong mạng ảo hoá.

## 2.1 Các nghiên cứu liên quan

Bài toán về đặt và định tuyến các VNF (VNF-PRP, VNF placement and routing problem) đã thu hút rất nhiều nhà nghiên cứu trong việc phát triển các phương pháp tối ưu hóa khác nhau. Tuy nhiên, bài toán được chứng minh là NP-khó [1] [2] [3] [4] [5] tức số lượng các giải pháp khả thi sẽ bị nhân lên theo cấp số mũ khi không gian tìm kiếm được mở rộng. Các kết quả nghiên cứu [6] [7] cho thấy giải pháp tối ưu chính xác cho một NP-khó chỉ có thể đạt được đối với trường hợp quy mô rất nhỏ, và không thể đạt được trong một thời gian hợp lý cho các trường hợp trong thực tế. Do đó, để giải quyết vấn đề này, các nghiên cứu sau đó chỉ tập trung vào các hướng tiếp cận dựa trên heuristic, metaheuristic.

Các nghiên cứu liên quan thường chỉ sử dụng mô hình ILP (Integer Linear Programming), giải thuật di truyền để tối ưu một hàm mục tiêu hoặc thiết lập trọng số của các mục tiêu để quy bài toán về một mục tiêu duy nhất. Quanying Sun đề xuất ra một phương pháp ILP [8] nhằm tối thiểu tổng chi phí triển khai SFC, bao gồm cả chi phí liên quan đến việc đặt VNF và ánh xạ liên kết. Cụ thể, thuật toán sẽ gộp các yêu cầu SFC dựa trên cặp nguồn-đích của chúng, tạo ra một đồ thị VNF. Sau đó, thuật toán sẽ sắp xếp các đồ thị VNF theo thứ tự giảm dần của băng thông yêu cầu và cố gắng triển khai các đồ thị VNF một cách lần lượt. Roberto Riggio đề xuất một phương pháp ILP [9] nhằm tối thiểu hóa tổng chi phí nhúng, bao gồm cả chi phí nhúng các nút và các cạnh. Đồng thời, ông cũng đề xuất 1 phương pháp heuristic cho mạng cảm biến không dây WiNE. D.Bhamare cùng cộng sự [10] đã điều tra vấn đề về đặt SFC trong đám mây hệ thống và áp dụng ILP để giảm lưu lượng liên đám mây và thời gian đáp ứng trong các đám mây phân tán. Defang Li

cùng các cộng sự [11] xây dựng mô hình ILP cho bài toán đặt VNF trong dữ liệu đám mây trung tâm để giảm tổng số máy vật lý và tài nguyên được sử dụng. Năm 2017, Sang và cộng sự [12] đã phát triển và sử dụng tuyến tính số nguyên hỗn hợp (MILP) cho việc bố trí và phân bổ VNF và giải quyết nó bằng thuật toán tham lam. Phương pháp này cũng được sử dụng cho các nghiên cứu sau đó [13] [14] [15].

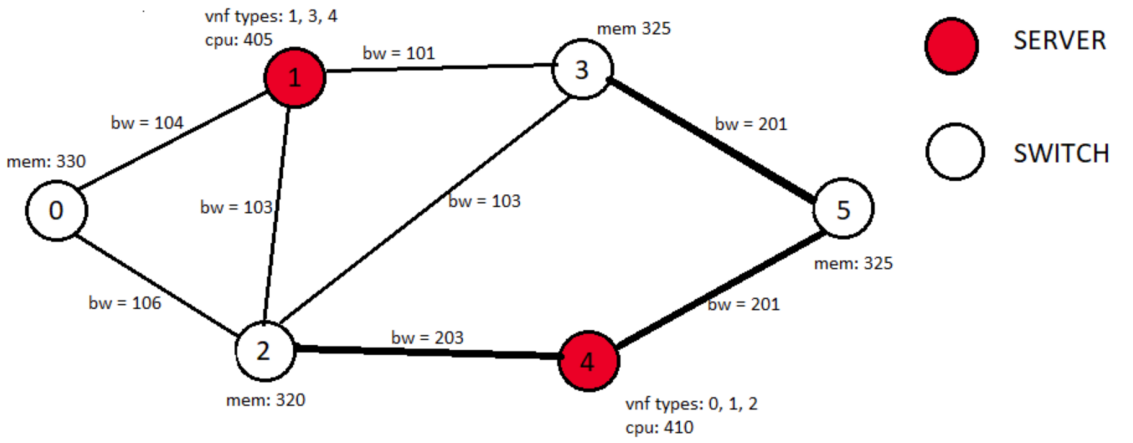
Heuristics rất nhanh và dễ thực hiện, có thể được sử dụng cho các yêu cầu định tuyến/đặt VNF trực tuyến. Tuy nhiên, trong các bài toán đa mục tiêu có không gian tìm kiếm lớn như VNF-PRP, việc thiết kế một heuristic hiệu quả có thể là một nhiệm vụ khó khăn, vì vậy các thuật toán metaheuristic được sử dụng trong trường hợp này. Otokura [16] đã đề xuất một kỹ thuật tiến hóa, được đặt tên là Eva VNFP, để giải bài toán đặt VNF bằng Genetic algorithm (GA), xem xét các mục tiêu khác nhau theo mô-đun. Kim [17] đã phát triển một phương pháp đặt VNF dựa trên GA hiệu quả hơn cho hệ thống đám mây để đạt được mức tiêu thụ năng lượng thấp nhất và các yêu cầu về độ trễ của dịch vụ được đáp ứng. Jamali và Malek-taji [18] đã đề xuất một phương pháp cải tiến dựa trên GGA (Grouping Genetic Algorithm) cho vấn đề đặt VM trong điện toán đám mây: sau khi mô hình hóa vấn đề này như là một bài toán đóng thùng, GGA cố gắng tìm các phương án đặt VM với mức tiêu thụ điện năng thấp nhất và sử dụng tài nguyên hiệu quả nhất . Chantre cùng cộng sự [19] đã sử dụng phương pháp tối ưu bầy đàn (PSO) để giải quyết vấn đề đặt VNF dư thừa, để giảm thiểu mô phỏng tổng độ trễ từ đầu đến cuối của tất cả các dịch vụ. H.Xing cùng cộng sự [1] đã đề xuất một mô hình sử dụng tối ưu bầy sói xám (GWO) để giải quyết đặt VNF, trong đó mục tiêu là đặt SFC với độ trễ đầu cuối thấp. J.Cao [20] giải quyết bài toán đặt VNF sử dụng thuật toán di truyền đa mục tiêu (NSGA-II) và xây dựng nó thông qua bài toán hai mục tiêu để tối đa hóa liên kết và giảm thiểu băng thông sử dụng.

Metaheuristic có thể thu được chất lượng giải pháp tốt hơn so với heuristic, nhưng chúng không thể được sử dụng cho các yêu cầu trực tuyến, do thủ tục tìm kiếm tốn nhiều tài nguyên và thời gian. Để có được sự kết hợp giữa tốc độ của heuristic và chất lượng lời giải của metaheuristic, gần đây các mô hình kết hợp để giải quyết chung cho cả bài toán VNF-PRP đã được giới thiệu. FH-ACO [21] là một mô hình dựa trên logic mờ và tối ưu bầy kiến (ACO) để giải quyết vấn đề chung của VNF-PRP và sử dụng đa mục tiêu trung bình có trọng số chức năng bao gồm độ trễ, mức tiêu thụ năng lượng, băng thông và độ tin cậy. Việc sử dụng logic mờ, có ưu điểm chính đối với VNF-PRP là xử lý sự không chắc chắn vốn có trong kiến trúc NFV và tính toán các yếu tố quan trọng của việc lựa chọn máy chủ/liên kết cũng như sử dụng chúng làm thông tin phỏng đoán để hướng dẫn ACO lựa chọn máy chủ/liên kết phù hợp. Hơn nữa, các tiêu chí khác nhau về tình trạng hiện tại

của cơ sở hạ tầng NFV và các yêu cầu SFC có thể được kết hợp hiệu quả thành một hệ thống suy luận mờ đa tiêu chí để hướng dẫn ACO. EmcFIS [22] đề xuất một mô hình kết hợp dựa trên fuzzy heuristic(như một giải pháp đặt/định tuyến trực tuyến) và metaheuristic(như một thuật toán điều chỉnh siêu tham số ngoại tuyến) cho VNF-PRP. Đây là nghiên cứu kết hợp ngoại-trực tuyến đầu tiên cho VNF-PRP để triển khai đầy đủ các yêu cầu SFC trực tuyến. Ngoài ra, các tính năng khác nhau của yêu cầu SFC có thể được kết hợp hiệu quả vào mô hình hệ thống suy luận mờ đa tiêu chí (mcFIS) để đạt được các giải pháp VNF-PRP tối ưu. Trong bài báo, để xử lý sự phức tạp của mô hình và đạt được hiệu suất tốt nhất, GA được sử dụng trong quy trình ngoại tuyến để điều chỉnh quy tắc mờ tự động của giao thức mcFIS, một lần trước khi áp dụng mcFIS cho ứng dụng thực. Do đó, nó không tăng bất kỳ độ trễ đầu cuối nào cho yêu cầu SFC trực tuyến. Như vậy, các phương pháp trên đều sẽ đánh đổi chất lượng của lời giải để đạt được tốc độ tính toán cao hơn.

## 2.2 Tổng quan về bài toán

Bài toán đặt chức năng mạng ảo và định tuyến các yêu cầu là một trong các vấn đề trong việc phân bổ tài nguyên mạng ảo hoá. Mục tiêu đưa ra cần tối ưu về việc kích hoạt server, cài đặt chức năng mạng và giảm thiểu độ trễ với các yêu cầu SFC. Sẽ cần đáp ứng những ràng buộc về tài nguyên khác trong mạng. Một mạng gồm một tập các nút, các cạnh liên kết giữa các nút và một tập các chức năng mạng ảo (VNF). Đưa ra một tập các yêu cầu (chuỗi chức năng dịch vụ - SFC), mỗi SFC được đặc trưng bởi một nguồn, một đích, lượng tài nguyên được sử dụng và một tập hợp các chức năng mạng ảo (VNF) cần được thực hiện. Mục tiêu của bài toán là đặt các chức năng mạng và định tuyến cho mỗi yêu cầu sao cho giảm thiểu chi phí kích hoạt các nút server, giảm thiểu chi phí cài đặt các VNF và giảm thiểu độ trễ khi thực hiện tất cả các yêu cầu.




**Hình 2.1:** Mô hình hoá một mạng ảo hoá.

Hình 2.1 mô tả một mạng nhỏ gồm tập các nút server có màu đậm, các nút switch là các nút trắng. Trên server thì được cài đặt các chức năng mạng là một tập các kiểu VNF, ngoài ra còn có thông tin về tài nguyên CPU. Trên các nút switch có thông tin về tài nguyên bộ nhớ, các nút server coi bộ nhớ là vô hạn. Liên kết giữa các nút được biểu thị qua các cạch, các cạnh đậm hơn có băng thông lớn hơn.

### 2.2.1 Đầu vào

- Một mạng được biểu diễn dưới dạng đồ thị vô hướng $G = (V, E)$ trong đó $V$ là tập hợp tất cả các nút và $E$ là tập hợp tất cả các liên kết giữa các nút. Các nút có thể được chia thành các nút server $V_{server}$ và nút switch $V_{switch}$ . Các nút server có thể chạy VNF, trong khi các nút switch được sử dụng cho chuyển tiếp luồng mạng. Ta kí hiệu $m_v^{mem}$ and $m_v^{cpu}$ là tài nguyên bộ nhớ và CPU của nút $v$, tương ứng. Mỗi nút server có tài nguyên bộ nhớ không giới hạn và mỗi nút switch không có tài nguyên CPU. Đối với mỗi nút server $v$, chi phí kích hoạt là $cost_v^{server}$ , số lượng VNF tối đa được cài đặt là $m_v^{vnf}$, và độ trễ trong khi chạy VNFs is $d_v$. Mỗi liên kết $(v, u)$ được liên kết với tài nguyên băng thông $m_{vu}^{bw}$, độ trễ $d_{vu}$.
- Ta ký hiệu $F = \{f_1, f_2, ..., f_{|F|}\}$ là tập các VNF, trong đó$|F|$là số loại VNFs. Mỗi $f_k$ có $cost_{vk}^{vnf}$, là chi phí cài đặt của VNF $f_k$ trong nút server $v$.
- Đặt $R = \{r_1, r_2, ..., r_{|R|}\}$ là tập hợp các yêu cầu SFC, trong đó $|R|$ là số lượng yêu cầu SFC. Mỗi yêu cầu SFC $r_j$ bắt đầu từ $sn_j$, đi qua các nút để đáp ứng một tập $k_j$ VNF đã được sắp xếp và kết thúc tại $dn_j$. Nhu cầu bộ nhớ, băng thông và CPU của SFC lần lượt là $w_j^{mem}$, $w_j^{bw}$, $w_j^{cpu}$.

### 2.2.2 Đầu ra

- Tập hợp các biến quyết định $x_v$ trong đó, $x_v = 1$ nếu nút server $v$ đang hoạt động và $x_v = 0$ nếu không.
- Tập hợp các biến quyết định $x_{vk}$ Trong đó, $x_{vk} = 1$ nếu VNF $f_k$ được cài đặt trong server $v$, $x_{vk} = 0$ nếu không.
- Tập hợp các biến quyết định $y^j_{vu}$ trong đó $y^j_{vu} = 1$ nếu SFC $r_j$ đi qua liên kết $(v, u)$, nếu không $y^j_{vu} = 0$

### 2.2.3 Ràng buộc

- Mức tiêu thụ băng thông của liên kết $(v, u)$ phải nhỏ hơn băng thông tối đa:

$$c^{bw}_{vu} \leq m^{bw}_{vu}, \forall (v, u) \in E \tag{2.1}$$

- Mức tiêu thụ bộ nhớ của nút switch $v$ phải nhỏ hơn bộ nhớ tối đa:

$$c^{mem}_{v} \leq m^{mem}_{v}, \forall v \in V_{switch} \tag{2.2}$$

- Mức độ tiêu thụ CPU của server $v$ phải nhỏ hơn CPU tối đa:

$$c^{cpu}_{v} \leq m^{cpu}_{v}, \forall v \in V_{server} \tag{2.3}$$

- Số lượng loại VNF được cài đặt trong mỗi nút không vượt quá số loại VNF tối đa:

$$c^{vnf}_{v} \leq m^{vnf}_{v}, \forall v \in V_{server} \tag{2.4}$$

- Tất cả các yêu cầu SFC phải được đáp ứng

### 2.2.4 Mục tiêu

- Giảm thiểu độ trễ của tất cả các yêu cầu:

$$DL = \frac{\sum_{r_j \in R}(\sum_{(u,v)\in E} y^i_{vu} * d_{vu} + \sum_{v \in V} \sum_{f_k \in F} x^j_{vk} * d_v)}{(\sum_{(v,u)\in E} d_{vu} + \sum_{v \in V_{server}} d_v) * q} \rightarrow \min \tag{2.5}$$

Trong (5), độ trễ của yêu cầu được tính bằng tổng độ trễ trên các liên kết mà yêu cầu đi qua và độ trễ khi chạy VNF trên các server được phân bổ. Sau đó,

mục tiêu được tính bằng độ trễ của tất cả các yêu cầu chia cho độ trễ tối đa có thể.

- Giảm thiểu chi phí kích hoạt server:

$$CS = \frac{\sum_{v \in V_{server}} x_v * cost_v^{server}}{\sum_{v \in V_{server}} cost_v^{server}} \rightarrow \min \tag{2.6}$$

Phương trình (6) xét chi phí kích hoạt của các nút server. Giá trị này được tính theo tỷ lệ chi phí kích hoạt server chia cho chi phí khi kích hoạt tất cả các server.

- Giảm thiểu chi phí lắp đặt VNF:

$$CV = \frac{\sum_{v \in V_{server}} \sum_{f_k \in F} x_v * x_{vk} * cost_{vk}^{vnf}}{\sum_{v \in V_{server}} \sum_{f_k \in F} cost_{vk}^{vnf}} \rightarrow \min \tag{2.7}$$

Phương trình (7) được tính tương tự như Phương trình (6) và phạm vi giá trị của chúng đồng nhất giữa $(0, 1)$.

- Tất cả các hàm mục tiêu DL, CS và CV từ Phương trình (2.5), Phương trình (2.6) và (2.7) được biểu diễn ở dạng chuẩn hóa, tất cả đều là tìm giá trị nhỏ nhất. Bài toán tối ưu đa mục tiêu có thể được kết hợp thành bài toán tối ưu đơn mục tiêu như sau:

$$DL + CS + CV \rightarrow \min \tag{2.8}$$

Trong chương này, đồ án đã tiến hành tổng hợp các nghiên cứu liên quan đã được thực hiện trước đó về bài toán đặt chức năng mạng và định tuyến trong mạng ảo hoá. Những nghiên cứu đã cung cấp cho chúng ta cái nhìn tổng quan về tối ưu tài nguyên mạng ảo và đã đóng góp quan trọng để hiểu rõ hơn về các vấn đề và thách thức của bài toán. Sau đó, đồ án đã trình bày tổng quan về bài toán đặt chức năng và định tuyến trong mạng ảo hoá. Đồ án đã xác định các yếu tố chính, bao gồm đầu vào, đầu ra, ràng buộc và mục tiêu của bài toán. Điều này là cơ sở quan trọng để ta hiểu rõ hơn về phạm vi và khó khăn của bài toán, từ đó chọn lựa và phát triển các phương pháp và thuật toán hiệu quả để tối ưu hóa quá trình đặt chức năng và định tuyến trong mạng ảo hoá. Chương 3 - Cơ sở lý thuyết đồ án sẽ trình bày về lý thuyết bài toán tối ưu, tối ưu đa mục tiêu và giải thuật tiến hóa, những hạn chế và điểm mạnh từ đó đề xuất những giải pháp phù hợp với bài toán mà đồ án đang nghiên cứu.

# CHƯƠNG 3. CƠ SỞ LÝ THUYẾT

Trong chương này đồ án trình bày những tìm hiểu về lý thuyết bài toán tối ưu, bài toán tối ưu đa mục tiêu và thuật toán tiến hóa. Đồ án sẽ trình bày về các khái niệm, quan hệ trong bài toán đa mục tiêu, đưa ra các phương pháp giải của loại bài toán này. Tiếp đến là thuật toán tiến hóa qua các giai đoạn và làm rõ từng giai đoạn mà đồ án sử dụng phù hợp với bài toán đồ án.

## 3.1 Bài toán tối ưu

### 3.1.1 Khái niệm

Bài toán tối ưu là bài toán tìm kiếm giải pháp tốt nhất trong một tập hợp các giải pháp khả thi, dựa trên các ràng buộc và hàm mục tiêu. Mục tiêu của bài toán tối ưu là tìm ra giá trị tối ưu (lớn nhất hoặc nhỏ nhất) của hàm mục tiêu thỏa mãn các ràng buộc cho trước [23].

Mỗi bài toán tối ưu gồm 2 thành phần $(X, f)$ [23]:

- $X$: Tập các lời giải khả thi (không gian tìm kiếm).
- $f$: Hàm mục tiêu của bài toán cần tối ưu.

### 3.1.2 Phân loại

Phân loại bài toán tối ưu có thể dựa trên nhiều tiêu chí như ràng buộc, kiểu biến tối ưu, loại hàm mục tiêu, số lượng hàm mục tiêu. Dưới đây là phân loại thông qua các tiêu chí phổ biến [23]:

- Phân loại bài toán tối ưu dựa trên loại ràng buộc:
  - Bài toán tối ưu không ràng buộc: Không có ràng buộc áp đặt lên biến tối ưu.
  - Bài toán tối ưu có ràng buộc: Có ràng buộc áp đặt lên biến tối ưu.
- Phân loại bài toán tối ưu dựa trên loại biến tối ưu:
  - Bài toán tối ưu liên tục: Biến tối ưu có giá trị liên tục trong một khoảng xác định.
  - Bài toán tối ưu rời rạc: Biến tối ưu chỉ nhận các giá trị rời rạc (thường là số nguyên).
- Phân loại bài toán tối ưu dựa trên loại hàm mục tiêu:
  - Bài toán tối ưu hóa tuyến tính: Hàm mục tiêu và ràng buộc đều là tuyến tính.

  - Bài toán tối ưu hóa phi tuyến: Hàm mục tiêu hoặc ràng buộc là phi tuyến.

- Phân loại bài toán tối ưu dựa trên số lượng hàm mục tiêu:
  - Bài toán tối ưu hóa đơn mục tiêu: Có một hàm mục tiêu cần tối ưu.
  - Bài toán tối ưu đa mục tiêu: Có nhiều hơn một hàm mục tiêu cần tối ưu.

### 3.1.3 Các hướng tiếp cận giải

- Sử dụng thuật toán chính xác: như phương pháp đơn hình, giải thuật nhánh cận, hoặc quy hoạch động là các thuật toán có khả năng đưa ra giải pháp tối ưu chính xác cho bài toán tối ưu. Tuy nhiên, phương pháp này thường tốn nhiều thời gian tính toán và không phù hợp cho các bài toán lớn với quy mô lớn và số lượng ràng buộc cao.
- Sử dụng thuật toán xấp xỉ gần đúng: Phương pháp giải gần đúng như thuật toán độ dốc (Gradient Descent), thuật toán leo đồi (Hill Climbing), thuật toán tiến hoá (Genetic Algorithms)... cố gắng tìm giải pháp gần đúng mà không đảm bảo tìm được giải pháp tối ưu chính xác. Tuy nhiên, phương pháp này thường nhanh hơn và có thể áp dụng cho các bài toán lớn và phức tạp hơn. Các phương pháp này thường dựa trên các kỹ thuật khai thác cấu trúc và thông tin của bài toán để tìm kiếm một giải pháp gần tối ưu.

Với các bài toán khó, độ phức tạp lớn thì không dễ dàng giải quyết bằng thuật toán chính xác, đặc biệt các bài toán thực tế. Các phương pháp giải gần đúng thường rất hữu ích khi thời gian tính toán là hạn chế hoặc khi bài toán tối ưu không thể giải chính xác trong thời gian quy định. Mặc dù chúng không đảm bảo tìm được giải pháp tối ưu toàn cục, nhưng thường mang lại giải pháp đủ tốt và gần như tối ưu trong thực tế.

Trong bài toán tối ưu thường đưa về một mục tiêu và tìm điểm tối ưu nhất của bài toán. Tuy nhiên khi giải quyết trong thực tiễn có nhiều mục tiêu có mức độ ưu tiên khác nhau. Ví dụ việc lựa chọn mua một chiếc xe có hai tiêu chí là giá và tốc độ, khi một người mức thu nhập cao chọn mua một chiếc xe thì tiêu chí về giá có mức độ ưu tiên thấp hơn người có mức thu nhập thấp. Vậy nên việc đưa ra một giải pháp phù hợp với mỗi người là khác nhau, cần chia ra thành các mục tiêu mà phù hợp với mỗi người. Trong bài toán tối ưu cũng vậy, khi ta có thể chia nhỏ bài toán thành các bài toán con và tối ưu các bài toán con đó ta sẽ tìm được các lời giải khác nhau cho bài toán và phù hợp cho mỗi đối tượng riêng. Phần tiếp theo đồ án xin được trình bày về bài toán tối ưu đa mục tiêu.

## 3.2 Bài toán tối ưu đa mục tiêu

Bài toán tối ưu đa mục tiêu là bài toán tối ưu hai hay nhiều hàm mục tiêu đồng thời [24]. Bài toán tối ưu đa mục tiêu thường xuất hiện khi có nhiều mục tiêu mâu thuẫn và không thể tối ưu đồng thời. Ví dụ, trong quá trình thiết kế có hai mục tiêu là tối thiểu hóa chi phí, tối đa hóa hiệu suất. Tuy nhiên việc thay đổi thiết kế để giảm chi phí có thể dẫn đến làm giảm hiệu suất.

### 3.2.1 Định nghĩa

Xét bài toán tối ưu đa mục tiêu [24]:

$$\text{minimize } F(x) = (f_1(x), f_2(x), ..., f_m(x))^T, \text{với } x \in \Omega$$

trong đó,

- $\Omega \in R^n$ là không gian thiết kế,
- $x = (x_1, ..., x_n)$ được gọi là véc tơ quyết định,
- $m$ là số lượng các hàm mục tiêu $m \geq 2$,
- $f_i(x) : R^n \to R$ với $i = 1, 2.., m$.

**Định nghĩa 3.2.1 Quan hệ dominates [24]:** Cho hai vec tơ quyết định $x$ và $x^*$ thuộc $\Omega$:

- $x \preceq x^*$ ($x$ dominates yếu $x^*$) nếu và chỉ nếu $f_i(x) \leq f_i(x^*)$, $i = 1, ...m$
- $x \prec x^*$($x$ dominates $x^*$) nếu và chỉ nếu $f_i(x) \leq f_i(x^*)$ và $\exists i : f_i(x) < f_i(x^*), i = 1, .., m$
- $x \prec\prec x^*$($x$ dominates tuyệt đối $x^*$) nếu và chỉ nếu $f_i(x) < f_i(x^*)$, $i = 1, ...m$
- $x||x^*$ ($x$ và $x^*$ không thể so sánh) nếu $x$ không dominates yếu $x^*$ và ngược lại.

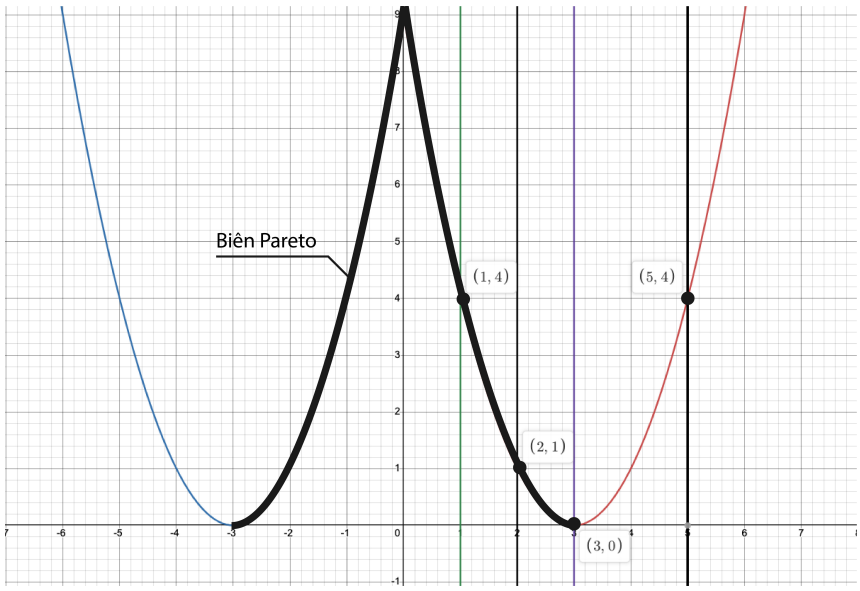
**Định nghĩa 3.2.2 Tối ưu Pareto và giải pháp Pareto [24]:** Một véc tơ quyết định $x^* \in \Omega$ được gọi là tối ưu Pareto nếu và chỉ nếu không tồn tại một véc tơ quyết định $x$ trong không gian thiết kế $\Omega$ mà $x$ dominates $x^*$. Một tập các giải pháp tối ưu Pareto được gọi là tập Pareto, và tập Pareto này khi biểu diễn trên đồ thị (theo hàm mục tiêu) được gọi là biên Pareto. Để giải quyết bài toán tối ưu đa mục tiêu, cần tìm được tập Pareto.

**Ví dụ 3.2.1:**

$$\text{minimize } F(x) = (f_1(x), f_2(x))^T, \text{với } x \in R$$

trong đó,

- $f_1(x) = (x-3)^2$
- $f_2(x) = (x+3)^2$




**Hình 3.1:** Đồ thị hai hàm số $f_1(x) = (x-3)^2$ và $f_2(x) = (x+3)^2$.

Trong hình 3.1 tại điểm:

$x^1 = 1 : f_1(x) = 4; f_2(x) = 16$

$x^2 = 2 : f_1(x) = 1; f_2(x) = 25$

$x^3 = 3 : f_1(x) = 0; f_2(x) = 36$

$x^4 = 5 : f_1(x) = 4; f_2(x) = 64$

Dễ thấy rằng từ một giải pháp $x^1 = 1$ khi ta tiến lên $x^2 = 2$ thì sẽ tối ưu cho hàm $f_1(x)$ nhưng lại làm tăng giá trị hàm $f_2(x)$. Do đó hai hàm mục tiêu mâu thuẫn và không thể tối ưu đồng thời. Ngoài ra ta sẽ thấy rằng $x^1$ trội hơn $x^4$ ($x^1 \prec x^4$), vì $f_1(x^1) = f_2(x^4)$ và $f_2(x^1) < f_2(x^4)$, khi đó $x^4$ được cho là bị trội.

Ba giải pháp $x^1, x^2, x^3$ được cho là tập không bị trội vì không có giải pháp nào dominates giải pháp trong tập đó. Do đó ba điểm tạo thành tập Pareto. Trong ví dụ trên khi tìm các giải pháp không bị trội đủ nhiều và phân bố đều thì tạo nên một đường được nối bởi hai đồ thị:

$$f(x) = (x+3)^2 \text{ nếu } 0 > x \geq -3; f(x) = (x-3)^2 \text{ nếu } 3 \geq x \geq 0;$$

### 3.2.2 Các phương pháp giải

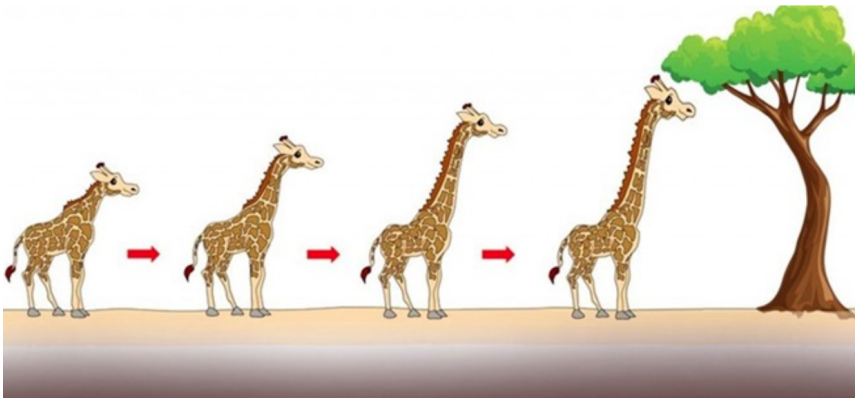
Để giải quyết bài toán đa mục tiêu, có nhiều phương pháp khác nhau được phát triển. Một số phương pháp phổ biến như: phương pháp Simplex (đưa về bài

toán tuyến tính đa mục tiêu), các phương pháp giải theo tiến hóa đa mục tiêu, các phương pháp dựa trên học máy (học tăng cường đa mục tiêu - Multi objective Reinforcement Learning). Mỗi phương pháp sẽ phù hợp với các bài toán cụ thể riêng.

Các phương pháp kể trên thì phương pháp giải theo tiến hoá đa mục tiêu có nhiều ưu điểm như: khả năng xử lý không gian tìm kiếm lớn, khả năng tránh các điểm cục bộ cao, hiệu suất tốt với các bài toán không liên tục. Với lẽ đó tác giả cho rằng tiến hoá đa mục tiêu phù hợp với bài toán đặt chức năng mạng ảo và định tuyến trong mạng ảo hóa. Vậy thuật toán tiến hoá là gì? Ta sẽ tìm hiểu trong mục tiếp theo.

## 3.3 Thuật toán tiến hoá

Học thuyết Darwin, hay Học thuyết tiến hóa của Darwin là một học thuyết về tiến hóa sinh học được đề xướng chủ yếu bởi nhà tự nhiên học người Anh Charles Darwin (1809–1882), cùng một số nhà nghiên cứu khác, phát biểu rằng mọi loài sinh vật xuất hiện và phát triển nhờ quá trình chọn lọc tự nhiên. Trong quá trình này, những biến dị cá thể (gọi là biến dị di truyền) nhỏ nhặt, nếu làm tăng khả năng cạnh tranh, sinh tồn và sinh sản của cá thể thì sẽ được chọn lọc - với nội dung là: giữ lại, củng cố và tăng cường - trở thành đặc điểm thích nghi. Học thuyết này ban đầu bao gồm các khái niệm rộng hơn về đột biến loài hay về tiến hóa, thứ được giới khoa học nói chung công nhận một cách rộng rãi sau khi Darwin xuất bản cuốn Nguồn gốc các loài vào năm 1859, và nó còn bao gồm cả những khái niệm có từ trước khi học thuyết của Darwin ra đời [25].



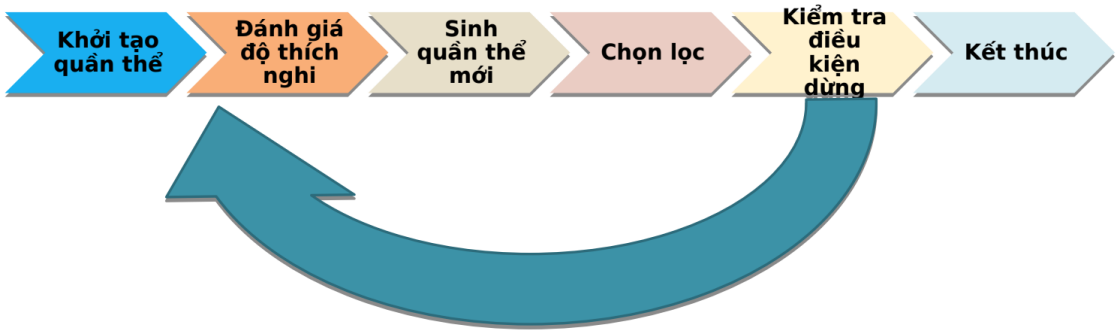
**Hình 3.2:** Sự tiến hóa hươu cao cổ.

Trong hình 3.2 mô tả quá trình tiến hoá của loài hươu. Lấy ý tưởng từ Charles Darwin – liên quan đến thuyết chọn lọc tự nhiên – và các nhà khoa học giải thích rằng trong số những con hươu cao cổ tổ tiên có một vài con có cổ dài hơn một chút

so với đồng loại của chúng. Điều này mang lại cho chúng lợi thế là có thể vươn đến những nhánh cây cao hơn và có thể có thêm thức ăn. Dần dần, những con hươu cao cổ này đã thành công hơn trong việc sinh sản (vì chúng tiếp cận được nguồn thức ăn phong phú hơn), trong khi số lượng những con có cổ thấp hơn dần thu hẹp lại [26].

Học thuyết Darwin đã tạo ra sự hiểu biết sâu sắc về quá trình tiến hóa, cung cấp cơ sở cho nhiều lĩnh vực trong khoa học, bao gồm sinh học, địa chất, và cả khoa học tính toán. Nó cũng đã góp phần thay đổi cách nhìn của con người về vị trí và nguồn gốc của chúng ta trong thế giới tự nhiên. Tuy học thuyết tiến hóa Darwin đã được phát triển từ thế kỷ 19, nhưng nó vẫn là một phần quan trọng của việc nghiên cứu và hiểu về sự đa dạng và tiến hóa của các loài sống cho đến ngày nay [25].

Thuật toán tiến hóa là phương pháp tối ưu hóa dựa trên quá trình tiến hóa trong tự nhiên. Nó là một phương pháp mô phỏng quá trình tiến hóa và sử dụng các cơ chế như lựa chọn, lai ghép, đột biến và tái sinh để tìm kiếm và cải thiện các lời giải gần đúng cho các bài toán tối ưu.




**Hình 3.3:** Sơ đồ thuật toán tiến hoá.

Hình 3.3 mô tả quá trình tiến hoá của thuật toán tiến hoá. Trong thuật toán tiến hóa, lời giải của bài toán được biểu diễn dưới dạng cá thể đặc trưng bởi gen, mỗi cá thể đại diện cho một lời giải trong không gian tìm kiếm. Mỗi gen có chứa các nucleotide cấu tạo nên gen. Quần thể là tập hợp các cá thể, và quá trình tiến hóa xảy ra trong không gian của quần thể.

Các bước cơ bản trong thuật toán tiến hóa gồm:

- Khởi tạo quần thể ban đầu: Một tập hợp các cá thể ngẫu nhiên được tạo ra để tạo thành quần thể ban đầu.
- Đánh giá: Mỗi cá thể trong quần thể được đánh giá dựa trên một hàm mục tiêu

để đo lường hiệu suất của nó.

- Chọn lọc: Các cá thể tốt nhất được chọn để tiếp tục vào thế hệ tiếp theo dựa trên quy tắc lựa chọn. Các cá thể tốt nhất có khả năng sinh sản cao hơn và truyền lại các đặc điểm tốt cho thế hệ tiếp theo.
- Lai ghép: Hai cá thể được chọn ngẫu nhiên từ quần thể để tạo ra các con cá thể mới thông qua các phép lai ghép. Quá trình lai ghép tạo ra sự kết hợp của các đặc điểm của hai cha mẹ, tạo ra sự đa dạng và khám phá trong không gian tìm kiếm.
- Đột biến: Một số cá thể con được chọn ngẫu nhiên để trải qua phép đột biến, nghĩa là thay đổi một số đặc điểm của chúng. Điều này giúp tăng tính đa dạng và khám phá trong quá trình tìm kiếm.
- Tái sinh: Thế hệ mới của quần thể được tạo ra thông qua lựa chọn, lai ghép và đột biến. Lặp lại bước 2 cho đến khi điều kiện dừng được đáp ứng.
- Đưa ra giải pháp tối ưu: Cuối cùng, giải pháp tốt nhất sau một số thế hệ được chọn làm kết quả tối ưu cho bài toán.

Các giai đoạn của thuật toán cho mỗi bài toán sẽ có các phương pháp khác nhau. Phần tiếp theo đồ án xin trình bày các phương pháp điển hình trong thuật toán tiến hoá, và các phương pháp được sử dụng trong bài toán đặt chức năng mạng ảo và định tuyến trong mạng ảo hóa sẽ được làm rõ hơn.

### 3.3.1 Biểu diễn cá thể

Mỗi bài toán khi sử dụng thuật toán tiến hoá để giải quyết đều phải mã hoá lời giải trước khi bước vào các giai đoạn của thuật toán tiến hoá. Sau đây đồ án trình bày các phương pháp mã hoá lời giải:

- Mã hoá nhị phân: Gen là tập các giá trị 0 và 1.
- Mã hoá hoán vị: Là một hoán vị của một tập các gen.
- Mã hoá đa giá trị: thường sử dụng mã hoá giá trị phức tạp (số thực, rời rạc từ một tập vô hạn hoặc hữu hạn).
- Mã hoá cây: Thường phức tạp dùng trong các bài toán đặc biệt.
- Rời rạc hóa.

### 3.3.2 Lai ghép

Với mỗi cách mã hoá, biểu diễn cá thể sẽ có phương pháp lai ghép khác nhau, phù hợp với cách mã hoá đó. Như đã trình bày các phương pháp mã hoá ở trên ta có:

- Mã hoá nhị phân, đa giá trị:

  Do cách mã hoá bài toán đặt chức năng mạng ảo và định tuyến trong mạng ảo hoá đồ án sử dụng mã hoá đa giá trị. Chính vì thế đồ án xin đi sâu hơn phương pháp lai ghép cho cách mã hoá này.

  – Lai ghép theo điểm cắt

    * Một điểm cắt: với hai giải pháp cha ta tìm điểm cắt nằm trong đồ dài gen đã cho, tiến hành ghép hai phần của hai cha lại cho ra hai con lai mới.

      **Ví dụ 3.3.2.1:**

      Cho hai cha có gen như sau:

      $x^1 = [\mathbf{1}, \mathbf{2}, \mathbf{2}, 4, 1, 3]; x^2 = [\mathbf{2}, \mathbf{4}, \mathbf{3}, 2, 1, 3]$

      Sau khi lai ghép điểm cắt tại vị trí thứ 3, ta nhận được hai con có gen:

      $y^1 = [\mathbf{1}, \mathbf{2}, \mathbf{2}, 2, 1, 3]; y^2 = [\mathbf{2}, \mathbf{4}, \mathbf{3}, 4, 1, 3]$

    * Nhiều điểm cắt ($n$ điểm): tương tự như một điểm cắt, ta thực hiện lai ghép chọn ra lần lượt $n + 1$ đoạn gen tương ứng và sẽ có $2^{n+1} - 2$ con lai được tạo ra.

  – Lai ghép đồng bộ: Ở mỗi nucleotide cấu tạo nên gen con sẽ được chọn ngẫu nhiên từ nucleotide của gen cha hoặc gen mẹ theo tỉ lệ $1 : 1$.

    **Ví dụ 3.3.2.2:**

    Cho hai cha có kiểu gen như sau: $x^1 = [\mathbf{1}, \mathbf{2}, 2, 4, \mathbf{1}, 3]; x^2 = [2, 4, \mathbf{3}, \mathbf{2}, 1, \mathbf{3}]$.

    Nucleotide đầu tiên của con lai sẽ được chọn ngẫu nhiên $(x_1^1, x_1^2)$ là $(\mathbf{1}, 2)$.

    Nucleotide thứ hai của con lai sẽ được chọn ngẫu nhiên $(x_2^1, x_2^2)$ là $(\mathbf{2}, 4)$.

    Nucleotide thứ ba của con lai sẽ được chọn ngẫu nhiên $(x_3^1, x_3^2)$ là $(2, \mathbf{3})$.

    Tương tự cho các nucleotide khác đến nucleotide cuối cùng. Ta thu được một con lai mới chẳng hạn như: $y = [$**1, 2, 3, 2, 1, 3**$]$.

- Mã hoá hoán vị: Lai ghép thứ tự, Lai ghép tương hợp bộ phận, Lai chu trình.
- Mã hoá cây: Lai ghép trộn cành.

### 3.3.3 Đột biến

Các phương pháp đột biến điển hình như đảo bít, đổi chỗ, đổi giá trị, đảo đoạn, đột biến cây. Tuỳ vào bài toán và cách mã hoá ta sẽ chọn ra cách đột biến phù hợp. Do cách mã hoá bài toán đặt chức năng mạng ảo và định tuyến trong mạng ảo hoá

đồ án sử dụng mã hoá đa giá trị. Đồ án xin trình bày phép đột biến đổi giá trị.

**Ví dụ 3.3.2.3:**

Cho cha có kiểu gen như sau: $x = [1, 2, 2, 4, 1, \mathbf{3}]$;

Sau khi đột biến: $y = [1, 2, 2, 4, 1, \mathbf{2}]$;

### 3.3.4 Cơ chế chọn lọc sinh tồn

Các phương pháp chọn lọc: Chọn lọc ngẫu nhiên, Chọn lọc theo vòng quay roulette, Chọn lọc theo xếp hạng, Chọn lọc theo thể thức giao đấu.

Như vậy trong mục này đã làm rõ được thuật toán di truyền qua các giai đoạn. Trong chương tiếp theo đồ án xin được trình bày tổng quan về bài toán đặt các chức năng mạng và định tuyến các yêu trong mạng ảo hoá.

# CHƯƠNG 4. TIẾN HOÁ ĐA MỤC TIÊU GIẢI BÀI TOÁN ĐẶT VÀ ĐỊNH TUYẾN TRONG MẠNG ẢO HOÁ

Chương này gồm cách mã hoá một lời giải, phương pháp định tuyến và các giải thuật tiến hoá đa mục tiêu để giải quyết bài toán đồ án. Mỗi thuật toán sẽ có các giai đoạn, phương pháp khác nhau, tuy nhiên đều dùng chung một cách mã hoá và định tuyến. Các tham số trong thuật toán tiến hóa đa mục tiêu sẽ được lựa chọn dựa trên thực nghiệm từ kết quả thu được và trình bày trong chương sau (**Chương 4**).

## 4.1 Mã hoá

Trong phần này có các ký hiệu thường hay được sử dụng là $N_{server}, N_{switch}$ và $N$. Trong đó $N_{server}$ là số lượng nút server, $N_{switch}$ là số lượng nút switch và $N$ là số lượng nút trong mạng, $N = N_{server} + N_{switch}$

Mỗi lời giải được biểu diễn bởi một véc tơ là $X$ gồm $2N_{server} + S$ thành phần trong đó $S$ là tổng tất cả VNF được cài đặt trong mạng

Ta có:

$X = [x_1, x_2, ..., x_{2N_{server}+S}]$, $x_i$ nhận các số nguyên không âm, $i = 1, 2, .., 2N_{server}+S$

Trong đó:

- Với $i = 1, ..N_{server}$ thì $x_i$ là id server $i$, $x_i \leq N$
- $S = x_{N_{server}+1} + x_{N_{server}+2} + x_{N_{server}+3}...x_{2N_{server}}$
- Với $i = N_{server} + 1, ..2N_{server}$ thì $x_i$ là số lượng VNF cài đặt cho server $i - N_{server}$, $x_i \leq m^{VNF}_{i-N_{server}}$, $m^{VNF}_{i-N_{server}}$ là số lượng VNF được cài đặt tối đa trong nút $i - N_{server}$
- Với $i = 2N_{server}, ..2N_{server} + x_{N_{server}+1}$ thì $x_i$ là kiểu VNF cài đặt trong server $1$ , $x_i$ đôi một khác nhau, tức là không cài đặt hai VNF giống nhau cho một server
- Với $i = 2N_{server} + x_{N_{server}+1}, ..2N_{server} + x_{N_{server}+1} + x_{N_{server}+2}$ thì $x_i$ là kiểu VNF cài đặt trong server $2$ , $x_i$ đôi một khác nhau, tức là không cài đặt hai VNF giống nhau cho một server
- Với $i = 2N_{server} + x_{N_{server}+1} + x_{N_{server}+2}, ..2N_{server} + x_{N_{server}+1} + x_{N_{server}+2} + x_{N_{server}+3}$ thì $x_i$ là các kiểu VNF cài đặt trong server $3$, $x_i$ đôi một khác nhau, tức là không cài đặt hai VNF giống nhau cho một server

  ...

- Với $i = 2N_{server} + S - x_{Nserver}, ..2N_{server} + S$ thì $x_i$ là các kiểu VNF cài đặt trong server $N_{server}$, $x_i$ đôi một khác nhau, tức là không cài đặt hai VNF giống nhau cho một server
- Với $i = 2N_{server}, .., 2N_{server} + S$ thì $x_i \leq |F| - 1$ , và $|F|$ là số loại VNF

**Ví dụ 4.1.1:**


**Hình 4.1:** Ví dụ về một cách mã hoá.

Hình 4.1 trên cho ta các thông tin về một lời giải:

- mạng gồm có 2 nút server là nút $1$ và $4$
- server $1$ cài đặt $3$ loại VNF: $1, 3, 4$
- server $4$ cài đặt $3$ loại VNF: $0, 1, 2$

Như vậy trong quá trình mã hoá ta đã cài đặt các VNF lên các nút. Chính vì thế ta sẽ tính toán được chi phí kích hoạt server (các nút server có số lượng VNF $> 0$) và chi phí cài đặt VNF trên các nút server.

## 4.2 Định tuyến

Ở phần này ta sẽ trình bày cách định tuyến cho toàn bộ yêu cầu SFC. Với mỗi lời giải được khởi tạo trong phần 4.1 là $x$ (cài đặt các VNF vào các nút server) ta thực hiện định tuyến như sau:

- Kiểm tra với cách đặt $x$ có đủ các kiểu VNF mà yêu cầu SFC cần không, nếu không thì cách đặt $x$ là không thỏa mãn và không thể định tuyến.
- Kiểm tra lại số lượng VNF cài đặt trên nút có lớn hơn số lượng VNF tối đa trên nút đó không (việc lai ghép tạo cá thể mới có thể sẽ vi phạm ràng buộc này). Nếu số lượng VNF lớn hơn số lượng VNF tối đa tức là $x$ không thoả mãn và không thể định tuyến.
- Định tuyến:
  - Duyệt tuần tự từng $SFC$.
  - Mỗi $SFC_j$ cho $k_j$ kiểu $VNF$ cần thực hiện vào hàng đợi $q$.
  - Đặt $start = sn_j$.

- Lấy $VNF_{top}$ ra khỏi $q$.
- Gọi $servers_{top}$ là các server có cài đặt $VNF_{top}$.
- Tìm đường đi từ nút $start$ đến các nút $servers_{top}$ tìm server có đường đi độ trễ là nhỏ nhất.
- Với mỗi nút $server_{top}$ trong các nút $servers_{top}$ ta tìm đường theo $P_{server_{top}start}$ trong đó:
  * $P_{server_{top}start}$ là tập tất cả các đường đi từ $start$ đến nút $server_{top}$ (cũng chính là đường đi từ nút $server_{top}$ đến nút $start$) được trình bày ở đoạn sau trong mục 4.3 này.
  * Với $P_{server_{top}start}$ ta chọn $rank$ bắt đầu từ $1$ (Đường đi có độ trễ nhỏ nhất). Nếu không thoả mãn về băng thông, bộ nhớ, CPU cần sử dụng thì ta tăng mức $rank$ lên $rank + 1$ (một đường đi khác tốt nhất có thể). Nếu $rank$ là lớn nhất cũng không thỏa mãn băng thông, bộ nhớ, CPU cần dùng thì cách đặt $x$ là không thoả mãn và không thể định tuyến.
- Sau đó đặt $start = server_{top}$ là nút server có cài đặt $VNF_{top}$ và độ trễ tốt nhất (nhỏ nhất có thể).
- Lặp lại bước lấy $VNF_{top}$ ra khỏi $q$ cho đến khi hàng đợi $q$ rỗng.
- Tìm đường đi từ server cài đặt $VNF$ cuối cùng trong hàng đợi $q$ đến $dn_j$.
- Tính chi phí băng thông, bộ nhớ, CPU của $SFC_j$ đã sử dụng. Tính chi phí về độ trễ (một hàm mục tiêu của bài toán).
- Lặp lại hết các yêu cầu $SFC$ và tính tổng độ trễ của toàn bộ yêu cầu $SFC$.

**Tìm tập đường đi từ nút server đến các nút khác**

Tại thời điểm khởi tạo mạng và các yêu cầu SFC ta đã có thể tính toán tìm ra một tập các kiểu VNF yêu cầu và tìm các đường đi có độ trễ là nhỏ dần từ các nút server đến các nút khác trong mạng bằng thuật toán vét cạn đường đi.

Mục đích nhằm giảm thiểu chi phí về thời gian tính toán trong mỗi thế hệ của thuật toán tiến hoá, tăng độ chính xác trong quá trình định tuyến và quá trình tiến hoá của thuật toán tiến hoá.

Có nhiều thuật toán tìm đường đi tốt hơn, nhưng lý do lựa chọn vét cạn còn là vấn đề về ràng buộc băng thông, bộ nhớ, cpu trong quá trình định tuyến vì thế ngoài đường đi tốt nhất (độ trễ là nhỏ nhất) ta cần tìm các đường đi thay thế trong trường hợp tài nguyên đã sử dụng đạt giới hạn.

Ta sẽ tìm các đường đi từ nút server $u$ đến nút $v$ và tính toán độ trễ (chỉ tính độ

trễ trên cạch $(u, v)$ là $d_{uv}$) và sắp xếp theo $rank$ bắt đầu từ 1, với $u \in V_{server}, v \in V$ với $V$ là các nút trong mạng, $V_{server}$ các nút server. Nếu $u = v$ thì độ trễ bằng 0.

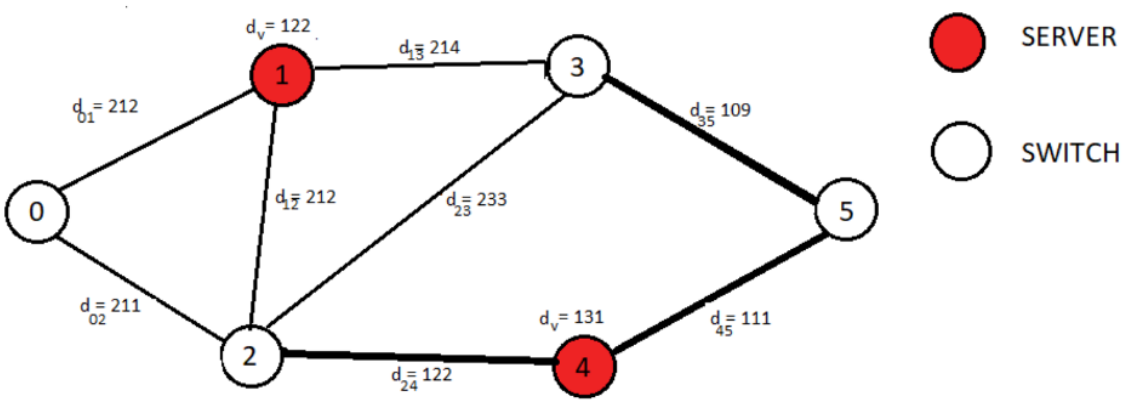
Phương pháp duyệt đồ thị em sử dụng duyệt theo chiều rộng. Và thu được một tập $P$ được kí hiệu như sau:

$P_{uv} = \{p^1_{uv}, p^2_{uv}, ...\}$

trong đó,

- $u \in V_{server}, v \in V$, $V_{server}$ là các nút server, $V$ là các nút trong mạng.
- $P_{uv}$ gồm các tập đường đi từ server $u$ đến nút $v$ được sắp xếp theo độ trễ tăng dần.
- $p^i_{uv}$ là tập các đỉnh được thăm bắt đầu từ $u$ và kết thúc ở $v$ với $rank = i$ ($rank = 1$ là tốt nhất, có độ trễ nhỏ nhất).
- nếu $u = v$ thì chỉ có duy nhất $rank = 1$ và $p^1_{uv}$ = $\{u\}$.

**Ví dụ 4.2.1:**




**Hình 4.2:** Mạng thể hiện độ trễ trên các cạnh và trên server.

Ví dụ như hình 4.2 trên ta có tập $P_{10} = \{p^1_{10}, p^2_{10}, p^3_{10}, p^4_{10}\}$

trong đó:

- $p^1_{10}$ = $\{0, 1\}$, với độ trễ là $d_{01} = 212$.
- $p^2_{10}$ = $\{0, 2, 1\}$, với độ trễ là $d_{02} + d_{12} = 211 + 212 = 423$.
- $p^3_{10}$ = $\{0, 2, 3, 1\}$, với độ trễ là $d_{02} + d_{23} + d_{13} = 211 + 233 + 214 = 658$.
- $p^4_{10}$ = $\{0, 2, 4, 5, 3, 1\}$, với độ trễ là $d_{02} + d_{24} + d_{45} + d_{35} + d_{13} = 211 + 122 + 111 + 109 + 214 = 767$.

Như vậy tập $P_{10}$ của nút server 1 có tất cả là 4 đường đi tương ứng với 4 $rank$ là

$1, 2, 3, 4$.

Tương tự như vậy ta cũng có các tập $P_{11}, P_{12}, P_{13}, P_{14}, P_{15}$ cho nút server $1$.

Với server $4$ ta cũng có tập các tập đường $P_{40}, P_{41}, P_{42}, P_{43}, P_{44}, P_{45}$.

Tóm tắt 2 mục đã trình bày ta có thể hiểu: Mã hoá một lời giải là cách kích hoạt server và cài đặt các VNF lên server đó từ đó ta có thể tính toán được hàm mục tiêu 2 và hàm mục tiêu 3 của bài toán (chi phí cài đặt server và chi phí cài đặt các VNF). Định tuyến là ràng buộc của lời giải đó, giúp ta kiểm tra lời giải có thể định tuyến không và tính toán được độ trễ là hàm mục tiêu 1 của bài toán. Ở phần sau đồ án trình bày cách để khởi tạo một lời giải.

## 4.3 Khởi tạo

Khởi tạo là quá trình đầu tiên, là tạo ra một quần thể ban đầu đa dạng và có tiềm năng để tìm kiếm và tiến hoá tới giải pháp tốt nhất. Bằng cách khởi tạo một quần thể phù hợp, ta có thể đặt nền tảng cho quá trình tiến hoá, cho phép thuật toán khám phá không gian tìm kiếm và tìm ra các giải pháp tiềm năng.

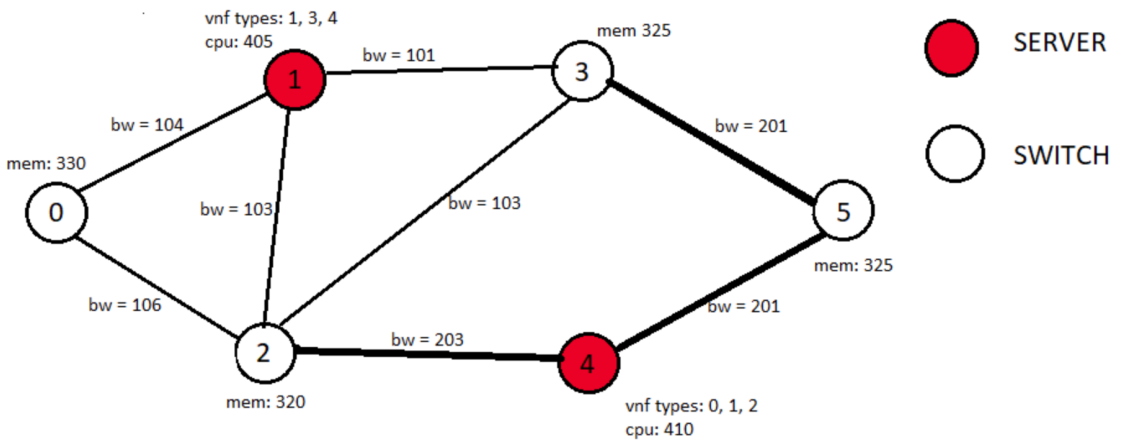
Quá trình khởi tạo một quần thể cho các thuật toán trong đồ án này là như nhau, điểm khác duy nhất là số lượng cá thể trong quần thể của mỗi thuật toán là khác nhau.

Khởi tạo ngẫu nhiên số lượng VNF trên server $i$ đó trong khoảng $[0, m_i^{VNF}]$ ta tạm gọi là $sl_{VNF}^i$, $i \in V_{server}$ và $m_i^{VNF}$ là số lượng VNF tối đa được cài đặt trong server $i$.

Tiếp tục tạo ngẫu nhiên $sl_{VNF}^i$ số nguyên khác nhau trong khoảng $[1, |F|]$ là các VNF được cài đặt trong server $i$ ta tạm gọi là tập $VNFTypes_i$.

Thực hiện ghép các server lại theo thứ tự (server, số lượng VNF cài đặt, loại VNF cài đặt) ta được một lời giải.

**Ví dụ 4.3 :**




**Hình 4.3:** Mạng khi đã cài đặt các VNF lên server.

Ở ví dụ trên ta có:

$(1, \mathbf{3}, 1, 3, 4)$: $i = 1$, $sl^i_{VNF} = \mathbf{3}$, $VNFTypes_i = \{1, 3, 4\}$

$(4, \mathbf{3}, 0, 1, 2)$: $i = 4$, $sl^i_{VNF} = \mathbf{3}$, $VNFTypes_i = \{0, 1, 2\}$

Sau đó ghép lại theo thứ tự (server, số lượng VNF cài đặt, loại VNF cài đặt) ta được một lời giải:

$$X = \{1, 4, \mathbf{3, 3}, 1, 3, 4, 0, 1, 2\}$$

Một cá thể được cho là thỏa mãn khi có thể định tuyến được và đã trình bày ở phần 4.2

Lặp lại để lấy số lượng cá thể khởi tạo cho mỗi thuật toán

Về không gian giải pháp của bài toán ta có thể tính như sau:

- Giả sử có $N$ node server, $M$ là số lượng VNF được cài đặt tối đa như nhau trong mọi nút.
- Không gian giải pháp của bài toán dễ dàng tính được là:

  $(M + 1)^N$ đã bao gồm cả các trường hợp không thể định tuyến được.

Phần này em đã trình bày giai đoạn khởi tạo quần thể. Phần sau em trình bày các thuật toán tiến hoá đa mục tiêu và sử dụng chung cách khởi tạo quần thể ban đầu như đã đề cập ở đầu mục.

## 4.4 Thuật toán tiến hoá dựa trên dạy-học (MOTLBO)

### 4.4.1 Giới thiệu thuật toán

Thuật toán Multi-Objective Teaching-Learning-Based Optimization (MOTLBO) là một thuật toán tối ưu đa mục tiêu được giới thiệu bởi các nhà nghiên cứu R.V. Rao và V. Jyothi vào năm 2010. Ý tưởng của thuật toán MOTLBO xuất phát từ việc mô phỏng quá trình giảng dạy và học tập trong giáo dục. Trong MOTLBO, các cá thể trong quần thể được coi là học sinh và các giá trị hàm mục tiêu thể hiện điểm số của cá thể đó. Nó đã được áp dụng thành công trong nhiều lĩnh vực, bao gồm kỹ thuật, kinh tế, và xử lý tín hiệu.

Các bước cơ bản của thuật toán như sau [27]:

- Khởi tạo: Tạo ra một quần thể ban đầu gồm các cá thể ngẫu nhiên. Mỗi cá thể đại diện cho một giải pháp tiềm năng trong không gian tìm kiếm.
- Đánh giá: Đánh giá chất lượng của từng cá thể trong quần thể dựa trên các hàm mục tiêu. Đối với bài toán tối ưu đa mục tiêu, mỗi cá thể sẽ được đánh giá trên nhiều hàm mục tiêu.
- Pareto Dominance: Xác định các giải pháp không bị chi phối trong quần thể, tạo thành biên Pareto.
- Phân lớp: Phân loại các cá thể trong quần thể thành hai nhóm: giáo viên và học sinh. Các cá thể giáo viên là những cá thể có chất lượng tốt hơn, trong khi các cá thể học sinh là những cá thể có chất lượng thấp hơn.
- Giảng dạy: Các cá thể học sinh học từ các cá thể giáo viên bằng cách nhân bản và kết hợp thông tin của các cá thể giáo viên. Quá trình này có thể bao gồm các phép toán như lai ghép (crossover) và đột biến (mutation).
- Học tập: Các cá thể học sinh cải thiện bản thân bằng cách học từ các cá thể học sinh khác. Quá trình này có thể bao gồm việc thực hiện các phép toán tối ưu để điều chỉnh các giá trị của cá thể học sinh.
- Cập nhật quần thể: Sau quá trình giảng dạy và học tập, quần thể được cập nhật bằng cách chọn lọc các cá thể tốt nhất từ cả hai nhóm giáo viên và học sinh. Quá trình này giúp duy trì và cải thiện chất lượng của quần thể.
- Kiểm tra điều kiện dừng: Kiểm tra xem điều kiện dừng của thuật toán đã được đáp ứng chưa. Điều kiện dừng có thể là số lượng thế hệ đã trải qua, đạt được giải pháp đủ tốt, hoặc một điều kiện xác định khác. Nếu điều kiện dừng chưa được đáp ứng, quay lại bước đánh giá và tiếp tục quá trình tiến hoá. Ngược lại, thuật toán kết thúc và trả về kết quả tốt nhất tìm được trong quần thể.

### 4.4.2 Thuật toán MOTLBO cho bài toán của đồ án

Trong quá trình triển khai, em lấy ý tưởng của thuật toán nhưng cũng thêm bớt các giai đoạn cho phù hợp với bài toán đặt và định tuyến nhất.

Sau đây là thuật toán em đề xuất và thiết kế:

- Khởi tạo: Khởi tạo $N$ cá thể, cách khởi tạo đã ở trình bày ở mục 4.3.
- Tính giá trị $fitness$ là giá trị 3 hàm mục tiêu của mỗi cá thể gồm: Độ trễ sau khi định tuyến và chi phí cài đặt server, chi phí cài đặt VNF sau khi khởi tạo một lời giải.
- Đánh giá: Tìm ra $fitness_{top}$ là tập các cá thể không bị dominate (cũng chính là tập Pareto cần tìm). Sau đó các cá thể còn lại được chọn ra $remove$ cá thể có tổng hàm mục tiêu là nhỏ nhất (do bài toán là $minimize$ 3 hàm mục tiêu, cũng có một hướng khác là theo rank nhưng như vậy sẽ trùng với cách đánh giá của thuật toán NSGA2), $remove$ là tham số truyền vào, là số lượng cá thể bị loại bỏ sau một thế hệ.
- Giảng dạy: Từ đánh giá trên, các cá thể trong tập $fitness_{top}$ được chọn làm giáo viên. Thực hiện giảng dạy từ các cá thể học sinh là các cá thể không nằm trong diện bị loại bỏ và không nằm trong tập giáo viên . Việc giảng dạy được thực hiện là một phương pháp lai ghép 1 điểm cắt (ngẫu nhiên trong khoảng từ $[N+1, 2N]$, $N$ là số lượng nút) của cá thể giáo viên và cá thể học sinh. Việc cắt một cá thể đã bao gồm cả việc cắt kiểu VNF của cá thể đó. Sau đó tính toán việc thay đổi học sinh đã có thánh một học sinh mới có tốt hơn không, nếu tốt thì thay đổi. Việc đánh giá học sinh tốt nếu tốt hơn chính cá thể học sinh đó (xét tính trội nếu không bị trội lẫn nhau xét đến tổng hàm mục tiêu).

**Ví dụ 4.4:**

Cho hai cá thể cha mẹ $x_{gv}, x_{hs}$:

$x_{gv} = [1, 4, 3, 3, 1, 3, 4, 0, 1, 2]$

$x_{hs} = [1, 4, 2, 2, 1, 4, 2, 3]$

Giả sử điểm cắt là $3$

$x_{gv} = [1, 4,$ **3**$, 3,$ **1, 3, 4**$, 0, 1, 2]$

$x_{hs} = [1, 4, 2,$ **2**$, 1, 4,$ **2, 3**$]$

Ta thu được hai cá thể mới:

$y_1 = [1, 4,$ **3, 2, 1, 3, 4, 2, 3**$]$

$y_2 = [1, 4, 2, 3, 1, 4, 0, 1, 2]$

- Học tập: Giai đoạn học tập là giai đoạn học giữa các học sinh lẫn nhau và thay thế gen cho cả hai nếu "kết quả học tập" là tốt. Việc chọn lựa hai cá thể có được học hay không bằng cách sinh số ngẫu nhiên nhỏ hơn $rate\_crossover$. Quá trình học là phép lai một điểm cắt tương tự như giai đoạn giảng dạy.
- Bổ sung cá thể: Trong giai đoạn đánh giá có loại bỏ $remove$ cá thể ra khỏi quần thể. Việc bổ sung cá thể ngẫu nhiên là cần thiết và cũng đầy tiềm năng.
- Kiểm tra điều kiện dừng: Trong đồ án này em dừng theo thời gian là quá $T$ phút sẽ dừng thuật toán và trả lại tập biên pareto là $fitness_{top}$. Nếu không quay lại bước Tính giá trị hàm mục tiêu.

  Trong mục này đồ án đã trình bày thuật toán MOTLBO qua các giai đoạn và các tham số cần cài đặt gồm: Số lượng cá thể trong quần thể $N$, thời gian dừng $T$, số cá thể bị loại sau một thế hệ $remove$, tỷ lệ lai ghép giai đoạn học tập $rate\_crossover$. Mục tiếp theo đồ án trình bày về thuật toán MOEA/D.

## 4.5 Thuật toán tiến hoá dựa trên phân giã (MOEA/D)

### 4.5.1 Giới thiệu thuật toán

Thuật toán Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) được giới thiệu lần đầu tiên bởi các nhà nghiên cứu Qingfu Zhang và Hui Li vào năm 2007. Ý tưởng gốc của thuật toán MOEA/D xuất phát từ việc kết hợp giữa giải thuật tiến hóa đa mục tiêu và phân rã không gian mục tiêu.

Trước khi MOEA/D ra đời, các thuật toán tiến hóa đa mục tiêu khác đã được phát triển như NSGA-II (Non-dominated Sorting Genetic Algorithm II) và SPEA2 (Strength Pareto Evolutionary Algorithm 2). Tuy nhiên, các thuật toán này thường đối mặt với hiệu suất tính toán chậm và không hiệu quả khi xử lý các bài toán có không gian mục tiêu lớn.

MOEA/D đã đưa ra một phương pháp phân rã hiệu quả để giảm bớt kích thước không gian mục tiêu và cải thiện hiệu suất tính toán. Bằng cách chia không gian mục tiêu thành các vùng con (ô) nhỏ hơn, MOEA/D giúp tập trung vào các phần quan trọng của không gian mục tiêu, giảm bớt số lượng cá thể cần tính toán và tăng tốc quá trình tìm kiếm.

Kể từ khi được giới thiệu, MOEA/D đã trở thành một trong những thuật toán tiến hóa đa mục tiêu phổ biến và được áp dụng rộng rãi trong nhiều lĩnh vực khác nhau, bao gồm tối ưu hóa kỹ thuật, lập lịch sản xuất, lập kế hoạch năng lượng và quản lý tài nguyên.

Nhờ tính hiệu quả và khả năng giải quyết bài toán đa mục tiêu, thuật toán MOEA/D đã góp phần quan trọng vào sự phát triển của lĩnh vực tối ưu hóa đa

mục tiêu trong hơn một thập kỷ qua.

Các bước cơ bản của thuật toán MOEA/D [28]:

- Khởi tạo quần thể: Tạo ra một tập hợp các cá thể (được gọi là quần thể) ban đầu dựa trên các biến ngẫu nhiên hoặc các phương pháp khởi tạo.
- Phân rã các mục tiêu: Chia không gian mục tiêu thành các ô con tương ứng với các vùng quan tâm khác nhau. Mỗi cá thể trong quần thể sẽ được gán cho một ô cụ thể.
- Cập nhật quần thể: Lặp lại quá trình cập nhật cho đến khi điều kiện dừng được đáp ứng. Trong mỗi lần lặp, các bước sau được thực hiện:
  - Lựa chọn các cá thể cha mẹ: Chọn một tập hợp con cá thể từ quần thể để tham gia quá trình tiếp tục.
  - Tạo ra cá thể con: Sử dụng các toán tử như lai ghép (crossover) và đột biến (mutation) để tạo ra con cá thể từ các cá thể cha mẹ đã chọn.
  - Đánh giá và phân loại con cá thể: Đánh giá giá trị mục tiêu của con cá thể mới tạo ra và xác định ô con mục tiêu mà nó thuộc về.
  - Cập nhật lân cận: Cập nhật quần thể lân cận của mỗi cá thể bằng cách so sánh và thay thế các cá thể lân cận hiện tại bằng các cá thể mới.
  - Cập nhật quần thể chính: Cập nhật quần thể chính bằng cách so sánh và thay thế các cá thể trong quần thể chính bằng các cá thể mới.
- Kiểm tra điều kiện dừng: Kiểm tra xem điều kiện dừng đã được đáp ứng chưa. Điều kiện dừng có thể là số lần lặp hoặc đạt được một mức độ hội tụ mong muốn.
- Trả về kết quả: Trả về quần thể cá thể cuối cùng hoặc các lời giải tốt nhất tìm thấy trong quá trình chạy thuật toán.

### 4.5.2 Thuật toán MOEA/D cho bài toán của đồ án

Các bước của thuật toán dùng trong bài toán của đồ án như sau:

- Khởi tạo: Khởi tạo $N$ cá thể, cách khởi tạo đã ở trình bày ở mục 4.3.
- Khởi tạo véc tơ trọng số cho mỗi cá thể $i$: Ta có 3 hàm mục tiêu vì thế ta khởi tạo 3 trọng số: $w_1^i, w_2^i, w_3^i$ tương ứng với hàm mục tiêu về độ trễ, chi phí kích hoạt server, chi phí cài đặt VNF, sao cho $w_1^i + w_2^i + w_3^i = 1$. Như vậy ta cần khởi tạo ngẫu nhiên $N$ bộ trọng số tương ứng với mỗi cá thể đã khởi tạo.
- Tìm láng giềng: Với mỗi cá thể $i$ ta cần tìm $k$ láng giềng gần với $i$ nhất. Ở đây khoảng cách láng giềng được tính bằng khoảng cách giữa hai véc tơ trọng số

theo khoảng cách Euclid giữa hai điểm trong không gian.

**Ví dụ 4.5.1:**

ta có $w_i = [0.4, 0.4, 0.2]; w_j = [0.3, 0.3, 0.4]; w_i = [0.1, 0.1, 0.8]$

khi đó khoảng cách giữa hai cá thể $i, j$ là:

$\sqrt{(w_1^i - w_1^j)^2 + (w_2^i - w_2^j)^2 + (w_3^i - w_3^j)^2}$

$= \sqrt{(0.4 - 0.3)^2 + (0.4 - 0.3)^2 + (0.2 - 0.4)^2} = 0.24495$

khoảng cách giữa hai cá thể $i, k$ là:

$\sqrt{(w_1^i - w_1^k)^2 + (w_2^i - w_2^k)^2 + (w_3^i - w_3^k)^2}$

$\sqrt{(0.4 - 0.1)^2 + (0.4 - 0.1)^2 + (0.2 - 0.8)^2} = 0.73485$

Như vậy cá thể $j$ được cho là láng giềng gần $i$ hơn cá thể $k$

- Tính giá trị $fitness$: Là tích trọng số 3 hàm mục tiêu nhân với giá trị mỗi hàm mục tiêu tương ứng của mỗi cá thể gồm: Độ trễ sau khi định tuyến và chi phí cài đặt server, chi phí cài đặt VNF sau khi khởi tạo một lời giải:

  $fitness_i = w_1^i * DL_i + w_2^i * CS_i + w_2^i * CV_i$

  trong đó,

  – $DL_i$ là giá trị hàm mục tiêu về độ trễ của cá thể $i$.

  – $CS_i$ là giá trị hàm mục tiêu về chi phí kích hoạt server của cá thể $i$.

  – $CV_i$ là giá trị hàm mục tiêu về chi phí cài đặt VNF của cá thể $i$.

  (có thể xem lại ở phần 2.2.4 của bài toán)

- Cập nhật quần thể:

  – Chọn ngẫu nhiên một cá thể $i$ trong quần thể, $i = 1, .., N$.

  – Tạo ra cá thể con từ cá thể $i$:

  * Lai ghép $k + 1$ cá thể trong quần thể: Bao gồm cá thể $i$ được chọn và $k$ cá thể láng giềng gần nhất của $i$. Phương pháp lai ghép sử dụng $k$ điểm cắt. Với cách chọn từng đoạn gen theo tổ hợp hoán vị sao cho mỗi cá thể có một đoạn gen được chọn. Như vậy ta sẽ tạo ra $(k + 1)!$ cá thể con mới .

    **Ví dụ 4.5.2:**

    cho ba gen sau và lai ghép theo 2 điểm cắt:

    $x_1 = [1, 3, 4, 2, 4, 3, 0, 1, 2, 3, 4, 5, 0, 2, 5]$

$x_2 = [1, 3, 4, 1, 2, 3, 0, 3, 4, 1, 2, 5]$

$x_3 = [1, 3, 4, 2, 3, 3, 0, 2, 2, 3, 5, 1, 4, 5]$

trong ví dụ có 3 server là $1, 3, 4$ vì thế 2 điểm cắt chia thành 3 đoạn gen, mỗi đoạn chứa 1 server. Ta chọn từng đoạn gen trên mỗi gen ở các vị trí cắt khác nhau (chọn trên 3 server).

$x_1 = [1, 3, 4, \mathbf{2}, 4, 3, \mathbf{0, 1}, 2, 3, 4, 5, 0, 2, 5]$

$x_2 = [1, 3, 4, 1, \mathbf{2}, 3, 0, \mathbf{3, 4}, 1, 2, 5]$

$x_3 = [1, 3, 4, 2, 3, \mathbf{3}, 0, 2, 2, 3, 5, \mathbf{1, 4, 5}]$

như vậy ta thu được một gen mới:

$y_1 = [1, 3, 4, \mathbf{2, 2, 3, 0, 1, 3, 4, 1, 4, 5}]$

tương tự như thế ta có tất cả là $3!$ gen mới

∗ Đột biến cá thể được chọn:

Phương pháp đột biến kiểu VNF: Chọn ra một kiểu VNF trong một server tiến hành thay thế bằng một kiểu VNF khác, không trùng với các kiểu VNF hiện đang có trong server.

**Ví dụ 4.5.3:**

$x_1 = [1, 3, 4, 2, 4, 3, 0, 1, 2, 3, 4, 5, 0, 2, \mathbf{5}] \rightarrow x_1 = [1, 3, 4, 2, 4, 3, 0, 1, 2, 3, 4, 5, 0, 2, \mathbf{1}]$

Phương pháp đột biến toàn bộ gen: Bằng cách khởi tạo $m$ cá thể mới tính hàm mục tiêu theo trọng số của cá thể $i$ sau đó chọn ra cá thể tốt là gen đột biến.

Cách đột biến toàn bộ gen sẽ là rất hiệu quả trong các thế hệ đầu của tiến hoá. Tuy nhiên giai đoạn giữa và sau đột biến kiểu VNF sẽ là tốt hơn khi mà trong gen đã có nhiều nucleotide tốt.

– Đánh giá và cập nhật: Nếu cá thể mới được tạo ra tốt hơn cá thể $i$ thì sẽ thay thế bằng cá thế mới đó.

– Cập nhật lân cận: Nếu cá thể mới được tạo ra tốt hơn cá thể thuộc $k$ cá thể láng giềng của cá thể $i$ thì sẽ thay thế, nếu không giữ nguyên gen hiện tại.

– Kiểm tra điều kiện dừng: Trong đồ án này em dừng theo thời gian là quá $T$ phút sẽ dừng thuật toán.

• Đưa ra tập lời giải là $N$ cá thể trong quần thể.

Trong mục này đồ án đã trình bày thuật toán MOEA/D qua các giai đoạn và các tham số cần cài đặt: Số cá thể $N$, số láng giềng gần nhất $k$. Mục tiếp theo đồ án trình bày về thuật toán NSGA-II.

## 4.6 Thuật toán di truyền sắp xếp không thống trị (NSGA-II)

### 4.6.1 Giới thiệu thuật toán

Non-dominated Sorting Genetic Algorithm II (NSGA-II) là một thuật toán tối ưu hóa đa mục tiêu được phát triển bởi Kalyanmoy Deb, Anindya Ghosh, Dipankar Dasgupta và Himanshu Joshi. Thuật toán NSGA-II được công bố lần đầu vào năm 2002 trong một bài báo mang tên "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II" [29].

NSGA-II là một phiên bản nâng cấp của thuật toán NSGA ban đầu, được Deb đề xuất vào năm 2000. NSGA được thiết kế để giải quyết các bài toán tối ưu đa mục tiêu bằng cách sử dụng ý tưởng sắp xếp không trội (non-dominated sorting) và xếp hạng của các cá thể. Tuy nhiên, NSGA gặp một số hạn chế về hiệu suất tính toán và độ phức tạp của thuật toán.

Với NSGA-II, Deb và đồng nghiệp đã giới thiệu một số cải tiến để nâng cao hiệu suất và độ phức tạp của thuật toán. NSGA-II sử dụng một phương pháp chọn lọc cá thể mới được gọi là "non-dominated sorting" dựa trên mức độ tối ưu của chúng. Thuật toán cũng sử dụng các toán tử di truyền như lai ghép và đột biến để tạo ra các thế hệ tiếp theo.

Một điểm đáng chú ý khác của NSGA-II là việc sử dụng một thuật toán là sắp xếp nhanh không bị trội (fast non-dominated sorting) để tăng tốc quá trình sắp xếp các cá thể. Thuật toán này giúp NSGA-II hiệu quả hơn so với NSGA trong việc xác định các tập hợp cá thể không bị trội.

Kể từ khi được công bố, NSGA-II đã nhận được sự quan tâm rộng rãi trong cộng đồng tối ưu hóa đa mục tiêu. Nó đã được áp dụng thành công trong nhiều lĩnh vực như tối ưu hóa kỹ thuật, thiết kế máy móc, lập lịch sản xuất và nhiều bài toán thực tế khác. NSGA-II được coi là một trong những thuật toán phổ biến và mạnh mẽ nhất trong lĩnh vực tối ưu hóa đa mục tiêu.

Trong bài báo "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", các tác giả trình bày các bước cơ bản của thuật toán NSGA-II như sau [29]:

- Khởi tạo quần thể ban đầu: Tạo một quần thể ban đầu chứa $N$ cá thể ngẫu nhiên từ không gian tìm kiếm.
- Đánh giá: Đánh giá tính khả thi của các cá thể trong quần thể ban đầu, xác định xem chúng có thỏa mãn các ràng buộc của bài toán hay không.

- Gán hạng: Sử dụng sắp xếp nhanh không bị trội (fast non-dominated sorting) để xếp hạng ($rank$) các cá thể trong quần thể. Các cá thể không bị trội được xếp vào hạng $1$, các cá thể bị trội bởi cá thể hạng $1$ được xếp vào hạng $2$, và tiếp tục cho đến khi xếp hạng cho tất cả các cá thể trong quần thể.
- Tính toán khoảng cách phân bố: Tính toán khoảng cách phân bố của các cá thể trong cùng một hạng. Khoảng cách phân bố đo lường độ "dày đặc" của các cá thể trong không gian mục tiêu. Các cá thể có khoảng cách phân bố lớn được ưu tiên để lựa chọn trong quá trình chọn lọc.
- Lựa chọn: Sử dụng phương pháp lựa chọn cá thể dựa trên hạng (rank) và khoảng cách phân bố, chọn lọc các cá thể để tạo thành quần thể con cho thế hệ tiếp theo.
- Lai ghép: Áp dụng toán tử lai ghép để tạo ra cá thể con từ quần thể được chọn lọc ở bước trước. Sử dụng các toán tử lai ghép như lai ghép hai điểm cắt hoặc lai ghép đơn điểm...
- Đột biến: Áp dụng toán tử đột biến để biến đổi ngẫu nhiên một số cá thể con đã được lai ghép. Điều này giúp đảm bảo tính đa dạng của quần thể.
- Gộp quần thể: Kết hợp quần thể gốc và quần thể con sau khi đã lai ghép và đột biến.
- Xóa cá thể: Nếu quần thể hiện tại lớn hơn kích thước quần thể cho phép, thực hiện việc loại bỏ một số cá thể dựa trên khoảng cách phân bố hoặc hạng để giảm kích thước quần thể về kích thước ban đầu.
- Lặp lại bước Đánh giá cho số đủ số lần lặp qua các thế hoặc một điều kiện dừng khi kết quả đủ tốt.

Trong bước gán hạng của thuật toán có nhắc đến sự cải tiến về việc sắp xếp các cá thể theo hạng vậy thuật toán có gì đặc biệt ta đi vào tìm hiểu trong mục tiếp theo.

### 4.6.2 Thuật toán sắp xếp nhanh không bị trội (fast non-dominated sorting)

Đầu vào: Tập hợp các cá thể trong quần thể $P$.

Đầu ra: Tập hợp các rank Pareto với thứ tự từ $1$ đến $k$, trong đó rank thứ $1$ chứa các cá thể không bị trội, rank thứ $2$ chứa các cá thể chỉ bị trội bởi rank thứ $1$, và tiếp tục cho đến khi không còn cá thể bị trội.

- Khởi tạo $S$ (tập hợp trống) và $n$ (số lượng tham số tối ưu).
- Khởi tạo mảng $Front$, trong đó $Front[i]$ là tập hợp các cá thể trong $rank$ thứ $i$.

- Tính toán rank dominates của từng cá thể trong $P$. Ban đầu, đặt rank dominates của tất cả các cá thể là $0$.
- Duyệt qua từng cá thể $p$ trong $P$:
  - Đặt $S[p]$ (tập hợp trống) và $n[p]$ (số lượng cá thể dominates $p$) là $0$.
  - Duyệt qua từng cá thể $q$ trong $P$, trừ cá thể $p$:
    - Nếu $p$ dominates $q$, thêm $q$ vào tập hợp $S[p]$.
    - Nếu $q$ dominates $p$, tăng giá trị $n[p]$ lên $1$.
  - Nếu $n[p] = 0$, đặt rank dominates của cá thể $p$ là $1$ và thêm $p$ vào $Front[1]$.
  - Thêm các cá thể trong $S[p]$ vào $Front[i]$ và tăng rank dominates của chúng lên $i$.
- Đặt rank thứ $l = 1$.
- Nếu $Front[l]$ không rỗng:
  - Duyệt qua từng cá thể $p$ trong $Front[l]$:

    Duyệt qua từng cá thể $q$ trong $S[p]$:

    1. Giảm giá trị $n[q]$ đi $1$.

    2. Nếu $n[q] = 0$, đặt rank dominates của cá thể $q$ là $l + 1$ và thêm $q$ vào $Front[l + 1]$.
  - Tăng lên $1$.
- Kết hợp các lớp Pareto từ $Front[1]$ đến $Front[l]$ để tạo ra tập hợp Pareto cuối cùng.

### 4.6.3 Thuật toán NSGA-II cho bài toán của đồ án

Để phù hợp với bài toán đặt chức năng mạng ảo và định tuyến trong mạng ảo hoá, ta cần thay đổi để phù hợp hơn với bài toán. Các bước của thuật toán được thực hiện như sau:

- Khởi tạo: Khởi tạo $N$ cá thể, cách khởi tạo đã ở trình bày ở mục 4.3.
- Tính giá trị $fitness$ là giá trị 3 hàm mục tiêu của mỗi cá thể gồm: Độ trễ sau khi định tuyến và chi phí cài đặt server, chi phí cài đặt VNF sau khi khởi tạo một lời giải.
- Gán hạng: Sử dụng thuật toán sắp xếp nhanh không bị trội (fast non-dominated sorting) để xếp hạng cá thể theo rank (được trình bày ở phía dưới sau phần này). Các cá thể không bị trội thuộc rank thứ $1$ là tập Pareto (lời giải của bài

toán). Các cá thể bị trội bởi cá thể rank thứ $1$ được xếp vào rank thứ $2$. Tiếp tục cho đến khi xếp hạng được toàn bộ cá thể trong quần thể.

- Tính khoảng cách phân bố: Sử dụng khoảng cách Euclid để tìm và tính khoảng cách giữa hai cá thể cùng $rank$ gần cá thể được xét nhất.
- Chọn lọc: Loại bỏ $remove$ cá thể, các cá thể ở mức rank cao hơn sẽ được chọn làm thế hệ tiếp theo. Nếu cùng mức rank ta sẽ xét đến khoảng cách phân bố, cá thể nào có khoảng cách phân bố lớn hơn thì sẽ được giữ lại.
- Sinh sản: Với cá cá thể $rank$ là $1$ luôn được chọn làm cha và lai ghép với các cá thể khác không thuộc trong các cá thể bị loại bỏ, các cá thể có $rank$ khác $1$ sẽ được lai ghép với tỷ lệ $rate\_crossover$. Sử dụng phép lai 1 điểm cắt cho hai cá thể con, phép đột biến kiểu VNF cho các cá thể có $rank$ khác $1$ theo tỷ lệ $rate\_mutation$. Sinh sản sao cho đủ số lượng cá thể cần bổ sung là $remove$ cá thể, khi không đủ số lượng cá thể thoả mãn định tuyến sẽ được sinh ngẫu nhiên.
- Cập nhật quần thể: Kết hợp các cá thể được giữ lại và các cá thể con mới sinh ta được quần thể mới.
- Kiểm tra điều kiện dừng, trong đồ án này em dừng theo thời gian là quá $T$ phút sẽ dừng thuật toán và trả lại tập biên pareto là các cá thể có $rank$ là $1$. Nếu không quay lại bước Tính giá trị hàm mục tiêu.

  Phần trên đồ án đã trình bày về các giai đoạn trong tiến hoá đa mục tiêu NSGA-II. Phần tiếp theo trong mục này sẽ trình bày về thuật toán sắp xếp nhanh không bị trội trong giai đoạn gán hạng của thuật toán NSGA-II.

Trong mục này đồ án đã trình bày thuật toán NSGA-II qua các giai đoạn, thuật toán sắp xếp nhanh không bị trội và các tham số cần cài đặt: Số lượng cá thể $N$, số lượng cá thể bị loại $remove$, tỷ lệ lai ghép $rate\_crossover$, tỷ lệ đột biến $rate\_mutation$ . Mục tiếp theo đồ án trình bày về thuật toán đơn mục tiêu.

# CHƯƠNG 5. THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

Trong Chương này, đồ án trình bày về quá trình thực nghiệm và đánh giá kết quả của các thuật toán tiến hóa đa mục tiêu giải bài toán đồ án. Đồ án sẽ tiến hành các thử nghiệm trên bộ dữ liệu đa dạng để đánh giá hiệu quả của các thuật toán và xác định những giải pháp tối ưu cho bài toán. Chương này bắt đầu bằng việc giới thiệu quy trình cài đặt thực nghiệm và các tập dữ liệu sử dụng. Đồ án sẽ trình bày cụ thể về cách thiết lập thử nghiệm và các thông số liên quan, để đảm bảo tính khách quan và đáng tin cậy của kết quả đánh giá. Tiếp theo, đồ án sẽ tiến hành các thử nghiệm với các thuật toán tiến hoá, bao gồm NSGA-II, MOEA/D và MOTLBO. Đồ án sẽ lựa chọn các tham số phù hợp cho từng thuật toán và thực hiện chạy thử nghiệm trên tập dữ liệu đã được chuẩn bị. Sau khi thực hiện các thử nghiệm, đồ án sẽ đánh giá kết quả bằng cách sử dụng các tiêu chí như C-metric, IGD và HV. Những đánh giá này sẽ cho ta cái nhìn toàn diện về hiệu quả và hiệu suất của từng thuật toán và so sánh giữa chúng.

## 5.1 Cài đặt thực nghiệm

Để có thể cài đặt các thuật thuật toán trên đồ án cần các môi trường thực nghiệm. Các thuật toàn đều được cài đặt trên máy tính cá nhân có cấu hình và cài đặt các môi trường, ứng dụng sau:

**Cầu hình phần cứng:**

- CPU: 2,2 GHz Intel Core i7.
- RAM: 16GB 1600MHz DDR3.

**Cầu hình về phần mềm:**

- Hệ điều hành: macOS 12.6.3.
- IDE sử dụng: Visual Studio Code.
- Ngôn ngữ: python 3.9.5.
- Các thư viện hỗ trợ: numpy, typing, networkx, random, matplotlib, collections, pygmo.
- Lưu trữ, quản lý code: github.

## 5.2 Dữ liệu thực nghiệm

Dữ liệu gồm 12 bộ dữ liệu gồm 3 mạng (cấu trúc liên kết) mỗi mạng gồm 4 cách phân bổ (phân khu), mỗi bộ có 5 bản khác nhau. Được tổ chức theo các thư mục có tên "topology_distribution_id", trong đó topology gồm có: {cogent,

conus, nsf}; distributuib gồm {center, rural, uniform, urban}; id từ 0 đến 4. Mỗi thư mục gồm các files: "input.txt", "request10.txt", "request20.txt", "request30.txt". File "input.txt" chứa thông tin về mạng, các nút, các liên kết giữa các nút, độ trễ, chi phí kích hoạt nút server, chi phí cài đặt vnf trên các server. Mỗi file "request10.txt", "request20.txt", "request30.txt" chứa số lượng SFC tương ứng 10, 20, 30 và các thông tin về nút nguồn, nút đích, các vnf cần được chạy theo thứ tự.

### 5.2.1 File input

Trong file input.txt chứa:

- Dòng đầu tiên gồm hai số nguyên $F$, $m^{vnf}$ lần lượt là số lượng kiểu VNF và số VNF tối đa được cài đặt trên một nút Server.

  $m^{vnf}$ được lấy ngẫu nhiên trong đoạn $[Ceil(F/|V_{server}|), 2 * Ceil(F/|V_{server}|)]$, $|V_{server}|$ là số lượng nút server trong mạng, $Ceil$ là hàm làm tròn số.

- Dòng tiếp theo gồm một số nguyên $N$ chỉ số lượng nút.

- $N$ dòng tiếp theo mỗi dòng gồm các số nguyên miêu tả một nút trên đồ thị lần lượt gồm: $id, delay_{server}, cost^{server}$ (Nếu $cost^{server} \geq 0$ thì nút $id$ là server, tiếp đến là $F$ số $cost_1^{VNF}, cost_2^{VNF}, ..., cost_F^{VNF}$ tương ứng với chi phí để cài đặt VNF $f_1, f_2, ..., f_F$ ở nút server $id$, Nếu $cost^{server} = -1$ thì nút $id$ là switch, $delay_{server}$ random trong khoảng $[200, 500]$, $cost^{server}$ random trong khoảng $[5000, 10000]$, $cost_i^{VNF}$ random trong khoảng $[1000, 2000]$. $i = 1, 2, 3..F$)

- Dòng tiếp theo gồm một số nguyên $M$ chỉ số lượng cạnh.

- $M$ dòng tiếp theo mỗi dòng gồm các số nguyên miêu tả một cạnh trên đồ thị: $u, v, delay_{link}$ (Một cạnh nối giữa nút $u$ và $v$, $delay_{link}$ random trong khoảng $[200, 500]$)

Như vậy trong file input.txt đồ án đã có thông tin về mạng gồm số kiểu vnf, số vnf cài đặt tối đa trên nút server, các nút là nút server, nút switch, các liên kết giữa các nút (các cạnh). Ngoài ra ta có các thông tin về chi phí kích hoạt, cài đặt và độ trễ trên liên kết, độ trễ khi chạy vnf trên server. Trong mạng còn thiếu thông tin về tài nguyên băng thông tối đa trên các liên kết, tài nguyên bộ nhớ tối đa trên nút switch và tài nguyên CPU tối đa trên nút server. Các thông tin về tài nguyên tối đa này sẽ được tính toán thông qua các yêu cầu SFC. Phần tiếp theo đồ án trình bày về thông tin các SFC trong file request.

### 5.2.2 File request

Trong file requestX.txt chứa:

- Dòng đầu tiên gồm một số nguyên $R$ chỉ số lượng yêu cầu SFC.

- $R$ dòng tiếp theo mỗi dòng gồm các số nguyên miêu tả một yêu cầu SFC gồm: $w^{bw}, w^{mem}, w^{cpu}, u, v, k$, tiếp theo là $k$ VNF yêu cầu thực hiện theo thứ tự (mỗi yêu cầu SFC khi đi qua liên kết tiêu tốn một lượng băng thông $w^{bw}$, bộ nhớ $w^{mem}$ trên nút switch, tài nguyên xử lý $w^{cpu}$ trên nút server, yêu cầu từ nút $u$ tới nút $v$ và yêu cầu thực hiện $k$ VNF theo thứ tự).

Như vậy ta đã có các thông tin về các yêu cầu SFC về nút nguồn, đích, các vnf cần thực thiện và nhu cầu về tài nguyên khi thực hiện yêu cầu đó. Phần tiếp theo trong đồ án sẽ trình bày về tài nguyên tối đa trong mạng thông qua các yêu cầu SFC.

### 5.2.3 Các thông tin về băng thông, bộ nhớ, CPU tối đa

Tóm tắt lại, từ hai file input và request ở trên đồ án đã đề cập đến các thông tin về số kiểu vnf, số vnf tối đa đặt trên một server rồi đến số nút trong mạng mỗi nút server có các thông tin về chi phí kích hoạt, chi phí cài đặt vnf, độ trễ chạy vnf trên server đó. Bên cạnh đó còn có các thông tin về liên kết giữa các nút trong mạng và độ trễ trên liên kết đó. Ngoài ra đồ án cũng đã trình bày các thông tin về số lượng yêu cầu, mỗi yêu cầu có các thông tin về lượng băng thông tiêu tốn trên các liên kết, bộ nhớ trên nút switch, lượng tài nguyên CPU trên nút server, yêu cầu từ nút nguồn tới nút đích và yêu cầu thực hiện các VNF theo thứ tự. Còn thiếu các thông tin về giới hạn các tài nguyên gồm: lượng băng thông tối đa trên các liên kết, bộ nhớ tối đa trên các nút switch, tài nguyên CPU tối đa trên các nút server. Các thông tin về tài nguyên giới hạn này sẽ được tính thông qua các yêu cầu SFC và mạng được khởi tạo.

Đầu tiên là băng thông tối đa trên các liên kết:

$$m^{bw} = \sum_{r=1}^{R} w_r^{bw} * (k_r + 1)$$

trong đó, $R$ là số lượng các yêu cầu SFC; $w_r^{bw}$ là nhu cầu về băng thông khi yêu cầu thứ $r$ đi qua; $k_r$ là số lượng VNF cần thực hiện trong một yêu cầu SFC.

Tiếp theo là bộ nhớ tối đa trên các nút switch:

$$m^{mem} = \sum_{r=1}^{R} w_r^{mem} * (k_r + 1)$$

trong đó, $R$ là số lượng các yêu cầu SFC; $w_r^{mem}$ là nhu cầu về bộ nhớ khi yêu cầu thứ $r$ đi qua; $k_r$ là số lượng VNF cần thực hiện trong một yêu cầu SFC.

Cuối cùng là CPU tối đa trên các nút server:

$$m^{cpu} = \sum_{r=1}^{R} w_r^{cpu} * (k_r)$$

trong đó, $R$ là số lượng các yêu cầu SFC; $w_r^{mem}$ là nhu cầu về tài nguyên CPU khi yêu cầu thứ $r$ đi qua; $k_r$ là số lượng VNF cần thực hiện trong một yêu cầu SFC.

Như vậy đã có đủ các dữ liệu về bài toán để tiến hành thử nghiệm. Phần tiếp theo đồ án trình bày về các tiêu chí đánh giá của các lời giải tìm được thông qua các thuật toán tiến hoá đa mục tiêu.

## 5.3 Lựa chọn tham số

Thử nghiệm với các tham số ứng với với từng thuật toán khác nhau qua từng thế hệ để có thể lựa chọn tham số phù hợp. Kết quả đa phần được chạy trên bộ dữ liệu là "nsf_urban_0" và file là request10.txt trong 10 phút, đối với tham số $N$ của thuật toán MOEA/D thì cần thử nghiệm nhiều bộ dữ liệu hơn. Lý do chọn bộ dữ liệu "nsf_urban_0" là vì là bộ nhỏ vừa, không gian thiết kế vừa (4 server cài đặt tối đa 2 vnf trên tổng 5 kiểu vnf khoảng $(6 * 5)^4 = 30^4 = 810.000$ cách kích hoạt server và cài đặt vnf, đã bao gồm các cá thể không định tuyến được) có thể tìm ra được tập các điểm tối ưu Pareto độ chính xác tương đối cao trong một khoảng thời gian 10 phút. Kết quả được đánh giá dựa trên thế hệ hội tụ về các điểm Pareto.

### 5.3.1 Thuật toán tiến hóa dựa trên dạy-học (MOTLBO)

Thuật toán MOTLBO có các tham số đầu vào là $N$, $T$, $remove$ và $rate_crossover$. Tuy nhiên hai tham số $N$ và $remove$ đồ án để cố định là 100 và 5 phút. Thử nghiệm các tham số $remove$ và $rate\_crossover$.

Thử nghiệm về tham số $remove$ (số cá thể bị loại sau một thế hệ) lần lượt là $10, 20, 30, 40, 50$. Các thông số về $N$ cá thể trong quần thể là 100 cá thể, $T$ thời gian là 5 phút, $rate\_crossover$ là 0.8 (cá nhân tác giả nghĩ rằng việc cố định tham số lai ghép là 0.8 sẽ không ảnh hưởng đến tham số $remove$ nhiều). Dự đoán kịch bản thử nghiệm với các tham số là $50, 40$ sẽ phù hợp cho các thế hệ đầu khi mà tạo thêm các cá thể mới sẽ có nhiều tiềm năng hơn quá trình lai ghép, các tham số là $10, 20, 30$ sẽ tốt hơn cho các thế hệ sau khi mà các cá thể trong quần thể đã có các kiểu gen tốt để lai ghép với nhau. Kết quả thử nghiệm như sau:

Quan sát bảng 5.1 có thể thấy rằng khi loại bỏ càng nhiều cá thể đi thì tốc độ tính toán quá trình dạy, học càng nhanh do đó trong cùng một khoảng thời gian cho được nhiều thế hệ hơn, tuy nhiên thì sự hội tụ về các điểm Pareto lại bị chậm đi. Do đó có thể nhận xét rằng khi ta loại bỏ nhiều cá thể và thêm vào các cá thể khởi

| $remove$ | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|
| Thế hệ | 118 | 137 | 171 | 200 | 239 |
| Tổng thế hệ | 125 | 147 | 174 | 207 | 251 |
| | 94.4% | 93.2% | 98.3% | 96.6% | 95.2% |

**Bảng 5.1:** Bảng đánh giá tham số $remove$ trong thuật toán MOTLBO.

tạo ngẫu nhiên chỉ tốt cho quá trình tiến hoá ban đầu (tức là các thế hệ đầu). Nhìn vào bảng có hai lựa chọn tốt nhất là $remove$ bằng 10 và 20, tuy nhiên đồ án chọn $remove$ là 20 vì tỷ lệ 93.2% cho thấy rằng với $remove$ là 20 thì thuật toán hội tụ sớm hơn.

Tiếp theo đồ án xin trình bày thử nghiệm với tỷ lệ lai ghép $rate\_crossover$ lần lượt từ $0.5$ đến $0.9$. Thuật toán được chạy trên bộ dữ liệu "nsf_urban_0" với các tham số $N$ là, $T$ là 10 phút, $remove$ là 20. Kết quả của thử nghiệm như sau:

| $rate\_crossover$ | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|
| Thế hệ | 122 | 105 | 104 | 91 | 82 |
| Tổng thế hệ | 131 | 119 | 107 | 92 | 82 |
| | 93.1% | 88.2% | 97.1% | 98.9% | 100% |

**Bảng 5.2:** Bảng đánh giá tham số $rate\_crossover$ trong thuật toán MOTLBO.

Nhận xét từ bàng 5.2 khi tỷ lệ lai ghép càng cao thuật toán càng cần tính toán nhiều, dẫn đến tổng số thế hệ ít hơn trong cùng một khoảng thời gian. Có thể các thế hệ đầu của quần thể không có quá nhiều tính trội do đó việc lai ghép quá nhiều không thực sự đem lại hiệu quả, trong khi sự hội tụ dừng lại trước đó. Nhận thấy điều đó, nên đồ án chọn tỷ lệ lai ở mức $0.5$ hoặc $0.6$ là hợp lý, đồ án xin chọn tỷ lệ lai ghép là $0.6$

### 5.3.2 Thuật toán tiến hóa dựa trên phân giã (MOEA/D)

Với thuật toán MOEA/D đồ án thử nghiệm với số lượng cá thể $N$, $T$ và $K$ số láng giềng gần. Đối với thử nghiệm về tham số $N$ thì đồ án ban đầu để là 100, $T$ là 10 phút sau khi tách trọng số và tính lại hàm mục tiêu ta tìm được nhiều nhất là 16 điểm trong tổng 100 điểm (thử nghiệm trên 3 bộ có là "cogent_center_0", "conus_center_0", "nsf_center_0", kết quả được trình bày ở bảng 5.3).

| | "cogent_center_0" | "conus_center_0" | "nsf_center_0" |
|---|---|---|---|
| Điểm tối ưu Pareto | 4 | 14 | 16 |

**Bảng 5.3:** Bảng đánh giá tham số $N$ trong thuật toán MOEA/D.

Kiểm tra đối với thuật toán MOTLBO và thuật toán NSGA-II số lượng điểm tối

ưu Pareto đều không vượt quá 20, do đó lựa chọn tham số $N$ là 20 thì hợp lý. Đối với tham số $K$ thì được thử nghiệm trên bộ dữ liệu "nsf_urban_0" trong 10 phút, với số lượng cá thể là 20, $K$ được thử nghiệm từ 1 đến 5. Kết quả được đánh giá thông qua thế hệ mà tại đó có đủ số điểm tối ưu Pareto trong gen cuối cùng, kết quả được trình bày trong bảng 5.4.

| $K$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Thế hệ | 17446 | 17028 | 16707 | 16482 | 16360 |

**Bảng 5.4:** Bảng đánh giá tham số $K$ trong thuật toán MOEA/D.

Khi $K$ tăng lên thì sự lai ghép chắt chọn những tính trội của cá thế láng giếng nhiều thêm tuy nhiên sẽ tăng thêm chi phí tính toán. Nhận thấy từ khoảng $K = 3$ trở đi sự hội tụ giảm dần tuy nhiên thì không giảm quá nhiều. Do đó thuật toán sử dụng tham số $K = 3$.

### 5.3.3 Thuật toán di truyền sắp xếp không thống trị (NSGA-II)

Thuật toán NSGA-II có các tham số lần lượt là $N$, $T$, $remove$, $rate\_crossover$ và $rate\_mutation$. Tuy nhiên thì hai tham số $N$ và $T$ không thử nghiệm và để cố định là 100 cá thể và chạy trong 5 phút.

Thử nghiệm về tham số $remove$ (số cá thể bị loại sau một thế hệ) lần lượt là $10, 20, 30, 40, 50$. Các thông số về $N$ cá thể trong quần thể là 100 cá thể, $T$ thời gian là 10 phút, $rate\_crossover$ là 0.8, $rate\_mutation$ là 0.1 (cá nhân tác giả nghĩ rằng việc cố định tham số lai ghép là 0.8 và tham số đột biến là 0.1 sẽ không ảnh hưởng đến tham số $remove$ nhiều). Dự đoán kịch bản thử nghiệm với các tham số là $50, 40$ sẽ phù hợp cho các thế hệ đầu khi mà tạo thêm các cá thể mới sẽ có nhiều tiềm năng hơn quá trình lai ghép, các tham số là $10, 20, 30$ sẽ tốt hơn cho các thế hệ sau khi mà các cá thể trong quần thể đã có các kiểu gen tốt để lai ghép với nhau. Kết quả thử nghiệm được ghi lại trên bảng 5.5:

| $remove$ | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|
| Thế hệ | 151 | 161 | 143 | 174 | 211 |
| Tổng thế hệ | 151 | 165 | 145 | 178 | 215 |
| | 100% | 97.5% | 98.6% | 97.8% | 98.1% |

**Bảng 5.5:** Bảng đánh giá tham số $remove$ trong thuật toán NSGA-II.

Với kết quả trên 5.5 thật khó để đánh giá. Với $remove$ là 10 thì có vẻ thuật toán chưa đủ điểm tối Pareto, các kết quả khác chạy trong 10 phút cũng chỉ ở mức tương đối gần với thế hệ cuối. Có thể thời điểm ban đầu khởi tạo với $remove$, thuật toán cần nhiều thế hệ cho việc chọn ra các cá thể tiềm năng. Còn với các $remove$ là 40,

50, cũng chưa thực sự tốt, nếu loại bỏ nhiều cá thể trong quần thể thế hệ sau thì sẽ giảm đi khả năng lai ghép giữa các cá thể tốt. Do đó đồ án lựa chọn $remove$ ở mức 20, 30.

Thử nghiệm về tham số $rate\_crossover$ từ $0.5$ đến $0.9$ với các tham số $N$ là 100, $T$ là 10 phút, $remove$ là 20 và $rate\_mutation$ là 0.1. Kết quả của thử nghiệm trong bảng 5.6:

| $rate\_crossover$ | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|
| Thế hệ | 204 | 170 | 173 | 161 | 165 |
| Tổng thế hệ | 205 | 172 | 178 | 162 | 166 |
| | 99.5% | 98.8% | 97.2% | 99.4% | 99.4% |

**Bảng 5.6:** Bảng đánh giá tham số $rate\_crossover$ trong thuật toán NSGA-II.

Với kết quả từ thực nghiệm như trên bảng 5.6, đồ án lựa chọn tham số $rate\_crossover$ là 0.7. Với tham số $rate\_mutation$ đồ án chọn là 0.05, tuy không qua thử nghiệm nào nhưng việc tạo ra một cá thể đột biến trong bài toán của đồ án có tiềm năng gần như quá trình thêm cá thể khởi tạo mới. Vì vậy tham số $rate\_mutation$ đồ án chọn là 0.05, trong các thuật toán tiến hoá thông thường tỷ lệ đột biến cũng thường để là từ 0.01 đến 0.1 (1% đến 10%).

### 5.3.4 Kết luận chọn tham số, chạy thuật toán

Sau khi tiến hành thử nghiệm với các tham số, đồ án lựa chọn các tham số để tiến hành chạy các thuật toán với bộ dữ liệu đề cập ở mục 5.2. Thuật toán MOTLBO với các tham số gồm: $N$ là 100, $remove$ là 20, $rate\_crossover$ là 0.6. Thuật toán MOEA/D với các tham số gồm: $N$ là 20, $K$ là 3. Thuật toán NSGA-II được chạy với $N$ là 100, $remove$ là 20, $rate\_crossover$ là 0.7, $rate\_mutation$ là 0.05. Các thuật toán đều chạy với thời gian $T$ là 5 phút.

Ngoài ra đồ án có thêm 6 thuật toán đơn mục tiêu với hàm mục tiêu lần lượt là tổng giá trị các hàm mục tiêu theo trọng số tương ứng $[1, 0, 0]$; $[0, 1, 0]$; $[0, 0, 1]$;$[0.4, 0.3, 0.3]$; $[0.3, 0.4, 0.3]$; $[0.3, 0.3, 0.4]$. Các tham số của thuật toán gồm: $N$ là 100 cá thể, $rate\_crossover$ là 0.7, $rate\_mutation$ là 0.05 được chạy trong 5 phút. Sau đó tổng hợp 6 kết quả của thuật toán để tạo thành tập Pareto tạm gọi là kết quả của thuật toán GA-6. Phần tiếp theo đồ án trình bày các cách đánh giá lời giải của các thuật toán.

## 5.4 Tiêu chí đánh giá

Có nhiều độ đo đánh giá lời giải của thuật toán đa mục tiêu. Tuy nhiên thì đồ án xin trình bày 3 độ đo phổ biến bao gồm: C-metric or coverage of two sets (tạm gọi là "độ đo C-metric"), Inverted generational distance (tạm gọi là "độ đo IGD"), Hyperarea/hypervolume metrics (tạm gọi là "độ đo HV"). Các độ đo này đều được

tham khảo từ bài báo "Performance indicators in multiobjective optimization"[24].

### 5.4.1 Độ đo C-metric

Cho $A$ và $B$ là hai tập xấp xỉ tập Pareto. Độ đo C-metric ánh xạ có thứ tự $(A, B)$ trong khoảng $[0; 1]$ và được xác định như sau [24]:

$$C(A, B) = \frac{|\{b \in B, \text{ tồn tại } a \in A \text{ sao cho } a \preceq b\}|}{B}$$

Nếu $C(A, B) = 1$ thì tất cả các phần tử của B đều bị dominate (hoặc dominate yếu) các phần tử của $A$. Nếu $C(A, B) = 0$ thì tất cả các phần tử của $B$ đều dominate hoàn toàn các phần tử của tập hợp $A$. Độ đo C-metric $C(A, B)$ và $C(B, A)$ phải được tính toán, vì $C(A, B)$ không phải lúc nào cũng bằng $1 - C(A, B)$. Số liệu này nắm bắt tỷ lệ các điểm trong tập Pareto $A$ bị dominates bởi tập hợp Pareto $B$.

Tuy nhiên, để thuật tiện trình bày kết quả thì đồ án đã tham khảo một độ đo khác biệt đi một chút gọi là độ đo $\delta$-metric nằm trong bài báo "A decomposition-based multi-objective optimization approach forbalancing the energy consumption of wireless sensor networks"[30]. Cụ thể độ đo $\delta$-metric được tính như sau:

$$\delta(A, B) = C(A, B) - C(B, A)$$

Độ đo sẽ nằm trong khoảng từ $[-1, 1]$. Với $\delta(A, B) = 1$ tức là tập $A$ lấn át hoàn toàn tập $B$, và ngược lại với $\delta(A, B) = -1$.

### 5.4.2 Độ đo IGD

Inverted Generational Distance (IGD) là một chỉ số đánh giá các giải thuật tối ưu đa mục tiêu trong thuật toán di truyền đa mục tiêu. Nó được sử dụng để đo đạc sự tương đồng giữa tập tối ưu Pareto tìm được và tập tối ưu Pareto tham chiếu. Khi IGD nhỏ hơn, điều đó cho thấy rằng tập tối ưu Pareto tìm được gần với tập hợp tối ưu Pareto tham chiều hơn, tức là thuật toán di truyền đa mục tiêu đã đạt được kết quả tốt hơn. IGD được xác định như sau[24]:

$$IGD(S, P) = \frac{1}{|P|}(\sum_{i=1}^{|P|} d_i^p)^{\frac{1}{p}}$$

trong đó, $|S|$ là số điểm trong tập Pareto và $P$ là một biểu diễn rời rạc của biên Pareto (tập tối ưu Pareto tham chiếu). $d_i = min_{x \in S}||F(x) - F(i)||$ và $p$ thường được chọn bằng 2.

Để có thể tính toán IGD cho bài toán của đồ án sẽ cần một tập tối ưu Pareto tham

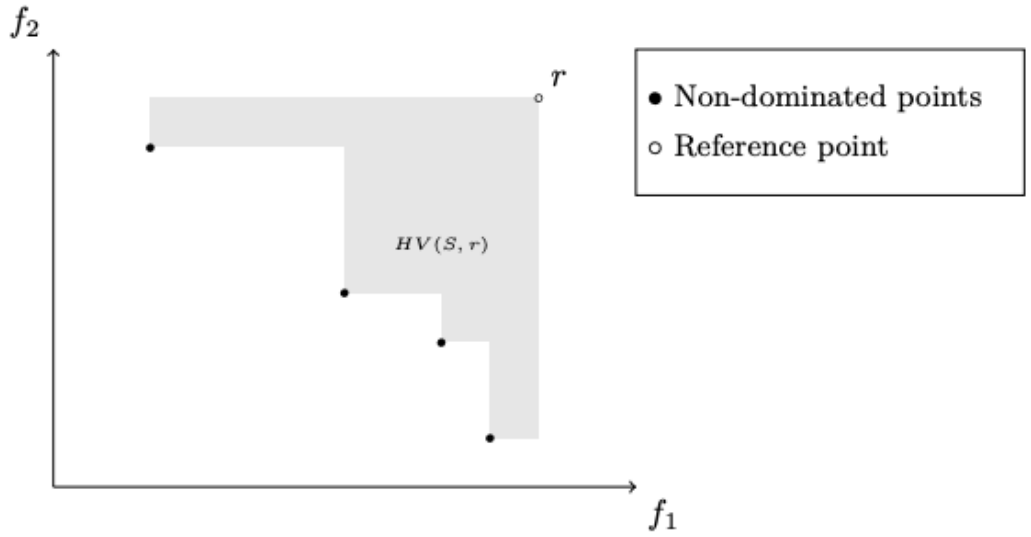
chiếu. Đồ án đã tính toán tập này bằng cách gộp các kết quả của 3 thuật toán lại và lấy ra các lời giải không bị dominates và không trùng lặp. Sau khi tính toán được tập tối ưu Pareto tham chiếu, ta sẽ tính khoảng cách từ các tập Pareto tìm được ứng với mỗi thuật toán tới tập tối ưu Pareto tham chiếu theo công thức trên.

### 5.4.3 Độ đo HV

HyperVolume (HV) là một chỉ số đo lường sự tốt của tập hợp các kết quả đã tối ưu hóa trong bài toán tối ưu đa mục tiêu. Nó cung cấp thông tin về không gian mà các kết quả đã tối ưu chiếm giữ trong không gian mục tiêu. HV đo lường khối lượng của không gian được chứa bởi các lời giải tìm được so với một lời giải tham chiếu được xác định trước. Cách tính toán HV thường được thực hiện bằng cách xác định diện tích, thể tích của một vùng trong không gian nhiều chiều mà tập hợp các kết quả đã tối ưu. Diện tích, thể tích này được tính bằng cách tích phân dọc theo các mục tiêu và theo các điểm của tập tối ưu Pareto với lời giải tham chiếu. HV càng lớn, tức là không gian mà tập hợp các kết quả đã tối ưu chiếm giữ càng lớn, và điều này cho thấy rằng tập hợp này chứa nhiều kết quả tốt và phân tán trong không gian mục tiêu. Với tập Pareto $S$ và điểm tham chiếu $r$ độ đo HV được xác định bởi [24]:

$$HV(S, r) = \lambda_m(\bigcup_{z \in S} |z; r|)$$

trong đó: $\lambda_m$ là thước đo Lebesgue $m$ chiều; $r \in R^m$, $m$ là số hàm mục tiêu của bài toán; $z \in S, z \prec r$. Một ví dụ minh hoạ được đưa ra trong hình 5.1 với không gian 2 chiều.




**Hình 5.1:** Hình minh hoạ về độ đo HV cho bài toán với hai mục tiêu $f_1$ và $f_2$ [24].

Ở bài toán này, do các hàm mục tiêu thu được đã chuẩn hóa về khoảng $(0, 1)$ do đó có thể set giá trị điểm tham chiếu của HV là $(1, 1, 1)$ và tính thể tích đến điểm đó. Như vậy đồ án đã làm rõ được 3 độ đo C-metric, IGD và HV. Chương sau sẽ trình bày kết quả theo 3 độ đo này.

## 5.5 Kết quả

Sau khi chạy các thuật toán, đồ án tiến hành lấy kết quả của gen cuối cùng sau 5 phút chạy. Đối với thuật toán MOEA/D tiến hành loại bỏ các giá trị trùng lặp. Như đã đề cập, đồ án có 12 bộ dữ liệu cho mỗi loại gồm 3 cấu trúc mạng (topology), 4 phân bố (distribution), mỗi bộ có 5 bản. Do đó sau khi tính toán được $\delta$-metric với 12 bộ dữ liệu, ta tính trung bình $\delta$-metric của 5 bản cho mỗi bộ đó, kết quả được thể hiện qua bảng 5.7:

Quan sát bảng 5.7 có thể thấy rằng thuật toán MOEAD có độ đo $delta$-metric tốt nhất, tiếp đến là thuật toán NSGA-II, thuật toán MOTLBO chưa được tốt với độ đo $delta$-metric.

Kết quả dựa trên độ đo IGD được thể hiện qua bảng 5.8, 5.9, 5.10:

Từ bảng 5.8 nhận thấy rằng với các bộ "cogent" thì thuật toán MOEAD chiếu ưu thế 15/20 bộ, nhưng sự chênh lệch về độ đo igd hai thuật toán MOTLBO và NSGA-II cũng không quá đáng kể. Bảng 5.9 cho ta thấy trong bộ "conus" thuật toán MOTLBO có số lượng 10/20 bộ có điểm cao nhất, hai thuật toán NSGA-II và MOEA/D ngang nhau. Trong bảng 5.10 bộ "nsf" thì thuật toán NSGA-II có 15/20 bộ có điểm cao nhất, thuật toán MOTLBO tuy không có số lượng điểm cao hơn

| Datasets | $\delta$(MOTLBO/MOEAD) | $\delta$(MOEAD/NSGA-II) | $\delta$(NSGA-II/MOTLBO) |
|---|---|---|---|
| cogent_center | -0.2333 | 0.2040 | 0.0704 |
| cogent_rural | -0.7008 | 0.6921 | 0.1280 |
| cogent_uniform | -0.8149 | 0.6551 | 0.3268 |
| cogent_urban | -0.6887 | 0.5842 | 0.1519 |
| conus_center | -0.1739 | 0.0701 | 0.1299 |
| conus_rural | -0.5050 | 0.4120 | 0.2183 |
| conus_uniform | -0.1598 | 0.1678 | 0.0153 |
| conus_urban | -0.4152 | 0.2673 | 0.3246 |
| nsf_center | 0.3885 | -0.4859 | 0.0735 |
| nsf_rural | -0.4059 | 0.3189 | 0.0989 |
| nsf_uniform | -0.3457 | 0.3732 | 0.0433 |
| nsf_urban | -0.1560 | 0.0973 | 0.0960 |

**Bảng 5.7:** Kết quả dựa trên độ đo $\delta$-metric.

| Datasets | MOTLBO | NSGA-II | MOEA/D |
|---|---|---|---|
| cogent_center_0 | 0.0065 | **0.0056** | 0.0084 |
| cogent_center_1 | **0.0057** | 0.0066 | 0.0099 |
| cogent_center_2 | 0.0017 | 0.0012 | **0.0006** |
| cogent_center_3 | 0.0020 | 0.0024 | **0.0016** |
| cogent_center_4 | 0.0027 | 0.0025 | **0.0014** |
| cogent_rural_0 | 0.0146 | 0.0148 | **0.0141** |
| cogent_rural_1 | 0.0086 | **0.0061** | 0.0093 |
| cogent_rural_2 | 0.0062 | 0.0052 | **0.0036** |
| cogent_rural_3 | 0.0057 | 0.0058 | **0.0028** |
| cogent_rural_4 | 0.0165 | 0.0132 | **0.0118** |
| cogent_uniform_0 | 0.0093 | 0.0087 | **0.0051** |
| cogent_uniform_1 | 0.0132 | 0.0173 | **0.0070** |
| cogent_uniform_2 | 0.0156 | 0.0183 | **0.0027** |
| cogent_uniform_3 | **0.0048** | 0.0064 | 0.0050 |
| cogent_uniform_4 | 0.0154 | 0.0163 | **0.0070** |
| cogent_urban_0 | 0.0118 | 0.0063 | **0.0037** |
| cogent_urban_1 | 0.0083 | 0.0068 | **0.0066** |
| cogent_urban_2 | 0.0140 | 0.0135 | **0.0037** |
| cogent_urban_3 | 0.0144 | **0.0081** | 0.0179 |
| cogent_urban_4 | 0.0065 | 0.0058 | **0.0048** |

**Bảng 5.8:** Kết quả dự theo độ đo IGD cho bộ "cogent".

nhưng điểm số không chênh lệch quá nhiều, MOEA/D tỏ ra yếu thế hơn trong bộ "nsf".

Kết quả dựa trên độ đo HV được thể hiện qua bảng 5.11:

Quan sát bảng 5.11 thì thuật toán MOEA/D và GA-6 luôn cho phần thể tích là lớn nhất, chỉ có bộ "nsf_urban" request 20, 30 kém hơn một chút so với NSGA-II

| Datasets | MOTLBO | NSGA-II | MOEA/D |
|---|---|---|---|
| conus_center_0 | **0.0149** | 0.0135 | 0.0303 |
| conus_center_1 | **0.0083** | 0.0086 | 0.0226 |
| conus_center_2 | 0.0095 | **0.0059** | 0.0132 |
| conus_center_3 | 0.0046 | 0.0052 | **0.0045** |
| conus_center_4 | 0.0066 | **0.0042** | 0.0046 |
| conus_rural_0 | **0.0028** | 0.0048 | 0.0073 |
| conus_rural_1 | 0.0084 | 0.0080 | **0.0057** |
| conus_rural_2 | **0.0035** | 0.0082 | 0.0132 |
| conus_rural_3 | 0.0102 | 0.0119 | **0.0030** |
| conus_rural_4 | 0.0037 | **0.0029** | 0.0036 |
| conus_uniform_0 | **0.0084** | 0.0119 | 0.0119 |
| conus_uniform_1 | 0.0069 | **0.0054** | 0.0092 |
| conus_uniform_2 | **0.0024** | 0.0052 | 0.0037 |
| conus_uniform_3 | 0.0080 | **0.0033** | 0.0099 |
| conus_uniform_4 | **0.0068** | 0.0071 | 0.0090 |
| conus_urban_0 | **0.0053** | 0.0101 | 0.0120 |
| conus_urban_1 | **0.0042** | 0.0069 | 0.0126 |
| conus_urban_2 | 0.0149 | 0.0116 | **0.0106** |
| conus_urban_3 | **0.0034** | 0.0056 | 0.0035 |
| conus_urban_4 | 0.0138 | **0.0109** | 0.0213 |

**Bảng 5.9:** Kết quả dự theo độ đo IGD cho bộ "conus".

tuy nhiên không đáng kể. Phần đa các bộ dữ liệu nsf đều ngang nhau nhưng GA-6 có phần hơi kém hơn so với các giải thuật đa mục tiêu. Tuy nhiên thì ở bộ dữ liệu lớn "cogent" và "conus" có vẻ như việc tìm ra được nhiều điểm tối ưu Pareto hơn đã khiến phần diện tích HV tìm được của GA-6 lớn hơn các thuật toán đa mục tiêu. Mặc dù MOEA/D cho được phần diện tích lớn hơn nhưng 2 thuật toán MOTLBO và NSGA-II cũng không thua thiệt nhiều. Thuật toán GA-6 so với MOEA/D khá ngang nhau, tuy có phần thua thiệt một chút ở các bộ dữ liệu "cogent" và "conus". Tuy nhiên thuật toán GA-6 yêu cầu 6 lần chạy liên tiếp để thu được tập Pareto (gấp 6 lần thời gian chạy thuật toán MOEA/D) gần tương đương với một lần chạy MOEA/D, nghĩa là MOEA/D vẫn có lợi thế về tính toán hơn. Kiểm tra cụ thể kết quả với bộ "nsf" thì đồ án nhận thấy rằng điểm tối ưu tìm được của thuật toán GA-6 bị rơi vào điểm tối ưu địa phương (so sánh với các điểm tối ưu pareto của thuật toán đa mục tiêu), có thể do việc cải thiện cá thể của của các thuật toán đa mục tiêu nhận được các tính trội từ một cá thể có trọng số khác để thoát khỏi điểm tối ưu địa phương. Do đó nếu như tăng thời gian tính toán các thuật toán đa mục tiêu lên đồ án nhận định có thể hiệu quả của các thuật toán đa mục tiêu có thể sẽ trội so với GA-6. Trong bảng ta thấy trong bộ "nsf_center" request 20, 30 có thể tích là 0, sau khi kiểm tra lại thì nguyên nhân do mạng đó có hai server và phải kích hoạt cả

| Datasets | MOTLBO | NSGA-II | MOEA/D |
|---|---|---|---|
| nsf_center_0 | 0.0071 | **0.0000** | 0.0177 |
| nsf_center_1 | 0.0018 | **0.0000** | 0.0133 |
| nsf_center_2 | **0.0000** | **0.0000** | 0.0150 |
| nsf_center_3 | 0.0010 | **0.0001** | 0.0773 |
| nsf_center_4 | **0.0000** | **0.0000** | 0.0078 |
| nsf_rural_0 | **0.0032** | 0.0049 | 0.0146 |
| nsf_rural_1 | 0.0073 | **0.0071** | 0.0303 |
| nsf_rural_2 | 0.0012 | 0.0012 | **0.0000** |
| nsf_rural_3 | 0.0110 | **0.0089** | 0.0127 |
| nsf_rural_4 | 0.0095 | **0.0033** | 0.0270 |
| nsf_uniform_0 | 0.0360 | **0.0173** | 0.0778 |
| nsf_uniform_1 | **0.0064** | 0.0122 | 0.0248 |
| nsf_uniform_2 | 0.0137 | **0.0056** | 0.0226 |
| nsf_uniform_3 | 0.0083 | **0.0036** | 0.0210 |
| nsf_uniform_4 | **0.0035** | 0.0041 | 0.0185 |
| nsf_urban_0 | **0.0014** | 0.0015 | 0.0063 |
| nsf_urban_1 | 0.0061 | **0.0039** | 0.0412 |
| nsf_urban_2 | 0.0126 | **0.0053** | 0.0343 |
| nsf_urban_3 | 0.0075 | **0.0074** | 0.0324 |
| nsf_urban_4 | 0.0126 | **0.0087** | 0.0205 |

**Bảng 5.10:** Kết quả dự theo độ đo IGD cho bộ "nsf".

hai, do đó hàm mục tiêu về chi phí kích hoạt luôn là lớn nhất (bằng 1) vì thế phần thể tích bằng 0.

| Datasets | MOTLBO | NSGA-II | MOEA/D | GA-6 |
|---|---|---|---|---|
| cogent_center_request10: | 0.2066 | 0.2136 | 0.2168 | **0.2322** |
| cogent_center_request20: | 0.1553 | 0.1525 | 0.1612 | **0.1897** |
| cogent_center_request30: | 0.1353 | 0.1426 | **0.1503** | 0.1485 |
| cogent_rural_request10 | 0.5124 | 0.5209 | **0.6036** | 0.5849 |
| cogent_rural_request20 | 0.5087 | 0.5109 | 0.5585 | **0.5840** |
| cogent_rural_request30 | 0.4865 | 0.5227 | 0.5444 | **0.5712** |
| cogent_uniform_request10 | 0.5250 | 0.5046 | 0.5881 | **0.6195** |
| cogent_uniform_request20 | 0.4912 | 0.4915 | 0.5789 | **0.6157** |
| cogent_uniform_request30 | 0.4925 | 0.4907 | 0.5357 | **0.5934** |
| cogent_urban_request10 | 0.5030 | 0.5342 | 0.5516 | **0.5966** |
| cogent_urban_request20 | 0.4967 | 0.5181 | 0.5463 | **0.5903** |
| cogent_urban_request30 | 0.4836 | 0.4966 | 0.5280 | **0.5632** |
| conus_center_request10: | 0.3216 | 0.3293 | 0.3896 | **0.3951** |
| conus_center_request20: | 0.3117 | 0.3462 | **0.3507** | 0.3505 |
| conus_center_request30: | 0.3212 | 0.3250 | **0.3610** | 0.3504 |
| conus_rural_request10 | 0.5198 | 0.5025 | **0.5539** | 0.5358 |
| conus_rural_request20 | 0.5002 | 0.5079 | 0.5202 | **0.5255** |
| conus_rural_request30 | 0.4786 | 0.4909 | 0.5261 | **0.5279** |
| conus_uniform_request10 | 0.5131 | 0.5138 | 0.5385 | **0.5650** |
| conus_uniform_request20 | 0.4822 | 0.4800 | 0.4981 | **0.5122** |
| conus_uniform_request30 | 0.4937 | 0.4800 | 0.5118 | **0.5187** |
| conu_urban_request10 | 0.5084 | 0.5233 | **0.5590** | 0.5444 |
| conus_urban_request20 | 0.4631 | 0.4691 | **0.5201** | 0.5180 |
| conus_urban_request30 | 0.4622 | 0.4886 | **0.5213** | 0.5138 |
| nsf_center_request10: | **0.0625** | **0.0625** | **0.0625** | 0.0563 |
| nsf_center_request20: | **0.0000** | **0.0000** | **0.0000** | 0.0000 |
| nsf_center_request30: | **0.0000** | **0.0000** | **0.0000** | 0.0000 |
| nsf_rural_request10 | 0.2572 | 0.2589 | **0.2594** | 0.2526 |
| nsf_rural_request20 | 0.2562 | 0.2583 | **0.2592** | 0.2518 |
| nsf_rural_request30 | 0.2544 | 0.2549 | **0.2569** | 0.2478 |
| nsf_uniform_request10 | 0.3711 | 0.3719 | **0.3761** | 0.3652 |
| nsf_uniform_request20 | 0.3745 | 0.3719 | **0.3762** | 0.3566 |
| nsf_uniform_request30 | 0.3710 | 0.3717 | **0.3737** | 0.3576 |
| nsf_urban_request10 | 0.3412 | 0.3422 | **0.3429** | 0.3377 |
| nsf_urban_request20 | 0.3369 | **0.3378** | 0.3370 | 0.3278 |
| nsf_urban_request30 | 0.3333 | **0.3348** | 0.3340 | 0.3256 |

**Bảng 5.11:** Kết quả dựa trên độ đo HV.

# CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 6.1 Kết luận

Đồ án đã trình bày về cách giải quyết bài toán đặt chức năng mạng ảo và định tuyến trong mạng ảo hoá bằng các thuật toán tiến hoá đa mục tiêu. Mỗi thuật toán cho lời giải và điểm mạnh khác nhau, điểm chung là đều tìm được ra các giải pháp tốt cho bài toán đặt chức năng mạng ảo và định tuyến trong mạng ảo hoá ứng với mỗi bộ dữ liệu.

Qua đồ án em học được:(i) Tìm hiểu về mạng ảo hoá, các kiến thức về hạ tầng mạng ảo, (ii) Các thuật toán tiến hoá giải bài toán đa mục tiêu. Quá trình làm đồ án giúp em nâng cao các kỹ năng về: phân tích và giải quyết bài toán, kỹ năng viết tài liệu nghiên cứu, quy trình thử nghiệm so sánh, đánh giá.

## 6.2 Hướng phát triển trong tương lai

Đồ án còn một số giải pháp khác chưa thử nghiệm như phương pháp về học máy giải bài toán đa mục tiêu, hướng giải quyết bài toán theo cách vừa đặt vừa định tuyến. Bên cạnh đó thì có thể cải thiện thuật toán ở quá trình khởi tạo quần thể để các quần thể được tạo nhiều tiềm năng hơn. Ngoài ra còn có thể phát triển thuật toán theo hướng tối ưu 2 mục tiêu khi gộp hai chi phí kích hoạt và chi phí cài đặt VNF làm một.

# TÀI LIỆU THAM KHẢO


[1] X. W. S. L. P. D. K. L. H. Y. H. Xing X. Zhou, *An integer encoding grey wolf optimizer for virtual network function placement*, 2019. **url**: https://www.sciencedirect.com/science/article/abs/pii/S1568494619300018.

[2] A. S. L. Z. B. B. F. L. R. Solozabal J. Ceberio, *Virtual network function placement optimization with deep reinforcement learning*, 2019. **url**: https://ieeexplore.ieee.org/document/8945291.

[3] P. W. A. R. Margaret Chiosi Don Clarke, *Network functions virtualisation (nfv): An introduction, benefits, enablers, challenges and call for action, in: Sdn and openflow world congress*, 2012. **url**: https://portal.etsi.org/nfv/nfv_white_paper.pdf.

[4] S. S. A. Farshin, *A modified knowledge-based ant colony algorithm for virtual machine placement and simultaneous routing of nfv in distributed cloud architecture*, 2019. **url**: https://link.springer.com/article/10.1007/s11227-019-02804-x.

[5] A. K. E. Z. A. Marotta F. D'Andreagiovanni, *On the energy cost of robustness for green virtual network function placement in 5g virtualized infrastructures*, 2017. **url**: https://www.sciencedirect.com/science/article/abs/pii/S1389128617301767.

[6] O. G. M. Shifrin E. Biton, *Optimal control of vnf deployment and scheduling*, 2016. **url**: https://ieeexplore.ieee.org/document/7806110.

[7] A. J. M. Shokouhifar, *Optimized sugeno fuzzy clustering algorithm for wireless sensor networks*, 2017. **url**: https://www.sciencedirect.com/science/article/abs/pii/S0952197617300076.

[8] W. L. Z. Z. Q. Sun P. Lu, *Forecast-assisted nfv service chain deployment based on affiliation-aware vnf placement*, 2016. **url**: https://ieeexplore.ieee.org/document/7841846.

[9] D. H. T. R. M. I. Roberto Riggio Abbas Bradai **and** T. Ahmed, *Scheduling wireless virtual networks functions*, 2016. **url**: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7445873.

[10] A. E. R. J. L. G. H. C. D. Bhamare M. Samaka, *Optimal virtual network function placement in multi-cloud service function chaining architecture*, 2017. **url**: https://www.sciencedirect.com/science/article/abs/pii/S0140366417301901.

[11] K. X. D. Li P. Hong, *Virtual network function placement considering resource optimization and sfc requests in cloud datacenter*, 2018. **url**: https://ieeexplore.ieee.org/document/8281644.

[12] G. G. X. D. L. Y. Y. Sang B. Ji, *Provably efficient algorithms for joint placement and allocation of virtual network functions, in: Ieee infocom 2017-ieee conference on computer communications*, 2017. **url**: https://ieeexplore.ieee.org/document/8057036.

[13] F. d. A. K. A. Marotta E. Zola, *A fast robust optimization-based heuristic for the deployment of green virtual network functions*, 2017. **url**: https://www.sciencedirect.com/science/article/abs/pii/S1084804517302461.

[14] K. S. M. K. L. Qu C. Assi, *A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks*, 2017. **url**: https://ieeexplore.ieee.org/document/7967831.

[15] J. L. X. J. Z. Zhu H. Lu, *Service function chain mapping with resource fragmentation avoidance*, 2017. **url**: https://ieeexplore.ieee.org/document/8254441.

[16] Y. K. D. K. T. M. M. Otokura K. Leibnitz, *Application of evolutionary mechanism to dynamic virtual network function placement*, 2017. **url**: https://www.researchgate.net/publication/312068490_Application_of_Evolutionary_Mechanism_to_Dynamic_Virtual_Network_Function_Placement.

[17] Y. K. S. K. K. L. S. Kim S. Park, *Vnf-eq: Dynamic placement of virtual network functions for energy efficiency and qos guarantee in nfv*, 2017. **url**: https://www.researchgate.net/publication/318186151_VNF-EQ_dynamic_placement_of_virtual_network_functions_for_energy_efficiency_and_QoS_guarantee_in_NFV.

[18] S. M. S. Jamali, *Improving grouping genetic algorithm for virtual machine placement in cloud data centers*, 2014. **url**: https://ieeexplore.ieee.org/document/6993461.

[19] N. d. F. H.D. Chantre, *Redundant placement of virtualized network functions for lte evolved multimedia broadcast multicast services*, 2017. **url**: https://ieeexplore.ieee.org/document/7996870.

[20] W. A. X. C. J. S. Y. H. J. Cao Y. Zhang, *Vnf-fg design and vnf placement for 5g mobile networks*, 2017. **url**: https://link.springer.com/article/10.1007/s11432-016-9031-x.

[21] M. Shokouhifa, *Fh-aco: Fuzzy heuristic-based ant colony optimization for joint virtual network function placement and routing*, 2021. **url**: https://www.sciencedirect.com/science/article/abs/pii/S1568494621003240?via%3Dihub.

[22] P. B. E. Seyed Reza Zahedi Shahram Jamali, *Emcfis: Evolutionary multi-criteria fuzzy inference system for virtual network function placement and routing*, 2022. **url**: https://www.sciencedirect.com/science/article/abs/pii/S1568494622000102.

[23] X.-S. Yang, "Engineering optimization: An introduction with metaheuristic applications," 2010. **url**: https://www.wiley.com/en-gb/Engineering+Optimization:+An+Introduction+with+Metaheuristic+Applications-p-9780470582466.

[24] D. C. S. L. D. L. S. Charles Audet Jean Bigeon, "Performance indicators in multiobjective optimization," 2021. **url**: https://www.sciencedirect.com/science/article/abs/pii/S0377221720309620.

[25] Darwin, *Học thuyết darwin*, 2021. **url**: https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_thuy%E1%BA%BFt_Darwin.

[26] B. khoa học và phát triển Việt Nam, *Tại sao hươu cao cổ... có cổ dài?* 2021. **url**: https://khoahocphattrien.vn/Giai%20ma/tai-sao-huou-cao-co-co-co-dai/20190325025935 40p879c938.htm.

[27] X. H. D. C. B. W. Feng Zou Lei Wang, *Multi-objective optimization using teaching-learning-based optimization algorithm*, 2013. **url**: https://www.sciencedirect.com/science/article/abs/pii/S0952197612003016.

[28] Q. Z. H. Li, *Moea/d: A multiobjective evolutionary algorithm based on decomposition*, 2007. **url**: https://ieeexplore.ieee.org/document/4358754.

[29] D. D. v. H. J. Kalyanmoy Deb Anindya Ghosh, *A fast and elitist multiobjective genetic algorithm: Nsga-ii*, 2002. **url**: https://ieeexplore.ieee.org/document/996017.

[30] H. T. T. B. L. T. V. Nguyen Thi Tam Tran Huy Hung, *A decomposition-based multi-objective optimization approach for balancing the energy consumption of wireless sensor networks*, 2021. **url**: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7445873.